COGNITIVE SCIENCE
A Multidisciplinary Journal

Cognitive Science 47 (2023) e13233
© 2022 Cognitive Science Society LLC.
ISSN: 1551-6709 online
DOI: 10.1111/cogs.13233

How Do Artificial Neural Networks Classify Musical Triads? A Case Study in Eluding Bonini's Paradox

Arturo Perez, Helen L. Ma, Stephanie Zawaduk, Michael R. W. Dawson

Department of Psychology, University of Alberta

Received 14 June 2021; received in revised form 6 December 2022; accepted 15 December 2022

Abstract

How might artificial neural networks (ANNs) inform cognitive science? Often cognitive scientists use ANNs but do not examine their internal structures. In this paper, we use ANNs to explore how cognition might represent musical properties. We train ANNs to classify musical chords, and we interpret network structure to determine what representations ANNs discover and use. We find connection weights between input units and hidden units can be described using Fourier phase spaces, a representation studied in musical set theory. We find the total signal coming through these weighted connection weights is a measure of the similarity between two Fourier structures: the structure of the hidden unit's weights and the structure of the stimulus. This is surprising because neither of these Fourier structures is computed by the hidden unit. We then show how output units use such similarity measures to classify chords. However, we also find different types of units—units that use different activation functions—use this similarity measure very differently. This result, combined with other findings, indicates that while our networks are related to the Fourier analysis of musical sets, they do not perform Fourier analyses of the kind usually described in musical set theory. Our results show Fourier representations of music are not limited to musical set theory. Our results also suggest how cognitive psychologists might explore Fourier representations in musical cognition. Critically, such theoretical and empirical implications require researchers to understand how network structure converts stimuli into responses.

Keywords: Artificial neural networks; Music theory; Discrete Fourier transform; Chord classification

1. Introduction

Cognitive scientists use models to understand cognitive phenomena. However, a model alone may not improve understanding (Lewandowsky, 1993). For example, computer simulations of cognition often face *Bonini's paradox* (Dutton & Starbuck, 1971). Bonini's

Correspondence should be sent to Michael R. W. Dawson, Department of Psychology, University of Alberta, Edmonton, AB T6G 2R3, Canada. E-mail: mdawson@ualberta.ca

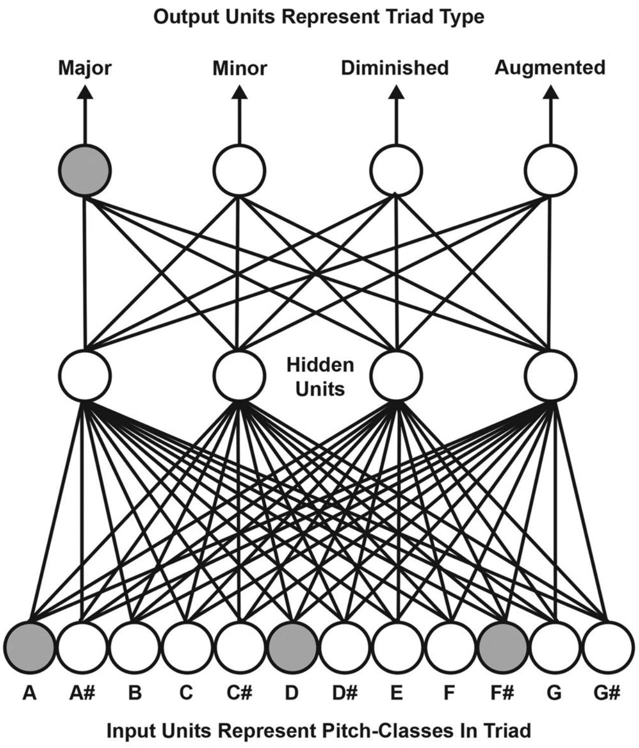
Output Units Represent Triad Type
Major
Minor
Diminished
Augmented
Hidden Units
A
A#
B
C
C#
D
D#
E
F
F#
G
G#
Input Units Represent Pitch-Classes In Triad

Fig. 1. An example artificial neural network (ANN), a multilayer perceptron, for classifying triads. Input units represent which pitch-classes are present in a stimulus. Hidden units detect higher-order regularities in stimuli. Output units process hidden unit activities and represent the ANN's response to a stimulus. In the example, the C major triad is presented to the network; the network responds by turning the "major" output unit on, and the other three output units off.

paradox occurs when a model is just as difficult to understand as is the phenomenon being modeled.

Artificial neural networks (ANNs) often confront Bonini's paradox. An ANN (Fig. 1) is a network of processors (i.e., input units, hidden units, and output units) connected to one another by weighted connections. Input units represent stimuli, output units represent responses to stimuli, and hidden units detect higher-order relationships between input units to help the network generate correct responses. ANNs lead to Bonini's paradox because researchers usually do not understand how networks accomplish stimulus/response mappings (Berkeley, Dawson, Medler, Schopflocher, & Hornsby, 1995; Mozer & Smolensky, 1989; Robinson, 1992; Swingler, 2014; Zhang et al., 2018). "One thing that connectionist networks have in common with brains is that if you open them up and peer inside, all you can see is a big pile of goo" (Mozer & Smolensky, 1989, p. 3).

ANNs are hard to understand for many reasons. One reason is that when multilayer perceptrons (e.g., Fig. 1) learn to solve a problem, the rule used to train the network does not force hidden units to efficiently represent detected features. For instance, different hidden units often detect similar features. Furthermore, hidden units often do not represent detected features locally; instead, activities of several hidden units collectively represent a meaningful feature (Hinton, 1986; Van Gelder, 1991). Another reason is that when many layers of hidden units exist in a network, as in deep learning networks, interpreting networks become much more difficult (Fan, Xiong, Li, & Wang, 2021; Huang et al., 2020; Zhang, Tino, Leonardis, & Tang, 2021). In general, ANNs are black box models that accomplish tasks in an unspecified manner.

However, if ANNs are black box models, then ANNs cannot provide new theories to cognitive science (McCloskey, 1991; Seidenberg, 1993). McCloskey (1991, p. 387) argues if ANNs cannot be interpreted then "connectionist networks should not be viewed as theories of human cognitive functions, or as simulations of theories, or even as demonstrations of specific theoretical points." To provide theory, cognitive scientists must understand what is inside an ANN's black box.

The current article provides a case study in attempting to avoid Bonini's paradox by interpreting the internal structure of trained ANNs. Novel representations can be discovered in ANNs and can be used to inform cognitive science. Researchers can interpret an ANN's internal structure and explain how the network uses its structure to transform stimuli into responses (Augasta & Kathirvalavakumar, 2012; Baesens, Setiono, Mues, & Vanthienen, 2003; Berkeley et al., 1995; Bullinaria, 1997; Dawson, 2004, 2005, 2018; Dawson, Perez, & Sylvestre, 2020; Dawson & Piercey, 2001; Gallant, 1993; Hayashi, Setiono, & Yoshida, 2000; Hinton, 1986; Moorhead, Haig, & Clement, 1989; Omlin & Giles, 1996; Setiono, Baesens, & Mues, 2011; Setiono, Thong, & Yap, 1998; Taha & Ghosh, 1999).

1.1. Bonini's paradox and musical ANNs

Our case study involves multilayer perceptrons trained to classify musical triads (Fig. 1). Many researchers use ANNs to study musical cognition (Bharucha, 1995, 1999; Dawson, 2018; Griffith, 1995; Griffith & Todd, 1999; Krumhansl et al., 2000; Krumhansl & Toiviainen, 2001; Mihelac & Povh, 2020; Russo, Vempala, & Sandstrom, 2013; Stevens & Latimer, 1992; Temperley, 2013; Todd & Loy, 1991). Such "musical networks," like most ANNs, face Bonini's paradox.

However, Bonini's paradox is often ignored in such research because researchers use musical ANNs to model *implicit learning* of musical properties and argue interpreting an ANN is not required (Bharucha, 1999; Tillmann, Bharucha, & Bigand, 2000, 2003). In implicit learning, participants learn statistical regularities without experiencing explicit formal rules; such learning is unconscious and cannot be articulated by the learner (Bigand & Poulin-Charronnat, 2006; Tillmann & Bigand, 2004). When in ANNs model implicit learning, researchers assume the overall pattern of network connections represents statistical regularities that cannot easily be formalized. "There are no explicit rules or concepts stored in the model" (Tillmann, Bharucha, & Bigand, 2000, p. 907). In short, musical cognitivists assume ANNs capture *informal* properties (Bharucha, 1999), which cannot be described *formally*.

However, our case study demonstrates that musical networks can have a rich, *formal* interpretation, even when ANNs adopt distributed representations. Below, we examine the connection weights in musical ANNs and discover the weights are related to formal music theory, in particular, to the Fourier structure of musical sets. We also show how our networks use Fourier structure to accomplish a particular musical task: classifying musical triads. To begin, let us briefly introduce the Fourier analysis of musical sets.

1.2. Musical sets and Fourier phase spaces

Formal music theory begins by assuming the basic elements of Western music are the 12 pitch-classes (C, C#, D, D#, E, F, F#, G, G#, A, A#, and B). A musical entity can be represented as a set composed of these elements (Forte, 1973; Lewin, 2007; Schuijer, 2008). For instance, set theorists often represent a musical entity as a set called an *indicator vector* (Milne, Bulger, & Herff, 2017). An indicator vector assumes pitch-classes are represented in a particular order: (C, C#, D, D#, E, F, F#, G, G#, A, A#, B). Each entry in an indicator vector represents the number of times each pitch-class occurs in a musical entity. That is, the first entry indicates the number of Cs in the entity, the second indicates the number of C#s, and so. For example, the only pitch-classes present in the C major triad are C, E, and G. As a result, the indicator vector for the C major triad is (1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0). Indicator vectors provide an important link between musical set theory and musical ANNs. For example, the input units in the Fig. 1 network represent the indicator vectors of musical entities. In the figure, the stimulus is the C major triad.

Formal music theory uses mathematical operations on musical sets to uncover surprising similarities between musical entities. One such operation is the *discrete Fourier transform* (DFT), which converts a complex signal into a set of simpler components. Each component is a cosine of a particular frequency, magnitude, and phase. Performing the DFT on an indicator vector delivers the magnitude and the phase of different cosines, each of a different frequency (Amiot, 2016). To reconstruct the original indicator vector, one adds together the different cosines delivered by the DFT. Fourier representations are useful for exploring numerous musical relationships (Amiot, 2016; Quinn, 2006, 2007; Yust, 2016, 2017a, 2019).

