CiwGAN and fiwGAN: Encoding information in acoustic data to model lexical learning with Generative Adversarial Networks

Gašper Beguš

Department of Linguistics, University of Washington, Guggenheim Hall 415H, Box 352425, Seattle, WA 98195

Abstract

How can deep neural networks encode information that corresponds to words in human speech into raw acoustic data? This paper proposes two neural network architectures for modeling unsupervised lexical learning from raw acoustic inputs, ciwGAN (Categorical InfoWaveGAN) and fiwGAN (Featural InfoWaveGAN), that combine a Deep Convolutional GAN architecture for audio data (WaveGAN; Donahue et al. 2019) with an information theoretic extension of GAN – InfoGAN (Chen et al., 2016), and propose a new latent space structure that can model featural learning simultaneously with a higher level classification. In addition to the Generator and the Discriminator networks, the architectures introduce a network that learns to retrieve latent codes from generated audio outputs. Lexical learning is thus modeled as emergent from an architecture that forces a deep neural network to output data such that unique information is retrievable from its acoustic outputs. The networks trained on lexical items from TIMIT learn to encode unique information corresponding to lexical items in the form of categorical variables in their latent space. By manipulating these variables, the network outputs specific lexical items. The network occasionally outputs innovative lexical items that violate training data, but are linguistically interpretable and highly informative for cognitive modeling and neural network interpretability. Innovative outputs suggest that phonetic and phonological representations learned by the network can be productively recombined and directly paralleled to productivity in human speech: a fiwGAN network trained on *suit* and *dark* outputs innovative *start*, even though it never saw *start* or even a [st] sequence in the training data. We also argue that setting latent featural codes to values well beyond training range results in almost categorical generation of prototypical lexical items and reveals underlying values of each latent code. Probing deep neural networks trained on well understood dependencies in speech bears implications for latent space interpretability, understanding how deep neural networks learn meaningful representations, as well as a potential for unsupervised text-to-speech generation in the GAN framework.

Keywords: artificial intelligence, generative adversarial networks, speech, lexical learning, neural network interpretability, text-to-speech

1. Introduction

How human language learners encode information in their speech is among the core questions in linguistics and computational cognitive science. Acoustic speech data is the primary source of linguistic input for hearing infants, and first language learners must learn to retrieve information from raw acoustic data. By the time language acquisition is complete, learners are able to not only

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

analyze, but also produce speech consisting of words (or lexical items henceforth) that carry meaning (Saffran et al., 1996, 2007; Kuhl, 2010). In other words, speakers learn to encode information in their acoustic output, and they do so by associating meaning-bearing units of speech (lexical items) with unique information. Lexical items in turn consist of units called *phonemes* that represent individual sounds. In fact, speakers not only produce lexical items that exist in their primary linguistic data, but are also able to generate new lexical items that consist of novel combinations of phonemes that conform to the phonotactic rules of their language. This points to one of the core properties of language: productivity (Hockett, 1959; Piantadosi and Fedorenko, 2017; Baroni, 2020).

Computational approaches to lexical learning have a long history. Modeling lexical learning can take many forms (for a comprehensive overview, see Räsänen 2012), but the shift towards modeling lexical learning from acoustic data, especially from raw unreduced acoustic data, has occurred relatively recently (Lee et al. 2015; Shafaei-Bajestan and Baayen 2018; Baayen et al. 2019, i.a.). Previously, the majority of models operated on either fully abstracted or already simplified features extracted from raw acoustic data. A variety of models have been proposed for this task including, among others, Bayesian and connectionist approaches (see, among others Goldwater et al. 2009; Feldman et al. 2009; Räsänen 2012; Heymann et al. 2013; Lee and Glass 2012; Elsner et al. 2013; Feldman et al. 2013; Lee et al. 2015; Arnold et al. 2017; Kamper et al. 2017; Shafaei-Bajestan and Baayen 2018; Baayen et al. 2019; Chuang et al. 2020).

As summarized in Lee et al. (2015), existing models of lexical learning that take some form of acoustic data as input can be divided into "spoken term discovery" models and "models of word segmentation" (Lee et al., 2015, 390). Proposals of the first approach most commonly involve clustering of similarities in acoustic data to establish a set of phonetic units from which lexical items are then established again based on clustering. The *word segmentation* models, on the other hand, "start from unsegmented strings of symbols and attempt to identify subsequences corresponding to lexical items" (Lee et al., 2015, 390).

Deep neural network models operating on acoustic data have recently been used to model phonetic, but not phonological or lexical acquisition. Several prominent autoencoder models have recently been proposed which are trained to represent data in a lower-dimensionality space (Räsänen et al., 2016; Eloff et al., 2019; Shain and Elsner, 2019). Clustering analyses of the reduced space in these autoencoder models suggest that the networks learn approximates to phonetic features. The disadvantage of the autoencoder architecture is that outputs simply reproduce inputs as closely as possible: the network's outputs are directly connected to its inputs, which is not an ideal setting for language acquisition. Current proposals in the autoencoder framework do not model phonological processes, and there is only an indirect relationship between phonetic properties and latent space.

Language acquisition has, to the author's knowledge, not been modeled with the GAN architecture prior to Beguš (2020), despite several aspects of the architecture that can be paralleled to language acquisition. Beguš (2020) proposes that phonetic and phonological learning can simultaneously be modeled as a dependency between latent space and output data in Deep Convolutional Generative Adversarial Networks (Goodfellow et al., 2014; Radford et al., 2015; Donahue et al., 2019). Unlike in the autoencoder architectures, the outputs of the GAN models are innovative, not directly connected to the inputs, and violate training data distributions in highly informative ways.

Despite their several advantages, to our knowledge, lexical learning has not yet been modeled with unsupervised generative deep convolutional neural network models. In this paper, we follow the proposal in Beguš (2020) that phonetic and phonological acquisition can be modeled as a dependency between latent space and generated data in the GAN architecture and add lexical learning component to the model. We modify the WaveGAN architecture and add the InfoGAN's Q-network (based partially on implementation in Rodionov 2018) to computationally simulate lexical learning from raw acoustic data. We introduce a deep convolutional network that learns

to retrieve the Generator's latent code and propose a new latent space structure that can model featural learning (fiwGAN). We train the networks on highly variable training data: lexical items from TIMIT database (Garofolo et al., 1993) that includes over 600 speakers from different dialectal backgrounds in American English. We present three computational experiments: on five lexical items in the ciwGAN architecture (Section 4.1), on ten lexical items in the ciwGAN architecture (Section 4.2), and on eight lexical items in the fiwGAN architecture (Section 4.3). Evidence for lexical learning emerges in all three experiments. The paper also features a section describing how to directly follow learning strategies of the Generator network (Section 4.1.2), a section on featural learning that discusses innovative outputs and productivity of the model (Section 4.4), and a section that proposes a technique for retrieving underlying representation of the latent variables in GANs (Section 4.5). We argue that exploration of innovative outputs and the latent space of deep neural networks trained on dependencies on speech data that are well understood due to extensive study of human phonetics and phonology in the past decades provides unique insights both for cognitive modeling and for neural network interpretability.

The basic principle in our proposal crucially differs from the existing computational treatments of lexical learning that employ clustering or classification of phonetic similarities from which a lexical inventory is established. The model proposed here does not primarily classify or cluster acoustic similarities into lexical items. Instead, the model is fully generative, in that a deep convolutional network generates innovative outputs and encodes unique information in its outputs. Lexical learning is modeled in the following way: a deep convolutional network learns to retrieve information from innovative outputs generated by a separate Generator network. The Generator network thus learns to generate data such that unique lexical information is retrievable from its acoustic outputs. Lexical learning is not *per se* incorporated in the model: instead, lexical learning emerges because the most informative way to generate outputs given speech data as input is to encode unique information into lexical items. The end result of the model is a Generator network that generates innovative data — raw acoustic outputs — such that each lexical item is represented by a unique code. Because the model diverges substantially from existing proposals of lexical learning, we will not compare its performance to the existing models — it would be difficult to find a metric to compare a fully generative model that outputs raw acoustic data. Instead, we propose to evaluate success in the model's performance in lexical learning with an inferential statistical technique — multinomial logistic regression (Section 4.1).

The model of lexical learning proposed here features some desired properties. First, he network is trained exclusively on raw unannotated acoustic data. Second, lexical learning emerges from the requirement on a deep convolutional network to output informative data. Only because associating a unique code in the latent space with lexical items is the optimal way to encode information such that another network will be able to retrieve it, does the lexical learning emerge. Third, the model is fully generative: a deep convolutional network (the Generator) generates raw acoustic outputs that correspond to lexical items in the training data. Crucially, the Generator network in the model does not simply replicate training data, but generates innovative outputs, because its main task is to increase the error rate of the network that distinguishes real from generated data (the Discriminator) and its outputs are not directly connected to the training data. Occasionally, the Generator outputs innovative data that violate distributions of the training data, but are linguistically interpretable and highly informative. The model thus features one of the basic properties of language: productivity. This allows us to compare lexical and phonological acquisition in language-acquiring children to the innovative generated data in the proposed computational model. The fiwGAN architecture has an additional advantage: it can model featural learning in addition to a higher level classification. This means that featural representations in phonology and phonetics can be modeled simultaneously with lexical learning. To be sure, there are also undesired

aspects of the model: one of the main undesired aspects of the current model is that the number of lexical classes that the network learns needs to be predetermined: the model requires a prior number of unique categories to encode lexical information to. Also, while the model learns from raw acoustic inputs, the individual lexical items in training data are sliced from the corpus (sliced at the lexical level rather on the phone level) instead of inferred by the model. These disadvantages are not insurmountable, but are left to be addressed in future work.

The proposed architectures and results of the computational experiments have implications for deep neural network interpretability as well as some basic implications for NLP applications. Beside modeling lexical learning, the novel latent space structure in the fiwGAN architecture can be employed as a general purpose unsupervised simultaneous feature extractor and classifier for audio data. We also propose a technique for exploring latent space representations: we argue that manipulating latent codes to marginal values that substantially excess the training range reveals underlying values for each latent code. Outputs generated with the proposed technique feature little variability and have the potential to reveal learning representations of the Generator network. The proposed model also allows a first step towards unsupervised text-to-speech synthesis at the lexical level using GANs: the Generator outputs specific lexical items when latent codes are set to different values.

2. Background

The main characteristics of the GAN architecture (Goodfellow et al., 2014) are two networks: the Generator and the Discriminator. The Generator generates data from latent space that is reduced in dimensionality (e.g. from a set of uniformly distributed variables z). The Discriminator network learns to distinguish between "real" training data and generated outputs from the Generator network. The Generator is trained to maximize the Discriminator's error rate; the Discriminator is trained to minimize its own error rate. In the DCGAN proposal (Radford et al., 2015), the two networks are deep convolutional networks. Recently, the DCGAN proposal was transformed to model audio data in WaveGAN (Donahue et al., 2019). The main architecture of WaveGAN is identical to that of DCGAN (Radford et al., 2015), with the main difference being that the Generator outputs a one-dimensional vector corresponding to time series data (raw acoustic output) and the Discriminator takes one-dimensional acoustic data as its input (as opposed to two-dimensional visual data in DCGAN). WaveGAN also adopts the Wasserstein GAN proposal for a cost function in GANs that improves training (Arjovsky et al., 2017). Instead of estimating the probability of whether the output is generated or real, WGAN estimates the Wasserstein distance between generated data and real data.

Learning is unsupervised in the GAN framework and the model results in a Generator that generates innovative outputs based on the principle of imitation, enforced by the Discriminator. Beguš (2020) models speech acquisition as a dependency between latent space and generated outputs in the GAN architecture. The paper proposes a technique for identifying latent variables that correspond to meaningful phonetic/phonological features in the output. The Generator network learns to encode phonetically and phonologically meaningful representations, such as the presence of a segment in the output with a subset of variables, i.e. with reduced representation. Using the technique proposed in Beguš (2020), we can identify individual variables that correspond to, for example, a sound [s] in the output. By manipulating these identified variables to values that are beyond the training range, we can force [s] in the output. Interpolating the values has an almost linear effect on the amplitude of frication noise of [s] in the output.

