Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ingeniería Eléctrica
IEE2985 – Investigación en Pregrado

KEMAR's Dataset Neural Spatial Interpolation

Guide Professor: Rodrigo Fernando Cádiz Cádiz
Estudiante: Juan Ignacio Lorca (`juan.lorca@uc.cl`)

Abstract

Estimation of accurate head-related transfer functions (HRTF) is crucial for spatial audio rendering. This transfer functions depend of source/listener locations and can be measured at a thousand or even more distinct spatial locations in an anechoic chamber. Therefore, data captures are expensive and require highly specialized equipment and capture stages. In this work, we present a data driven approach were we intend to use neural networks and generative models to reconstruct and interpolate between spatial measurements of the KEMAR's HRTF Dataset.

1 INTRODUCTION

In many multimedia applications, such as virtual reality, gamming, spatial music, etc., the head related impulse response (HRIR) or transfer functions are required to achieve realistic binaural acoustic experience. HRIRs or HRTFs are functions which parameterize the acoustic transfer from a sound source to the ears of a listener, incorporating cues for sound localization such as internaural time and level differences (ITDs, ILDs) as well as spectral cues that arise due to the interaction of our auditory system, head an torso with the sound field [1].

When a sound signal is filtered by convolving in time domain with a pair of HRIRs (or in frequency domain through multiplication with the respective pair of HRTFs), and presented in stereo form to headphones; it gives the listener the impression that the sound is playing or coming from and specific spatial direction in the 3D space [2].

HRTFs can be obtained by meaning of numerical calculations using spherical harmonics or other methods such as the boundary element method [3]. Another way to get this recordings is by measuring spatial-distinct impulse responses in an anechoic chamber which is rather expensive, complicated and time-consuming, specially when you want to obtain personalized HRTFs. In this work, we propose a data-driven deep learning approach that intends to do data augmentation by generating those points in the 3D space that are not presented in the KEMAR's HRTF dataset. In other words, we are interpolating between 3D spatial points in order to obtain that specific HRTF pair for binaural rendering. By doing this, we circumvent the need for such expensive data captures and we overcome the time-consuming problem.

While the proposed method is fully data-driven and only limited by the amount of data that the dataset can provide, other spatial interpolation techniques require to deal explicitly with problems due to limited spatial directions. Methods based on bilinear interpolation or barycentric weighting interpolation [4] that use neighboring HRTFs measurements do not provide sufficiently accurate or high quality HRTFs from sparse measurements, due to the high spatial complexity of the HRTF, especially at high frequencies [5].

Setting aside the conventional signal processing pipeline for HRTFs identification and interpolation by using one of the aforementioned methods; this report explores a new machine learning (ML) approach for data generation and reconstruction. The motivation is based on the fact that ML generative models have yield great results in audio generation tasks like in [6], or in [7]. Also, ML models can efficiently encode and interpolate data by leveraging domain-specific appearance of signals, and do not impose implicit assumptions (e.g., linearity, minimum phase) constraining conventional HRTFs identification and interpolation methods. Thus the problem can be formulated as a conditional probabilistic generative model such as a Varational

Auto-Encoder (VAE) or a Generative Adversarial Network (GAN) that must learn to encode the spatial conditioned data distribution into a lower-dimensional space called latent space, and to be able to reconstruct the encoded input by sampling in the learnt distribution. To solve this task, we propose a Conditional Varational Auto-Encoder (CVAE) which is a VAE conditioned on some label, in this case given by the 3D spatial position of the sound emitting source -see Figure 1, that predicts the two ear time-domain HRIR. The motivation behind the use of this model is that as a probabilistic take of an Auto-Encoder (AE), produces a continuous, structured latent space, which is useful for data generation.

Note that this approach is the first to address the task of learning to generate binaural impulse responses. There has been some work on generating room impulse responses for a given acoustic environment using GANs in [8]. In terms of binaural audio generation, great work has been pulled off by [9] and [10], where they trained Temporal Convolution Networks (TCN) to predict a binaural representation from a monaural input by learning HRTFs implicitly.

2 GENERATIVE MODEL AS A SPATIAL INTERPOLATOR

2.1 Impulse Response Generators

Extensive research has been conducted in the field of neural network-based audio generation and synthesis. Models that combine digital signal processing (DSP) modules with deep neural network models (DNN) [11, 12] have achieved high fidelity audio synthesis and waveform generation since DSP modules incorporate strong domain knowledge of signal processing and perception and neural networks introduce the power of expressiveness to the whole model. In fact, the model proposed in [12] shows a conditional network that pre-processes the input acoustic features for a source module, that generate a sine-based signal as excitation for the model, and for a dilated-convolution-based filter module that transforms the excitation into a waveform.

Now, for the impulse response generation task, the model proposed in [8] uses GANs to take rectangular room dimensions, listener and speaker positions, and reverberation time (T_{60}) as inputs through an embedding vector and generates specular and diffuse reflections for a given acoustic environment. Another work [13], has done a parameterized impulse response estimation. In this work, they considered linear time invariant (LTI) systems characterized by spatio-temporal impulse responses to be parametrized by using multi-layer perceptrons (MLPs) and a Fourier features,

$$\gamma(p) = \{\sin(2^l \pi p), \cos(2^l \pi p);\ 0 \leq l \leq L\} \tag{1}$$

where they argued that computing this features enables the MLPs to learn high frequency functions in low dimensional domains.

2.2 Proposed Model

The primary goal of this study is to synthetically generate spatial points that encapsulate the specific Head-Related Impulse Response (HRIR) pair. Notably, the approach aims to achieve this without relying on any interpolation techniques. Within this framework, the task involves the reconstruction and generation of new impulse responses, a process intricately tied to the spatial location $\mathbf{p}_s = (\mathbf{x}_s, \mathbf{y}_s, \mathbf{z}_s)$ of the sound emitting source, by sampling from a high expressive encoding derived from the given data. To do so, we used a VAE architecture that was conditioned order to use the spatial position of the source as an input to reconstruct the specific pair of impulse responses. A Varational Auto-Encoder aims to understand a set $\mathbf{x}$ by modeling the underlying probability distribution of the data $p(\mathbf{x})$. This generative model encodes a lower-dimensional space $\mathbf{z}$ called latent space by learning a *prior* distribution $p(\mathbf{z})$ from the dataset. Then, the data $\mathbf{x}$ is generated from a generative distribution $p_\theta(\mathbf{x}|\mathbf{z})$ where θ are the generative model's parameters [13]. The complete model is defined by by the joint distribution:

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \tag{2}$$

Unfortunately, real-world data follow complex distributions, which cannot be found analytically. The idea of variational inference (VI) is to solve this problem through optimization by assuming a simpler approximate distribution $q_\phi(\mathbf{z}|\mathbf{x})$ from a family of approximate densities [13]. The goal of VI is to minimize the difference between this approximation and the real distribution, by minimizing the Kullback-Leibler (KL) divergence between these probability densities:

$$q^*_\phi(\mathbf{z}|\mathbf{x}) = \text{argmin}_{q_\phi(\mathbf{z}|\mathbf{x})} \mathcal{D}_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})\right] \tag{3}$$

By developing this KL divergence and re-arranging terms, we obtain:

$$\log p(\mathbf{x}) - \mathcal{D}_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})\right] = \mathbb{E}_\mathbf{z}\left[\log p(\mathbf{x}|\mathbf{z})\right] - \mathcal{D}_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right] \tag{4}$$

This formulation describes the quantity we want to model $\log p(\mathbf{x})$ minus the error we make by using an approximate q_ϕ instead of the true p_θ [15]. Therefore, we can optimize this alternative objective, called the evidence *lowerbound* (ELBO):

$$\mathcal{L}_{\phi,\theta} = \mathbb{E}_{q_\phi(\mathbf{x}|\mathbf{z})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathcal{D}_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right] \leq \log p_\theta(\mathbf{x}) \tag{5}$$

The ELBO intuitively minimizes the reconstruction error through the likelihood of the data given a latent $\log p_\theta(\mathbf{x}|\mathbf{z})$, while regularizing the distribution $q_\phi(\mathbf{z}|\mathbf{x})$ to follow a given prior distribution $p_\theta(\mathbf{z})$. We can see that this equation involves $q_\phi(\mathbf{z}|\mathbf{x})$ which encodes the data $\mathbf{x}$ into the latent representation $\mathbf{z}$ and a decoder $p_\theta(\mathbf{x}|\mathbf{z})$, which generates $\mathbf{x}$ given a $\mathbf{z}$. This structure defines the Variational Auto- Encoder, where we can use parametric neural networks to model the encoding (q_ϕ) and decoding (p_θ) distributions.