Our previous research discovered another important link between musical set theory and musical ANNs (Dawson et al., 2020). In musical ANNs trained on several different musical tasks, we discovered weights from input units to a hidden unit (see Fig. 1) represent a cosine function of the type delivered by the DFT of a musical set. While Dawson et al. (2020) demonstrated a link between the DFT of musical sets and the structure of musical ANNs, they did not explore *how* a musical ANN uses Fourier components to perform musical judgments. The current article fills this gap by exploring how the Fourier-like representations in our networks are used to classify musical triads.

Our case study proceeds as follows: We begin by training two different types of ANNs to classify triads (Fig. 1). We discover all hidden units in both types of networks represent Fourier phase spaces (e.g., Yust, 2016) in hidden unit connection weights, replicating and extending the results of Dawson et al (2020). We then show, for both ANN types, the signal sent to a hidden unit by a stimulus triad measures the similarity between the DFT of the hidden unit weights and the DFT of the presented triad. This is astonishing because the network does *not* compute the DFT of either the weights or the stimulus. Finally, we show that even though both types of hidden units compute the same measure of DFT similarity, the measure is used quite differently by each network type to classify triads.

Why do hidden units represent Fourier phase space projections? Below, we show our hidden units take advantage of periodic regularities, which relate one triad to another. For instance, many hidden units produce identical responses to same-typed triads whose roots are a tritone apart (e.g., C is separated from F# by a tritone or six semitones). This produces a simpler representation of triads in hidden unit space, making it easier for output units to classify triads. Such periodic regularities emerge directly from the phase space projections represented by hidden unit weights because—being cosine functions—phase space projections are themselves periodic.

2. Method

2.1. *Task*

We train all networks to classify musical triads represented as indicator vectors. In the triad classification task, we present a network of a triad—a stimulus comprised three different pitch-classes. The network learns to classify the stimulus as belonging to one of four different triad types: major, minor, diminished, or augmented. Each triad type is defined by the number of semitones separating each of its component pitch-classes (e.g., Hanson, 1960). For example, in a major triad like C major, C is the tonic, the pitch-class E is four semitones above the tonic, and the pitch-class G is seven semitones above the tonic. In contrast, in a minor triad like A minor, A is the tonic, the pitch-class C is three semitones above the tonic, and the pitch-class E is seven semitones above the tonic.

2.2. *Network architectures*

We train two different kinds of networks to classify triads. The first are networks of *value units* (Ballard, 1986). In such a network, all output and hidden units are value units. A value unit generates a strong response to a very narrow range of incoming signals (called *net input*) as shown in Fig. 2. If net input falls outside this range (too high or too low), then a value unit generates a weak response. Value units use the bell-shaped Gaussian equation as the activation function (Dawson & Schopflocher, 1992): activity $= exp(-\pi(net - \mu)^2)$. In the equation, *net* represents the total signal received by a processor (i.e., net input) and μ is the Gaussian's mean. When net input equals μ, the equation generates a maximum activity of 1. As net input becomes greater than or less than μ, processor activity quickly becomes near zero. In Fig. 2a, μ equals 0. To classify triads, a network of value units uses 12 input units to represent a triad's indicator vector and four output units to represent the triad type (see Fig. 1). However, unlike Fig. 1, a network of value units only requires four hidden units to correctly classify the stimuli. Our standard practice uses pilot studies to determine the minimum number of hidden units required by a network to reliably learn to solve a problem (e.g., Dawson, 2018). Our pilot simulations indicated only four hidden value units are necessary for classifying triads.

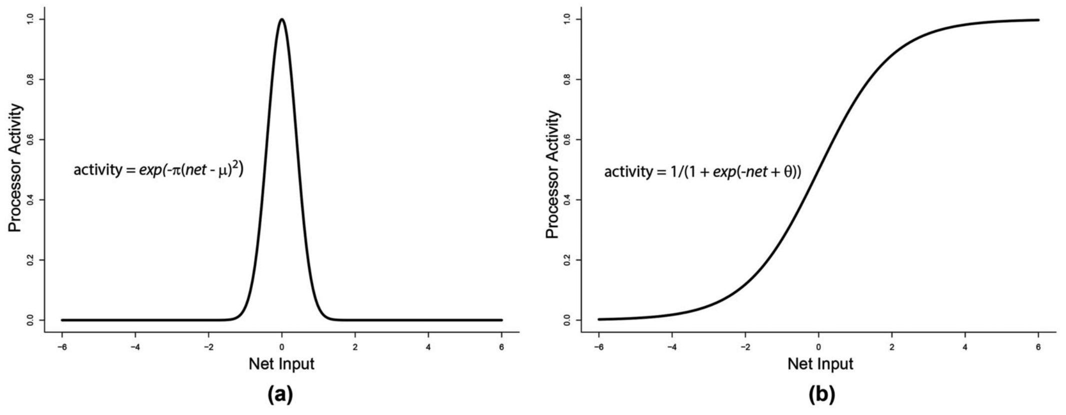

Fig. 2. Two examples of different activation functions used by processors in multilayer perceptrons. (a) The bell-shaped activation function of a value unit, defined by the Gaussian equation. (b) The sigmoid-shaped activation function of an integration device, defined by the logistic equation.

The second architecture we trained were networks of *integration devices* (Ballard, 1986). In such a network, all output and hidden units are integration devices. An integration device uses a sigmoid-shaped activation function, "squashing" net input into a range between 0 and 1 (Fig. 2b). Our integration devices use the logistic equation as the activation function (Rumelhart, Hinton, & Williams, 1986): activity $= 1/(1 + exp(\text{-}net + \theta))$. In the equation, *net* represents net input, and θ is the bias of the equation. Bias is analogous to a neuron's threshold. When net input equals θ, the equation generates activity of 0.5. In Fig. 2b, θ is equal to 0. Pilot studies revealed the simplest network of integration devices capable of reliably solving the triad classification problem requires eight hidden units as illustrated in Fig. 1.

We studied two different kinds of networks to explore three different questions. First, previous research shows networks of value units represent phase space projections (Dawson et al., 2020). We wanted to replicate this result, and then explain how such projections are used. Second, we wanted to determine if networks of integration devices also represent phase space projections. Third, if both types of networks represent phase space projections, we wanted to determine if both use such a representation in the same way to classify triads.

2.3. Training set

The training set consists of 48 different triads: the 12 major triads, the 12 minor triads, the 12 diminished triads, and the 12 augmented triads. We create these stimuli by taking each of the 12 possible pitch-classes in Western music as a tonic for each of the four different triad types.

We represent each stimulus as a binary vector of 12 values (i.e., as an indicator vector), where each value indicates the presence or absence of a particular pitch-class (see the input unit labels in Fig. 1). To illustrate our stimulus representation, the C major triad is the indicator vector (1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0), while the C minor triad is the indicator vector (1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0). The entries in a stimulus indicate whether an input unit is turned on (activity = 1) or off (activity = 0) when the triad is presented to a network.

2.4. Training

We train a network to classify each triad in the training set. To say a network learns is to say the network's connection weights are modified to improve response accuracy. When a triad is presented, the network learns to activate the output unit representing the triad's type and to deactivate the other three output units as illustrated in Fig. 1. In Fig. 1, the gray shading indicates the presentation of pitch-classes C, E, and G to input units (i.e., the C major triad).

We use the generalized delta rule to train integration device networks on the triad classification problem (Rumelhart et al., 1986). Before training, we initialize all connection weights in a network to a random value between -0.1 and 0.1. The bias of each processor—the value of θ in the logistic equation—is held at 0 throughout training. We use a 0.1 learning rate to train networks of integration devices.

We use a modified version of the generalized delta rule to train networks of value units on the triad classification problem (Dawson & Schopflocher, 1992). Before training, we initialize all connection weights in a network to a random value between -0.1 and 0.1. The value of μ in the Gaussian equation is held at 0 throughout training. We use a 0.005 learning rate to train networks of value units.

Training proceeds for both network types as follows: We present a stimulus, causing a network to respond. We use the learning rule to modify connection weights based on response error. We update network weights after each stimulus presentation. We repeat this procedure for the next stimulus. We present stimuli in an epoch-by-epoch fashion as is our standard practice (Dawson, 2004, 2005). One epoch involves training a network once on each stimulus in the training set. We randomize stimulus order before each epoch. We train a network until it converges upon a solution. We define a converged network as one generating a "hit" for each output unit for each stimulus in the training set. We define a "hit" as output activity 0.9 or higher when the desired response is 1, or as output activity 0.1 or lower when the desired response is 0.

We train 20 different networks of value units and 20 different networks of integration devices because connection weights are randomized prior to training, and each network is a different "participant" in our study. Each network solves the triad classification problem. On average, value unit networks converge after 6483.7 epochs (range between 2052 and 27,702 epochs). On average, integration device networks converge after 16,530.2 epochs (range between 4888 and 39,509 epochs). It is typical for networks of value units to learn faster than networks of integration devices (e.g., Dawson & Schopflocher, 1992).

3. Results

3.1. Defining phase space projections

Our first research question is whether the hidden units of our networks adopt the same kind of representations previously reported by Dawson et al. (2020). Before answering this question, let us quickly define precisely what phase space projections are.

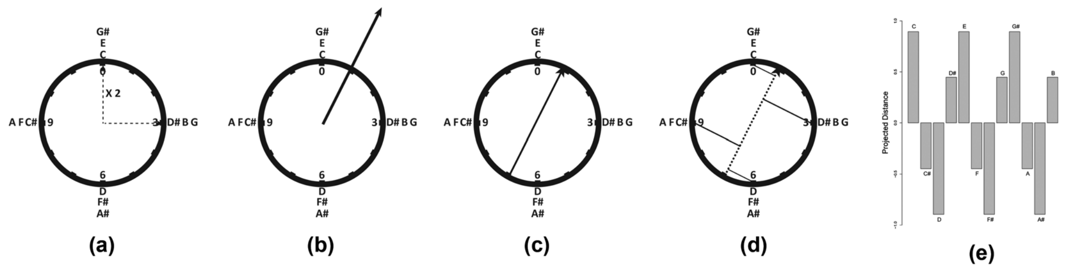

Fig. 3. An illustration of Yust's (2016) circular averaging to determine the discrete Fourier transform (DFT) components for musical sets. (a) The representation of the C major triad (C, E, G) in Ph3. The "X 2" in the Ph3 diagram indicates that the same vector occurs twice, once for C and once for E. (b) The circular averaging (vector addition) of the three vectors in A. The magnitude of the DFT component for Ph3 is the length of the vector, 2.24. (c) The phase of the DFT component is represented by a vector spanning the diameter of the phase space and pointing in the same direction as the circular average in (b). (d) Projections are taken from each pitch-class position around the phase space to the phase representation from (c). (e) Phase space projections are the distances from the middle of the phase representation in (d) to the projection in D and are plotted in the bar graph.

