HYPERBOLIC EMBEDDINGS FOR ORDER-AWARE CLASSIFICATION OF AUDIO EFFECT CHAINS

Aogu Wada[1,2] , *Tomohiko Nakamura*[2] and *Hiroshi Saruwatari*[1]

[1]Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan
[2]National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, Japan
wada-aogu@g.ecc.u-tokyo.ac.jp

ABSTRACT

Audio effects (AFXs) are essential tools in music production, frequently applied in chains to shape timbre and dynamics. The order of AFXs in a chain plays a crucial role in determining the final sound, particularly when non-linear (e.g., distortion) or time-variant (e.g., chorus) processors are involved. Despite its importance, most AFX-related studies have primarily focused on estimating effect types and their parameters from a wet signal. To address this gap, we formulate AFX chain recognition as the task of jointly estimating AFX types and their order from a wet signal. We propose a neural-network-based method that embeds wet signals into a hyperbolic space and classifies their AFX chains. Hyperbolic space can represent tree-structured data more efficiently than Euclidean space due to its exponential expansion property. Since AFX chains can be represented as trees, with AFXs as nodes and edges encoding effect order, hyperbolic space is well-suited for modeling the exponentially growing and non-commutative nature of ordered AFX combinations, where changes in effect order can result in different final sounds. Experiments using guitar sounds demonstrate that, with an appropriate curvature, the proposed method outperforms its Euclidean counterpart. Further analysis based on AFX type and chain length highlights the effectiveness of the proposed method in capturing AFX order.

1. INTRODUCTION

Audio effects (AFXs) are essential tools in modern music composition, live performance, and studio production [1]. Each type of AFX (e.g. delay, chorus, distortion) has its own unique characteristics [2], and sound engineers leverage these characteristics to obtain the desired sound. In practice, musicians and sound engineers apply multiple AFXs in sequence to a given audio signal to achieve their intended result [3]. The sequence of AFXs is referred to as an *AFX chain*. The resulting sound is highly dependent on the order of AFXs in the chain, particularly when it includes non-linear or time-variant processors such as distortion or chorus. Therefore, considering the order of AFXs is essential when addressing AFX-related tasks.

Most AFX studies address the estimation of the types and parameters of AFXs from audio signals processed by AFX chains, typically under simplified assumptions about the structure of the chain. In [4], this estimation was performed under the assumption that both the number and the order of AFXs in the chain are fixed. In [5], the number of AFXs was fixed, but the order was disregarded. A few studies addressed a broader problem [6, 7]: recovering the original (dry) signal from a signal processed by an unknown AFX chain. One approach first detects which AFXs are present and then removes their effects in an order-agnostic manner [6], while another iteratively estimates and removes the most recently applied effect [7]. A method proposed in [8] uses a graph neural network to handle more complex, graph-like structure of AFX chains.

Despite progress in AFX studies, the order of AFXs within a chain has received less attention. Many previous studies consider AFX order only partially or indirectly—for example, by predicting only the types of AFXs used in the chain [4,5], or by iteratively estimating the last-applied AFX from the output signal [6, 7]. However, these approaches have limitations. Since they do no explicitly handle the order of AFXs, they may fail to capture differences in sound that arise from different effect chains. In addition, some methods rely on iterative inference over effect permutations, the number of which increases exponentially with the length of the AFX chain. This motivates us to address the problem of jointly estimating AFX types and their order, which we refer to as the AFX chain classification problem.

For the AFX chain classification problem, we focus on hyperbolic space, a non-Euclidean space characterized by constant negative curvature. In this space, the distance from the origin increases exponentially, in contrast to the linear growth in Euclidean space. This geometric property aligns naturally with the structure of hierarchical data, which can be represented as trees where the number of nodes increases exponentially with depth. As a result, hyperbolic space enables more efficient embedding of tree-like structures than Euclidean space [9]. Leveraging this property, hyperbolic space has shown promising results in various audio signal processing tasks, including musical instrument sound synthesis [10], audio source separation [11, 12], and anomalous sound detection [13].

Building on this insight, we propose a neural-network-based method that jointly estimates AFX types and their order by embedding audio signals into hyperbolic space. AFX chains can be represented as trees, with AFXs as nodes and edges encoding the effect order. Consequently, hyperbolic space is better suited for modeling the exponentially growing and non-commutative structure of AFX combinations than Euclidean space. To this end, we design a neural network that learns hyperbolic embeddings of input signals and performs AFX chain classification using multinomial logistic regression (MLR) in hyperbolic space. This architecture enables the model to capture the structural properties of effect chains in a geometrically consistent manner. To the best of our knowledge, this paper is the first to incorporate hyperbolic embeddings into AFX chain classification.

