Biologically inspired methods in speech recognition and synthesis: closing the loop

by

Trevor Bekolay

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Computer Science

Waterloo, Ontario, Canada, 2016

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Current state-of-the-art approaches to computational speech recognition and synthesis are based on statistical analyses of extremely large data sets. It is currently unknown how these methods relate to the methods that the human brain uses to perceive and produce speech. In this thesis, I present a conceptual model, *Sermo*, which describes some of the computations that the human brain uses to perceive and produce speech. I then implement three large-scale brain models that accomplish tasks theorized to be required by Sermo, drawing upon techniques in automatic speech recognition, articulatory speech synthesis, and computational neuroscience.

The first model extracts features from an audio signal by performing a frequency decomposition with an auditory periphery model, then decorrelating the information in that power spectrum with methods commonly used in audio and image compression. I show that the features produced by this model implemented with biologically plausible spiking neurons can be used to classify phones in pre-segmented speech with significantly better accuracy than the features typically used in automatic speech recognition systems. Additionally, I show that this model can be used to compare auditory periphery models in terms of their ability to support phone classification of pre-segmented speech.

The second model uses a symbol-like neural representation of a sequence of syllables to generate a trajectory of premotor commands that can be used to control an articulatory synthesizer. I show that the model can produce trajectories up to several seconds in length from a static syllable sequence representation that result in intelligible synthesized speech. The trajectories reflect the high temporal variability of human speech, and smoothly transition between successive syllables, even in rapid utterances.

The third model classifies syllables from a trajectory of premotor commands. I show that the model is able to classify syllables online despite high temporal variability, and can produce the same syllable representations used by the second model. These two models can be connected in future work in order to implement a closed-loop sensorimotor speech system.

Unlike current computational approaches, all three of these models are implemented with biologically plausible spiking neurons, which can be simulated with neuromorphic hardware, and can interface naturally with artificial cochleas. All models are shown to scale to the level of adult human vocabularies in terms of the neural resources required, though limitations on their performance as a result of scaling will be discussed.

Acknowledgements

The CNRG lab at the University of Waterloo has provided a rich and life-changing environment for learning and producing world-class research. In particular, I would like to thank the PhD students whose journeys have overlapped with mine, and who will soon be my coworkers at ABR: Travis DeWolf, Daniel Rasmussen, Xuan Choo, and Eric Hunsberger. Not only have they provided innumerable fruitful modeling discussions, they have been great friends for the past six years. I would also like to thank Terry Stewart, Jan Gosmann, Peter Blouw, Jen Eliasmith, Gillian Martin, and Julie DeWolf, whose weighted sum of technical and personal support is remarkably high, though the weights may vary.

I cannot overstate how grateful I am to Chris Eliasmith, who has not only created the amazing environment at the CNRG, but is the best supervisor a grad student could hope for. He has been an incredible role model for research, a friend, a bandmate, a hockey linemate, and I look forward to growing ABR together.

At the end of 2013, Bernd J. Kröger emailed Chris about whether Nengo could be used for speech processing, which I had just started thinking about seriously. I would like to thank Bernd for being a truly wonderful collaborator, providing a multitude of ideas and important linguistic background that have shaped the ideas in this thesis, as well as being a kind person who is a joy to work with. In particular, I am grateful for his comments on an early version of some chapters of this thesis, which are now much improved. I would also like to thank Peter Birkholz for his help with VocalTractLab, and Jeff Orchard, Dan Brown, Mathias Schulze, and Louis Goldstein for serving on my thesis committee and providing thoughtful questions, intriguing comments, and helpful corrections.

My family (Cathy, David and Jason Bekolay) have always been supportive of my possibly terrible decision to go to grad school, for which I am grateful. Special thanks also for providing a basement in which to finish this thesis. Thanks also to a new member of my family, Alexandra Irvine, for help with aesthetics.

I would also like to give a shoutout to the developers of the tools that I use every day to get work done. I have no idea where I would be now Without Emacs, LaTeX, Python, NumPy, zsh, git, and Github. Also, a shoutout to Arin, Dan, Brian, Jake, and Amir who provided necessary downtime during the writing process.

Finally, I was lucky enough to meet Emily Irvine shortly after starting my PhD. Now, finishing up my PhD, we have been together for four years and married for one, and they have been the most fulfilling years of my life thanks in large part to Emily. I cannot thank her enough for supporting me through this process, and I am looking forward to our life together sans thesis!

Dedication

For Emily, who is the best.

Table of Contents

List of Tables

List of Figures

Typographical Conventions

This thesis uses the International Phonetic Alphabet (IPA) to denote utterance pronunciations. As is common practice in linguistics, we will use square brackets (e.g., $[\cdot]$) to denote phonetic transcriptions (i.e., the actual phones uttered) and slashes (e.g., $/\cdot/$) to denote phonemic transcriptions (i.e., the phonemes of interest). As little attention is paid to prosodic elements like stress, phone and phoneme strings will be presented without stress markers.

Chapter 1

Introduction

Speech is arguably the most important medium through which humans communicate. As the role of human-computer interaction in modern society increases, so does the need for computers to recognize and synthesize speech. Currently, state-of-the-art approaches to computational speech recognition and synthesis are based on statistical analyses of extremely large data sets. While our understanding of the neurobiology of speech has advanced in recent years, there currently exist no models perceiving or producing speech using neurobiological parts, in particular spiking neurons. Spiking neurons are the basic building blocks of the brain. Each neuron takes input from many other neurons, and with enough input, experiences a spike in membrane voltage, which results in neurotransmitter delivery to downstream neurons. The activities of many billions of spiking neurons with trillions of interconnections are responsible for producing and processing complex behaviors including speech.

Recent advances in large-scale neural modeling enable spiking neuron models of speech. The Neural Engineering Framework (NEF; Eliasmith and Anderson (2004)) provides a framework to implement dynamical systems with networks of spiking neurons. The Semantic Pointer Architecture (SPA; Eliasmith (2013)) provides a method to connect these dynamical systems to symbol-like representations. In terms of speech, dynamical systems enable processing of temporally varying sensory inputs to effect motor outputs. The symbol-like representations in SPA allow us to interpret the states of these dynamical systems as linguistically relevant concepts, which connects the low-level sensorimotor aspects of speech with the high-level linguistic aspects of speech. Taken together, the NEF and SPA provide the essential tools required to apply biological methods to the fields of speech recognition and synthesis, which are currently dominated by computational statistics.

In addition to the application of biological methods, we aim to provide a framework for

closed-loop speech systems which perform both recognition and synthesis with common representations. Currently, state-of-the-art approaches to speech recognition and synthesis are independent; that is, speech recognition and synthesis are seen as independent problems, and are solved by independent systems. In humans, however, speech perception and production overlaps and interacts in a closed-loop manner. When learning to speak, we use the perception of our own voice to improve future vocalizations. While the role of such a closed-loop system diminishes over time, we still regularly monitor our own speech and can adapt when our speech perception or production are perturbed through illness or other means.[1]

The long-term goal of the line of research described in this thesis is a closed-loop speech system that uses biological methods implemented in computers to recognize and synthesize speech. Such a system could produce its own training data, and therefore would require far less hand-labeled real-world data than current purely statistical systems. It would also enable interrogation of how the system recognizes and synthesizes speech, and enable synergistic interactions between theoretical model developers and experimental researchers in linguistics and neuroscience. As the model develops and makes predictions about the computations required for speech and how they might be implemented, linguists and neuroscientists can test those predictions and update the model accordingly.

In order to achieve the long-term goal of a closed-loop system implemented with biologically realistic neurons that can learn from itself, we must first envision what a large-scale closed-loop system would look like, and detail the computations required by that system. We must also verify that neural implementations of these computations are possible with the available tools. In this thesis, I address these two concerns by proposing a closed-loop speech system called Sermo (Speech execution and recognition model organism), and constructing simulated neural implementations of three subsystems of Sermo. In the conceptual Sermo model, I have synthesized literature in linguistics, psychoacoustics, neuroscience, automatic speech recognition, and articulatory synthesis to present a closed-loop system with well-defined computations. In the three subsystems I have implemented, I show that the NEF and SPA can be successfully applied to the computations required by the Sermo model, providing the first steps toward a neurally implemented closed-loop speech system.

The remainder of the thesis is organized as follows. In Chapter 2, I review the relevant background informing the conceptual Sermo model. In Chapter 3, I present Sermo, which provides context for the three neural models presented in the subsequent chapters. In Chapter 4, I review existing approaches to solving the problems proposed by Sermo and addressed by the

[1] Here, and for the remainder of this thesis, we will refer to the sensory aspect of speech as "speech perception" when referring to human speech understanding, and as "speech recognition" when referring to computer speech understanding. Similarly, the motor aspect of speech will be referred to as "speech production" when referring to humans speaking, and as "speech synthesis" when referring to computers producing speech.

three neural models. In Chapter 5, I provide details on the mathematical techniques used by the neural models, and used to construct the neural models. In Chapter 6, I describe the three neural models. In Chapter 7, I propose metrics through which to evaluate those models, and present the results of collecting those metrics while varying parameters of the neural models. In Chapter 8 I discuss the results, summarize the contributions and predictions of the models, and discuss avenues for future work. Finally, I conclude in Chapter 9.

Note that this chapter organization differs from a prototypical thesis because I make both conceptual contributions in the Sermo model and modeling contributions in the three subparts of Sermo for which I have made neural models. As such, the background chapter is designed to provide background for the conceptual model, which in turn provides context for the neural models. The previous work chapter provides background for the neural models, which are then explained fully in the methods and implementation chapters.

Chapter 2

Background

This thesis draws upon what would traditionally be thought of a disparate fields: phonetics and phonology from linguistics, psychoacoustics from psychology, knowledge representation and artificial neural networks from computer science, spiking neural networks from computational neuroscience, and dynamical systems and control theory from engineering. For clarity, the background presented in this chapter is limited to subjects specifically related to speech. I assume basic knowledge in computer science or engineering, but will review the specific techniques used in this thesis in Chapter 5.

2.1 Speech perception & recognition

On the sensory side of speech, we aim to apply aspects of speech perception to automatic speech recognition (ASR) in order to improve the performance of ASR systems, and make progress toward the long-term goal of a closed-loop computational speech system that recognizes and synthesizes speech in a single interconnected network. In this section, we review fundamental concepts and literature in basic ear physiology, psychoacoustics, auditory neurobiology, and automatic speech recognition.

2.1.1 Ear physiology

The human ear transduces fluctuations in air pressure level to neural signals that we interpret as sounds. It does this by mechanically separating the air pressure level fluctuations into

instantaneous frequency components, which forms the basic representation that the brain receives. Figure 2.1 illustrates the major structures involved in sound transduction.

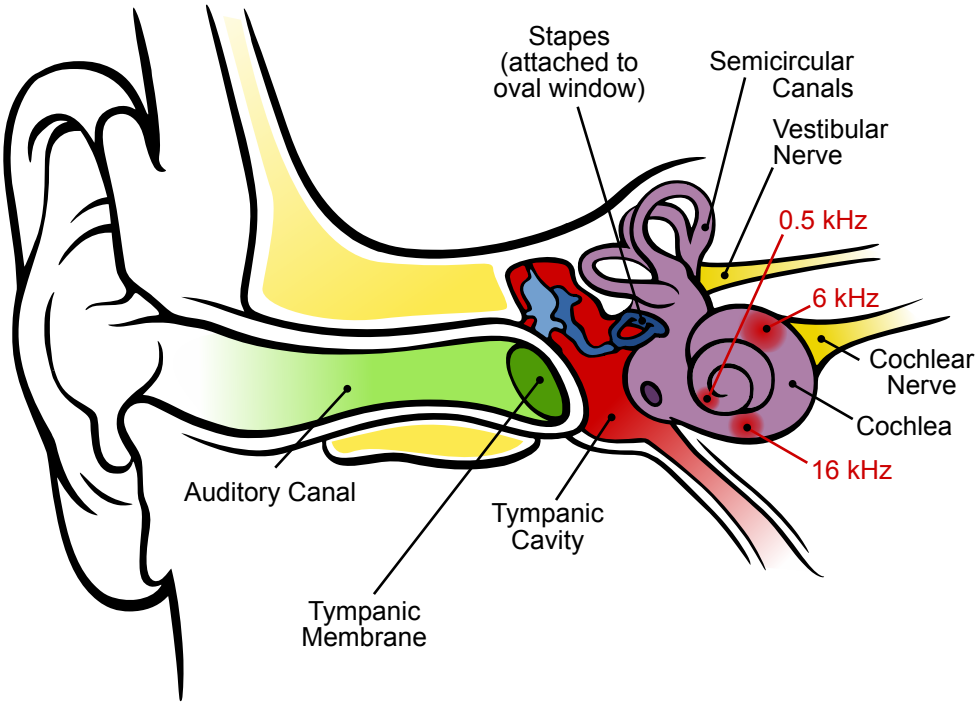


Figure 2.1: Overview of the human ear. Fluctuations in air pressure travel down the auditory canal and cause the tympanic membrane to vibrate. These vibrations result in a tiny bone called the stapes rapidly moving in and out of the oval window of the cochlea. The disturbances of the liquid in the cochlea deflect the basilar membrane inside, with high frequency disturbances causing deflections at the base of the cochlea, and low frequency disturbances causing deflections at the apex of the cochlea. Electrical activity in the form of action potentials are produced in the cochlea, and enter the brain through the cochlear nerve. Adapted from Chittka and Brockmann (2005).

The outer ear (pinna) funnels air into the ear canal, and modifies the air pressure wave such that the directionality of sound can be determined. In the middle ear, air pressure fluctuations cause the eardrum (tympanic membrane) to vibrate; these minuscule vibrations cause three tiny bones, the malleus, incus, and stapes to move. The stapes sits on the oval window of the cochlea, such that when the eardrum vibrates, the stapes moves in and out of the oval window, causing disturbances in the liquid in the cochlea (see Figure 2.2). Importantly for modeling,

the external and middle ear also attenuate low frequency sounds and amplify mid frequency sounds (2–7 kHz) (Rosowski, 1996; Ballachanda, 1997).

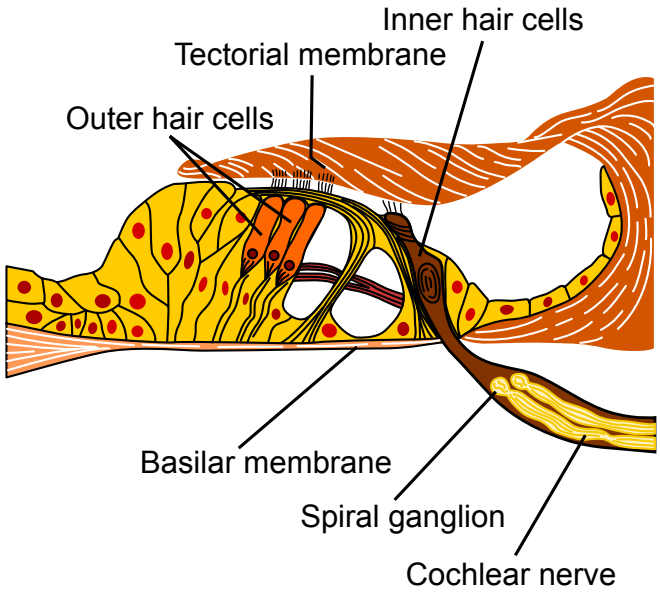


Figure 2.2: Cross-section of the cochlea. As the basilar membrane deflects due to the motion of the stapes in and out of the oval window of the cochlea, inner hair cells are pushed against the tectorial membrane. When deflected, the stereocilia on the top of the hair cells allow positive charged ions to enter the cell. The change in inner hair cell voltage causes action potentials to fire in spiral ganglion cells, which travel down the cochlear (auditory) nerve. Outer hair cells are attached to the tectorial membrane, and act as a dynamic amplifier (see text for more details). Adapted from Kandel et al. (2000).

The inner ear transduces vibrations of the liquid in the cochlea to electrical signals transmitted to the brain. As liquid in the cochlea moves past the basilar membrane, it causes it to deflect based on the width and thickness of the membrane at that point in the cochlea. The basilar membrane is shaped such that the base of the membrane (near the stapes) is deflected when incoming vibrations have power at high frequencies, up to 20,000 Hz. The membrane becomes sensitive to lower and lower frequencies going further down the membrane to the apex in the center of the cochlea, which is deflected when incoming vibrations have frequency as low as 20 Hz.

The basilar membrane's surface is covered with hair cells that transduce the deflections of the membrane to electric current. Stereocilia on the top of inner hair cells rests on or near

the tectorial membrane (see Figure 2.2). When the basilar membrane deflects, the stereocilia's orientation relative to the tectorial membrane changes, which mechanically opens receptors on top of the stereocilia, allowing positive charged ions (mostly potassium and calcium) to enter the hair cell. Inner hair cells synapse with spiral ganglion cells, which accumulate the continuously changing voltage of the inner hair cells and send action potentials down the auditory nerve conveying how much power is present at the current moment at the frequency characteristic of that section of the basilar membrane. Outer hair cells play a more nuanced role in transduction. Their activity is tuned more broadly in both space and time, and seem to act as a dynamic amplifier, amplifying quiet sounds and attenuating loud sounds (Dallos, 1992). This dynamic amplification allows the human ear to have a wide dynamic range, able to safely hear sounds from 0–130 dB, which represents 13 orders of magnitude of absolute air pressure (see Figure 2.3).

2.1.2 Psychoacoustics

Psychoacoustics provides a quantitative account of how the human auditory system responds to incoming air pressure levels. While the human ear can be thought of as a spectrum analyzer for air pressure waves, there are important differences between a straightforward Fourier analysis and how the ear and early auditory brain regions respond to air pressure waves. The psychoacoustical effects discussed in this chapter are incorporated to differing degrees in the auditory periphery models described in Section 5.1.

Spatial psychoacoustical effects

Sound pressure levels can be objectively measured and expressed in terms of the logarithmic decibel (dB) scale. Human perception of loudness can help determine the transfer functions at different parts of the auditory system. Early studies investigated subjective loudness in response to pure tones, as tones of equal sound pressure level are perceived as louder for higher frequencies. The exact relationship between sound pressure level and loudness was originally proposed by Fletcher and Munson (1933), and further standardized by the international standards organization (ISO-226:2003); see Figure 2.3. In general, perceived loudness, as expressed in the phon scale, increases for higher frequency sounds up to around 7000 Hz.

Another scale, the sone, quantifies the relative differences to pure tones played with different sound pressure levels. Specifically, the sone scale is linear with perceived loudness; a sound played with twice the sone value should be perceived as twice as loud. The sone for a

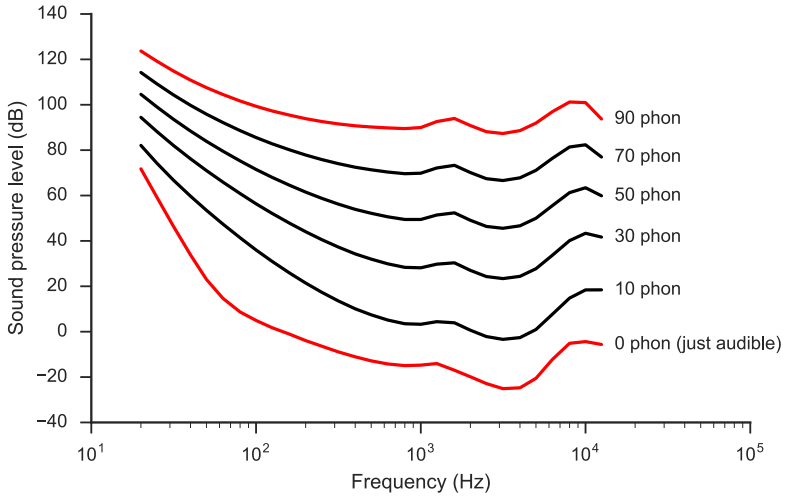


Figure 2.3: Equal loudness curves. Each line denotes a contour of equal subjective loudness for a loudness level given in units of phon (determined empirically through many studies, beginning with Fletcher and Munson 1933). Generally, low frequency sounds must have higher sound pressure levels to reach the same loudness as sounds with lower pressure at higher frequencies.

particular frequency is

$$\text{sone} \approx 2^{\frac{\text{phon}-40}{10}},$$

above signals of 40 phon or louder.

The difference in perceived loudness across the frequency spectrum, along with the organization of the inner ear, suggests that a better model of the ear (instead of as a spectrum analyzer) is as a bank of narrowband filters. Support for this model was provided in Zwicker et al. (1957), who showed that when two tones have frequencies that are close enough, their perceived loudness sums, suggesting that the two tones are within the bandwidth of one of the auditory system's filters. Moore and Glasberg (1983) summarized other attempts to determine the bandwidth of auditory filters in a simple equation describing the equivalent rectangular bandwidth (ERB) of auditory filters:

$$\text{ERB} = 6.23f^2 + 93.39f + 28.52,$$

where f is the center frequency of the auditory filter of interest. See Figure 2.4 for a visualization of this measure for frequencies of relevance for the human auditory system. Auditory

filters are not rectangular, but for simplification this equation treats them as such; despite this simplification, the ERB measure has been productive and is still used to determine the bandwidths of filters in auditory periphery models.