We use Yust's method to determine an indicator vector's DFT (Yust, 2016). Yust uses six different representations that he calls *phase spaces* (Ph1, Ph2, etc.). Ph1 represents a cosine with a frequency of 1, Ph2 represents a cosine with a frequency of 2, and so on. Yust represents each phase space as a clock face; pitch-classes are arranged around the clock face according to the phase space's frequency (see Fig. 3). Each phase space represents different harmonic qualities (Hanson, 1960). Each phase space relates to a category proposed by Hanson (Quinn, 2006, 2007). For example, musical entities consisting mostly of perfect fifths are best described by Ph5.

Yust (2016) computes an indicator vector's DFT as follows: Yust represents each pitch-class present in an indicator vector by drawing a vector from a phase space's clock center to the pitch-class's position on the clock face (Fig. 3). Fig. 3a represents the C major triad as three vectors (dashed arrows) in Ph3, each vector pointing to a triad's component (C, E, or G) arranged around the phase space's clock face. Fig. 3b represents the sum of the three vectors, which Yust calls the *circular average*. A circular average for a phase space determines all required properties of a component (i.e., a cosine) used to reconstruct an indicator vector. The phase space (Ph1, Ph2, etc.) provides the frequency of the component. The length of a circular average's vector represents the magnitude of the component. The direction of a circular average's vector represents the phase of the component. Each circular average can be represented as the (*x, y*) coordinate of its vector in its phase space. Later, we use the set of six pairs of (*x, y*) coordinates to represent a complete DFT and compare these coordinate sets to measure the similarity between two DFTs.

Yust (2016) uses the circular average to determine the specific cosine components to be weighted and summed to recreate an indicator vector. Yust begins by representing phase as a vector drawn diagonally across the phase space's clock face (Fig. 3c). The direction of the phase vector is the same as the direction of the circular average (e.g., compare Fig. 3b to c). The indicated "clock time" provides a numerical representation of phase. Yust then projects

Table 1
The mean values of the fit between hidden unit weights (i.e., the weights from the input units to a hidden unit) and phase space projections for six phase spaces, with standard deviations provided in parentheses

Architecture	Best Fit	Second Best Fit	Third Best Fit	Fourth Best Fit	Fifth Best Fit	Worst Fit
Value unit	0.86	0.38	0.20	0.02	0.01	0.01
	(0.12)	(0.38)	(0.20)	(0.02)	(0.01)	(0.01)
Integration device	0.83	0.35	0.25	0.16	0.11	0.07
	(0.13)	(0.17)	(0.13)	(0.11)	(0.08)	(0.07)

each pitch-class from its position on the clock face to the phase vector (Fig. 3d). Finally, Yust measures the distance from the clock's center, along the phase vector, to each projection. A positive distance is from the center in the direction of the arrow on the phase vector; otherwise, the distance is negative. The 12 distances define the *phase space projections* for a phase space; the phase space projections for the C major triad in Ph3 are graphed in Fig. 3e. The phase space projections represent components of a cosine that has the frequency associated with the phase space and has been shifted by the phase indicated by the direction of the circular average. If one multiplies each set of phase space projections by magnitude (the length of the circular average) and adds the weighted phase space projections together, then the original indicator vector is reproduced.

Dawson et al (2020) found graphs of hidden unit weights were strikingly similar to graphs of phase space projections like Fig. 3e and reported high correlations between hidden unit weights and a single phase space projection for all hidden units in their musical networks of value units. We now consider, for our triad classifying ANNs, whether we replicate this result in our value unit networks, and whether we can extend this result to networks of integration devices.

3.2. Hidden unit weights represent phase space projections

Do our hidden units represent phase space projections to classify triads? To answer this question, we fit phase space projections from each of Yust's six phase spaces to each hidden unit's connection weights, a method used in our previous research (Dawson et al., 2020). For each phase space, we find the best fitting phase space projections from each of the six phases spaces to one hidden unit's connections, where we measure fit as the correlation between weights and phase space projections. We then repeat this process for the next hidden unit.

Table 1 summarizes our results for all hidden units from all networks. It provides the average best fit, second best fit, and so on for the hidden units. We compute the average fit by taking the mean correlation over the 80 hidden value units or the mean correlation over the 160 hidden integration devices. We find a very high average fit between each hidden unit's connection weights (i.e., the weights from the input units to the hidden unit) and phase space projections from one phase space. For our 80 hidden value units, correlations for the best-fitting phase space range from a maximum of 1.00 to a minimum of 0.57. The average correlation for the phase space projections for the next best-fitting phase space is much lower,

Table 2
The number of hidden units that are best fit by each of the 6 phase spaces delivered by the discrete Fourier transform (DFT), taken over all 20 networks of each type of architecture

Architecture	Ph1	Ph2	Ph3	Ph4	Ph5	Ph6
Value unit	1	5	9	21	26	18
Integration device	3	10	26	58	35	28

ranging from a maximum of 0.70 to a minimum of 0.01. These results replicate our previous results for value unit networks trained on this task (Dawson et al., 2020).

For our 160 hidden integration devices, correlations for the best-fitting phase space range from a maximum of 1.00 to a minimum of 0.49. The average correlation for the phase space projections for the next best-fitting phase space is much lower, ranging from a maximum of 0.64 to a minimum of 0.01. The results for integration device networks are similar to our results for value unit networks. Hidden unit connection weights in our integration device networks also represent phase space projections.

Note. Fit is the correlation between phase space projections and connection weights. Fits are averaged across 80 hidden units from the 20 different value unit networks and across 160 hidden units from the 20 different integration device networks. The "best fit" is the correlation between the weights and the phase space that produces the highest correlation; the "second best fit" is the correlation between the weights and the phase space that produces the second highest correlation, and so on.

Table 1 indicates that hidden unit weights are the best fit by a phase space projection from one phase space. Table 2 provides the count of how many hidden units are the best fit by Ph1, by Ph2, and so on. Table 2 provides our first suggestion different architectures use different methods to classify triads. For instance, while Ph5 is the most preferred space for value unit networks, Ph4 is most preferred by integration device networks. Similarly, phase space preferences for value unit networks show the highest preferences for Ph4 through Ph6, while integration device network preferences are more broadly distributed.

As we noted earlier, one hurdle to interpreting ANNs arises when different hidden units respond to similar features. Our networks demonstrate such redundancy. For each network, we count the number of hidden units that share a best-fitting phase space with at least one other hidden unit in the same network. Thirteen of our 20 value unit networks had two (out of four) hidden units best fit by the same phase space. All 20 integration device networks included at least six hidden units best fit by the same phase space.

3.3. Net inputs to hidden units measure the similarity between DFTs

Our results from the preceding section indicate that we can describe all hidden unit weights as phase space projections. We now consider how networks use phase space projections to classify triads. We begin by showing when a hidden unit computes net input, net input measures the similarity between two DFT structures: one that results if we perform a DFT on the hidden unit's weights, and the other that results if we perform a DFT on the indicator vector.

The activation functions (see Fig. 2) convert the total incoming signal, net input, into internal activity. For a hidden unit, net input is the sum of the signals being sent by the input units, after each signal is scaled by a connection weight. In other words, net input is the inner product of a vector of input unit activities and a vector of hidden unit weights.

Generally, inner products measure the similarity between two vectors. For instance, the Pearson product-moment correlation is an inner product between two vectors after each vector is normalized to unit length. Does a hidden unit's net input also measure the similarity between the unit's connection weights and a stimulus? If so, how might we describe the measured similarity?

Because hidden unit connection weights represent phase space projections, we hypothesize net inputs measure the similarity between two DFT structures. To be more precise, we can perform a DFT of a hidden unit's weights, as well as of the indicator vector that serves as a stimulus. We hypothesize net input measures the similarity between a unit's and a stimulus's DFT. Such a result would be startling because a hidden unit's basic operation (computing the inner product between a set of weights and a set of input unit activities) does not compute either DFT.

To test our hypothesis, we compute the DFT for the weights of each hidden unit. We also compute the DFT for the indicator vector of each triad. Each DFT produces six different circular averages—vectors like the one depicted in Fig. 3b—one for each phase space. As discussed earlier, we describe each circular average by the (x, y) coordinate of the vector's endpoint in a phase space, and each circular average provides all the required information about a cosine used when reconstructing an indicator vector. Thus, the set of circular averages generated by Yust's method provides a complete DFT. Let the (x, y) coordinates of Phase 1 circular average be named $(\mathrm{Ph1}_x, \mathrm{Ph1}_y)$, the coordinates of Phase 2 circular average be named $(\mathrm{Ph2}_x, \mathrm{Ph2}_y)$, and so on. We represent the entire DFT in one vector that contains all six pairs of coordinates $(\mathrm{Ph1}_x, \mathrm{Ph1}_y, \mathrm{Ph2}_x, \mathrm{Ph2}_y, \mathrm{Ph3}_x, \mathrm{Ph3}_y, \mathrm{Ph4}_x, \mathrm{Ph4}_y, \mathrm{Ph5}_x, \mathrm{Ph5}_y, \mathrm{Ph6}_x, \mathrm{Ph6}_y)$. We measure the similarity between two different DFTs by computing the inner product between two such vectors. The higher the inner product, the greater the similarity between the two DFTs. We compute DFT similarity between each set of hidden unit weights and each triad (i.e., we compute 48 different inner products for each hidden unit, one for each stimulus).

We can now examine the relationship between net input and our inner product measure of DFT similarity. We correlate the 48 net inputs and the 48 DFT similarity measures for each hidden unit. We find a near-perfect relationship between net input and DFT similarity for every hidden unit in every network. For the 80 hidden value units, the average correlation between net input and our measure of DFT similarity is 0.99 ($SD = 0.016$). For the 160 hidden integration devices, the average correlation between net input and our measure of DFT similarity is 0.98 ($SD = 0.018$).

In short, we can describe the net input calculated by a hidden unit as measuring the similarity between the DFT of a set of hidden unit connection weights and the DFT of a stimulus (indicator vector). This result is surprising because when net input is calculated, neither DFT is computed: The hidden unit merely takes the inner product between a set of connections (phase space projections) and a set of input unit activities (the indicator vector). The hidden unit behaves *as if* the similarity of two DFTs is computed; it measures DFT similarity without performing a DFT. How can hidden units calculate DFT similarity?

