HD-AD: A NEW APPROACH TO AUDIO ATOMIC DECOMPOSITION WITH HYPERDIMENSIONAL COMPUTING

Christian Yost and Philippe Depalle

CIRMMT[†]
McGill University
Montréal, Canada
christian.yost@mail.mcgill.ca | philippe.depalle@mcgill.ca

ABSTRACT

In this paper, we approach the problem of atomic decomposition of audio at the symbolic level of atom parameters through the lens of *hyperdimensional computing* (HDC) – a non-traditional computing paradigm. Existing atomic decomposition algorithms often operate using waveforms from a redundant dictionary of atoms causing them to become increasingly memory/computationally intensive as the signal length grows and/or the atoms become more complicated. We systematically build an atom encoding using *vector function architecture* (VFA), a field of HDC. We train a neural network encoder on synthetic audio signals to generate these encodings and observe that the network can generalize to real recordings. This system, we call Hyperdimensional Atomic Decomposition (HD-AD), avoids time-domain correlations all together. Because HD-AD scales with the sparsity of the signal, rather than its length in time, atomic decompositions are often produced much faster than real-time.

1. INTRODUCTION

Atomic decomposition is a powerful tool for audio analysis/synthesis [1], coding [2], and transformation [3]. The atomic decomposition model assumes a linear relationship between a signal $\mathbf{y} \in \mathbb{C}^N$ and a set of M atoms $\mathcal{D}$ – the dictionary – in matrix form $\mathbf{\Phi} \in \mathbb{C}^{N \times M}$, expressed as

$$\mathbf{y} = \mathbf{\Phi}\mathbf{x} + \epsilon \tag{1}$$

where $\mathbf{x} \in \mathbb{C}^M$ is a vector of weights describing the contribution of each atom to $\mathbf{y}$ and ϵ is the noise/error term [1]. In audio, atoms are waveforms typically parameterized by a prototype, such as a damped sine wave for example. It is desirable to design such atoms so that audio from a wide range of signals can be represented by only a few of them, i.e. *sparsely*, such that $\mathbf{x}$ has only a few non-zero elements.

In order to sparsely represent a wide variety of audio structures, atom prototypes with many parameters are required. Thus sparsity promoting atoms require storing many different combinations of parameters. Because these prototypes are typically of infinite duration (e.g. damped sine wave), they cannot take advantage of fast algorithms, such as the short time Fourier Transform (STFT). Furthermore, for longer signals the dictionary must be updated with atoms which fully extend into the new time. Thus $\mathcal{D}$ must be recomputed for each signal, and can require a large amount of memory to store, as well as result in long wait times for a decomposition to be computed [4]. Since each position of $\mathbf{x}$ corresponds to an atom in the dictionary, as $\mathcal{D}$ grows, so does $\mathbf{x}$. This limits the use of certain tools which require fixed size data structures, for example many deep neural networks (DNN).

An alternative to storing the atomic information in $\mathbf{x}$, where each coordinate corresponds to a set of parameters, is instead to store the same information by *distributing* it across an entire vector of a different kind. Such types of data representation are the basis of the field of hyperdimensional computing (HDC) – a brain-inspired computing paradigm that encodes values equally across high dimensional vectors [5]. By encoding an atom's information across an entire vector of high but fixed dimension N_{HD}, the vector encoding becomes highly redundant (since $N_{\text{HD}} \gg 1$). The high redundancy introduced by HDC encodings allows multiple vectors to be "superposed" (added) on top of each other, without obscuring each individual encoding. The resulting representation maintains a fixed size of N_{HD} no matter how many encodings are added. Encoding the atom parameters such that only the non-zero elements of $\mathbf{x}$ are represented (i.e. a signal's sparse representation) is the basis of our approach to atomic decomposition of audio.

In this paper we present a fundamentally different approach to atomic decomposition of audio – one which operates at the symbolic level of atoms through the HDC encodings of their parameters. Properties of HDC encodings, such as a fixed size, along with using time-shift frequency-shift invariant atoms – i.e. Weyl Heisenberg (WH) atoms – overcome many of the difficulties encountered by approaching atomic decomposition using waveforms. We show that our atomic decomposition system – called Hyperdimensional Atomic Decomposition (HD-AD) – requires very little memory to run and can often produce atomic decompositions in much faster than real-time – a speed unknown to existing atomic decomposition algorithms with complicated prototypes.

The paper is structured as follows. Section 2 gives an overview of important audio atomic decomposition features and properties. Section 3 reviews the fundamentals of HDC, as well as HDC aspects particular to this project. The new atomic decomposition approach, HD-AD, is detailed in Section 4. Section 5 demonstrates HD-AD in practice. Section 6 discusses how this work can fit into existing atomic decomposition algorithms. Finally, in Section 7 we reflect on the project and give our thoughts on future work in this area. Audio examples can be heard at the companion website.[1]

[†] The Centre for Interdisciplinary Research in Music Media and Technology

[1] https://htmlpreview.github.io/?https://github.com/ChristianYost/HD-AD/blob/main/HDAD.html

2. ATOMIC DECOMPOSITION OF AUDIO

In audio, we often use WH atoms [6] which have the general form

$$\phi[n] = E[n]e^{i2\pi f_c n} \tag{2}$$

where E is the envelope (windowing) function, $0 \leq f_c \leq \frac{1}{2}$ is the center frequency of the atom, and n is discrete time. The particular ϕ is referred to as the *prototype*, and is controlled by a set of parameters $\boldsymbol{\lambda}$, and denoted $\phi_{\boldsymbol{\lambda}}$. Designing ϕ is often informed by which class of signals it is to decompose (e.g. plucked instrument, voice, etc.) Decompositions with sparsity promoting atoms are usually more *meaningful*, in that each atom reveals more about the signal structure. The shape of $E[n]$ has a large effect on ϕ and can be broadly divided into two categories – *symmetric* and *asymmetric*. In the case of audio, sound features are generally asymmetric, therefore decomposition via asymmetric atoms is generally more meaningful.