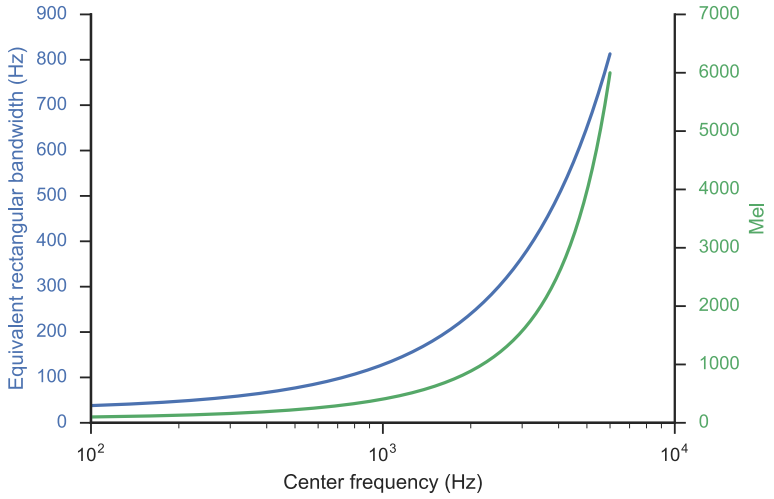


Figure 2.4: Two scales based on psychoacoustic experiments quantifying perception of sounds of different frequencies. The equivalent rectangular bandwidth scale (ERB; Moore and Glasberg, 1983), plotted in blue, summarizes the bandwidth of auditory filters at different frequencies. The Mel scale (Stevens et al., 1937), plotted in green, summarizes the perceived pitch for the given frequency.

Like volume, humans perceive the frequency of incoming sounds in subjective ways. The subjective perception of frequency is referred to as pitch. Stevens et al. (1937) quantified the relationship between frequency and pitch with a simple equation,

$$\text{Mel} = 2595 \log_{10}\left(1 + \frac{f}{700}\right),$$

resulting in the Mel scale (see Figure 2.4). As with volume, humans are adept at noticing when two tones differ in pitch; however, absolute perceptions of pitch have far less granularity.

Temporal psychoacoustical effects

To this point, we have discussed spatial psychoacoustical effects, which depend on the sound pressure level at a moment in time. There are also several temporal effects which are important

for speech. The first is that the length of a sound affects its perceived volume. Specifically, short sounds are harder to hear (see Figure 2.5). As a result, humans find consonant sounds more difficult to recognize in situations with background noise than vowel sounds, as consonants are shorter than vowels (Everest et al., 2001, Chapter 3). There is evidence that sounds with constant volume are perceived as being monotonically louder up to approximately 200 ms; this finding suggests that at some point in the auditory system, the power for a given characteristic frequency is integrated over a time window of approximately 200 ms (Kollmeier et al., 2008, p.64).

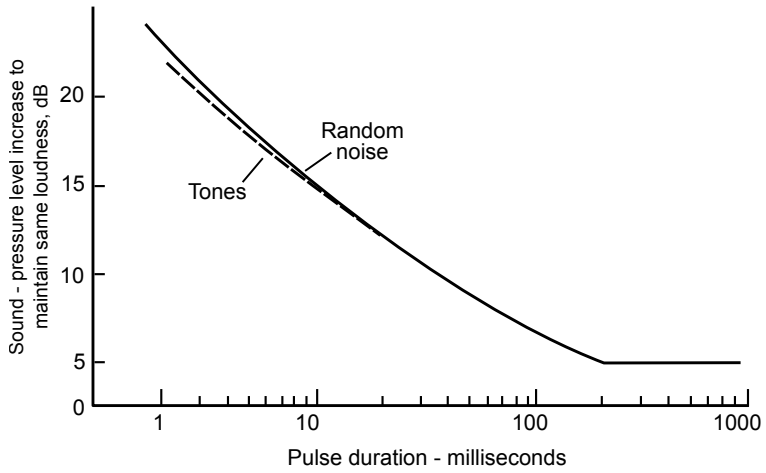


Figure 2.5: When audio samples are played for shorter durations, they are perceived as quieter, for both random noise and pure tone samples. Perceived loudness can be equalized by raising the sound pressure level of the audio pulse; the relationship between how much the sound pressure level must be raised and the duration of the audio sample is plotted. Adapted from Everest et al. (2001, p.60).

Another temporal effect is forward and backward masking. When two sounds are made in quick succession, the louder sound of the two can "mask" the quieter sound, rendering it inaudible, even though it would be audible were it made in isolation. When the masked (quieter) sound precedes the masker (louder sound), it is called backward masking; the reverse is called forward masking. Spatial masking also occurs when two sounds differ in loudness, and are within the bandwidth of an auditory filter.

Our perception of echoes can also tell us something about how the auditory system responds to temporally transformed sounds. We perceive all sounds arriving within a certain time window as occurring at essentially the same time; for example, in an enclosed room, we

hear our own speech as we voice it, and shortly thereafter when it is reflected back to us off of the walls. Yet, we perceive this as a single utterance, a phenomenon known as auditory fusion. Haas (1972) found that auditory fusion occurs best when similar sounds are separated by roughly 20 to 30 milliseconds. When similar sounds are separated by 50 to 80 milliseconds, we perceive them as discrete echoes.

Together, these temporal psychoacoustical effects suggest that the incoming frequency information from the filters implemented by the auditory periphery are further temporally filtered by auditory regions of the brain.

2.1.3 Linguistics

Following psychoacoustics, linguistics provides the next level of insight into how we perceive speech. While a full treatment of linguistics is impossible in this short space, we will briefly introduce key concepts in phonology, phonetics, morphology, and semantics in order to explain the speech perception hierarchy summarized in Figure 2.6.[1][2]

Phones and phonemes

The smallest linguistically relevant unit is the phoneme, which conceptually summarizes the linguistically relevant aspects of a phone, which is a short speech sound on the order of tens of milliseconds. Phonemes are noted by their ability to change the meaning of some word when swapped in speech; for example, the only difference between the word "bad" and "mad" is the consonant phoneme at the beginning (/b/ or /m/). Phones, on the other hand, describe the actual sounds produced by a speaker. More than one phone can map to a phoneme; for example, the English word "pin" can be voiced as [pɪn] or [$\mathrm{p^h}$ɪn] and will be recognized by all English speakers as /pɪn/ because the [$\mathrm{p^h}$] phone is used interchangeably with [p] in English (i.e., [$\mathrm{p^h}$] and [p] are allophones).

Even though different individuals voice each phone differently, the important quality is that the particular sound is perceived as a particular phoneme by a listener in the context of an utterance. A helpful analogy can be made between phones in speech and alphabetical letters in writing. With no knowledge of English, seeing the sentence, "A bird has a wing," one might think that "A" and "a" are different letters. However, with enough examples, one could surmise

[1] When not explicitly cited, the information in Sections 2.1.3 and 2.2.2 is basic linguistics that would be covered in an introductory undergraduate level class or textbook (Roach (2010) is one general reference).

[2] The International Phonetic Alphabet (IPA) will be used this and subsequent sections. See Figure 2.7 and Table 2.1 for more details.

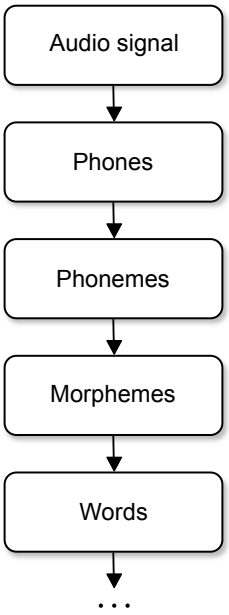


Figure 2.6: Hierarchy of linguistic concepts in speech perception. A continuous audio signal contains a string of discrete phones, which are processed in a linguistic context to yield a string of phonemes. A morpheme is composed of one or more phonemes, and a word is composed of one or more morphemes. While there are further levels in the speech perception hierarchy, we limit our discussion to the levels up to words.

that in every situation where "A" is used, it appears at the start of a sequence of letters, and it could have instead been replaced by "a" had it not appeared at the start of the sequence. Therefore, "A" and "a" represent the same underlying "letter." Similarly, despite individual differences in the pitch, speed, volume, and quality of how one realizes a particular phoneme, that realization is still considered the same if it plays the same role in a linguistically relevant sequence of phonemes. Phonemes, therefore, are theoretical constructs that are useful for describing a language, while phones are the actual sounds produced by speakers of a language.

Most phones can be categorized as either a vowel phone or a consonant phone. Vowels are longer sounds made when the vocal tract is mostly open. Consonants are shorter sounds made when some part of the vocal tract is constricted or transiently closed. In almost all languages, there are more consonant phonemes than there are vowel phonemes, though pronunciation varies significantly between dialects, and transcribing the full set of phonemes in a language is not a purely objective exercise.

Vowel phones occur when air is freely moving through the open vocal tract. The shape of

the vocal tract determines the quality of the sound. Three factors influence vowel vocal tract shape: openness, backness, and roundedness. Openness refers to the general position of the jaw (open or closed) and tongue (low or high); backness refers to the position of the tongue relative to the back of the mouth; and roundedness refers to whether the lips are rounded. Roundedness can vary independently of the other factors; therefore, each vowel sound has a rounded and unrounded variant. Openness and backness are partially coupled, such that when the vocal tract is open, it must be mostly (but not completely) back. The three possible extremes, then, are [ɑ] (open, back), [i] (closed, front), and [u] (closed, back). Most of the remaining vowel sounds can be expressed as being a blend of one of these three vowel sounds (see Figure 2.7 for a visualization).

The frequency spectrum of vowel sounds typically shows three frequencies of relatively high power, which are called formants. In most cases, the first two formants (i.e., the lower two frequencies) are sufficient to disambiguate between vowel phones.

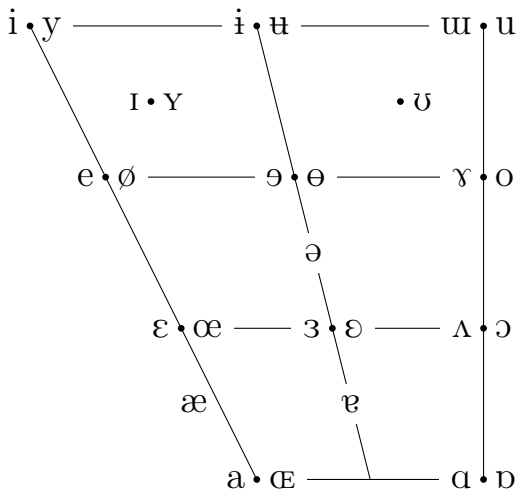


Figure 2.7: Vowels defined in the International Phonetic Alphabet (IPA), organized along three axes. The vertical axis reflects whether the vocal tract is open or closed. The horizontal axis reflects whether the tongue is near the front or back of the mouth. The last axis, on either side of each dot (·) reflects whether the lips are rounded or unrounded.

Consonant phones occur when some point of the vocal tract is constricted. The place and manner of constriction determines the consonant that will be uttered. Place refers to the location in the vocal tract that becomes constricted. For example, in bilabial consonants, both lips come together, as in [m] and [b]; in velar consonants, the tongue moves toward the velum (i.e., soft palate), as in [k] and [g]. Manner refers to how the vocal tract is constricted

in that location. For example, in a plosive, the vocal tract is completely closed at the place of articulation; air compresses behind the place of constriction, and is then released, producing a sound classified as a plosive, like [t] and [k]. In fricatives, the articulators move close together such that there is a narrow channel for air to pass through. The narrow channel causes the hissing sounds associated with phones like [s] and [f]. Several other places and manners exist; those defined by the IPA are shown in Table 2.1.

	Bilabial	Labiodental	Dental	Alveolar	Post-alveolar	Retroflex
Plosive	p b			t d		ʈ ɖ
Nasal	m	ɱ		n		ɳ
Trill	ʙ			r		
Tap/Flap				ɾ		ɽ
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ
Lateral Fricative				ɬ ɮ		
Approximant		ʋ		ɹ		ɻ
Lateral approximant				l		ɭ

	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	c ɟ	k g	q ɢ		ʔ
Nasal	ɲ	ŋ	ɴ		
Trill			ʀ		
Tap/Flap					
Fricative	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral Fricative					
Approximant	j	ɰ			
Lateral approximant	ʎ	ʟ			

Table 2.1: Pulmonic consonant phones defined in the International Phonetic Alphabet (IPA). Table headers denote the place of articulation; the first column of each row denotes the manner of articulation.

While no phone's pronunciation is consistent, consonant pronunciations can vary more than vowels because consonant sounds are produced in the context of the vocal tract position for the prior or upcoming vowel sound. In the word "bat" for example, the [b] and [t] phones occur in the context of the vowel [æ]. Kröger (1993) theorizes that each consonant sound is composed of speech gestures which move parts the vocal tract in a particular way, but allow other parts of the vocal tract, like those involved in the vocalic sound, to change freely; see Section 2.2.2 for more details.

The vowels and consonants described so far represent the most common phones used in

English. However, as with most aspects of phonetics and phonology, there are many examples that do not match the convenient criteria listed above. For example, [w], as in "weep" is a consonant that is articulated with a mostly open vocal tract, like a vowel; for this reason, it is sometimes called a semivowel. Some languages use clicks, in which inward suction releases a complete constriction resulting in a loud consonantal sound. However, the general principles of vowels and consonants hold for these phones, and do not require further treatment in this thesis.

Morpheme

The next level of meaning in the speech perception hierarchy is the morpheme. Morphemes are the smallest unit of grammatical significance in a language, and is made up of a sequence of phonemes. A morpheme is distinct from a word in that a morpheme need not be freestanding. For example, in the word "walked", both "walk" and the suffix "-ed" are morphemes, as they both contribute meaning to the word "walked."

Word

Finally, the last level of the speech perception hierarchy that we will consider is the word. A word is made up of one or more morphemes, and is the smallest freestanding unit with meaning.

A speaker's repository of known words is called a lexicon, with the words in that lexicon called lexemes. It has been theorized that people also have a "mental lexicon," which contains additional information about words, such as their meaning, pronunciation, and grammatical properties. It is clear that people have access to this information, but whether the mental lexicon is a purely theoretical construct or has clear neural correlates is an open question. We will examine this question further in Section 2.1.4.

From words, further subfields of linguistics (e.g., semantics and syntax) study how words are composed into higher order linguistic structures like phrases and sentences; however, we will only consider speech perception up to the word level in this thesis. The goal of the models described in this thesis is to map from sensory inputs to lexical items, and to map from lexical items to motor outputs, enabling a clear interface with higher order linguistic models.

2.1.4 Neurobiology

Auditory neurobiology describes both the physical substrate and organization of the system that we aim to emulate. We do not aim to fully replicate all aspects of biological neurons; instead, we will use an approximation of biological neurons, but review auditory neurobiology in order to constrain our algorithmic choices to those that could be implemented in a biological system. Similarly, by examining the organization of the auditory brain structures, we constrain the space of possible network topologies to those that match a network that we know to be successful.[3]

Brain structures in the primate auditory system are organized in several parallel hierarchical pathways. Information from the auditory periphery arrives at the brainstem. That information is relayed to the cortex through the medial geniculate nuclei in the thalamus. The medial geniculate sends information to a part of cortex in the temporal lobe called the core. The core is a small portion of auditory cortex that serves as the first stage in the mostly feedforward hierarchy of auditory cortical areas. The core is made up of three subareas, A1, R, and RT. The core is surrounded by a set of areas called the belt, which primarily receives input from the core. The belt provides input to the parabelt, which projects to higher cortical regions. See Figure 2.8 for a comprehensive view of auditory cortical connections.

The parabelt provides input to several areas, including brain areas associated with language in the superior temporal gyrus (STG) and frontal cortex, which is associated with working memory and executive function. Historically, the most salient linguistic area in the superior temporal gyrus was Wernicke's area (Brodmann area 22), as lesions in this area result in a type of aphasia in which speech can be produced fluently, but lacks meaning Ellis et al. (1983). However, more recent hypotheses concerning how auditory information becomes discrete linguistic information assigns roles to much larger portions of the superior temporal gyrus and the superior temporal sulcus. We will discuss this in more detail in Section 2.3.2.

While we do not yet know how auditory information is transformed through the connections described above, we get some clues by investigating how various auditory brain areas respond to sounds with different spatial and temporal properties. The first clue (which was used to determine the boundaries between auditory cortical regions) is that all of the areas up to the parabelt are tonotopically organized; that is, each neuron in these areas responds preferentially to a part of frequency spectrum, and cells close to one another are likely to be sensitive to the same frequency band. In general, lower level regions like the medial geniculate

[3] One notable omission from our discussion of the human auditory system is the integration of information coming from both ears. There are binaural effects at many levels, and while a full human auditory model would incorporate these effects, at this stage of research we only aim to build a monaural speech system.

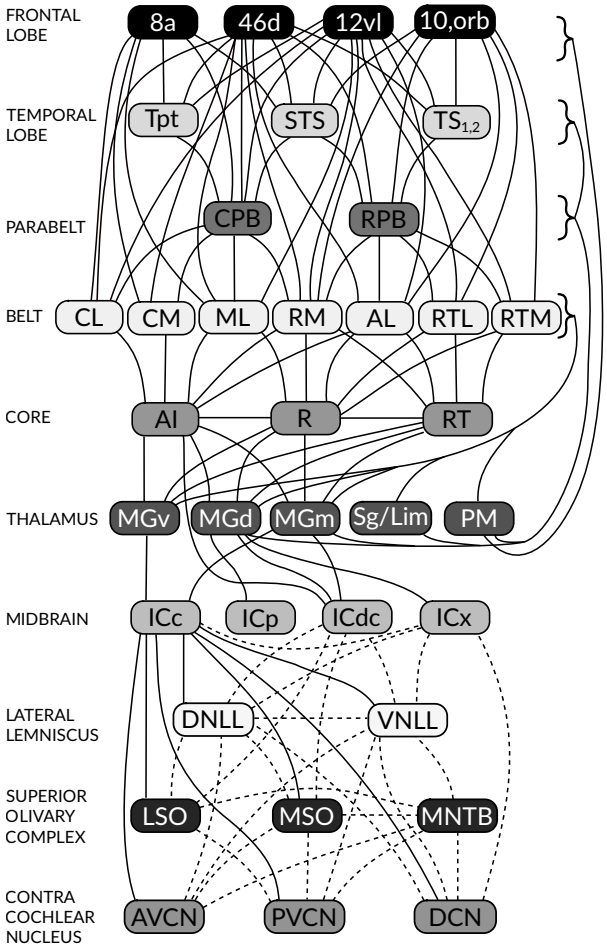


Figure 2.8: Cortical and subcortical structures involved in the primate auditory system. Solid lines denote connections seen empirically in primate auditory cortex; dashed lines denote connections theorized to exist based on findings in other mammals. The brain areas and specific connections are not germane to this thesis, but are presented here to illustrate the scope of the problem in attempting to model auditory cortex. Adapted from Kaas and Hackett (2000); see that paper for explanations of acronyms.

and core are more tightly tuned to a smaller part of the frequency spectrum (sometimes even more tightly tuned than the auditory filters in the cochlea), while higher level regions like the parabelt are more broadly tuned to larger parts of the frequency spectrum.

Neurons at various levels in the auditory hierarchy respond to their characteristic frequency

range with different temporal characteristics. In general, we can express the temporal relationship between the incoming frequency spectrum and the resulting neural activity with a causal filter. Since we can also consider the spatial tuning of cells as a filter, many auditory researchers summarize the spatial and temporal tuning properties of auditory neurons using a single filter called the Spectral-Temporal Receptive Field (STRF). STRFs summarize how a neuron responds to incoming stimuli both spatially and temporally (Aertsen and Johannesma, 1981). In general, STRFs show delays of 5–250 ms; see, Mesgarani et al. (2014, Supplementary material) for example STRFs, and Kaas et al. (1999); Kaas and Hackett (2000); Scott and Johnsrude (2003); Semple and Scott (2003) for reviews.

It should be noted that the information gleaned from spectro-temporal tuning curves describes a linear relationship between the incoming frequency spectrum and how a particular neuron responds. Tian and Rauschecker (2004) suggest that belt neurons in rhesus monkeys respond nonlinearly to frequency-modulated sweep stimuli, but Kowalski et al. (1995) found that cat A1 responses were linear. It is difficult to evaluate the responses of cells, as traditionally the STRF is determined using simple stimuli like white noise; however, see Theunissen et al. (2000), for a method using complex sounds. Escabı and Schreiner (2002) found nonlinear responses in approximately 40% of the midbrain inferior colliculus neurons they recorded, suggesting that nonlinearities are widespread throughout the auditory hierarchy.

The nature of the nonlinear processing of speech in the human auditory cortex was explored in Pasley et al. (2012) through recordings of intracranial field potentials in superior temporal gyrus and middle temporal gyrus. Pasley et al. built two models that attempted to reconstruct the acoustic signal using the data recorded from the intracranial electrodes. One model was linear, convolving each electrode's data with its corresponding spectro-temporal receptive field; the other model used a nonlinear filter based on modulation energy. They found that words could be decoded from reconstructed acoustic signals with up to 50% accuracy using data from the best subject; average accuracy was around 30%, chance accuracy was 2.1%. Notably, the linear reconstruction model was best suited to decoding speech with slow to intermediate temporal fluctuations (e.g., syllables), while the nonlinear model was best suited to decoding speech with fast temporal fluctuations (e.g., syllable onsets and offsets). This study implies that modulation (i.e., how quickly the acoustic signal is changing) may be part of important nonlinearities in the human auditory system up to and including superior temporal cortex.