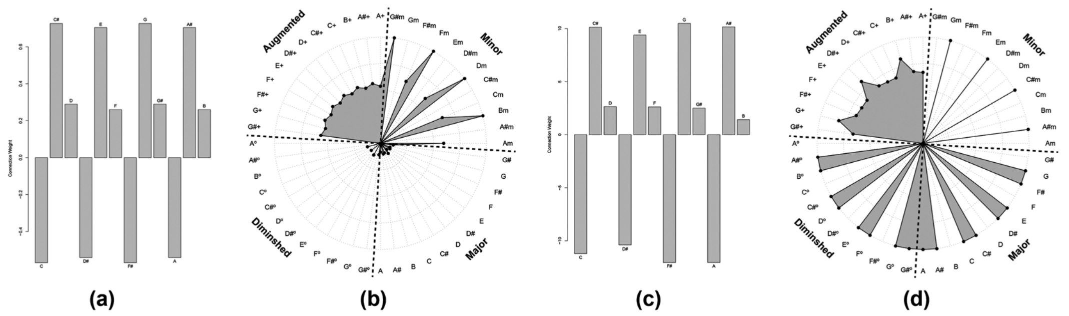

Fig. 4. (a) Connection weights for one hidden value unit from a network trained to classify triads. (b) Responses of the unit from (a) to each of the 48 stimulus triads. (c) Connection weights for one hidden integration device trained on the same problem. Note the similar pattern of weights in (a) and (c). (d) Responses of the unit from (c) to each of the 48 stimulus triads.

A hidden unit's net input is the inner product between a vector of connection weights and an indicator vector. The connection weights represent phase space projections. If we were to perform a DFT on an indicator vector, we could express the indicator vector as a sum of weighted phase space projections (Yust, 2016). We could compute the inner product between the connection weights and each of these DFT components. If we sum each of these inner products, we will produce the inner product of the weights with the original indicator vector because the indicator vector is a linear combination of the DFT components.

Why is it important to describe a basic network operation (computing net input) in terms of measuring the similarity between DFT components? If we do not express network operations in terms related to a domain of interest (in our case, triad classification), then we are not interpreting network structure in a meaningful way. When we fail to relate network properties to a domain of interest, we only have a mathematical account of network behavior, such as "a particular output unit turns on because of the hidden unit activities produced by the stimuli." The mathematical account ignores the domain of interest, and therefore cannot help researchers develop cognitive theory.

3.4. *Different responses to DFT similarity*

Net input measures unit/stimulus DFT similarity for both value units and integration devices. However, the differences between the Gaussian and logistic activation functions (see Fig. 2) cause value units and integration devices to respond to DFT similarity in different manners.

To illustrate, consider two example hidden units, one from a value unit network and the other from an integration device network. There is a high correlation ($r = .99$) between the connection weights of the two example units shown in Fig. 4. Each hidden unit in Fig. 4 represents a similar phase space projection.

However, the two hidden units generate very different activations to identical stimuli, despite having similar connection weights. Fig. 4's radar charts plot the responses of each hidden unit to each triad. A radar chart's center represents a minimum response of 0, while a radar chart's outer ring represents a maximum response of 1. Each radar chart arranges the triads according to the triad root; each radar chart quadrant illustrates responses to a different triad type. Fig. 4 shows that two different hidden units, with very similar patterns of connection weights, respond differently to the same stimuli because the units use different activation functions.

The value unit generates very weak responses to diminished and major triads and moderate responses to augmented triads. It produces strong responses to four minor triads, moderate responses to another four, and no response to the remaining four.

In contrast, the integration device produces strong responses to eight diminished triads and eight major triads, but the unit generates no response to the other four triads of either type. It also generates moderately strong responses to all augmented triads and strong responses to four minor triads. The integration device generates no response to the other eight minor triads.

The different patterns of responses in Fig. 4 make sense after considering net input, as a measure of the similarity between two DFT structures, in the context of the two Fig. 2 activation functions. Highly similar DFT structures produce a strong, positive net input. Highly dissimilar DFT structures produce a strong, negative net input. Completely unrelated DFT structures produce a near-zero net input. However, different activation functions will produce different responses to identical net inputs (i.e., to identical DFT similarities).

First, consider an integration device. The logistic function monotonically increases, so an increase in net input must produce an increase in an integration device's activity (Fig. 2b). As the similarity increases between the DFT structure of the hidden unit weights and the DFT structure of the stimulus, the activity of the integration device will also increase. Metaphorically, an integration device's activity represents the processor's "sympathetic vibration" to the similarity between DFT structures of the stimulus and the weights.

Now consider a value unit using the Gaussian activation function (Fig. 2a). A value unit produces weak activity when the DFT structure of the hidden unit weights is strongly unrelated to the DFT structure of the stimulus (negative net input). However, a value unit also produces weak activity when the DFT structure of the hidden unit weights is strongly related to the DFT structure of the stimulus (positive net input). A value unit only produces strong activity when the DFT structure of the hidden unit weights is *unrelated* to the DFT structure of the stimulus (near-zero net input). Metaphorically, a value unit's activity represents "unsympathetic vibration" to the similarity between DFT structures of the stimulus and the weights.

The strikingly different responses of different unit types to the same stimuli, even when both units represent similar phase space projections (Fig. 4), provide further evidence that our musical networks use Fourier representations atypically. For instance, if hidden units represent phase space projections, one might hypothesize that output unit weights represent phase space magnitudes, and output units classify triads by combining magnitude and phase information. However, such an account cannot apply to both network types because responses of hidden units to the same kind of information (DFT similarity) differ by network type. The same explanation of network operations cannot apply to both network types.

The two response patterns illustrated in Fig. 4 also reveal another typical characteristic of our hidden units: broad tuning. First, a hidden unit in our networks typically generates *different* responses to triads belonging to the *same* type. Second, a hidden unit in our networks typically generates the *same* responses to some triads belonging to *different* types. Both properties are evident in the Fig. 4 radar charts. However, we can also discover musical systematicity in such broad tuning as we now describe.

3.5. *Classifying hidden value units*

Dawson (2018) notes that particular types of hidden unit value units often appear in networks trained on musical tasks. Different types of hidden units produce particular correlations between the unit's weights and the same weights rotated by a musical interval. Let a set of weights between a hidden unit and the input units in one of our networks (Fig. 1) be the vector $\mathbf{w} = (w_0, w_1, w_2, \dots w_{11})$, where w_0 is the weight from the input unit representing the pitch-class C, is the weight from the input unit representing the pitch-class C#, and so on. We rotate **w** by shifting each weight to the right by the same amount, moving the rightmost vector entries to the start of the vector. The amount of the rotation is measured in semitones. For example, to rotate **w** by one semitone, we shift each weight to the right by one place in the vector, producing $(w_{11}, w_0, w_1, \dots, w_9, w_{10})$. Similarly, to rotate **w** by two semitones, we shift each weight to the right by two places in the vector, producing $(w_{10}, w_{11}, w_0, \dots, w_8, w_9)$, where the amount is measured in semitones. Each vector we create can be described as a row in the circulant matrix created from **w** (Amiot, 2016).

We can identify different types of hidden units by examining the correlation between the unit's weights and the same weights rotated a particular amount. For instance, in a *tritone equivalence unit* connection weights from input units representing pitch-classes a tritone (six semitones) apart are assigned nearly identical weights. Therefore, if a tritone equivalence unit's weights are rotated by six semitones, a nearly perfect correlation emerges between the original weights and the rotated weights.

We use correlations between a vector of weights and its rotations to identify types of hidden units. First, we take each weight vector and create 12 different rotations of it (i.e., a rotation of 0 semitones, 1, semitone, 2 semitones, etc., up to 11 semitones). We then compute the correlation between the original vector and each rotated vector. Finally, we represent the weights as a vector of the resulting correlations (similar to the rows in Table 3). The vector of correlations represents the result of autocorrelating the weight vector with each of its possible rotations.

Second, we analyze the vectors of correlations to identify different types of hidden units. We perform k-means cluster analysis on the vectors (i.e., the 80 different vectors of correlations, each representing a hidden value unit). We expect each cluster to reveal a different hidden unit type.

We determine the optimum number of clusters to extract using the elbow method (Thorndike, 1953). We perform a k-means cluster analysis for values of k from 1 to 10, we sum the squared distances between each hidden unit and the centroid of its cluster, and we plot the sum as a function of the number of clusters. We then look for the number of clusters

Table 3
Summary statistics for the four different clusters to which hidden value units were assigned using k-means clustering

			Rotation of Weights in Semitones											
Cluster	Type	*N*	0	1	2	3	4	5	6	7	8	9	10	11
1	Tritone equivalence	21	1.00	−0.56	0.43	−0.73	0.43	−0.56	1.00	−0.56	0.43	−0.73	0.43	−0.56
2	Cycle of minor thirds	22	1.00	−0.42	−0.48	0.80	−0.49	−0.42	0.99	−0.42	−0.49	0.80	−0.48	−0.42
3	Tritone balance	28	1.00	−0.42	0.32	−0.02	-0.33	0.43	−0.95	0.43	−0.33	−0.02	0.32	−0.42
4	Cycle of major thirds	9	1.00	0.05	−0.78	−0.01	0.77	−0.04	-0.97	−0.04	0.77	−0.01	−0.78	0.05

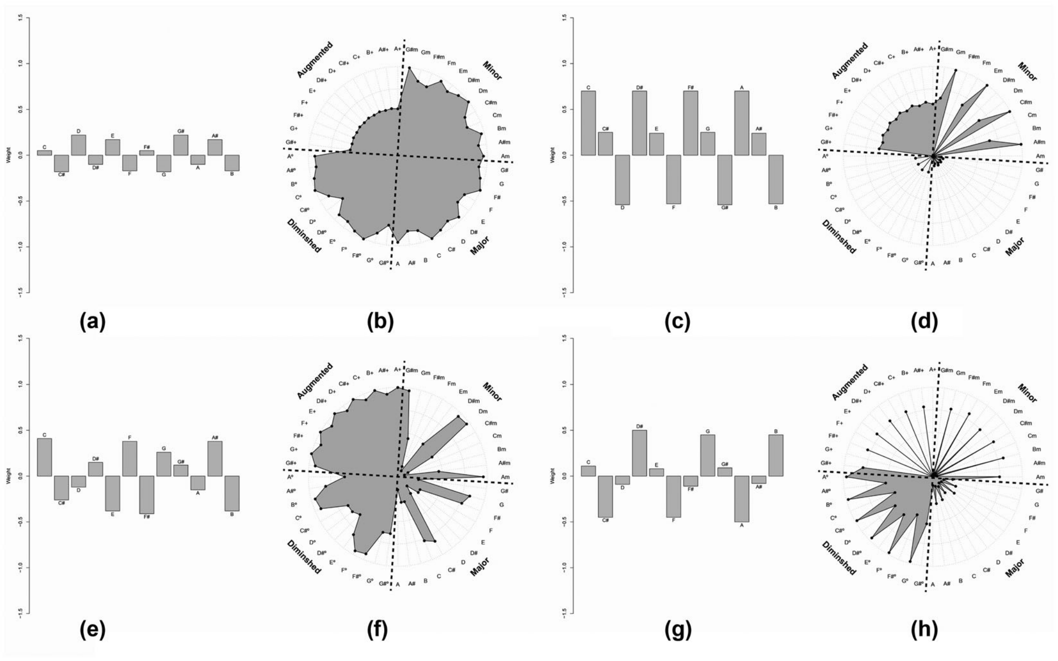

Fig. 5. Weights and stimulus responses for prototypical hidden value units belonging to each Table 3 cluster. (a) Weights for a Cluster 1 unit, whose responses are plotted in (b). (c) Weights for a Cluster 2 unit, whose responses are plotted in (d). (e) Weights for a Cluster 3 unit, whose responses are plotted in (f). (g) Weights for a Cluster 4 unit, whose responses are plotted in (h).

at which the plot exhibits an inflection or elbow. The elbow method indicates that our hidden value units are best described by four clusters. Table 3 provides each cluster's summary properties: the average correlation between weights and rotated weights for each possible rotation, where the average is taken over all units belonging to the same cluster.