One of the advantages of the proposal in Beguš (2020) is that the model learns phonological alternations, i.e. context-dependent changes in realization of speech sounds, simultaneously with

learning of acoustic properties of human speech. The WaveGAN model is trained on a simple phonological process: aspiration of stops /p, t, k/ conditioned on the presence of [s] in the input. English voiceless stops /p, t, k/ are aspirated (produced with a puff of air [p^h, t^h, k^h]) word-initially before a stressed vowel (e.g. in *pit* ['p^hɪt]) except if an [s] precedes the stop (e.g. *spit* ['spɪt]) A computational experiment suggests that the network learns this distribution, but imperfectly so. The network learns to output shorter aspiration duration when [s] is present, in line with distributions in the training data. Outputs, however, also violate data in a manner that can be directly paralleled to language acquisition. Occasionally, the Generator network outputs aspiration durations that are longer in the [s] condition than in any example in the training data: the generator outputs [sp^{h}ɪt], which violates the phonological rule in English. In other words, the network violates the distributions in the training data, and these violations correspond directly to phonological acquisition stages: children acquiring English start with significantly longer aspiration durations in the [s]-condition, e.g. [sp^{h}ɪt] (Bond and Wilson, 1980).

In sum, GANs have been shown to represent phonetically or phonologically meaningful information in the latent space that has approximate equivalent in phonetic/phonological representations and language acquisition. The latent variables that correspond to features can be actively manipulated to generate data with or without some phonetic/phonological properties. These representations, however, are limited to the phonetic/phonological level exclusively in Beguš (2020) and contain no lexical information.

We propose a GAN architecture that combines WaveGAN with the InfoGAN proposal (Chen et al., 2016). The InfoGAN proposal introduces the latent code to the GAN architecture and a Q-network. The Q-network usually shares convolutions with the Discriminator, but instead of estimating the "realness" of the generated and real inputs, it estimates the latent code that the Generator takes as an input. The weights of both the Q-network and the Generator networks are updated based on the Q-network's loss function. This forces the Generator network to output data such that the Q-network is successful in retrieving its latent code.

Representing semantic information can take many forms in computational models. The current proposal is a model of lexical learning, which is why unique lexical items are represented with either a one-hot vector in the ciwGAN architecture or a binary code in the fiwGAN architecture. In other words, the objective of the model is to associate each unique lexical item in the training data with a unique representation. For example, in a corpus with four words, *word1* can be associated with a representation [1, 0, 0, 0], *word2* with [0, 1, 0, 0], *word3* with [0, 0, 1, 0] in the ciwGAN architecture. In the fiwGAN architecture, *word1* can be associated with [0, 0], *word2* with [0, 1], *word3* with [1, 0], et cetera.

3. Model

The proposed ciwGAN and fiwGAN architectures involve three deep convolutional networks: the Generator, the Discriminator, and the Q-network (or the lexical learner). The models are based on WaveGAN (Donahue et al., 2019), an implementation of the DCGAN architecture (Radford et al., 2015) for audio data and the InfoGAN proposal (Chen et al., 2016).[1] Unlike in most InfoGAN implementations, the Q-network is a separate deep convolutional network[2].

[1]Barry and Kim (2019) in a recent presentation models piano with InfoWaveGAN. Their proposal, however, focuses on continuous variables and feature only one categorical latent variable with no apparent function. It is unclear from the poster what the architecture of their proposal is.

[2]The InfoGAN model based on DCGAN in Rodionov (2018) also proposes the Q-network to be a separate network.

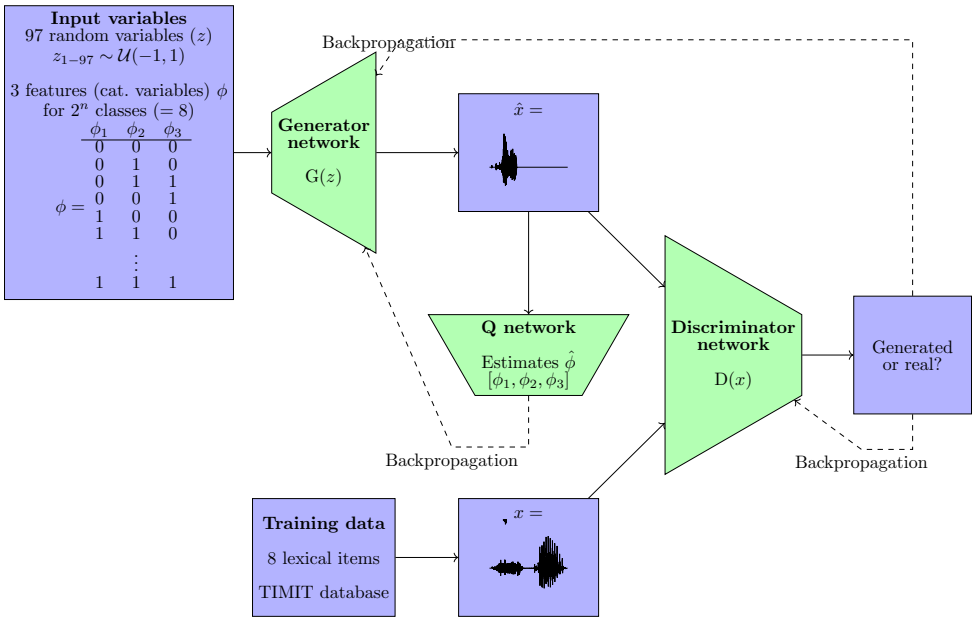

Figure 1: Architecture of fiwGAN: green trapezoids represent deep convolutional neural networks; purple squares illustrate inputs to each of the three networks. The Generator network takes 3 latent features ϕ (constituting binary code) and 97 latent variables z uniformly distributed ($z \sim \mathcal{U}(-1, 1)$) as its input. The Generator outputs a vector of 16384 values ($\hat{x}$) that constitute approximately 1 s of audio file (sampled at 16000 Hz). The Discriminator takes generated data ($\hat{x}$) and real data and estimates the Wasserstein distance between them. The Q-network (lexical learner) takes generated data as its input and outputs estimates of the unique feature values that the Generator uses for generation of each data point.

In the GAN architecture, the Generator network usually takes as its input a number of uniformly distributed latent variables ($z \sim \mathcal{U}(-1, 1)$). In the InfoGAN proposal (Chen et al., 2016), the Generator's input additionally includes a latent code: a set of binary variables that constitutes a one-hot vector as well as uniformly distributed code variables. Because we model lexical learning, we exclude uniformly distributed code variables. While the binary variables in InfoGAN implementations usually constitute a one-hot vector, we propose two different architectures. The ciwGAN architecture includes a one-hot vector as its latent code (c); but in the fiwGAN architecture, we introduce binary code as the categorical input (labeled as ϕ). This new structure in the fiwGAN latent space allows the network to treat the binary variables as features, where each variable corresponds to one feature (ϕ_n). As a consequence, the two networks differ in how the Q-network is trained. In ciwGAN, the Q-network is trained on retrieving information from the Generator's output with a softmax function in its final layer. In fiwGAN, the categorical variables or features are binomially distributed and the Q-network is trained to retrieve information with a sigmoid function in the final layer accordingly. In sum, the Generator in our proposal takes two sets of variables as its input (latent space): (i) categorical variables (c or ϕ) which constitute a one-hot vector (ciwGAN) or a binary code (fiwGAN) and (ii) random variables z that are uniformly distributed ($z \sim \mathcal{U}(-1, 1)$). Figure 1 illustrates the fiwGAN architecture. The code is available at `github.com/gbegus/ciwgan-fiwgan`.

The Generator network is a five-layer deep convolutional network (from WaveGAN; Donahue et al. 2019) that takes the input variables (referred to as the latent variables or the latent space) and outputs a 1D vector of 16,384 datapoints that constitute just over 1 second of acoustic audio output with 16 kHz sampling rate. These generated outputs are fed to the Discriminator network and the

Q-network. The Discriminator network takes raw audio as its input: both generated data and real data sliced at the lexical level from the TIMIT database (Garofolo et al., 1993). It is trained on estimating the Wasserstein distance between generated and real data distributions, according to Arjovsky et al. (2017). It outputs "realness" scores which estimate how far from the real data distribution an input is (Brownlee, 2019). The Generator's objective is to increase the error rate of the Discriminator: such that the Discriminator assigns high "realness" score to its generated outputs.

To model lexical learning, we add the Q-network to the architecture (InfoGAN; Chen et al. 2016). Unlike in most InfoGAN implementations, the Q-network is in the proposed architecture independent of the Discriminator network. The Q-network is thus a separate network, but in its architecture identical to the Discriminator. It takes only generated outputs ($G(z)$) as its input in the form of 16384 data points (approximately 1 s of audio data sampled at 16 kHz). The Q-network has 5 convolutional layers. The only difference between the Discriminator and the Q-network is that the final layer in the Q-network includes n number of nodes, where n corresponds to the number of categorical variables (c in ciwGAN) or features (ϕ in fiwGAN) in the latent space.

The Q-network is trained on estimating the categorical part of the latent space (c- or ϕ-values). Its output is thus a unique code that approximates the latent code in the Generator's latent space — either a one-hot vector or a binary code. The training objective of the Q-network is to approximate the unique latent code in the Generator's hidden input. At each evaluation, weights of the Q-network as well as the Generator network are updated with cross-entropy according to the loss function of the Q-network. This forces the Generator to generate data, such that the latent code or latent features (c or ϕ) will be retrievable: the Generator's objective is to maximize the success rate of the Q-network. The Generator and the Discriminator networks are trained with the Adam optimizer, whereas the Q-network is trained with the RMSProp algorithm (with the learning rate set at .0001 for all optimizers). The Generator and the Q-networks are updated once per five updates of the discriminator network.

To summarize the architecture, the Discriminator network learns to distinguish "realness" of generated speech samples. The Generator is trained to maximize the loss function of the Discriminator. The Q-network (or the lexical learner network) is trained on retrieving the categorical part of the latent code in the Generator's output based on only the Generator's acoustic outputs. Because the weights of the Q-network as well as the Generator are updated based on the Q-network's loss function, the Generator learns to associate lexical items with a unique latent code (one-hot vector or binary code), so that the Q-network can retrieve the code from the acoustic signal only. This learning that resembles lexical learning is unsupervised: the association between the code in the latent space and individual lexical items arises from training and is not pre-determined. In principle, the Generator could associate any acoustic property with the latent code, but it would make harder for the Q-network to retrieve the information if the Generator encoded some other distribution with its latent code. The association between a unique code value and individual lexical item that the Generator outputs thus emerges from the training.

The result of the training in the architecture outlined in Figure 1 is a Generator network that outputs raw acoustic data that resemble real data from the TIMIT database, such that the Discriminator becomes unsuccessful in assigning "realness" scores (Brownlee, 2019). Crucially, unlike in other architectures, the Generator's outputs are never a full replication of the input: the Generator outputs innovative data that resemble input data, but also violate many of the distributions in a linguistically interpretable manner (Beguš, 2020). In addition to outputting innovative data that resemble speech in the input, the Generator also learns to associate each lexical item with a unique code in its latent space. This means that by setting the code to a certain value, the network should output a particular lexical item to the exclusion of other lexical items.

word	IPA	data points
oily	[ˈɔɪli]	638
rag	[ˈɹæg]	638
suit	[ˈsut]	630
water	[ˈwɔɾɚ]	649
year	[ˈjɪɹ]	650
Total		3205

Table 1: Five lexical items use for training in the five-word ciwGAN model with their corresponding IPA transcription (based on general American English) and counts of data points for each item.

There are two supervised aspects of the model. First, the number of classes needs to be predetermined and the number of lexical items in the training data needs to match the number of classes. For example, a one-hot vector in the ciwGAN architecture with 5 variables can categorize 5 lexical items. We feed the network with 5 different lexical items from the TIMIT database. In the fiwGAN architecture, n features (ϕ) can categorize 2^n classes. For example, 3 features ϕ allow $2^3 = 8$ classes and we feed the network 8 different lexical items. Second, the network is trained on sliced lexical items and does not performed slicing in an unsupervised manner. Addressing these two disadvantages is left for future work.

