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

Gašper Beguš[a]

[a]*Department of Linguistics, University of California, Berkeley, 1203 Dwinelle Hall #2650, Berkeley, CA 94720*

Abstract

Identity-based patterns for which a computational model needs to output some feature together with a copy of that feature are computationally challenging, but pose no problems to human learners and are common in world's languages. In this paper, we test whether a neural network can learn an identity-based pattern in speech called reduplication. To our knowledge, this is the first attempt to test identity-based patterns in deep convolutional networks trained on raw continuous data. Unlike existing proposals, we test learning in an unsupervised manner and we train the network on raw acoustic data. We use the ciwGAN architecture (Beguš, 2020a) in which learning of meaningful representations in speech emerges from a requirement that the deep convolutional network generates informative data. Based on four generative tests, we argue that a deep convolutional 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 changes to the output in the majority of cases. We also argue that the network extends the identity-based pattern to unobserved data: when reduplication is forced in the output with the proposed technique for latent space manipulation, the network generates reduplicated data (e.g., it copies an [s] e.g. in [sə-siju] for [siju] although it never sees any reduplicated forms containing an [s] in the input). Comparison with human outputs of reduplication show a high degree of similarity. Exploration of how meaningful representations of identity-based patterns emerge 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.

Keywords: artificial intelligence, generative adversarial networks, speech, identity-based learning, neural network interpretability

1. Introduction

What computational models can and cannot learn has been a topic of focus both in the field of machine learning and in computational cognitive science. One advantage of probing the learning of neural networks on language data is that dependencies in speech are well understood and unique. Particularly informative is precisely phonetic and phonological data. Phonetics, a subfield of linguistics, is primarily interested in one of the few continuous aspect of human language — the physical properties of speech sounds. Phonology is primarily concerned with the first discretization that human language users perform: from continuous phonetic data to discretized mental representation of meaning-distinguishing sounds called *phonemes*. Phonological grammar not only discretizes and groups sounds into mental representational units, but also manipulates these units

Email address: begus@berkeley.edu (Gašper Beguš)

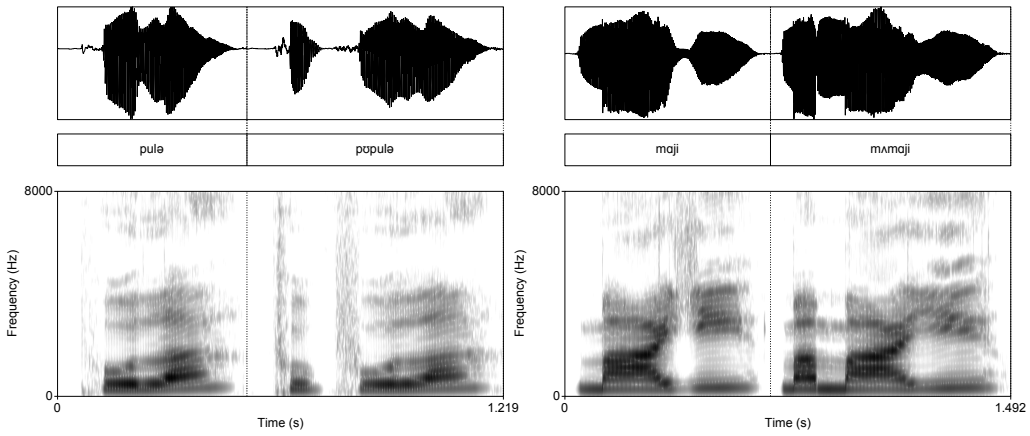

Figure 1: Waveforms and spectrograms (0-8000 Hz) of two pairs of bare and reduplicated items read by an L1 American English speaker: [ˈpʰulə] ∼ [pʊˈpʰulə] **(left)** and [ˈmaji] ∼ [mʌˈmaji] **(right)**. Note that the Generator is trained only on waveforms, not spectrograms, which are provided here for the purpose of analysis.

with a computation that can be expressed by subregular classes of computational languages (Heinz and Idsardi, 2013) and can be called phonological computation. Scientific study of speech with its long tradition (an overview in van der Hulst 2013 and MacMahon 2013) resulted in a strong understanding of how dependencies and computations in speech data work as well as how to represent these dependencies formally, such that they correspond at least superficially to human behavior (Chomsky and Halle, 1968; Wilson, 2006).

Another advantage of probing learning of deep neural networks with language data is that we can superficially compare outputs of computational models to human linguistic behavior. This in turn provides information on which aspects in human language can be explained with deep neural networks alone without having to posit language-specific computational devices. On the other hand, such comparisons can inform us about which aspects of language the deep neural networks can and cannot learn and potentially why this is the case.

Identity-based patterns, repetition, or copying are dependencies that pose challenges in machine learning and computational cognitive science. One of the most intriguing processes in phonological computation is precisely an identity-based pattern — reduplication. It 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). Reduplication 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]).

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; however, raw audio files are a more accurate representation of what hearing human language learners are actually faced with as their primary linguistic data in spoken languages. Figure 1 features two waveforms representing differences in sound pressure when a speaker of English reads an unreduplicated bare nonce word and its reduplicated counterpart (/pula/ ∼ /[pu-pula/ and /maji/ ∼ /[ma-maji/). As is clear from Figure 1, an identity-based pattern such as reduplication is considerably more complex in raw data compared to discretized representations. Additionally, reduplication is often complicated with other phonetic processes. In our training data, for example, reduplicative vowels get reduced (e.g. from [ɑ] to [ʌ]) and aspiration of [pʰ] is reduced in the reduplicative syllable because the reader

is an L1 speaker of American English. Nevertheless, the waveforms and spectrograms clearly show that the reduplicated output contains repeated continuous material from the unreduplicated base. For example, in the reduplicated [mʌˈmɑji] there is a clear period characteristic of a nasal [m], which is followed by a vocalic element and another period characteristic of the nasal [m].

Reduplication has a clear function in the linguistic system by adding various meanings/semantic values across languages: from forming plural nouns to forming adjectives from nouns. For example, reduplication is used to form adjectives from nouns in Rotuman. A noun [rosi] 'fraud' becomes an adjective [ros-rosi] 'cunning' after reduplication (Inkelas and Zoll, 2005). In Motu, on the other hand, the verb [raka] 'to walk' appears as [ra-raka] when used with plural subjects (Lister-Turner et al., 1941).