Note. Each number is the average correlation between weights and rotated weights, where the amount of rotation is indicated in the table. Averages are taken across the *N* members of each cluster. The reason for naming each type of hidden unit is provided in the text.

When we examine graphs of the connection weights of hidden units assigned to the same cluster, we note consistent patterns. To illustrate the patterns, we choose a prototypical member of each cluster by finding the hidden unit whose weight correlations most closely match the correlations in Table 2. We present the weights and responses of the four prototypical units in Fig. 5. We use our examination of Fig. 5 to name each unit type in Table 3.

Cluster 1 hidden units represent what Dawson (2018) calls a *tritone equivalence unit*. In such a unit, connection weights from input units representing pitch-classes a tritone (six semitones) apart are assigned near identical weights (Fig. 5a). Note in the figure the connections weights from input units for C and F# are nearly identical, as are the weights from C# and G, and so on.

Cluster 2 hidden units represent another type of unit, a special case of a tritone equivalence unit. Cluster 2 hidden units have weights organized according to an interval cycle of a minor third: input units representing pitch-classes separated by a minor third (three semitones), which therefore belong to the same single interval cycle (Susanni & Antokoletz, 2012), are assigned nearly identical weights. For instance, in Fig. 5c, pitch-classes D, F, G#, and B, each belonging to one of the three cycles of minor thirds, are assigned similar negative weights. This type of unit is a special case of a tritone equivalence unit because pitch-classes a tritone apart also belong to the same cycle of minor thirds, and therefore have nearly identical weights.

Cluster 3 hidden units represent what Dawson (2018) calls a *tritone balance unit.* In such a unit, connection weights from input units representing pitch-classes a tritone (six semitones) apart are assigned weights of nearly identical magnitudes but of opposite signs (Fig. 5e). Note the bars for input units for C and F# are nearly identical but point in opposite directions, as do the bars for C# and G, and so on.

Cluster 4 hidden units represent a fourth type of unit, a special case of a tritone balance unit. Cluster 4 hidden units have weights organized according to an interval cycle of a major third. That is, input units representing pitch-classes separated by a major third (four semitones), which therefore belong to the same interval cycle (Susanni & Antokoletz, 2012), are assigned nearly identical weights. For instance, in Fig. 5g, pitch-classes C, E, and G#, which each belong to one of the four cycles of minor thirds, are assigned similar positive weights. Such a unit is a special case of a tritone balance unit because pitch-classes a tritone apart in Fig. 5g are equal in magnitude but opposite in sign.

Fig. 5 also provides radar plots of each type of hidden value unit's responses to each stimulus triad. Again, these radar plots reveal the broad tuning described in our earlier discussion of Fig. 4. However, there is also a surprising degree of musical systematicity within this broad tuning. For each Fig. 5 unit, and for each triad type, the unit generates identical responses to triads with roots separated by a tritone. A closer inspection of the Fig. 5 radar charts reveals each quadrant of each radar chart can be described as one pattern of responses that repeats twice.

The musical regularity within each radar plot is a direct consequence of the regular structure of connection weights (i.e., tritone equivalence or tritone balance) and the Gaussian activation function. For any two triads of the same type with roots a tritone apart, the other components of the triads will also be a tritone apart. For example, when considering C major triad (C, E, G) and the F# major triad (F#, A#, C#), C and F# are a tritone apart, as are E and A# and G and C#. In a tritone equivalence unit, connection weights from units representing pitch-classes a tritone apart are equal. Therefore, the net input from the C major triad and the net input from the F# major triad are identical. For a tritone balance unit, the two triads have the same magnitude weights, but weights have opposite signs, causing their net input to be the same magnitude by opposite signs. However, the Gaussian activation function produces identical values to two such net inputs (see Fig. 2a).

Table 3 and Fig. 5 indicate hidden units in our value unit networks belong to one of two different types, tritone equivalence units (Fig. 5a or c) or tritone balance units (Fig. 5e or g). We noted earlier that hidden units in our value unit networks can also be described as representing phase space projections from one of Yust's six different phase spaces (Table 1). What is the relationship between a hidden unit's phase space projections and a hidden unit's type?

Tritone balance unit weights represent phase space projections from odd phase spaces and are unrelated to phase space projections from even phase spaces. Tritone balance units such as Fig. 5a are most representative of Yust's Ph5 (mean fit = 0.81, *SD* = 0.13), while tritone balance units such as Fig. 5c are most representative of Yust's Ph3 (mean fit = 0.91, *SD* = 0.13). In contrast, tritone equivalence unit weights represent phase space projections from even phase spaces and are unrelated to phase space projections from odd phase spaces. Tritone equivalence units such as Fig. 5e are most representative of Yust's Ph6 (mean fit = 0.78, *SD* = 0.12), while tritone balance units such as Fig. 5g are most representative of Yust's Ph4 (mean fit = 0.95, *SD* = 0.07).

In general, what combinations of hidden unit types, or combinations of represented phase space sensitivities, are present in our 20 different value unit networks? With respect to unit types, 13 of our networks contain two tritone equivalence units and two tritone balance units. Five networks contain three tritone equivalence units and one tritone balance unit. The remaining two networks contain one tritone equivalence unit and three tritone balance units. With respect to represented phase spaces, 13 of our networks have two hidden units whose weights represent phase space projections from the same phase space (three have duplicate units for PH3, two have duplicate units for PH4, seven have duplicate units for PH5, and one has duplicate units for PH6). The remaining networks have four hidden units, which each represent projections from a different phase space.

In summary, after classifying hidden units, and relating hidden unit types to represented phase spaces, we find that each of our networks contains hidden units that capture only a few of the phase spaces defined by Yust (2016).

How do our networks use such sparse representations to classify triads? Some potential hypotheses can be rejected. For example, recognizing that our hidden units represent phase space projections, one might hypothesize our networks perform Fourier similar to those discussed by Amiot (2016). For instance, if hidden unit weights represent phase space projections, then perhaps weights between hidden units and output units represent magnitudes associated with different Fourier phase spaces, and networks reconstruct triad type by combining phases and magnitudes. However, such operations transform one distribution or vector into another, and hidden units are not powerful enough to transform distributions—they deliver single numbers (activity), not distributions. Furthermore, Fourier operations like the DFT produce different results for different stimuli. In contrast, our hidden units represent phase space projections that are applied to any to-be-classified triad—the same phase space representations are used to process each stimulus. Finally, while individual hidden units do not convert one distribution into another, perhaps distributions of Fourier components could be represented by the activities of a set of hidden units. However, such processing is unlikely because the value unit networks use only four hidden units, and in many networks, more than one of the hidden units represents projections from the same phase space.

What other hypotheses might we explore? The sparse nature of hidden unit representations in our networks suggests the networks reduce the dimensionality of the triad classification problem. To be more precise, each indicator vector can be represented as a point in a 12-dimensional pattern space; the coordinates of each point are given by input unit activities. Hidden unit activities condense the pattern space into a four-dimensional hidden unit space. Again, each stimulus is a point in the hidden unit space whose coordinates are provided by hidden unit activities. Furthermore, each hidden unit provides an axis in hidden unit space that represents the similarity between the hidden unit's phase space projections and the DFT of the stimulus. Thus, when hidden units condense the pattern space, they provide new positions for stimuli in the hidden unit space, which reflect the relationship between each stimulus and the four hidden unit axes.

In the next section, we show that the hidden unit axes (i.e., the learned phase space projections) permit stimuli to be positioned in hidden unit space in an arrangement that allows output units to identify triad types. We show all triads belonging to the same type are coplanar in hidden unit space. Different types of triads are captured by different planes. Each output unit classifies triads by determining whether a stimulus falls in a particular plane in hidden unit space. Thus, our networks demonstrate a novel use of phase space projections for classifying triads. We detail this novel use in the next section.

3.6. Coarse coding in networks of value units

How do output value units use such broadly tuned detectors to correctly classify triads? The output units employ *coarse coding*. Coarse coding is a distributed representation (Hinton, 1986; Van Gelder, 1991). A distributed representation uses activity from several hidden units to represent a single concept. ANNs use distributed representations when individual hidden unit activities do not accurately distinguish one concept from another (Churchland & Sejnowski, 1992; Hinton, McClelland, & Rumelhart, 1986). Even when individual hidden units are poor discriminators, combined activities can accurately represent one concept when each hidden unit represents information from different perspectives.

Earlier, we saw net inputs to hidden value units measure the similarity between the DFT structure represented by a unit's connection weights and the DFT structure of a triad. The Gaussian activation function converts measured similarity into a response that is high when there is little relationship between the DFT structures (e.g., zero correlation), which becomes smaller as the relationship becomes stronger (e.g., a more positive or a more negative correlation). We can view the activity caused in each hidden unit by a triad as the triad's coordinates in a four-dimensional hidden unit space.

Importantly, the musical systematicity within each hidden unit's broadly tuned responses produces one key property of triad coordinates: The coordinates of all triads belonging to the same type are coplanar. For instance, the four-dimensional coordinates produced by every major triad intersect the same three-dimensional hyperplane embedded in the four-dimensional hidden unit space.