The remainder of this paper is organized as follows: Sec-

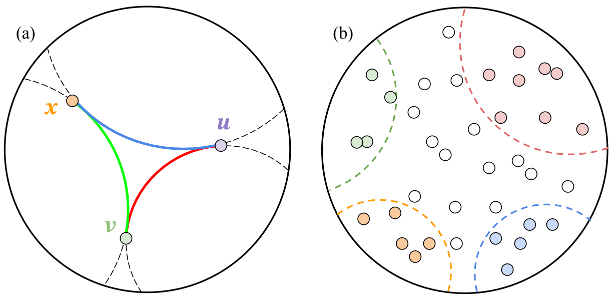


Figure 1: Schematic illustration of Poincaré ball model. (a) Geodesics (colored curves) between points in two-dimensional Poincaré ball, representing shortest paths in hyperbolic geometry. (b) Multinomial logistic regression in Poincaré ball, with geodesics as decision boundaries partitioning space into classification regions.

tion 2 reviews the mathematics of hyperbolic space and MLR in this space. Section 3 introduces the proposed network architecture and method, explaining how hyperbolic space is incorporated into the network. Section 4 presents experiments on AFX chain classification, including comparisons between the proposed method and its Euclidean counterpart. Finally, Section 5 concludes the paper.

2. HYPERBOLIC SPACE

In this section, we provide the mathematical background necessary for extending MLR to hyperbolic space. We begin by introducing the basic concepts of Riemannian geometry, focusing on the Poincaré ball, one realization of hyperbolic space. We then describe key mathematical tools used for classification in this space and finally review an extension of MLR to hyperbolic space, following [11].

2.1. Riemannian Manifold

A Riemannian manifold is defined as a pair $(\mathcal{M}, g)$, where $\mathcal{M}$ is a differentiable manifold and g is a Riemannian metric. Informally, a manifold is a space that locally resembles Euclidean space, allowing us to define smooth operations such as differentiation. At each point $\boldsymbol{x} \in \mathcal{M}$, there exists a local linear approximation of the space called the tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$. The Riemannian metric $g = (g_{\boldsymbol{x}})_{\boldsymbol{x}\in\mathcal{M}}$ assigns an inner product to each tangent space:

$$\forall \boldsymbol{u}, \boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathcal{M}, \quad \langle \boldsymbol{u}, \boldsymbol{v} \rangle_{\boldsymbol{x}} = \boldsymbol{u}^{\mathsf{T}} g_x \boldsymbol{v}. \tag{1}$$

This allows us to measure geometric quantities such as angles and distances in a smoothly varying manner across the manifold.

By integrating local information defined by $g_{\boldsymbol{x}}$, we can compute global quantities such as the length of curves on $\mathcal{M}$. The shortest path between two points, with respect to g, is called a geodesic, which generalizes the concept of a straight line in Euclidean space. The exponential map, denoted by $\exp_{\boldsymbol{x}}$, maps a tangent vector at $\boldsymbol{x}$ to a point on the manifold along the geodesic starting at $\boldsymbol{x}$. Its inverse, the logarithmic map $\log_{\boldsymbol{x}}$, projects a point on the manifold back to the tangent space $\mathcal{T}_{\boldsymbol{x}}\mathcal{M}$.

2.2. Poincaré Ball

A hyperbolic space is a Riemannian manifold with a constant negative curvature $-c$ and there are several equivalent representations of hyperbolic space. In this paper, we adopt the n-dimensional Poincaré ball model because it provides closed-form expressions for the metric and geodesic operations, making it well-suited for integration with neural networks.

This model is defined as a Riemannian manifold $(\mathbb{B}_c^n, g_c)$. The manifold $\mathbb{B}_c^n$ is the open n-dimensional ball of radius $1/\sqrt{c}$:

$$\mathbb{B}_c^n = \{\boldsymbol{x} \in \mathbb{R}^n \mid c\,||\boldsymbol{x}||^2 < 1\}. \tag{2}$$

The Riemannian metric $g_c(\boldsymbol{x})$ is given by

$$g_c(\boldsymbol{x}) = (\lambda_{\boldsymbol{x}}^c)^2 g^{\mathrm{E}}, \tag{3}$$

where g^{E} is the standard Euclidean metric, i.e., the n-dimensional identity matrix, and $\lambda_{\boldsymbol{x}}^c = 2/(1 - c\|\boldsymbol{x}\|^2)$ is known as the conformal factor. This factor adjusts the local scale of the Euclidean metric to account for the negative curvature of the space. As a result, the volume of this space increases exponentially with distance from the origin, in contrast to the polynomial growth observed in Euclidean geometry.