Our model considers a Conditional Varational Auto-Encoder [14] which has some changes in the *prior* distribution as well as in the generative distribution. Here, we define a label $\mathbf{y}$ that allows our model to approximate a conditional distribution $p(\mathbf{x}|\mathbf{y})$ where the latent $\mathbf{z}$ is will be drawn from the prior $p(\mathbf{z}|\mathbf{y})$ and the output $\mathbf{x}$ is generated from the distribution $p(\mathbf{x}|\mathbf{y}, \mathbf{z})$. In order to do so, now the encoder tries to learn $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$ which is equivalent to learning hidden representation of data $\mathbf{x}$ or encoding the $\mathbf{x}$ into the hidden representation conditioned by $\mathbf{y}$, and the decoder part tries to learn $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y})$ which decodes the hidden representation to the input $\mathbf{x}$ space conditioned by $\mathbf{y}$. The ELBO for this model is the following:

$$\mathcal{L}_{\mathbf{y},\phi,\theta} = \mathbb{E}_{q_\phi(\mathbf{x}|\mathbf{z},\mathbf{y})}\left[\log p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{y})\right] - \mathcal{D}_{KL}\left[q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})||p(\mathbf{z}|\mathbf{y})\right] \leq \log p_\theta(\mathbf{x}|\mathbf{y}) \tag{6}$$

For both models a reparametrization trick [14] is used when encoding the latent space: $\mathbf{z} = \mu_\phi(\mathbf{x}) + \epsilon \cdot \sigma_\phi(\mathbf{x})$, which is done to be able to compute the gradient, when differentiating, of the variational lower bound with respect to the parameters given by ϕ. Both mean $\mu(\mathbf{x})$ and standard deviation $\sigma(\mathbf{x})$ are encoded by the distribution $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})$ and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, where $\mathbf{I}$ is the identity matrix.

The block diagram of the proposed neural network model is shown in -Figure 1. As we said earlier, the model's architecture is inspired by a traditionally CVAE with a conditioned label $\mathbf{p}_s$. The model consists in two main modules: an encoder that tries to approximate the data distribution to a Normal distribution into a lower-dimensional space by encoding $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$, and a decoder that is in charge of reconstructing back the encoded input $\mathbf{z}$ to its main form $\mathbf{x}$. Despite by the reference architecture mentioned above, we propose different encoders and decoders layer compositions as well as different ways to condition the information that the model is seeing.

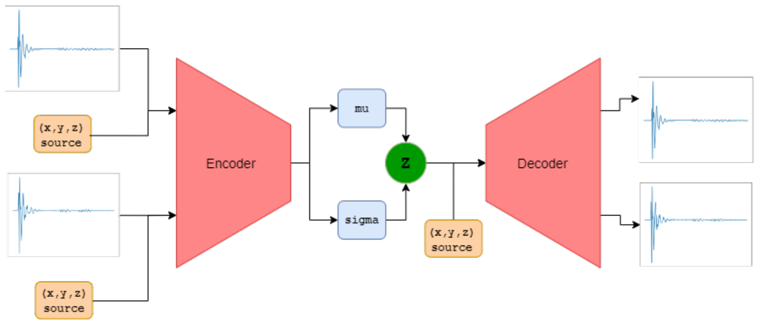


Figure 1: Varational Auto-Encoder Diagram

2.2.1 Encoder

The encoder module in -Figure 1 transforms both right $h_r(n)$ and left $h_l(n)$ impulse responses into an intermediate probabilistic space by encoding the mean and standard deviation of the input data. The latent representation is reached by stacking multiple 1-D convolutional and hyper-dilated convolutional layers with a rectified linear unit (ReLU), or a parametric rectified linear unit (PReLU) as activation functions to introduce non-linearity to the model enabling it to learn complex patterns and representations of the data. For this module we encoded two latent representations $\mathbf{z}_{left} \in \mathbb{R}^{32}$ and $\mathbf{z}_{right} \in \mathbb{R}^{32}$ by doing the reparametrization trick:

$$\mathbf{z}_{left} = \mu(\mathbf{h}_{left}) + \epsilon \cdot \sigma(\mathbf{h}_{left}) \quad \mathbf{z}_{right} = \mu(\mathbf{h}_{right}) + \epsilon \cdot \sigma(\mathbf{h}_{right}) \tag{7}$$

We took two different approaches when configuring and designing the encoder module. The first one was simply a multiple stack of 1-D convolutional layers and dilated convolutions, reducing the number of filters by two in each $Conv1D$ block to down sample the data. We also kept the same kernel size for each convolution layer and we added hyper-dilated convolutions to address the problem of reaching a sufficient receptive field size instead of increasing the kernel size which leads to large number of parameters of the convolutional neural network (CNN) and a high computational complexity [15]. Every two $Con1D$ stack we added a ReLU non linearity layer.

The second encoder model has the same architecture and layers as the first one, but instead of having a ReLU non-linearity as an activation function, we added a PReLU layer like in [16]. The parametric rectified linear unit performs the following operation:

$$PReLU(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha \cdot x & \text{otherwise} \end{cases} \tag{8}$$

where $\alpha \in \mathbb{R}$ is a trainable scalar controlling the negative slope of the rectifier.

2.2.2 Decoder

The decoder module in -Figure 1 is in charge of reconstructing the impulse responses $\hat{h}_l(n)$ and $\hat{h}_r(n)$ from the encoded normally distributed latent space. For the encoder module, we adopted two distinct design approaches. The initial approach involved employing MLPs at the beginning of the network, followed by concatenating their outputs with the result of a linear projection applied to the conditional input (see -Figure 2). Then, those layers were followed by 1-D transpose convolutions with sine activation functions. We used the same MLP network as in [11], but without the recurrent layer Gate Recurrent Unit (GRU). The transpose convolution network $Conv1DTranspose$ tries to mirror the encoder architecture, where kernel size is double in each convolutional layer. As we said, this convolutional layers have sine activation functions which have been shown to be beneficial for modeling higher frequencies if convolutional weights are initialized appropriately [17].

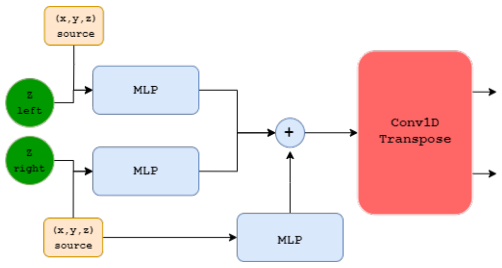
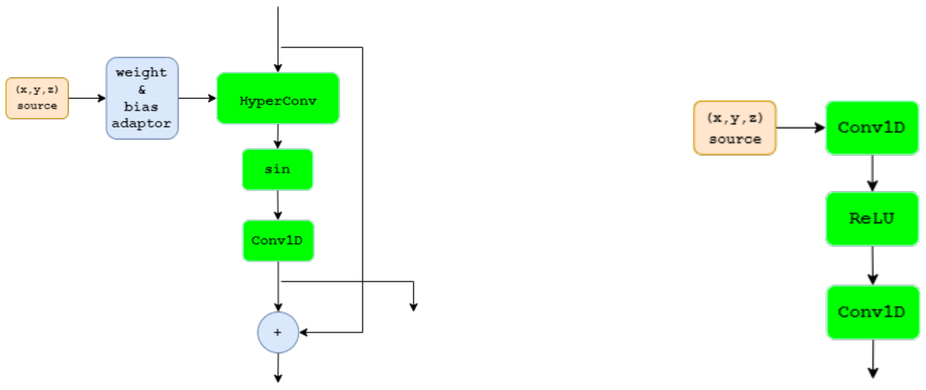


Figure 2: Decoder Diagram

For the second decoder we used again MLPs as our first stage network and then, we applied TCNs to each left and right outputs, followed by a *Concat* layer with a PReLU activation function. Then, those conditioned outputs were passed to the $Conv1DTranspose$ network to reconstruct $\hat{h}_l(n)$ and $\hat{h}_r(n)$. We used the same temporal convolution network as in [10]. Figure 3a depicts the layers that are involved in this block as well as the adaptor layer to generate the weights an hyper-dilated convolution network.



(a) Temporal Convolutional Network

(b) Weights & Biases adaptor layers

Figure 3: TCN design

2.2.3 Conditional Network

The conditional network was designed to condition the model by using the 3-D audio source position: $\mathbf{p}_s = (\mathbf{x}_s, \mathbf{y}_s, \mathbf{z}_s)$. As we said in the decoder section, this module has adaptor that could be the same MLP as in the first decoder or could be a stack of convolutional layers that outputs the weights for hyper-dilated convolutions. We concatenate the condition input to both latent space obtained from the encoder and then this kind of embedding is passed through a condition stage given by the MLPs or the TCNs.