While it is still unknown how these linear and nonlinear transformations of acoustic information map to lexical items or other information in the mental lexicon, we have some idea of the general timecourse of the acoustic-to-lexical mapping. Kutas and Hillyard (1980) recorded the electric potentials on human subjects' scalps (i.e., electroencephalography [EEG]) and found a significantly negative evoked potential 400 ms after a semantically unexpected word (see Kutas and Federmeier 2000 for a review). Using magnetoencephalography (MEG), Pylkkä-

nen and Marantz (2003) obtained better spatial resolution, theorizing that the negativity at 400 ms (the so-called N400) is a summation of three separate signals. At 150–200 ms, electrical activity peaks, and is thought to be related to letter-string processing. At 250 ms, there is an activity peak related to phonological factors, such as the frequency of the sounds in the word. At 350 ms, there is an activity peak related to accessing lexical information.[4]

Finally, while little concrete is known about how words and word meanings are represented, it is clear that word representations are distributed across networks of neurons distributed across cortical areas (Pulvermüller, 2001). The conceptual information associated with a word is represented in brain regions associated with the perception or action associated with those words (Martin, 2007). Patterson et al. (2007) studied cases of semantic dementia and noted that the meanings associated with a word can be damaged even when modality-specific information was retained. Patterson et al. therefore argued that, in addition to the information represented across brain regions, the anterior temporal lobes act as amodal semantic "hubs" that can activate modality-specific representations in other parts of cortex. The mental lexicon, therefore, may have close links to the semantic hubs in the anterior temporal lobes, but in general the information contained in the mental lexicon is distributed across cortical areas, and therefore is most likely a theoretical construct.

2.1.5 Automatic speech recognition (ASR)

Automatic speech recognition (ASR) has been an active field of research in artificial intelligence since 1952, when Davis et al. (1952) presented a system for recognizing spoken digits by a single individual with accuracy above 97%. This system was implemented with special purpose hardware, matching the spectral properties of the sound to known spectral patterns. From these humble beginning, modern speech recognition systems can transcribe speech in realistic environments with accuracy high enough to be deployed commercially. Two significant advancements have enabled progress in ASR.

The first significant advancement was the development of Hidden Markov Models (HMMs) and related techniques for speech recognition. Briefly, HMM-based speech recognition models attempt to find the word sequence with maximum probability given an audio waveform. They do this through a pipeline in which short time-slices of the audio waveform (called "frames") are analyzed to yield lower dimensional feature vectors through signal processing techniques. A sequence of feature vectors is then decoded into words using an HMM-based acoustic model that uses sub-phonemic states to emit phoneme labels, which are then composed into words

[4] It should be noted, however, that the studies described in Pylkkänen and Marantz (2003) used visual information (written words) rather than speech, so the timecourse of lexical access in speech may differ.

using large corpora of lexical and linguistic information (see Figure 2.9, Bahl et al. (1983), and Rabiner (1989); Gales and Young (2008) for reviews). The first part of the pipeline that produces feature vectors is often called the "frontend;" the second part that uses statistical methods to infer labels is called the "backend."

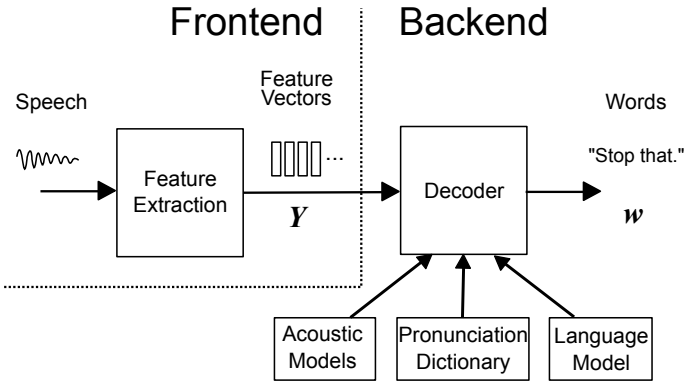


Figure 2.9: Basic organization of an ASR system using HMMs. In the frontend, feature vectors are extracted from the speech signal. Words are decoded from those feature vectors using acoustic models, which statistically model the acoustic-to-phoneme mapping, a pronunciation dictionary, which uses domain knowledge to determine likely phoneme strings, and a language model, which statistically models which words are most likely in the current context. Adapted from Gales and Young (2008).

HMM-based systems were among the first systems successful enough to be used commercially (see Huang et al. 2014). In 1989, Lee and Hon (1989) achieved a phone accuracy rate of 66.08% on a subset of the TIMIT speech corpus (Garofolo et al., 1993). Four years later, Lamel and Gauvain (1993) used a more sophisticated HMM-based system to achieve 72.9% phone accuracy on all of the TIMIT speech corpus. Improvements from then until the mid-2000s were modest (see Lopes and Perdigão 2011).

Since 2006, a series of learning algorithms and network structures in artificial neural networks (ANNs) that are now referred to as "deep neural networks" (DNNs) have been applied to the domain of automatic speech recognition with considerable success both in academic research and in commercial application. Recently, Graves et al. (2013) achieved phone classification accuracy on TIMIT of 82.3%, significantly better than the most sophisticated HMM-based model's error rate of 72.9%.

The architecture of DNN-based systems is typically simpler than that of HMM-based systems. The backends of DNN-based systems rarely incorporate explicit prior information, like

the language model and pronunciation information. In some cases, frontends uses raw spectral information, as in Mohamed et al. (2012) (see Hinton et al. 2012 for a review). These simpler architectures may be responsible for much of DNN's success, as its internal representations are not locked to specific representations like triphones, as are used in some HMM-based systems. However, using simpler architectures does not necessarily mean that using DNNs is straightforward; the effort in developing ASR models with DNNs shifts from developing complicated architectures to developing complicated learning algorithms that have many parameters which must be carefully tuned.

One potential benefit of DNN-based approaches is that they have analogies to how humans perceive speech. DNNs leverage simple computational units (artificial neurons), which operate in parallel and communicate through unidirectional weighted connections. While the computations done by these neurons and the learning algorithms that adjust the connection weights are not directly implementable with biologically plausible components, mappings between DNNs and biologically plausible spiking neural networks are currently under development (Hunsberger and Eliasmith, 2015).

2.1.6 Auditory periphery modeling

Most ASR frontends do an ideal power spectral analysis using Fourier and Fourier-like transforms, mimicking the function of the human auditory periphery. However, as discussed in Section 2.1.2, the human ear's frequency decomposition is far from ideal. Some differences in how the ear analyzes the frequencies in sound are due to the continuous nature of the real world; it is not possible for the ear to maintain a perfect history of the past 50 milliseconds of sound. Some differences are due to the nature of biological computation, which is distributed, analog, and robust to noise. Unfortunately, it is difficult to know the reason for each difference. Historically, auditory periphery models have attempted to reproduce the auditory periphery as closely as possible, whether or not that difference is advantageous for speech recognition.

Several approaches to auditory periphery modeling have progressed in parallel over the past century. Detailed mechanical models like those reviewed in Ni et al. (2014) aim to model ear physiology as accurately as possible. Unfortunately, efficient implementations of these models are not readily available. Phenomenological models, on the other hand, aim to reproduce the output of the human ear as closely as possible using simplified mathematical models; efficient implementations of these models are freely available (Fontaine et al., 2011). We will focus only on phenomenological models of the auditory periphery.

Generally, phenomenological auditory periphery models are made up of a cascade of linear and nonlinear filters. Together, the filters can model how the sound pressure level

is transformed through the middle ear, how it deflects the basilar membrane, how those deflection are transduced into electrical signals by the inner hair cells, and how those signals result in action potentials traveling down the auditory nerve. Not all models include all of these stages. For reviews of prominent auditory periphery models and the auditory-nerve response characteristic reproduced by those models, see Lopez-Poveda (2005) and Lyon et al. (2010). The auditory periphery models used in this thesis will be discussed in further detail in Section 5.1.

2.2 Speech production

We limit our coverage of speech production to the topics drawn upon in the model that will be presented in Chapters 3 and 6: basic vocal tract physiology, linguistics, neurobiology, and articulatory speech synthesis.

2.2.1 Vocal tract physiology

The human vocal tract performs the opposite role as the human ear: it translates electrical activity in the brain into muscle activations, which cause fluctuations in air pressure.

The vocal tract consists of the laryngeal cavity, the pharynx, the oral cavity, and the nasal cavity (see Figure 2.10). Air expelled from the lungs is transformed by these four structures in turn to form the specific air pressure fluctuations that we interpret as speech. Major changes in air pressure waves occur in the laryngeal cavity, which contains the glottis. The glottis is made up of the vocal folds, which are muscles that are able to vibrate rapidly, and the opening between the vocal folds. When the glottis is completely open, air passes through mostly undisturbed, resulting in low-frequency breathy sounds. When the glottis narrows, air passes through more turbulently, resulting in higher frequency breathy sounds, as in the phone [h] in the English word "had." In all voiced phones (e.g., all vowels), the vocal folds vibrate, resulting in a "buzzing" quality; compare voicing "ah" to forcing air out of the mouth in the same vocal tract shape. The exact action of the glottis is responsible for many of the subtleties in human speech, such as intensity (speaking versus shouting), frequency (low versus high pitch), and quality (e.g., harsh, breathy, murmured, creaky).

The pharynx and oral cavity can be moved into many different shapes by the muscles of the vocal tract. The shape of these remaining portions of the vocal tract cause further turbulence of the air passing through, resulting in specific air pressure patterns (sounds) that we interpret as speech. The areas of the vocal tract that can be moved in order to effect a linguistically relevant

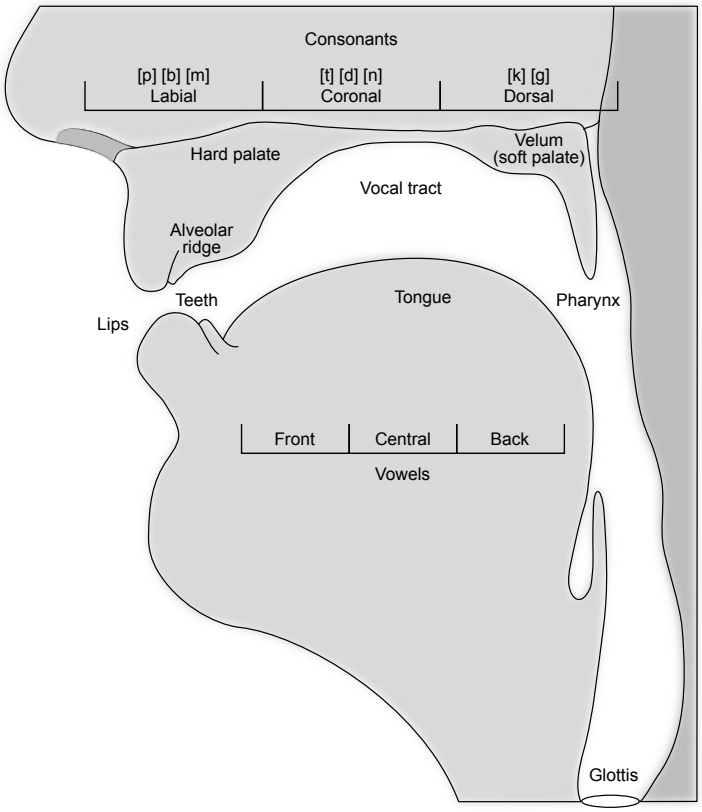


Figure 2.10: Sagittal view of the human vocal tract, showing the seven main articulators; refer to the text for details on how the articulators contribute to speech production. Adapted from MacNeilage and Davis (2001).

sound are called articulators. As will be discussed further in subsequent sections, the positions of these articulators relative to each other determines the phoneme that will be voiced when air passes through the vocal tract. The seven articulators are as follows.

1. The **pharynx** is a tube at the back of the throat. It carries air from the larynx to the oral and nasal cavities.

2. The **velum** or soft palate opens or closes the velopharyngeal port to the nasal cavity, which is important for distinguishing nasal and oral sounds. It is also a constriction target for the tongue, as in the phones [k] and [g].

3. The **hard palate** is the hard surface at the roof of the mouth. It cannot move, but serves as a constriction target for the tongue.

4. The **alveolar ridge** is the area between the hard palate and the top front teeth. Like the hard palate, it serves as a constriction target for the tongue.

5. The upper and lower **teeth** are at the front of the mouth. They primarily serve as constriction target in fricative phones.

6. The **tongue** is a large, flexible, muscular structure that can reach several different constriction targets. The tongue is often divided into the tip, blade, dorsum, and root, though this is not a strict anatomical separation.

7. The **lips** are another important flexible, muscular articulator that is involved in many speech sounds. The lips can move toward other articulators to constrict airflow, or can become rounded to change the overall quality of a sound.

Other parts of the vocal tract are important to speech; for example, nasals require air to pass through the nasal cavity. However, air is routed to the nasal cavity through constrictions of the seven articulators above; the nose does not move and therefore is not considered an articulator. Exact definitions of the articulators do vary between phoneticians, and play an important role in the design of a speech production system. We will discuss this in further detail in Section 7.2.

2.2.2 Linguistics

Our treatment of linguistics in the context of speech production aims to explain the hierarchy presented in Figure 2.11 (which contrasts with the speech perception hierarchy in Figure 2.6). We pay particular attention to two sets of characteristics in hierarchy levels and in the mapping between levels: temporal patterns and constraints, and the existence of rule-like regularities. For example, if our model were to deal with sentences, it would be important to know that, temporally, sentences occur sequentially, and cannot co-occur. There are grammatical rules about what words can be in a sentence, and in what order they can be arranged, but there are few rules placing constraints between sentences—any sentence *could* follow any other sentence, though there may be semantic issues. Similar constraints exist at the lower levels of language that we will examine in this section, and these constraints will inform the structure of the model we present in Chapter 3.

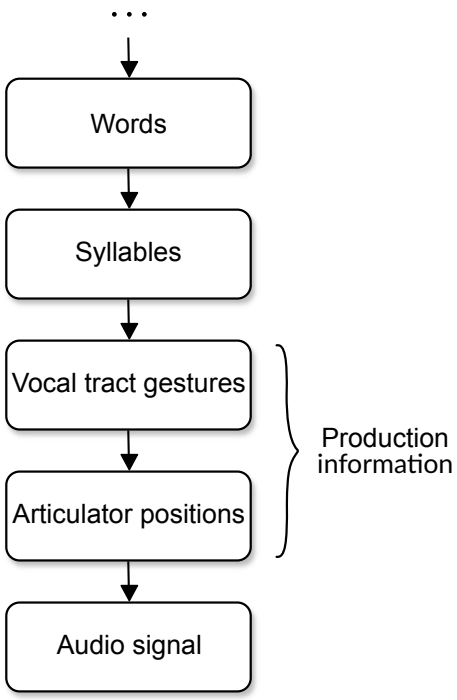


Figure 2.11: Hierarchy of linguistic concepts in speech production. While there are many important linguistic processes generating sequences of words, we focus on the hierarchy levels below words. Words are composed of one or more syllables, which are the last discrete units in the speech production hierarchy. Syllables are defined by a coupled set of vocal tract gestures (a gesture score), which define target articular positions over time. Successful movement of vocal tract articulators to those target positions results in disturbances in air pressure perceived as speech.

Vocal tract gestures

According to many phoneticians, the fundamental unit of speech production is the "gesture," which describes some movement of the articulators in the vocal tract (Browman and Goldstein, 1989; Goldstein et al., 2006).[5] Gestures are the basic unit of the speech production hierarchy (playing a similar role to phonemes in the speech perception hierarchy), making them integral to language. However, there is evidence that children learning to say their first words often use

[5] Some authors refer to vocal tract gestures as vocal tract actions or speech actions, which may better match terminology used in general motor control literature. However, we will refer to sets of learned vocal tract movements as vocal tract gestures in this thesis.

the correct gestures, but take time to order them correctly to produce speech, indicating that gestures develop before language, and are leveraged when developing language (Browman and Goldstein, 1989).

Gestures are learned movements of sets of articulators in the vocal tract. Each gesture has its own temporal dynamics; that is, a gesture is a function defined over space (what muscles are contracted) and time (when contractions occur). In general, a gesture can be thought of as a point attractor in which the position of one or more vocal tract articulators is parameterized. A minimal gesture is parameterized by the set of articulators engaged by that gesture, and the degree to which the articulators are constricted; for example, widening and narrowing the velic aperture is a gesture consisting of the articulator involved (the velum) and degree to which it is constricted (low degree for narrow, high degree for wide). Other gestures require constriction location and constriction shape parameters to disambiguate them from other similar gestures; for example, the tongue tip is involved in gestures that depend on location (e.g., constriction at the teeth, alveolar ridge, or hard palate) and shape (e.g., tongue tip can be straight or curved backwards). All gestures are also affected by global parameters affecting all gestures, most notably "stiffness," which defines how quickly the gesture attempts to reach the point of attraction.

Gestures are rarely discussed in isolation as phonemes only occur when certain sets of gestures overlap or occur in tight succession. Therefore, gestures are always grouped into gestural scores (see Figure 2.12) which embody the spatiotemporal coordination of several gestures to produce a linguistically relevant sound or series of sounds (e.g., a syllable or word). The vertical axis of a gestural score is the articulator set. Each articulator set can be used for multiple gestures; these gestures can even overlap in time for an articular set, resulting in a combination of the effects of both gestures. The horizontal axis of a gestural score represents time, though the time axis may be stretched depending on the global stiffness parameter. Gestural scores can also be visualized as a graph, as in Figure 2.12, which highlights that certain gestures must co-occur or have tight temporal couplings such that one gesture starts as another gesture ends.

As shown in Figure 2.11 and as will be discussed in detail in Section 3.1.5, there is a mapping between gestures and vocal tract articulator positions. In articulatory speech synthesizers, the mapping depends on what articulators are available in the articulatory synthesizer, and how those articulators can be manipulated. Unfortunately, few models make a clear distinction between the set of gestures and the mapping from gestures to articulator trajectories, resulting in there being no agreed upon set of vocal tract gestures in speech. We will discuss our interpretation of gestures in Section 7.2.

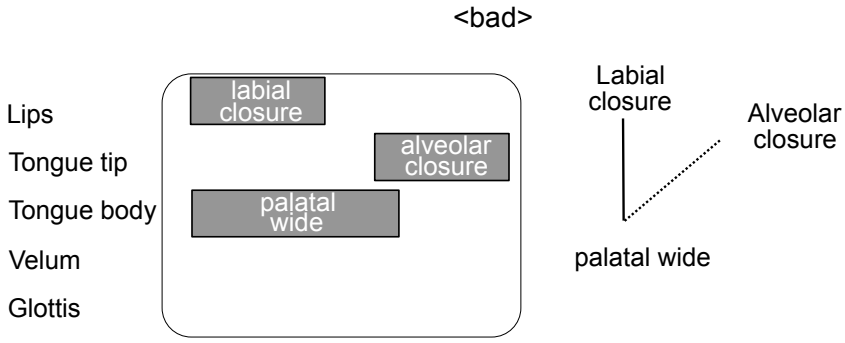


Figure 2.12: Example gesture score for the word "bad." The gesture score is composed of three gestures. The coupling graph on the right denotes that the labial closure and palatal wide gestures should co-occur, while the alveolar closure gesture's onset is tied to the offset of the palatal wide gesture. Adapted from Goldstein et al. (2006).

Syllables & words

Words can be decomposed into one or more syllables (syllabified), which provides information about how that word is pronounced. While the syllabification of a word may vary between speakers or even utterances by the same speaker, syllables are thought to be the basic free-standing unit of production (see Levelt and Wheeldon 1994; Levelt et al. 1999; Cholin et al. 2004 for arguments and empirical support).

In terms of speech perception concepts, syllables are groups of one or more phonemes with a well-specified structure. They consist of a loud vocalic center, with optional quieter consonantal components before and after the center. Unlike other levels of organization, the phonemes in a syllable may not be strictly sequential; some phonemes may co-occur when voicing a syllable.

The phonemes that make up a syllable are typically grouped into the onset and rhyme, where the onset is a cluster of consonant phonemes, and the rhyme is a vowel phoneme (called the nucleus) and an optional cluster of consonant phonemes (called the coda). In English, the onset is also optional, though this is not true in all languages. Figure 2.13 gives examples with this grouping. Note that other ways to describe syllables exist (e.g., the *mora*; Otake et al. 1993), but we adopt this method as it is the most common for Germanic languages like English.

Phonetically, syllables are a useful level of organization because the higher-level aspects of an utterance—rhythm, prosody and stress, for example—are easier to analyze in terms of their effects on sequences of syllables rather than on sequences of phonemes; it is easy to

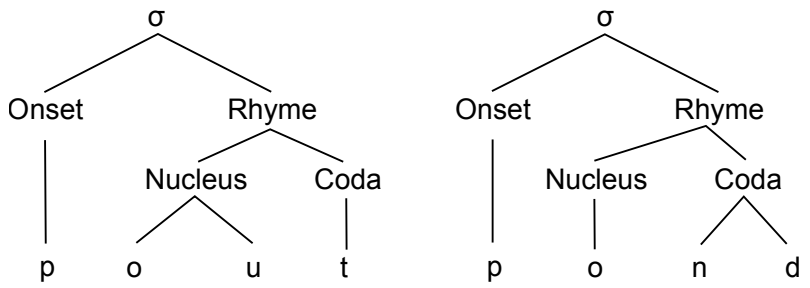


Figure 2.13: Syllable structures of monosyllabic words "pout" and "pond." Syllables in English are composed of an optional onset consonant cluster, a vocalic nucleus, and an optional coda consonant cluster. Adapted from https://en.wikipedia.org/wiki/File:Syllable_illustrations_3and4.svg.

distinguish changes in pitch and volume within and between two syllables, but not between two phonemes, because they occur quickly and can co-occur.