Reduplication has been central in discussions about what artificial neural networks can learn. An early contribution to this debate is Marcus et al. (1999), arguing that simple neural networks are unable to learn a simple reduplication pattern that 7-month old humans infants are able to learn (see also Gasser 1993). Marcus et al. (1999) argue that 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 results 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).

Reduplication is indeed unique among processes 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/ $\sim$ /papala/ and /tala/ $\sim$ /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). In /$\mathrm{k}_i\mathrm{a}_j\mathrm{k}_i\mathrm{a}_j$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.

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. The inputs to the models are either characters (phones or phonemes) or discrete binary featural representations of phonemes rather than raw acoustic data. 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 or, in other words, raw acoustic inputs. Hearing infant learners need to acquire reduplication from continuous speech data that is substantially more complex than already discretized characters or binary features

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

(as demonstrated in Figure 1).

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.

To better understand learning in deep convolutional networks, this paper tests learning of an identity-based pattern, reduplication, with a deep convolutional network architecture ciwGAN (proposed in Beguš 2020a) within the GAN framework (Goodfellow et al., 2014; Donahue et al., 2019). The networks are trained on a common reduplication pattern and tested on novel unobserved acoustic forms. To the author's knowledge, this is the first proposal to test learning of an identity-based pattern with deep convolutional networks. The proposed ciwGAN model not only learns to output data that resembles human speech, but also learns to encode meaningful information into raw acoustic outputs (Beguš, 2020a). In other words, association between some meaningful property of the speech data and unique representation of that property in the ciwGAN architecture emerges automatically from the requirement that the Generator output informative data (Beguš, 2020a). Another advantage of the GAN architecture is that the Generator network which learns to produce data from only positive raw acoustic inputs has no direct connections to the input data. The networks produce innovative data that violate training data distributions in structured and highly informative ways (Beguš, 2020b,a). Learning is unsupervised in the GAN architecture and the model requires no prior levels of abstraction: it is trained on raw acoustic data. For example, we train the networks on acoustic data of items such as /pala/ $\sim$ /papala/ and /tala/ $\sim$ /tatala/, but test reduplication on acoustic forms of items such as /sala/, which is never reduplicated in the training data. While equivalents of copying/identity-based patterns can be constructed in the visual domain, the author is not aware of studies that test identity-based visual patterns with deep convolutional neural networks.

How can we test learning of reduplication in a deep convolutional network that is trained only 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 instance, we can measure the learning of reduplication in testing the ability of the generator to output the reduplicated form /sasana/ when only the base /sana/ is present in the training set. Using the technique proposed in Beguš (2020b), we can identify latent variables that correspond 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 (Beguš, 2020b), we can 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. 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. By 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 (proposed for non-identity-based patterns in Beguš 2020b) allows us to directly explore the 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.

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 data that resemble speech, but also learn to encode phonetic and phonological representations in the latent space (Beguš, 2020b).

The major advantage of the GAN architecture for modeling speech is that the Generator network does not have direct access to the training 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. Beguš (2020b) argues that GANs not only learn to encode phonetic and phonological representations in their latent space, but also learn phonological processes (approximates of phonological computation). For example, the Generator learns to output unaspirated stops [p, t, k] after an [s] and aspirated stops (produced with a puff of air) [p^h, t^h, k^h] when no [s] precedes when trained on TIMIT (Garofolo et al., 1993) in accordance with a simple allophonic rule in English where voiceless stops surface as unaspirated when proceeded by an [s] (e.g. [ˈ**p^h**ɪt] 'pit' vs. [ˈs**p**ɪt]). While the network learns this distribution, it also violates training data and occasionally outputs an aspirated [p^h, t^h, k^h] even when [s] precedes. As argued in (Beguš, 2020b), GANs violate training data in a structured way that can be paralleled to violations of primary linguistic data in L1 acquisition. For example, human L1 learners occasionally output aspirated stops in the [s]-condition which are substantially more aspirated than stops in the same condition in the input data (Bond and Wilson, 1980; Beguš, 2020b).

In the first experiment, we use the ciwGAN (Categorical InfoWaveGAN) model proposed in Beguš (2020a). 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. 2015) 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 is an extension of the GAN architecture in which the Discriminator 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.

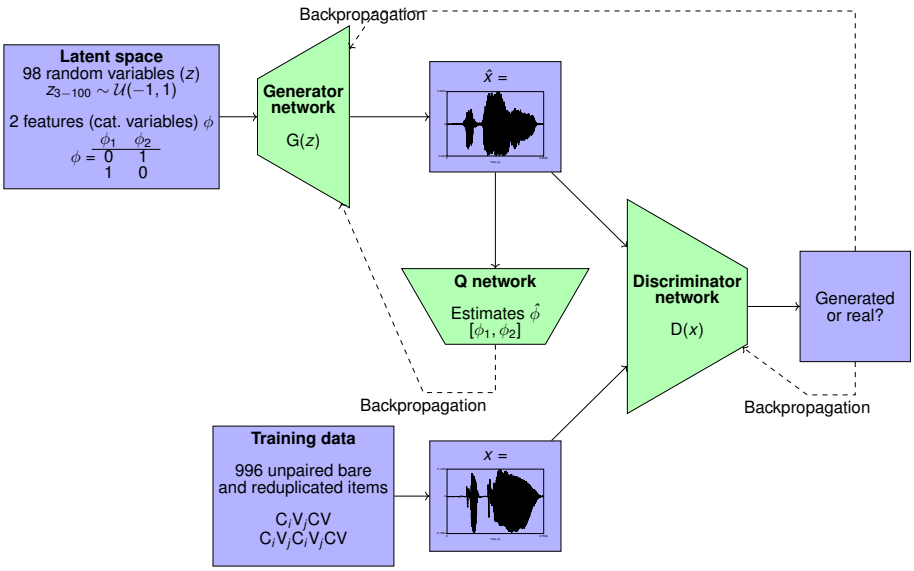

Figure 2: The ciwGAN architecture as proposed in Beguš (2020a) used in this paper with training data as described in Section 3. Green trapezoids represent deep convolutional neural networks.

In Beguš (2020a), we propose 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. Code is available here: `https://github.com/gbegus/fiwGAN-ciwGAN`. For all details about the architecture, see Beguš (2020a).

The Generator network is a deep convolutional network that takes as its input 100 latent variables (see Figure 2). Two of the 100 variables are code variables (ϕ_1 and ϕ_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. We introduce a separate Q-network (also in Rodionov 2018). 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, 2020a). In the ciwGAN architecture, the Q-network is trained on estimating the latent code variables with a softmax function (Beguš, 2020a). In other words, the Q-network takes the Generator's outputs and estimates the latent code variables ϕ_1 and ϕ_2. Weights of both the Generator network and the Q-network are updated according to the Q-network's loss function. This forces the Generator to output informative data.