Figure 1(a) illustrates points in the two-dimensional Poincaré ball and the geodesics connecting them. Using $\lambda_{\boldsymbol{x}}^c$, the inner product and norm at a point $\boldsymbol{x} \in \mathbb{B}_c^n$ are defined as

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle_{\boldsymbol{x}}^c = (\lambda_{\boldsymbol{x}}^c)^2 \langle \boldsymbol{u}, \boldsymbol{v} \rangle, \quad \|\boldsymbol{v}\|_{\boldsymbol{x}}^c = \lambda_{\boldsymbol{x}}^c \|\boldsymbol{v}\|, \tag{4}$$

for $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{T}_{\boldsymbol{x}}\mathbb{B}_c^n$. Here, $\langle \cdot, \cdot \rangle$ denotes the standard Euclidean inner product. The geodesic distance between two points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{B}_c^n$ is given by

$$d_c(\boldsymbol{x}, \boldsymbol{y}) = \frac{2}{\sqrt{c}} \tanh^{-1}\left(\sqrt{c}\,\| -\boldsymbol{x} \oplus_c \boldsymbol{y}\|\right), \tag{5}$$

where $\oplus_c$ denotes the Möbius addition, a generalization of vector addition that preserves the geometry of hyperbolic space. This operation is defined as

$$\boldsymbol{x} \oplus_c \boldsymbol{y} = \frac{(1 + 2c\langle \boldsymbol{x}, \boldsymbol{y} \rangle + c\|\boldsymbol{y}\|^2)\boldsymbol{x} + (1 - c\|\boldsymbol{x}\|^2)\boldsymbol{y}}{1 + 2c\langle \boldsymbol{x}, \boldsymbol{y} \rangle + c^2\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2}. \tag{6}$$

Unlike standard vector addition in Euclidean space, Möbius addition is non-commutative: $\boldsymbol{x} \oplus_c \boldsymbol{y} \neq \boldsymbol{y} \oplus_c \boldsymbol{x}$ in general. This highlights how hyperbolic geometry preserves directional relationships that depend on operation order.

Although exponential and logarithmic maps can be defined at any point in $\mathbb{B}_c^n$, we restrict our attention to the origin $\mathbf{0}$ to simplify computation and retain closed-form expressions. These maps are given by

$$\exp_{\mathbf{0}}^c(\boldsymbol{v}) = \frac{\tanh(\sqrt{c}\,\|\boldsymbol{v}\|)}{\sqrt{c}\,\|\boldsymbol{v}\|}\boldsymbol{v}, \quad \log_{\mathbf{0}}^c(\boldsymbol{y}) = \frac{\tanh^{-1}(\sqrt{c}\,\|\boldsymbol{y}\|)}{\sqrt{c}\,\|\boldsymbol{y}\|}\boldsymbol{y}, \tag{7}$$

for $\boldsymbol{v} \in \mathbb{R}^n \setminus \{\mathbf{0}\}$ and $\boldsymbol{y} \in \mathbb{B}_c^n \setminus \{\mathbf{0}\}$.

2.3. Hyperbolic Multinomial Logistic Regression

In Euclidean space, MLR computes class logits based on the distance between an input embedding $\boldsymbol{z} \in \mathbb{R}^n$ and each of the K class hyperplanes. Let $k = 1, \ldots, K$ be the class index. The hyperplane corresponding to class k, denoted by $H_{\boldsymbol{a}_k, \boldsymbol{p}_k}$, is defined by its normal vector $\boldsymbol{a}_k \in \mathbb{R}^n$ and a point $\boldsymbol{p}_k \in \mathbb{R}^n$ lying on the hyperplane. The probability of class k is given by

$$p(\kappa = k \mid \boldsymbol{z}) \propto \exp\left(\operatorname{sign}(\langle -\boldsymbol{p}_k + \boldsymbol{z}, \boldsymbol{a}_k\rangle)\|\boldsymbol{a}_k\|\, d(\boldsymbol{z}, H_{\boldsymbol{a}_k,\boldsymbol{p}_k})\right), \tag{8}$$

where κ is a random variable denoting the class label. The function $d(\boldsymbol{z}, H_{\boldsymbol{a}_k,\boldsymbol{p}_k})$ denotes the Euclidean distance from the point $\boldsymbol{z}$ to the hyperplane $H_{\boldsymbol{a}_k,\boldsymbol{p}_k}$.

Using the mathematical tools introduced in Section 2.2, the Euclidean MLR can be extended to the Poincaré ball model [14]. We first map Euclidean embeddings $\boldsymbol{z} \in \mathbb{R}^n$ to hyperbolic embeddings $\boldsymbol{z}^{\mathrm{h}} \in \mathbb{B}_c^n$ using the exponential map defined in Eq. (7):

$$\boldsymbol{z}^{\mathrm{h}} = \exp_{\boldsymbol{0}}^{c}(\boldsymbol{z}). \tag{9}$$