The coplanarity of a set of four-dimensional coordinates is established by showing that one coordinate can be predicted from the other three (Carlson, 2014). To demonstrate coplanarity in our networks, we use multiple regression to predict the activity in Hidden Unit 1 from the activities produced in the other three hidden units. We perform four different regressions

Table 4
The average R^2 for predicting Hidden Unit 1 activity from the activities from the other three hidden units for each triad type. Each average is taken over 20 different regression equations (one equation per network)

Augmented	Diminished	Major	Minor
1.000	0.989	0.987	0.963
(0.001)	(0.035)	(0.039)	(0.095)

for each network: one for each triad type. As shown in Table 4, we discover we can nearly perfectly predict Hidden Unit 1 activity from the other three hidden unit activities for every triad type in every value unit network.

When in a network's output layer, value units carve two parallel and closely separated hyperplanes through a hidden unit space (Dawson, 2013). The weights between output units and hidden units define a hyperplane's position and orientation. If a stimulus' coordinates (i.e., hidden unit activities) fall between the two hyperplanes, then the output unit produces high activity. Otherwise, the output unit generates a weak response. In our triad networks, same-typed triads fall on the same hyperplane, which in turn lies between the two hyperplanes positioned by an output unit. For instance, the output unit for detecting major triads positions its hyperplanes, so only the (coplanar) major triads fall between them.

The coplanarity of same-type triads in hidden unit space depends upon the periodic structure of hidden unit weights (i.e., tritone equivalence, tritone balance), which in turn is represented by phase space projections. For instance, the periodic structure of connection weights causes musical systematicity (i.e., same-type triads whose roots are a tritone apart produce identical hidden unit responses). Musical systematicity makes it easier for hidden units to make same-type triads coplanar because only six different sets of triad coordinates need positioning, instead of 12.

3.7. Classifying types of hidden integration devices

Given our success in classifying hidden value units, we also use k-means clustering to classify hidden integration devices. We again represent each hidden unit as a vector of correlations between original weights and rotated weights. The elbow method (Thorndike, 1953) again indicates four clusters best describe our hidden integration devices. Table 5 provides each cluster's summary properties.

Note. Each number is the average correlation between weights and rotated weights, where the amount of rotation is indicated in the table. Averages are taken across the *N* members of each cluster. The identification of hidden unit type is detailed in the text.

Consistent patterns emerge when we examine the connection weights of hidden units in the same cluster. We illustrate these patterns in Fig. 6, which provide the weights and stimulus responses for four typical hidden units, one from each cluster. The properties of the illustrated units are most similar to the average properties for a cluster; each unit represents a cluster's prototype.

Table 5
Summary statistics for the four different clusters to which hidden integration devices were assigned using k-means clustering

			Rotation of Weights in Semitones											
Cluster	Type	*N*	0	1	2	3	4	5	6	7	8	9	10	11
1	Cycle of major seconds	36	1.00	−0.42	0.37	−0.61	0.37	−0.51	0.62	−0.51	0.37	0.61	0.37	−0.42
2	Cycle of minor thirds	57	1.00	−0.45	−0.40	0.79	−0.44	−0.40	0.80	−0.40	−0.44	0.79	−0.40	−0.45
3	Negative weights perfect fifth apart	38	1.00	−0.52	0.31	−0.07	−0.40	0.56	−0.76	0.56	−0.40	−0.07	0.31	−0.52
4	Cycle of major thirds	29	1.00	−0.11	−0.56	−0.10	0.53	0.02	−0.56	0.02	0.53	−0.10	−0.56	−0.11

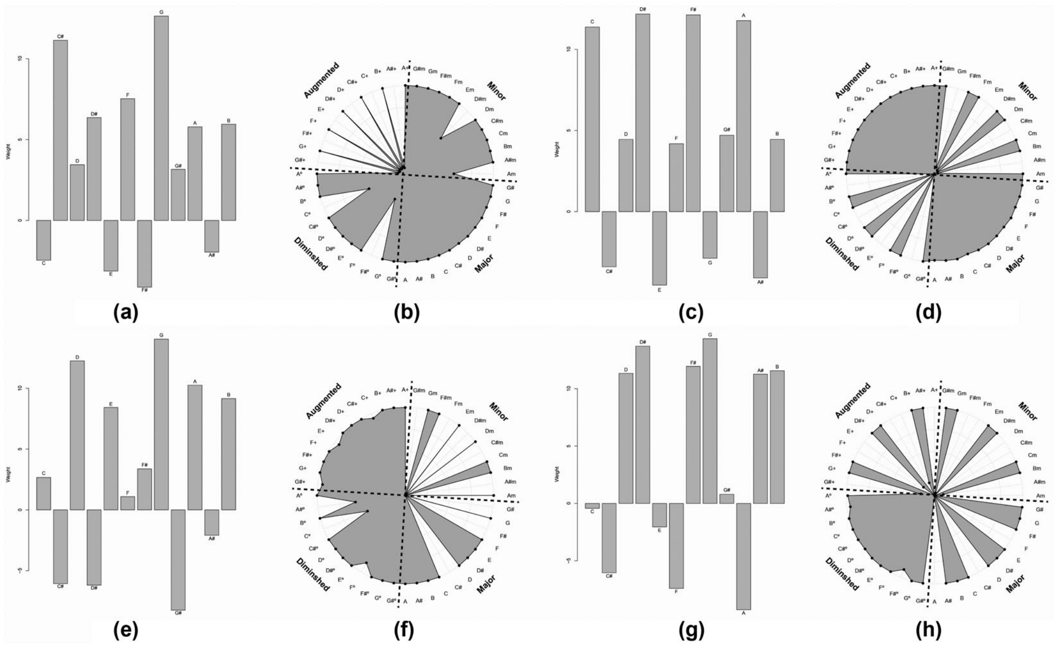

Fig. 6. Weights and stimulus responses for prototypical hidden integration devices belonging to each Table 5 cluster. (a) Weights for a Cluster 1 unit, whose responses are plotted in (b). (c) Weights for a Cluster 2 unit, whose responses are plotted in (d). (e) Weights for a Cluster 3 unit, whose responses are plotted in (f). (g) Weights for a Cluster 4 unit, whose responses are plotted in (h).

Cluster 1 hidden units relate to one type of unit described by Dawson (2018) in which weights are organized by interval cycles of a major second (Susanni & Antokoletz, 2012). In such units, one cycle of major seconds (C, D, E, F#, G#, A#) had connection weights from input units associated with one weight value, while the other cycle of major seconds (C#, D#, F, G, A, B) had connection weights associated with a similar weight value opposite in sign. The hidden integration devices rarely adopt such a striking organization. However, units belonging to Cluster 1 overwhelming organize connection weights so strong positive weights are associated with one cycle of major seconds, and strong negative weights are associated with the other. For example, in Fig. 6a, all six of the highest connection weights come from input units representing pitch-classes in one cycle of major seconds. All the weights coming from input units belonging to the other cycle are either weakly positive or more strongly negative.

To measure such organization, we take the absolute value of the difference between the sum of weights from units associated with one cycle of major seconds and the sum of weights associated with the other. For equally distributed positive and negative weights across the two cycles, the absolute value will be small. However, on average the absolute difference for Cluster 1 units is 58.95, while the average units of each other cluster have differences of 8.48, 7.18, and 19.79. The summary correlations for Cluster 1 (see Table 5) indicate more evidence for such organization. Note, every even rotation of the weights is associated with a moderate positive correlation, while every odd rotation of the weights is associated with a moderate negative correlation, only occurring if weights are organized by the two cycles of major seconds.

Cluster 2 hidden units have weights organized according to the interval cycles of minor thirds. For instance, in Fig. 6c one set of pitch-classes is associated with highly positive weights, a second is associated with moderately positive weights, and a third is associated with negative weights. Earlier, we identified many hidden value units as exhibiting such organization (e.g., compare Figs. 6c to 5c).

Cluster 3 hidden units have a characteristic pattern of spacing for negative weights. Typically, these units have two pairs of negative weights, the members separated by two semitones. For instance, in Fig. 6e, weights from C# and D# define one pair, while G# and A# define the other, with the pairs separated by five semitones (e.g., the distance between D# and G# in Fig. 6e). In the resulting organization, seven semitones (a perfect fifth) separated two. Thus, Cluster 3 units tend to turn off to particular major or minor triads because only these triads include pitch-classes separated by a perfect fifth. The phase information encoded in the unit's phase space projections determines the pitch-classes assigned negative connection weights, in turn determining which instances cause such behavior.

Cluster 4 hidden units have weights organized according to the interval cycles of major thirds. For instance, in Fig. 6g, input units associated with one of these four cycles (C, E, G#) have very small weights; another set (C#, F, A) has strong negative weights; a third set (D, F#, A#) has strong positive weights; and the fourth set (D#, G, B) has even stronger positive weights. Earlier, we identified many hidden value units as exhibiting such organization (e.g., compare Figs. 6g to 5g). This organization means pairs of negative weights are separated by a perfect fifth (e.g., C# and G#, D# and A#, D# and G#).

Fig. 6 also provides radar plots of the responses of each hidden integration device type to each stimulus triad. Again, these radar plots reveal the type of broad tuning described in our earlier discussion of Figs. 4 and 5. In addition, when we compare the Fig. 6 radar plots to Fig. 5 radar plots, hidden integration devices seem more likely to generate stronger responses to a larger number of triads than do hidden value units. We define an "off" response as activity less than 0.2 and an "on" response as activity greater than 0.8. We then count the number of "off"' and "on" responses generated by each hidden unit to the 48 stimuli. On average, hidden value units generate "off" responses to 14.11 triads and generate "on" responses to 14.63 triads. In contrast, hidden integration devices generate "off" responses to 12.34 triads and generate "on" responses to 32.54 triads. One general account of hidden integration devices then is they turn off to a small subset of triads but turn on to most others.

However, we again find musical systematicity within this broad tuning. For example, consider the hidden unit presented in Fig. 6e,f, with negative weights associated with C#, D#, G#, and A#, which generate strong responses to most triads. The unit does not respond to triads with negative weights. For instance, the unit generates near zero responses to the C#, D#, G#, and A# minor triads, each containing two pitch-classes associated with negative weights. For the same reason, the unit also generates weak responses to the C#, F#, and G# major triads. We can also find other musical regularities. For example, some (but not all) members of

Clusters 1, 3, and 4 are hidden units similar to those discussed in Fig. 5. For example, Fig. 6a,c is a noisy example of tritone balance units and generates similar responses to same-type triads with roots a tritone apart.

In short, hidden integration devices generally generate strong responses to many stimulus triads. However, the subset of triads to which a unit does *not* respond shares musical properties, typically because these triads include pitch-classes associated with negative weights, and the spacing of negative weights is systematic, as illustrated in Fig. 6.

3.8. Coarse coding in integration device networks

The previous section shows hidden integration devices are broadly tuned, making each individual hidden unit's activity, on its own, a poor indicator of triad type. As a result, integration device networks must also employ coarse coding to classify triads. However, because integration devices do not use the Gaussian activation function, a kind of coarse coding is used in comparison to the coarse coding described earlier for networks of value units.