Asymmetric atoms are characterized by an asymmetric envelope E, where, in general, the portion before the envelope's maximum (the *attack*) is shorter than the portion after the envelope maximum (the *decay*). The damped sinusoid (DS) is an essential asymmetric atom in atomic modelling of audio given its relationship to a vibrating mode of a resonant structure [7]. An envelope damping parameter is introduced into the atom prototype which allows control over its decay characteristics. The amplitude envelope of a damped sine wave is

$$E_{\text{DS}}[n] = e^{-\alpha n}u[n] \tag{3}$$

where $\alpha \in \mathbb{R}_{\geq 0}$ is the damping factor and u is the unit step function. Many asymmetric atoms are modulations of a DS, such as a gammatone (GT) [8] and the formant-wave-function (FOF) [9], however neither was designed specifically for the sparse decomposition of audio.

The ramped exponentially damped sinusoid (REDS) is an asymmetric atom designed specifically for sparse atomic decomposition of audio [10]. A REDS atom can be seen as a DS modulated by an attack envelope, A_{REDS}, where

$$A_{\text{REDS}}[\beta, p; n] = (1 - e^{-\beta n})^p \tag{4}$$

The attack parameter β allows for precise control over its attack characteristics. The polynomial order p smooths the onset of the attack envelope and can be chosen to allow REDS to approximate other asymmetric atoms – an advantage the REDS prototype has over other atoms.

2.1. Selecting Atom Parameters

Many atomic decomposition algorithms rely on a dictionary of waveforms, perhaps the most well-known method is Matching Pursuit (MP) [1]. Waveform dictionary approaches rely on computing many correlations between a time signal and the dictionary of atoms. Since signals are generally of different length, when decomposing with asymmetric atoms the waveform dictionaries must be recomputed for each signal. For short, symmetric atoms, the STFT can be used to save memory and computation [4]. However, for sparser decompositions requiring more complicated atom prototypes, few solutions exist ([11]) which do not result in a large memory footprint and a large amount of computation.

These observations inspired our investigation into encodings which maintain a fixed size in order to overcome the burden of long signals, and allow for decomposition with complicated atom prototypes using a modest amount of memory.

3. HYPERDIMENSIONAL COMPUTING

Hyperdimensional computing (HDC) [5] is a computing paradigm which maps (encodes) data (atom parameters in this paper) to high dimensional vector space of size N_{HD}, typically between 10^3 and 10^4, in the form of *hypervectors*. Because the HDC mappings are of high dimension, hypervectors are seen to distribute their encoding equally at each position, making them highly redundant. High redundancy means highly robust to noise. HDC can be used to encode non-numerical [12] or real values [13] into hypervectors. Hypervectors can be combined to represent single or composite entities. A field of HDC, vector symbolic architecture (VSA) [14], pairs a vector representation space with a set of simple algebraic operations to form a "ring-like" structure. Recently in [15], the VSA framework is generalized to a vector function architecture (VFA), where a *similarity kernel* is used as a "generator" of sorts for the space. This allows for a systematic encoding where certain relationships in the source domain are preserved in the encoding, a property known as locality preserving encoding (LPE) [15].

Notably, the elements of the hypervectors belonging to a VFA are chosen randomly from a distribution, and independently from each other. Given this inherent randomness, VFA encodings are structured around relationships *between* different states of the HD space, rather than any particular state of the space itself.

3.1. Building VFA Data Structures

The basic operations for building data structures with hypervectors are *binding* [5] and *superposition* [16]. Importantly, these operations can be chosen so that the size of the vectors being computed does not change [14].

3.1.1. Binding

The binding operation (generically denoted $\circ$), analogous to multiplication, is used to associate two or more hypervectors with each other. Binding can be used to generate a new hypervector by associating two others, such as $\mathbf{u} \circ \mathbf{v} = \mathbf{s}$, as well as encode real numbers through fractional power encoding (FPE) [17]. Common binding operations include the Hadamard product ($\odot$) [14] and circular convolution ($\circledast$) [18]:

$$\begin{aligned}(\mathbf{u} \odot \mathbf{v})_j &= u_j \cdot v_j \\ (\mathbf{u} \circledast \mathbf{v})_j &= \mathcal{F}^{-1}\big(\mathcal{F}(\mathbf{u}) \odot \mathcal{F}(\mathbf{v})\big)_j\end{aligned} \tag{5}$$

where $\mathcal{F}$ denotes the Fourier Transform and u_j is the j^{th} element of $\mathbf{u}$. FPE starts with self-binding, which describes binding a hypervector with itself k times as a way to encode integers.

$$\mathbf{u}(k) = (\mathbf{u})^{(\circ k)} = \mathbf{u} \underbrace{\circ \ldots \circ}_{k-1 \text{ times}} \mathbf{u} \tag{6}$$

The hypervector $\mathbf{u}$ is referred to as the *base* hypervector. For Hadamard product and circular convolution binding these are

$$\begin{aligned}(\mathbf{u})_j^{(\odot k)} &= (\mathbf{u} \underbrace{\odot \ldots \odot}_{k-1 \text{ times}} \mathbf{u})_j = u_j^k \\ (\mathbf{u})_j^{(\circledast k)} &= (\mathbf{u} \underbrace{\circledast \ldots \circledast}_{k-1 \text{ times}} \mathbf{u})_j = \mathcal{F}^{-1}\left(\mathcal{F}(\mathbf{u})^{(\odot k)}\right)_j\end{aligned} \tag{7}$$

where $\mathcal{F}^{-1}$ denotes the inverse Fourier transform. In both cases, k is encoded by raising the elements of base hypervector $\mathbf{u}$ (or $\mathcal{F}(\mathbf{u})$ for $\circledast$) to the k^{th} power. This is then generalized to encode real numbers r:

$$\begin{aligned}(\mathbf{u})_j^{(\odot r)} &= u_j^r \\ (\mathbf{u})_j^{(\circledast r)} &= \mathcal{F}^{-1}\left(\mathcal{F}(\mathbf{u})^{(\odot r)}\right)_j\end{aligned} \tag{8}$$

Fractional Power Encoding (FPE) encodes real numbers r using a base hypervector $\mathbf{u}$ and a binding operation $\circ$.

$$f_{\text{FPE}} : r \in \mathbb{R} \rightarrow \mathbf{u}(r) = (\mathbf{u})^{(\circ r)} \in \mathbb{C}^{N_{\text{HD}}} \tag{9}$$

Typically base hypervectors are generated for each field or property which is being encoded. To generate base hypervectors, a distribution is randomly sampled. For example, sampling a phasor vector $\boldsymbol{\theta} = [0, 2\pi)^{N_{\text{HD}}}$ and generating $\mathbf{u}$ via $u_j = e^{i\theta_j}$ is a popular method for generating (base) hypervectors. The sampled hypervectors are stored and used for encoding values for their corresponding property via equation (9).

