Identity-Based Patterns in Deep Convolutional Networks: Generative Adversarial Phonology and Reduplication

Gašper Beguš
University of California, Berkeley
begus@berkeley.edu

Abstract

This paper models unsupervised learning of an identity-based pattern (or copying) in speech called reduplication from raw continuous data with deep convolutional neural networks. We use the ciwGAN architecture (Beguš, 2021a) in which learning of meaningful representations in speech emerges from a requirement that the CNNs generate informative data. We propose a technique to wug-test CNNs trained on speech and, based on four generative tests, argue that the network learns to represent an identity-based pattern in its latent space. By manipulating only two categorical variables in the latent space, we can actively turn an unreduplicated form into a reduplicated form with no other substantial changes to the output in the majority of cases. We also argue that the network extends the identity-based pattern to unobserved data. Exploration of how meaningful representations of identity-based patterns emerge in CNNs and how the latent space variables outside of the training range correlate with identity-based patterns in the output has general implications for neural network interpretability.

1 Introduction

The relationship between symbolic representations and connectionism has been subject to ongoing discussions in computational cognitive science. Phonology offers a unique testing ground in this debate as it is concerned with the first discretization that human language users perform: from continuous phonetic data to discretized mental representations of meaning-distinguishing sounds called *phonemes*.

Identity-based patterns, repetition, or copying have been at the center of this debate (Marcus et al., 1999). Reduplication is a morphophonological process where phonological content (phonemes) get copied from a word (also

called the base) with some added meaning (Inkelas and Zoll, 2005; Urbanczyk, 2017). It can be total, which means that all phonemes in a word get copied (e.g. /pula/ → [**pula**-pula]), or partial, where only a subset of segments gets copied (e.g. /pula/ → [**pu**-pula]).

Reduplication is indeed unique among processes in natural language because combining learned entities based on training data distributions does not yield the desired outputs. For example, a learner can be presented with pairs of bare and reduplicated words, such as /pala/ ∼ /papala/ and /tala/ ∼ /tatala/. The learner can then be tested on providing a reduplicated variant of a novel unobserved item with an initial sound /k/ that they have not been exposed to (e.g. /kala/). If the learner learns the reduplication pattern, they will output [kakala]. If the learner simply learns that /pa/ and /ta/ are optional constituents that can be attached to words based on data distribution, they will output [pakala] or [takala]. Reduplication is thus an identity-based pattern (similar to copying), which is computationally more challenging to learn (Gasser, 1993), both in connectionist (Brugiapaglia et al., 2020) and non-connectionist frameworks (Savitch, 1989; Dolatian and Heinz, 2020). In /$k_i a_j k_i a_j$la/, the two sounds in the reduplicative morpheme, /k_i/ and /a_j/, need to be in an identity relationship with the first two segments of the base, /k_i/ and /a_j/, and the learner needs to copy rather than recombine learned elements.

Marcus et al. (1999) argue that connectionist models such as simple neural networks are unable to learn a simple reduplication pattern that 7-month old human infants are able to learn (see also Gasser 1993). According to Marcus et al. (1999), the behavioral outcomes of their experiments cannot be modeled by simple counting, attention to statistical trends in the input, attention to repetition, or connectionist (simple neural network) computational models. Instead, they argue, the re-

sults support the claim that human infants employ "algebraic rules" (Marcus et al., 1999; Marcus, 2001; Berent, 2013) to learn reduplication patterns (for a discussion, see, among others, McClelland and Plaut 1999; Endress et al. 2007).

With the development of neural network architectures, several studies revisited the claim that neural networks are unable to learn reduplicative patterns (Alhama and Zuidema, 2018; Prickett et al., 2018; Nelson et al., 2020; Brugiapaglia et al., 2020), arguing that identity-based patterns can indeed be learned with more complex architectures.[1] All these computational experiments, however, operate on an already discretized level and most of these experiments model reduplication with supervised learning.

Examples like [**pu**-pula] and [**pula**-pula] are discretized representations of reduplication, using characters to represent continuous sounds. Most, if not all, computational models of reduplication, to the author's knowledge, model reduplication as character or feature manipulation (the inputs to the models are either characters representing phones or phonemes or discrete binary featural representations of phonemes). For example, a seq2seq model treats reduplication as a pairing between the input unreduplicated sequence of "characters" (such as /tala/) and an output — a reduplicated sequence (such as /tatala/). Already abstracted and discretized phonemes or "characters", however, are not the primary input to language-learning infants. The primary linguistic data of most hearing infants is raw continuous speech. Hearing infant learners need to acquire reduplication from continuous speech data that is substantially more complex than already discretized characters or binary features.

Furthermore, most of the existing models of reduplication are also supervised. Seq2seq models, for example, are fully supervised: the training consists of pairs of unreduplicated (input) and reduplicated strings of characters or binary features (output). While the performance can be tested on unobserved data or even on unobserved segments, the training is nevertheless supervised. Human language learners do not have access to input-output pairings: they are only presented with positive, surface, and continuous acoustic data.

While equivalents of copying/identity-based patterns can be constructed in the visual domain, we are not aware of studies that test identity-based visual patterns with deep convolutional neural networks.

In this paper, we model reduplication, one of the computationally most challenging processes, from raw unlabeled acoustic data with deep convolutional networks in the GAN framework. The advantage of the GAN framework for cognitive modeling is that the network has to learn to output raw acoustic data from a latent noise distribution without directly accessing the training data. We argue that CNNs discretize continuous phonetic data and encode linguistically meaningful units into individual latent variables. The emergence of a discretized representation of an identity-based pattern (reduplication) is induced by a model which forces the Generator network to output informative data (ciwGAN; Section 4). Additionally, we add inductive bias towards symbolic-like representations by binarizing code variables with which the Generator encodes meaningful representations. We also test whether a deep convolutional network learns reduplication without the two inductive biases (without the requirement on the Generator to output informative data and without binarization of the latent space) in the bare WaveGAN architecture (Section 5).

The experiments bear implications for the discussion between symbolic and connectionist approaches to language modelling by testing the emergence of rule-like symbolic representations within the connectionist framework from raw speech in an unsupervised manner. Results of the experiments suggest that both models, ciwGAN and WaveGAN learn the identity-based patterns, but inductive biases for informative representation and binarization facilitate learning and yield better results. We discuss properties of symbolic-like representations and how they emerge in the models: discretization, causality (in the sense that manipulation of individual elements results in desired outcome), and categoricity (for discussions on these and other aspects of the debate, see Rumelhart et al. 1986; McClelland et al. 1986; Fodor and Pylyshyn 1988; Minsky 1991; Dyer 1991; Marcus et al. 1999; Marcus 2001; Manning 2003; Berent 2013; Maruyama 2021).

How can we test learning of reduplication in a deep convolutional network that is trained only

---

[1]Wilson (2018, 2020) proposes another approach that allows modeling reduplication. For a non-connectionist computational model of reduplication, see Dolatian and Heinz (2018, 2020).

on raw positive data? We propose a technique to test the ability of the Generator to produce forms absent from the training data set. For example, we train the networks on acoustic data of items such as /pala/ ∼ /papala/ and /tala/ ∼ /tatala/, but test reduplication on acoustic forms of items such as /sala/, which is never reduplicated in the training data. Using the technique proposed in Beguš (2020), we can identify latent variables that correspond to some phonetic or phonological representation such as reduplication. By manipulating and interpolating a single latent variable, we can actively generate data with and without reduplication. In fact, we can observe a direct relationship between a single latent variable (out of 100) and reduplication that with interpolation gradually disappears from the output. Additionally, we can identify latent variables that correspond to [s] in the output. By forcing both reduplication and [s] in the output through latent space manipulation, we can "wug-test" the network's learning of reduplication on unobserved data. In other words, we can observe what the network will output if we force it to output reduplication and an [s] at the same time. A comparison of generated outputs with human outputs that were withheld from training reveals a high degree of similarity. We perform an additional computational experiment to replicate the results from the first experiment (from Section 4). In the replication experiment, evidence for learning of the reduplicative pattern also emerges. To the author's knowledge, this is the first attempt to model reduplication with neural network architectures trained on raw acoustic speech data.