We then replace the standard addition with the Möbius addition $\oplus_c$, and compute inner products and norms using the geometry of $\mathbb{B}_c^n$ to obtain the hyperbolic counterpart of Eq. (8). Specifically, we use the hyperbolic inner product $\langle\cdot,\cdot\rangle_{\boldsymbol{x}}^{c}$ and norm $\|\cdot\|_{\boldsymbol{x}}^{c}$ defined in Section 2.2. The hyperplane $H^{\mathrm{h}}_{\boldsymbol{a}_k^{\mathrm{h}},\boldsymbol{p}_k^{\mathrm{h}}}$ in $\mathbb{B}_c^{n,\mathrm{h}}$ is defined as the set of points equidistant to $\boldsymbol{p}_k^{\mathrm{h}}$ along the geodesic direction determined by the tangent vector $\boldsymbol{a}_k^{\mathrm{h}}$:

$$H^{\mathrm{h}}_{\boldsymbol{a}_k^{\mathrm{h}},\boldsymbol{p}_k^{\mathrm{h}}} = \left\{\boldsymbol{x} \in \mathbb{B}_c^n \mid \langle -\boldsymbol{p}_k^{\mathrm{h}} \oplus_c \boldsymbol{x}, \boldsymbol{a}_k^{\mathrm{h}}\rangle = 0\right\}. \tag{10}$$

Using these notations, the hyperbolic extension of Eq. (8) is given as

$$p(\kappa = k \mid \boldsymbol{z}^{\mathrm{h}}) \propto \exp\left(\frac{\lambda_{\boldsymbol{p}_k^{\mathrm{h}}}\|\boldsymbol{a}_k^{\mathrm{h}}\|}{\sqrt{c}}\sinh^{-1}(r_k)\right), \tag{11}$$

$$r_k := \frac{2\sqrt{c}\langle \boldsymbol{p}_k^{\mathrm{h}} \oplus_c \boldsymbol{z}^{\mathrm{h}}, \boldsymbol{a}_k^{\mathrm{h}}\rangle}{(1-c\|-\boldsymbol{p}_k^{\mathrm{h}} \oplus_c \boldsymbol{z}^{\mathrm{h}}\|^2)\|\boldsymbol{a}_k^{\mathrm{h}}\|}, \tag{12}$$

where $\boldsymbol{p}_k^{\mathrm{h}} \in \mathbb{B}_c^n$ and $\boldsymbol{a}_k^{\mathrm{h}} \in \mathcal{T}_{\boldsymbol{p}_k^{\mathrm{h}}}\mathbb{B}_c^n \setminus \{\boldsymbol{0}\}$. As the curvature parameter $c \to 0$, Eq. (11) converges to the Euclidean case.

In Euclidean space, MLR separates data using linear decision boundaries. In contrast, when applied to the Poincaré ball model, it uses geodesic decision boundaries (see Fig. 1(b)), resulting in nonlinear decision regions that better reflect hierarchical relationships in the data.

3. PROPOSED METHOD

3.1. Motivation and Strategy

The goal of the AFX chain classification task is to identify both the types and the order of AFXs applied to a given audio signal $\boldsymbol{s}$. Let $\mathcal{A}$ denote the set of all possible AFX chains constructed from F effect types, with a maximum chain length of L. Each AFX chain $a \in \mathcal{A}$ is treated as a distinct class, including the empty chain that applies no effects. The number of possible chains is given by $|\mathcal{A}| = \sum_{l=0}^{L} F!/(F-l)!$, where l denotes the length of the AFX chain and $F!$ is the factorial of F.

To address this problem, we design a neural network that embeds input audio signals into hyperbolic space and performs classification based on the applied AFX chain. Hyperbolic space is particularly suitable for this task for two main reasons. First, AFX chains can be interpreted as tree-like structures, where the signal is progressively modified by each effect unit in sequence. The number of possible chains increases exponentially with the number of effects, mirroring the exponential expansion property of hyperbolic space. This structural analogy allows the space to efficiently

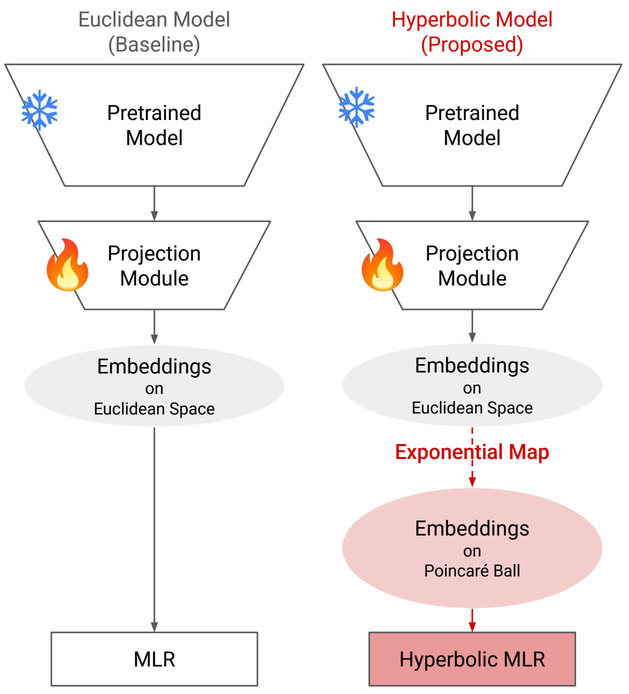

Euclidean Model (Baseline)
Hyperbolic Model (Proposed)
Pretrained Model
Pretrained Model
Projection Module
Projection Module
Embeddings on Euclidean Space
Embeddings on Euclidean Space
Exponential Map
Embeddings on Poincaré Ball
MLR
Hyperbolic MLR