The advantage of the ciwGAN architecture is that the network not only learns to output innovative data that resemble speech in the input, but also provides meaningful representations about data. These meaningful representations arise in an unsupervised manner. For example, 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, we run an independent experiment on a bare WaveGAN architecture using the same training data. The difference between the two architecture 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 $(-1, 1)$. In other words, there is no requirement that the Generator outputs informative data. For the GAN architecture in Experiment 2, see Figure 7.

Beguš (2020b,a) also proposes 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 languages: 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, 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 ([ˈpʰɑlə] ∼ [pəˈpʰɑlə]). 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 and (ii) deaspiration of voiceless stops in the unstressed reduplication syllable. The training data includes two unique repetitions of each item and two repetitions of the corresponding reduplicated forms. Table 1 illustrates the training data. All items used in training are given in Tables A.3, A.4, and A.5.

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.[2] 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 "across the board", i.e to novel feature values not only to novel segments (Berent, 2013; Prickett et al., 2018). 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.

[2]Items ['sala], ['suru], and ['suju] each miss one repetition (four altogether).

voiceless C_1	$C_1V_2C_3V_4$	ˈpʰɑli	ˈtʰɑli	ˈkʰɑli
	$C_1V_2C_1V_2C_3V_4$	pʌˈpʰɑli	tʌˈtʰɑli	kʌˈkʰɑli
voiced C_1	$C_1V_2C_3V_4$	ˈbɑli	ˈdɑli	ˈgɑli
	$C_1V_2C_1V_2C_3V_4$	bʌˈbɑli	dʌˈdɑli	gʌˈgɑli
C_1 = [m, n, v]	$C_1V_2C_3V_4$	ˈmɑli	ˈnɑli	ˈvɑli
	$C_1V_2C_1V_2C_3V_4$	mʌˈmɑli	nʌˈnɑli	vʌˈvɑli
C_1 = [s]	$C_1V_2C_3V_4$	ˈsɑli	ˈsɑli	ˈsɑli
	$C_1V_2C_1V_2C_3V_4$	—	—	—

Table 1: A schematic illustration of the training data in the International Phonetic Alphabet. For the entire training data set, see Tables A.3, A.4, and A.5.

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/.

The training data was recorded in a sound attenuated booth at the University of Washington. A female speaker of English who was compensated for her time read words from sheets of paper and was recorded with a MixPre 6 (SoundDevices) preamp/recorder and the AKG C544L head-mounted microphone positioned approximately 2 cm to the side of the mouth, initially with 16-bit quantization and 44.1 kHz sampling, but then downsampled to 16 kHz. Altogether 996 unique sliced items were used in training data.

4. CiwGAN (Beguš, 2020a)

Our model features two latent code variables, ϕ_1 and ϕ_2 (Figure 2). In the training phase, the two variables 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 meaningful 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 ϕ_1 and ϕ_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 form. 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, 2020a). Despite these guidelines, the choice of number of steps is somewhat arbitrary (for discussion, see Beguš 2020b).

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 ϕ_1 and ϕ_2 are set to [1, 0], [0, 1], [5, 0], and [0, 5].

We generate 100 outputs for each latent code [0, 1] and [1, 0] (200 total) and annotate them for presence or absence of reduplication. 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 (odd 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 2020b,a).

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 (ϕ_1 and ϕ_2) works, we created sets of generated outputs based on interpolated values of the code ϕ_1 and ϕ_2. We manipulate ϕ_1 and ϕ_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 furher 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 53 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 25 out of 45 sets (55.6%) of generated outputs that undergo the change from base to a reduplicated form, the base form is identical to the reduplicated form with the only major difference between the two being the presence of reduplication. 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 even one sound changes beside the reduplication syllable (for example, if [ˈnɑɹi] changes to [nʊˈnuɹi], we count the output as unsuccessful). 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 he probability of 1/243 (0.004). Even if we assume very conservatively that each base form could be associated with only each subgroup of

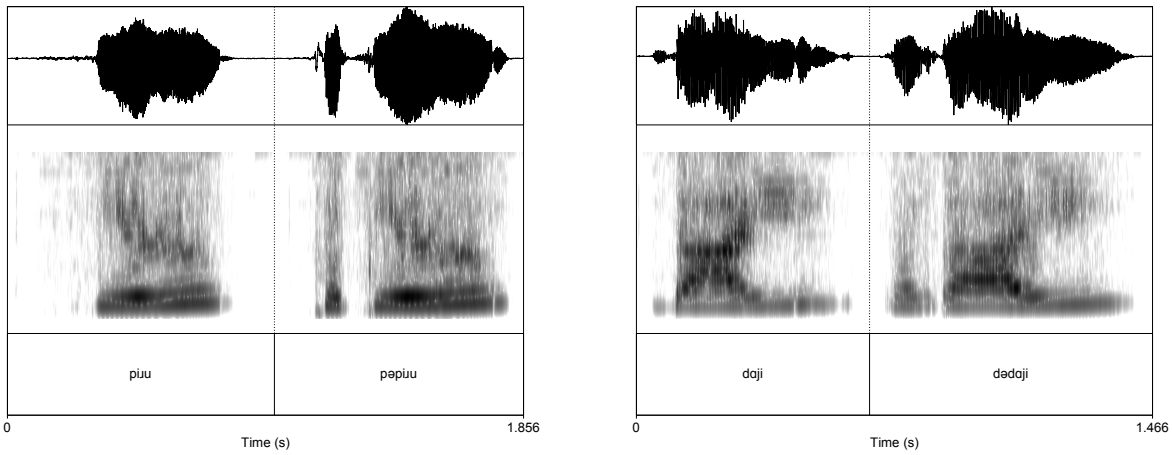

Figure 3: Waveform and spectrograms (0-4000 Hz) of bare and reduplicated forms by the Generator network in the CiwGAN architecture. **(left)** When latent codes are set at [0, 0.125], the network outputs unreduplicated [ˈpʰiɹu]. When the latent code is set at [0, 1.5], the network outputs a reduplicated [pəˈpʰiɹu]. The waveform and spectrogram illustrate that there are no other major changes between the bare and reduplicated form. **(right)** Similarly, the Generator in the ciwGAN architecture outputs an unreduplicated [ˈdɑji] when the latent code is set to [0.875, 0] and reduplicated [dəˈdɑji] when the code is set to [0, 0.75]. Note that the Generator only outputs waveforms, not spectrograms, which are provided here for the purpose of analysis.

reduplicated consonant (C_1) disregarding the vowel (e.g. voiceless stops, voiced stops, [m], [n], [v]) and 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 ($\text{CI} = [0.4, 0.7], p < 0.0001$ for both cases according to Exact Binomial Test).