The logistic activation function uses input signals as evidence to compute the conditional probability that some property is true given a set of input unit activities (Dawson, 2022; Dawson & Gupta, 2017; Dawson, Kelly, Spetch, & Dupuis, 2010). Evidence supporting the property's truth (i.e., increasing the probability of being true) is associated with a positive connection weight. Evidence not supporting the property's truth (i.e., decreasing the probability of being true) is associated with a negative connection weight. An integration device combines both kinds of evidence by computing net input; when net input is transformed by the logistic, the resulting activity is literally a conditional probability of a property being true.

The output units in an integration device network combine evidence provided by hidden units. Each hidden unit responds to a triad (e.g., Fig. 6); a hidden unit's response provides evidence. Each output unit weighs this evidence to generate a response (i.e., to classify a triad). If a hidden unit provides information supporting a response, then its connection to the output unit will have a strong positive weight. If a hidden unit provides information against a response, then its connection to the output unit will have a strong negative weight. An output unit will generate a strong response when the pattern of hidden unit activities is consistent with the connection weights: Hidden units associated with negative weights will have low activities, while hidden units associated with positive weights will have high activities. As the consistency between connection weights and hidden unit activities decreases, so too will the output unit's response. A sufficient amount of inconsistency will cause the output unit to turn off.

Table 6 illustrates how output unit responses depend upon the consistency between hidden unit responses and connection weights to the output units. It presents the connection weights from the eight hidden units to each of the four output units for an example network. Table 6 also presents the average correlation, taken over the 12 different triads of each triad type, between hidden unit responses and the output unit's connection weights. Note that for each output unit, the highest correlation is observed for triads belonging to the triad type being detected by the output unit. All other triad types produce weaker, and often negative, correlations.

Table 6
The average correlation between hidden unit responses and connection weights from hidden units to an output unit

	Hidden Unit Weights								Average Correlation By Stimulus Type			
Output Unit	H1	H2	H3	H4	H5	H6	H7	H8	Aug	Dim	Maj	Min
Aug	2.31	2.36	3.30	5.82	−1.26	−10.11	2.93	−11.00	0.59	−0.10	−0.33	−0.11
Dim	8.77	1.44	0.97	−10.67	−12.52	−2.09	8.80	−2.17	0.06	0.50	−0.02	−0.22
Maj	−5.51	−12.36	2.70	−6.92	−5.71	12.74	−6.51	12.98	−0.53	0.03	0.37	−0.05
Min	−10.15	3.96	−11.56	4.70	25.29	−4.89	−10.29	−5.17	0.11	−0.30	−0.09	0.30

Note. Averages are taken over the 12 different triads of each type.

One important property of Table 6 is each hidden unit always detects the same property because the hidden unit weights are constant after training and the same weights generate responses to each stimulus. The weights from the hidden units to the different output units indicate that the property detected by a hidden unit can provide strong evidence a triad belongs to one category and also provide strong evidence against a triad belonging to a different category. For instance, consider Hidden Units 6 and 8. Table 6 indicates the two units detect a property, which, if true, indicates a triad is not augmented because the two connection weights to the augmented output unit are extremely negative. However, the same property supports classifying the triad as major because the two connection weights to the major output unit are extremely positive.

4. Discussion

The current article explored how ANNs classify musical triads when triads are represented as indicator vectors. We replicated our discovery of Yust's (2016) phase space projections connection weights to hidden value units and extended this result to hidden integration devices.

We then considered how networks use phase space projections to classify triads. We discovered a hidden unit's net input (i.e., the inner product between a set of phase space projections and an indicator vector) measures the similarity between the DFT structure of the hidden unit's weights and the DFT structure of a triad. That is, if one performs a DFT on a hidden unit's weights, another on an indicator vector, and measures the similarity between the two DFTs, one can nearly perfectly predict the hidden unit's net input when presented with the stimulus. This is surprising and interesting because a hidden unit does *not* compute either DFT when net input is calculated—but it behaves *as if* the DFTs are available.

While both types of hidden units compute the same measure of DFT similarity, they respond differently to DFT similarity because of differences in activation functions (Fig. 2). Value units generate weak responses when DFT similarity is extreme and when DFT dissimilarity is extreme. Value units only generate strong responses when the hidden unit and stimulus DFT structure are unrelated. In contrast, integration devices generate higher responses as the similarity between the DFT structures increases.

Although both unit types respond differently to the same measure of DFT similarity, their responses share one key property: broad tuning. A single hidden unit's response to a stimulus is a poor indicator of triad type. First, different triads belonging to the same type may generate different responses in the same hidden unit (Figs. 4, 5, and 6). Second, triads belonging to different triad types may generate the same responses in the same hidden unit. As a result, our musical ANNs must use coarse coding, combining multiple hidden units' activity to classify triads.

The musical systematicity found with the broad tuning of hidden units aids the coarse codes we discover in our ANNs. Musical systematicity emerges from the periodicity of hidden unit weights, periodicity reflected in the fact hidden unit weights represent phase space projections strongly related to one phase space.

The hidden value unit weights are organized by tritone relations between pitch-classes. Weights associated with input units representing pitch-classes a tritone apart tend to be equal (tritone equivalence unit) or equal in magnitude but opposite in sign (tritone balance unit), causing same-type triads with roots a tritone apart to generate the same activity in a hidden unit (Fig. 5). The weights of hidden integration device are organized so the unit responds to most triads, but does not respond to a select few, because a small number of weights are negative. However, the spacing between negative weights is systematic (Table 5, Fig. 6), causing the unit to generate weak responses to a subset of triads sharing musical properties.

The different musical systematicities evident in different types of hidden units cause architectures to adopt alternative coarse codes. Responses of hidden value units position each stimulus as a point in a four-dimensional hidden unit space. All same-type triads have coplanar positions in hidden unit space (Table 4). Output value unit weights position two parallel hyperplanes in hidden unit space to detect if a stimulus intersects a particular hyperplane. In contrast, output integration devices treat each hidden unit response as providing evidence for a particular type of triad. An individual hidden unit's inaccurate evidence is overcome by combining evidence from all eight hidden units; output unit weights have values that combine evidence appropriately. Triads belonging to an output unit's category generate responses most consistent with the pattern of output unit weights.

Systematicity of broadly tuned responses permits efficient coarse coding. Value units place stimuli in hyperplanes in hidden unit space because pairs of triads (whose roots are a tritone apart) fall at the same location. Integration devices place stimuli in hidden unit space to permit optimal weighting of evidence provided by each hidden unit (Table 6).

In short, our musical networks discover DFT structure—phase space projections. Hidden units can be described as comparing the DFT structure of their weights to the DFT structure of a stimulus, *without* performing either DFT. The periodic nature of hidden unit weights permits an efficient coding of stimuli in hidden unit space, in turn permitting accurate coarse coding of triad type.

While our case study focuses on musical triads, we expect our results to extend to other musical networks. First, we have already discovered phase space projections in ANNs trained on other musical problems (Dawson et al., 2020). Second, connection weights in other ANNs trained on additional musical problems appear strongly related to phase space projections (Dawson, 2018), and we expect these weights can be easily related to phase space projections.

We believe our results provide both theoretical and empirical implications. Theoretically, our results indicate that musical Fourier representations are not limited to musical set theory. Critiques of musical set theory have deemed it too abstract and too technical to account for musical experience (Schuijer, 2008). However, music theorists hope the importance of the DFT extends to theories of musical cognition. For example, Amiot (2016, p. 179) argues Fourier space "is closest to our perception of music [...] since Fourier qualities seem to mirror exactly musical features processed by the human brain." Discovering Fourier phases spaces in musical ANNs suggests extensions of Fourier properties from musical set theory to musical cognition, suggesting a new representation to be explored by cognitive psychologists.

The empirical implications of our results concern how cognitive psychologists might explore Fourier representations in musical cognition. Our networks have revealed surprising relationships between responses to different triads, such as hidden value units generating identical responses to triads with roots a tritone apart. We suggest exploring Fourier representations in musical cognition by examining whether the musical systematicities described earlier also appear as human participants perceive, or learn to classify, musical entities.

Critically, the potential implications of our research only emerge when we examine and interpret ANN structure (Dawson, 2018). Unfortunately, researchers who study ANNs rarely determine how networks represent solutions to problems (Dawson, 2009). While some researchers focus on ANNs capturing informal musical properties (e.g., Bharucha, 1999), our case studies also discover surprising, formal, musical regularities. By examining musical networks, and finding Fourier phase spaces, we reveal exciting links between formal music theory and musical cognition.

References

Amiot, E. (2016). *Music through Fourier space: Discrete Fourier transform in music theory*. Cham: Springer International Publishing

Augasta, M. G., & Kathirvalavakumar, T. (2012). Reverse engineering the neural networks for rule extraction in classification problems. *Neural Processing Letters*, *35*(2), 131–150. https://doi.org/10.1007/s11063-011-9207-8

Baesens, B., Setiono, R., Mues, C., & Vanthienen, J. (2003). Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science*, *49*(3), 312–329. https://doi.org/10.1287/mnsc.49.3.312.12739

Ballard, D. (1986). Cortical structures and parallel processing: Structure and function. *The Behavioral And Brain Sciences*, *9*, 67–120.

Berkeley, I. S. N., Dawson, M. R. W., Medler, D. A., Schopflocher, D. P., & Hornsby, L. (1995). Density plots of hidden value unit activations reveal interpretable bands. *Connection Science*, *7*, 167–186.

Bharucha, J. J. (1995). Neural nets and music cognition. In R. Steinberg (Ed.), *Music and the mind machine* (pp. 99–103). Berlin: Springer.

Bharucha, J. J. (1999). Neural nets, temporal composites, and tonality. In D. Deutsch (Ed.), *The psychology of music* (2nd ed., pp. 413–440). San Diego, CA: Academic Press.

Bigand, E., & Poulin-Charronnat, B. (2006). Are we "experienced listeners"? A review of the musical capacities that do not depend on formal musical training. *Cognition*, *100*(1), 100–130. https://doi.org/10.1016/j.cognition.2005.11.007

Bullinaria, J. (1997). Analyzing the internal representations of trained neural networks. In A. Browne (Ed.), *Neural network analysis, architectures, and applications* (pp. 3–26). Philadelphia, PA: Institute of Physics.

Carlson, J. E. (2014). A generalization of Pythagoras's theorem and application to explanations of variance contributions in linear models. *ETS Research Report Series*, *2014*(2), 1–17. https://doi.org/10.1002/ets2.12018

Churchland, P. S., & Sejnowski, T. J. (1992). *The computational brain*. Cambridge, MA: MIT Press.