The computational experiments reveal another property about representation learning in deep neural networks: we argue that the network extracts information in the training data and represents a continuous acoustic identity-based pattern with discretized representation. Out of 100 variables, the network encodes reduplication with one or two variables, which is suggested by the fact that a small subset of variables are substantially more strongly correlated with presence of reduplication. In other words, there is a near categorical drop in regression estimates between one variable and the rest of the latent space. Setting the identified variables to values well beyond the training range results in near categorical presence of a desired variable in the output. This technique (pro-

posed for non-identity-based patterns in Beguš 2020) allows us to directly explore how the networks encode dependencies in data, their underlying values, and interactions between variables, and thus get a better understanding of how exactly deep convolutional networks encode meaningful representations.

Recent developments in zero-resource speech modeling (Dunbar et al., 2017, 2019, 2020) enable modeling of speech processes in an unsupervised manner from raw acoustic data. Several proposals exist for modeling unsupervised lexical learning (Kamper et al., 2014; Lee et al., 2015; Chung et al., 2016) that include generative models such as variational autoencoders (Chung et al., 2016; Baevski et al., 2020; Niekerk et al., 2020) and GANs (Beguš, 2021a). This framework allows not only unsupervised lexical term discovery, but also phone-level identification (Eloff et al., 2019; Shain and Elsner, 2019; Chung et al., 2016; Chorowski et al., 2019). While zero-resource speech modeling has yielded promising results in unsupervised labeling, the proposals generally do not model phonological or morphophonological processes. This paper thus also tests applicability of the unsupervised speech processing framework for cognitive modeling and network interpretability.

2 Model

Generative Adversarial Networks (Goodfellow et al., 2014) are a neural network architecture with two main components: the Generator network and the Discriminator network. The Generator is trained on generating data from some latent space that is randomly distributed. The Discriminator takes real training data and the Generator's outputs and estimates which inputs are real and which are generated. The minimax training, where the Generator is trained on maximizing the Discriminator's error rate and the Discriminator is trained on minimizing its own error rate, results in the Generator network outputting data such that the Discriminator's success in distinguishing them from real data is low. It has been shown that GANs not only learn to produce innovative data that resemble speech, but also learn to encode phonetic and phonological representations in the latent space (Beguš, 2020). The major advantage of the GAN architecture for modeling speech is that the Generator network does not have direct access to the training data and is not trained on replicat-

ing data (unlike in the autoencoder architecture; Räsänen et al. 2016; Eloff et al. 2019; Shain and Elsner 2019). Instead, the network has to learn to generate data from noise in a completely unsupervised manner — without ever directly accessing the training data.

In the first experiment, we use the ciwGAN (Categorical InfoWaveGAN) model proposed in Beguš (2021a). The ciwGAN model combines the WaveGAN and InfoGAN architectures. WaveGAN, proposed by Donahue et al. (2019), is a Deep Convolutional Generative Adversarial Network (DCGAN; proposed by Radford et al. 2016) adapted for time-series audio data. The basic architecture is the same as in DCGAN, the main difference being that in the WaveGAN proposal, the deep convolutional networks take one-dimensional time-series data as inputs or outputs. The structure of the Generator and the Discriminator networks in the ciwGAN architecture are taken from Donahue et al. (2019). InfoGAN (Chen et al., 2016) is an extension of the GAN architecture that aims to maximize mutual information between the latent space and generated outputs. The Discriminator/Q-network learns to retrieve the Generator's latent categorical or continuous codes (Chen et al., 2016) in addition to estimating realness of generated outputs and real training data.

Beguš (2021a) proposes a model that combines these two proposals and introduces a new latent space structure (in the fiwGAN architecture). Because we are primarily interested in simple binary classification between bare and reduplicated forms, we use the ciwGAN variant of the proposal. The model introduces a separate deep convolutional Q-network that learns to retrieve the Generator's internal representations. Separating the Discriminator and the Q-network into two networks is advantageous from the cognitive modeling perspective: the architecture features a separate network that models speech production (the Generator) and a separate network that models speech categorization (the Q-network). The latter introduces an inductive bias that forces the Generator to output informative data and encode linguistically meaningful properties into its code variables. The network learns to generate data such that by manipulating these code variables, we can force the desired linguistic property in the output (Beguš, 2021a).

The architecture involves three networks: the Generator that takes latent codes (a one-hot vector) and uniformly distributed z-variables and generates waveforms, a Discriminator that distinguishes real from generated outputs, and a Q-network that takes generated outputs and estimates the latent code (one-hot vector) used by the Generator. More specifically, the Generator network is a deep convolutional network that takes as its input 100 latent variables (see Figure 1).[2] Two of the 100 variables are code variables (c_1 and c_2) that constitute a one-hot vector. The remaining 98 z-variables are uniformly distributed on the interval $(-1, 1)$. The Generator learns to take as the input the 2 code variables and the 98 latent variables and output 16384 samples that constitute just over one second of audio file sampled at 16 kHz through five convolutional layers. The Discriminator network takes real and generated data (again in the form of 16384 samples that constitute just over one second of audio file) and learns to estimate the Wasserstein distance between generated and real data (according to the proposal in Arjovsky et al. 2017) through five convolutional layers. In the majority of InfoGAN proposals, the Discriminator and the Q-network share convolutions. Beguš (2021a) introduces a separate Q-network (also in Rodionov 2018).[3]

The Q-network is in its structure identical to the Discriminator network, but the final layer is fully connected to nodes that correspond to the number of categorical variables (Beguš, 2021a). In the ciwGAN architecture, the Q-network is trained on estimating the latent code variables with a softmax function (Beguš, 2021a). In other words, the Q-network takes the Generator's outputs (waveforms) and estimates the Generator's latent code variables c_1 and c_2. Weights of both the Generator network and the Q-network are updated according to the Q-network's loss function: to minimize the distance between the actual one-hot vector (c_1 and c_2) used by the Generator and the one-hot vector estimated with a softmax in the Q-network's final layer using cross-entropy. This forces the Generator to output informative data.

The advantage of the ciwGAN architecture is that the network not only learns to output innova-

---

[2]The number of latent variables were adopted from Radford et al. (2016) and Donahue et al. (2019). Probing how the number of z-variables affects learning of speech representations is left for future work.

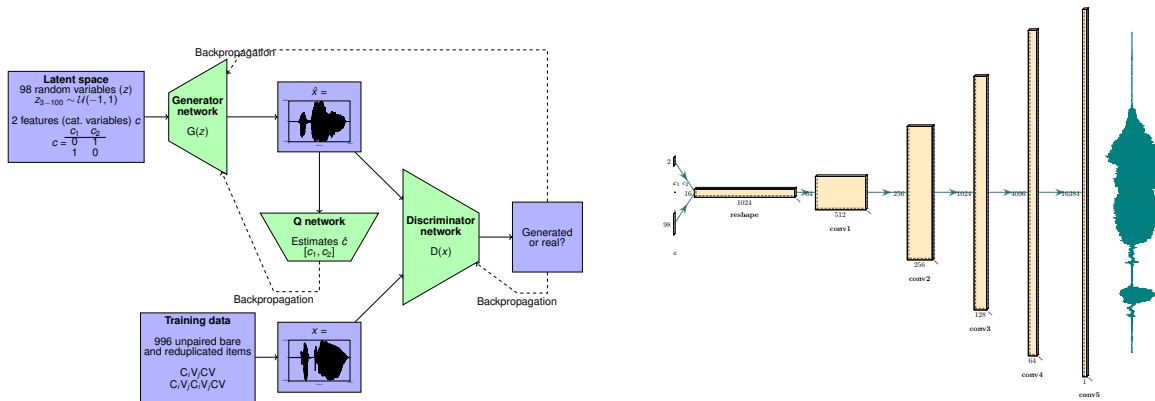[3]For all details about the architecture, see Beguš (2021a).