Figures 3 and 4 illustrate 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].[3] All other properties of the output are unchanged, as is clear from the spectrograms in Figure 3. 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 (ϕ_1 and ϕ_2) to represent reduplication. Following Beguš (2020b,a), we can force any phonetic property in the output by manipulating the latent variables well beyond the training range results. 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.

[3]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].

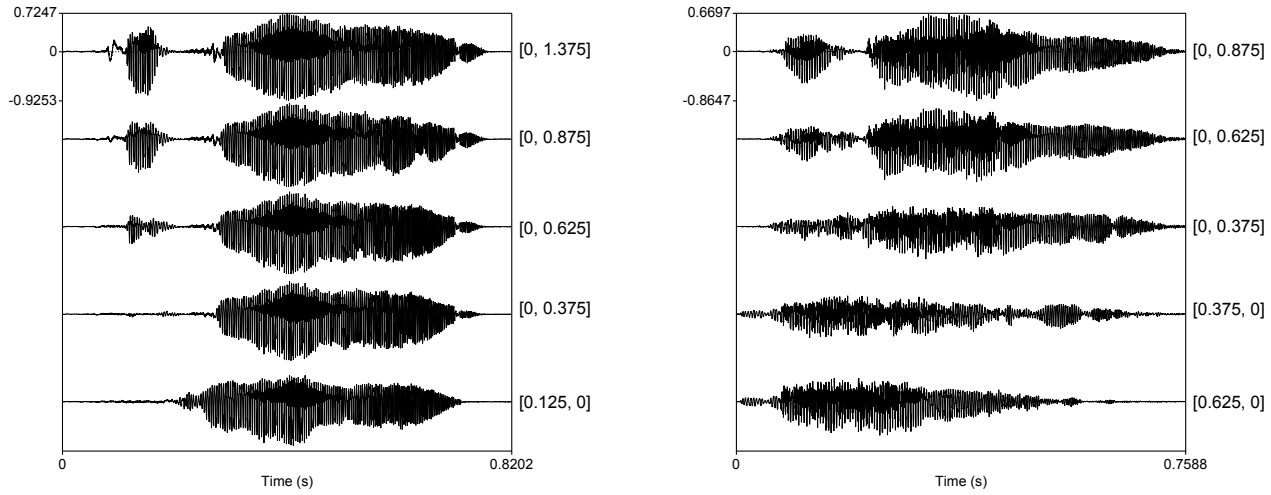

Figure 4: Waveforms showing how interpolation of latent codes ϕ_1 and ϕ_2 has a direct effect on presence of reduplicattion: as the values are interpolated from [1.5, 0] to [0, 1.5], the reduplication gradually appears/disappears from the output. Waveforms on the left represent reduplication of [ˈpʰiɹu] to [pəˈpʰiɹu]; waveforms on the right represent reduplication of [ˈdɑji] to [dəˈdɑji] (for spectrograms of the endpoints, see Figure 3).

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 (ϕ_1 and ϕ_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 variable uniformly distributed on the interval $(-1, 1)$ (for the technique, see Beguš 2020b). Lambda is computed with 10-fold cross validation. Estimates of the Lasso regression model (Figure 5) suggest that z_{90} with the highest regression estimates is the variable 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 (2020b).

We can thus set z_{90} to marginal levels well beyond the training range and the latent code (ϕ_1, ϕ_2) to levels well beyond [0, 1]. 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] 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 6 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 40 (out of 100 tested or 40%) 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

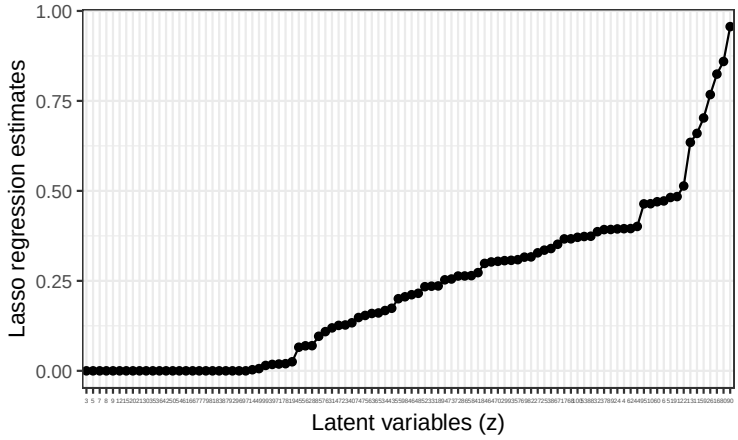

Figure 5: Absolute Lasso regression estimates (sorted from highest on the right-hand side) for a ciwGAN model identifying presence of [s] after 1000 transcribed outputs, 500 for each latent code (with the same latent variable structure of the remaining 98 variables across the two conditions). Variable z_{90} is identified as the variable corresponding to presence of [s] (the variable with the highest regression estimates).

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. A waveform and spectrogram of this output are given in Figure 6.

For comparison, the same L1 speaker of English who read the words in the training data read the reduplicated [səˈsiji], [səˈsiru], and [səˈsɑɹu] which were not included in the training data. Figure 6 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 almost identical acoustic properties in 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. Reproduction: 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 one crucial difference in the replication experiment: we train the Generator without the requirement to produce informative data. 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 architecture is summarized in Figure 7. The data used for training is the same as in the first experiment (described 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