We found another two ways of conditioning our model, but those were more fitted to an AE instead of a VAE, and both resort to conditioned hyper convolutions where base weights for the layers are generated from another adaptor network that has as an input the 3-D spatial information for the binaural signals. The first approach taken on this neural spatial conditioning was done in [10] where they used a neural time warpping before feeding a source sequence to the hyper convolutional block. This neural time warpping maps a temporal sequence to a target sequence and it has been used in tasks like speech recognition or audio retrieval. The main idea of this warpping is to find a warpfield ρ_T that warps a source signal x_T (where T represents the sequence's length) to a target signal $\hat{x}_T$ such that the distance between the signals is minimized:

$$\rho_T = \underset{\overline{\rho}_T}{\operatorname{argmin}} \sum_t ||\hat{x}_t - x_{\overline{\rho}_t}||, \quad \text{where } \rho_t \in \{1, ..., T\} \tag{9}$$

where the warpfield is typically constrained to respect physical properties such as monotonicity ($\rho_t \geq \rho_{t-1}$) and causality ($\rho_t \leq t$). This is would be useful since for binaural audio, there is a clear monotonous and causal relationship between source and target signal but the target signal is unknown at inference time. The warpfield is estimated from an input conditioning $\mathbf{c}_T \in \mathbb{R}^{14}$ where 3 of those 14 entries depicts the sound source position $\mathbf{p}_s$ and the the other 11 entries are related to the orientation of source and listener. A geometric warping approach is taken which is based on the speed of sound $\nu_{sound} \approx 343\ \frac{m}{s}$ and the distance between the source position ($\mathbf{p}_t^{(src)} = \mathbf{p}_s$) and listener position ($\mathbf{p}_t^{(lst)}$) at time t, then:

$$\rho_t^{(geom)} = t - ||\mathbf{p}_t^{(src)} - \mathbf{p}_t^{(lst)}|| \cdot \frac{f_s}{\nu_{sound}} \tag{10}$$

where f_s represents the audio sample rate. Finally, the warped signal can now be computed using the predicted warpfield. Since the warpfield elements ρ_t are typically not integers, the warped signal $\hat{x}_T$ is define to be the linear interpolation of the original signal x_T at position $\lfloor \rho_t \rfloor$ and $\lceil \rho_t \rceil$:

$$\hat{x}_t = (\lceil \rho_t \rceil - \rho_t) \cdot x_{\lfloor \rho_t \rfloor} + (\rho_t - \lfloor \rho_t \rfloor) \cdot x_{\lceil \rho_t \rceil} \tag{11}$$

So, two warpfields are generated, one for each ear.

A second way to conditon the neural network on spatial information was done in [9] where they masked a learned frequency domain monaural input through multiplication with two masking functions $\mathbf{M}^{left}$ and $\mathbf{M}^{right}$. They argued that the masks embed ITDs, ILDs and spectral cues which are HRTFs primary cues for auditory localization. This functions are obtained by passing the mono input through a left and right temporal hyper-convolutional network, where the a dilated convolutional layer have its weights and biases generated by an adaptor network with the spatial condition $\mathbf{c} = (\mathbf{R}_l^k, \mathbf{P}_s^k, \mathbf{R}_s^k)_{k=1}^K$ as an input. Here, $\mathbf{R}_l$ represents a rotation vector that describes the listener's head orientation, $\mathbf{P}_s$ is the source 3-D position, $\mathbf{R}_s$ is the rotation vector associated to the audio source and K being the number of source/listener's 3-D positions and orientations.

The main idea presented in both [9] and [10] is to learn implicitly the filtering action of an HRTF by training a neural network that learns the mapping given by the following equation:

$$(\hat{\mathbf{y}}^{left}(n), \hat{\mathbf{y}}^{right}(n)) = f(\mathbf{x}(n), \mathbf{c}) \tag{12}$$

when being trained with binaural signals $(\mathbf{y}^{left}(n), \mathbf{y}^{right}(n))$.

2.3 A Differentiable Digital Signal Processing Approach

Soon to be discussed with the mentor. Some neural deconvolution method that outputs both right $\hat{h}_r$ and left $\hat{h}_l$ HRIRs and the those are passed to a filter module that applies cyclic convolution through matrix dot multiplication inspired by [19] has been approached.

3 EXPERIMENTS

3.1 Dataset

We used Massachusetts Institute of Technology (MIT) HRTF Measurements of a KEMAR Dummy-Head Microphone [20] that consist of the left and right ear head related impulse responses from a Realistic Optimus Pro 7 loudspeaker mounted 1.4 meters from the KEMAR manikin. Maximum length pseudo-random binary sequences were used to obtain the impulse responses at a sampling rate of 44.1 kHz. A total of 710 different positions were sampled at elevations ϕ from $-40°$ to $+90°$ with a variable spatial resolution for the azimuth angle θ. Each HRIR has 512 samples which yields to 8.2431 second of audio. We considered that such a short time for recording falls short considerably compared to the 2.6 hours of recordings taken in [9] or the 2 hours in [10]. Also, the data collected by this dataset was recorded using only one subject (the manikin) with a head breath of 15.2 cm, a head length of 19.1 cm and a head height of 12.5 cm that was measured from the ears to the top of the head [21].

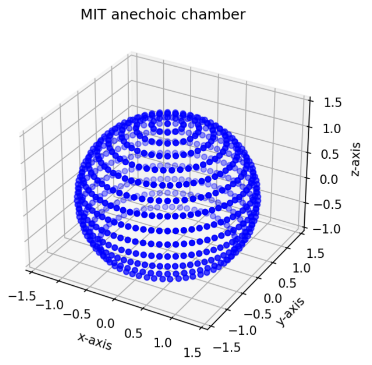
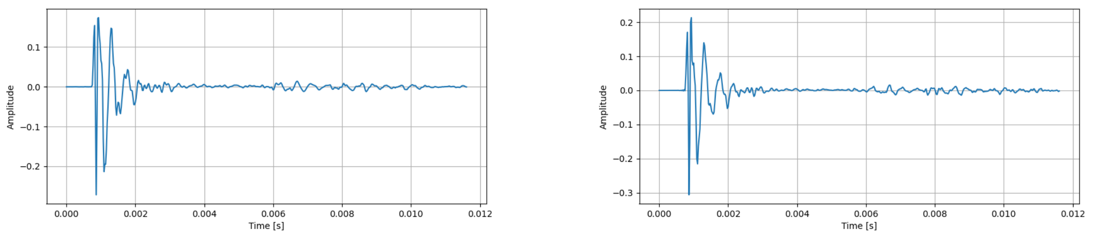


Figure 4: Spherical points measured in MIT's anechoic chamber

Cartesian coordinates $\mathbf{p}_s = (\mathbf{x}_s, \mathbf{y}_s, \mathbf{z}_s)$ where used to condition our model. Each 3-D vector was created using the elevation ϕ and the azimuth θ angles of the sound source placed at a distance of 1.4 m which we took as the radius [20].

We can also examine Figure 5 to observe the appearance of the HRIR pair (left and right), consisting of 512 samples at 44.1 kHz. This representation corresponds to coordinates with a $\phi = 90°$ elevation and $\theta = 0°$ azimuth.



(a) HRIR left at $\phi = 90°$ and $\theta = 0°$

(b) HRIR right at $\phi = 90°$ and $\theta = 0°$

Figure 5: Time-domain HRIRs

3.2 Implementation

We outline the implementation details and further refer the interested reader to the code[1]. The whole neural network model was programmed in python using mostly the ML library TensorFlow `2.12.0`.

For the encoder module we set the kernel size and strides with values 4 and 1 respectively. For the decoder module we kept the same number of strides but we increased by two the kernel size for the $Conv1DTranspose$. The number of output channels for the encoder is the dimension of the latent space and it was set to 32.

[1]GitHub repository: `https://github.com/juanpiLorca/iPre_HRTF_Interpolation.git`

We also applied l_2-normalization to the filter's coefficients in the encoder module before computing the convolution to prevent the network to learn scaling factors and overcome problems related to little data for training and the use of deep neural networks with more than ten million parameters. For the temporal encoder network we used 1 block of 8 layers of ResBlock which are the layers used in the TCN. The desing of the ResBlock was the same as in [10] with a custom `HyperConv` network followed by a residual $Conv1D$ and a skip layer that is also a 1-D convolution. Here, the hyper-dilated convolutional layer has a sine activation function and its kernel is initialized like [18]. We also incorporate increasing dilation factors that are doubled after each layer. All kernels size in this module were set to 4, for the adaptor set to 2, and all strides were set to 1. As we said earlier in this report the MLP used in the decoder and the adaptor is a standard MLP that has three dense layers, three ReLUs nonlinearity and three layer normalization.