Dawson, M. R. W. (2004). *Minds and machines: Connectionism and psychological modeling*. Malden, MA: Blackwell Pub.

Dawson, M. R. W. (2005). *Connectionism : A hands-on approach* (1st ed.). Oxford, England: Blackwell Pub.

Dawson, M. R. W. (2009). Computation, cognition,—-and connectionism. In D. Dedrick & L. Trick (Eds.), *Cognition, computation, and Pylyshyn* (pp. 175–199). Cambridge, MA: MIT Press.

Dawson, M. R. W. (2013). *Mind, body, world: Foundations Of cognitive science*. Edmonton, Canada: Athabasca University Press.

Dawson, M. R. W. (2018). *Connectionist representations of tonal music: Discovering musical patterns by interpreting artificial neural networks*. Edmonton, Canada: Athabasca University Press.

Dawson, M. R. W. (2022). Probability learning by perceptrons and people. *Comparative Cognition and Behavior Reviews*, *15(Monograph)*, 1–188.

Dawson, M. R. W., & Gupta, M. (2017). Probability matching in perceptrons: Effects of conditional dependence and linear nonseparability. *Plos One*, *12*(2), e0172431. https://doi.org/10.1371/journal.pone.0172431

Dawson, M. R. W., Kelly, D. M., Spetch, M. L., & Dupuis, B. (2010). Using perceptrons to explore the reorientation task. *Cognition*, *114*(2), 207–226.

Dawson, M. R. W., Perez, A., & Sylvestre, S. (2020). Artificial neural networks solve musical problems with Fourier phase spaces. *Scientific Reports*, *10*(1), 7151. https://doi.org/10.1038/s41598-020-64229-4

Dawson, M. R. W., & Piercey, C. D. (2001). On the subsymbolic nature of a PDP architecture that uses a nonmonotonic activation function. *Minds and Machines*, *11*, 197–218.

Dawson, M. R. W., & Schopflocher, D. P. (1992). Modifying the generalized delta rule to train networks of nonmonotonic processors for pattern classification. *Connection Science*, *4*, 19–31.

Dutton, J. M., & Starbuck, W. H. (1971). *Computer simulation of human behavior*. New York: John Wiley & Sons.

Fan, F. - L., Xiong, J., Li, M., & Wang, G. (2021). On interpretability of artificial neural networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, *5*(6), 741–760. https://doi.org/10.1109/trpms.2021.3066428

Forte, A. (1973). *The structure of atonal music*. New Haven, CT: Yale University Press.

Gallant, S. I. (1993). *Neural network learning and expert systems*. Cambridge, MA: MIT Press.

Griffith, N. (1995). Connectionist visualization of tonal structure. *Artificial Intelligence Review*, *8*(5-6), 393–408. https://doi.org/10.1007/bf00849727

Griffith, N., & Todd, P. M. (1999). *Musical networks: Parallel distributed perception and performance*. Cambridge, MA: MIT Press.

Hanson, H. (1960). *Harmonic materials of modern music; resources of the tempered scale*. New York: Appleton-Century-Crofts.

Hayashi, Y., Setiono, R., & Yoshida, K. (2000). A comparison between two neural network rule extraction techniques for the diagnosis of hepatobiliary disorders. *Artificial Intelligence in Medicine*, *20*(3), 205–216. https://doi.org/10.1016/s0933-3657(00)00064-6

Hinton, G. E. (1986). Learning distributed representations of concepts. *The 8th Annual Meeting of the Cognitive Science Society*, Ann Arbor, MI.

Hinton, G. E., McClelland, J., & Rumelhart, D. (1986). Distributed representations. In D. Rumelhart & J. McClelland (Eds.), *Parallel distributed processing* (Vol. *1*, pp. 77–109). Cambridge, MA: MIT Press.

Huang, X., Kroening, D., Ruan, W., Sharp, J., Sun, Y., Thamo, E., Wu, M., & Yi, X. (2020). A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability? *Computer Science Review*, *37*, 100270. https://doi.org/10.1016/j.cosrev.2020.100270

Krumhansl, C. L., Toivanen, P., Eerola, T., Toiviainen, P., Jarvinen, T., & Louhivuori, J. (2000). Cross-cultural music cognition: cognitive methodology applied to North Sami yolks. *Cognition*, *76*(1), 13–58. https://doi.org/10.1016/s0010-0277(00)00068-8

Krumhansl, C. L., & Toiviainen, P. (2001). Tonal cognition. In R. J. Zatorre & I. Peretz (Eds.), *Biological foundations of music* (Vol. *930*, pp. 77–91). New York: New York Academy of Sciences.

Lewandowsky, S. (1993). The rewards and hazards of computer simulations. *Psychological Science*, *4*, 236–243.

Lewin, D. (2007). *Generalized musical intervals and transformations*. New York: Oxford University Press.

McCloskey, M. (1991). Networks and theories: The place of connectionism in cognitive science. *Psychological Science*, *2*, 387–395.

Mihelac, L., & Povh, J. (2020). AI based algorithms for the detection of (ir)regularity in musical structure. *International Journal of Applied Mathematics and Computer Science*, *30*(4), 761–772. 10.34768/amcs-2020-0056

Milne, A. J., Bulger, D., & Herff, S. A. (2017). Exploring the space of perfectly balanced rhythms and scales. *Journal of Mathematics and Music*, *11*(2-3), 101–133. https://doi.org/10.1080/17459737.2017.1395915

Moorhead, I. R., Haig, N. D., & Clement, R. A. (1989). An investigation of trained neural networks from a neurophysiological perspective. *Perception*, *18*, 793–803.

Mozer, M. C., & Smolensky, P. (1989). Using relevance to reduce network size automatically. *Connection Science*, *1*, 3–16.

Omlin, C. W., & Giles, C. L. (1996). Extraction of rules from discrete-time recurrent neural networks. *Neural Networks*, *9*, 41–52.

Quinn, I. (2006). General equal-tempered harmony: Introduction and Part 1. *Perspectives of New Music*, *44*, 114–158.

Quinn, I. (2007). General equal-tempered harmony: Parts 2 and 3. *Perspectives of New Music*, *45*, 4–63.

Robinson, D. A. (1992). Implications of neural networks for how we think about brain function. *Behavioral and Brain Sciences*, *15*(4), 644–655.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.

Russo, F. A., Vempala, N. N., & Sandstrom, G. M. (2013). Predicting musically induced emotions from physiological inputs: Linear and neural network models. *Frontiers in Psychology*, *4*, 468. https://doi.org/10.3389/fpsyg.2013.00468

Schuijer, M. (2008). *Analyzing atonal music: Pitch-class set theory and its contexts*. Rochester, NY: University of Rochester Press.

Seidenberg, M. (1993). Connectionist models and cognitive theory. *Psychological Science*, *4*, 228–235.

Setiono, R., Baesens, B., & Mues, C. (2011). Rule extraction from minimal neural networks for credit card screening. *International Journal of Neural Systems*, *21*(4), 265–276. https://doi.org/10.1142/s0129065711002821

Setiono, R., Thong, J. Y. L., & Yap, C. S. (1998). Symbolic rule extraction from neural networks: An application to identifying organizations adopting IT. *Information & Management*, *34*(2), 91–101. https://doi.org/10.1016/s0378-7206(98)00048-2

Stevens, C., & Latimer, C. (1992). A comparison of connectionist models of music recognition and human-performance. *Minds and Machines*, *2*(4), 379–400. https://doi.org/10.1007/bf00419420

Susanni, P., & Antokoletz, E. (2012). *Music and twentieth-century tonality: Harmonic progression based on modality and the interval cycles*. New York: Routledge.

Swingler, K. (2014). Opening the black box: Analysing MLP functionality using Walsh functions. *6th International Joint Conference on Computational Intelligence (IJCCI)*, Rome, Italy.

Taha, I. A., & Ghosh, J. (1999). Symbolic interpretation of artificial neural networks. *IEEE Transactions on Knowledge and Data Engineering*, *11*(3), 448–463. https://doi.org/10.1109/69.774103

Temperley, D. (2013). Computational models of music cognition. In D. Deutsch (Ed.), *The psychology of music* (3rd ed., pp. 327–368). San Diego, CA: Elsevier Academic Press.

Thorndike, R. L. (1953). Who belongs in the family? *Psychometrika*, *18*(4), 267–276.

Tillmann, B., Bharucha, J. J., & Bigand, E. (2000). Implicit learning of tonality: A self-organizing approach. *Psychological Review*, *107*(4), 885–913. https://doi.org/10.1037//0033-295x.107.4.885

Tillmann, B., Bharucha, J. J., & Bigand, E. (2003). Learning and perceiving musical structures: Further insights from artificial neural networks. In I. Peretz & R. Zatorre (Eds.), *The cognitive neuroscience of music* (pp. 109–123). Oxford, England: Oxford University Press.

Tillmann, B., & Bigand, E. (2004). The relative importance of local and global structures in music perception. *Journal of Aesthetics and Art Criticism*, *62*(2), 211–222. https://doi.org/10.1111/j.1540-594X.2004.00153.x

Todd, P. M., & Loy, D. G. (1991). *Music and connectionism*. Cambridge, MA: MIT Press.

Van Gelder, T. (1991). What is the "D" in "PDP"? A survey of the concept of distribution. In W. Ramsey, S. P. Stich, & D. E. Rumelhart (Eds.), *Philosophy and connectionist theory* (pp. 33–59). Hillsdale, NJ: Lawrence Erlbaum Associates.

Yust, J. (2016). Special collections: Renewing set theory. *Journal of Music Theory*, *60*(2), 213–262. https://doi.org/10.1215/00222909-3651886

Yust, J. (2017a). Harmonic qualities in Debussy's "Les sons et les parfums tournent dans l'air du soir." *Journal of Mathematics and Music*, *11*(2-3), 155–173. https://doi.org/10.1080/17459737.2018.1450457

Yust, J. (2019). Stylistic information in pitch-class distributions. *Journal of New Music Research*, *48*(3), 217–231. https://doi.org/10.1080/09298215.2019.1606833

Zhang, Y., Tino, P., Leonardis, A., & Tang, K. (2021). A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, *5*(5), 726–742. https://doi.org/10.1109/tetci.2021.3100641

Zhang, Z., Beck, M. W., Winkler, D. A., Huang, B., Sibanda, W., Goyal, H., & Written on behalf of AME Big-Data Clinical Trial Collaborative Group . (2018). Opening the black box of neural networks: Methods for interpreting neural network models in clinical applications. *Annals of Translational Medicine*, *6*(11), 216. 10.21037/atm.2018.05.32