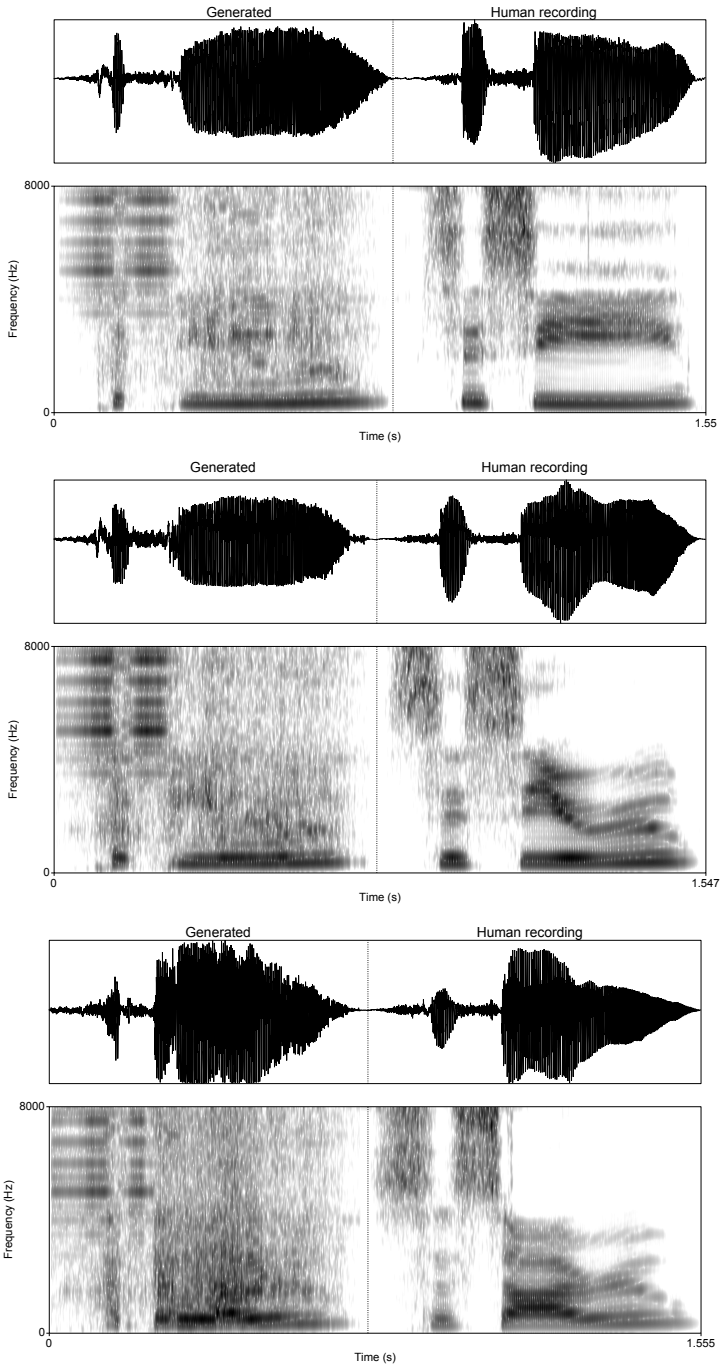

Figure 6: Waveforms and spectrograms (0–8000 Hz) of reduplicated forms containing an [s] which were absent from the training data suggesting that the ciwGAN network learn to extend reduplication o novel unobserved data. The generated forms on the left are paired with recordings of a female speaker reading reduplicated forms that were absent from the training data. The comparison shows a high degree of similarity between the generated outputs and human recordings. **(top)** When the latent code is set to [0, 3] and z_{90} to 4, the network outputs a reduplicated [sə'siji]. **(middle)** When the latent code is set to [0, 7.25], and z_{90} to 7, the network outputs [sə'siru]. **(bottom)** When the latent code is set to [0, 7.5] and z_4 to 9.5, we get [sə'sɑɹu]. Note that the Generator only outputs waveforms, not spectrograms, which are provided here for the purpose of analysis.

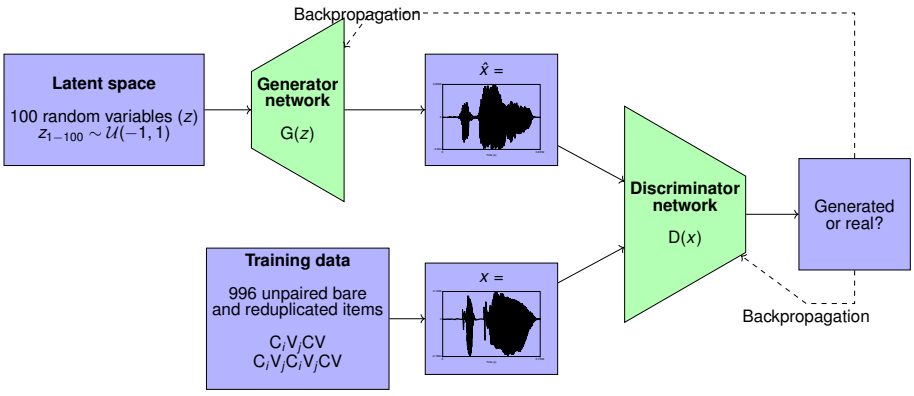

Figure 7: The bare GAN architecture as proposed in Donahue et al. (2019). Green trapezoids represent deep convolutional neural networks.

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 8.

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 a 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 of reduplication and [s], respectively.

It has been argued in Beguš (2020b) 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 (2020b), 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 (2020b). 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 2020b). 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 outputs 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 ($\text{OR} = 48.1, p < 0.0001$; Fisher