Figure 1: (left) The ciwGAN architecture as proposed in Beguš (2021a) and used in this paper with training data as described in Section 3. (right) The structure of the Generator in the ciwGAN architecture as proposed in Beguš (2021a) (based on Donahue et al. 2019).

tive data that resemble speech in the input, but also provides meaningful representations about data in an unsupervised manner. For example, as will be argued in Section 4, the ciwGAN network encodes reduplication as a meaningful category: it learns to assign a unique code for bare and reduplicated items. This encoding emerges in an unsupervised fashion from the requirement that the Generator output data such that unique information is retrievable from its acoustic outputs. Given the structure of the training data, the Generator is most informative if it encodes presence of reduplication in the code variables.

To replicate the results and to test learning of an identity-based pattern without binarization and without the requirement on the Generator to output informative data, we run an independent experiment on a bare WaveGAN (Donahue et al., 2019) architecture using the same training data. The difference between the two architectures is that the bare GAN architecture does not involve a Q-network and the latent space only includes latent variables uniformly distributed on the interval $(-1, 1)$.

Beguš (2020) and Beguš (2021a) also propose a technique for latent space interpretability in GANs: manipulating individual variables to values well beyond the training range can reveal underlying representations of different parts of the latent space. We use this technique throughout the paper to evaluate learning of reduplication.

3 Reduplication in training data

The training data was constructed to test a simple reduplication pattern, common in human lan-

guages: partial CV reduplication found in languages such as Paamese, Roviana, Tawala, among others (Inkelas and Zoll, 2005). Base items are of the shape $C_1V_2C_3V_4$ (C = consonant; V = vowel), e.g. /tala/. Reduplicated forms are of the shape $C_1V_2C_1V_2C_3V_4$, where the first syllable (C_1V_2) is repeated. The items were constructed so that C_1 contains a voiceless stop /p, t, k/, a voiced stop /b, d, g/, a labiodental voiced fricative /v/, and nasals /m, n/. The vowels V_2 and V_4 consist of /ɑ (ə), i, u/. C_3 consists of /l, ɹ, j/. All permutations of these elements were created. The stress was always placed on V_2 in the base forms and on the same syllable in reduplicated forms (['pʰɑlə] ~ [pə'pʰɑlə]). Because the reader of the training data was a speaker of American English, the training data is phonetically even more complex. The major phonetic effects in the training data include (i) reduction of the vowel in the unstressed reduplicated forms and in the final syllable (e.g. from [ɑ] to [ʌ/ə]) and (ii) deaspiration of voiceless stops in the unstressed reduplication syllable (e.g. from [pʰ] to [p]). The training data includes two unique repetitions of each item and two repetitions of the corresponding reduplicated forms. Table 1 illustrates the training data.

The training data also includes base forms $C_1V_2C_3V_4$ with the initial consonant C_1 being a fricative [s]. These items, however, always appear unreduplicated in the training data — the purpose of [s]-initial item is to test how the network extends the reduplicative pattern to novel unobserved data. All 27 permutations of $sV_2C_3V_4$ were included. To increase representation of [s]-initial words, four or five repetitions of each unique [s]-

initial base were used in training.[4] Altogether 132 repetitions of the 27 unique unreduplicated words with an initial [s] were used in training.

Sibilant fricative [s] was chosen as C_1 for testing learning of reduplication because its frication noise is acoustically prominent and sufficiently different from C_1s in the training data both acoustically and phonologically. This satisfies the requirement that a model learns to generalize to novel segments and feature values (Berent, 2013; Prickett et al., 2018).[5] In phonological terms, the model is tested on a novel feature (sibilant fricative or [±strident]; Hayes 2009) — the training data did not consist of any bare or reduplicated forms with other sibilant fricatives. To make the learning even more complex, voiceless fricatives ([f, θ, ʃ]) are altogether absent from the training data. All voiced fricatives except for [v] are absent too. Spectral properties of the voiced non-sibilant fricative [v] in the training data (and in Standardized American English in general) are so substantially different from a voiceless sibilant fricative [s] that we kept them in the training data. We excluded all items with initial sequences /ti/, /tu/, and /ki/ from the training data, because acoustic properties of these sequences, especially frication of the aspiration of /t/ and /k/, are similar to those of frication noise in /s/. Altogether 996 unique sliced items used in training were recorded in a sound attenuated booth by a female speaker of American English with a MixPre 6 (SoundDevices) preamp/recorder and the AKG C544L head-mounted microphone.

4   CiwGAN (Beguš, 2021a)

The Generator features two latent code variables, c_1 and c_2 and 98 uniformly distributed variables z (Figure 1). In the training phase, the two code variables (c_1 and c_2) compose the one-hot vector with two levels: [0, 1] and [1, 0]. This means that the network can encode two categories in its latent space structure that correspond to some meaning-

---

[4] Items [ˈsala], [ˈsuru], and [ˈsuju] each miss one repetition (four altogether).

[5] For an "across the board" generalization, Berent (2013) requires that generalization occur to segments fully absent from the inventory. It is challenging to elicit reduplication of segments that are fully absent from the training data in the proposed models. Even in human subject experiments testing the "across the board" generalization, subjects need to be exposed to the novel segment at least as a prompt. In our case, the novel segment needs to be part of the training data, but only in unreduplicated forms.

voiceless C_1	$C_1V_2C_3V_4$	ˈpʰɑli
	$C_1V_2C_1V_2C_3V_4$	pʌˈpʰɑli
voiced C_1	$C_1V_2C_3V_4$	ˈbɑli
	$C_1V_2C_1V_2C_3V_4$	bʌˈbɑli
C_1 = [m, n, v]	$C_1V_2C_3V_4$	ˈmɑli
	$C_1V_2C_1V_2C_3V_4$	mʌˈmɑli
C_1 = [s]	$C_1V_2C_3V_4$	ˈsɑli
	$C_1V_2C_1V_2C_3V_4$	—

Table 1: A schematic illustration of the training data in the International Phonetic Alphabet.

ful feature about the data. The Q-network forces the Generator to encode information in its latent space. In other words, the loss function of the Q-network forces the Generator to output data such that the Q-network is effective in retrieving the latent code c_1 and c_2 from the Generator's acoustic outputs only. Nothing in the training data pairs base and reduplicated forms. There is no overt connection between the bases and their reduplicated correspondents. Yet, the structure of the data is such that given two categories, the most informative way for the Generator to encode unique information in its acoustic outputs is to associate one unique code with base forms and another with reduplicated forms. The Generator would thus have a meaningful unique representation of reduplication that arises in an unsupervised manner exclusively from the requirement on the Generator to output informative data.

To test whether the Generator encodes reduplication in latent codes, we train the network for 15,920 steps (or approximately 5,114 epochs) with the data described in Section 3. The choice of the number of steps is based on two objectives; first, the output data should approximate speech to the degree that allows acoustic analysis. Second, the Generator network should not be trained to the degree that it replicates data completely. As such, overfitting rarely occurs in the GAN architecture (Adlam et al., 2019; Donahue et al., 2019). The best evidence against overfitting in the ciwGAN architecture comes from the fact that the Generator outputs data that violate training distributions substantially (see Section 4.2 below) (Beguš, 2021a,b). Despite these guidelines, the choice of number of steps is somewhat arbitrary (for discussion, see Beguš 2020).