Figure 2: Illustrative comparison of baseline and proposed networks for AFX chain classification. Baseline network classifies input audio signals in Euclidean space, whereas proposed network classifies them in hyperbolic space.

represent the combinatorial complexity of AFX chains. Second, AFX chains are inherently order-sensitive: changing the order of processors can drastically alter the resulting sound. For example, distortion followed by chorus yields a different result than the reverse order. This characteristic aligns with the non-commutativity of Möbius addition, as mentioned in Section 2.2, where the order of operations affects the outcome. Hyperbolic geometry thus provides a natural framework for capturing the directional dependencies present in ordered effect chains.

3.2. Network Architecture

On the basis of the considerations in Section 3.1, we develop a neural network architecture utilizing the Poincaré ball. It consists of four modules: *feature extraction module*, *projection module*, *mapping module*, and *classifier module* (see Fig. 2).

The feature extraction module analyzes the input audio to extract musically and acoustically meaningful representations. While previous work has often relied on hand-crafted features such as spectral, cepstral, or harmonic descriptors [4], or time-frequency features like mel-frequency cepstral coefficients and mel spectrograms [5], such representations may not fully capture the musical or perceptual characteristics relevant to modeling AFX chains. To extract richer and more context-aware features, we use MERT [15], a large-scale pretrained model for music representation learning. MERT is trained on a diverse collection of musical audio and is capable of capturing both low-level acoustic information and higher-level temporal structure through its transformer-based architecture. It processes raw waveforms via convolutional layers, followed by 24 stacked Transformer encoders. We use the output of the final layer and apply attention pooling to obtain a 1024-dimensional embedding in Euclidean space. The projection module is a feedforward neural network that maps the embedding to a J-dimensional feature vector. It consists of I fully connected (FC) blocks: the first $I-1$ blocks each include a FC layer, a rectified linear unit non-linearity, and layer normalization; the last block has only a FC layer. This transformation reshapes the representation to make it more amenable to geometric classification. The mapping module adapts the projected embedding to the target geometric space by applying the exponential map at the origin, as defined in Eq. (7), to project the Euclidean embedding onto the Poincaré ball. The classifier module performs multi-class classification using MLR for the Poincaré ball, as described in Section 2.3. The projection and classifier modules are trained to predict the AFX chain label from the input waveform using the cross-entropy loss.

The proposed network can be reduced to its Euclidean counterpart by replacing the exponential map in the mapping module with the identity function, and substituting the hyperbolic MLR in the classifier module with its Euclidean version. This results in a standard architecture that operates entirely in Euclidean space, serving as a baseline for comparison with the hyperbolic approach. The Euclidean network is trained in the same way as the hyperbolic one, as illustrated in Fig. 2.

4. EXPERIMENTS

4.1. Experimental Setup

Data preparation. We focus on guitar recordings for the AFX chain classification task, following [5]. We first collected clean (dry) recordings from Dataset 4 of the IDMT-SMT-GUITAR corpus, totaling 1.5 hours of audio. Then, we divided them into chunks of 10 seconds, collecting 533 samples of clean audio. To construct pairs of dry and AFX-chain-applied audio signals, we used Pedalboard[1], a Python library that can apply AFXs to audio signals without a digital audio workstation. Specifically, we selected three commonly used AFX types—*delay, chorus, and distortion*—and generated all possible permutations where each type appears at most once. This yielded $|\mathcal{A}| = 16$ chains (including the empty chain), as $F = 3$ and $L = 3$. We note that changing the order of these effects produces different audio outputs. For each chain, AFX parameters were randomly sampled within realistic ranges (see Table 1) to reflect practical usage scenarios. The resultant dataset consisted of 8,528 samples of dry and processed signals (23.7 hours in total), which we randomly split into training, validation, and test sets with a 70/15/15 ratio.

Compared networks. We compared the proposed network with its Euclidean counterpart described in Section 3, varying the output feature size of the projection module: J = 64, 128, 256, and 512. In both models, the number of fully connected (FC) layers in the projection module was fixed at $I = 3$, with the hidden layers set to $J/2$ units. For the proposed network, we additionally varied the curvature parameter with values $c = 0.001$, 0.01, 0.1, and 1.0.

Training setting. Training was performed for 100 epochs using the cross-entropy loss between the predicted and ground-truth AFX chain classes. For optimization, we used the AdamW optimizer [16] for the Euclidean model and the Riemannian Adam optimizer [17] for the hyperbolic model. The latter is a hyperbolic extension of the Adam optimizer, implemented in the geoopt library [18]. We applied weight decay with a coefficient of 1.0×10^{-5} and used a batch size of 32. The learning rate was initialized at 1.0×10^{-4} and halved whenever the validation loss did not improve for five consecutive epochs. For each configuration, we chose the model that achieved the lowest validation loss.