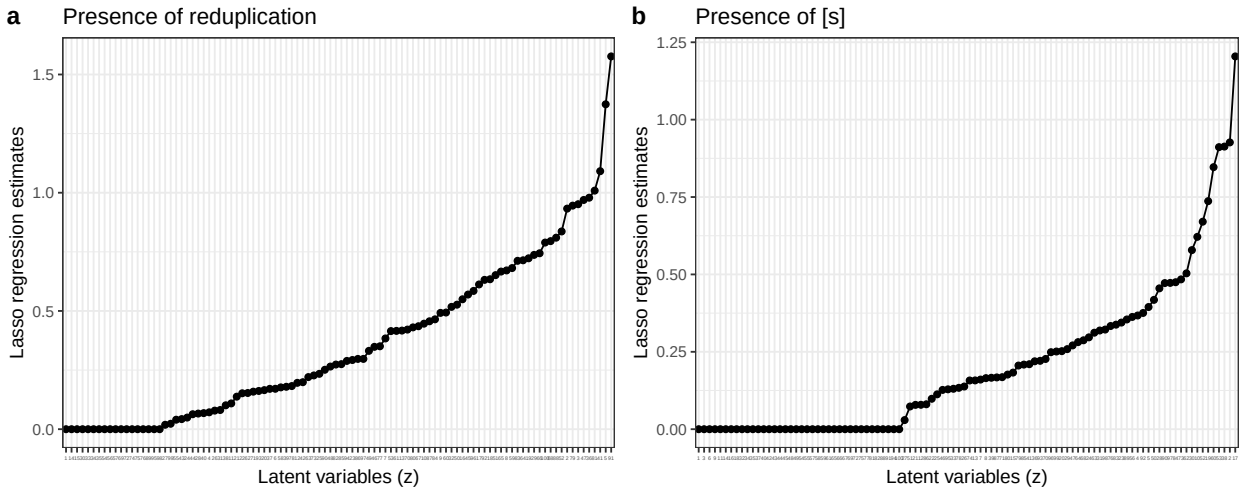

Figure 8: Absolute Lasso regression estimates (sorted from highest on the right-hand side) for two models identifying **(a)** presence of reduplication and **(b)** presence of [s] in the generated outputs of the bare GAN model (Section 5). The models are based on 500 transcribed outputs. Variables z_{91} and z_5 are identified as the variables corresponding to presence of reduplication (the variable with the highest regression estimates). Variables z_{17} is identified as the variables corresponding to presence of [s] (the variable with the highest regression estimates).

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 ($\text{OR} = 11.7, p < 0.0001$; 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 9 illustrates an example of a reduplicated [s]-item from the Generator network trained in the bare GAN architecture: [sɔˈsiːi]. 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.

6. Discussion

We perform four generative tests to asses 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 can 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 ϕ_1 and ϕ_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 replication 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

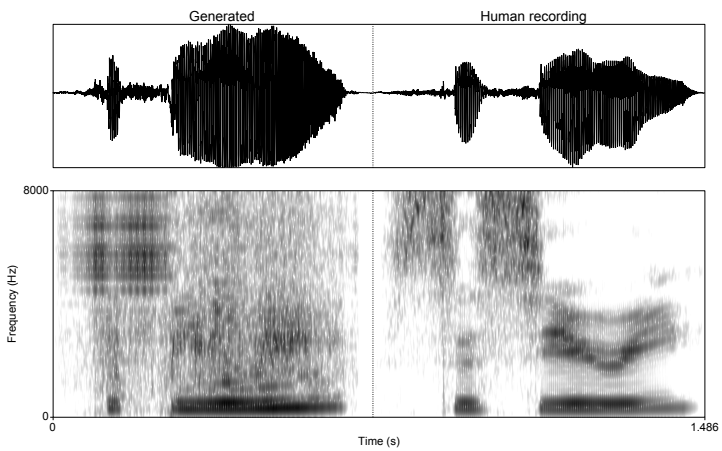

Figure 9: Waveforms and spectrograms (0–8000 Hz) of a reduplicated form containing an [s] which were absent from the training data suggesting that even the bare GAN network learns to extend reduplication o novel unobserved data. The generated form on the left is paired with a recordings of a female speaker reading the same reduplicated form that were absent from the training data. The comparison shows a high degree of similarity between the generated outputs and human recordings. 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 [səˈsiːi] which is absent from the training data. Note that the Generator only outputs waveforms, not spectrograms, which are provided here for the purpose of analysis.

"discretizes" the continuous and highly variable phonetic feature — presence of reduplication — and uses a small subset of the latent space to represent this phonetic/phonological property.

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. Reduplicated 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: the human language learner needs to represent reduplication and to pair bare and reduplicated forms from raw unlabeled acoustic data (for hearing learners). 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 (ϕ_1 and ϕ_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 dependencies that deep neural networks can and cannot learn has been an ongoing line of inquiry. The results of the computational experiments presented in this paper suggest that the Generator network learns to extend the learned identity-based patterns 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 (2020b). First, we identify variables that correspond to some phonetic/phonological representation such as presence of [s], based on Beguš (2020b), which proposes that setting single variables well above training range can reveal the underlying value for each latent variable and forces 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]. We also excluded reduplicated and unreduplicated items that contain sequences that are acoustically similar to [s]. This suggests that the network extends reduplication to novel forms even in absence of acoustically similar reduplication patterns.

Thus, these experiments again confirm that the network uses individual latent variables to represent linguistically meaningful representations (Beguš, 2020b,a). 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 make apparent 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 output 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 world's languages feature a large number of other reduplicative patterns in which only a C, CVC, or other types of phonological content is copied. Additionally, reduplication can precede or follow the base or can be inserted inside the base item. This paper is thus also an appeal to use these well-understood identity-based patterns in speech with various degrees of complexity to further test which patterns deep convolutional networks can and cannot learn and how self-organization of meaningful representations and discretization of a continuous space emerges in deep convolutional networks.

Acknowledgements

This research was funded by a grant to new faculty at the University of Washington. I would like to thank Ella Deaton for reading the training data.

Declaration of interests

The author declares no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Adlam, B., Weill, C., Kapoor, A., 2019. Investigating under and overfitting in wasserstein generative adversarial networks. arXiv:1910.14137.

Alhama, R.G., Zuidema, W.H., 2018. Pre-wiring and pre-training: What does a neural network need to learn truly general identity rules? Journal of Artificial Intelligence Res. 61, 927–946. doi:10.1613/jair.1.11197.

Arjovsky, M., Chintala, S., Bottou, L., 2017. Wasserstein generative adversarial networks, in: Precup, D., Teh, Y.W. (Eds.), Proceedings of the 34th International Conference on Machine Learning, PMLR, International Convention Centre, Sydney, Australia. pp. 214–223. URL: http://proceedings.mlr.press/v70/arjovsky17a.html.

Beguš, G., 2020a. Ciwgan and fiwgan: Encoding information in acoustic data to model lexical learning with generative adversarial networks. arXiv:2006.02951.

Beguš, G., 2020b. Generative adversarial phonology: Modeling unsupervised phonetic and phonological learning with neural networks. Frontiers in Artificial Intelligence 3, 44. URL: https://www.frontiersin.org/article/10.3389/frai.2020.00044, doi:10.3389/frai.2020.00044.

Berent, I., 2013. The phonological mind. Trends in Cognitive Sciences 17, 319 – 327. URL: http://www.sciencedirect.com/science/article/pii/S1364661313001034, doi:https://doi.org/10.1016/j.tics.2013.05.004.

Bond, Z.S., Wilson, H.F., 1980. /s/ plus stop clusters in children's speech. Phonetica 37, 149–158. URL: https://www.karger.com/DOI/10.1159/000259988, doi:10.1159/000259988.

Brugiapaglia, S., Liu, M., Tupper, P., 2020. Generalizing outside the training set: When can neural networks learn identity effects? arXiv:2005.04330.

Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P., 2016. Infogan: Interpretable representation learning by information maximizing generative adversarial nets, in: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 29. Curran Associates, Inc., pp. 2172–2180. URL: http://papers.nips.cc/paper/6399-infogan-interpretable-representation-learning-by-information-maximizing-generative-adve
pdf.

Chomsky, N., Halle, M., 1968. The Sound Pattern of English. Harper & Row, New York.

Dolatian, H., Heinz, J., 2018. Modeling reduplication with 2-way finite-state transducers, in: Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology, Association for Computational Linguistics, Brussels, Belgium. pp. 66–77. URL: https://www.aclweb.org/anthology/W18-5807, doi:10.18653/v1/W18-5807.

Donahue, C., McAuley, J.J., Puckette, M.S., 2019. Adversarial audio synthesis, in: 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019, OpenReview.net. URL: https://openreview.net/forum?id=ByMVTsR5KQ.

Eloff, R., Nortje, A., van Niekerk, B., Govender, A., Nortje, L., Pretorius, A., Biljon, E., van der Westhuizen, E., Staden, L., Kamper, H., 2019. Unsupervised acoustic unit discovery for speech synthesis using discrete latent-variable neural networks, in: Proc. Interspeech 2019, pp. 1103–1107. doi:10.21437/Interspeech.2019-1518.

Endress, A.D., Dehaene-Lambertz, G., Mehler, J., 2007. Perceptual constraints and the learnability of simple grammars. Cognition 105, 577 – 614. URL: http://www.sciencedirect.com/science/article/pii/S0010027706002605, doi:https://doi.org/10.1016/j.cognition.2006.12.014.

Garofolo, J.S., Lamel, L., M Fisher, W., Fiscus, J., S. Pallett, D., L. Dahlgren, N., Zue, V., 1993. Timit acoustic-phonetic continuous speech corpus. Linguistic Data Consortium .

Gasser, M., 1993. Learning words in time: Towards a modular connectionist account of the acquisition of receptive morphology. Technical Report. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.73.6474.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets, in: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 27. Curran Associates, Inc., pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Hayes, B., 2009. Introductory Phonology. Wiley-Blackwell, Malden, MA.

Heinz, J., Idsardi, W., 2013. What complexity differences reveal about domains in language*. Topics in Cognitive Science 5, 111–131. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/tops.12000, doi:10.1111/tops.12000, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/tops.12000.

van der Hulst, H., 2013. Discoverers of the phoneme, in: Allan, K. (Ed.), The Oxford Handbook of the History of Linguistics. Oxford University Press, pp. 167–191. doi:10.1093/oxfordhb/9780199585847.013.0009.

Inkelas, S., Zoll, C., 2005. Reduplication: Doubling in Morphology. Cambridge Studies in Linguistics, Cambridge University Press. doi:10.1017/CBO9780511627712.

Lister-Turner, R., Chatterton, P., Clark, J.B., 1941. A grammar of the Motu language of Papua. 2nd ed. / edited by percy chatterton. ed., Government Printer Sydney.

MacMahon, M.K.C., 2013. Orthography and the early history of phonetics, in: Allan, K. (Ed.), The Oxford Handbook of the History of Linguistics. Oxford University Press, pp. 105–122. URL: https://www.oxfordhandbooks.com/view/10.1093/oxfordhb/9780199585847.001.0001/oxfordhb-9780199585847-e-6, doi:10.1093/oxfordhb/9780199585847.013.0006.

Marcus, G.F., 2001. The algebraic mind: Integrating connectionism and cognitive science. MIT press.

Marcus, G.F., Vijayan, S., Bandi Rao, S., Vishton, P.M., 1999. Rule learning by seven-month-old infants. Science 283, 77–80. URL: https://science.sciencemag.org/content/283/5398/77, doi:10.1126/science.283.5398.77, arXiv:https://science.sciencemag.org/content/283/5398/77.full.pdf.

McClelland, J.L., Plaut, D.C., 1999. Does generalization in infant learning implicate abstract algebra-like rules? Trends in Cognitive Sciences 3, 166 – 168. URL: http://www.sciencedirect.com/science/article/pii/S1364661399013200, doi:https://doi.org/10.1016/S1364-6613(99)01320-0.

Nelson, M., Dolatian, H., Rawski, J., Prickett, B., 2020. Probing rnn encoder-decoder generalization of subregular functions using reduplication. Proceedings of the Society for Computation in Linguistics 3, 31–42.

Prickett, B., Traylor, A., Pater, J., 2018. Seq2Seq models with dropout can learn generalizable reduplication, in: Proceedings of the Fifteenth Workshop on Computational Research in Phonetics, Phonology, and Morphology, Association for Computational Linguistics, Brussels, Belgium. pp. 93–100. URL: https://www.aclweb.org/anthology/W18-5810, doi:10.18653/v1/W18-5810.

Radford, A., Metz, L., Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434 .

Räsänen, O., Nagamine, T., Mesgarani, N., 2016. Analyzing distributional learning of phonemic categories in unsupervised deep neural networks. CogSci ... Annual Conference of the Cognitive Science Society. Cognitive Science Society (U.S.). Conference 2016, 1757–1762. URL: https://pubmed.ncbi.nlm.nih.gov/29359204.

Rodionov, S., 2018. info-wgan-gp. https://github.com/singnet/semantic-vision/tree/master/experiments/concept_learning/gans/info-wgan-gp.

Shain, C., Elsner, M., 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), Association for Computational Linguistics, Minneapolis, Minnesota. pp. 69–85. URL: https://www.aclweb.org/anthology/N19-1007.

Simon, N., Friedman, J., Hastie, T., Tibshirani, R., 2011. Regularization paths for cox's proportional hazards model via coordinate descent. Journal of Statistical Software 39, 1–13. URL: http://www.jstatsoft.org/v39/i05/.

Steinert-Threlkeld, S., Szymanik, J., 2020. Ease of learning explains semantic universals. Cognition 195, 104076. URL: http://www.sciencedirect.com/science/article/pii/S0010027719302495, doi:https://doi.org/10.1016/j.cognition.2019.104076.

Urbanczyk, S., 2017. Phonological and morphological aspects of reduplication. URL: https://oxfordre.com/linguistics/view/10.1093/acrefore/9780199384655.001.0001/acrefore-9780199384655-e-80, doi:10.1093/acrefore/9780199384655.013.80.

Wilson, C., 2006. Learning phonology with substantive bias: An experimental and computational study of velar palatalization. Cognitive Science 30, 945–982.