We generate 100 outputs for each latent code [0, 1] and [1, 0] (200 total) and annotate them for

Code	Bare	Redup.	% Redup.
[1, 0]	78	22	22%
[0, 1]	40	60	60%
[5, 0]	98	2	2%
[0, 5]	13	87	87%

Table 2: Counts of bare and reduplicated (redup.) outputs when the latent codes c_1 and c_2 are set to [1, 0], [0, 1], [5, 0], and [0, 5].

presence or absence of reduplication. All annotations here and in other sections are performed by the author in Praat (Boersma and Weenink, 2015). Distinguishing unreduplicated from reduplicated is very salient; for less salient annotations, we provide waveforms and spectrograms (e.g. Figures 4 and 6).[6]

There is a significant correlation between the two levels of latent code and presence of reduplication. Counts are given in Table 2. When the code is set to [1, 0], 78% of the generated outputs are base forms; when set to [0, 1], 60% of outputs are reduplicated (odds ratio = 5.27, $p < 0.0001$, Fisher Exact Test). When the latent codes are set to [0, 5] and [5, 0], we get a near categorical distribution of bare and reduplicated forms. For [5, 0], the Generator outputs an unreduplicated bare form in 98% samples. For [0, 5], it outputs a reduplicated form in 87% outputs (odd ratio = 308.3, $p < 0.0001$, Fisher Exact Test). These outcomes suggest that the Generator encodes reduplication in its latent codes and again confirm that manipulating latent variables well beyond training range reveals the underlying learning representations in deep convolutional networks (as proposed in Beguš 2020; Beguš 2021a).

4.1 Interpolation

That the Generator uses latent codes to encode reduplication is further suggested by another generative test performed on interpolated values of the latent code. To test how exactly the relationship between the latent codes (c_1 and c_2) works, we created sets of generated outputs based on interpolated values of the code c_1 and c_2. We manipulate c_1 and c_2 from the value 1.5 towards 0 in increments of 0.125. For example, we start with [1.5, 0] and interpolated first to [0, 0] (e.g. [1.375, 0],

[1.25, 0], etc.). From [0, 0] we further interpolate in increments of 0.125 to [0, 1.5] (e.g. [0, 0.125], [0, 0.25]). All other variables in the latent space are kept constant across all interpolated values. Each such set thus contains 25 generated samples. We generate 100 such sets (altogether 2500 outputs) and analyze each output. Out of the 100 sets, the output was either bare or reduplicated throughout the interpolated values and did not change in 55 sets. As suggested by Section 4 and Table 2, the number of bare and reduplicated forms for each level rises to near categorical values as the variables approach values of 5.

In the 45/100 sets, the output changes from the base form to a reduplicated form at some point as the codes are interpolated. If the network only learned to randomly associate base and reduplicated forms with each endpoint of the latent code, we would expect base forms to be unrelated to reduplicated forms. For example, a base form [ˈkʰulu] could turn into reduplicated [dəˈdɑlə]. An acoustic analysis of the generated sets, however, suggests that the latent code directly corresponds to reduplication. In approximately 25 out of 45 sets (55.6%) of generated outputs that undergo the change from base to a reduplicated form (or 25% of the total sets), the base form is identical to the reduplicated form with the only major difference between the two being the presence of reduplication (waveforms and spectrograms of the 25 outputs are in Figure 6). This proportion would likely be even higher with a higher interpolation resolution (higher than 0.125) and because we do not count cases in which major changes of sounds occur besides the addition of the reduplication syllable (for example, if [ˈnɑɹi] changes to [nʊˈnuɹi], we count the output as unsuccessful). In the remaining 20 outputs, several outputs undergo changes, where several segments or their features are kept constant, but the degree to which they differ can vary (e.g. [ˈpʰilə] ∼ [pəˈpʰiɹi], [ˈtʰiju] ∼ [dəˈdɑji], [ˈnɑɹə] ∼ [dəˈdɑɹi], or [ˈpʰiɹə] ∼ [təˈtʰɑli]).

Under the null hypothesis, if the Generator learns to pair the base and reduplicated forms randomly, each base form could be associated with any of the unique 243 reduplicated forms at the probability of 1/243 (0.004). Even if we assume very conservatively that each base form could be associated with only each subgroup of reduplicated consonant (C_1; e.g. voiceless stops, voiced stops, [m], [n], [v]) disregarding the vowel and
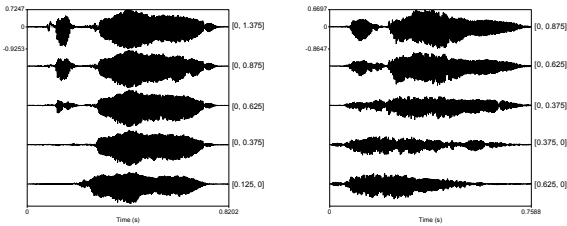
Figure 2: Waveforms showing how interpolation of latent codes c_1 and c_2 has a direct effect on presence of reduplicattion: as the values are interpolated from [1.5, 0] to [0, 1.5], the reduplication gradually appears/disappears from the output. Waveforms on the left represent reduplication of [ˈpʰiɹu] to [pəˈpʰiɹu]; waveforms on the right represent reduplication of [ˈdɑji] to [dəˈdɑji].

disregarding changes in the base, the probability of both forms being identical would still be at only 0.2 (for each of the five subgroups). In both cases, the ratio of identical base-reduplication pairs, while not categorical, is highly significant (CI = [0.4, 0.7], $p < 0.0001$ for both cases according to Exact Binomial Test).

Figure 2 illustrates how, keeping the latent space constant except for the manipulation of the latent code with which the Generator represents reduplication, the generated outputs gradually transition from the base forms [ˈpʰiɹu] and [ˈdɑji] to the reduplicated forms [pəˈpʰiɹu] and [dəˈdɑji].[7] Other major properties of the output are unchanged.

This interpolative generative test again suggests that the network learns reduplication and encodes the process in the latent codes. By interpolating the codes we can actively force reduplication in the output with no other substantial changes in the majority of cases.

4.2 Reduplication of unobserved data

To test whether the ciwGAN network learns to generalize the reduplicative pattern on unobserved data, we use latent space manipulation to force reduplication at the same time as presence of [s] in the output. Items with a [s] as the initial consonants (e.g. [ˈsiju]) appear only in bare forms in the training data. In Sections 4 and 4.1, we established that the network uses the latent code (c_1

---

[7]The exact vowel quality estimation in the generated outputs is challenging, especially in short vocalic elements of reduced vowels in the reduplicative syllables. For this reason, we default transcriptions to a [ə].
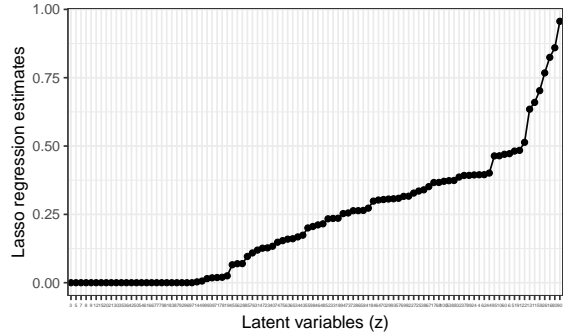

Figure 3: Absolute Lasso regression estimates (sorted from highest on the right-hand side) for a ciwGAN model identifying presence of [s] after 1000 transcribed outputs, 500 for each latent code (with the same latent variable structure of the remaining 98 variables across the two conditions). Variable z_{90} is identified as the variable corresponding to presence of [s] (the variable with the highest regression estimates).

and c_2) to represent reduplication. Following Beguš (2020) and Beguš (2021a), we can force any phonetic property in the output by manipulating the latent variables well beyond the training range. Reduplication is forced by setting the latent code to values higher than [0, 1]. We can simultaneously force [s] in the output to test the network's performance on reduplication in unseen data.