Of note is the set of unitary hypervectors which are norm-preserving: $A_\circ = \{\mathbf{u} : \|\mathbf{u} \circ \mathbf{v}\|_2 = \|\mathbf{v}\|_2 \ \forall \mathbf{u}\}$. Unitary hypervectors under the Hadamard product are $\{\mathbf{v} : |v_j| = 1 \ \forall j\}$, and under circular convolution are $\{\mathbf{v} : |\mathcal{F}(\mathbf{v})_j| = 1 \ \forall j\}$.

3.1.2. Superposition

Superposition ($+$), analogous to addition, is used to "bundle" two or more hypervectors together, such as

$$\mathbf{u} + \mathbf{v} = \mathbf{s} \tag{10}$$

where $s_i = u_i + v_i$. Typically binding is used to associate values of different fields for a single entity (e.g. shape and color of an object); superposition is typically used to represent multiple entities in a single hypervector.

3.1.3. Dot product

The set of VFA hypervectors with a conventional dot product ($\langle \cdot, \cdot \rangle$) makes up a Hilbert space. Thus the distance between two hypervectors belonging to the space can be computed, which is used as a way to compare two hypervectors. Combining hypervectors via binding or superposition has a different effect on the similarity (distance) between the input and output hypervectors [5]. Bundling two hypervectors creates a new hypervector which is *similar* (close) to both input hypervectors. On the other hand, binding two hypervectors creates a new hypervector which is *dissimilar* (far) to (from) either hypervector that went into the binding [5].

This similarity destroying nature of binding applies even when a hypervector is bound with itself. This situation arises for encoding integers, as was discussed in section 3.1.1 in equation (6). Thus when two hypervectors are maximally similar (identical) binding them together still produces a completely dissimilar hypervector. However, for non-integer encodings (i.e. FPE shown in equation (9)), fractional binding produces a hypervector whose similarity with the base hypervector is defined by a *similarity kernel*, K. Similarity kernels have many interesting properties relating to functional analysis [15]. For our purposes, it is important to note that the similarity between the hypervector encodings for two real numbers r_1 and r_2 is translation invariant and given by

$$\langle \mathbf{u}(r_1), \mathbf{u}(r_2) \rangle = K(r_1 - r_2) \tag{11}$$

To reiterate the integer encoding behavior mentioned earlier, when $r_1 - r_2 \in \mathbb{Z}$, then $K(r_1 - r_2) = 0$ when $r_1 \neq r_2$. We will see how this affects our decisions in section 5.

Finally, because binding destroys the similarity between input and output hypervectors, two bound hypervectors which share many but not all of the same inputs will appear distinct from one another. And because superposition preserves the similarity of input and output hypervectors, those two bound hypervectors can be added without obscuring either constituent hypervector in the summed output. The similarity destroying nature of binding means that retrieving the original hypervectors, in other words decoding, is typically a hard combinatorial search problem [12].

4. HYPERDIMENSIONAL ATOMIC DECOMPOSITION

Using the VFA concepts from the previous section, we create atom encodings which are the basis of our atomic decomposition system called Hyperdimensional Atomic Decomposition (HD-AD). HD-AD replaces waveform atomic decomposition algorithms by selecting atoms at the symbolic level of the parameters themselves, whose HD encodings are of fixed length N_{HD}. HD-AD, to the best of our knowledge, is the first audio atomic decomposition system which operates at the symbolic level of the atomic parameters.

4.1. Encoding Atoms

For an atom prototype $\phi_{\boldsymbol{\lambda}}$, who is defined by a vector $\boldsymbol{\lambda}$ of Q parameters, we generate a unitary base hypervector $\mathbf{u}_{\lambda_q}$ for each parameter $\lambda_q \in \boldsymbol{\lambda}$. The vector $\dot{\boldsymbol{\lambda}}$ specifies the particular values for parameters $\boldsymbol{\lambda}$. To encode a specific $\phi_{\dot{\boldsymbol{\lambda}}}$ we encode each $\dot{\lambda}_q \in \dot{\boldsymbol{\lambda}}$ using the corresponding base hypervector for parameter λ_q, $\mathbf{u}_{\lambda_q}$, via the FPE from Eq. (9). The HD vector which encodes the parameters of $\phi_{\dot{\boldsymbol{\lambda}}}$ is built by binding together all $\mathbf{u}_{\lambda_q}(\dot{\lambda}_q)$:

$$\boldsymbol{\Lambda}_\circ(\dot{\boldsymbol{\lambda}}) = \mathbf{u}_{\lambda_1}(\dot{\lambda}_1) \circ \mathbf{u}_{\lambda_2}(\dot{\lambda}_2) \circ \ldots \circ \mathbf{u}_{\lambda_Q}(\dot{\lambda}_Q) \tag{12}$$