One aspect of the stress of an utterance is the weight of a syllable. "Heavy" or "strong" syllables have a branching rhyme, or a branching nucleus, meaning that they end in a consonant, or contain a long vowel or diphthong, respectively. "Light" or "weak" syllables do not have branching rhyme or nucleus, and in general are shorter and quieter. In English, most vowel phonemes appear most frequently in either heavy or light syllables; for example, the schwa, [ə], usually appears in light syllables, and diphthongs like [aʊ] usually appear in heavy syllables. The weight of a syllable determines whether or not it can be stressed in an utterance; specifically, only heavy syllables can be stressed.

Like all levels of phonetics and phonology, the above description is a useful simplification of more complex phenomena. In terms of syllable production, the onset-nucleus-coda grouping is not always sufficient. In English, syllables that may have once been typical VC syllables have morphed into "syllabic consonants," in which the vowel is no longer present; for example, "bottle" is often pronounced [bɒtl], where [l] is a syllabic consonant. In terms of syllable perception, agreement between native English speakers is surprisingly low when asked to segment an utterance into its component syllables. Yet, the meaning of the utterance is understood by all subjects. Therefore, the syllable's role in speech perception may be limited.

Syllables, however, play a critical role in speech production. A series of theoretical and experimental studies by Levelt, Roelofs, Indefrey and colleagues has proposed that speech production involves several stages, including phonological encoding, and phonetic encoding (Levelt et al., 1999; Roelofs, 2000; Cholin et al., 2004; Indefrey and Levelt, 2004; Indefrey, 2011).

In phonological encoding, a sequence of phonemes is retrieved from the mental lexicon corresponding to a lexical item, and are organized into syllables through an explicit syllabification stage (called "prosodification" as it includes prosodic elements of syllables). This method of phonological encoding contrasts with other speech production theories which assume that the syllabification of a word is stored in the mental lexicon (e.g., Dell 1988). Using phoneme strings and constructing syllables on the fly is necessary to capture the phenomenon of "resyllabification," in which the same word has a different syllabification in different contexts (e.g., "pre-dict" versus "pre-dic-ted"; see Cholin et al. 2004).

In phonetic encoding, phonological syllables are mapped to "syllabic gestures," which are learned gesture scores corresponding to frequently used syllables. Levelt (1992); Levelt and Wheeldon (1994) theorize that we have a repository of learned gesture scores for frequently used syllables called a "mental syllabary," which is queried in the phonetic encoding process. These learned gesture scores are then executed by a motor system to generate speech. See Figure 2.14 for an illustration of both the phonological and phonetic encoding processes.

2.2.3 Neurobiology

Traditionally, speech production was believed to be tightly linked to Broca's area, a sizable region of the dominant (usually left, for right-handed individuals) hemisphere of the frontal lobe, the inferior frontal gyrus. Paul Broca made this area famous through study of two patients with profound difficulty producing language; one patient could only produce the word "tan," while the other had only a vocabulary of five words which he would utter repetitively (Broca, 1861). While we now have a much better understanding of the brain regions involved in speech production, how these regions are connected and what information they communicate to connected areas is still the subject of active research.

Modern understanding of the neurobiology of speech production views Broca's area among several areas in the inferior frontal gyri of both brain hemispheres that temporally controls a series of tightly timed activations of motor cortex. Flinker et al. (2015) showed through Granger causality analysis that Broca's area is activated before motor cortex, and while the motor cortex is engaged in producing vocal tract movements, Broca's area is relatively silent. We will discuss its role in transforming incoming linguistic information from sensory areas to representations useful for production in Section 2.3.2, and instead focus here on how motor areas effect vocal tract movements.

Controlling vocal tract muscles in order to produce recognizable speech can be thought of as a special case of general motor control, in which we aim for relatively low dimensional targets (i.e., position in three-dimensional space) with high dimensional controls (i.e., muscle

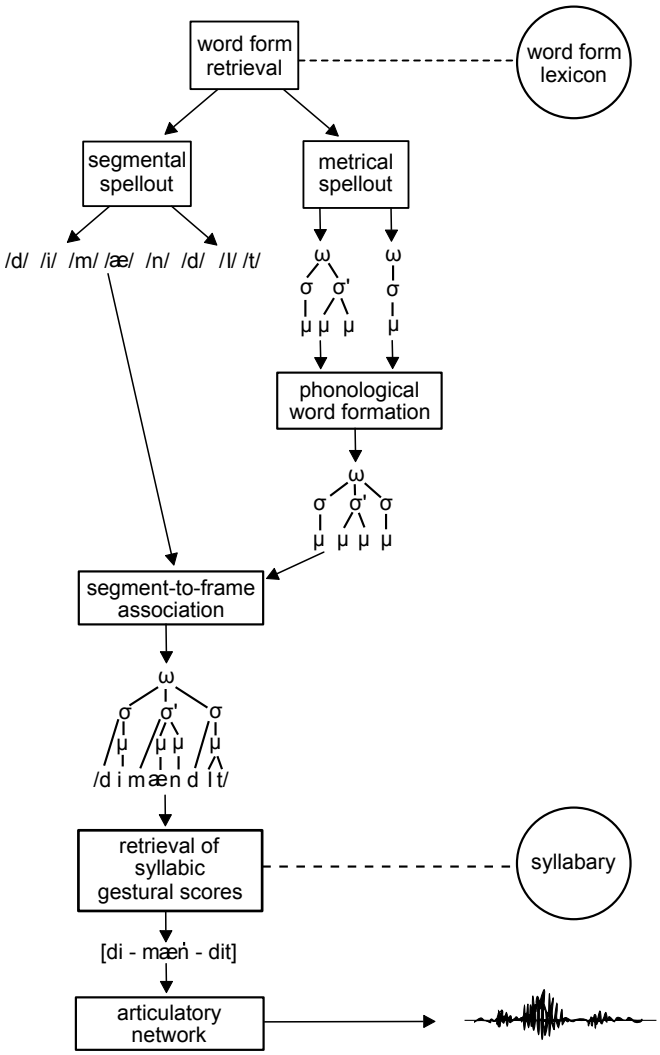


Figure 2.14: Phonological and phonetic encoding processes theorized by Levelt and Wheeldon (1994); Roelofs (2000). In the phonological encoding process, a sequence of phonemes is generated and organized according to learned syllable structures. Gesture scores corresponding to these syllables, which may contain phonemes from multiple words, are retrieved from the learned mental syllabary and then voiced. Adapted from Levelt and Wheeldon (1994).

contractions). Despite the number of tightly coordinated muscle activations that come about when we make an intentional action, we do not have to explicitly think about those individual muscle activations, which has led to the concept of motor synergies. Motor synergies are

learned sets of muscle activations that result from activation of relatively few neurons in motor cortex. Synergies can be flexibly weighted and combined; for example, if one has a synergy for reaching forward and one for reaching to the left, activating both synergies equally could effect a reach diagonally forward-left. Many different variations on the idea of motor synergies exist, and neurobiological support for these variants can be found if certain analysis techniques are used (see Tresch and Jarc (2009) for a review in general and Smith and Zelaznik (2004); Smith (2006) for evidence of speech synergies). We limit our discussion of motor synergies to the abstract concept, and do not choose a specific variant. As such, our primary interest is in what brain areas correspond to the levels of the speech production hierarchy (see Figure 2.11).

We theorize that motor synergies are responsible for moving vocal tract articulators to target positions. There is evidence that motor synergies related to speech exist in the ventral sensorimotor cortex (vSMC). Bouchard et al. (2013) found that the activity in vSMC during the voicing of a syllable is highly correlated with activation of specific vocal tract articulators, and that the vSMC is organized somatotopically (i.e., distinct regions of the vSMC influence different vocal tract articulators). Breshears et al. (2015) built on this work by directly stimulating vSMC in humans who were undergoing brain surgery. vSMC stimulation in different subsections resulted in movement of vocal tract articulators. They also found that the somatotopic organization of vSMC varied for each subject, but all subjects' vSMCs were organized in dorsal-ventral articulator order.

DeWolf (2010) integrated neurobiological research into a model of the motor hierarchy that involves the interaction of supplementary motor areas, basal ganglia, cerebellum, primary motor cortex, and nuclei in the brain stem and spinal cord (see Figure 2.15). We assume that vocal tract movements use the same hierarchy, with vSMC as the top level of that hierarchy (i.e., it is the supplementary motor area for the vocal tract). Wildgruber et al. (2001) provides support for the role of basal ganglia and cerebellum in speech, and Brown et al. (2009) provides support for the role of primary motor cortex in speech.

Motor synergies for vocal tract movements provide a method for implementing vocal tract articulator movements, but there remains a link between vocal tract gestures and articulator movements. Neurobiologically, that missing link appears to be implemented by the anterior insula, which is located in the lateral sulcus separating the temporal lobe from the parietal and frontal lobes. As reviewed in Ackermann and Riecker (2004), the insula seems to be responsible for the coordination of the up to 100 muscles that shape the vocal tract during speech and non-speech vocalizations. They (and others like Ivry and Robertson 1998) hypothesize that the mapping from from vocal tract gestures to articulator movements is lateralized such that the left anterior insula operates discretely on a fast phoneme-level timescale, while the right anterior insula operates continuously on a slower syllable or word timescale suggesting that these two timescales would be represented with separate neural resources in a neural model of

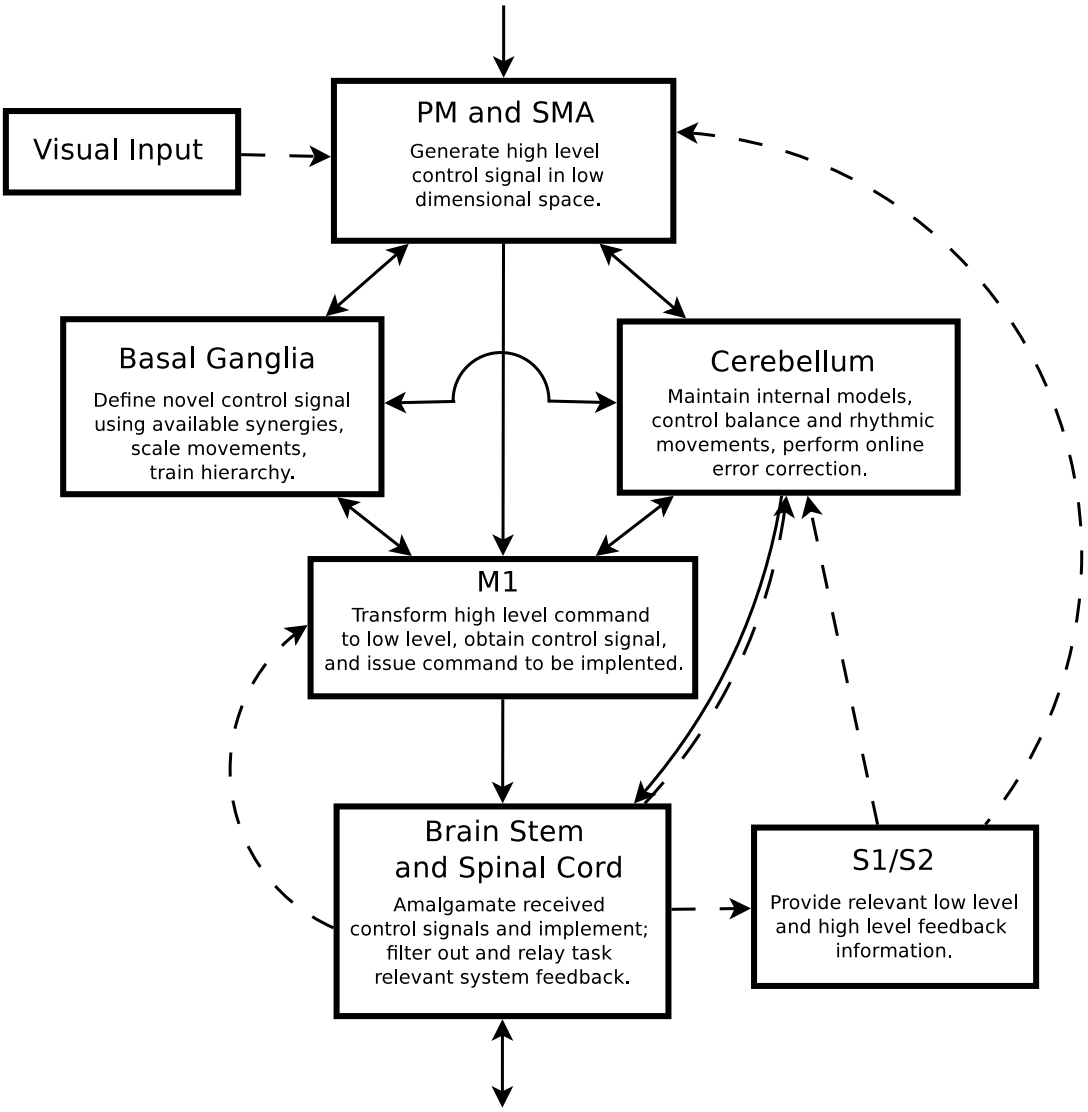


Figure 2.15: Illustration of the main components of the Neural Optimal Control Hierarchy (NOCH) model, which summarizes available empirical data of the function of motor-related brain regions. Acronyms: PM is premotor cortex; SMA are supplementary motor areas; M1 is primary motor cortex; and S1/S2 are primary and secondary somatosensory cortices. Reproduced from DeWolf (2015).

the speech system.

2.2.4 Articulatory speech synthesizers

The final piece in speech synthesis is the translation of vocal tract gestures of continuous articulator positions to audio signals. Articulatory speech synthesizers include an explicit model of the vocal tract, a model of the glottis, and an acoustical model which generates audio signals from the glottis and vocal tract models. Many systems have been proposed for the three models that make up an articulatory synthesizer. We will briefly review the main approaches, but leave a detailed treatment of the articulatory synthesizer used in this thesis for Section 5.6.

Glottis models emulate the function of the glottis, which generates a pulse of air that is modified by the shape of the vocal tract. Glottis models generate periodic pulses that are modified by the vocal tract model to generate an audio waveform. One of the earliest glottis models is the two-mass model presented in Ishizaka and Flanagan (1972), which models the vocal folds as two stiffness-coupled spring-mass systems. Titze (1989) proposed a geometric model parameterized such that the glottal flow, glottal area, and vocal fold contact area can be varied. Birkholz et al. (2011) proposed a modified two-mass model in which the two masses are angled, instead of parallel to one another. Birkholz et al.'s two-mass model is able to simulate different vocal qualities, including whispery and falsetto voices.

Vocal tract models fall into a few different classes. One class of vocal tract models are statistical models, which extract relevant characteristics from a large set of data, mostly magnetic resonance imaging (MRI) paired with audio recordings (e.g., Badin et al., 2002). A full three-dimensional representation of a subject's vocal tract can be reconstructed with sufficient data. Using dimensionality reduction techniques, the articulatory control parameters can be of relatively low dimensionality while still producing a full 3D representation of a vocal tract.

Another class of vocal tract models are biomechanical models, which attempt to fully reconstruct the physiology of the vocal tract (e.g., Dang and Honda, 2004). This type of model is similar to the statistical model, in that MRI and other anatomical measurements are used to develop a 3D representation of the vocal tract. However, instead of statistically reducing the dimensionality to determine the degrees of freedom available to the control system, these models also incorporate how muscle activations affect the shape of the vocal tract, and use each muscle as a degree of freedom.

A final class of vocal tract models are geometric models, which generalize the overall shapes observed in MRI and anatomical studies (e.g., (Mermelstein, 1973)). A set of parameters are used to generate a 3D model of a vocal tract. While these models are typically less detailed

than those generated from precise measurements of one subject's vocal tract, they can be used to generate vocal tracts for different subjects (both male and female). The parameters of the model can therefore be tuned to a specific speaker, though the resulting vocal tract will not be as detailed as one determined solely through measurements of that speaker.

The acoustic model takes a vocal tract representation and simulates the airflow from the lungs, through the vocal tract, to the air pressure coming out of the mouth. These models typically relate the vocal tract to some other device with known physics. One set of models treat the vocal tract as an analog transmission line, similar to models simulating arterial blood flow (Meyer et al., 1989). Another set of models treat the vocal tract as an analog electrical circuit (Maeda, 1982). A final set of models, called finite element wave propagation models, explicitly model the air pressure wave as it travels through each section of the vocal tract (El-Masri et al., 1996). Hybrid techniques combine one of the above techniques with an analogous transformation in the frequency domain for added accuracy, as in Sondhi and Schroeter (1987).

2.3 Sensorimotor integration

At some point in the human speech system, sensory and motor aspects of speech are interconnected. These interconnections are especially important during speech development and other situations which require adaptation. Here, we briefly review leading theories of how we develop and learn speech, then look at neurobiological studies of sensorimotor integration.

2.3.1 Development and learning

There are two main findings in speech development literature that are relevant to our model. The first is that in early developmental phases, sensorimotor learning takes place rapidly and in ways that are difficult to achieve later in development; these phases are sometimes referred to as "critical" or "sensitive" periods. Babies up to around a year of age have a sensitivity for phonemes; they become increasingly sensitive to the phonemes that they hear regularly, and increasingly less sensitive to phonemes that they hear rarely or never (Kuhl et al., 2008).[6] Up until puberty, children have a sensitivity for acquiring language in general; after puberty, formal instruction is less effective, and speaking foreign languages without an accent is more difficult (Hurford, 1991). These findings suggests that parts of our model that include learning through

[6] Interestingly, it has been shown that remaining sensitive to phonemes that are not regularly encountered is predictive of poor language performance at two years of age (Kuhl et al., 2008).

changes in synaptic connection weights may have qualitative or large quantitative differences at different developmental stages.

The second finding is that auditory learning and speech understanding precede speech production. While this may seem obvious, analogies between how humans acquire speech and how birds acquire songs suggests that the vocal babbling of children may be directed toward the goal of matching speech templates learned through auditory input (Bolhuis et al., 2010). This finding suggests that we develop a repository of speech templates that we attempt to match when learning to produce speech.

Despite the prominent role of sensorimotor integration in development, it is not the case that this system plays no role in the adult speech system. One influential study by Houde and Jordan (1998) investigated sensorimotor learning by manipulating adult speech through a device that recorded a subject voicing a syllable, adjusted the sound such that the syllable's vowel sound was manipulated in a predictable way, and played the adjusted sound back through headphones. Subjects were able to compensate for the adjustment such that subsequently uttered syllables more closely matched the desired syllable. That adjustment generalized to other syllables that had not yet been tested.

2.3.2 Neurobiology

Unlike situations in which we have a sensor and actuator to measure, sensorimotor neurobiology can only be investigated in neural recordings from cortical areas. While we do not yet have a full understanding of how the brain's speech perception and production systems are organized and interact, two influential theories have provided a fruitful framework for many studies.

The first theory is the dual-stream model of speech processing proposed by Hickok and Poeppel (2007). This model is conceptually similar to the dual-stream model of visual processing, in which the ventral stream is involved in object recognition and the dorsal stream is involved in guiding actions.[7] In the speech processing case, the ventral stream is involved in lexical processing (i.e., the meanings of incoming sounds), and the dorsal stream is involved in sensorimotor action mapping. In this thesis, we develop a model of parts of the dorsal pathway, but developing an interacting ventral pathway is an important next step.

[7] The dual-stream model for visual processing has traditionally been though of as the ventral "what" pathway, and the dorsal "where" pathway, assigning the dorsal stream the role of determining object locations (Ungerleider, 1982). However, more recent studies have made strong links between dorsal stream activity and motor actions (Andersen, 1997; Rizzolatti et al., 1997; Rizzolatti and Matelli, 2003).

Hickok and Poeppel (2007) suggest that the dorsal stream is strongly involved in speech development, as development requires auditory feedback to guide the creation and fine-tuning of motor pathways to effect speech; the role of the dorsal pathway may be reduced over time as motor pathways reliably produce recognizable speech (Schmidt, 1975; Doyon et al., 2003). Anatomically, the dorsal stream begins with the dorsal superior temporal gyrus (STG; i.e., Wernicke's area) and the superior temporal sulcus (STS), which both project to a parietal-temporal boundary area at the Sylvian fissure, called area Spt. They propose that area Spt is a sensorimotor interface, which also takes in input from other sensory modalities (i.e., visual and somatosensory information). Spt then projects to higher-level articulatory areas, including the posterior inferior frontal gyrus (pIFG; i.e., Broca's area), premotor cortex (PM) and the anterior insula. As was discussed in Sections 2.1.4 and 2.2.3, the STG has clear roles in auditory processing, and the anterior insula has clear roles in motor processing; area Spt, therefore, is a strong candidate for where sensorimotor integration might occur.

Hickok et al. (2009) provide human fMRI evidence that area Spt is significantly active during both sensory and motor aspects of a speech task. However, Cogan et al. (2014) performed electrocorticography in human epilepsy patients and found electrodes that activated for both sensory and motor aspects of a speech task across all areas of the dorsal stream (including STG, middle temporal gyrus, PM, and inferior frontal gyrus). Clearly, there is a large overlap between the information required for sensory processing and the information required for motor processing.