3.3 Loss Function

In order to train our model, the loss function intends to minimize the multi-scale Short-Time Fourier Transform (STFT) which has been used in works related to generate audio through neural networks [11] and to replace commonly point-wise losses in raw audio waveform. Let L_i be a single STFT complex spectogram l_1 loss with a given Fast Fourier Transform (FFT) of size i. The total loss is then the sum of all the spectral losses for the 512 samples left and right impulse response channels:

$$L_{total} = \sum_i L_i^{left} + \sum_i L_i^{right} \tag{13}$$

where we used FFT sizes (512, 256, 128, 64) and the neighboring frames in the STFT overlap by 75% like in [9, 11]. Also, the l_1 loss is defined:

$$Loss_{l_1} = \sum_{j=1}^{N} |\mathbf{x} - \hat{\mathbf{x}}| \tag{14}$$

and it minimize the error which is the sum of the all the absolute differences between the true value and the predicted value. This multi-scale loss function allow us to train our network in frequency-domain, but also using different time resolutions through STFT windows we let the model to be able to keep some time-domain information during training. We also took care of adding the Kullback-Leibler (KL) divergence to approximate our latent space to a normal distribution and scale it by a $\lambda \in \mathbb{R}$ factor to be able to penalize this loss.

$$L_{total} = \sum_i l_{1,i}^{left} + \sum_i l_{1,i}^{right} + \lambda \cdot \mathcal{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{15}$$

3.4 Training

We train the model using Adam optimizer with a batch size of 64. The learning rate was set to 0.0001, instead of using any learning rate schedule like inverse time or exponential decay that demonstrated in early experiments a slower convergence. We train the model using the 512 samples given by each HRIR channel at 44.1 kHz, and the dataset was split into 610 samples per channel (1220 total samples) and a 100 samples for validation. We make sure to shuffle the dataset before doing any splitting and feeding into a model. Finally, we train our models for 500 epochs.

4 RESULTS & EVALUATION

We compare binaural head related impulse response synthesized by the proposed model with the ground-truth binaural recordings given by the MIT HRTF measurements [20]. In order to comprehensively analyze our system, it would be beneficial to conduct both quantitative and perceptual evaluations. For the perceptual evaluation, we can reconstruct or generate a given HRIR pair from a specific spatial location and apply them to various acoustic stimuli such as noise with different color characteristics or a sine sweep. Subsequently, this spatial audio is presented to users through headphones. Like in [9], three main aspects could be considered in the evaluation: naturalness of the signal, spatialization quality, and similarity of the synthesized binaural audio to the actual binaural recording. Naturalness evaluation aims to quantify the presence of artifacts or distortion in the generated binaural signals. Spatialization measures how well the path of the virtual sound source matches the path rendered in the binaural audio. Similarity aims to measure how well the generated binaural signals (acoustic stimuli convolved with the HRIR pair) resembles the binaural recording during playback. We have set aside the perceptual evaluations for consideration in future work.

4.1 Evaluation metric

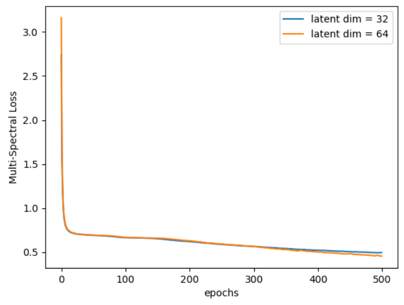
For a quantitative evaluation, log-spectral distortion (LSD) was employed [22]:

$$LSD(\mathbf{H}, \hat{\mathbf{H}}) = \sqrt{\frac{1}{LK} \sum_{\phi,\theta} \sum_{k} (20 \log_{10} | \frac{\mathbf{H}(k,\phi,\theta)}{\hat{\mathbf{H}}(k,\phi,\theta)} |)^2} \tag{16}$$

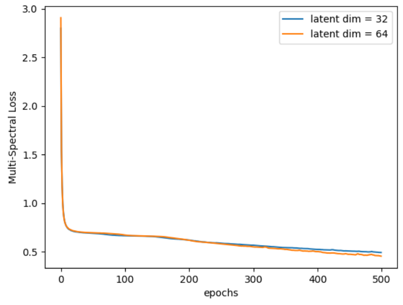
where k denotes the frequency index. L and K are the numbers of spatial locations and frequency bins, respectively. $\mathbf{H}(k,\phi,\theta)$ and $\hat{\mathbf{H}}(k,\phi,\theta)$ indicate the linear-scale magnitude of the ground-truth HRTF and the predicted HRTF of the k-th frequency bin ($\mathbf{H}(\mathbf{k},\phi,\theta) = \mathcal{F}\{\mathbf{h}(n,\phi,\theta)\}$), respectively, at the direction of azimuth angle θ and elevation angle ϕ. This metric facilitated a comprehensive evaluation across all spherical positions, gauging the fidelity of the reconstructed/generated spectrum in comparison to the original data.

4.2 HRIR reconstruction with the learned model

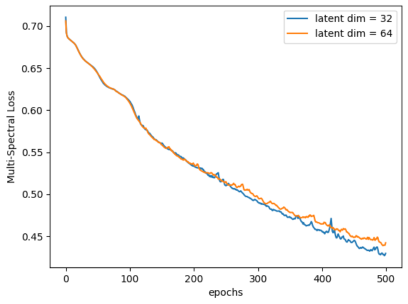
The performance of the three trained models (Model 1, Model 2 and Model 3), along with the corresponding loss functions and epochs, can be visualized in Figure 6.



(a) Model 1: Encoder[1] + Decoder[1]



(b) Model 2: Encoder[2] + Decoder[1]



(c) Model 3: Encoder[2] + TCN Decoder[2]

Figure 6: Loss function progress across epochs

As demonstrated earlier, configuring the latent dimension to 64 channels yields a slightly superior performance compared to the configuration with 32 dimensions. Additionally, it is evident that further model iterations over hundreds of epochs could potentially lead to a better convergence.

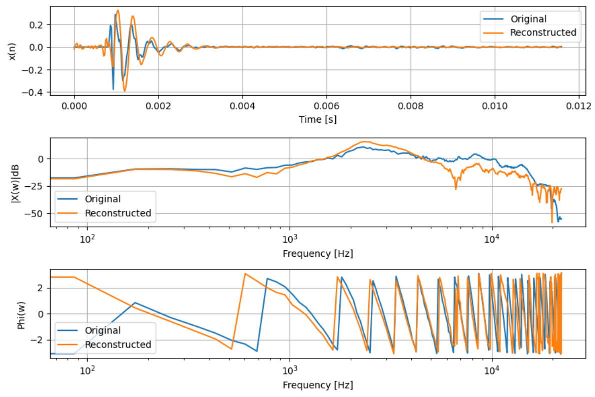
To validate our model within the data distribution, we used the 64-dim latent space for the three models. Table 1 summarizes the results obtained for the loss function convergence and the proposed metric. For the purpose of the loss function, we utilized a training set consisting of 610 samples. For metric computation, we employed the validation set of 100 samples, along with the 710 samples in the dataset.

Model	$Loss_{l_1}$	LSD_{100} left [dB]	LSD_{100} right [dB]	LSD_{710} left [dB]	LSD_{710} right [dB]
Model 1	0.45509	1.51064	2.58548	1.69430	1.93377
Model 2	0.45363	2.25460	2.33946	1.88247	1.64510
Model 3	0.44222	2.58148	4.94418	1.49232	1.52399

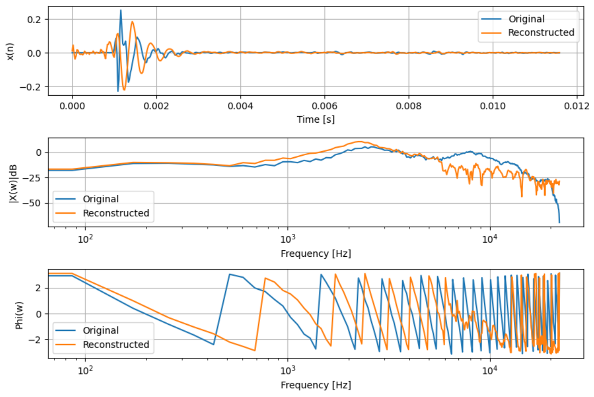
Table 1: Validation table

As depicted in Table-1, Model 1 produced superior results for the validation set in the left channel and the reconstruction loss almost matches the other two losses. Therefore, we will adhere to this model. However, prior to doing so, we will present some HRIRs that have been reconstructed from the validation set in Figure-7. The sample was extracted from the validation set at coordinates $\phi = 70°$ and $\theta = 105°$.