To identify latent variables with which the Generator encodes the sound [s] in the output, we generate 1000 samples with randomly sampled latent variables, but with the latent code variables (c_1 and c_2) set at [0, 1] and [1, 0] (500 samples each with the same latent variable structure of the remaining 98 variables across the two conditions). We annotate outputs for presence of [s] for the two sets and fit the data to a Lasso logistic regression model in the *glmnet* package (Simon et al., 2011). Presence of [s] is the dependent variable coded as a success; the independent variables are the 98 latent variables uniformly distributed on the interval $(-1, 1)$ (for the technique, see Beguš 2020). Lambda is computed with 10-fold cross validation. Estimates of the Lasso regression model (Figure 3) suggest that z_{90} with the highest regression estimates is one of the variables with which the Generator encodes presence of [s] in the output. For a generative test providing evidence that Lasso regression estimates correlate with network's internal representations, see Beguš (2020).

We can thus set z_{90} to marginal levels well beyond the training range and the latent code (c_1, c_2) to levels well beyond [0, 1] in order to force

reduplication and [s] in the output simultaneously. For example, when the latent code is set to [0, 3] (which forces reduplication in the output) and z_{90} to 4 (forcing [s] in the output), the network outputs a reduplicated [səˈsiji] (among other outputs) even though items containing an [s] are never reduplicated in the training data. When the code is set to even higher number, [0, 7.25], and z_{90} to 7, the network outputs [səˈsiru] in a different output. The spectrograms in Figure 4 show a clear period of frication noise characteristic of a sibilant fricative [s], interrupted by a reduplicative vowel and followed by a repeated period of frication noise characteristic of [s].

In fact, at the values [0, 7.25], and $z_{90} = 7$, the network generates approximately 33 (out of 100 tested or 33%) outputs that can be reliably analyzed as reduplicated forms with initial sV- reduplication unseen in the training data. The other 67 outputs are reduplicated forms containing other C_1s or unreduplicated [s]-forms. No outputs were observed in which C_1 of the reduplication syllable and C_1 of the base would be substantially different. While all the cases when z_{90} is manipulated involve a front vowel [i] in the base item, we can also elicit reduplication for other vowels. For example, we identify variable z_4 as corresponding to an [s] and a low vowel [ɑ] in the output (with the same technique as described for z_{90} above but with presence of [sɑ] as the dependent variable in the Lasso regression model). By manipulating z_4 to 9.5 (forcing [sɑ] in the output) and setting the latent codes to [0, 7.5], we get [səˈsɑɹu] in the output (Figure 4).

For comparison, the same L1 speaker of English who read the words in the training data read the reduplicated [səˈsiji] and [səˈsɑɹu] which were not included in the training data. Figure 4 parallels the generated reduplicated forms based on unobserved data (which were elicited by forcing [s] and reduplication in the output) and the recording of the same reduplicated form read by a human speaker. The spectrograms show clear acoustic parallels between the Generated outputs and the recording read by a human speaker (who read the words prior to computational experiments and did not hear or analyze the generated outputs).

5 Replication: Bare WaveGAN (Donahue et al., 2019)

To test whether the learning of reduplicative patterns in GANs is a robust or idiosyncratic property of the model presented in Section 4, we conduct a replication experiment. We introduce two crucial differences in the replication experiment: we train the Generator without the requirement to produce informative data and without binary latent codes. We use the model in Donahue et al. (2019) which features a "bare" GAN architecture for audio data: only the Generator and Discriminator networks without the Q-network. This architecture has the potential to inform us how GANs represent reduplicative patterns without an explicit requirement to learn informative data, i.e. without an explicit requirement to encode some salient feature of the training data in the latent space. The data used for training is the same as in the experiment in Section 3. We train the network for 15,930 steps or approximately 5,118 epochs, which is almost identical to the number of steps/epochs in the ciwGAN experiment (Section 4).

5.1 Identifying variables

Testing the learning of reduplication in the bare GAN architecture requires that we force reduplication and presence of [s] in the output simultaneously. To identify which latent variables correspond to the two properties, we use the same technique as described in Section 4. We generate and annotate 500 outputs of the Generator network with randomly sampled latent variables. We annotate the presence of [s] and the presence of reduplication. The annotations are fit to a Lasso logistic regression (as in Section 4.2): presence of reduplication or [s] are the dependent variables and each of the 100 latent z-variables are the independent predictors. Lambda values were computed with 10-fold cross validation. Regression estimates are given in Figure 5.

The plots illustrate a steep drop in regression estimates between the few latent variables with the highest estimates and the rest of the latent space. In fact, in both models, one or two variables per model emerge with substantially higher regression estimates: z_{91} and z_5 when the dependent variable is PRESENCE OF REDUPLICATION and z_{17} when the dependent variable is PRESENCE OF [s] in the output. We can assume the Generator network uses these two variables to encode presence
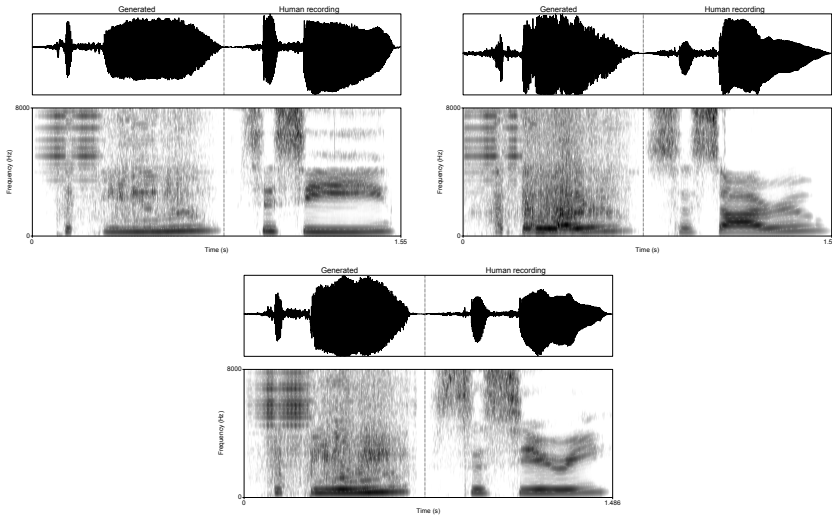
Figure 4: Waveforms and spectrograms (0–8000 Hz) of reduplicated forms containing an [s] which were absent from the training data. The generated forms on the left are paired with recordings of a female speaker reading reduplicated forms that were absent from the training data. (left) When the latent code is set to [0, 3] and z_{90} to 4, the network outputs a reduplicated [sə'siji]. (right) When the latent code is set to [0, 7.5] and z_4 to 9.5, we get [sə'sɑɹu]. (bottom) In the bare GAN architecture, when z_5 (forcing reduplication) is set to -9.25 and z_{17} (forcing [s] in the output) to -9.0, the Generator outputs a reduplicated [sə'siɹi].

of reduplication and [s], respectively.