The second influential theory in speech neurobiology is the motor theory of speech perception. The central tenet of this theory is that perceiving speech is perceiving vocal tract gestures (Liberman and Mattingly, 1985; Galantucci et al., 2006).[8] This makes a hypothesis about the internal representations of speech, and is therefore difficult to interrogate directly, but several experiments appear to support this hypothesis. First, we are able to integrate gestural information from multiple modalities; visual perception of gestures influences how we perceive the sound. A well-known demonstration of this phenomenon is the McGurk effect (McGurk and MacDonald, 1976), in which the same audio waveform is perceived as a different syllable when accompanied by a video showing the lip gesture that produces the audio waveform.[9] Second, in a speech reaction task, humans react approximately 26 ms quicker when imitating a cue syllable compared to when reacting to the cue syllable with a predetermined syllable

[8] The motor theory of speech perception also originally claimed that speech is "special," and therefore cannot be studied with the same techniques used in other modalities (e.g., visual perception or proprioception in the context of feedback control). In their review of the motor theory of speech perception, Galantucci et al. (2006) found insufficient empirical evidence supporting the claim that speech is special.

[9] The McGurk effect can be experienced firsthand through a video demonstration at https://www.youtube.com/watch?v=G-lN8vWm3m0.

(Fowler et al., 2003).[10] Typically, reaction time experiments in which the subject produces the same reaction regardless of the cue (e.g., pressing a button) achieve faster reaction times than experiments in which the subject has a choice of reactions and the cue is arbitrarily related to the reaction. If the cue is non-arbitrarily related to the response (e.g., the cue appears on the right side when a button on the right is to be pressed) then reaction times are closer to those of the simple experiments. The fact that imitating the syllable (i.e., having a choice of three possible responses, but receiving non-arbitrary cues) is faster than a simple reaction implies that the the information gathered from the cue is directly related to the motor response; since vocal tract gestures are thought to be the basic unit of speech motor action, then it is likely that vocal tract gestures are also an early unit in speech perception as well.

Neurobiologically, the motor theory of speech perception cites similar empirical evidence as the dual-stream model; i.e., there are brain areas (like Spt) that are selectively activated when engaged in both speech perception and speech production tasks, so it is likely that they share a representation which might be vocal tract gestures. Indeed, while much of the auditory literature attempts to decode phonemes from brain areas (see e.g., Mesgarani et al. 2014 and Figure 2.16), neural activities representing vocal tract gestures would necessarily be capable of decoding phonemes due to the fact that vocal tract gestures are responsible for effecting movements that produce those phonemes. While they do not make this link directly, Mesgarani et al. (2014) showed support for the motor theory of speech perception by showing that in the STG, which is typically thought of as an auditory-specific brain region, there is topographic organization based on the phonetic properties of the phoneme being listened to (see Figure 2.16). These phonetic properties are directly linked to the articulators producing the phone (e.g., labial versus dorsal consonant phone) and therefore it is plausible that the STG represents vocal tract gestures, despite being an area primarily linked to speech perception.

Given that neurobiological evidence can support both theories, we therefore adopt the view that the dorsal auditory stream represents vocal tract gestures explicitly, though this may or may not be the case in the ventral stream. Our model, which purports to replicate functions of the dorsal stream, will reflect these two theories by explicitly representing vocal tract gestures.

Automatic speech recognition with production information

In terms of higher-level biological organization, almost all speech recognition systems can be mapped to the ventral stream of the human speech system, as they attempt to decode lexical

[10] For clarity, in the task, subjects shadowed a recording voicing the [ɑ] phone, which switched to either [pɑ], [tɑ] or [kɑ] at an unspecified time. Subject either voiced the syllable that they heard, or one of [pɑ], [tɑ] or [kɑ], which was kept constant regardless of what was heard.

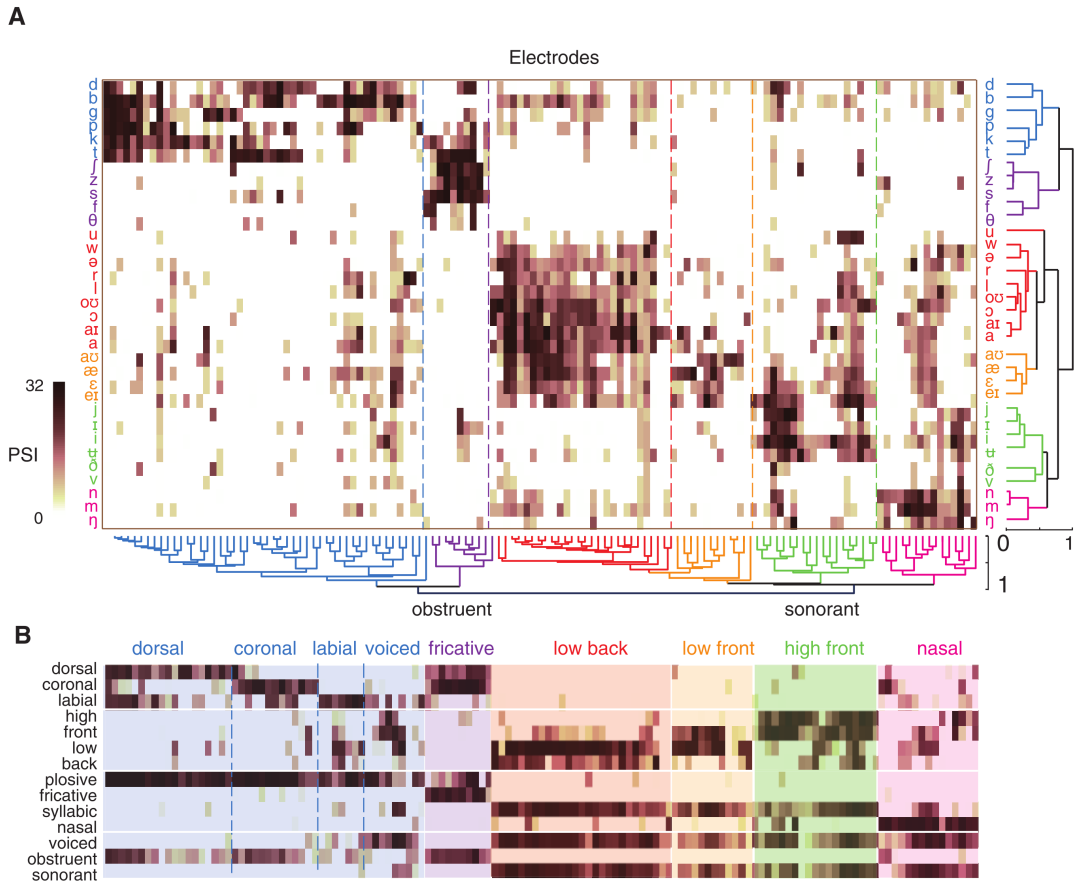


Figure 2.16: Phonetic feature encoding in human STG. (A) Intracranial recordings yield many recording electrodes with activity strongly correlated with the presentation of realizations of certain phonemes. PSI stands for Phoneme Selectivity Index, and is a measure of how selective a certain electrode is to a given phoneme (scale: 0–32). Rows correspond to phonemes, and columns correspond to electrodes. (B) The same data now associated with phonetic features rather than individual phonemes; rows correspond to phonetic features, and columns correspond to electrodes. Many electrodes are highly selective for a small set of phonetic features, which indicates that the STG represents information related to vocal tract articulators, possibly vocal tract gestures. Reproduced from Mesgarani et al. (2014).

information in order to transcribe speech to text. In this thesis, we aim to model parts of the dorsal stream of the human speech system, which we hypothesize follows the motor theory of speech perception, meaning that it uses vocal tract gestures as the basic unit of representation.

Several studies have investigated using measurements related to speech production as part of a speech recognition system (see King et al. 2007 for a review). The primary advantage of this approach is that speech production information is not locked to each phoneme, so it is not as sensitive to the differing acoustic realizations of the same phoneme. The existence of large datasets with continuous acoustic and articulator position recordings (e.g., Westbury et al. 1990; Wrench 2000; Steiner et al. 2012) has spurred development of systems that use both acoustic and articulatory information as input to an ASR system. Zlokarnik (1995) used continuous acoustic and articulatory information with an HMM-based system and reduced word error rates by more 60% (relative). Eide (2001) augmented the feature vector in a standard HMM system with articulatory features and reduced word error rates by 34% (relative). Similar studies have been done with traditional artificial neural networks (e.g., Kirchhoff et al. 2002) and dynamic Bayesian networks (e.g., Stephenson et al. 2000, 2004).

Deep neural networks have been used for a related task, acoustic-articulatory speech inversion. Here, articulatory information is decoded from acoustic information; that is, the input of the network is still an acoustic feature vector, but the desired output is articulator positions. Uria et al. (2011) obtained a root mean square error of 0.95 mm on the MNGU0 dataset (Steiner et al., 2012), which is 0.04 mm better than prior efforts using Gaussian mixture models (Richmond, 2009). The extracted articulatory information could be used by another ASR system to lower phone or word error rates.

The aforementioned results use continuous articulator positions. Some work has also been done relating this work to vocal tract gestures. For example, Zhuang et al. (2008, 2009) showed that the vocal tract gesture score can be estimated from the continuous articulator positions, which can then be used to do word classification with over 80% accuracy on a synthesized dataset. Nam et al. (2010, 2012) showed that a gesture score can be estimated from natural speech through a landmark-based time alignment procedure; however, this model assumed that the words in the utterance were known *a priori*, and applied their technique primarily for annotating data sets with time-aligned gesture scores. We have been unable to find any literature attempting to extract vocal tract gesture scores from acoustic information directly, nor through a pipeline in which articulator positions are decoded, and then gesture scores are estimated from those articulator positions.

2.3.3 Sensorimotor integration models

Currently, there are only a few models of sensorimotor integration in speech. In this thesis, we aim to build an integrated system in which the speech recognition and synthesis systems share internal representations (i.e., representations other than the incoming audio waveform or text),

and are connected such that the output of the synthesis system can be used as the input to the recognition system. By this definition, the most widely used speech recognition and synthesis systems are not considered integrated even though they can be used in a conversational manner; virtual agents like Apple's *Siri* can both recognize and synthesize speech, but recognized speech is converted to text before speech synthesis occurs, and the speech synthesis system cannot be modified over time as it relies on a large corpus of recorded speech.

The most well known system that could be considered integrated[11] is called DIVA (Direction Into Velocities of Articulators; see Figure 2.17 and Guenther 1995; Guenther and Perkell 2004; Guenther 2006; Guenther et al. 2006). DIVA consists of an articulator synthesizer and several interconnected artificial neural networks that drive the articulatory synthesizer and process auditory and somatosensory feedback to improve future synthesized speech. DIVA is not pre-programmed with a repository of programs to effect various sounds; instead, it mimics human speech development by learning to control the synthesizer through an initial babbling phase in which random articulator movements are associated with auditory and somatosensory feedback. DIVA has been used to explain speech production phenomena like motor equivalence, contextual variability, and anticipatory and carryover coarticulation (Guenther, 1995; Guenther and Ghosh, 2003; Nieto-Castanon et al., 2005). Additionally, all of the neural networks in the model have been mapped to brain regions that are hypothesized to perform similar functions as the neural networks in the model.

DIVA is a productive research tool for studying speech development. However, I do not believe that DIVA in its current form can scale to maintaining conversations with adult vocabularies. Currently, DIVA is focused on speech synthesis, so while it can incorporate acoustic feedback to better synthesize speech, there is no clear path to incorporate speech recognition and higher-level linguistic capabilities. Additionally, DIVA currently represents stored speech plans with one artificial neuron per speech plan, where a speech plan can correspond to a single phoneme, a syllable, or one or more words. In Guenther et al. (2006), they acknowledge that "it is expected that premotor cortex sound maps in the brain involve distributed representations of each speech sound." Using such distributed representations would require several changes to DIVA's structure and learning algorithms. While these changes are possible, they would represent a significant modification that may affect empirical results obtained with the current version of DIVA.

In addition to using highly localist representations, DIVA has other biological plausibility issues. It represents auditory feedback using the first three formant frequencies, but does

[11] Note that while DIVA uses sensorimotor associations, it does not have a speech recognition system. Auditory information is processed online to improve synthesis, but the lexical identity of auditory information is not decoded.

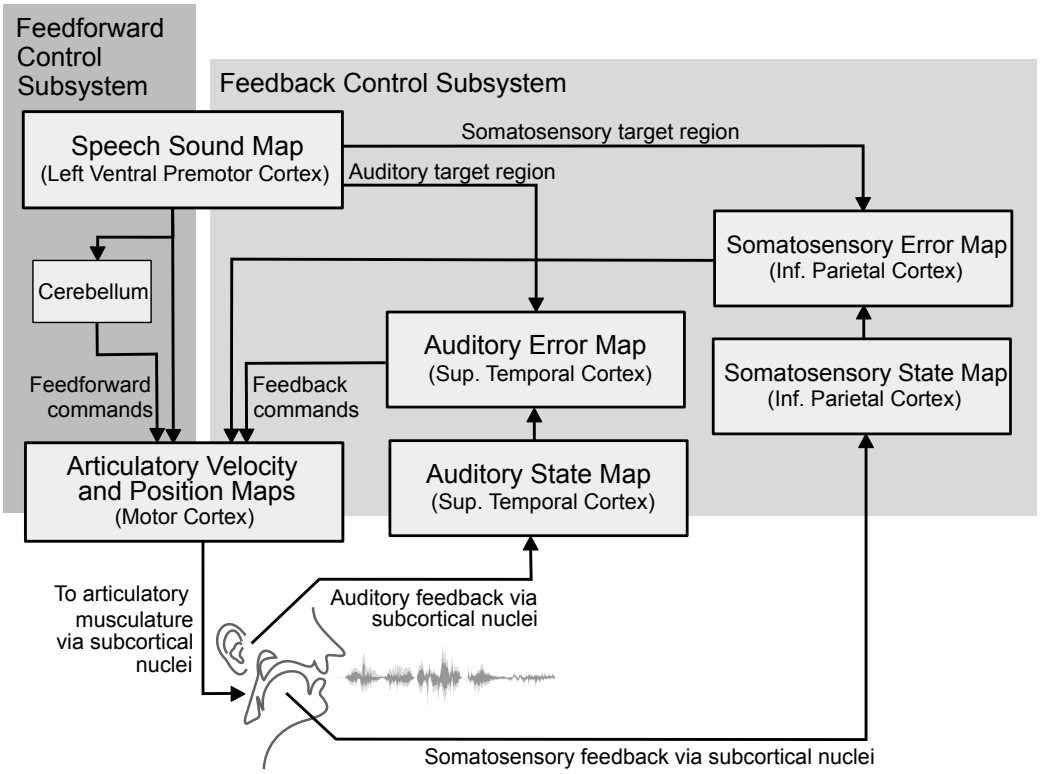


Figure 2.17: Overview of the DIVA system. DIVA is a speech production model that uses auditory and somatosensory feedback to improve speech synthesis. An utterance is initiated by the feedforward control system, and is improved through auditory and somatosensory error signals computed by the feedback control system. Adapted from Guenther et al. (2006).

not provide an account of how the brain might extract those frequencies from the incoming air pressure levels. The artificial neural networks consist of homogeneous neurons with non-spiking linear activation functions, though the learning procedure uses a local Hebbian learning rule. Auditory and somatosensory feedback is provided to the model with no delay, making the learning procedure much easier than a learning procedure that must deal with delayed feedback, as happens in the real world.

Kröger et al. (2009, 2014); Kröger and Cao (2015) proposed a large integrated model similar to DIVA in that it is composed of several artificial neural networks and learns to produce syllables through a babbling stage in which feedback from auditory and somatosensory systems

tune the weights between neural populations (see Figure 2.18). The primary improvements in Kröger's model compared to DIVA are the introduction of semantic and phonetic maps, and a motor planning network.

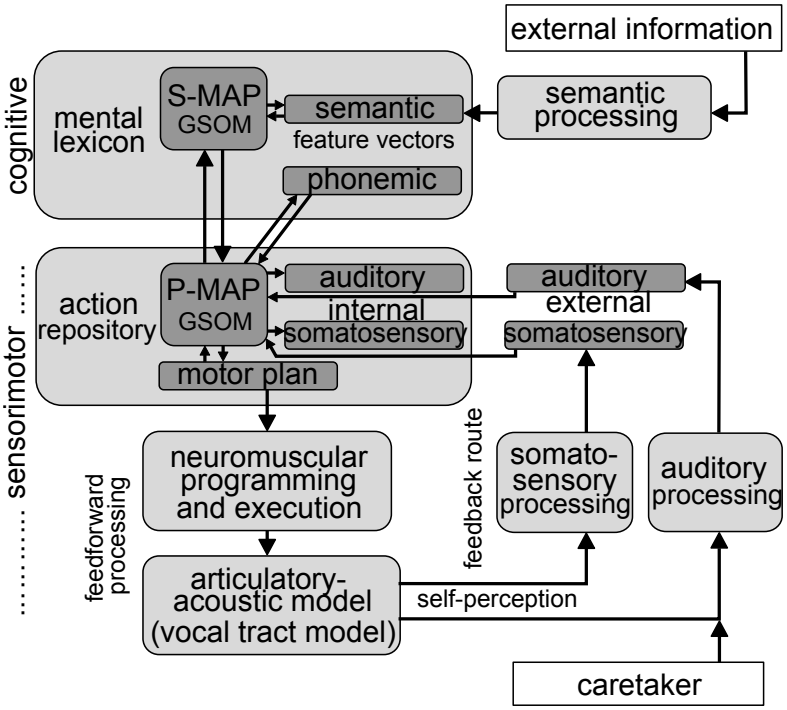


Figure 2.18: Kröger's neurocomputational model of speech acquisition. Rounded dark gray boxes indicate neural maps. Like DIVA, Kroöger's model uses auditory and somatosensory information to improve future speech synthesis. The inclusion of a semantic map and a phonetic map links the model to additional linguistic theory (the mental lexicon and mental syllabary, respectively). Adapted from Kröger and Cao (2015).

In DIVA, auditory and somatosensory feedback is directly associated with a single neuron in the speech sound map, meaning that each sound must be learned separately. This approach does not generalize across similar sounds, nor will it scale to large vocabularies of speech sounds. Kröger's model overcomes this limitation by explicitly decoding phonetic information from auditory feedback. It then uses that phonetic information to generate explicit motor plans. Motor plans act as parameterizations of the possible vocal tract gesture scores that can be produced by the model; each motor plan is defined by five parameters which define the

vocalic state, the gesture-performing articulator, and the location of articulation. These five parameters are used to generate vocal tract gestures, which are then mapped onto movements of articulators in a three-dimensional vocal tract model.

In Kröger's neurocomputational model, phonetic information is extracted in an unsupervised manner using self-organizing maps (Kohonen, 1982; Kohonen and Honkela, 2007), which cause local clusters of neurons in the phonetic map to become active when incoming auditory information (in the form of the first three formants) is similar to certain patterns seen during training. The maps are able to discriminate between three vowel phonemes and three consonant phonemes; it is not clear whether maps trained in this way can scale to discriminating between the number of phonemes in a realistic language (over 40). However, it is clear that an approach which uses a finite set of intermediate representations (i.e., phonemes) between auditory input and learned motor sequences will scale better than DIVA's approach that does not use an intermediate representation.

In contrast to these two models, the model that we present in the subsequent chapters does not focus on modeling speech development,[12] and instead implements portions of the human speech system not modeled by these two models, and aims to be more biologically realistic. Biological realism is achieved using spiking neural network models to represent information in a distributed manner that can scale to the level of adult vocabularies, and by using an auditory periphery model to process incoming sounds rather than preprocessing the sound to obtain the first three formants. In addition, while Kröger's model improves on the DIVA model by introducing phoneme representations, we follow the motor theory of speech representation by explicitly decoding vocal tract gestures from incoming sound, which makes integration between the perception and production aspects of the model more straightforward.

[12] The methods used by DIVA and Kröger's neurocomputational model to emulate speech development could be adapted and added to our model. However, since these problem have existing solutions in these two models, we have focused on other aspects of speech recognition and synthesis as being of greater scientific interest.

Chapter 3

Conceptual model

The long term goal of the research presented in this thesis is a complete neurally realized model of the human speech system, which is able to maintain natural conversations. Implementing such a system requires the collaborative efforts of many domain experts. In this chapter, I present a conceptual model of the human speech system that I believe can be implemented in biologically plausible spiking neural networks.

The model, dubbed Sermo (Speech execution and recognition model organism), is a synthesis of the background material summarized in the previous chapter. The goal of Sermo is to break the larger problem of human speech into concrete subproblems that have existing partial solutions, or provide a reasonable challenge for machine learning and computational neuroscience researchers. Pragmatically for this thesis, Sermo provides context for the subproblems for which we have implemented concrete solutions in subsequent chapters.

3.1 Sermo description

Sermo is split into five interacting systems, as summarized in Figure 3.1.

- **Auditory feature extraction** converts incoming audio waveforms into features suitable for linguistic processing and sensorimotor integration. This system is analogous to the frontend of an ASR system, and to the auditory periphery and early auditory pathways in the human speech system.
- **Non-auditory sensory modalities** gather relevant information from non-auditory sources to inform or correct other systems.