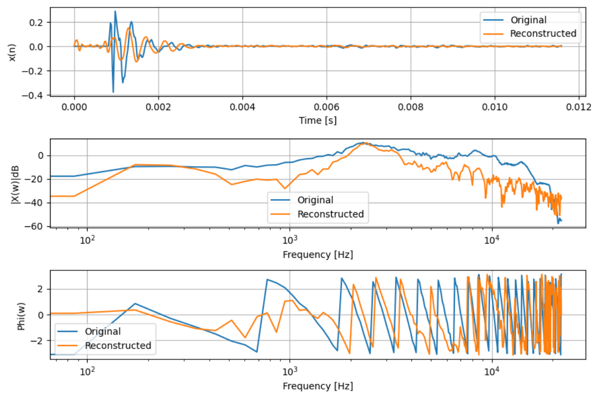
Hence, we deduce that the design models encounter difficulty in reconstructing high-frequency information.



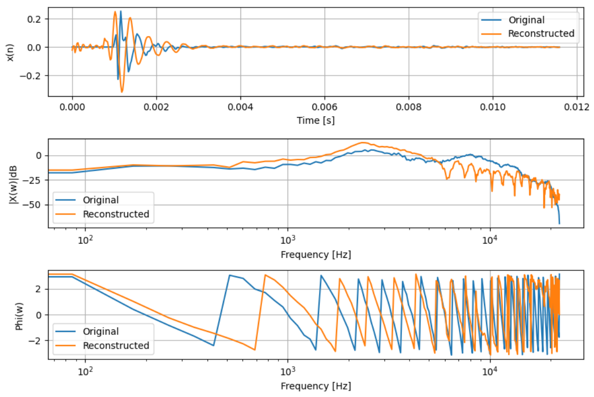
(a) Model 1: Reconstructed $\hat{h}_{left}(n)$



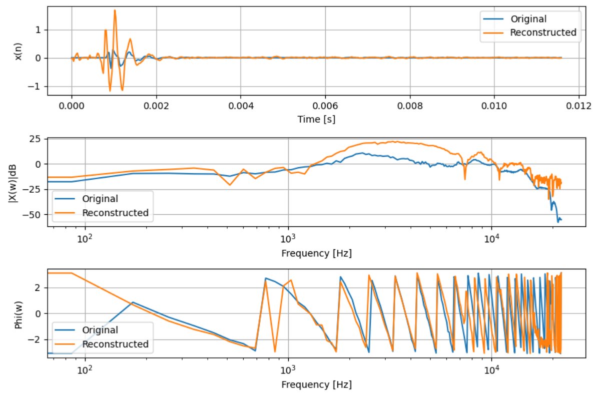
(b) Model 1: Reconstructed $\hat{h}_{right}(n)$



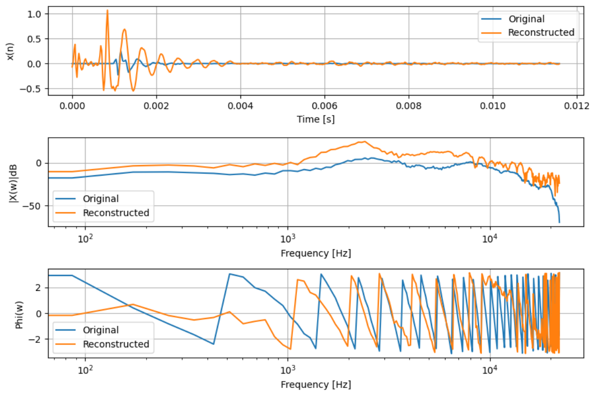
(c) Model 2: Reconstructed $\hat{h}_{left}(n)$



(d) Model 2: Reconstructed $\hat{h}_{right}(n)$



(e) Model 3: Reconstructed $\hat{h}_{left}(n)$



(f) Model 3: Reconstructed $\hat{h}_{right}(n)$

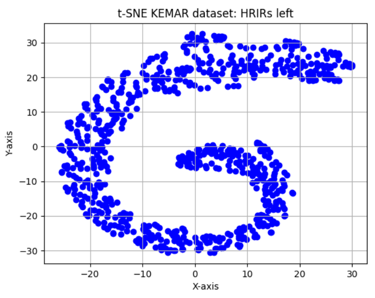
Figure 7: Models performance for sample extracted at coordinates $\phi = 70^{\circ}$ and $\theta = 105^{\circ}$

4.3 Conditional generation from randomly observed locations: spatial interpolation

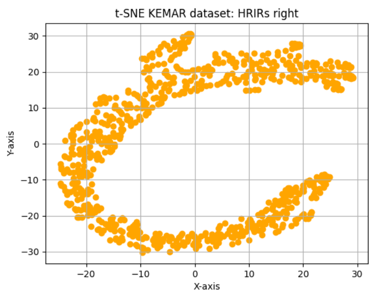
We aim to condition our model to accept input source positions, allowing it to generate the corresponding head related impulse response pair for the specified 3D position of the sound source: $\mathbf{p}_s$. To do so, two ways were proposed in section 2.2.3. Both the Encoder and Decoder models can take the spatial 3-D location of the sound-emitting source as input, which may be provided in either Cartesian or Spherical coordinates. We trained using spherical coordinates: $\mathbf{p}_s = (r_s, \theta_s, \phi_s)$, where r represents the radius as documented in [20], and θ and ϕ denote the azimuth and elevation angles in radians. While it was possible to conduct training using Cartesian coordinates, we have opted to reserve this for future exploration. However, we speculate that employing this coordinate system may yield additional benefits, given the sparsity observed in various positions.

4.3.1 Generation by sampling from latent space

Figure-8 illustrates the behaviour of the data by using the t-SNE data compression algorithm in the real data given by MIT's dataset. Our models encode a latent space distribution for each channel given an input vector (HRIRs samples) and a condition label (source spatial location). The distribution closely resembles that of a normal Gaussian distribution. To visualize the encoded latent space, we can utilize the t-SNE algorithm, as illustrated in Figure 9. It becomes evident that the distribution of all encoded impulse responses closely approximates a bivariate normal distribution. This aligns with our expectations, given the VAE configuration and the use of the reparametrization trick. This information is not particularly beneficial, as the encoded $\mathbf{z}_{left}$ and $\mathbf{z}_{right}$ values for VAE do not contain useful information about which HRIR is produced. However, the essential question remains: What specific information do they contain?



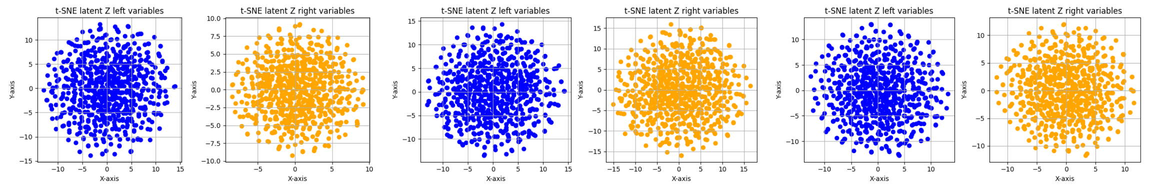
(a) left HRIRs



(b) right HRIRs

Figure 8: t-SNE visualization of MIT's KEMAR dataset

We will begin by taking a location that is not in MIT's dataset. We picked: $\mathbf{p}_s = (1.4, 95°, 65°)$. This three-dimensional point denotes a sound-emitting source located 1.4 meters away, with an azimuth of 95 degrees and an elevation of 65 degrees. Since the radius is the same for all directions, we are only looking for sensitivity by changing ϕ and θ. Then the condition vector in radians becomes: $\mathbf{p}_s = (1.4, 1.65806279, 1.13446401)$



(a) $(\mathbf{z_{1,left}}, \mathbf{z_{1,right}})$ (b) $(\mathbf{z_{2,left}}, \mathbf{z_{2,right}})$ (c) $(\mathbf{z_{3,left}}, \mathbf{z_{3,right}})$

Figure 9: t-SNE visualization of the 6 encoded $\mathbf{z}$ variables

Then, we will select a pre-existing data point close to the desired point for generating new data within the dataset. From this, we will extract a mean vector μ and a standard deviation vector σ using the encoder model. The selected point is denoted as $\mathbf{p}_s^c = (1.4, 90°, 60°)$. Subsequently, we will exclusively extract the HRIR pair from this location and input it into the encoder alongside (appending) the desired location $\mathbf{p}_s$ from which we aim to generate new data. In Figure-10, we can observe the data that it was generated by the three different models.