In order to scale the atoms by their gain coefficient x (from equation (1)), we bind the atom parameter encoding $\boldsymbol{\Lambda}_\circ(\dot{\boldsymbol{\lambda}})$ with a hypervector encoding of x, $\vec{\mathbf{x}}_{\text{HD}}$. We define the function

$$\gamma_\circ(x) : x \in \mathbb{C} \mapsto \vec{\mathbf{x}}_{\text{HD}} \in \mathbb{C}^{N_{\text{HD}}} \tag{13}$$

which maps an atom's gain coefficient x to the corresponding HD vector $\vec{\mathbf{x}}_\circ$. For Hadamard product and circular convolution binding these are

$$\begin{aligned}\gamma_\odot(x) &= \vec{\mathbf{x}}_\odot \Rightarrow (\vec{\mathbf{x}}_\odot)_j = x \\ \gamma_\circledast(x) &= \vec{\mathbf{x}}_\circledast = \mathcal{F}^{-1}(\vec{\mathbf{x}}_\odot)\end{aligned} \tag{14}$$

Thus, the HDC encoding of atom $x \cdot \phi_{\dot{\boldsymbol{\lambda}}}$ is a hypervector

$$\mathbf{s}_\circ(x, \dot{\boldsymbol{\lambda}}) = \gamma_\circ(x) \circ \boldsymbol{\Lambda}_\circ(\dot{\boldsymbol{\lambda}}) \tag{15}$$

A time-signal which is the linear combinations of time-frequency atoms is encoded by the HD vector which is the superposition of the HDC encodings of their parameters.

$$\mathbf{z} = \sum_k \mathbf{s}_\circ(x_k, \dot{\boldsymbol{\lambda}}_k) \tag{16}$$

For a variety of reasons, we choose a deep neural network (DNN) to be the HDC encoder (see [19] for discussion). Our neural network structure includes 2D convolutional gated linear units (convGLU) [20], max-pooling layers [21], ELU activation functions [22], and fully-connected (FC) layers, all shown in Figure 1. Once encoded, retrieving the atom parameters is a non-trivial

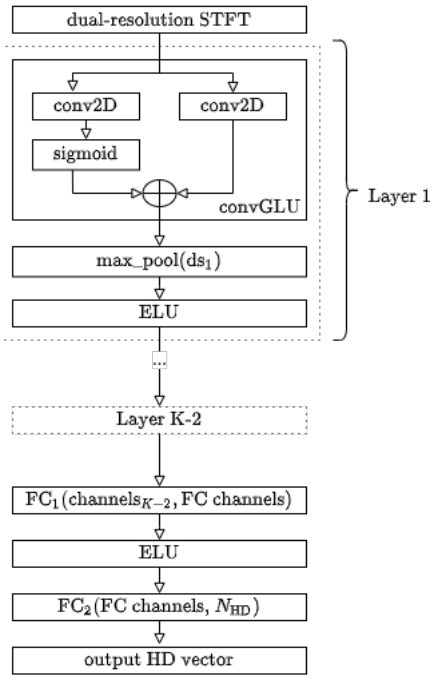

Figure 1: *HD-AD deep neural network encoder structure. The local features (attack/decay shape) are identified by the CNN kernels and translated into global encodings by the FC layers.*

process because of the similarity destroying nature of binding, as discussed in section 3.1.1. Inspired by the connection between signals and VFA hypervectors we sought to apply refinement methods from the signal space to that of the hypervector encodings, namely Newton's method.

4.2. Decoding VFA Hypervectors

Once an atom $x \cdot \phi_{\dot{\boldsymbol{\lambda}}}$ is mapped to the HD vector space, x and $\dot{\boldsymbol{\lambda}}$ must be retrieved by decoding $\mathbf{s}_\circ(x, \dot{\boldsymbol{\lambda}})$. Because only the atoms with non-zero contribution are encoded, or alternatively atoms with zero contribution do not affect the encoded HD atomic decomposition, only non-zero contributing atoms need to be decoded. As a result, HD-AD scales with the sparsity of a signal's atomic decomposition, and not necessarily the signal's length in time.

The standard way to decode hypervector encodings is by comparison to set of hypervectors with known decoding. Because this method relies on a discrete set encodings, when decoding real values only an approximation is returned. In [15] an iterative algorithm based on gradient descent is presented to decode a single real number. In [19] we present a similar algorithm which instead relies on Newton's Method, where the full discussion and details of our decoding algorithm can be found. Using Newton's Method allows for the decoding of multiple encoded real values simultaneously. We refer to this algorithm as "HD Newton's method".

5. EXPERIMENTAL RESULTS

Here we discuss what must be considered when generating atomic HDC encodings in practice, and present the results of performing atomic decomposition with HD-AD. As mentioned in section 3.1.3, the similarity between the encodings of two real numbers r_1 and r_2 depends on $r_1 - r_2$. Because atom parameters have vastly different ranges, encoding their synthesis values ($\dot{\lambda}$) can lead to sub-optimal similarity behavior in the encoding domain. For example, consider the time parameter ($\lambda = \tau$) of an atom prototype $\phi_{\boldsymbol{\lambda}}$, where τ is given in integer samples. The similarity between the encodings of two time values $\dot{\tau}_1$ and $\dot{\tau}_2$ is zero, since $\dot{\tau}_1 - \dot{\tau}_2 \in \mathbb{Z}$. In the other extreme, consider the damping parameter, $\lambda = \alpha$, of a damped sinusoid. The dynamic range of this parameter is often very small, on the order of 10^{-3}. Thus for two damping parameter values $\dot{\alpha}_1$ and $\dot{\alpha}_2$, since their difference is always close to zero, the similarity of their encodings is always close to one. Both of these situations are sub-optimal for training a neural network. On the one hand, any $\dot{\tau}$ estimate by the network has an encoding similarity of 0 with the ground truth unless it is sample accurate. On the other, the worst the network can do for predicting $\dot{\alpha}$ in any situation results in an encoding which is nearly identical to the one it is trying to learn.

Because of this, the atom's synthesis parameters $\dot{\boldsymbol{\lambda}}$ are mapped to their encoded values $\hat{\boldsymbol{\lambda}}$ via a function $\hat{g}_\lambda : \dot{\lambda} \mapsto \hat{\lambda}$. $\hat{g}_\lambda$ rescales the variation of each parameter into a normalized range and allows for better behavior in the encoding domain [19].Once $\hat{\lambda}$ is retrieved by HD Newton's Method, the inverse function $\hat{g}_\lambda^{-1} : \hat{\lambda} \mapsto \dot{\lambda}$ is used to generate the synthesis parameters to produce the time-domain waveform. The details and full discussion of $\hat{g}_\lambda$ for each λ is given in [19].