4. Experiments

4.1. ciwGAN on 5 lexical items

4.1.1. 8011 steps

The first model is trained on the ciwGAN architecture with 5 lexical items from TIMIT: *oily*, *rag*, *suit*, *water*, and *year*. The latent space of this network includes 5 categorical variables (c) constituting a five-level one-hot vector and 95 random variables z. The five lexical items were chosen based on frequency: they are chosen from the most frequent content words with at least 600 data points in TIMIT. A total of 3205 data points were used in training and each of the five items has > 600 data points in the training data. The input data are 16-bit .wav slices of lexical items (as annotated in TIMIT) sampled with 16 kHz rate. Input lexical items with counts are given in Table 1.

Since we are primarily interested in a generative model of lexical learning, we test the model's performance on generated outputs. To test whether the Generator network learns to associate each lexical item with a unique code, the ciwGAN architecture is trained after 8011 ($\sim$ 800 epochs) and 19244 steps ($\sim$ 1921 epochs) and 100 outputs are generated for each one-hot vector. Beguš (2020) show that manipulating the latent space of the Generator network to values outside of the training interval can reveal the underlying feature encoded with each variable. Additionally, Beguš (2020) argues that the relationship between the latent variables and meaningful phonetic properties can be almost linear. Based on these findings, the code variables (c) in the generated samples are manipulated not to 1 (as in the training stage), but rather to 2. The rest of the latent space (all z-variables) are sampled randomly, but kept constant across the five categorical variables.

One hundred outputs are thus generated for each unique code (e.g. [2, 0, 0, 0, 0], [0, 2, 0, 0, 0] ...).[3] We analyze outcomes at two points during the training: after 8011 steps ($\sim$ 800 epochs) and after 19244 steps ($\sim$ 1921 epochs). Since we are modeling language acquisition, we are not

[3]All acoustic analyses are performed in Praat (Boersma and Weenink, 2015).

interested in full convergence of the model: it is more informative to probe the network as it is being trained. The number of steps at which we probe the networks are somewhat arbitrary, but the main consideration in choosing the number of steps is a balance between interpretability of outputs and minimizing the number of epochs (for a more detailed discussion, see Beguš 2020). The outputs were analyzed by a phonetically trained female speaker of American English who is not a co-author in this research and was not aware of the exact details of the experiment. The results below are reported based on the transcriber's analysis as well as based on an acoustic analysis by the authors. Altogether, 1000 outputs are thus analyzed and transcribed.

Results of the analysis suggest that the network associates each unique code with a different lexical item. The success rate, however, differs across lexical items. For example, when the latent code is set at [0, 0, 0, 0, 2] the Generator trained after 8011 steps outputs samples that are transcribed as *rag* in 98/100 cases. In other words, the Generator learns to associate [0, 0, 0, 0, 2] with *rag*.[4] The Generator thus not only learns to generate speech-like outputs, it also represents distinct lexical items with a unique representation: information that can be retrieved from its outputs by the lexical learning network. We can argue that [0, 0, 0, 0, 2] is the underlying representation of *rag*.[5]

To determine the underlying code for each lexical item, we use success rates (or estimates from the multinomial logistic regression model in Table 2 and Figure 4): the lexical item that is the most frequent output for a given latent code is assumed to be associated with that latent code (e.g. *rag* with [0, 0, 0, 0, 2]). Occasionally, a single lexical item is the most frequent output for two latent codes. As will be shown below, it is likely the case that this reflects imperfect learning where the underlying lexical item for a latent code is obscured by a more frequent output (perhaps the one that is easier to distinguish from the data). In this case, we associate such codes to the lexical item for which the given code outputs the highest proportion of that lexical item with respect to other latent codes. For example, [0, 0, 0, 2, 0] outputs *water* most frequently with *oily* accounting for approximately a quarter of outputs. The assumed lexical item for [0, 0, 0, 2, 0] is *oily*, because the code that outputs *water* most frequently is [0, 0, 2, 0, 0], while highest proportion of *oily* relative to other latent codes is [0, 0, 0, 2, 0]. Observing the progress of lexical learning provides additional evidence that *oily* is the underlying representation of [0, 0, 0, 2, 0]: as the training progresses the network increases accuracy (see Section 4.1.2).

The success rate for the other four lexical items is lower than for *rag*, but the outputs that deviate from the expected values are highly informative. For $c = [2, 0, 0, 0, 0]$, the Generator (8011 steps) outputs 72/100 data points that can be reliably transcribed as *suit*. In seven additional cases, the network outputs data points that can be transcribed with a sibilant [s] (79 total). In these outputs, [s] is followed by a sequence that either cannot reliable be transcribed as *suit* or does not correspond to *suit*, but rather to *year* (transcribed as *sear* [sɪɹ]). The remaining 21 outputs do not include the word for *suit* or a sibilant [s]. However, they are not randomly distributed across other four lexical items either — they include lexical item *year* or its close approximation.

An acoustic analysis of the training data reveals motivations for the innovative deviating outputs. As already mentioned, the network occasionally generates an innovative output, *sear*. The sources of this innovation are likely four cases in the training data in which [j] in *year* ([jɪɹ]) is realized as a post-alveolar fricative [ʃ], probably due to contextual influence (something that could be transcribed as *shear* [ʃɪɹ]). Figure 2 illustrates all four examples. The innovative generated

[4] Occasionally, a short vocalic element precedes the [ˈɹæg].

[5] In the remaining two cases, the outputs include [ɹ] in the initial position, which is followed by a diphthong [aɪ] and a period of a consonantal closure. One output was transcribed as *right*.

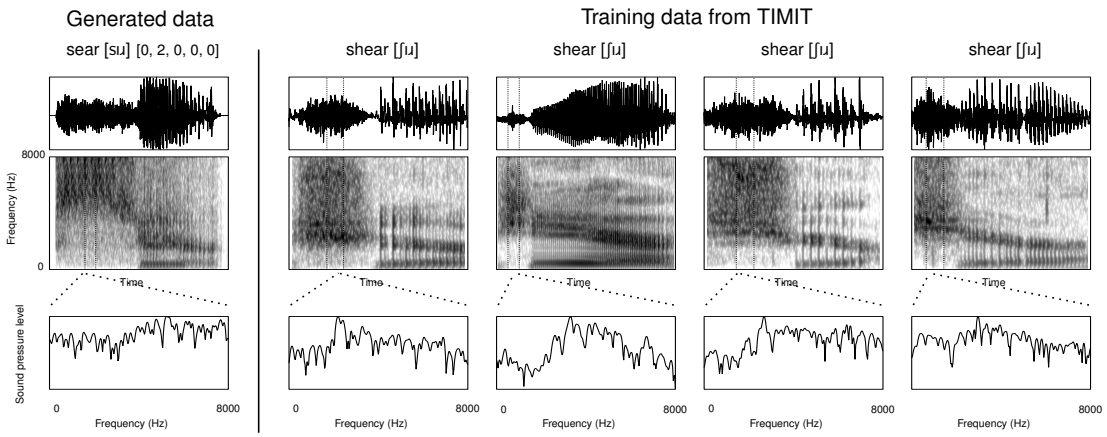

Figure 2: Waveforms (top), spectrograms (mid, from 0–8000 Hz), and 25 ms spectra (slices indicated in the spectrograms) (bottom) of four data points of the lexical item *year* with clear frication noise in the training data (from TIMIT) and the generated innovative output *sear*.

output *sear* differs from the four examples in the training data in one crucial aspect: the frication noise in the generated output is that of a post-alveolar [s] rather than that of a palato-alveolar [ʃ]. Spectral analysis in Figure 2 clearly shows that the center of gravity in the generated output is substantially higher than in the training data (which is characteristic of the alveolar fricative [s]).

The innovative *sear* output likely results from the fact that the training data contains four data points that pose a learning problem: *shear* that features elements of *suit* and *year*. The innovative generated *sear* [sɹɹ] consequently features (i) frication noise that is approximately consistent with *suit* [sut] and (ii) formant structure consistent with *year* [jɹɹ]. It appears that the network treats *sear* as a combination of the two lexical items. The network generates innovative outputs that combines the two elements (*sear* [sɹɹ]). Additionally, the *sear* output seems to be equally distributed among the two latent codes, [2, 0, 0, 0, 0] representing *suit* and [0, 2, 0, 0, 0] representing *year*. In other words, the error rate distribution of the two latent codes suggests that the network classifies the output *sear* as the combination of elements consistent with [2, 0, 0, 0, 0] and [0, 2, 0, 0, 0].

For $c = [0, 2, 0, 0, 0]$, the Generator outputs 68 data points that can be reliably transcribed as *year* or at least have a clear [ɹɹ] sequence (without an [s]).[6] 22 outputs feature a sibilant [s]. In 22 cases, 16 can reliably be transcribed as *suit*, while the others are mostly variants of the innovative *sear*. The remaining cases (approximately 10) are difficult to categorize based on acoustic analysis.

For [0, 0, 2, 0, 0], the Generator outputs 84 data points that are transcribed as containing *water* [wɔɾɚ]. In approximately 15 of the 84 cases, the output involves an innovative combination transcribed as *watery* [ˈwɔɾəɹi]. Figure 3 illustrates one such case. *Watery* is an innovative output that combines segment [i] from *oily* ([ˈɔɪli]) with [ˈwɔɾəɹ] from *water* into a linguistically interpretable innovation. This suggests that the Generator outputs a novel combination of segments, based on analogy to *oily*. Unlike for *sear*, the training data contained no direct motivations based on which *watery* could be formed.[7]

Finally, for [0, 0, 0, 2, 0], the Generator outputs only 27 outputs that can reliably be transcribed

[6]The initial consonant is sometimes absent from transcriptions, but this is primarily because the glide interval is acoustically not prominent, especially before [ɹ].

[7]In 10 further cases of [0, 0, 2, 0, 0], the Generator outputs data points that contain a sequence *oil* [ˈɔɪl]. Transcription of the remaining 6 outputs is uncertain.

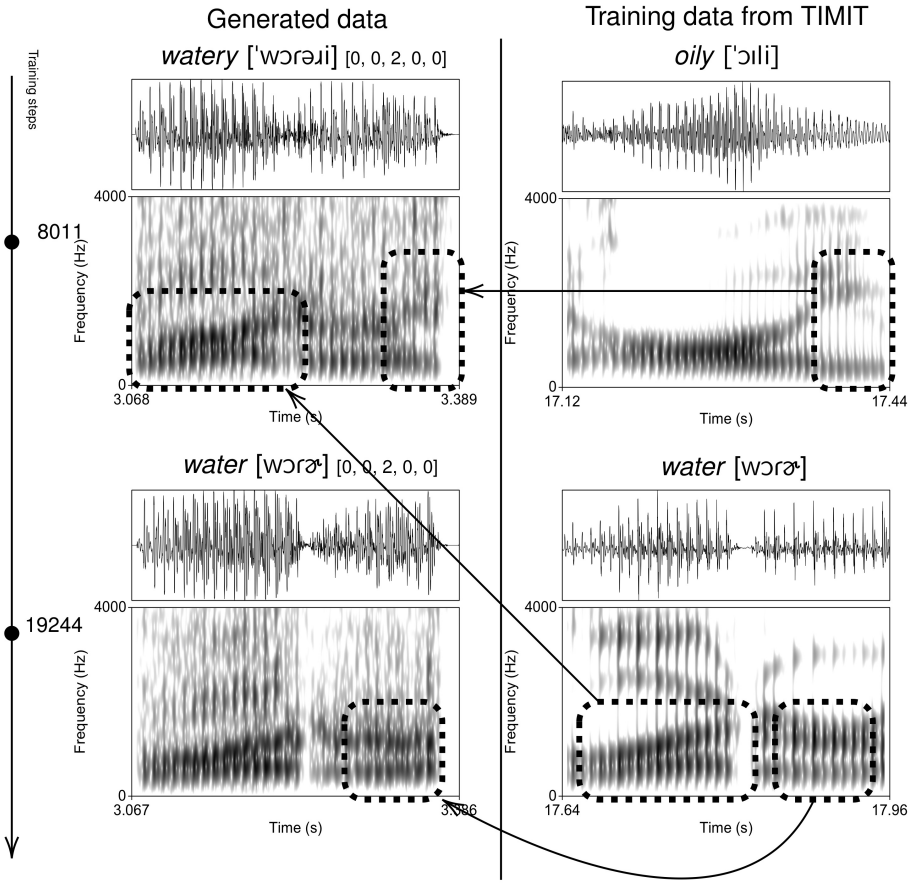

Figure 3: A waveform and spectrogram (0–4000 Hz) of an innovative output *watery* [wɔrəɹi] (top right). That innovative *watery* is a combination of *water* ['wɔrɚ] and *oily* ['ɔɪli] is illustrated by two examples from the training data (top and bottom right). The innovative output *watery* features a clear formant structure of *water* with a high front vowel [i], characteristic of the lexical item *oily* (see marked areas of the spectrograms). At 19244 steps, the vocalic structure of [i] is not present in the output, given the exact same latent code and random latent space. The network thus corrects the formant structure from an innovative *watery* into *water* [wɔrəɹ(ə)] as the training progresses. In some other cases, the network at 19244 steps outputs *oily* for what was *watery* at 8011 steps.