An alternate approach involves examining what the Decoder has learned. To accomplish this, we concatenate the condition label $\mathbf{p}_s$ with white Gaussian noise $\eta \sim \mathcal{N}(0,1) \in \mathbb{R}^{64}$ (64-dim latent space), and subsequently input this combined signal into the Decoder. As white noise has all frequencies, and our models exhibit limitations in reconstructing high frequencies, examining the models' response to this stimulus becomes valuable, particularly, by observing whether our models can indeed generate new data from pure noise. Figure-11 depicts the result of this experiment where we see that the Decoder model has learned to reconstruct and generate new impulse response from random noise, mainly the proposed Decoder in Model 1.

To validate the generation of this HRIR pair, one of the methods outlined in section 1 could be employed. We found a MATLAB script that performs the interpolation using barycentric weights of four HRTF/HRIR measurements forming a tetrahedron that encloses the desired source position. We will postpone this task for future work, concentrating on refining our model and enhancing training methods.

4.3.2 Latent space interpolation

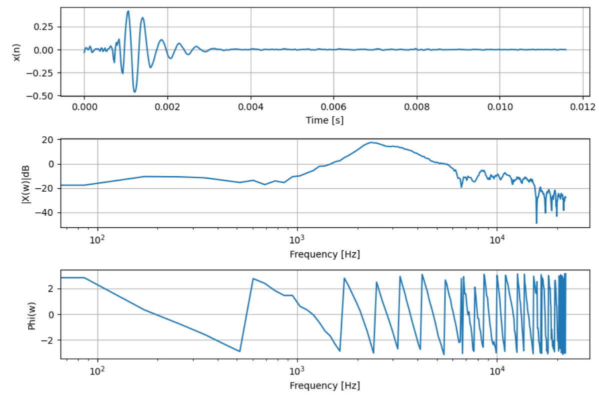
We train Model 1 (given that it was the one with superior performance) on the entire KEMAR dataset to learn the latent $\mathbf{z}_{left}$ and $\mathbf{z}_{right}$ space over the HRIRs. We sample two latent codes for both channels $\mathbf{z}_{1,left}$, $\mathbf{z}_{1,right}$, $\mathbf{z}_{2,left}$ and $\mathbf{z}_{2,right}$ of two HRIRs pair locations. The latent codes are computed through the Encoder. Then we linearly combine the two latent codes by:

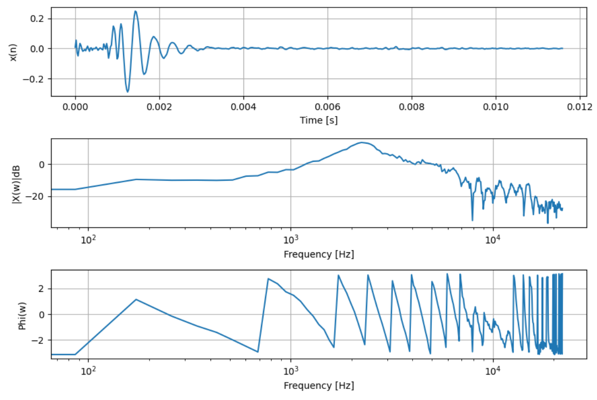
$$\mathbf{z}_{\mathbf{t},left} = (1-t)\cdot \mathbf{z}_{1,left} + t\cdot \mathbf{z}_{2,left} \qquad \mathbf{z}_{\mathbf{t},right} = (1-t)\cdot \mathbf{z}_{1,right} + t\cdot \mathbf{z}_{2,right} \tag{17}$$

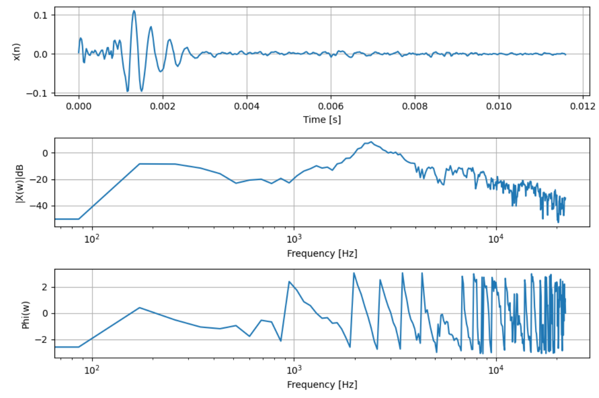
as well as the source position:

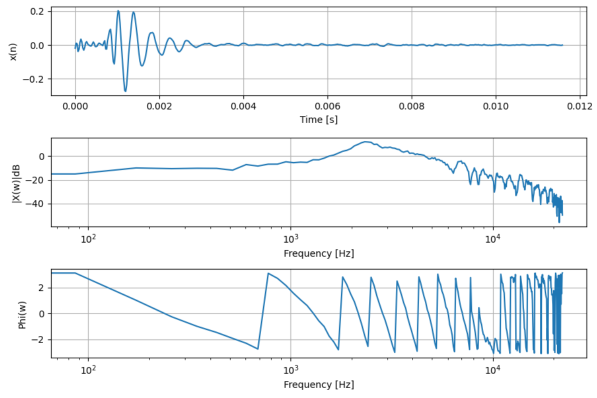
$$\mathbf{P}_{\mathbf{t},s} = (1-t)\cdot \mathbf{p}_{1,s} + t\cdot \mathbf{p}_{2,s} \tag{18}$$

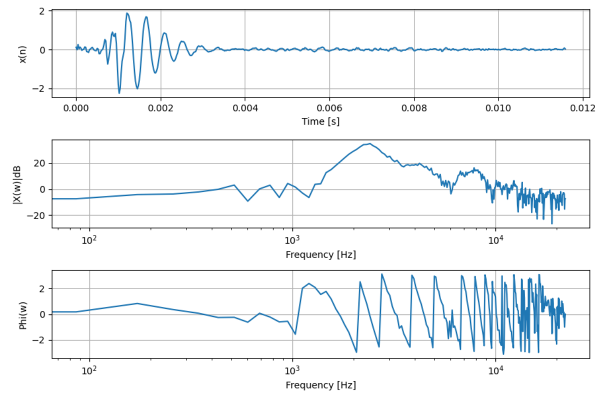
where t is the weight parameter. We synthesize new HRIR samples with the VAE Decoder and their respective source locations $\mathbf{p}_{s1}$ and $\mathbf{p}_{s2}$. In our experiment, we defined $\mathbf{p}_{s1} = (1.4, 90°, 60°)$ and $\mathbf{p}_{s2} = (1.4, 105°, 70°)$ as the source positions, as these locations are present in the dataset. The outcomes are visible in Figure- **??** and in Figure- 13.


(a) Model 1: $\mathbf{h_{1,left}}$


(b) Model 1: $\mathbf{h_{1,right}}$


(c) Model 2: $\mathbf{h_{2,left}}$


(d) Model 2: $\mathbf{h_{2,right}}$


(e) Model 3: $\mathbf{h_{3,left}}$

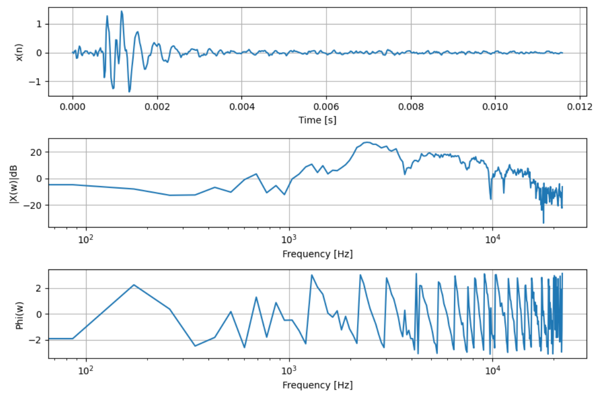
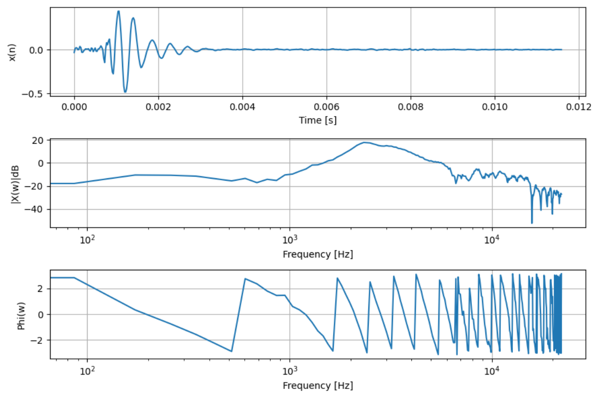
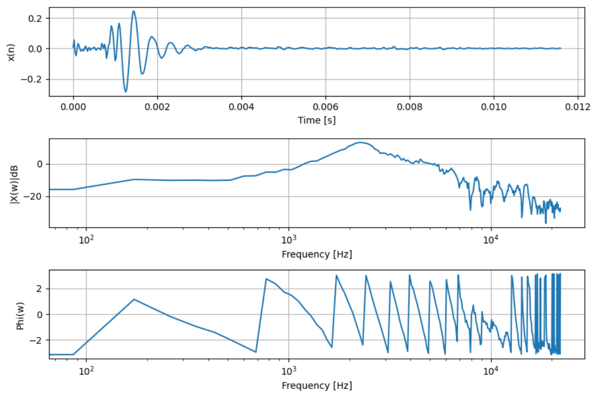