[1]`https://github.com/spotify/pedalboard`

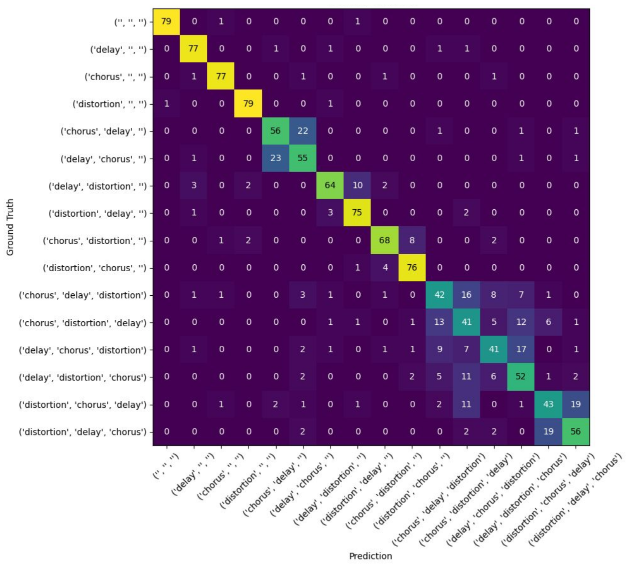
Table 1: Functions of Pedalboard library corresponding to AFXs used, their parameters, and randomization ranges. For parameters not listed here, default values provided by library were used

Pedalboard Functions	Parameter	Range	Unit
pedalboard.Chorus	rate_hz	0.1–1.5	Hz
	depth	0.1–1.0	-
	feedback	0.0–0.5	-
pedalboard.Distortion	drive_db	5–15	dB
pedalboard.Delay	delay_seconds	0.1–1.0	s
	feedback	0.0–0.75	-

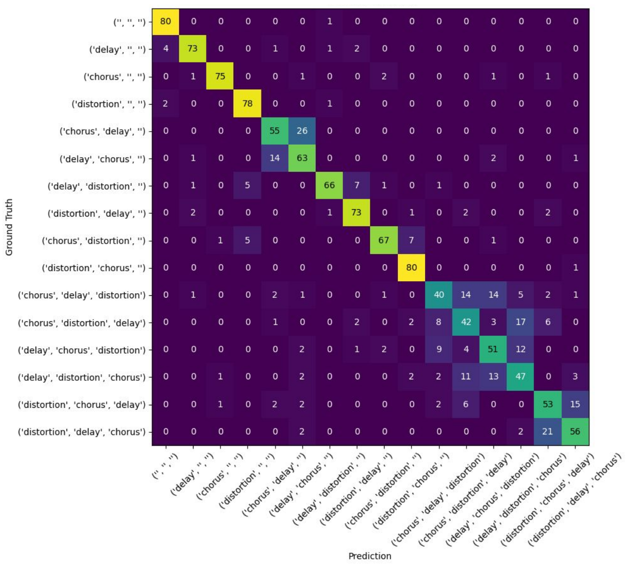
Table 2: Averages and standard errors of macro and micro F_1 scores obtained with Euclidean and proposed (hyperbolic) networks

Geometry	c	J	Macro F_1	Micro F_1
Euclidean	-	64	0.748 ± 0.003	0.750 ± 0.003
	-	128	0.737 ± 0.011	0.740 ± 0.011
	-	256	0.743 ± 0.006	0.747 ± 0.005
	-	512	0.724 ± 0.002	0.727 ± 0.002
Hyperbolic	0.001	64	0.748 ± 0.007	0.752 ± 0.007
	0.001	128	0.736 ± 0.003	0.739 ± 0.003
	0.001	256	0.725 ± 0.003	0.729 ± 0.003
	0.001	512	**0.740** ± 0.004	**0.742** ± 0.004
Hyperbolic	0.01	64	0.736 ± 0.005	0.740 ± 0.005
	0.01	128	0.745 ± 0.007	0.747 ± 0.007
	0.01	256	0.718 ± 0.003	0.721 ± 0.004
	0.01	512	0.736 ± 0.007	0.739 ± 0.006
Hyperbolic	0.1	64	0.744 ± 0.003	0.747 ± 0.003
	0.1	128	0.731 ± 0.006	0.734 ± 0.006
	0.1	256	0.697 ± 0.026	0.705 ± 0.024
	0.1	512	0.722 ± 0.007	0.725 ± 0.007
Hyperbolic	1.0	64	**0.751** ± 0.006	**0.755** ± 0.006
	1.0	128	**0.756** ± 0.007	**0.758** ± 0.007
	1.0	256	**0.746** ± 0.004	**0.752** ± 0.003
	1.0	512	0.734 ± 0.009	0.740 ± 0.007

Evaluation metrics. As evaluation metrics, we used *macro and micro F_1 scores* as in [5]. The macro F_1 score computes the F_1 score for each class independently and averages them, giving equal weight to all classes regardless of their frequency. In contrast, the micro F_1 score aggregates the contributions of all classes to compute an overall F_1 score, which tends to reflect performance on more frequent classes. To account for variance due to random initialization, we report the average and standard error of each metric over three different random seeds.