Assumed word	Latent code c	Most frequent	%	2nd most freq.	%	Else
suit	[2, 0, 0, 0, 0]	*suit* [ˈsut]	72%	*year* [ˈjɪɹ]	21%	7%
year	[0, 2, 0, 0, 0]	*year* [ˈjɪɹ]	70%	*suit* [ˈsut]	12%	18%
water	[0, 0, 2, 0, 0]	*water*[ˈwɔɾɚ˞]	84%	*oily* [ˈɔɪli]	10%	6%
oily	[0, 0, 0, 2, 0]	*water* [ˈwɔɾɚ˞]	61%	*oily* [ˈɔɪli]	26%	13%
rag	[0, 0, 0, 0, 2]	*rag* [ˈɹæg]	98%	—	—	2%

Table 2: Generated outputs and their percentages across the five one-hot vectors in the latent code. Transcriptions of the outputs were coded as detailed in footnote 8.

as *oily* [ˈɔɪli]. On the other hand, 61 outputs contain *water*. *Oily* is the less frequent output for [0, 0, 0, 2, 0] compared to *water*, but *water* is assigned to [0, 0, 2, 0, 0] because it is its most frequent output, while [0, 0, 0, 2, 0] is the code that outputs the highest proportion of *oily*. This is why we analyze *oily* as the underlying lexical item for the [0, 0, 0, 2, 0] code. Another evidence that *oily* might underly the [0, 0, 0, 2, 0] code is that as the training progresses, the Generator increases the number of outputs transcribed with *oily* for this code and decreases the number of outputs *water* for the same code (see Section 4.1.2 and Figure 4). For a confirmation that the proposed method for assigning underlying assumed words for a given code based on annotated outputs yields valid results, see Section 4.5.

To evaluate lexical learning in the ciwGAN model statistically, we analyze the results with a multinomial logistic regression model. To test significance of the latent code as the predictor of the lexical item, annotations of the generated data were coded and fit to a multinomial logistic regression model using the *nnet* package (Venables and Ripley, 2002) in R Core Team (2018). The dependent variable is the transcriptions of the generated outputs for the five lexical items and the *else* condition.[8] The independent variable is a single predictor: the latent code with the five levels that correspond to the five unique one-hot values in the latent code. The difference in AIC between the model with the latent code as a predictor ($AIC = 674.7$) vs. the empty model ($AIC = 1707.1$) suggests that the latent code is indeed a significant predictor of the lexical item in the output. Counts are given in Table 2. Estimates from the multinomial logistic model in Figure 4 clearly show that each lexical item is associated with a unique latent code.

4.1.2. 19244 steps

The proposed model of lexical learning allows not only the ability to test learning of lexical items, but also to probe learning representations as training progresses. We propose that the progress of lexical learning can be directly observed by keeping the random variable constant across training steps. In other words, we train the Generator at various training steps and generate outputs for models trained after different number of steps with the same latent code (c) and the same latent variables (z). This reveals how encoding of lexical items with unique latent codes changes with training.

To probe lexical learning as training progresses, we train the 5-word model at 8011 steps for an additional 11233 steps (total 19244) and generate outputs. The generation is performed as described in Section 4.1.1: for each unique latent code (one-hot vector), we generate 100 outputs with latent variables identical to the ones used on the model trained after 8011 steps. The latent

[8]The following conditions were used for coding the transcribed output: if the annotator transcribed an output as containing "suit", the coded lexical item was *suit*), if "ear" or "eer" (and no "s" immediately preceeding), then year, if involving "water", "oily", and "rag", then *water*, *oily*, *rag*, respectively. In all other cases, the output was coded as *else*.

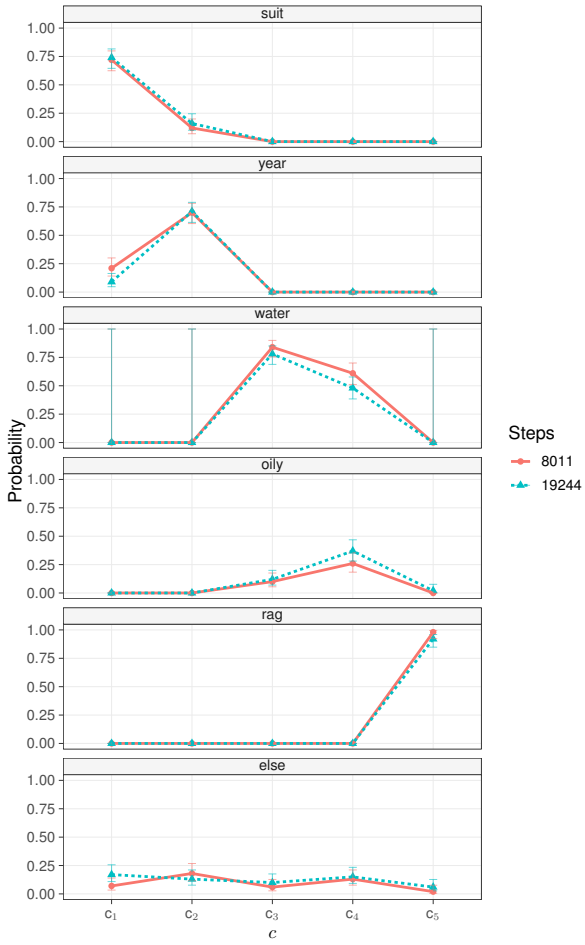

Figure 4: Estimates of two multinomial logistic regression models (for 5-word models trained after 8011 and 19244 steps) with coded transcribed outputs as the dependent variable and the latent code with five levels that correspond to the five unique one-hot vectors in the model.

code is again manipulated to value 2 (e.g. [2, 0, 0, 0, 0]) in order to probe the underlying effects of the latent code on generated outputs.

The latent code remains a significant predictor in a multinomial logistic regression model (AIC = 759.9 for a model with the predictor and 1760.2 for an empty model). In fact, success rates remain almost identical across the training steps as is clear from regression estimates in Figure 4 with one notable exception. The most substantial improvement in success rate is observed for *oily*: +10% in raw counts. The overall success rate is lowest in the 8011-step model precisely for lexical item *oily*. In fact, the success rate for [0, 0, 0, 2, 0] with assumed lexical representation *oily* is only 26%. At 19244 steps, the success rate (give the exact same latent variables) increases to 37%.[9]

Generating data with identical latent variables allows us to observe how the network transforms an output that violates the underlying lexical representation to an output that conforms to it. Figure 5 illustrates how an output *water* at 8011 steps for latent code [0, 0, 0, 2, 0] changes to *oily* at 19244 steps.[10] Both outputs have the same latent code and latent variables (z). Spectrograms in Figure 5 clearly show how the formant structure of *water* and its characteristic period of reduced amplitude for a flap [ɾ] change to a formant structure characteristic for *oily* with a consonantal period that corresponds to [l]. The figure also features spectrograms of two training data points, *water* and *oily*, which illustrate a degree of acoustic similarity between the two lexical items. Similarly, Figure 3 illustrates how an output *watery* that violates the training data in a linguistically interpretable manner at 8011 steps changes to *water* consistent with the training data.

In sum, the results of the first model suggest that the Generator in the ciwGAN architecture trained on 5 lexical items learns to generate innovative data such that each unique latent code corresponds to a lexical item. In other words, the network encodes unique lexical information in its acoustic outputs based solely on its training objective: to generate data such that unique code is retrievable from its outputs. Figure 4 illustrates that each lexical item is associated with a unique code. Modeling of lexical learning is thus fully generative: when the latent code is manipulated outside of the training range to value 2, the network mostly outputs one lexical items per unique code with error rates from approximately 98% to 27%. The errors are not randomly distributed: the pattern of errors as well as innovative outputs suggests that (i) *suit* and *year* and (ii) *water* and *oily* are the items that the Generator associates more closely together. Output errors fall almost exclusively within these groups. The Generator also outputs innovative data that violate training data distributions. Acoustic analysis of the training data reveals motivations for the innovative outputs. When we follow learning across different training steps, we observe the Generator's repair of innovative outputs or outputs that deviate from the expected values. The highest improvement is observed in the lexical item with overall highest error rate.

4.2. ciwGAN on 10 lexical items

To evaluate how the Generator performs on a higher number of lexical classes, another model was trained on 10 content lexical items from the TIMIT database, each of which is attested at least 600 times in the database. All 10 lexical items with exact counts and IPA transcriptions are listed in Table 3.

To evaluate lexical learning in a generative fashion, we use the same technique as on the 5-item Generator in Section 4.1. The Generator is trained for 27103 steps ($\sim$ 1346 epochs). The number of epochs is thus approximately at the halfway point between the models in Sections 4.1.1 (8011) and 4.1.2 (19244). We generate 100 outputs for each unique one-hot vector with the value set outside of

[9] The model does not seem to improve further after 46002 steps.

[10] Occasionally, a change in the opposite direction is also present.