(f) Model 3: $\mathbf{h_{3,right}}$

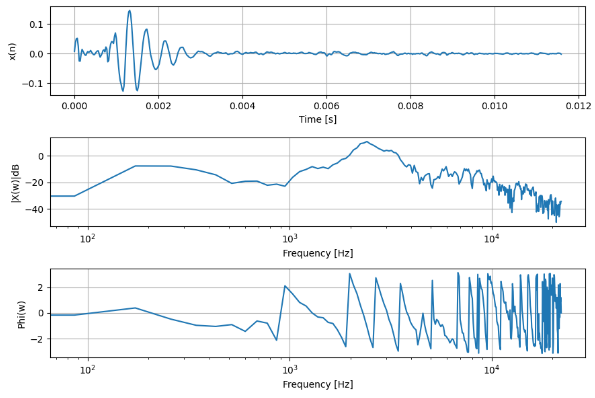
Figure 10: HRIRs generation at $\mathbf{p}_s = (1.4, 95°, 65°)$ using the proposed models



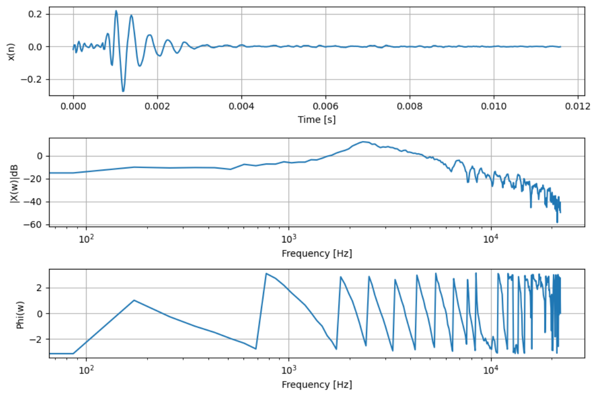
(a) Model 1: $\mathbf{h_{1,left}}$



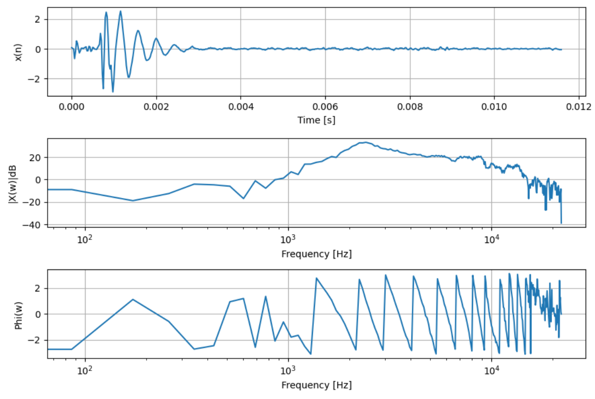
(b) Model 1: $\mathbf{h_{1,right}}$



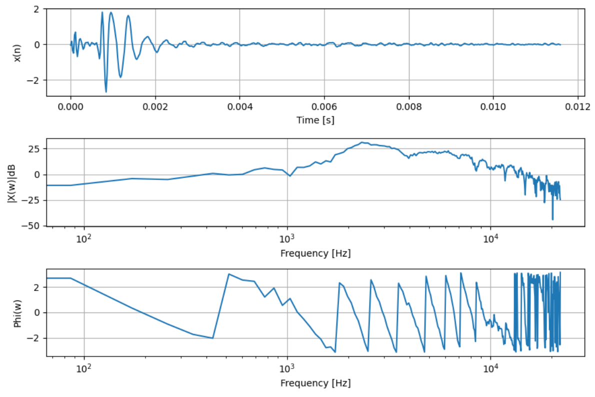
(c) Model 2: $\mathbf{h_{2,left}}$



(d) Model 2: $\mathbf{h_{2,right}}$

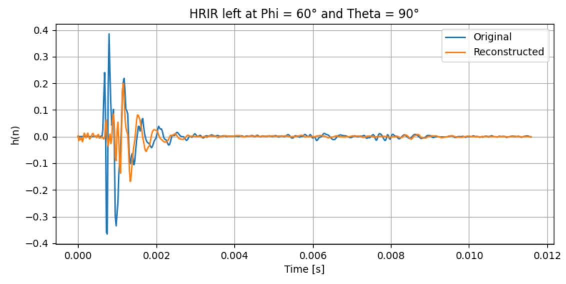


(e) Model 3: $\mathbf{h_{3,left}}$

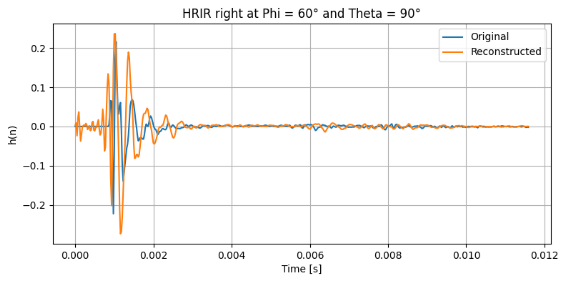


(f) Model 3: $\mathbf{h_{3,right}}$

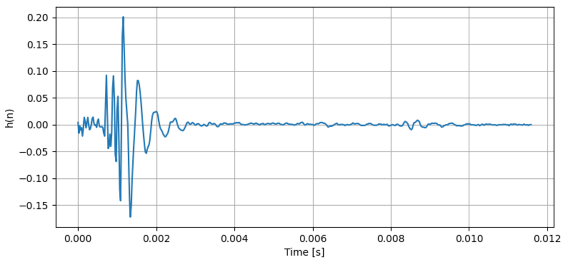
Figure 11: HRIRs generation at $\mathbf{p}_s = (1.4, 95°, 65°)$ using only the Decoder with input $\eta \sim \mathcal{N}(0, 1)$