(a) Euclidean network.



(b) Proposed network.

Figure 3: Confusion matrices of (a) Euclidean and (b) proposed networks. Each label is represented as a tuple of three elements, where each position corresponds to an effect slot in the chain. For example, (", ", ") indicates no AFX applied, ('delay', ", ") represents a single delay effect, and ('chorus', 'delay', 'distortion') denotes a chain where chorus is applied first, followed by delay and distortion.

4.2. Results

Table 2 shows the macro and micro F_1 scores achieved by the Euclidean network and the proposed hyperbolic networks across different embedding dimensions J and curvature values c. The proposed network with $c = 1.0$ consistently achieved higher average macro and micro F_1 scores than the Euclidean network for $J = 64$, 128, and 256. The best performance was observed at $J = 128$ and $c = 1.0$, with average macro and micro F_1 scores of 0.756 and 0.758, respectively. Although the performance gains over the Euclidean network are modest, the improvements are consistent when an appropriate curvature is used. These results highlight the potential of hyperbolic space as a stable and effective embedding space for AFX chain classification.

4.3. Analysis

To better understand what aspects of AFX chain classification benefit from hyperbolic space, we analyzed the results in terms of AFX type and order sensitivity.

Effect of AFX type and length. Figure 3 shows the confusion matrices for the Euclidean and proposed hyperbolic networks. For both models, we used the configurations that achieved the best F_1 scores reported in Table 2. Prediction became increasingly difficult as the length of the AFX chain increased. In particular, AFX chains containing both chorus and delay effects tend to be confused. This may be because lowering the modulation parameter of the chorus effect causes its output to resemble a signal where a delayed version of the input is added to the input itself, making it virtually similar to a delay effect.

Notably, both networks exhibited low confusion between chains of different lengths, suggesting that the models accurately inferred which AFX types were applied. When evaluating only the presence of AFX types, where a prediction is considered correct if it includes the correct set of effects regardless of their order, the average macro and micro F_1 scores were 0.986 and 0.986 for the Euclidean networks, and 0.987 and 0.987 for proposed networks. This negligible difference indicates that the improvement of the proposed method primarily stems from its enhanced ability to capture AFX order, rather than simply identifying which effects are present.

Analysis of order sensitivity. To further explore this, we analyzed partial AFX order prediction using two metrics: *first-N F_1 score* and *latest-N F_1 score*. The first-N F_1 score quantifies how well the first N effects in the predicted and ground-truth chains match in order. If a chain contains fewer than N effects, it is padded with empty entries to ensure a consistent length. The latest-N F_1 score is computed in the same way, but considers the last N effects instead. Since the results for $N = 3$ are identical to those in Table 2, we report only the scores for $N = 1$ and 2.

Tables 3 and 4 show the first- and latest-N F_1 scores, respectively. For each network, we used the configuration that achieved the best performance in Table 2: $J = 64$ for the Euclidean network and $(c, J) = (1.0, 128)$ for the proposed network. Across all evaluated settings, the proposed network achieved comparable or better performance than its Euclidean counterpart. This consistent advantage supports the effectiveness of hyperbolic geometry in capturing the order-sensitive structure of AFX chains.

This analysis also reveals a common tendency across both models. Both metrics dropped from $N = 1$ to $N = 2$ as chain length increased, which is consistent with the performance degradation seen in Table 2. Interestingly, first-1 scores consistently outperformed latest-1 scores, suggesting that both models are better at identifying the earliest-applied AFX. However, for $N = 2$, the latest-N scores surpassed the first-N ones. This may indicate that while the first AFX dominates the representation, errors accumulate more rapidly when attempting to track the entire sequence from the beginning. In contrast, although predicting later AFXs

Table 3: Micro and macro first-N F_1 scores obtained with Euclidean and proposed network

Geometry	**First-1**		**First-2**	
	Macro	**Micro**	**Macro**	**Micro**
Euclidean	0.860 ± 0.003	0.833 ± 0.003	0.791 ± 0.003	0.753 ± 0.003
Hyperbolic	**0.866** ± 0.004	**0.840** ± 0.006	**0.798** ± 0.005	**0.762** ± 0.007

Table 4: Micro and macro latest-N F_1 scores obtained with Euclidean and proposed network

Geometry	**Latest-1**		**Latest-2**	
	Macro	**Micro**	**Macro**	**Micro**
Euclidean	0.848 ± 0.002	0.817 ± 0.003	0.801 ± 0.002	0.765 ± 0.002
Hyperbolic	**0.852** ± 0.003	**0.822** ± 0.005	**0.806** ± 0.004	**0.771** ± 0.005

is generally more difficult, the degradation from 1 to 2 effects is less steep when evaluated in reverse. In contrast, although predicting later AFXs is generally more difficult, the degradation from 1 to 2 effects is less steep when evaluated in reverse. These observations may raise questions about the suitability of order-agnostic methods [6] or approaches that rely on detecting the last-applied AFX [7]. We leave its further investigation as future work.