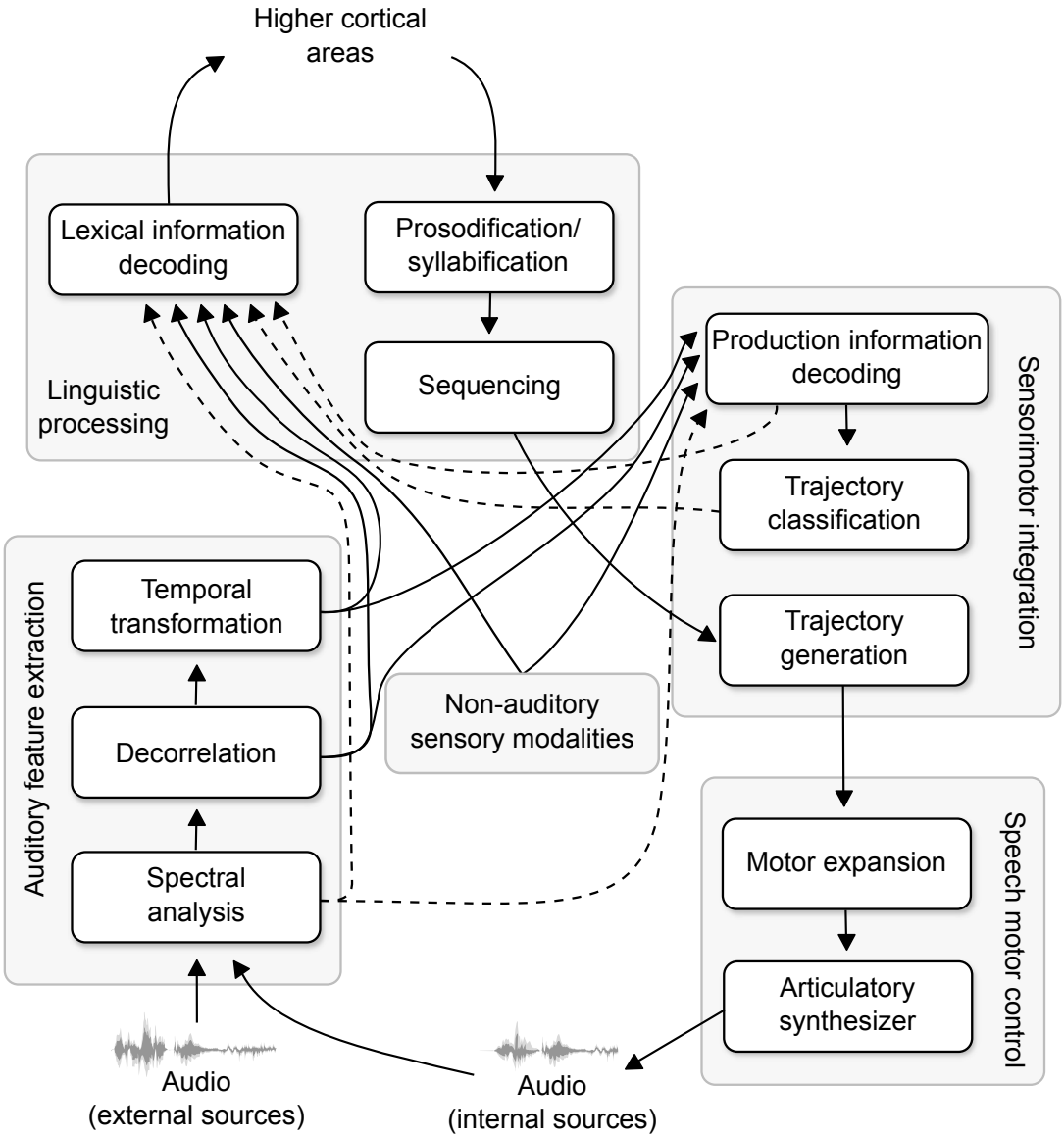


Figure 3.1: Architecture of the Sermo model. The goal of the model is to explicitly define computations needed for speech recognition and synthesis. Solid lines indicate connections that are necessary for Sermo to function. Dashed lines indicate connections that we hypothesize to exist, but are not essential for Sermo to function. See text for more details.

- **Sensorimotor integration** uses sensory information to recognize and predict the motor intentions of speakers, including one's self. This system is analogous to the dorsal stream in the human speech system.
- **Linguistic processing** converts sensory and sensorimotor information into lexical representations suitable for high level linguistic and cognitive reasoning. This system is analogous to the backend in an ASR system, and to the ventral stream in the human speech system.
- **Speech motor control** uses sensorimotor information to generate motor commands that drive an articulatory synthesizer, which produces audio waveforms.

Further details on each of these systems follow in the subsequent sections.

In addition to the overall organization of subsystems and how they interact, we also impose the following constraints on Sermo in order to make explicit that it is modeled after a biological system, and should be able to interact with biological systems naturally.

- Subsystems must operate in a continuous, online fashion.
- Subsystems must be implementable in biologically plausible spiking neurons.
- Learning rules may only use local error information, and must compute error signals with spiking neurons.

Another way to summarize these constraints is to emphasize that Sermo is a mechanistic model of the human speech system. We are not only concerned with recognizing and synthesizing speech with human-like accuracy, we also wish to gain insight into what information is required to recognize and synthesize speech.

Solutions to subproblems that do not meet these criteria (e.g., statistical models) are still a crucial component of progress in this model. Some parts of the model are necessarily statistical, and the information being transformed may not be easily explained by a label like "lexical representation." In these situations, however, it is still a critical research activity to adapt statistical models that work in discrete time, with rate based neuron models, or with idealized learning rules like backpropagation to meet the above criteria, for the following reasons.

- Models meeting these criteria can be compared directly to experimental data at multiple levels, from single unit activity recordings to behavior.

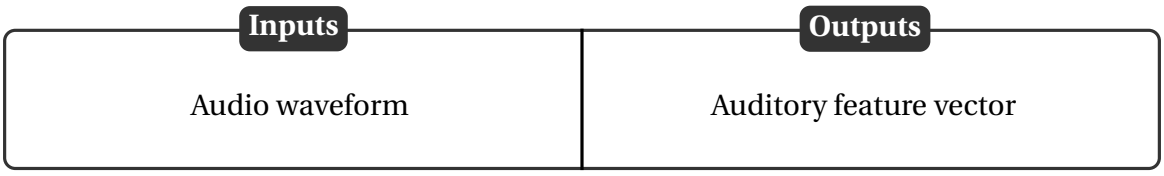
- Models meeting these criteria can be implemented in neuromorphic hardware.

Direct experimental comparisons enable iterative development of the model. As Sermo subsystems are implemented, they will be used to make testable predictions. As these predictions are tested, the model can be invalidated or updated to incorporate new empirical results.

Neuromorphic hardware presents a plausible avenue to real time interaction between humans and Sermo instances. Many of the subsystems already implemented run at many times slower than real time, despite the use of recent general purpose CPUs and GPUs, and reasonably efficient algorithms. While incremental advances in these technologies may bring subsystems closer to real time speeds, a fully integrated system with all subsystems interacting online is intractable for the foreseeable future. Neuromorphic systems that are designed to simulate millions of spiking neuron models in real time are currently under development (e.g., Furber et al. 2013; Benjamin et al. 2014), and may make a full Sermo instance possible in the next few decades.

A final note about the general architecture of Sermo and its subsystems is that we will focus on an "adult" version of Sermo that would be used in a conversational setting that does not require learning. We treat each subsystem as a separate module that is defined primarily by the input it expects, and the output it provides; however, during development and learning, there would be a proliferation of interconnections between modules providing error information. Detailing these connections in each subsystem and across subsystems would be an important area for further developing the model, but we believe that presenting a static endpoint for learning is important as a first step before considering how that endpoint might come about. Regardless, we will provide a sketch of how learning can be incorporated in Sermo in Section 8.6.2.

3.1.1 Auditory feature extraction



The auditory feature extraction system preprocesses audio waveforms in order to support linguistic and sensorimotor analysis. As has been shown through the gap between automatic

speech recognition and speech perception, the temporal characteristics of speech make these problems difficult to solve with traditional computers. Digital computation is precise and discrete; human speech is noisy and continuous. We hypothesize that the parallel and distributed nature of biological computation is central to its ability to process temporal information.

Sermo defines three subproblems that must be solved by the auditory feature extraction system: *spectral analysis*, *decorrelation*, and *temporal transformation*. In spectral analysis, the incoming auditory signal is projected into frequency space by analyzing the spectral density of the signal in the recent past. The spectral density of the signal is expected to have high correlation between nearby frequencies. Generally, these correlations make feature extraction more difficult, so the decorrelation step projects the spectral density onto a set of basis functions that are closer to orthogonal. Typically, fewer basis functions are used than frequency components, making decorrelation also a form of dimensionality reduction. Finally, while the information in the spectral densities and decorrelated vectors contain information from a certain time window, the temporal characteristics of speech may require that more temporal information is available at the current moment. Temporal transformations maintain past information to filter spectral and decorrelated spectral information for better linguistic and sensorimotor information decoding.

Spectral analysis

We theorize in Sermo that spectral analysis is performed by a model of the auditory periphery. An auditory periphery model emulates the human ear up to the signals traveling down the auditory nerve.

There are many existing models of the auditory periphery that meet the constraints imposed by Sermo (see Section 3.1). Each auditory periphery model captures certain psychoacoustic phenomena at a particular computational cost. In general, these two characteristics are traded off; the more accurately the ear is modeled, the most costly the model is to simulate.

It is difficult to assert *a priori* which psychoacoustic phenomena are important for speech. It would be reasonable to assume that the majority of what the ear does is important for an individual's survival; however, we process many types of sound, not just speech. An ideal auditory filter model for the spectral analysis subsystem in Sermo would capture all of the phenomena that are important for speech, but no additional phenomena that increase the computational cost of the model.

Decorrelation

Decorrelation techniques like linear filters and the discrete cosine transform are widely used in digital computing for compression (Khayam, 2003). In compression, a signal is projected onto a set of basis functions and only the coefficients are stored. An approximation of the original signal can later be recovered by linearly combining the basis functions weighted by those coefficients. The coefficients are much smaller to store than the original signal. A similar argument can be made to explain why automatic speech recognition systems decorrelate spectral information in their frontends: by operating on the coefficients of a set of basis functions, the set of parameters that must be optimized by the backend is reduced.

While this line of reasoning does not necessarily imply that the brain also explicitly decorrelates spectral information provided by the auditory periphery, it is clear that biological systems aim to minimize energy expenditure whenever possible. Learning through synaptic plasticity takes more energy than does typical signal transmission. Since much of the speech system is developed and refined through synaptic plasticity, using decorrelated representations as early in the speech system as possible reduces the brain's energy expenditure over its lifetime. Therefore, we hypothesize that a system implemented with a biologically plausible substrate, like Sermo, decorrelates spectral information provided by the auditory periphery.

Temporal transformation

The extent to which the recent past is used to perceive speech at the current moment is of crucial importance, yet is difficult to study systematically. In the brain, spectrotemporal receptive fields have been measured using tone ramps and other temporally varying signals, revealing neurons that appear to be active hundreds of milliseconds after activity at the cell's characteristic frequency. However, it is not clear if those neurons are involved in speech, nor is it clear what transformation they are implementing. In automatic speech recognition, the length of the audio frame and how much it increments on each timestep is often chosen arbitrarily, or uses values known to work.

Until recently, the most successful ASR systems were based on Hidden Markov Models (HMMs), which are based on the Markov property: the future states of the system depend only on the current state, not the sequence of states preceding it. Despite the fact that speech is highly temporally correlated, these systems have been relatively successful, due in large part to a great deal of engineering effort applying HMMs to speech, primarily because of its temporal nature (see e.g., Bahl et al. 1983; Rabiner 1989; Lee and Hon 1989). One such improvement for HMM-based ASR systems is to include the first and second temporal derivatives in the state representation, which provides the HMM additional temporal information without violating the

Markov property. Current state-of-the-art systems are based on hierarchies of bidirectionally connected recurrent neural networks (RNNs), which are able to maintain information about previous states through recurrent connections. It is reasonable to assume that a more flexible representation of the recent past is part of why these networks fare better than HMM-based systems which have had decades of incremental improvements.

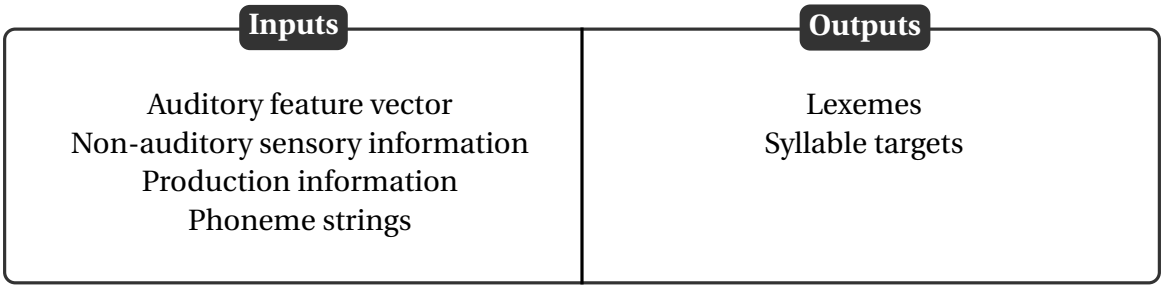
Further study into how much temporal information is maintained by deep RNNs, and how that temporal information is transformed would benefit Sermo and other speech systems. In particular, expressing deep RNNs' temporal transformations as linear time-invariant (LTI) filters would be of particular interest, as LTI filters can be efficiently implemented with spiking neural networks (see Eliasmith and Anderson 2004). Alternatively, LTI filters commonly used in engineering applications can be experimented with to investigate whether decoding accuracy is improved in downstream linguistic and sensorimotor systems.

The system described above is primarily feed-forward. It is important to note that a complete auditory processing system would match the brain, which has feedback connections between almost all layers in order to refine each layer's ability to provide useful output downstream layers (see Figure 2.8). Feedback is an undeniably important component of a full system that operates for long periods of time with sensors that are constantly changing and degrading. In this work, however, we assume that the auditory periphery does not change its ability to process sounds over a long timescale, and therefore we do not include corrective feedback signals between layers in the auditory system. This type of feedback could be added to this system in future work.

3.1.2 Non-auditory sensory modalities

Biological systems are remarkably adept at integrating information from different sensory modalities in order to guide behavior. In speech, we use visual information from other speakers to improve perception, and somatosensory information from vocal tract articulators to improve production. A fully realized version of Sermo would include these non-auditory sensory systems.

3.1.3 Linguistic processing



As summarized in Section 2.3.2 the ventral stream of the human speech system represents high-level linguistic features of speech. In Sermo, we hypothesize that the linguistic processing system *decodes lexical information* from sensory and production information, and *provides syllable targets* to be voiced to the sensorimotor integration system. Much of these processes require higher level linguistic processing, which would be implemented by models of higher cortical areas.

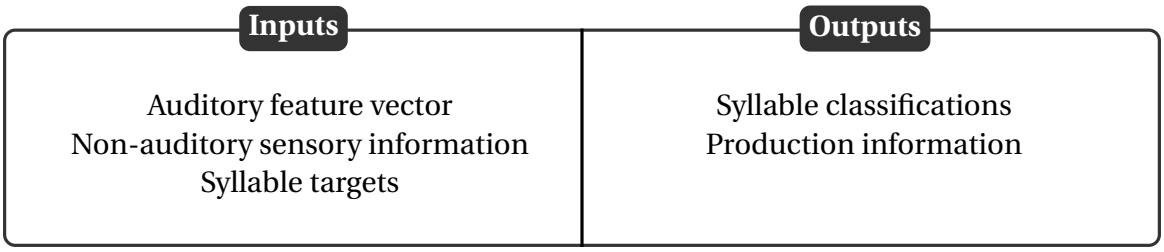
Decoding lexical items from speech has been the primary topic of research in ASR systems for decades. As such, we are confident that the methods used in ASR systems can be implemented in a mechanistic model suitable for incorporation with Sermo, as has been done for similar visual classification systems (e.g., Hunsberger et al. 2013; Hunsberger and Eliasmith 2015). Once classified, lexical items will be used in higher level linguistic systems that will associate semantics (as in Blouw and Eliasmith 2013; Blouw et al. 2015) and syntax (as in Stewart et al. 2014, 2015) to these representations. As these higher level systems have been developed with the Semantic Pointer Architecture (SPA; see Section 5.4), we hypothesize that the linguistic processing system produces lexical representations in the form of semantic pointers. However, we do not further constrain these representations as constraints will come from the needs of higher-level linguistic and cognitive systems, which are not currently part of Sermo.

The output of these higher-level systems is constrained by the needs of Sermo's sensorimotor integration system. Specifically, Sermo requires that the output of the linguistic processing system be in the form of syllable targets. Therefore, for the portion of the linguistic system connected to the sensorimotor system, semantic pointers representing sentences or words must contain syllable representations that can be queried through unbinding.

We follow the theoretical work embodied in the WEAVER++ model Roelofs (2000, 2008, 2014) explaining how syllable targets might be generated in the linguistic processing system. Higher cortical areas are responsible for providing a sequence of phonemes, which are composed into syllables on the fly through a prosodification process. Not included in the WEAVER++ model is

the notion of syllable sequencing, which we believe is necessary. The syllables constructed by the prosodification process may or may not be constructed at the same rate as syllables are being uttered. Specifically, it is likely that the next syllable, or the next several syllables, is conceptualized well before it is uttered. For this reason, a process for temporally sequencing syllables is necessary. Syllable targets are therefore provided to the sensorimotor integration system at the appropriate times.

3.1.4 Sensorimotor integration



As summarized in Section 2.2.3, the dorsal stream of the human speech system represents both sensory and motor information when perceiving and producing speech. In Sermo, we hypothesize that the sensorimotor integration system is responsible for decoding production information from sensory information, and maintaining two types of associations. First, sensory input, both auditory and non-auditory, is used to *decode the production information* that generated the sensory input. Second, that decoded production information is associated with static syllable representations; we call this *trajectory classification*, as each syllable requires a time series of sensory features on the order of hundreds of milliseconds to be reliably classified. Finally, syllable targets are associated with trajectories of speech production information; we call this *trajectory generation*, as a time series of vocal tract movement commands are necessary to voice a syllable.

As in the auditory feature extraction system, a great deal of effort is involved in dealing with the temporal nature of speech. Were the sensory and motor representations static, the associations in the sensorimotor system could be accomplished with simple associative memories (see Eliasmith et al. 2012; Eliasmith 2013). Instead, we are effectively associating sensory trajectories with motor trajectories. The key insight that enables these associations in Sermo is to use an intermediate syllable representation that can be transmitted between brain areas and queried for the full trajectory. A convenient side effect of the intermediate syllable representation is that it also enables top-down influence from other systems, most notably the linguistic processing system. It is not reasonable to expect linguistic areas to communicate

production information directly to motor cortex; however, it is reasonable to expect that the lexical representations in the linguistic system can be used to generate syllable targets which are used by sensorimotor areas.

Note that Sermo follows the theoretical and empirical speech production work in Levelt and Wheeldon (1994); Levelt et al. (1999); Cholin et al. (2004), which says that the speech production is organized at the level of syllables. While this literature is sufficient reason to use syllables, Sermo also allows us to provide two additional reasons for organizing speech production at the level of syllables: enabling transfer learning, and efficient use of neural resources.

Syllables enable flexible speech because we can use learning to fine-tune the trajectory generated by any one syllable representation individually, and that fine-tuning transfers to other words using that syllable. For example, if Sermo can improve its ability to voice the syllable [bɑ], then the quality of all words containing the syllable [bɑ] improves. Had we chosen word representations, then this transfer learning would not be possible. On the other hand, had we chosen phoneme representations, then it would be difficult to fine tune trajectories at all, as the motor trajectory of a phoneme is dependent on the previous and next phonemes, and we therefore could not fine tune the trajectory of each phoneme individually.

Syllables minimize the expenditure of neural resources because syllables are mostly independent of one another, and because there are a finite number of syllables. To illustrate why these reasons are important, let us assume that a certain amount of neural resources (neuron and synapses) are associated with the basic unit of representation. We will denote this as n, and assume that this amount is the same regardless of the representation used. If we choose phonemes as the basic representation, then we have a small number of basic units; for the sake of argument, say around 50 phonemes, which is slightly above the amount in English. Because the motor trajectory of a phoneme is dependent on the phonemes before and after it, we need neural resources not only for each phoneme, but in the worst case, also for all possible permutations of three phones; i.e., we would need $50^3 n$ or $125{,}000n$ resources. On the other hand, if we chose words as the basic unit of representation, we would not have to worry about between word interactions, but the number of words in a language is much larger than the number of syllables in a language. Using a conservative estimate of 30,000 words and 500 frequent syllables in English (Schiller et al., 1996), word representations would use approximately $30{,}000n$ resources, compared to $500n$ resources for syllables. Syllable representations are clearly the most tractable given these assumptions.

So far, we have not been explicit about our representation of production information. As will be discussed in more detail in Section 4.4.1, we could use a discrete representation of production information (i.e., vocal tract gestures) or a continuous representation of production

information (i.e., vocal tract articulator positions). Unlike in the choice of using syllable representations, the number of possible vocal tract gestures and the number of possible articulator positions is similar, assuming a reasonable parameterization of the vocal tract articulators as is done in all articulatory synthesizers. Therefore, we consider the choice of production information representation to be an empirical question that we aim to explore with Sermo.

Trajectory generation

In trajectory generation, the syllable target is used to generate a time series of production information.

There are several criteria that a trajectory generation system must meet in order to be useful in Sermo's sensorimotor integration system. First, the system must be able to produce trajectories with high degrees of freedom. There are at least twenty degrees of freedom for both vocal tract gestures and articulator positions. Despite the high dimensionality, the neural resources required for each syllable trajectory must be relatively low, as there may be up to a few thousand of these trajectory generators in an adult speech system.