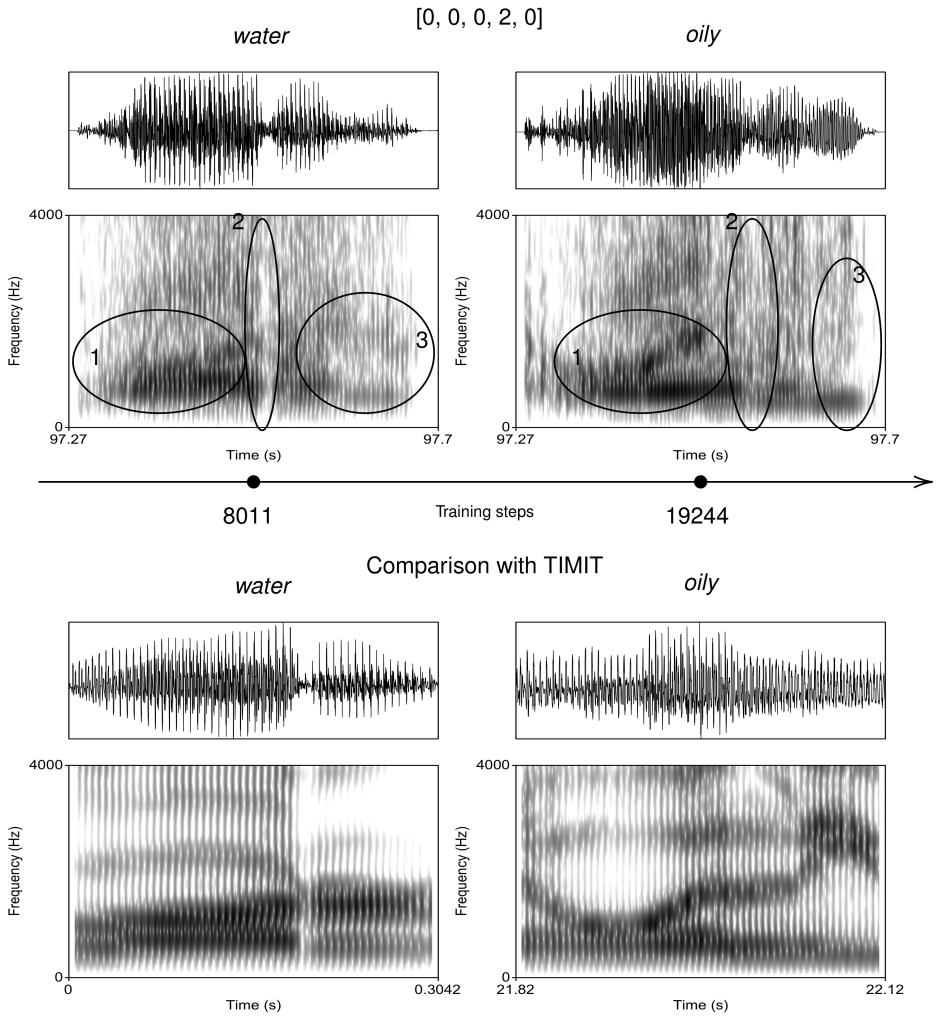

Figure 5: Waveforms and spectrograms (0-4000 Hz) of a generated output at 8011 steps (trained on five lexical items) for latent code [0, 0, 0, 2, 0] that can be transcribed as *water* (top left); and of a generated output for the exact same latent code as well as other 95 latent variables, but generated by a model trained after 19244 steps (top right) transcribed as *oily*. Circled areas point to three major changes on the spectrogram that occur from the output at 8011 steps to the output at 19244 steps: vocalic formants change from [wɔ] to [oɪ] (area1), periods characteristic of a flap [ɾ] change to [l] (area 2) and formant structure for [ɚ] turns into an [i]. Examples for *water* and *oily* from the TIMIT database (bottom left and right) illustrate close similarity of the generated outputs to the training data. While the opposite change (from *oily* to *water*) also occurs, it appears less common.

word	IPA	data points
ask	[ˈæsk]	633
carry	[ˈkʰæɹi]	632
dark	[ˈdɑɹk]	644
greasy	[ˈgɹisi]	630
like	[ˈlaɪk]	697
oily	[ˈɔɪli]	638
rag	[ˈɹæg]	638
suit	[ˈsut]	630
water	[ˈwɔɾɚ]	649
year	[ˈjɪɹ]	650
Total		6441

Table 3: Ten content lexical items from the TIMIT database used for training.

the training range to 2 (e.g. [2, 0, 0, 0, 0, 0, 0, 0, 0, 0]), while keeping the uniform latent variables (z) constant across the 10 groups. 1000 outputs were thus annotated.

Similarly to the 5-word model in Section 4.1.1, the generated data suggests that the Generator learns to associate each lexical item with a unique representation. To test the significance of the latent code as a predictor, the coded annotated data were fit to a multinomial logistic regression model (as described in Section 4.1).[11] The AIC test suggests that the latent code is a significant predictor ($AIC = 4555.6$ for an empty model vs. 1909.4 for a model with the predictor).

Estimates from the multinomial logistic regression model in Figure 6 illustrate that each unique one-hot vector is associated with a unique lexical item. Each lexical item has a single substantial peak in estimates per latent code. The only exception appears to be *rag* without a clear representation. The highest proportion of *rag* appears for $c_1 = 2$ at approximately 20%. However, this particular latent code ($c_1 = 2$) already outputs a substantially higher proportion of *dark*. It thus appears that the Generator fails to generate outputs such that the difference between the two outputs would be substantial. There is a high degree of phonetic similarity precisely between these two lexical items: the vowels [æ] and [ɑ] are acoustically similar and both lexical items contain a rhotic [ɹ] and a voiced stop. Success rates for the other nine lexical items range from 39%–99% in raw counts.

To illustrate that the network learns to associate lexical items with unique values in the latent code (one-hot vector), we generate outputs by manipulating the one-hot vector for each value and by keeping the rest of the latent space (z) constant. Such a manipulation can result in generated samples, where each latent space outputs a distinct lexical item associated with that value ([20000000000] outputs *dark*, [02000000000] *water*, etc.).[12] Note that the acoustic contents of the generated outputs that correspond to each lexical item are substantially different (as illustrated by the spectrograms in Figure 7), which means that the latent code (c) needs to be strongly associated with the individual lexical items, given that all the other 90 variables in the latent space (the z-variables which constitute 90% of all latent space) are kept constant and that the entire change of the output occurs only due to change of the latent code c. In other words, by only changing the

[11] The outputs were coded according to the following criteria: if transcription included "su[ie][td]", then *suit*, if "[^s]e[ae]r" then *year*, if "water" then *water*, if "dar" then *dark*, if "greas" then *greasy*, if "[kc].*r" then *carry*, if "[ao][wia]ly" then *oily*, if "rag" then *rag*, if "as" then *ask*, if "li" then like.

[12] Often each series outputs one or two divergences from the ideal output.

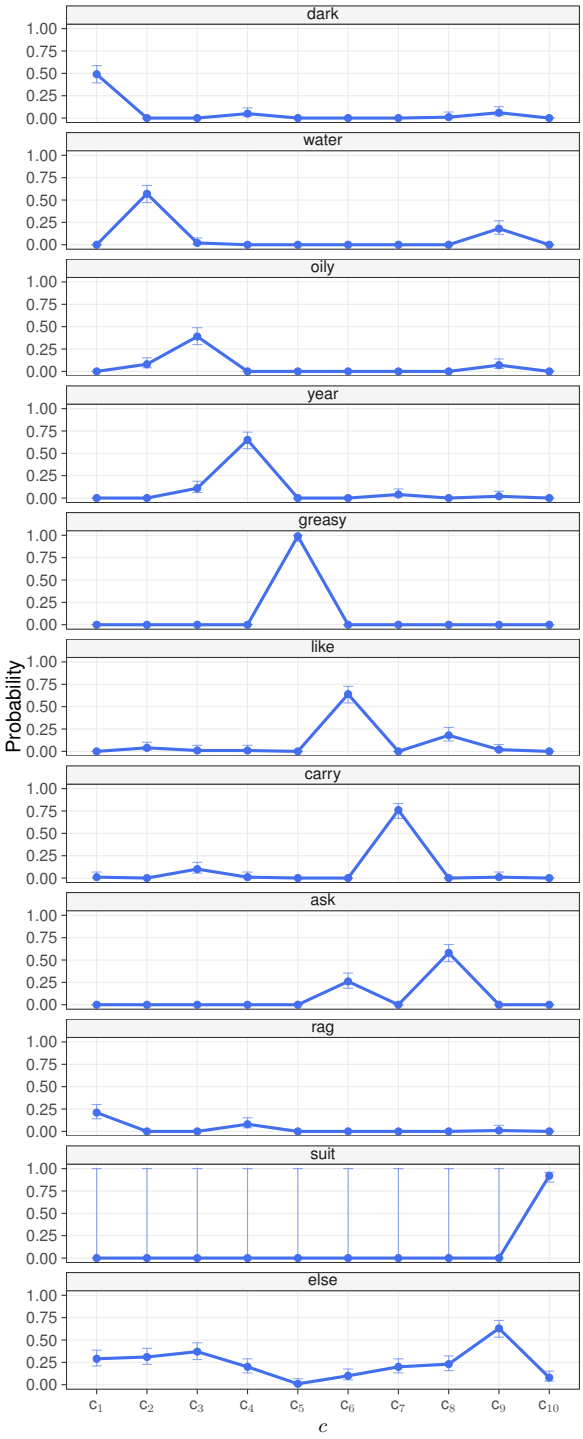

Figure 6: Estimates of a multinomial logistic regression model with coded transcribed outputs as the dependent variable and the latent code with five levels that correspond to the five unique one-hot vectors in the model trained on 10 lexical items from TIMIT after 27103 steps.

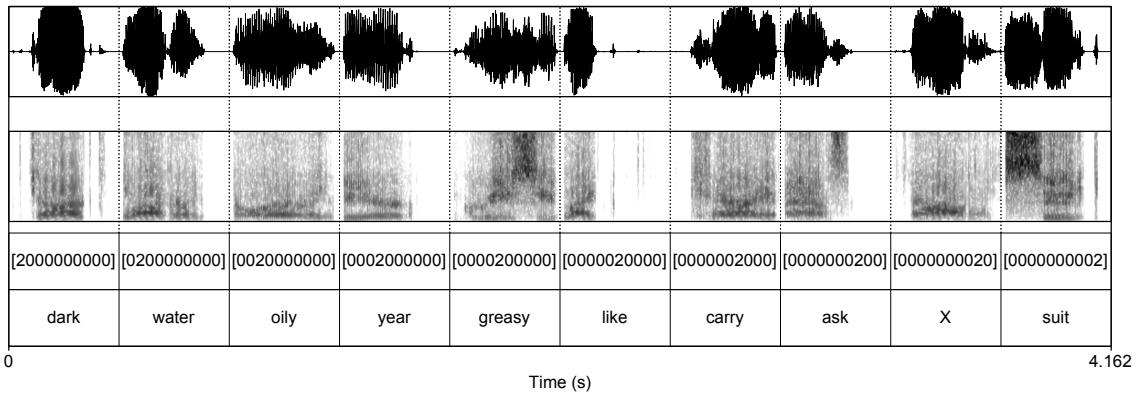

Figure 7: Waveforms and spectrograms (0-8000 Hz) of generated outputs (of a model trained on 10 items after 27103 steps) when only the latent code is manipulated and the remaining 90 latent random variables are kept constant across all 10 outputs. Transcriptions (by the authors) suggest that each lexical item is associated with a unique representation.

word	IPA	data points
ask	[ˈæsk]	633
carry	[ˈkʰæɹi]	632
dark	[ˈdɑɹk]	644
greasy	[ˈgɹisi]	630
like	[ˈlaɪk]	697
suit	[ˈsut]	630
water	[ˈwɔɾɚ]	649
year	[ˈjɪɹ]	650
Total		5165

Table 4: Eight content lexical items from the TIMIT database used for training in the fiwGAN architecture.

latent code and setting the variables to desired values while keeping the rest of the latent space constant, we can generate desired lexical items with the Generator network.

4.3. fiwGAN on 8 lexical items

To evaluate lexical learning in a fiwGAN architecture, we train the fiwGAN model with three featural variables (ϕ). Because the latent code in fiwGAN is binomially distributed, three featural variables correspond to $2^3 = 8$ categories. The model was trained on 8 content lexical items with more than 600 attestations in the TIMIT database (listed in Table 4). The model used for the analysis was trained after 20026 steps which correspond to a similar number of epochs as the 10-word ciwGAN model in Section 4.2 ($\sim$ 1241 epochs). Like for the ciwGAN models (Sections 4.1 and 4.2), we generate 100 outputs for each unique binary code given the 3 featural variables with the values of the features set outside of the training range to 2 instead of 1: [0, 0, 0], [0, 0, 2], [0, 2, 0], [0, 2, 2], [2, 0, 0], etc.

As expected, learning in the fiwGAN architecture is more challenging compared to ciwGAN. The network has only $\log_2(n)$ variables to encode n lexical items (compared to n variables for n classes in ciwGAN). Despite the latent space for lexical learning being reduced, an analysis of generated data in the fiwGAN architecture suggests that the Generator learns to associate each binary code with a distinct lexical item (for an additional test, see Section 4.5).

To test significance of the featural code (ϕ) as a predictor, the annotated data were fit to a multinomial logistic regression model as in Section 4.1 and 4.2. The dependent variables are again coded transcriptions[13] and the independent variable is the featural code (ϕ) with the eight unique levels as predictors: each for unique binary code. The difference in AIC between the model that includes the unique featural codes as predictors (ϕ) and the empty model (2038.5 vs. 3409.7) suggest that featural values are significant predictors.