It has been argued in Beguš (2020) that GANs learn to encode phonetic and phonological representations with a subset of latent variables. The discretized representation of continuous phonetic properties in the latent space appears even more radical in the present case. For example, in Beguš (2020), presence of [s] as a sound in the output is represented by at least seven latent variables, each of which likely controls different spectral properties of the frication noise. In the present experiment, the Generator appears to learn to encode presence of [s] with a single latent variable, as is suggested by a steep drop of regression estimates after the first variables with the highest estimates. For a generative test showing that regression estimates correlate to actual rates of a given property in generated data, see Beguš (2020). Such near-categorical cutoff is likely a consequence of the training data in the present case being considerably less variable compared to TIMIT (used for training in Beguš 2020). The network also represents an identity-based process, reduplication, with only two latent variables and features a substantial drop in regression estimates after these two variables. This discretized representation thus emerges even without the requirement of the Generator to output informative data.

In the replication experiment too, the Generator network outputs reduplicated forms for unobserved data when both reduplication and [s] are forced in the output via latent space manipulation, but significantly less so than in the ciwGAN architecture. When z_{91} (forcing reduplication) and z_{17} (forcing [s] in the output) are set to value -8.5, a higher level compared to the generated samples in the ciwGAN architecture (7 and 7.25), the network outputs only one reduplicated form with [s]-reduplication out of 100 generated outputs. By comparison, the proportion of the [s]-reduplication in the ciwGAN architecture is 33/100 – a significantly higher ratio (OR $= 48.1, p < 0.0001$; Fisher Exact Test). When z_5 (forcing reduplication) is set to -9.25 and z_{17} (forcing [s] in the output) to -9.0, the proportion of reduplicated [s]-items is slightly higher (4/100), but still significantly lower than in the ciwGAN architecture (OR $= 11.7, p < 0.0001$; Fisher Exact Test). Despite these lower proportions of reduplicated [s] in the output, the bare GAN network nevertheless extends reduplication on novel unobserved data. Figure 4 illustrates an example of a reduplicated [s]-item from the Generator network trained in the bare GAN architecture: [sə'siɹi]. The spectrogram reveals a clear period of frication noise characteristic of an [s], followed by a reduplicative vowel period, followed by another period of frication.
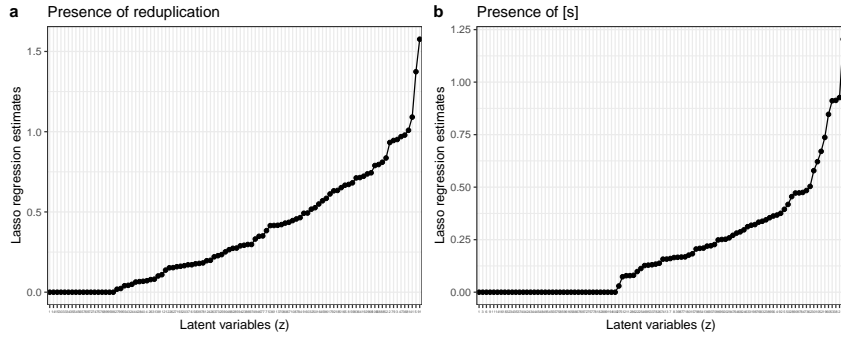
Figure 5: Absolute Lasso regression estimates (sorted from highest on the right-hand side) for two models identifying (a) presence of reduplication and (b) presence of [s] in the generated outputs of the bare GAN model (Section 5).

6 Discussion

We perform four generative tests to model learning of reduplication in deep convolutional networks: (i) a test of proportion of outputs when latent codes are manipulated to marginal values, (ii) a test of interpolating latent variables, (iii) a test of reduplication on unobserved data in the ciwGAN architecture, and (iv) a replication test of reduplication on unobserved data in the bare WaveGAN architecture. All four tests suggest that deep convolutional networks learn a simple identity-based pattern in speech called reduplication, i.e. a process that copies some phonological material to express new meaning. The ciwGAN network learns to encode a meaningful representation — presence of reduplication into its latent codes. There is a near one-to-one correspondence between the two latent codes c_1 and c_2 and reduplication. By interpolating latent codes, we cause the bare form to gradually turn into a reduplicated form with no other major changes in the output in the majority of cases. These results are close to what would be considered appearance of symbolic computation or algebraic rules.

Additional evidence that an approximation of symbolic computation emerges comes from the bare GAN experiment: there is a substantial drop in regression estimates after the first one or two latent variables with highest regression estimates, suggesting that even without the requirement to produce informative data, the network discretizes the continuous and highly variable phonetic feature — presence of reduplication — and uses a small subset of the latent space to represent this morphophonological property.

Finally, we can force the Generator to output reduplication at nearly categorical levels. When latent codes are set to marginal levels outside of training range (e.g. to [5, 0] or [0, 5]), the outputs are almost categorically unreduplicated or reduplicated (at 98 % for [5, 0]). Beguš (2021a) shows that even higher values (e.g. 15) result in performance at 100% for a subset of variables. Not all aspects of the models in this paper are categorical (e.g. interpolation of latent codes does not always change an unreduplicated to a reduplicated form without other major changes). Improving performance on this particular task is left for future work. Inability to derive categorical processes has long been an argument against the connectionist approaches to language modeling. The results of this experiments add to the work suggesting that manipulating variables to extreme marginal values results in near categorical or categorical outputs (depending on the value) of a desired property (Beguš, 2020; Beguš, 2021a).

In sum, three properties of rule-like symbolic representations emerge in deep convolutional network tested here: discretized representations, the ability to generate desired property by manipulating a small number of variables, and near categoricity for a subset of representations. These symbolic-like outcomes are facilitated by two inductive biases: the binary nature of latent codes and the requirement on the Generator to output informative data (forced by the Q-network). At least a subset of these properties also emerges in the bare WaveGAN architecture that lacks these biases, but at a reduced performance.

Encoding an identity-based pattern as a meaningful representation in the latent space emerges in a completely unsupervised manner in the ciwGAN architecture — only from the requirement that the Generator output informative data. Redu-

plicated and unreduplicated forms are never paired in the training data. The network is fed bare and reduplicated forms randomly. This unsupervised training approximates conditions in language acquisition (for hearing learners): the human language learner needs to represent reduplication and to pair bare and reduplicated forms from raw unlabeled acoustic data. The ciwGAN learns to group reduplicated and unreduplicated forms and assign a unique representation to the process of reduplication. In fact, the one-hot vector (c_1 and c_2) that the Generator learns to associate with reduplication in training can be modeled as a representation of the unique meaning/function that reduplication adds, in line with an approach to represent unique semantics with one-hot vectors (e.g. in Steinert-Threlkeld and Szymanik 2020).

The paper also argues that deep convolutional networks can learn a simple identity-based pattern (copying) from raw continuous data and extend the pattern to novel unobserved data. While the network was not trained on reduplicated items that start with an [s], we were able to elicit reduplication in the output following a technique proposed in Beguš (2020). First, we identify variables that correspond to some phonetic/phonological representation such as presence of [s]. We argue that setting single variables well above training range can reveal the underlying value for each latent variable and force the desired property in the output. We can thus force both [s] and reduplication in the output simultaneously. For example, the network outputs [səsiju] if we force both reduplication and [s] in the output; however, it never sees [səsiju] in the training data — only [siju] and other reduplicated forms, none of which included an [s].

Thus, these experiments again confirm that the network uses individual latent variables to represent linguistically meaningful representations (Beguš, 2020; Beguš, 2021a). Setting these individual variables to values well above the training interval reveals their underlying values. By manipulating these individual variables, we can explore how the representations are learned as well as how interactions between different variables work (for example, between the representation of reduplication and presence of [s]). The results of this study suggest that the deep convolutional network is not only capable of encoding different phonetic properties in individual latent variables, but also processes as abstract as copying or reduplication.