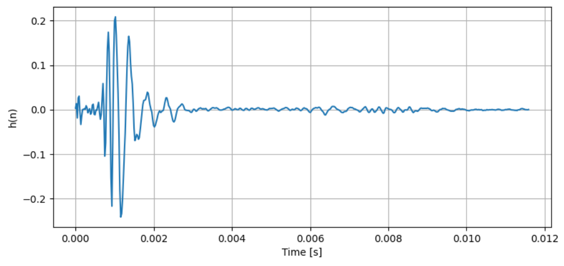
(a) Left channel: $t = 0$



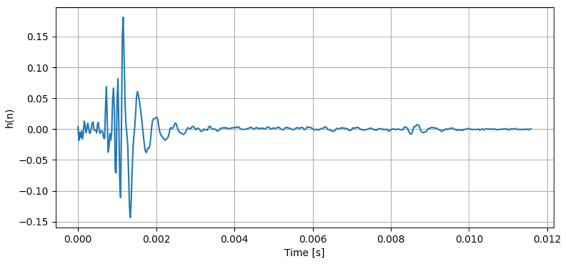
(b) Right channel: $t = 0$



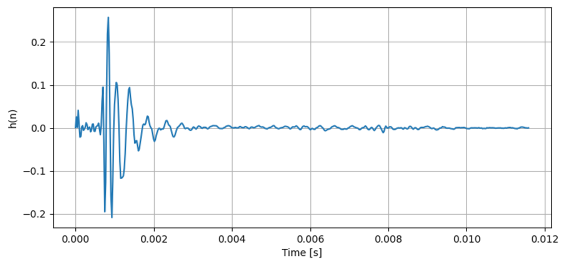
(c) Left channel: $t = 0.25$



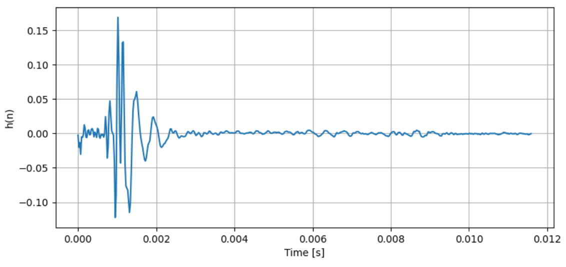
(d) Right channel: $t = 0.25$



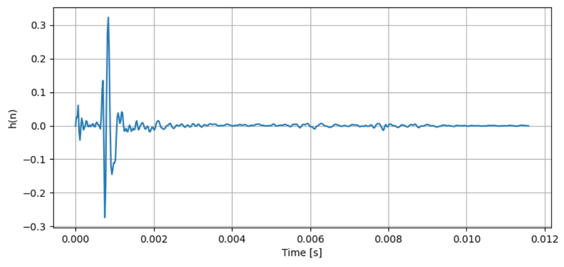
(e) Left channel: $t = 0.5$



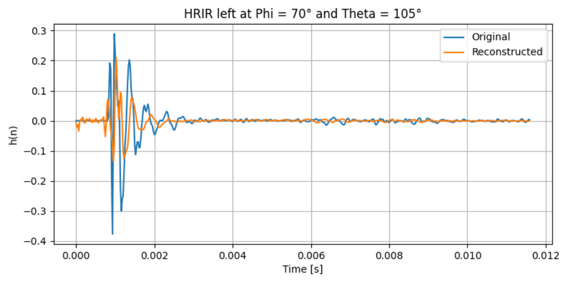
(f) Right channel: $t = 0.5$



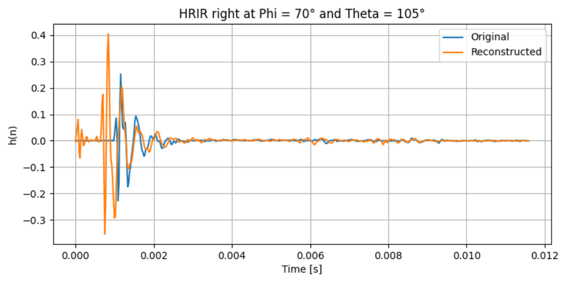
(g) Left channel: $t = 0.75$



(h) Right channel: $t = 0.75$



(a) Left channel: $t = 1$



(b) Right channel: $t = 1$

Figure 13: HRIRs latent space interpolation

5 CONCLUSION

In this, work we explore the use of neural networks to interpolate spatially the HRIRs recordings that come in MIT's dataset. We proposed differentiable representation for HRIRs with three different VAEs designs that attempt to follow most of the bibliography that was read. While none of the VAE models produced outstanding results, it is noteworthy that they were able to successfully reconstruct and generate new data for low to mid-frequency information. Additionally, the developed Decoder demonstrated the capability to generate new data from Gaussian noise. This accomplishment underscores one of the primary capabilities essential for a generative audio model. In our pursuit of improvement, we will explore new methods and techniques to enhance the model's capabilities, aiming to achieve the synthesis of high-fidelity Head-Related Impulse Responses across continuous sampling positions on the spherical surfaces representing sound-emitting sources. There are still crucial aspects to validate when generating Head-Related Impulse Responses (HRIRs), such as Interaural Time Differences (ITDs) or Interaural Level Differences (ILDs). These aspects will be one of the focus of further work.

6 Acknowledgment

I extend my thanks to my mentor Professor Rodrigo Cádiz who was really patient in waiting for my progress in this work and also gave me this project idea that I have been working on since August 2021. I met with Rodrigo about seven times over the span of three semesters, and with each encounter, I gained progressively more knowledge about topics related to signal processing, machine learning, and neural networks. He even introduced me to generative models which is sort of the "engine" of this work. Despite of the obtained results, I'm really happy of how this undergraduate research (iPre) turned out because I improved exponentially my programming skills, I was able to apply signal processing theory, which afforded me a comprehensive understanding of certain aspects within it, I had the opportunity to develop a machine learning model without taking any related courses in college, and I greatly enjoyed the experience. A big shout out to Professor Rodrigo for his guidance and support throughout the research. I would recommend taking an iPre with him as a mentor, especially suited for those already acquainted with the majority of topics discussed in this report.

References

[1] C. I Cheng and G. H Wakefield, "Introduction to head-related transfer functions (hrtfs): Representations of hrtfs in time, frequency, and space," *Journal of the Audio Engineering Society*, vol. 49, no. 4, pp. 231–249, 2001.

[2] C. Hendrix and W. Barfield, "The sense of presence within auditory virtual environments," *Presence: Teleoperators and Virtual Environments*, vol. 5, no. 3, pp. 290–301, 1996.

[3] Brian F. G. Katz, "Boundary element method calculation of individual head-related transfer function," *The Journal of the Acoustical Society of America*, vol. 110, no. 5, 2001.

[4] H. Gamper, "Head-related transfer function interpolation in azimuth, elevation, and distance," *The Journal of the Acoustical Society of America*, vol. 134, pp. 547–553, 2013.

[5] Z. Ben-Hur, D. L Alon, R. Mehra, and B. Rafaely, "Efficient representation and sparse sampling of head-related transfer functions using phase-correction based on ear alignment," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 12, pp. 2249–2262, 2019.

[6] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, I Sutskever, "Jukebox: A Generative Model for Music," *arXiv:2005.00341v1*, 2020.

[7] A. Van Den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, K. Kavukcouglu, "WaveNet: A Generative Model For Raw Audio," *arXiv:1609.03499v2*, 2016.

[8] A. Ratnarajah, S. X. Zhang, M. Yu, Z. Tang, D. Manocha, D. Yu, "Fast RIR: Fast Neural Diffuse Room Impulse Response Generator," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2022.

[9] I. D. Gebru, D. Marković, A. Richard, S. Krenn, G. A. Butler, F. De la Torre, Y. Sheikh, "Implicit HRTF Modeling Using Temporal Convolution Networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2021.

[10] A. Richard, D. Markovic, I. D. Gebru, S. Krenn, G. Butler, F. de la Torre, Y. Sheikh, "Neural Synthesis From Binaural Speech From Mono Audio," in *International Conference on Learning Representations (ICLR)*, 2021.

[11] J. Engel, L. Hantrakul, C. Gu, A. Roberts, "DDSP: Differentiable Digital Signal Processing," *arXiv preprint arXiv:2001.04643*, 2020.

[12] X. Wang, S. Takaki, J. Yamagishi, "Neural source-filter waveform models for statistical parametric speech synthesis," *arXiv:1904.12088v2*, 2019.

[13] A. Richard, P. Dodds, V. Krishna Ithapu, "Deep Iimpulse Responses: Estimating and Parametrizing Filters With Deep Networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).* IEEE, 2022.

[14] P. Esling, N. Masuda, Ad. Bardet, R. Despres, A. Chemla, R. Santos, "Universal Audio Synthesizer Control With Normalizing Flows," *arXiv:1907.00971v1*, 2019.

[15] O. Ivanov, M. Figurnov, D. Vetrov, "Varational Autoencoder With Arbitrary Conditioning," in *International Conference on Learning Representations (ICLR)*, 2019.

[16] He. Purwins, B. Li, T. Virtanen, J. Schlüter , S. Y. Chang, T. Sainath, "Deep Learning for Audio Signal Processing," *IEEE Journal of Selected Topics In Signal Processing*, vol. 13, no. 2, 2019.

[17] Y. Luo, N. Mesgarani, "Conv-TasNet: Surpassing Ideal Time–Frequency Magnitude Masking for Speech Separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 8, 2019.

[18] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, G. Wetzstein, "Implicit Neural Representations with Periodic Activation Functions," *arXiv:2006.09661v1*, 2020.

[19] C. Roads, "Cyclic Convolution Matrix," *Center for Computer Research in Music and Acoustics (CCRMA)*, Stanford University, (n.d.). `https://ccrma.stanford.edu/~jos/fp/Cyclic_Convolution_Matrix.html`

[20] MIT Media Lab, "KEMAR - Knowles Electronics Mannequin for Acoustic Research," *MIT Media Lab Sound Media Group*, 1994. `https://sound.media.mit.edu/resources/KEMAR.html`

[21] M. D. Burkhard, "Anthropocentric Manikin For Acoustic Research," *Industrial Research Products, Inc*, 1978.

[22] Y.Zhang, Y. Wang, Z. Duan, "HRTF Feild: Unifying Measured HRTF Magnitude Representation With Neural Fields," *personalarXiv: 2210.15196v3*, 2023.