We train a DNN on synthetic signals which are the mixture of scaled atoms plus noise. Training time for the neural network is typically between $1-2$ days. The parameters for atoms in the mixture are sampled randomly from a given range per parameter. For time and frequency, this is the time-frequency support of the signal. For the envelope parameters, ranges are specified to encompass a range of audio structures, such as partials and transients. The timing results presented were recorded on a personal computer with an Intel Core i5 2.9GHz CPU with 16 GB of RAM.

The inputs to the DNN encoder are time-frequency tiles from a dual-resolution STFT [23] of 16 kHz audio. The high-frequency resolution STFT has an FFT size of 1024; the high-time resolution STFT has an FFT size of 256. Each STFT is upsampled by a factor of 4 in the dimension they have low resolution, in order for points to agree between the two resolutions. Once sub-sampled, the tiles are 64 bins (1000 Hz) "tall" by 128 frames (2.096 s) "wide". The time and frequency parameters of the atoms are encoded relative to the tile they occupy, and then shifted to the global position after decoding. This is possible because of the Weyl Heisenberg nature of atoms discussed in Section 2.

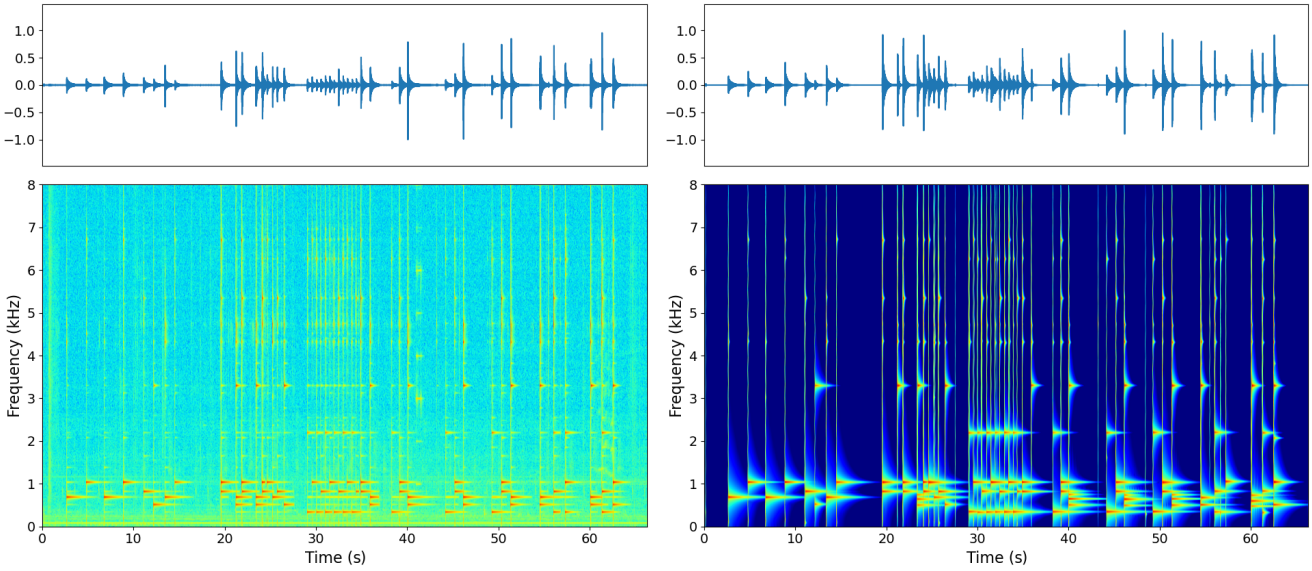

Figure 2: *HD-AD decomposition of a long kalimba recording.* ***(left)*** *Input signal of length* 66.316 *s.* ***(right)*** *HD-AD approximation of* 288 *DS atoms done in* 16.656 $s = 4.236\times$*'s real-time.*

The DNN encoder for these experiments has 7 CNN layers, and 2 FC layers. For CNN layer i there are 2^{i+4} output channels. The time and frequency dimensions are each down-sampled by a factor of 2 at each layer, except the the 7th, where the time dimension is down-sampled by a factor of 2 and the frequency by 1 (no change). The first FC layer outputs 2048 channels, and the final FC layer outputs $N_{\text{HD}} = 1000$ channels.

5.1. DS Atoms

For experiments with damped sinusoid atoms, we create encoding base hypervectors for time, frequency, and damping parameters. After being trained on only synthetic mixtures of atoms the neural network can generate HDC atomic encodings for real audio recordings. Figure 2 shows the HD-AD decomposition of a 66.316 second kalimba recording from freesound.org[2]. HD-AD decomposes the signal into 288 DS atoms in 16.656 seconds, which is $4.236\times$ faster than real-time.

5.2. REDS Atoms

For experiments REDS atoms, we create encoding base hypervectors for time, frequency, attack, and damping parameters. Because HD-AD only returns a signal's sparse decomposition, it scales with a signal's sparsity, rather than its length in time. In order to demonstrate this, Figure 3 shows the HD-AD decomposition of a synthesized toy-piano signal, as well as a decomposition of the same signal but zero-padded on either end by 10 seconds. Because zero-padding a signal does not change its sparsity, HD-AD decomposes both signals (one over $4\times$ longer than the other) in roughly the same amount of time.