Estimates of the regression model in Figure 8 illustrate that most lexical items receive a unique featural representation. Six out of eight lexical items (*dark, ask, suit, greasy, year* and *carry*) all have distinct latent featural representations that can be associated with these lexical items. Success rates for the six items have a mean of 50.8% (in raw counts) with the range of 46% to 61%. Crucially, there appears to be a single peak in regression estimates per lexical item for these six words, although the peaks are less prominent compared to the ciwGAN architecture (expectedly so, since learning is significantly more challenging in the featural condition). *Water* and *like* are more problematic: [0, 2, 0] outputs *like* and *water* at approximately the same rate. It is possible that learning of the two lexical items is unsuccessful. Another possibility is that [0, 2, 0] is the underlying representation of *water* because it is *water*'s most frequent code that is not already taken by another lexical item. According to the guidelines in Section 4.1, *like* would have to be represented by [0, 0, 0], because it outputs the highest proportion of *like* that is not already taken for another lexical items. That this assignment of underlying values of each featural representation is valid is additionally suggested by another test in Section 4.5.

In the fiwGAN architecture, we can also test significance of each of the three unique features (ϕ_1, ϕ_2, and ϕ_3). The annotated data were fit to the same multinomial logistic regression model as above, but with three independent variables: the three features each with two levels (0 and 2). AIC is lowest when all three variables are present in the model (2135.5) compared to when ϕ_1, ϕ_2, or ϕ_3 are removed from the model (2527.2, 2413.0, and 2773.3, respectively).

4.4. Featural learning

An advantage of the fiwGAN architecture is that it can model classification (i.e. lexical learning) and featural learning of phonetic and phonological representations simultaneously. We can assume that lexical learning is represented by the unique binary code for each lexical item. Phonetic and phonological information can be simultaneously encoded with each unique feature (ϕ). That phonetic and phonological information is learned with binary features has been the prevalent assumption in linguistics for decades (Clements, 1985; Hayes, 2009). Recently, neuroimaging evidence suggesting that phonetic and phonological information is stored as signals that approximate phonological features has been presented in Mesgarani et al. (2014).

An analysis of featural learning in fiwGAN — how featural codes simultaneously represent unique lexical items and phonetic/phonological representations, can be performed by using logistic regression as proposed in the following paragraphs as well as with a number of exploratory techniques described in this section.

Three out of eight lexical items used in training of the fiwGAN model include the segment [s]: a voiceless alveolar fricative with a distinct phonetic marker — a period of frication noise. The assumed binary codes for the three items containing [s], *ask*, *greasy*, and *suit* are [2, 2, 0], [2, 0, 0], and [2, 0, 2] (see Figure 8). We observe that value 2 for feature ϕ_1 is common to all three of the lexical items containing an [s].

[13]The outputs are coded as described in fn. 11 for the 10-word ciwGAN model, except that if "[ae].*[sf]", then *ask*, because outputs contain a large proportion of *s*-like frication noise that can also be transcribed with *f*.

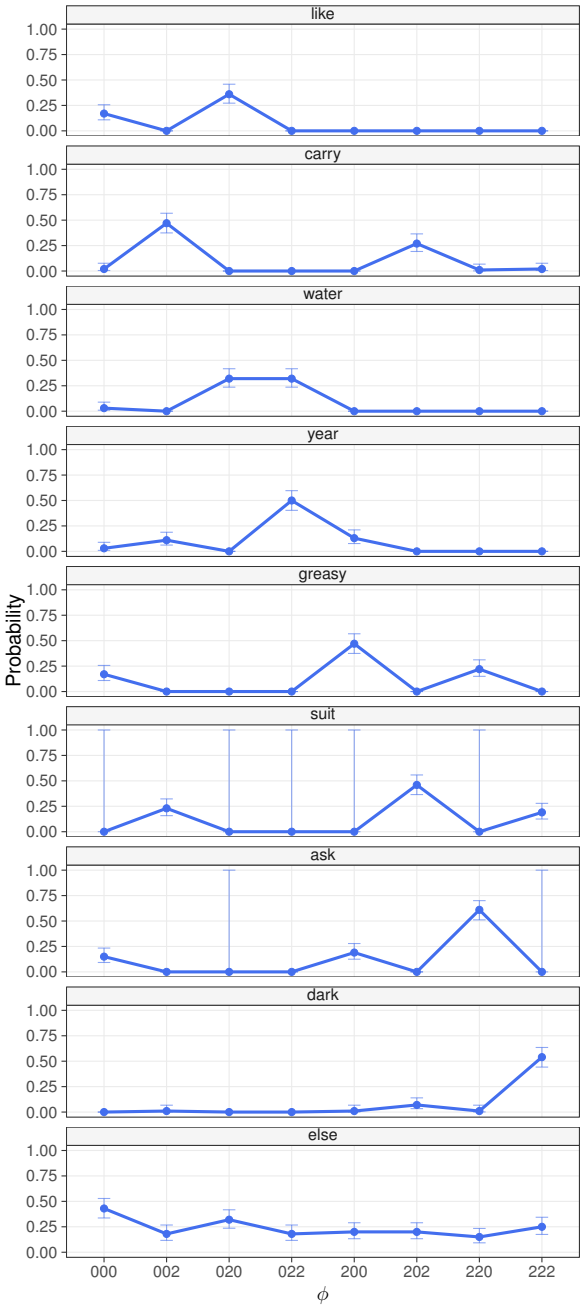

Figure 8: Estimates of a multinomial logistic regression model with coded transcribed outputs as the dependent variable and the latent code with five levels that correspond to the five unique one-hot vectors in the model trained on 8 lexical items from TIMIT after 20026 steps.

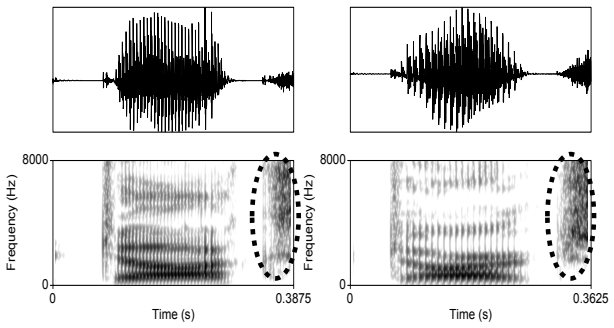

Figure 9: Waveforms and spectrograms (0-8000 Hz) of two lexical items *dark* from TIMIT with a clear *s*-like frication noise during the aspiration after the closure of [k] (highligted).

To test the effects of ϕ_1 on presence of [s] in the output, 800 annotated outputs (100 for each of the eight unique binary code) were fit to a logistic regression model. The dependent variable is presence of a *s*-like frication noise: if a transcribed output contains an *s*, *z*, or *f*, the output is coded as success. The independent predictors in the model are the three features without interactions: ϕ_1, ϕ_2, and ϕ_3, each with two levels (0 and 2). Figure 10 features estimates of the regression model. While all three features are significant predictors, the effect appears to be most prominent for ϕ_1.

It is possible that the Generator network in the fiwGAN architecture uses feature ϕ_1 to encode presence of segment [s] in the output. This distribution can also be due to chance. Further work is needed to test whether presence of phonetic/phonological elements in the output can be encoded with individual features. Two facts from the generated data, however, suggest that the Generator in the fiwGAN architecture associates ϕ_1 with presence of [s].

First, while *ask*, *greasy*, and *suit* all have $\phi_1 = 2$ in common, the fourth unique featural code with $\phi_1 = 2$ ([2, 2, 2]) is associated with *dark*. Spectral analysis of lexical item *dark* in the training data reveals that aspiration of [k] in *dark* is in the training data from TIMIT frequently realized precisely as an alveolar fricative [s] (likely due to contextual influences). Approximately 27% data points for *dark* in the training data from TIMIT contain a [s]-like frication noise during the aspiration period of [k].[14] Figure 9 gives two such examples from TIMIT of *dark* with a clear frication noise characteristic of an [s] sound after three aspiration noise of [k]. In other words, 3 lexical items in the training data contain an [s] as part of their phonemic representation and therefore feature it consistently. The Generator outputs data such that a single feature ($\phi_1 = 2$) is common to all three items. An additional item often involves a *s*-like element and the network uses the same value ($\phi_1 = 2$) for its unique code ([2, 2, 2]). There is approximately a 8.6% chance this distribution is random (of 70 possible featural code assignment for eight items, four of which contain some phonetic feature such as [s], six or 8.6% combinations contain the same value in one feature).

As already mentioned, the network outputs mostly *dark* for the featural code [2, 2, 2], but a substantial portion of outputs also deviate from *dark*. A closer look at the structure of the innovative outputs for the [2, 2, 2] code reveals that a substantial proportion of them (35) contain an [s]. As a comparison, other unique codes with ϕ_1 set at the opposite value 0 ([0, 0, 0], [0, 0, 2], [0, 2, 0], [0, 2, 2]) output 43, 41, 1, and 0 outputs containing an *s*, *z*, or *f*. In other words, for two

[14]This estimate is based on acoustic analysis of the first 100 training data points from the TIMIT database.

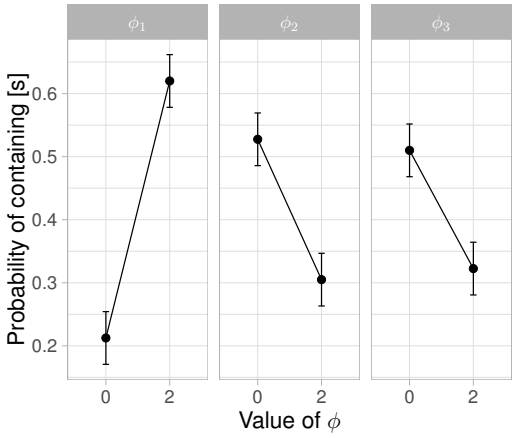

Figure 10: Fitted values with 95% CIs of a logistic regression model with presence of [s] in the transcribed outputs as the dependent variable and the three features ϕ_1, ϕ_2, and ϕ_3 as predictors.

unique codes given $\phi_1 = 0$, the network generates 0 or 1 outputs containing an *s*-like segment. For the two other codes, the network generates outputs with a similar rate of *s*-containing sequences as [2, 2, 2] (*dark*). However, the motivation for an *s*-containing output in [0, 2, 2] is clear: *year* is in three training data points actually realized as [ʃɪɹ] (*shear*). The [0, 0, 0] does not have a distinct underlying lexical item, so the high proportion of outputs with [s] is not unexpected.

The second piece of evidence suggesting that ($\phi_1 = 2$) represents presence of [s] in the output are innovative outputs that violate the training data distribution. The majority of *s*-containing outputs when $\phi_1 = 0$ are non-innovative sequences that correspond to lexical items from the training data. The most notable feature of the *s*-containing outputs for [2, 2, 2] (*dark*), on the other hand, is their innovative nature. Sometimes, these outputs can indeed be transcribed as *suit*, but in some cases the Generator outputs an innovative sequence that violates training data, but is linguistically interpretable. In fact, some of the outputs with [2, 2, 2] are directly interpretable as adding an [s] to the underlying form *dark*. Two innovative sequences that can be reliably transcribed as *start* ['stɑɹt] are given in Figure 11 and additional one transcribed as *sart* ['sɑɹt] in Figure 11. The network is never trained on a [st] sequence, let alone on the lexical item *start*, yet the innovative output is linguistically interpretable and remarkably similar to the [st] sequence in human outputs that the network never "sees". Spectral analysis illustrates a clear period of frication noise characteristic of [s] followed by a period of silence and release burst characteristic of a stop [t]. Figure 12 provides two examples from the TIMIT database with the [st] sequence that was never part of the training data, yet illustrates how acoustically similar the innovative generated outputs in the fiwGAN architecture are to real speech data. This example constitutes one of the prime cases of high productivity of deep neural networks (for a recent survey on productivity in deep learning, see Baroni 2020).

4.5. Underlying representations

Beguš (2020) argues that the underlying value of a feature can be uncovered by manipulating a given feature well beyond the training range. For example, Beguš (2020) proposes a technique for identifying variables that correspond to phonetic/phonological features in the outputs. By setting the values of the identified features well beyond the training range, the network almost exclusively outputs the desired feature (e.g. a segment [s] in the output).