Second, the trajectory generators must be flexible with respect to the timescale at which they generate a trajectory. There is significant variability in how we voice syllables. Syllables are often elongated or shortened for emphasis, and during a stressful situation, all of a speaker's syllables may be voiced quickly. A trajectory generator must be able to produce appropriate production information at speeds that vary between and within syllables.

Third, the trajectory generators must be able to operate in tight succession, and possibly even overlap in time. Speech is remarkably continuous; to robustly segment speech into syllables often requires speech to be slowed down or overenunciated. Therefore, if our notion of production requires representation at the level of syllables, then the motor trajectories resulting from those syllables must be able to blend naturally into one another without obvious pauses between different syllables, or multiple utterances of the same syllable.

Trajectory classification

In trajectory classification, we are essentially solving the inverse problem of trajectory generation. Given some time series of production information, we must infer the syllable that would produce that trajectory.

As an inverse problem, the trajectory classification system has the same challenges as the trajectory generation system. The trajectories to be classified will have relatively high degrees

of freedom; the trajectories may not advance uniformly through time; and subsequent syllables will follow in quick succession, even possibly overlapping.

Decoding production information

In the trajectory classification system, we assume that the trajectories we are classifying are trajectories of production information. It may also be possible to directly classify sensory information into syllables; however, we follow the motor theory of speech perception (Liberman and Mattingly, 1985) which theorizes that production information is directly inferred from auditory information. One theoretical reason why we assume that production information is decoded from sensory information is that it allows us to frame trajectory generation and classification as inverse problems of each other. In doing so, there may be ways in which the classification and generation systems can correct each other's errors.

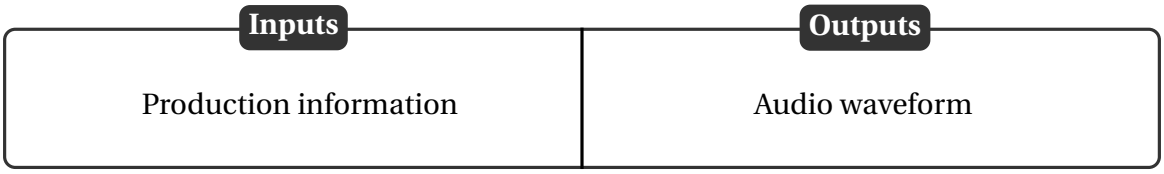
A more practical justification for decoding production information is that it provides a site in which to aggregate production information gathered from multiple sensory modalities. The McGurk effect discussed in Section 2.3.2 indicates that we use both auditory and visual information to determine which syllable is being voiced. If part of our syllable classification system is informed by a decoding of production information, then both auditory and visual systems can contribute to that decoding. The speech development systems embodied in DIVA and the Kröger model also learn to associate sensory information with one's own production information through babbling and imitation stages.

Additionally, it is possible to vocally imitate novel syllables. Novel syllables that are made up of sounds (and therefore vocal tract gestures) that are frequently encountered in one's language can be repeated readily. Novel syllables that feature novel vocal tract gestures are more difficult to repeat readily. Since both types of novel syllables do not have a learned production information trajectory (i.e., are not in the mental syllabary), it is reasonable to hypothesize that the syllable's production trajectory is decoded from the speech information, and passed through to the speech motor control system unchanged. Repeated experience with the novel syllable would lead to a production trajectory being learned and consolidated. Both trajectory decoding and learning are likely to use existing elements in the mental syllabary that closely resemble the infrequent syllable to be learned (e.g., a similar CV syllable could be used to learn a different CV syllable, following the frame/content theory of MacNeilage and Davis (2001)).

Uria et al. (2011) has shown that it is possible to do this decoding, which some call auditory-articulatory inversion (see Section 2.3.2). Currently, this has only been done with continuous articulator positions; determining if discrete vocal tract gestures can be decoded from speech

would guide further development of Sermo. We do not solve this problem in this thesis; however, we are able to use Sermo to produce a corpus of time-locked synthesized auditory information and vocal tract gestures, which can serve as a benchmark data set.

3.1.5 Speech motor control



The speech motor control system uses a continuous trajectory of speech production information to drive an articulatory speech synthesis system, which produces audio waveforms. It is composed of two components. The first is the *motor expansion* system, which projects production information into the vector space that directly controls the articulatory synthesizer. The second is the *articulatory synthesizer* itself.

Motor expansion

To this point, we have discussed production information abstractly, either as vocal tract gestures or vocal tract articulator positions. In the neurobiological case, the vocal tract model used by each individual corresponds to their own physical vocal tract, an internal model of which would be learned through sensorimotor associations. In Sermo, however, there is no perfect choice for an articulatory synthesizer. Indeed, it is a long-term goal of Sermo to provide a unified control system that can be used to compare the speech quality of different articulatory synthesizers by providing a control system that can be used for any synthesizer. In order to reach that goal, we must choose a canonical production information representation. However, since different synthesizers have different control parameters, a motor expansion system is necessary to translate Sermo's production information representation to control signals for a specific synthesizer. It allows the rest of Sermo to be designed agnostic to the articulatory synthesizer chosen because the motor expansion system will handle the mapping from production information to control signals.

One important first step, then, is to choose a canonical production information representation, so that only one mapping is required for each articulatory synthesizer. Currently, it is not clear what production representation should become that standard. Previously, we

have argued vocal tract gestures are a good choice for representing production information. However, we have not yet determined if vocal tract gestures can be decoded from acoustic information (recall that Uria et al. 2011 has done this for vocal tract parameters), nor do we have empirical evidence that vocal tract gesture trajectories can be generated and classified robustly. Additionally, there are unfortunately at least two systems that employ a disparate set of vocal tract gestures. The researchers that first proposed the task dynamics framework and the HLSyn (Hanson et al., 1999) and CASY synthesizers (Iskarous et al., 2003) use vocal tract gestures which can be expressed as a vector of eight scalars (Zhuang et al., 2008, 2009). Kroger et al. (2014) and Birkholz et al. (2006); Birkholz (2013), on the other hand, use 11 scalar gestures in their 2D synthesizer, and 46 binary gestures and 3 scalar gestures in the VocalTractLab synthesizer, respectively. If a vocal tract gesture representation is shown to be advantageous for the operations required by Sermo, then standardizing on a single vocal tract gesture representation will be a critical next step. Similarly, if it is determined that continuous production information in the form of articulator positions is a more advantageous representation for Sermo, then a sufficiently expressive set of control points will have to be chosen.

While the motivation for the motor expansion could be interpreted as meaning that it does not have an analogous system in the brain, a brain area playing a similar role as the motor expansion system may be useful for compensating for vocal tract perturbations. When our vocal tracts are modified in some way, perhaps by illness or injury, or by a deliberate manipulation as in Houde and Jordan (1998), we can adapt our speech such that changes in how we voice a phoneme transfer to other syllables containing that phoneme. Since it is unrealistic to think that each syllable trajectory is modified when adapting to the perturbation, it is reasonable to assume that there is an intermediate representation between the production information produced by the sensorimotor integration system and the control signal that is sent to the motor system. In Sermo, the representations used in the motor expansion system would play that role.

Articulatory synthesizer

Articulatory synthesizers generate audio waveforms by simulating the airflow through a model of the human vocal tract. The general organization of an articulatory synthesizer was reviewed in Section 2.2.4. As with auditory periphery models, Sermo uses existing solutions to articulatory synthesis, and aims to be a testbed in which articulatory synthesizers can be compared to one another using the same control system. From the perspective of Sermo, the details of the vocal tract and acoustic models should be abstracted away and only considered in the motor expansion system. Primarily, we are concerned with how each synthesizer is controlled, and whether the synthesizer can be used online. However, even synthesizers that cannot

be used online can still be tested with Sermo by generating the control signals online and providing them to the synthesizer offline. These synthesizers would not be suitable for real time conversation without being adapted for online use.

3.2 Evaluation

As a conceptual model, Sermo outlines what we believe a fully integrated speech system would look like macroscopically. The model provides context motivating research of each of the subproblems that must be solved for the full model to work in real time. It is important, therefore, to consider how to evaluate subsystems in Sermo, as the fully integrated model will take many years to develop.

Progress in the auditory feature extraction system is difficult to measure because it is most useful in the context of other systems. It is not readily apparent what impact an improved decorrelation network might have on the sensorimotor integration system's ability to decode production information, for example. Although we can assume that an output representation with less autocorrelation will be easier to decode, the gains from this improvement may be modest compared to changing the time constants on temporal transformations, or even increasing the number of neurons used to represent certain pieces of information.

Fortunately, the decoding systems that are the primary use of the feature extraction system can be evaluated directly through phoneme and word error rates, in the case of linguistic decoding, and through the mean squared error in the case of production information decoding. There are several existing data sets with full training and testing input-output pairs, such as TIMIT for linguistic decoding (Garofolo et al., 1993), and the X-ray microbeam database (Westbury et al., 1990) and MNGU0 (Steiner et al., 2012) for continuous sensorimotor decoding. Currently, there are no data sets for discrete sensorimotor decoding (i.e., inferring vocal tract gestures from acoustic information). We plan to synthesize a vocal tract gesture data set to catalyze progress in this area.

These data sets not only offer a method to evaluate decoding methods, but allow us to evaluate choices made in the feature extraction system. In general, we see the strength of large integrated systems like Sermo as allowing for candidates solutions to different subproblems to be evaluated in the context of known best solutions to subproblems in connected systems. In other words, one of the primary contributions of Sermo is to propose standardized input and output formats required by each subsystem. We will show a proof of concept that Sermo supports such comparisons in Section 7.1.2 in which we compare five auditory periphery models on a phoneme classification task.

The strongest evaluation of the sensorimotor integration and speech motor control systems is for listeners to perceptually evaluate synthesized speech. If a desired sequence of syllables is identified by all listeners as the intended sequence, then these systems can be considered successful. However, en route to such an experiment, the trajectory generation and classification systems can compare their outputs to known trajectories. While a version of Sermo that explains development would require many presentations in order to generalize trajectories associated with syllables, the current system is provided a desired trajectory for each syllable. Therefore, we can directly test the sensorimotor integration system's ability to classify and generate known trajectories. Classification is either done correctly or incorrectly, though some classification methods may provide confidence information that can be used to further evaluate the system. Generation, on the other hand, cannot be easily evaluated as correct or incorrect. One method of evaluation would be to measure how much the generated trajectory deviates from the target trajectory (comparing, for example, the squared jerk of the trajectories; see Hogan and Sternad 2009). However, a stronger evaluation is to use the generated trajectories to drive the speech motor control system, and perceptually evaluate generated audio. We will present other evaluation methods for the implemented subsystems in their respective sections in Chapter 7.

3.3 Relation to other models

As discussed in Section 2.3.3, the models most similar to Sermo are the DIVA model by Guenther and the Kröger model by Kröger et al.. Unlike Sermo, both of these models focus on the speech production development, rather than attempting to model an adult speech system directly. Nevertheless, there are significant overlaps between these models and Sermo.

DIVA (see Figure 2.17) primarily models the trajectory generation step in Sermo's sensorimotor integration system, but employs feedback signals to learn how to generate trajectories. Activation of a cell in the "speech sound map" generates feedforward control signals that drive an articulatory synthesizer. It also includes auditory somatosensory feedback systems analogous to Sermo's auditory feature extraction and non-auditory sensory systems.

The primary strength of DIVA that is not currently captured in Sermo is an account of the development of feedforward control signal trajectories through repeated trials in which auditory and somatosensory feedback corrects parts of the trajectory in which the synthesized sound differs from the expected sound. We believe that Sermo can be adapted to learn trajectories online in a similar way, using biologically plausible learning rules (e.g., MacNeil and Eliasmith 2011; Bekolay et al. 2013b).

There are several critical differences between how Sermo and DIVA view the overlapping systems. First, while Sermo asserts that the motor system is organized at the level of syllables in order to ensure realistic scaling to adult vocabularies, DIVA allows its speech sound cells to correspond to phonemes, syllables, words, or even short phrases. Additionally, in DIVA the motor trajectory is initiated by activating a single speech sound cell, and it is not clear how sequences of sound cells would be temporally coordinated. In Sermo, considerable effort is taken to create reasonable representations for the speech target (which in Sermo are always syllables), and to temporally coordinate sequences of syllables.

Second, the form of auditory feedback differs significantly. Sermo sees auditory processing as part of the model, and therefore uses a biologically inspired auditory filter and statistical methods that can be performed by spiking neurons. DIVA does not constrain how it processes audio waveforms, and provides the model with representations that are well suited for the operations performed by the model; specifically, they provide the model with the first three formant frequencies at each moment in time, though they have used log formant ratios and wavelet-based transformations with similar results (Guenther et al., 2006). Regardless, this ideal auditory processing gives DIVA a significant advantage in terms of being able to associate auditory feedback with control signals, which is the primary function of the model.

Finally, despite positioning itself as a biologically plausible neural network model of speech production (the "most thoroughly defined and tested" as of 2006; Guenther et al., 2006) it performs many operations that are not easily implementable in a biological system. Most notably, while simulated neurons communicate by sending signals through synaptic connection weight matrices, perfect representation and communication are assumed. In other words, there is no neural or synaptic model in DIVA; neural activations are defined as differential equations which are computed directly. The state variables in these equations are multiplied by connection weights, but signals propagate and with no delay, degradation or noise. In order to represent temporal effects, signals can be perfectly delayed by arbitrary amounts of time (Nieto-Castanon, 2011), an operation that is difficult to perform with biologically realistic neuron and synapse models. While it may be possible to approximate the computations that DIVA assumes are possible with biologically plausible spiking neural networks, it is not clear whether neural approximations will be sufficient to produce similar behavior as the idealized DIVA model.

The Kröger model also focuses on learning and development, rather than a full adult speech system. The primary system modeled is the sensorimotor trajectory generation system, which is learned using auditory and somatosensory information. This model also adds a linguistic processing system, allowing the motor system to be driven by syllable representations. Unlike DIVA, the Kröger model assumes speech motor organization at the level of syllables, which matches Sermo's motor organization, and linguistic theory. The Kröger model also introduces

an intermediate representation between the motor plan (i.e., trajectory) and the articulatory synthesizer, playing a similar role as Sermo's motor expansion system. Thus, in terms of gross structure, the Kröger model covers a larger subset of Sermo compared to DIVA.

The Kröger model also uses a slightly more sophisticated neural network model, the growing self-organized map (GSOM). GSOMs build on the self-organized map (Kohonen, 1982; Kohonen and Honkela, 2007) by introducing a growing procedure in which new neurons can be added to existing maps when the error in a certain region is above a certain threshold (Cao et al., 2014). While spiking versions of self-organized maps have been proposed (e.g., Choe and Miikkulainen, 1998), a growing self-organized map would require either neurogenesis (which only occurs in a few brain areas) or recruiting neurons in nearby areas with novel synaptic connections, a form of structural plasticity which is currently not well understood.

In terms of whether other operations of the model can be realized in spiking neurons, the outlook is more promising for the Kröger model. The auditory representation used is the power spectrum of the audio signal, which is computed as a Bark-scaled spectrogram.[1] While this is an ideal mathematical transform, it performs the same function as the auditory periphery models used in Sermo. However, while the neural network model is an improvement over DIVA, it still uses an overly simplistic representation scheme. Words in the phonemic map are represented with one neuron each; this localist representation scheme is unrealistic, and when considering the number of possible interactions between words, does not scale to adult vocabularies (Crawford, 2014).

3.4 Subsystems modeled in this thesis

In the subsequent chapters of this thesis, I will present mechanistic models implemented in spiking neurons spanning several systems in the conceptual Sermo model (see Figure 3.2).

The first model presents an implementation of the auditory feature extraction system that performs similar operations as the frontend in an automatic speech recognition system, but does so in an online fashion using an auditory periphery model and a spiking neural network. The resulting representations are called neural cepstral coefficients, and we will compare their ability to classify speech with existing approaches used in ASR.

The second model implements a syllable sequencing model as a stand-in for a full linguistic processing system, and generates production information trajectories from those syllables.

[1] The Bark scale is a psychoacoustic perceptual pitch scale, similar in many respects to the Mel scale (Zwicker, 1961).

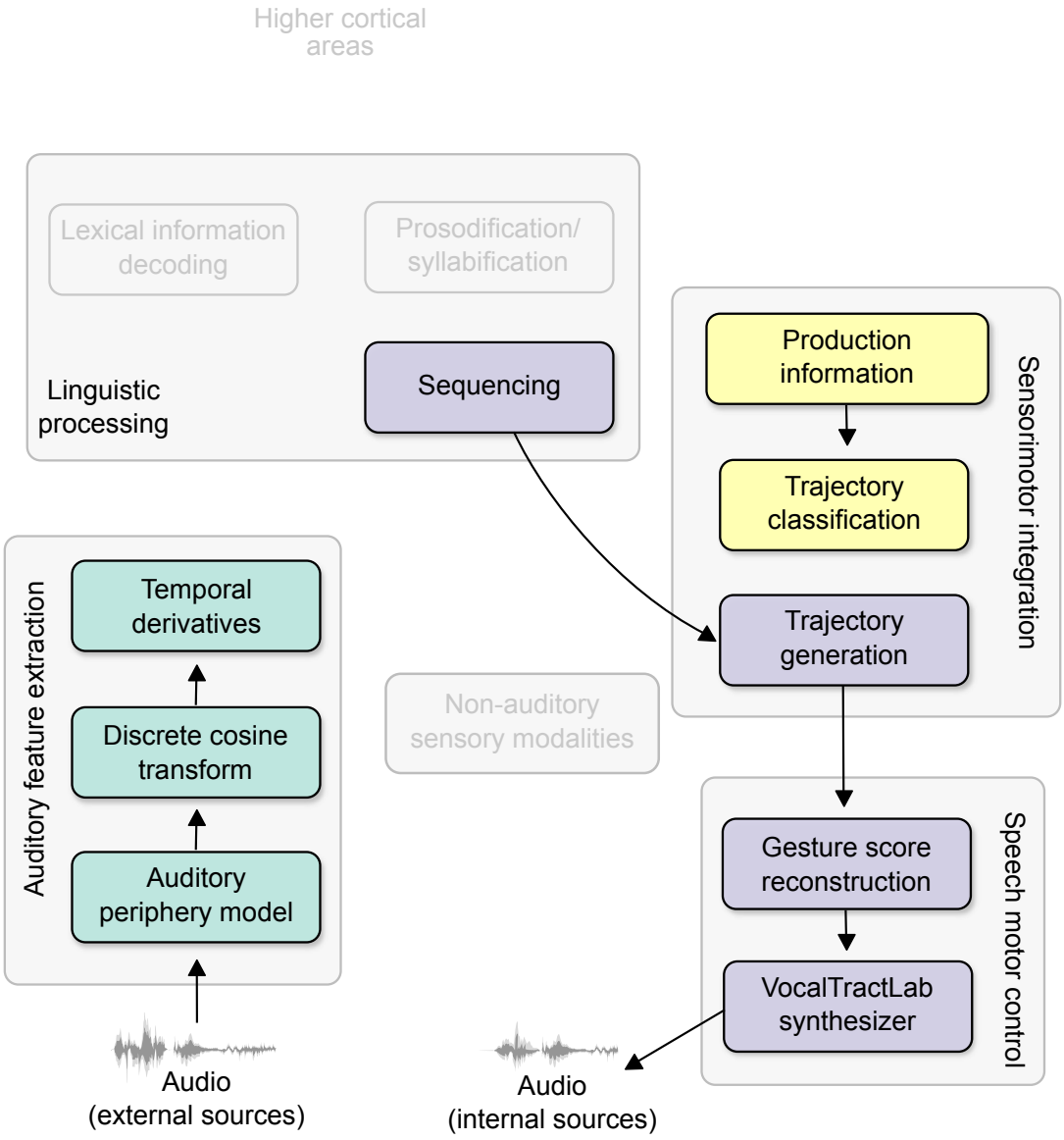


Figure 3.2: The parts of Sermo implemented in this thesis. Colors correspond to neural models. Green (left) refers to the auditory feature extraction model (i.e., neural cepstral coefficients). Purple (right) refers to the syllable sequencing and production model. Yellow (top-right) refers to the syllable classification model.

The production information will be used to synthesize speech samples with the VocalTractLab articulatory synthesizer (Birkholz et al., 2006; Birkholz, 2013). We will compare the generated trajectories to target trajectories, and make qualitative observations of the synthesized speech.

The third model implements the trajectory classification portion of the sensorimotor integration system. Production information trajectories will be provided to the trajectory classifier, which we evaluate by comparing classified syllables to the intended target syllables.

3.5 Limitations

Two primary issues limit the Sermo model as presented in this chapter. First, the subset of Sermo that is implemented in this thesis is more thoroughly defined than the parts that are not modeled in detail, such as the linguistic processing system and non-auditory sensory modalities. Existing models like the WEAVER++ model have given us guidelines as to what computations are necessary, but how those computations might be implemented in spiking neural networks remains unanswered, and may require different computations than have been assumed thus far.

It is therefore important to emphasize that we envision the Sermo model to be a continually growing and adapting model, incorporating new modeling ideas and new empirical results across disciplines. The current presentation of Sermo was developed independently, primarily informed by the literature summarized in Chapter 2. We hope that upon presentation, feedback from other researchers will fill in missing pieces of the model, and clarify existing pieces, making the development of Sermo a collaborative effort.