Wilson, C., 2018. Modeling morphological affixation with interpretable recurrent networks: sequential rebinding controlled by hierarchical attention, in: Kalish, C., Rau, M., Zhu, J., Rogers, T. (Eds.), CogSci 2018, pp. 2693–2698.

Appendix A. Training data

		voiceless						voiced					
V_2	C_3	labial		coronal		dorsal		labial		coronal		dorsal	
		pala	papala	tala	tatala	kala	kakala	bala	babala	dala	dadala	gala	gagala
	[l]	pali	papali	tali	tatali	kali	kakali	bali	babali	dali	dadali	gali	gagali
		palu	papalu	talu	tatalu	kalu	kakalu	balu	babalu	dalu	dadalu	galu	gagalu
		para	papara	tara	tatara	kara	kakara	bara	babara	dara	dadara	gara	gagara
[a]	[r]	pari	papari	tari	tatari	kari	kakari	bari	babari	dari	dadari	gari	gagari
		paru	paparu	taru	tataru	karu	kakaru	baru	babaru	daru	dadaru	garu	gagaru
		paya	papaya	taya	tataya	kaya	kakaya	baya	babaya	daya	dadaya	gaya	gagaya
	[y]	payi	papayi	tayi	tatayi	kayi	kakayi	bayi	babayi	dayi	dadayi	gayi	gagayi
		payu	papayu	tayu	tatayu	kayu	kakayu	bayu	babayu	dayu	dadayu	gayu	gagayu
		pila	pipila	—	—	—	—	bila	bibila	dila	didila	gila	gigila
	[l]	pili	pipili	—	—	—	—	bili	bibili	dili	didili	gili	gigili
		pilu	pipilu	—	—	—	—	bilu	bibilu	dilu	didilu	gilu	gigilu
		pira	pipira	—	—	—	—	bira	bibira	dira	didira	gira	gigira
[i]	[r]	piri	pipiri	—	—	—	—	biri	bibiri	diri	didiri	giri	gigiri
		piru	pipiru	—	—	—	—	biru	bibiru	diru	didiru	giru	gigiru
		piya	pipiya	—	—	—	—	biya	bibiya	diya	didiya	giya	gigiya
	[y]	piyi	pipiyi	—	—	—	—	biyi	bibiyi	diyi	didiyi	giyi	gigiyi
		piyu	pipiyu	—	—	—	—	biyu	bibiyu	diyu	didiyu	giyu	gigiyu
		pula	pupula	—	—	kula	kukula	bula	bubula	dula	dudula	gula	gugula
	[l]	puli	pupuli	—	—	kuli	kukuli	buli	bubuli	duli	duduli	guli	guguli
		pulu	pupulu	—	—	kulu	kukulu	bulu	bubulu	dulu	dudulu	gulu	gugulu
		pura	pupura	—	—	kura	kukura	bura	bubura	dura	dudura	gura	gugura
[u]	[r]	puri	pupuri	—	—	kuri	kukuri	buri	buburi	duri	duduri	guri	guguri
		puru	pupuru	—	—	kuru	kukuru	buru	buburu	duru	duduru	guru	guguru
		puya	pupuya	—	—	kuya	kukuya	buya	bubuya	duya	duduya	guya	guguya
	[y]	puyi	pupuyi	—	—	kuyi	kukuyi	buyi	bubuyi	duyi	duduyi	guyi	guguyi
		puyu	pupuyu	—	—	kuyu	kukuyu	buyu	bubuyu	duyu	duduyu	guyu	guguyu

Table A.3: All items in which C_1 is a voiceless or voiced stop used in training data. All items feature two unique repetitions and were read by an L1 American English speaker. The items are in transcription that was presented to the reader.

V_2	C_3		[m]		[n]		[v]
		mala	mamala	nala	nanala	vala	vavala
	[l]	mali	mamali	nali	nanali	vali	vavali
		malu	mamalu	nalu	nanalu	valu	vavalu
		mara	mamara	nara	nanara	vara	vavara
[a]	[r]	mari	mamari	nari	nanari	vari	vavari
		maru	mamaru	naru	nanaru	varu	vavaru
		maya	mamaya	naya	nanaya	vaya	vavaya
	[y]	mayi	mamayi	nayi	nanayi	vayi	vavayi
		mayu	mamayu	nayu	nanayu	vayu	vavayu
		mila	mimila	nila	ninila	vila	vivila
	[l]	mili	mimili	nili	ninili	vili	vivili
		milu	mimilu	nilu	ninilu	vilu	vivilu
		mira	mimira	nira	ninira	vira	vivira
[i]	[r]	miri	mimiri	niri	niniri	viri	viviri
		miru	mimiru	niru	niniru	viru	viviru
		miya	mimiya	niya	niniya	viya	viviya
	[y]	miyi	mimiyi	niyi	niniyi	viyi	viviyi
		miyu	mimiyu	niyu	niniyu	viyu	viviyu
		mula	mumula	nula	nunula	vula	vuvula
	[l]	muli	mumuli	nuli	nunuli	vuli	vuvuli
		mulu	mumulu	nulu	nunulu	vulu	vuvulu
		mura	mumura	nura	nunura	vura	vuvura
[u]	[r]	muri	mumuri	nuri	nunuri	vuri	vuvuri
		muru	mumuru	nuru	nunuru	vuru	vuvuru
		muya	mumuya	nuya	nunuya	vuya	vuvuya
	[y]	muyi	mumuyi	nuyi	nunuyi	vuyi	vuvuyi
		muyu	mumuyu	nuyu	nunuyu	vuyu	vuvuyu

Table A.4: All items in which C_1 is a [m], [n], or [v] used in training data. All items feature two unique repetitions and were read by an L1 American English speaker. The items are in transcription that was presented to the reader.

$C_1 =$	[a]	Count	[i]	Count	[u]	Count
	sala	4	sila	5	sula	5
	sali	5	sili	5	suli	5
	salu	5	silu	5	sulu	5
	sara	5	sira	5	sura	5
	sari	5	siri	5	suri	5
	saru	5	siru	5	suru	4
	saya	5	siya	5	suya	5
	sayi	5	siyi	5	suyi	4
	sayu	5	siyu	5	suyu	5

Table A.5: All items in which C_1 is a [s] with corresponding number of unique repetitions in the training data per item. These items were never reduplicated in the training data. All items were read by an L1 American English speaker. The items are in transcription that was presented to the reader.