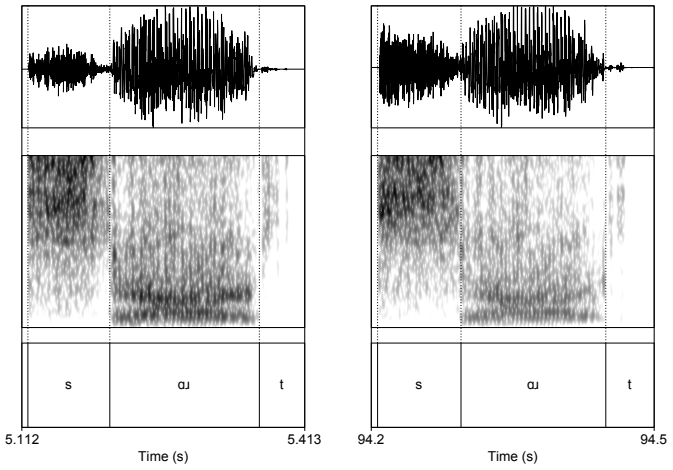

Figure 11: Waveforms and spectrograms (0-8000 Hz) of two innovative outputs for $\phi = [2, 2, 2]$ transcribed as ['sɑɹt].

This effect of the underlying value of a variable is even more prominent in the fiwGAN architecture. When the values of latent features (ϕ) are set at 0 and 2, success rates appear at approximately 50% (Figure 8). Value 2 was chosen for analysis in Section 4.3, because non-categorical outcomes yield more insights into learning. However, we can reach almost categorical accuracy when the values are set substantially higher than 2. For example, when we generate outputs with values of featural code set at 5 ([5, 5, 5]), the network generates 93/100 outputs[15] that can be reliably transcribed as *dark* and another 7 that closely resemble *dark*, but have a period of frication instead of the initial stop (*sark*). With even higher values such as 15, the Generator outputs 100/100 samples transcribed as *dark* for [15, 15, 15]. Similarly, [5, 5, 0] yields *ask* in 97/100 cases; it yields an innovative output with final [i] in three cases. At [15, 15, 0], the Generator outputs 100/100 *ask*. The success rates differ across featural codes, but value 15 triggers almost categorical outputs for most of them. [15, 0, 0] yields 93/100 *greasy* (1 unclear and 6 *ask*). For [15, 0, 15], the network outputs a [sVt] for *suit* (where V=vowel) sequence 87/100 times. In 13 examples, the frication noise does not have a pronounced s-like frication noise, but is more distributed and closer to aspiration noise of [k]. The identity of the vowel is intriguing: the formant values are not characteristic of [u] (as in *suit*), but rather of a lower more central vowel (F1 = 663 Hz, F2 = 1618 Hz, F3 = 2515 Hz for one listing). Since formant variability is high in the training data, the underlying prototypical representation likely defaults to a more central vowel.

[0, 0, 15] yields *carry* in 100/100 outputs, but in 13 of these outputs the aspiration noise of [k] is distributed with a peak in higher frequencies for an acoustic effect of [ts]. [0, 15, 0] yields 100/100 *water*. The acoustic output is reduced to include only the main acoustic properties of *water*: formant structure for [wɔ] followed by a consonantal period for [ɾ] and a very brief (sometimes missing) vocalic period (Figure 13).

The only two codes that do not yield straightforward underlying representations are [0, 0, 0] and [0, 2, 2]. It appears that the Generator is unable to strongly associate [0, 0, 0] with any lexical representation, likely due to lack of positive values in this particular code. This means that the network needs to learn underlying representations for two remaining lexical items with a single code: *like* and *year*, both likely associated with [0, 2, 2]. When set to [0, 5, 5], the Generator

[15]Counts in this sections are performed by the author only.

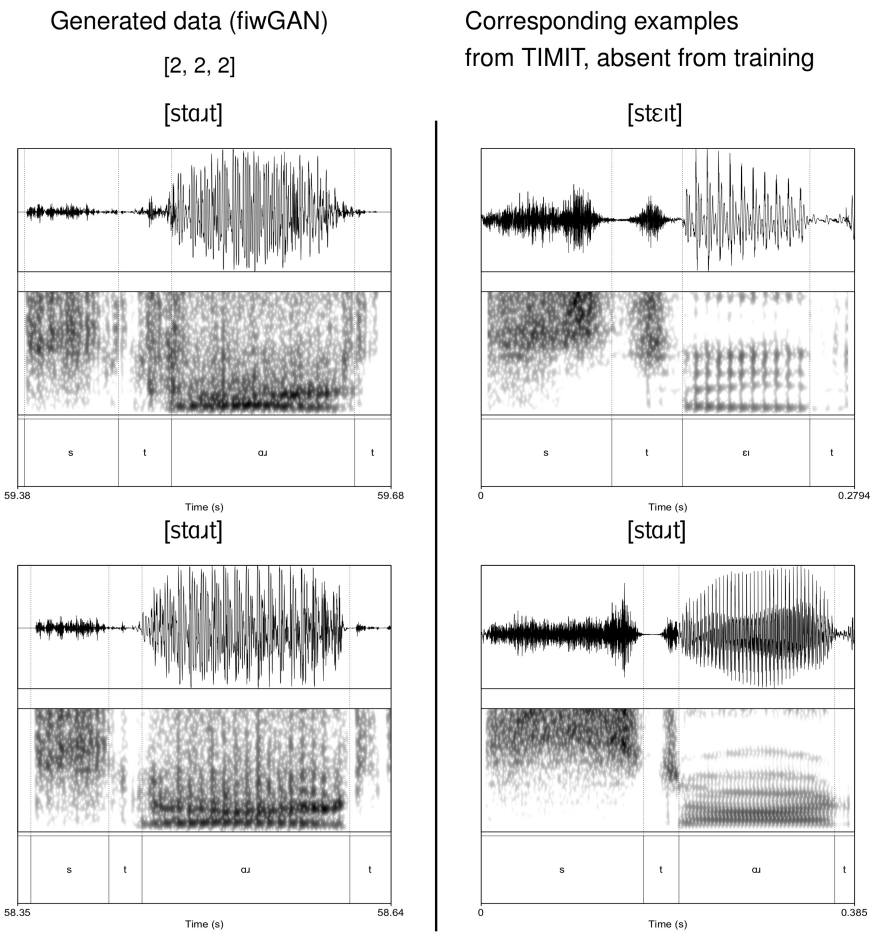

Figure 12: (left) Waveforms and spectrograms (0-8000 Hz) of two innovative outputs for $\phi = [2, 2, 2]$ transcribed as ['stɑɹt]. The ciwGAN network trained after 20026 steps thus outputs innovative sequence [st] that is absent from the training data, but is a linguistically interpretable output that can be interpreted as adding [s] to *dark*. (right) Waveforms and spectrograms (0-8000 Hz) of two lexical items from TIMIT that are *not* part of training data, but illustrate that the innovative [st] sequence in the generated data is acoustically very similar to the [st] sequence in human outputs that the network has no access to.

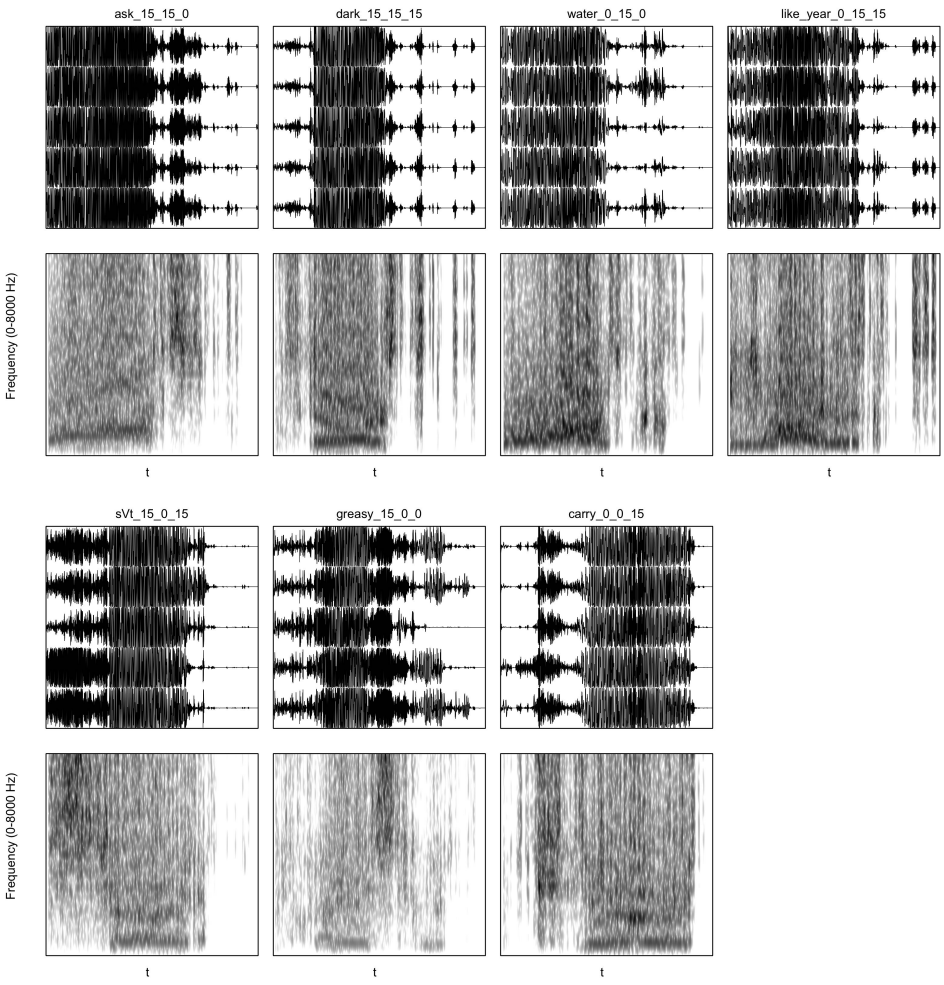

Figure 13: Waveforms of the first five generated outputs for each featural code when values are set at 15. The waveforms clearly show that the outputs feature minimal variability. Below each waveform is a spectrogram (0–8000 Hz) of the first output (the topmost waveform). All seven outputs have the exact same values for 97 random latent variables (z); they only differ in the three featural codes ϕ.

outputs both *like* and *year*,[16] but at [0, 10, 10] and [0, 15, 15] the underlying representation is an acoustic output that is difficult to characterize and is likely a blend of the two representations (acoustically closer to *like*; see Figure 13). Future analyses should thus include $\log_2(n)$ variables for $n - 1$ classes in the fiwGAN architecture.

Another interesting fact emerges when we set the featural codes to high values such as 5 or 15. The outputs at these high values are minimally variable: the outputs are almost identical despite the 97 random latent variables z being randomly sampled for each output, as illustrated by Figure 13. It appears that the network associates unique featural codes with prototypical underlying representations of lexical items. When values are lower, other random features (z) cause variation in the outputs, but the high values (such as 5 or 15) override this variation and reveal the underlying lexical representation for each featural code.

The generative test with values set well above the training range strongly suggests that the

[16]Occasionally, [0, 5, 5] also yields an output that can be characterized as *water*.

Generator associates lexical items with unique codes and likely represents them with prototypical acoustic values. The test also confirms that the assumed underlying lexical items identified with multinomial logistic regression (Figure 8) are correct.

5. Discussion and future directions

This paper proposes two architectures for unsupervised modeling of lexical learning from raw acoustic data with deep neural networks. The ability to retrieve information from acoustic data in human speech is modeled with a Generator network that learns to output data that resembles speech and, simultaneously, learns to encode unique information in its outputs. We also propose techniques for probing how deep neural networks trained on speech data learn meaningful representations.