[2]"Kalimba" by user "dermotte" (`https://freesound.org/s/244025/`) licensed under CCBYNC 3.0

5.3. Vibrato REDS Atoms

We extend REDS atoms to include vibrato behavior – called as vibrato REDS (vREDS) – by introducing parameters for frequency modulation and vibrato amplitude. Being able to quickly decompose signals with such a complicated prototype demonstrates the full range of capabilities of our proposed method. The argument to the complex exponential becomes

$$\begin{aligned}&\arg(n,\tau,f_c,s,f_m) = \\ &(n-\tau)2\pi f_c + \frac{s}{2\pi f_m}\sin\big(2\pi f_m(n-\tau)\big)\end{aligned} \tag{17}$$

where s is the vibrato amplitude and f_m is the frequency modulation, or vibrato rate. The full Vibrato REDS (vREDS) atom prototype is

$$\begin{aligned}&\phi(n,\tau,f_c,\alpha,\beta,p,s,f_m) = \\ &A_{\text{REDS}}[\beta,p;n]E_{\text{DS}}[\alpha;n]e^{i\cdot\arg(n,\tau,f_c,s,f_m)}\end{aligned} \tag{18}$$

In Figure 4 we demonstrate that creating more complicated atom prototypes, which naturally introduce more parameters, does not prohibit HD-AD from generating a decomposition in a timely manner. This is not the case with MP or many of its derivatives.
Table 1 compares the memory required to store the HD-AD dictionary and a waveform dictionary for the examples shown in Figure 2 and Figure 4. HD-AD requires many orders of magnitude less memory, and produces an atomic decomposition in less time than the signal length.

5.4. Sound Processing Example

Audio processing is an attractive application of atomic decomposition, which allows for high quality filtering, prototype substitution, and atom parameters' manipulation amongst others. However, for decompositions produced by certain prototypes, manipulation of

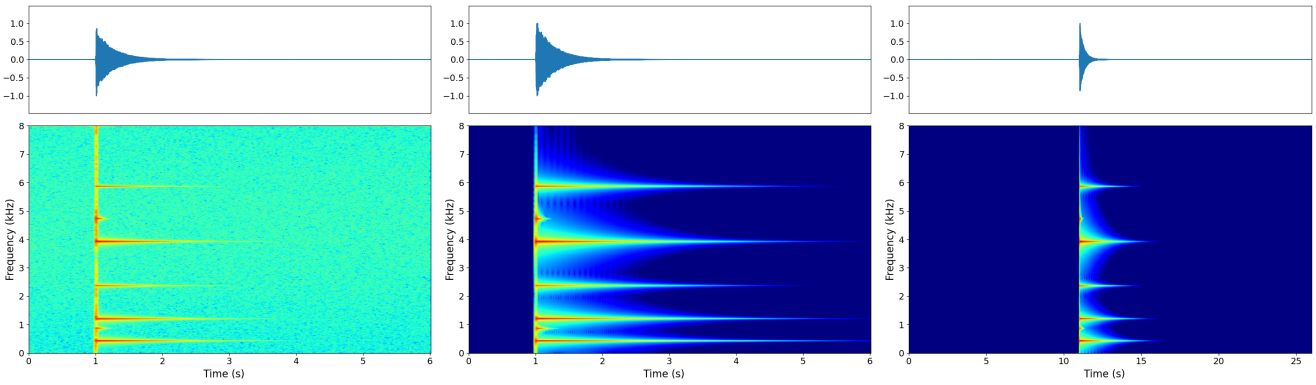

Figure 3: ***(left)*** *A synthesized toy-piano note.* ***(middle)*** *HD-AD decomposes the signal into* 7 *REDS atoms in* $2.384\ s = 2.518\times$*'s real-time.* ***(right)*** *HD-AD decomposition of zero-padding the signal in (left) by 10 seconds on each end. HD-AD decomposes the signal into* 7 *REDS atoms in* $6.904\ s = 3.767\times$*'s real-time. HD-AD scales with the signal's sparsity, not its length in time.*

Atom (Figure)	**Signal Length** (s)	**HD-AD Time** (s)	**times real-time** (faster)	**HD-AD dict. size** (GB)	**waveform dict. size** (GB)
DS (2)	66	15	4.236	0.016	4030
REDS (3*middle*)	6	2	2.518	0.113	238
REDS (3 *right*)	26	7	3.767	0.113	3520
vREDS (4)	8	4	1.860	0.405	3090

Table 1: *Timing results and memory comparison between HD-AD and waveform dictionary approaches for a given atom prototype.*

the parameters can result in less "natural" sounding transformations, as well as require additional processing to account for complex relationships between atoms in the decomposition [3]. In particular, this can happen when decomposing asymmetric audio features with symmetric atoms. Prototypes which better match the audio structure, like the atoms we have discussed for asymmetric sounds, can preserve the original qualities in the transformed audio, as well as potentially avoid complex inter-atom relations and thus the associated post-processing. However, accurate atomic decomposition on these more elaborated prototypes requires robust techniques such as the one presented in this paper.

Figure 5 shows the HD-AD decomposition using DS atoms of a kalimba recording playing alternately notes C6 and C7, and transforming that sound through the manipulation of the atom parameters. The original audio is successively pitched, shifted down an octave, and then down three octaves by multiplying f by $\frac{1}{2}$ and $\frac{1}{8}$, respectively. In addition, the character of the original notes is transformed by decreasing the damping through multiplying α by $\frac{1}{4}$ and by $\frac{1}{8}$, causing them to ring longer than in the original recording.

6. HYBRID APPROACH

In addition to HD-AD, in [19] we present a variation of HD-AD, called neural network accelerated Matching Pursuit (NN-MP). NN-MP is compromise between a fully HDC atomic decomposition, and one which relies only on waveforms. The idea is that once all of the $\mathbf{s}_\circ(x_k, \dot{\boldsymbol{\lambda}}_k)$ are identified via HD-AD, the x_k can be discarded and instead a dictionary of waveforms can be populated with $\dot{\boldsymbol{\lambda}}_k$. Then a standard MP can be run with this truncated dictionary in order to compute the x_k. In this algorithm, once a stopping criteria is reached, the residual signal can be fed-back into the neural network and the process can be repeated. The same methodology can be applied to MP variants, such as orthogonal Matching Pursuit (OMP) [24]. This method avoids storing a large number of asymmetric waveforms since the dictionary of atoms is customized to the signal being decomposed.