One of the advantages of probing learning in deep convolutional neural networks on speech data trained with GANs is that the innovative outputs violate training data in structured and highly informative ways. The innovative outputs with reduplication of [s]-initial forms such as [səsiju] can be directly paralleled to acoustic outputs read by L1 speaker of American English that were absent from the training data. Acoustic analysis shows a high degree of similarity between the generated reduplicated forms and human recordings, meaning that the network learns to output novel data that are linguistically interpretable and resemble human speech processes even though they are absent from the training data. Thus, the results of the experiments have implications for cognitive models of speech acquisition. It appears that one of the processes that has long been held as a hallmark of symbolic computation in language, reduplication, can emerge in deep convolutional network without language-specific components in the model even when they are trained on raw acoustic inputs.

The present paper tests a simple partial reduplicative pattern where only CV is copied and appears before the base item. This is perhaps computationally the simplest reduplicative pattern. The training data are also highly controlled and recorded by a single speaker. We can use the well-understood identity-based patterns in speech with various degrees of complexity (longer reduplication, embedding into non-reduplicative patterns) to further test how inductive biases and hyperparameter/architecture choices interact with learning in deep convolutional networks. Finally, learning biases in the ciwGAN model can be (superficially) compared to learning biases in human subjects in future work. This paper suggests that the Generator provides informative outputs even if trained on comparatively small data sets (for a similar conclusion for other processes, see Beguš 2021b). This means we can use the same training data to probe learning in CNNs and in human artificial grammar learning experiments (for a methodology, see Beguš 2021b). While these comparisons are necessarily superficial at this point, they can provide insights into common learning biases between human learners and computational models.

Acknowledgements

References

Ben Adlam, Charles Weill, and Amol Kapoor. 2019. Investigating under and overfitting in Wasserstein Generative Adversarial Networks. In *ICML Understanding and Improving Generalization in Deep Learning Workshop (2019)*. arXiv 1910.14137v1.

Raquel G. Alhama and Willem H. Zuidema. 2018. Pre-wiring and pre-training: What does a neural network need to learn truly general identity rules? *Journal of Artificial Intelligence Research*, 61:927–946.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein Generative Adversarial Networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223, International Convention Centre, Sydney, Australia. PMLR.

Alexei Baevski, Steffen Schneider, and Michael Auli. 2020. vq-wav2vec: Self-supervised learning of discrete speech representations. In *International Conference on Learning Representations*, pages 1–12.

Gašper Beguš. 2021a. Ciwgan and fiwgan: Encoding information in acoustic data to model lexical learning with Generative Adversarial Networks. *Neural Networks*, 139:305–325.

Gašper Beguš. 2021b. Local and non-local dependency learning and emergence of rule-like representations in speech data by Deep Convolutional Generative Adversarial Networks. *Computer Speech & Language*, page 101244.

Gašper Beguš. 2020. Generative adversarial phonology: Modeling unsupervised phonetic and phonological learning with neural networks. *Frontiers in Artificial Intelligence*, 3:44.

Iris Berent. 2013. The phonological mind. *Trends in Cognitive Sciences*, 17(7):319 – 327.

Paul Boersma and David Weenink. 2015. Praat: Doing phonetics by computer [computer program]. version 5.4.06. Retrieved 21 February 2015 from http://www.praat.org/.

Simone Brugiapaglia, Matthew Liu, and Paul Tupper. 2020. Generalizing outside the training set: When can neural networks learn identity effects?

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. InfoGAN: Interpretable representation learning by information maximizing Generative Adversarial Nets. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2172–2180. Curran Associates, Inc.

Jan Chorowski, Ron J. Weiss, Samy Bengio, and Aäron van den Oord. 2019. Unsupervised speech representation learning using WaveNet autoencoders. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(12):2041–2053.

Yu-An Chung, Chao-Chung Wu, Chia-Hao Shen, Hung-Yi Lee, and Lin-Shan Lee. 2016. Audio word2vec: Unsupervised learning of audio segment representations using sequence-to-sequence autoencoder. In *Interspeech 2016*, pages 765–769.

Hossep Dolatian and Jeffrey Heinz. 2018. Modeling reduplication with 2-way finite-state transducers. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 66–77, Brussels, Belgium. Association for Computational Linguistics.

Hossep Dolatian and Jeffrey Heinz. 2020. Computing and classifying reduplication with 2-way finite-state transducers. *Journal of Language Modelling*, 8(1):179–250.

Chris Donahue, Julian J. McAuley, and Miller S. Puckette. 2019. Adversarial audio synthesis. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Ewan Dunbar, Robin Algayres, Julien Karadayi, Mathieu Bernard, Juan Benjumea, Xuan-Nga Cao, Lucie Miskic, Charlotte Dugrain, Lucas Ondel, Alan W. Black, Laurent Besacier, Sakriani Sakti, and Emmanuel Dupoux. 2019. The zero resource speech challenge 2019: TTS without T. In *Proc. Interspeech 2019*, pages 1088–1092.

Ewan Dunbar, Xuan Nga Cao, Juan Benjumea, Julien Karadayi, Mathieu Bernard, Laurent Besacier, Xavier Anguera, and Emmanuel Dupoux. 2017. The zero resource speech challenge 2017. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 323–330.

Ewan Dunbar, Julien Karadayi, Mathieu Bernard, Xuan-Nga Cao, Robin Algayres, Lucas Ondel, Laurent Besacier, Sakriani Sakti, and Emmanuel Dupoux. 2020. The zero resource speech challenge 2020: Discovering discrete subword and word units. *Interspeech 2020*.

Michael G. Dyer. 1991. Connectionism versus symbolism in high-level cognition. In Terence Horgan and John Tienson, editors, *Connectionism and the Philosophy of Mind*, pages 382–416. Springer Netherlands, Dordrecht.

Ryan Eloff, André Nortje, Benjamin van Niekerk, Avashna Govender, Leanne Nortje, Arnu Pretorius, Elan Biljon, Ewald van der Westhuizen, Lisa Staden, and Herman Kamper. 2019. Unsupervised acoustic unit discovery for speech synthesis using discrete latent-variable neural networks. In *Proc. Interspeech 2019*, pages 1103–1107.

Ansgar D. Endress, Ghislaine Dehaene-Lambertz, and Jacques Mehler. 2007. Perceptual constraints and the learnability of simple grammars. *Cognition*, 105(3):577 – 614.

Jerry A. Fodor and Zenon W. Pylyshyn. 1988. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28(1):3–71.

Michael Gasser. 1993. Learning words in time: Towards a modular connectionist account of the acquisition of receptive morphology. Technical report, Indiana University, Bloomington.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc.

Bruce Hayes. 2009. *Introductory Phonology*. Wiley-Blackwell, Malden, MA.

Sharon Inkelas and Cheryl Zoll. 2005. *Reduplication: Doubling in Morphology*. Cambridge Studies in Linguistics. Cambridge University Press.

Herman Kamper, Aren Jansen, Simon King, and Sharon Goldwater. 2014. Unsupervised lexical clustering of speech segments using fixed-dimensional acoustic embeddings. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 100–105.

Chia-ying Lee, Timothy J. O'Donnell, and James Glass. 2015. Unsupervised lexicon discovery from acoustic input. *Transactions of the Association for Computational Linguistics*, 3:389–403.

Christopher D. Manning. 2003. Probabilistic syntax. In Rens Bod, Jennifer Hay, and Stefanie Jannedy, editors, *Probabilistic Linguistics*, pages 289–342. MIT Pres, Cambridge, MA.

Gary F. Marcus. 2001. *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. MIT press.

Gary F. Marcus, Sujith Vijayan, Shoba Bandi Rao, and Peter M. Vishton. 1999. Rule learning by seven-month-old infants. *Science*, 283(5398):77–80.