The proposed fiwGAN and ciwGAN models are based on the Generative Adversarial Network architecture and its implementations in WaveGAN (Donahue et al., 2019), DCGAN (Radford et al., 2015), and InfoGAN (Chen et al., 2016; Rodionov, 2018). Following Beguš (2020), we model language acquisition as learning of a dependency between the latent space and generated outputs. We introduce a network that forces the Generator to output data such that information is retrievable from its acoustic outputs and propose a new structure of the latent variables that allows featural learning. Lexical learning thus emerges from the architecture: the most efficient way for the Generator network to output acoustic data such that unique information is retrievable from its data is to encode unique information in its acoustic outputs such that latent codes coincide with lexical items in the training data. The result is thus a deep convolutional neural network that takes latent codes and variables and outputs innovative data that resembles training data distributions as well as learns to associate lexical items with unique representations.

Three experiments tested lexical learning in ciwGAN and fiwGAN architectures trained on tokens of five, ten, and eight sliced lexical items in raw audio format from a highly variable database — TIMIT. The paper proposes that lexical learning can be evaluated with multinomial logistic regression on generated data. Evidence of lexical learning is present in all three experiments. It appears that the Generator learns to associate lexical items with unique latent code — categorical (as in ciwGAN) or featural (as in fiwGAN). By manipulating the values of latent codes to value 2, the networks output unique lexical items for each unique code and reach accuracy that ranges from 98% to 27% in the five-word model. To replicate the results and test learning on a higher number of lexical items, the paper presents evidence that the model learns to associate unique latent codes with lexical items in the 10-words model as well with only one exception. The paper also proposes a technique for following how the network learns representations as training progresses. We can directly observe how the network transforms an output that violates training data into an output that conforms to it by keeping the latent space constant as training progresses.

The fiwGAN architecture features, to our knowledge, a new proposal within the InfoGAN architecture: to model classification with featural codes instead of one-hot vectors. This shift yields the potential to model featural learning and higher-level classification (i.e. phonetic/phonological features and unique lexical representations) simultaneously. The paper presents evidence suggesting that the network might use some feature values to encode phonetic/phonological properties such as presence of [s]. Regression models suggest that ϕ_1 is associated with presence of [s] in the output (and simple probabilistic calculation reveals about 8.6% probability that the distribution is due to chance). The strongest evidence for simultaneous lexical and featural learning comes from innovative outputs in the fiwGAN architecture. The network trained on lexical items that lack a sequence of a fricative and a stop [st] altogether outputs an innovative sequence *start* or *sart*. These innovative outputs can be analyzed as adding a segment [s] (from *suit*) to *dark*, likely under the influence of the fact that ϕ_1 represents presence of [s].

Innovative outputs that violate training data are informative for both computational models of language acquisition as well as for our understanding of what types of dependencies the networks are able to learn. We discuss several cases of innovative outputs. Some innovations are motivated by training data distributions (e.g. *sear*) and reveal how the networks treat acoustically similar lexical items. For other innovative outputs, such as *watery*, the training data contains no apparent motivations. We also track changes from innovative to conforming outputs as training progresses.

We argue that innovative outputs are linguistically interpretable and acoustically very similar to actual speech data that is absent from the training data. For example, an innovative [st] sequence in *start* corresponds directly to human outputs with this sequence that were never part of the training data. Further comparisons of this type should yield a better understanding on how the combinatorial principle in human language can arise without language-specific parameters in a model.

The paper also discusses how internal representations in the GAN architecture can be identified and explored. We argue that by setting the latent values substantially beyond the training range (as suggested for phonological learning in Beguš 2020), the Generator almost exclusively outputs one unique lexical item per each unique featural code (with only one exception) in the fiwGAN architecture. In other words, for very high values of the featural code (ϕ), lexical learning appears to be categorical. The variability of the outputs is minimal at such high values (e.g. 15). It appears that setting the featural code to such extreme values reveals the underlying representation of each featural code. This property is highly desirable in a model of language acquisition and has the potential to reveal the underlying learned representations in the GAN architecture.

Several improvements to the model are left to future work. For example, future directions should include developing a model that does not require a pre-determined number of classes that correspond to the number of lexical items and that could parse lexical items from a continuous acoustic stream.

The proposed model of lexical learning has several further implications. Dependencies in speech data are significantly better understood than dependencies in visual data. A long scientific tradition of studying dependencies in phonetic and phonological data in human languages yields an opportunity to use linguistic data to probe the types of dependencies deep neural networks can or cannot learn. The proposed architectures allow us to probe what types of dependencies the networks can learn, how they encode unique information in the latent space, and how self-organization of retrievable information emerges in the GAN architecture. The models also have some basic implications for NLP tasks such as unsupervised text-to-speech generation: manipulating the latent variables to specific values results in the Generator outputting desired lexical items.

Acknowledgements

This research was funded by a grant to new faculty at the University of Washington. I would like to thank Sameer Arshad for slicing data from the TIMIT database and Ella Deaton for annotating data. All mistakes are my own.

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

Arjovsky, M., Chintala, S., Bottou, L., 2017. Wasserstein generative adversarial networks. In: International Conference on Machine Learning. pp. 214–223.

Arnold, D., Tomaschek, F., Sering, K., Lopez, F., Baayen, R. H., 04 2017. Words from spontaneous conversational speech can be recognized with human-like accuracy by an error-driven learning algorithm that discriminates between meanings straight from smart acoustic features, bypassing the phoneme as recognition unit. PLOS ONE 12 (4), 1–16.
URL https://doi.org/10.1371/journal.pone.0174623

Baayen, R. H., Chuang, Y.-Y., Shafaei-Bajestan, E., Blevins, J. P., 2019. The discriminative lexicon: A unified computational model for the lexicon and lexical processing in comprehension and production grounded not in (de) composition but in linear discriminative learning. Complexity 2019.

Baroni, M., 2020. Linguistic generalization and compositionality in modern artificial neural networks. Philosophical Transactions of the Royal Society B: Biological Sciences 375 (1791), 20190307.
URL https://royalsocietypublishing.org/doi/abs/10.1098/rstb.2019.0307

Barry, S. M., Kim, Y. E., 2019. InfoWaveGAN: Informative latent spaces for waveform generation, poster at the North East Music Information Special Interest Group.
URL http://nemisig2019.nemisig.org/images/kimSlides.pdf

Beguš, G., 2020. Generative adversarial phonology: Modeling unsupervised phonetic and phonological learning with neural networks. Accepted at Frontiers in Artificial Intelligence.
URL https://www.frontiersin.org/articles/10.3389/frai.2020.00044/abstract

Boersma, P., Weenink, D., 2015. Praat: doing phonetics by computer [computer program]. version 5.4.06. Retrieved 21 February 2015 from http://www.praat.org/.

Bond, Z. S., Wilson, H. F., 1980. /s/ plus stop clusters in children's speech. Phonetica 37 (3), 149–158.
URL https://www.karger.com/DOI/10.1159/000259988

Brownlee, J., 2019. Generative Adversarial Networks with Python: Deep Learning Generative Models for Image Synthesis and Image Translation. Machine Learning Mastery.

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-ma
pdf

Chuang, Y.-Y., Vollmer, M. L., Shafaei-Bajestan, E., Gahl, S., Hendrix, P., Baayen, R. H., 2020. The processing of pseudoword form and meaning in production and comprehension: A computational modeling approach using linear discriminative learning. Behavior Research Methods.
URL https://doi.org/10.3758/s13428-020-01356-w

Clements, G. N., 1985. The geometry of phonological features. Phonology Yearbook 2 (1), 225252.

Donahue, C., McAuley, J., Puckette, M., 2019. Adversarial audio synthesis. In: ICLR. github.com/chrisdonahue/wavegan.

Eloff, R., Nortje, A., van Niekerk, B., Govender, A., Nortje, L., Pretorius, A., Biljon, E., van der Westhuizen, E., Staden, L., Kamper, H., 09 2019. Unsupervised acoustic unit discovery for speech synthesis using discrete latent-variable neural networks. In: Proc. Interspeech 2019. pp. 1103–1107.

Elsner, M., Goldwater, S., Feldman, N., Wood, F., Oct. 2013. A joint learning model of word segmentation, lexical acquisition, and phonetic variability. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Seattle, Washington, USA, pp. 42–54.
URL https://www.aclweb.org/anthology/D13-1005

Feldman, N. H., Griffiths, T. L., Goldwater, S., Morgan, J. L., 2013. A role for the developing lexicon in phonetic category acquisition. Psychological review 120 (4), 751.

Feldman, N. H., Griffiths, T. L., Morgan, J. L., 2009. Learning phonetic categories by learning a lexicon. In: Scott, J., Waugtal, D. (Eds.), Proceedings of the 31st Annual Conference of the Cognitive Science Society. pp. 2208–2213.

Garofolo, J. S., Lamel, L., M Fisher, W., Fiscus, J., S. Pallett, D., L. Dahlgren, N., Zue, V., 11 1993. Timit acoustic-phonetic continuous speech corpus. Linguistic Data Consortium.

Goldwater, S., Griffiths, T. L., Johnson, M., 2009. A bayesian framework for word segmentation: Exploring the effects of context. Cognition 112 (1), 21 – 54.
URL http://www.sciencedirect.com/science/article/pii/S0010027709000675

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.

Heymann, J., Walter, O., Haeb-Umbach, R., Raj, B., 2013. Unsupervised word segmentation from noisy input. In: 2013 IEEE Workshop on Automatic Speech Recognition and Understanding. pp. 458–463.

Hockett, C. F., 1959. Animal "languages" and human language. Human Biology 31 (1), 32–39.
URL http://www.jstor.org/stable/41449227

Kamper, H., Jansen, A., Goldwater, S., 2017. A segmental framework for fully-unsupervised large-vocabulary speech recognition. Computer Speech & Language 46, 154 – 174.
URL http://www.sciencedirect.com/science/article/pii/S0885230816301905

Kuhl, P. K., 2019/06/27 2010. Brain mechanisms in early language acquisition. Neuron 67 (5), 713–727.
URL https://doi.org/10.1016/j.neuron.2010.08.038

Lee, C.-y., Glass, J., Jul. 2012. A nonparametric Bayesian approach to acoustic model discovery. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Jeju Island, Korea, pp. 40–49.
URL https://www.aclweb.org/anthology/P12-1005

Lee, C.-y., O'Donnell, T. J., Glass, J., 2015. Unsupervised lexicon discovery from acoustic input. Transactions of the Association for Computational Linguistics 3, 389–403.
URL https://www.aclweb.org/anthology/Q15-1028

Mesgarani, N., Cheung, C., Johnson, K., Chang, E. F., 2014. Phonetic feature encoding in human superior temporal gyrus. Science 343 (6174), 1006–1010.
URL https://science.sciencemag.org/content/343/6174/1006

Piantadosi, S. T., Fedorenko, E., 04 2017. Infinitely productive language can arise from chance under communicative pressure. Journal of Language Evolution 2 (2), 141–147.
URL https://doi.org/10.1093/jole/lzw013

R Core Team, 2018. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria.
URL https://www.R-project.org/

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

Räsänen, O., 2012. Computational modeling of phonetic and lexical learning in early language acquisition: Existing models and future directions. Speech Communication 54 (9), 975 – 997.
URL http://www.sciencedirect.com/science/article/pii/S0167639312000672

Räsänen, O., Nagamine, T., Mesgarani, N., 08 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.

Saffran, J. R., Aslin, R. N., Newport, E. L., 1996. Statistical learning by 8-month-old infants. Science 274 (5294), 1926–1928.
URL https://science.sciencemag.org/content/274/5294/1926

Saffran, J. R., Werker, J. F., Werner, L. A., 2007. The Infant's Auditory World: Hearing, Speech, and the Beginnings of Language. American Cancer Society, Ch. 2.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9780470147658.chpsy0202

Shafaei-Bajestan, E., Baayen, R. H., 2018. Wide learning for auditory comprehension. In: Proc. Interspeech 2018. pp. 966–970.
URL http://dx.doi.org/10.21437/Interspeech.2018-2420

Shain, C., Elsner, M., Jun. 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

Venables, W. N., Ripley, B. D., 2002. Modern Applied Statistics with S, 4th Edition. Springer, New York, iSBN 0-387-95457-0.
URL http://www.stats.ox.ac.uk/pub/MASS4