7. CONCLUSION

In this paper a new atomic decomposition approach was introduced and explored. Rather than at the level of waveforms, we chose to operates at the symbolic level of atom parameters by encoding them into VFA hypervectors, and training a neural network to generate these encodings. Through our experiments we showed that this approach has many advantages over waveform dictionary atomic decomposition algorithms with asymmetric atoms – mainly time/memory savings. The new speeds we demonstrated for producing atomic decompositions with asymmetric atom prototypes has the potential to bring atomic decomposition into many new hands, and perhaps as a consequence, further explore its potential as a creative tool.

8. REFERENCES

[1] S. Mallat and Z. Zhang, "Matching Pursuit with Time-Frequency Dictionaries," *Signal Processing, IEEE Transactions on*, vol. 41, pp. 3397–3415, 12 1993.

[2] E. Ravelli, G. Richard, and L. Daudet, "Union of MDCT Bases for Audio Cxoding," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 16, no. 8, pp. 1361–1372, 2008.

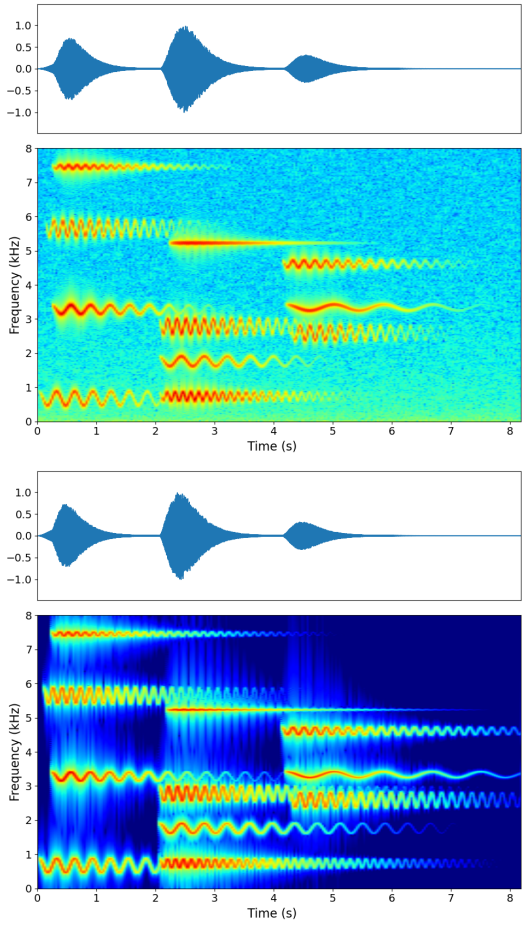

Figure 4: ***(top)*** *Input signal:* 11 *vibrato REDS atoms plus noise.* ***(bottom)*** *HD-AD approx:* 11 *vibrato REDS atoms. Signal length:* 8.192 *sec, HD-AD time:* 4.404 $s = 1.860\times$*'s real-time.*

[3] B. Sturm, C. Roads, A. McLeran, and J. Shynk, "Analysis, Visualization, and Transformation of Audio Signals Using Dictionary-based Methods," *Journal of New Music Research*, vol. 38, no. 4, pp. 325–341, 2009.

[4] L. Daudet, "Sparse and Sructured Decompositions of Signals with the Molecular Matching Pursuit," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 5, pp. 1808–1816, 07 2006.

[5] P. Kanerva, "Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-Dimensional Random Vectors," *Cognitive Computation*, vol. 1, no. 2, pp. 139–159, 2009.

[6] P. Flandrin, *Time-Frequency/Time-Scale Analysis, Volume 10*, Academic Press, Inc., USA, 1st edition, 1998.

[7] M. Goodwin, "Matching Pursuit with Damped Sinusoids," *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 3, pp. 2037–2040, April 1997.

[8] J. L. Flanagan, "Models for Approximating Basilar Membrane Displacement," *The Bell System Technical Journal*, vol. 39, no. 5, pp. 1163–1191, 1960.

[9] X. Rodet, "Time-Domain Formant-Wave-Function Synthesis," *Computer Music Journal*, vol. 8, no. 3, pp. 9–14, 1984.

[10] J. Neri and P. Depalle, "REDS: A New Asymmetric Atom for Sparse Audio Decomposition and Sound Synthesis," in *Proceedings of the 20th International Conference on Digital Audio Effects (DAFx-17)*, Edinburgh, UK, 09 2017, pp. 268–275.

[11] J. Neri, "Sparse Representations of Audio Signals with Asymmetric Atoms," M.S. thesis, McGill University, 2018.

[12] P. Frady, S. Kent, B. Olshausen, and F. Sommer, "Resonator Networks, 1: An Efficient Solution for Factoring High-Dimensional, Distributed Representations of Data Structures," *Neural Computation*, vol. 32, no. 12, pp. 2311–2331, 2020, PMID: 33080162.

[13] B. Komer, T. Stewart, A. Voelker, and C. Eliasmith, "A Neural Representation of Continuous Space Using Fractional Binding," in *Proceedings of the 41st Annual Meeting of the Cognitive Science Society*, Montréal, Canada, 2019.

[14] R. Gayler, "Vector Symbolic Architectures Answer Jackendoff's Challenges for Cognitive Neuroscience," in *ICCS/ASCS International Conference on Cognitive Science*, Sydney, Australia, 13-17 July 2003.

[15] E. Frady, D. Kleyko, C. Kymn, B. Olshausen, and F. Sommer, "Computing on Functions Using Randomized Vector Representations," *arXiv*, vol. abs/2109.03429, 2021.

[16] E. P. Frady, D. Kleyko, and F. Sommer, "A Theory of Sequence Indexing and Working Memory in Recurrent Neural networks," *CoRR*, vol. abs/1803.00412, 2018.

[17] T. Plate, *Distributed Representations and Nested Compositional Structure*, Ph.D. thesis, University of Toronto, 1994.

[18] T. Plate, "Holographic Reduced Representations," *IEEE Transactions on Neural Networks*, vol. 6, no. 3, pp. 623–41, May 1995.