5. CONCLUSIONS

We proposed an AFX chain classification method that jointly estimates AFX types and their order. To construct the proposed method, we used the Poincaré ball model, a realization of hyperbolic space, to better capture the order-sensitive and combinatorially growing nature of AFX chains. To perform classification on hyperbolic space, we added an extra mapping at the end of a Euclidean baseline network, then applied hyperbolic MLR to embeddings that correspond to input audio signals. Experimental results on guitar recordings demonstrated that, with an appropriate curvature, the proposed method consistently outperforms its Euclidean counterpart. Further analysis revealed that these performance gains stem primarily from improved modeling of AFX order, rather than AFX type identification alone. These findings highlight the potential of hyperbolic geometry for the order-aware AFX chain classification.

6. ACKNOWLEDGMENTS

This work was supported by JSPS Grants-in-Aid for Scientific Research JP23K28108.

7. REFERENCES

[1] T. Wilmering, D. Moffat, A. Milo, and M. B. Sandler, "A history of audio effects," *Applied Sciences*, vol. 10, no. 3, 01 2020, 791:1–791:27.

[2] U. Zölzer, *DAFX-Digital Audio Effects (Second Edition)*, John Wiley & Sons, 2011.

[3] B. De Man, *Audio Effects in Sound Design*, pp. 113–128, Taylor & Francis, 06 2019.

[4] M. Stein, "Automatic detection of multiple, cascaded audio effects in guitar recordings," in *Proceedings of International Conference on Digital Audio Effects*, Jan. 2010.

[5] J. Guo and B. McFee, "Automatic recognition of cascaded guitar effects," in *Proceedings of International Conference on Digital Audio Effects*, 2023, pp. 189–195.

[6] M. Rice, C. J. Steinmetz, G. Fazekas, and J. D. Reiss, "General purpose audio effect removal," in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2023.

[7] O. Take, K. Watanabe, T. Nakatuka, T. Cheng, T. Nakano, M. Goto, S. Takamichi, and H. Saruwatari, "Audio effect chain estimation and dry signal recovery from multi-effect-processed musical signals," in *Proceedings of International Conference on Digital Audio Effects*, 2024.

[8] S. Lee, M. A. Martínez-Ramírez, W.-H. Liao, S. Uhlich, G. Fabbro, K. Lee, and Y. Mitsufuji, "Searching for music mixing graphs: A pruning approach," in *Proceedings of International Conference on Digital Audio Effects*, 2024.

[9] C. De Sa, A. Gu, C. Ré, and F. Sala, "Representation tradeoffs for hyperbolic embeddings," in *Proceedings of International Conference on Machine Learning*, 2018, vol. 80, pp. 4460–4469.

[10] F. Nakashima, T. Nakamura, N. Takamune, S. Fuayama, and H. Saruwatari, "Hyperbolic timbre embedding for musical instrument sound synthesis based on variational autoencoders," in *Proceedings of Asia Pacific Signal and Information Processing Association Annual Summit and Conference*, 2022, pp. 736–743.

[11] D. Petermann, G. Wichern, A. Subramanian, and J. Le Roux, "Hyperbolic audio source separation," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2023.

[12] D. Petermann and M. Kim, "Hyperbolic distance-based speech separation," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2024, pp. 1191–1195.

[13] F. G. Germain, G. Wichern, and J. Le Roux, "Hyperbolic unsupervised anomalous sound detection," in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*, 2023.

[14] R. Shimizu, Y. Mukuta, and T. Harada, "Hyperbolic neural networks++," in *Proceedings of International Conference on Learning Representations*, 2021.

[15] Y. Li, R. Yuan, G. Zhang, Y. Ma, X. Chen, H. Yin, C. Xiao, C. Lin, A. Ragni, E. Benetos, N. Gyenge, R. Dannenberg, R. Liu, W. Chen, G. Xia, Y. Shi, W. Huang, Z. Wang, Y. Guo, and J. Fu, "MERT: Acoustic music understanding model with large-scale self-supervised training," in *Proceedings of International Conference on Learning Representations*, 2024.

[16] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *Proceedings of International Conference on Learning Representations*, 2019.

[17] H. Kasai, P. Jawanpuria, and B. Mishra, "Riemannian adaptive stochastic gradient algorithms on matrix manifolds," in *Proceedings of International Conference on Machine Learning*, 2019, pp. 3262–3271.

[18] M. Kochurov, R. Karimov, and S. Kozlukov, "Geoopt: Riemannian optimization in pytorch," 2020.

[19] S. Lee, J. Park, S. Paik, and K. Lee, "Blind estimation of audio processing graph," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, 2023.