Yoshihiro Maruyama. 2021. Symbolic and statistical theories of cognition: Towards integrated artificial intelligence. In *Software Engineering and Formal Methods. SEFM 2020 Collocated Workshops*, pages 129–146, Cham. Springer International Publishing.

James L. McClelland and David C. Plaut. 1999. Does generalization in infant learning implicate abstract algebra-like rules? *Trends in Cognitive Sciences*, 3(5):166 – 168.

James L. McClelland, David E. Rumelhart, and PDP Research Group. 1986. *Parallel distributed processing: Explorations in the microstructure of cognition*, volume 2. MIT Press, Cambridge, MA.

Marvin L. Minsky. 1991. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI Magazine*, 12(2):34.

Max Nelson, Hossep Dolatian, Jonathan Rawski, and Brandon Prickett. 2020. Probing RNN encoder-decoder generalization of subregular functions using reduplication. *Proceedings of the Society for Computation in Linguistics*, 3(1):31–42.

Benjamin van Niekerk, Leanne Nortje, and Herman Kamper. 2020. Vector-quantized neural networks for acoustic unit discovery in the ZeroSpeech 2020 challenge. *Interspeech 2020*.

Brandon Prickett, Aaron Traylor, and Joe Pater. 2018. Seq2Seq models with dropout can learn generalizable reduplication. In *Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology*, pages 93–100, Brussels, Belgium. Association for Computational Linguistics.

Alec Radford, Luke Metz, and Soumith Chintala. 2016. Unsupervised representation learning with deep convolutional generative adversarial networks. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016*. arXiv 1511.06434v2.

Okko Räsänen, Tasha Nagamine, and Nima Mesgarani. 2016. Analyzing distributional learning of phonemic categories in unsupervised deep neural networks. *CogSci 2016*, 2016:1757–1762.

Sergey Rodionov. 2018. info-wgan-gp. GitHub repository. https://github.com/singnet/semantic-vision/tree/master/experiments/concept_learning/gans/info-wgan-gp. Last accessed on July 16, 2021.

David E. Rumelhart, James L. McClelland, and PDP Research Group. 1986. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 1. MIT Press, Cambridge, MA.

Walter J. Savitch. 1989. A formal model for context-free languages augmented with reduplication. *Computational Linguistics*, 15(4):250–261.

Cory Shain and Micha Elsner. 2019. Measuring the perceptual availability of phonological features during language acquisition using unsupervised binary stochastic autoencoders. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 69–85, Minneapolis, Minnesota. Association for Computational Linguistics.

Noah Simon, Jerome Friedman, Trevor Hastie, and Rob Tibshirani. 2011. Regularization paths for Cox's proportional hazards model via coordinate descent. *Journal of Statistical Software*, 39(5):1–13.

Shane Steinert-Threlkeld and Jakub Szymanik. 2020. Ease of learning explains semantic universals. *Cognition*, 195:104076.

Suzanne Urbanczyk. 2017. Phonological and morphological aspects of reduplication. In *Oxford Research Encyclopedia of Linguistics*. Oxford University Press.

Colin Wilson. 2018. Modeling morphological affixation with interpretable recurrent networks: Sequential rebinding controlled by hierarchical attention. In *CogSci 2018*, pages 2693–2698.

Colin Wilson. 2020. Re(current) reduplication: Interpretable neural network models of morphological copying. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2020*, volume 2, pages 379–380.
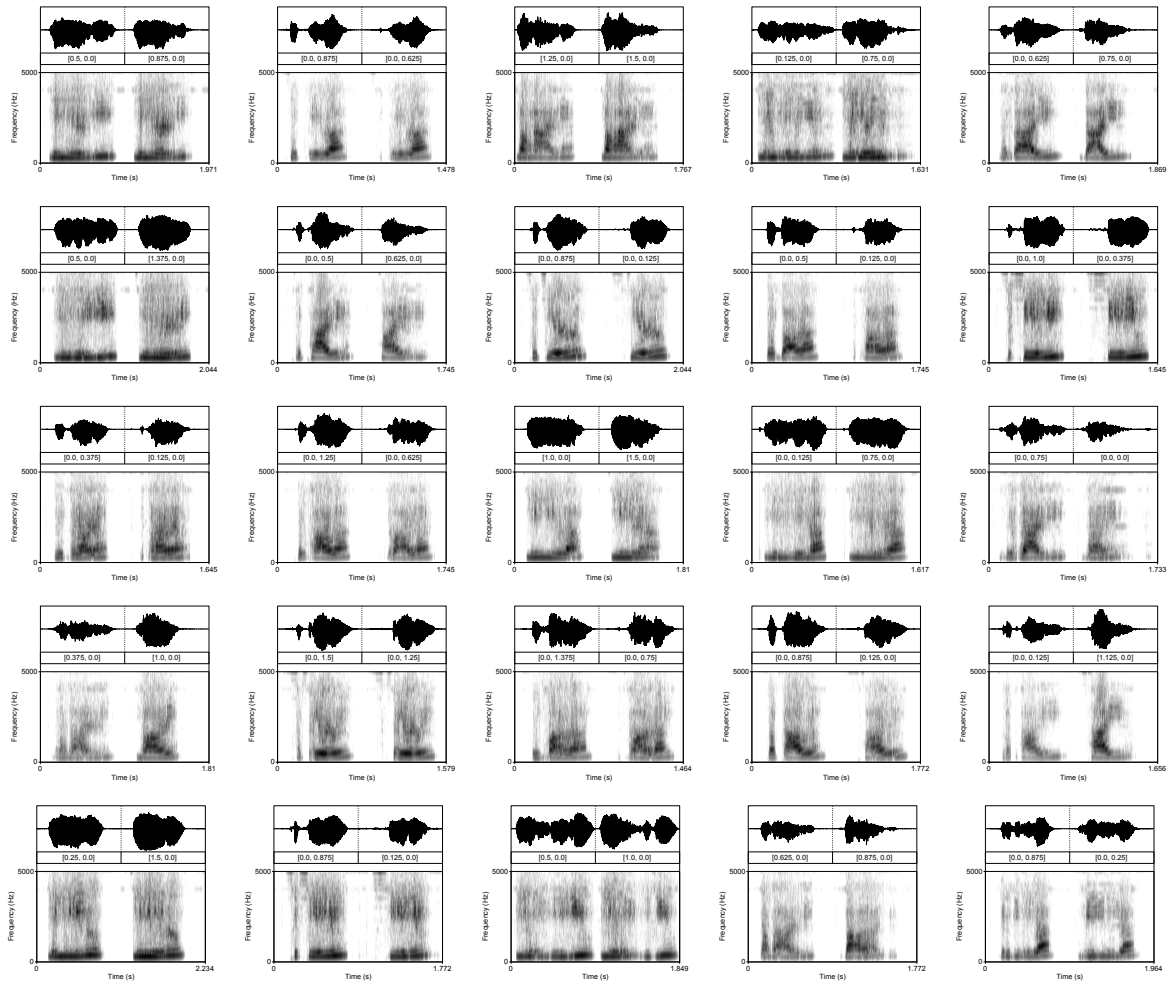
A Appendix

Figure 6: Waveforms and spectrgrograms (0-5000 Hz) of the 25 reduplicated-unreduplicated form pairs annotated as involving no other major changes than the presence of the reduplicative syllable. Reduplicated outputs are on the left; unreduplicated outputs on the right. Values of the latent code are listed below each output; all other 98 z-variables are kept constant across the two pairs. In two cases involving a nasal C_1, it is challenging to distinguish between very short remnants of the vocalic period of reduplication and periodic vibration of the nasal (the difference between reduplicated and unreduplicated form), but there is a clear contrast between the two forms of the pair.