[19] C. Yost, "Atomic Decomposition of Audio with High Dimensional Distributed Representations," M.S. thesis, McGill University, 2022, `https://github.com/ChristianYost/HD-AD/blob/main/CY__MA_Thesis_05_2022.pdf`.

[20] W. Ma, Y. Hu, and H. Huang, "Dual Attention Network for Pitch Estimation of Monophonic Music," *Symmetry*, vol. 13, pp. 1296, July 2021, doi: 10.3390/sym13071296.

[21] M. Ranzato, F. Huang, Y. Boureau, and Y. LeCun, "Unsupervised Learning of Invariant Feature Hierarchies with Applications to Object Recognition," in *Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, 06 2007, pp. 1–8.

[22] D. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," in *4th International Conference on Learning Representations*, San Juan, Puerto Rico, 02-04 May 2016, ICLR 2016.

[23] D. Toledano, M. Fernández-Gallego, and A. Lozano-Diez, "Multi-Resolution Speech Analysis for Automatic Speech Recognition Using Deep Neural Networks: Experiments on TIMIT," *PLoS ONE*, vol. 13, 2018, doi: https://doi.org/10.1371/journal.pone.0205355.

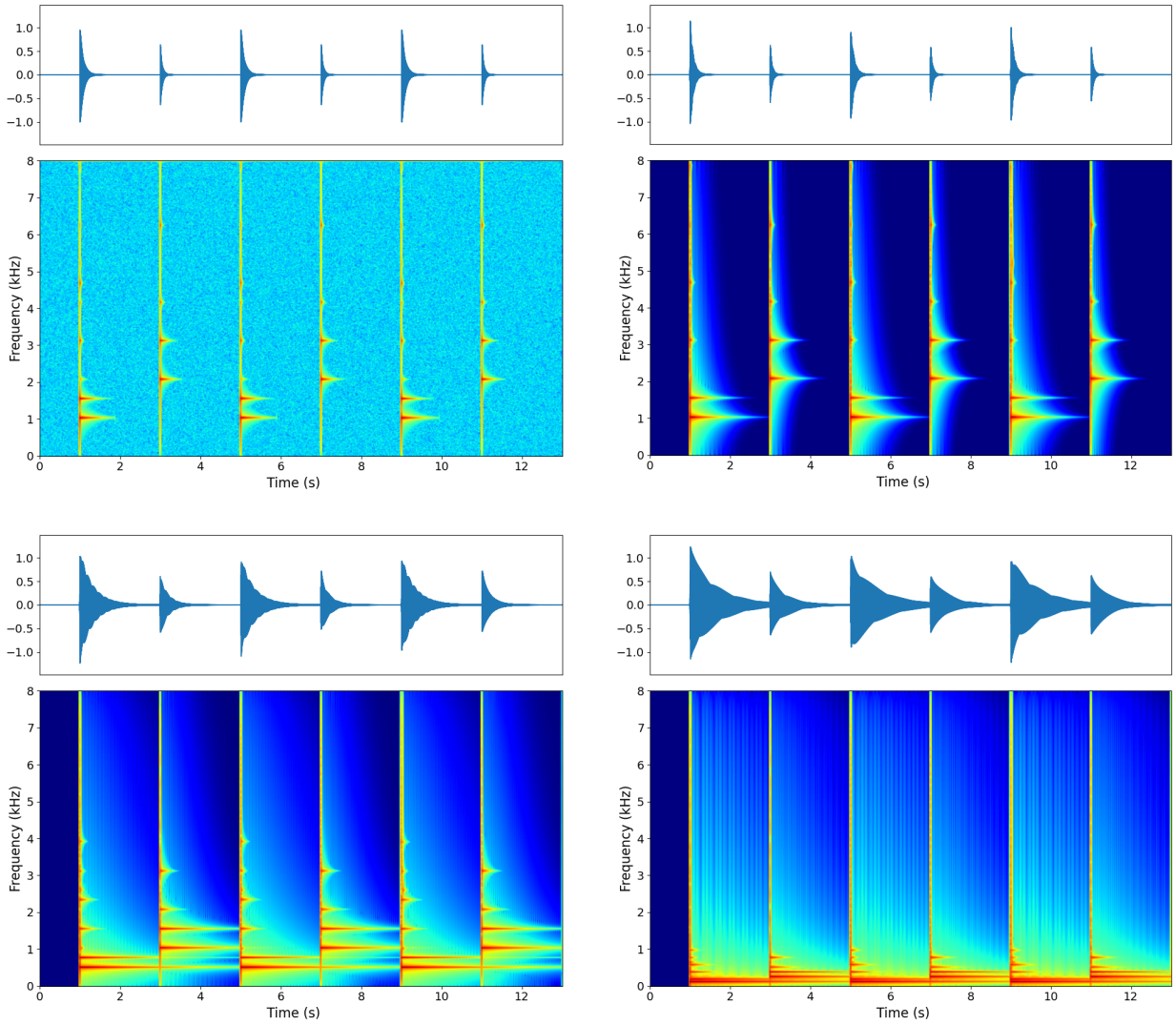

Figure 5: *Sound processing example.* ***(top left)*** 13.000 *s input signal – K-sound,* ***(top right)*** *HD-AD approximation of K-sound with* 54 *DS atoms, computed in* 3.988 $s = 3.260\times$*'s real-time.* ***(bottom left)*** *Audio manipulation of K-sound by multiplying* f *by* $\frac{1}{2}$ *– down one octave – and multiplying* α *by* $\frac{1}{4}$*.* ***(bottom right)*** *Audio manipulation of K-sound by multiplying* f *by* $\frac{1}{8}$ *– down three octaves – and multiplying* α *by* $\frac{1}{8}$*.*

[24] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal Matching Pursuit: Recursive Function Approximation with Applications to Wavelet Decomposition," in *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, 11 1993, pp. 40–44 vol.1.

[25] C. Kereliuk and P. Depalle, "Sparse Atomic Modeling of Audio: A Review," in *Proceedings of the International Conference on Digital Audio Effects (DAFx)*, Paris, France, 09 2011, pp. 81–92.