However, I recognize that organizing a collaborative effort like Sermo brings additional challenges. Therefore, to enable easier collaboration, we will use modern Internet-based tools like Github and the Jupyter notebook to visualize and disseminating information, in addition to the traditional scientific publication system. These tools provide mechanisms for proposing and discussing changes to a shared repository of materials, which will include text, figures, and code.

A second issue that directly opposes the idea that this model must be developed through a large collaborative effort is that Sermo's insistence on using mechanistic models that can be implemented with spiking neural networks limits the number of researchers that have the background necessary to contribute models that implement Sermo subsystems. However, we believe that with the growing interest in neuromorphic hardware and the development of deep learning approaches in spiking neural networks, interest and expertise in spiking neural networks will steadily increase in the coming years. In the same vein, it may be argued

that the constraints placed on candidate Sermo models are too stringent given our current understanding of the human speech system, and spiking neural networks in general. I believe that the models presented in the rest of this thesis serve as evidence that progress can be made with the tools currently available. As interest and expertise in the modeling techniques used in this thesis grows, so will the possibility that all of the components in Sermo can be implemented efficiently.

Chapter 4

Previous work

In this chapter, we review existing complete or partial solutions to the three problems solved by the models presented in the subsequent chapters.

4.1 Auditory feature extraction

As summarized in Section 3.1.1, the auditory feature extraction system is based on the feature extraction pipeline used as the frontend of automatic speech recognition (ASR) systems. The pipeline pictured in Figure 2.9 is similar across most systems.

4.1.1 Mel-frequency cepstral coefficients (MFCCs)

The most common feature extracted and used in ASR systems is called Mel-frequency cepstral coefficients (MFCCs). It has been widely used in both hidden Markov model-based (HMM; Hain et al., 1999; Gales and Young, 2008; Gaikwad et al., 2010; Alam et al., 2013) and deep learning-based (Graves et al., 2006; Graves, 2008; Fernández et al., 2008) ASR systems, including those that achieve the lowest error rates on the popular corpus TIMIT (Garofolo et al., 1993; Lopes and Perdigão, 2011). It has also been shown that MFCCs are well suited for both speech and music inputs (Logan, 2000).

Mel-frequency coefficients are inspired by the human auditory system, in the sense that they perform frequency decomposition of the audio signal in order to determine the relative power at frequencies distributed over the Mel scale. A simple algorithm for computing a Mel-frequency cepstral vector for a frame of audio is as follows.

1. Compute the discrete Fourier transform of the audio frame.
2. Take the log of the power spectrum.
3. Convolve the power spectrum with triangular filters distributed according to the Mel scale.
4. Apply the inverse discrete cosine transform (iDCT) to the triangular filter outputs.

The resulting coefficients obtained from the iDCT are called "cepstral" coefficients.[1]

Recall from Section 2.1.2 that the Mel scale describes the relationship between absolute frequency and perceived pitch; it is a logarithmic function of frequency. The triangular filters are spaced equidistantly on the Mel scale, resulting in more filters at lower frequencies than at higher frequencies. Typically, at least twenty triangular filters are used in order to ensure that all frequencies are captured in more than one filter (i.e., there is overlap between adjacent filters). The inverse discrete cosine transform is designed to decorrelate the signal, which results in the same amount of information being represented with fewer coefficients; typically, the first thirteen coefficients are used in the feature vector of ASR systems.

4.1.2 Delta MFCCs

In addition to the thirteen MFCCs, many ASR systems, both HMM-based (Hain et al., 1999; Gales and Young, 2008) and deep learning-based (Graves et al., 2006; Graves, 2008; Fernández et al., 2008), also append the first and (sometimes) second temporal derivatives of the MFCC to the feature vector.

The justification for including derivatives in the feature vector is typically a practical one, in that most ASR systems achieve higher accuracy with derivative information than without. Theoretically, most sources justify time derivatives by noting that the derivatives incorporate dynamics into the state representation. However, even recurrent deep learning systems like Graves (2008) use MFCC derivatives despite the fact that state information from many previous frames is available at the current timestep. It therefore seems likely that a sufficiently sophisticated machine learning algorithm would learn that temporal derivatives are a useful feature; incorporating it into the feature vector is not necessary, but effectively bootstraps the learning process by providing it a feature that it would have learned regardless.

[1] The term "cepstrum" comes from the word "spectrum" with the first four letters reversed, as the spectrum is obtained with the Fourier transform and the cepstrum is obtained with the inverse Fourier transform. Similarly, the domain of the cepstrum is not frequency, but "quefrency."

Temporal derivatives are not the only MFCC transformation that are done in ASR frontends. Some systems apply an additional transformation analogous to low-pass filtering called "liftering" that emphasizes the lower part of the cepstrum. However, while we are confident that these other transforms can be computed with spiking neural networks, we limit our model to producing MFCC and delta-MFCC-like features in spiking neural networks.

4.1.3 Spectral analysis with auditory periphery models

While the analogy between the frontend of ASR systems and the human auditory system is usually just an analogy, many systems have experimented with more physiologically accurate auditory periphery models to replace the idealized spectral analysis step in the feature extraction pipeline described above.

Tchorz and Kollmeier (1999); Dimitriadis et al. (2005); Schluter et al. (2007); Shao et al. (2009) have separately proposed variants of MFCCs that use Gammatone filters to do the spectral analysis step in an ASR system. All four of these studies found that using Gammatone filters lowered word or phone error rates on recognition tasks in which noise was added to speech samples. Other studies has achieved similar results; see Stern and Morgan (2012) for a review.

While all of these systems have been applied successfully for noisy ASR tasks, they are not suitable for inclusion in Sermo in their current form. To my knowledge, none of the methods currently available produce spiking behavior that could be easily integrated with the rest of Sermo. Additionally, these networks compute functions of the filter output that may be difficult for neurons to implement. Tchorz and Kollmeier (1999), for example, implement short-term adaption through loops that perform a lowpass filter and a division. Division is, in general, difficult to approximate with spiking neural networks. A full neural implementation may be possible, but not trivial.

On the other hand, silicon cochlea models face the opposite problem. Silicon cochlea models are hardware implementations of auditory filter models designed to interact with other neuromorphic systems, or to be directly implanted in patients with hearing loss in order to partially recover the sense of hearing. Existing silicon cochlea models include Chan et al. (2007); Hamilton et al. (2008); Wen and Boahen (2009); Karuppuswamy et al. (2013) (see Liu and Delbruck 2010 for a review of silicon cochleas and other neuromorphic sensory systems). These systems produce spikes that emulate the spikes traveling down the auditory nerve, and since they are implemented in hardware, they run much faster than most software systems. However, because they produce spikes, they have not been used as the frontend to any ASR

systems, to my knowledge, as it is not straightforward to construct feature vectors out of spike trains.

4.2 Syllable sequencing

Currently, we are aware of few existing models explaining how syllables might be represented and temporally sequenced in the brain. However, there have been many attempts to solve similar problems and could be applicable to speech; specifically, models of song generation in songbirds and serial working memory.

A notable omission in this section is the WEAVER++ model previously discussed in Sections 2.2.2 and 3.1.3. While a neural implementation of WEAVER++ would be an important contribution to Sermo, WEAVER++ does not solve the syllable sequencing problem. At its lowest level, it produces syllable targets over time, but those syllable targets are not temporally coordinated with the voicing of other syllables. As such, a neural implementation of WEAVER++ would still require a syllable sequencing model like the ones described in this section.

4.2.1 Song generation in songbirds

The avian song system exhibits remarkable similarities to the human speech system (see Bolhuis et al. 2010). As such, models of song sequencing and generation in birds may be applicable to models of speech.

Troyer and Doupe (2000) presented a model of birdsong sensorimotor learning in which song sequencing is broken into two subproblems: syllable learning, in which the system learns to associate an ensemble of neurons with a top-down syllable representation, and sequence learning, in which the activation of an ensemble of neurons associated with a particular syllable is linked to the next syllable in the learned sequence. The end result of the model is that activating a particular ensemble of syllable-specific neurons begins a sequence of activations representing a particular syllable sequence associated with a target song.

The model uses a simple associative learning rule to learn sensory predictions of motor representations, and motor predictions of sensory representations. The sequence of syllables is produced by a motor representation making a prediction of the sound that will be produced, which is associated with the next motor action to be produced, which activates a sensory prediction, and so on.

While the learning method may be useful for future iterations of Sermo, the rest of the model is not suitable for Sermo because it does not allow for flexible temporal dynamics in

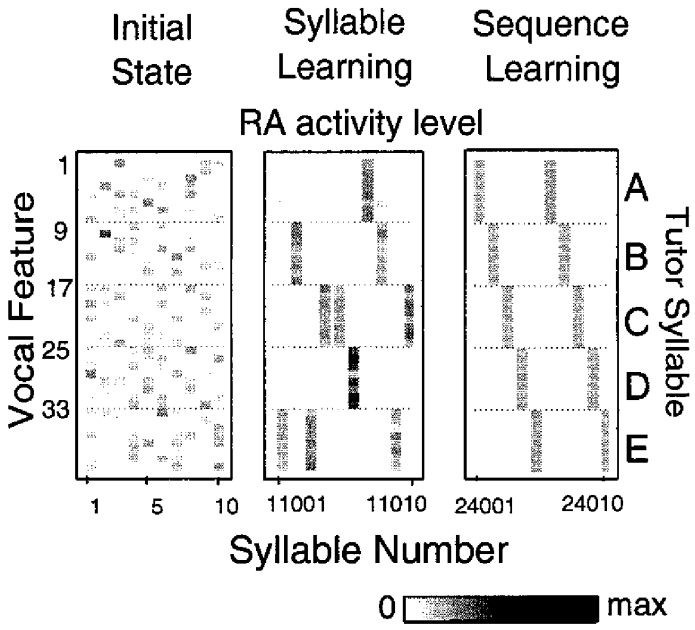


Figure 4.1: Illustration of model results from Troyer and Doupe (2000). Through learning, neurons in avian brain area RA activate sequentially, which is theorized to underlie song sequencing. In the first learning stage, neurons that are part of the same syllable are associated together (middle). In the second learning stage, sensorimotor predictions enable sequence learning such that song syllables sequentially activate in the correct order. Reproduced from Troyer and Doupe (2000).

the motor trajectories within and between syllables, which is one of the hallmarks of human speech. In Troyer and Doupe (2000), all syllables are assumed to take the same amount of time, and activate all of the neurons within the ensemble for the entirety of the syllable activation (see Figure 4.1). The time taken for each syllable is not easily modifiable since it is defined by the amount of time it takes for the motor action to activate the sensory prediction and switch to the next motor action, which is an intrinsic property of the synapse connecting the neuron models.

Drew and Abbott (2003) presented a model which learns to associate specific neural ensembles to particular sensory inputs, similar to Troyer and Doupe (2000), but also to sequences of sensory inputs. For example, given syllables A and B, some neurons would activate when syllable A is presented, some when syllable B is presented, and some only when syllable B is presented immediately following syllable A. Since the temporal sequence is coded in the

connections between these ensembles, Drew and Abbott hypothesized that sequences could be generated, rather than recognized, by delivering a generic timing pulse which emulated hearing all possible sensory inputs at once. The first time the pulse is delivered, ensembles sensitive to a single sound would activate. On the next time the pulse is delivered, ensembles representing sequences of length two would activate, and so on for longer sequences (see Figure 4.2). Ensembles representing sequences of length one are not active on the second timing pulse due to the intrinsic properties of the neurons in the ensembles; specifically, after spiking for the previous timing pulse, they enter a refractory period in which they become insensitive to further input.

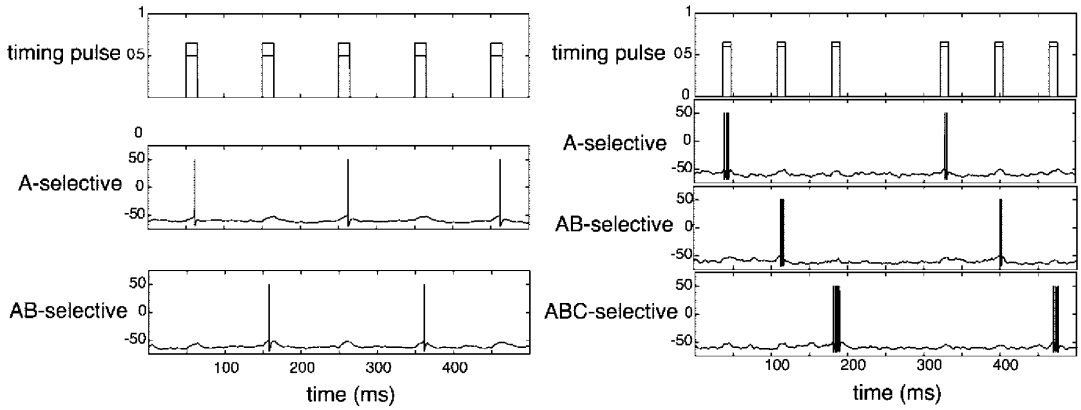


Figure 4.2: Illustration of model results from Drew and Abbott (2003). A timing pulse is used to control when the next element in the sequence should be activated. See text for more details. Reproduced from Drew and Abbott (2003).

While this approach is more temporally flexible than Troyer and Doupe (2000) because the timing pulse could arrive at any moment, it only moves the responsibility for flexible timing from the neurons involved in the sequence to whatever mechanism generates the timing pulse. In the paper, the timing pulse is provided by the experimenter; they note that the spacing of syllables can be controlled by varying the frequency of timing pulses, but do not provide any mechanism for generating pulses or varying the frequency of pulses. Additionally, the length of the sequence is extremely limited, both in terms of the total length of time and the number of syllables. The refractory period of each neuron is only long enough to be insensitive to the next timing pulse; therefore, for a sequence of length three, Drew and Abbott added inhibitory connections from sequence-specific ensembles to non-sequence-specific ensembles. The connectivity patterns required for sequences of longer lengths are not obvious, and depends on the refractory period, which can quickly become abnormally long. We therefore do not

think that this model is useful for Sermo.

While other models for songbird generation exist (e.g., Fee et al. 2004), none exhibit the temporal flexibility that is required for speech. As shown in Fee and Scharff (2010), cooling a part of the avian brain that projects to motor cortex results in slowing the trajectory in proportion to the temperature. Therefore, birds may not possess the ability to flexibly time their songs in the same manner that humans time speech, meaning we must look elsewhere for more temporally flexible models.

4.2.2 Serial working memory

We assume that a sequence of syllables is likely to be represented in a similar manner as sequences of other static representations. Therefore, a useful paradigm for studying how humans represent syllable sequences is to investigate serial working memory tasks; e.g., a subject is asked to remember a list of numbers, and then later recall that list, or elements from that list. Several models have been proposed to solve these tasks in biologically plausible ways. The most sophisticated such model is the ordinal serial encoding model (OSE; Choo, 2010), which has been incorporated into Spaun (Eliasmith et al., 2012).

The model takes inspiration from earlier serial working memory models, namely CADAM, TODAM, and TODAM2 (see Choo 2010 for more details), which are all based on vector symbolic architectures (see Section 5.4 for details on vector symbolic architectures). Unlike these models, Choo's model is able to remember and recall lists of up to seven items, and exhibits primacy and recency effects, meaning that items at the beginnings and ends of lists are more likely to be recalled. Choo's model is also implemented in a spiking neural network, making it applicable to Sermo. Interestingly, the OSE model does not exhibit the primacy and recency effects that are seen in humans unless the model is implemented in neurons; direct simulation of the differential equations behind the model show poor performance for the second item even though it should be easily remembered due to primacy.

The ordinal serial encoding model, then, is well suited for representing discrete sequences of syllables in Sermo, which typically have few elements if we assume that prosodification operates only slightly faster than speech production. However, the model is limited to discrete sequences, and therefore cannot be used for detailed trajectory generation. Additionally, while the model is temporally flexible, in that it can be queried for a list element at any time, there is a slight lag between when the next element is queried and when it has been recalled. Syllables, however, are voiced in quick succession, sometimes even blending into one another slightly. In Sermo, we will extend the OSE model to do syllable sequencing (see Section 6.2.2).

4.3 Trajectory generation

In trajectory generation, we must generate a sequence of production information corresponding to a syllable target. We will review two robust trajectory generation techniques, one primarily applied to speech (Task Dynamics), and one used for general motor control (REACH).

4.3.1 Task Dynamics

A line of research encompassing work by several investigators at Haskins Laboratories developed a set of techniques under the name Task Dynamics for generating temporal sequences of production information and using that information to drive an articulatory synthesizer (Saltzman and Munhall, 1989; Nam et al., 2004).

The key insight in Task Dynamics is to dissociate the task state from the motor trajectory along that state. By doing this, the dynamics of the task state can be considered separately from motor trajectories, while trajectories can be implemented as a function of the task state. In the initial publication of task dynamics (Saltzman and Kelso, 1987), two types of task dynamics are presented. Point attractor dynamics, modeled by critically damped mass-spring systems, are useful for one-time actions. Cyclic attractor dynamics, modeled by harmonic oscillators with a nonlinear escapement function, are useful for repetitive actions. Initially, these task dynamics were related to articulator trajectories for a two degree-of-freedom arm model by mapping the task space to body-centered coordinates, joint coordinates, and finally articulator positions, called the task network. In a subsequent publication, Saltzman and Munhall (1989) applied task dynamics to speech production using a simplified mapping, from task space to gesture space, and then to articular space, similar to that described in Sermo (see Section 3.1.4). Realistic articulatory trajectories were achieved by associating a task dynamic network with point attractor dynamics for each speech gesture. Each gesture defines the point which the task dynamic network is attracted to. The state of each gesture dynamical system influences one or more articulator positions. The final articulator trajectory is the combined effect of all of the gesture systems on all of the articulators.

Given a complete gesture trajectory for an utterance, the task dynamic approach generates articular position trajectories which can be synthesized by the CASY synthesizer (Iskarous et al., 2003). While the Task Dynamics model for inter-articulator coordination has not been modified significantly since its introduction in 1989, the same group has done significant work in automatically generating gesture scores with a similar approach. In this extension, instead of knowing *a priori* when each gesture should occur, the point attractors associated with each gesture are coupled to one another, such that the timing of each gesture is controlled by the

state of the gestures to which it is coupled (Saltzman and Byrd, 2000). Task dynamic gestural timing is able to capture precise timing in syllables with complex onsets and codas (Nam and Saltzman, 2003; Goldstein et al., 2006). Perhaps more importantly, it allows for the articulatory synthesizer CASY to accept orthographic text as input, which is converted to a gesture score by looking up a syllabification of the word in a database and using gestural coupling rules defined by linguists to create a system of coupled oscillators whose activities represent a gestural score (Nam et al., 2004; Goldstein et al., 2009).

In all, the task dynamic approach to inter-gestural and inter-articulator timing is the most temporally flexible syllable production system currently published (to my knowledge). The dynamical systems approach also makes it a natural fit for Sermo, as it is defined in continuous time, and should be readily implementable in spiking neural networks, though we are not aware of any neural implementations currently available.

4.3.2 REACH

The final model informing Sermo's trajectory generation model is the Recurrent Error-driven Adaptive Control Hierarchy (REACH) model (DeWolf, 2015). The REACH model is a general motor control model implemented in spiking neurons. It uses dynamic movement primitives (DMPs) to generate trajectories for the system to follow; these trajectories are mapped into motor space using operational space control, and unexpected changes in system dynamics are accounted for online using nonlinear adaptive control (Slotine and Li, 1987). The model is able to control a nonlinear three-link arm model in handwriting and reaching tasks, even when an unknown force field is applied to the end effector.

DMPs are a general method for generating motor trajectories, and are similar to Task Dynamics in many respects. Both define methods to generate trajectories for one-time and rhythmic actions. Both use point attractor dynamics for one-time actions and cyclic attractor dynamics for rhythmic actions. Both dissociate the temporal dynamics of the task from trajectories in motor space, allowing them to advance the system state at variable rates, and compute the trajectory as a function of the system state.

The primary difference between DMPs and Task Dynamics is that Task Dynamics generates the trajectory as a function of the system state, while DMPs generates the trajectory as the system state plus a separate function of another dynamical system. Dissociating the system state from the nonlinear task-specific function makes DMPs more flexible and general. Another important difference for Sermo is that DMPs have been implemented in spiking neural networks successfully (DeWolf, 2015). We will explain DMPs in more detail in Section 5.2 and present a model using rhythmic DMPs for syllable production in Section 6.2.2.

4.4 Syllable recognition

Syllable recognition, in general, is a task that can be solved by most ASR systems using a labeled acoustic data set. In the sensorimotor integration system in Sermo, however, we aim to recognize syllables based only on production information which is decoded from acoustic information. It should be noted that we do not expect this system to perform perfectly, as the dominant speech decoding system will be linguistic; however, humans are nevertheless able to recognize infrequent syllables and voice them. The ability to differentiate between frequent and infrequent syllables may depend on whether they can be classified on the basis of production information.

To my knowledge, the only attempt to recognize speech based solely on production information is Mitra et al. (2014). Mitra et al. were able to decode continuous production information from auditory information using MFCCs and Gammatone filter-based cepstral features (Mitra et al., 2012), and were able to use a combined feature vector consisting of MFCCs and production information to lower word error rates in various noisy environments. However, word error rates when using only continuous production information as the feature vector were high (~70% with no noise).

The apparent difficulty in recognizing words based on production information alone could be due to several factors. For one, the details of the decoding mechanism in Mitra et al. (2014) are not clear. The authors note that they use a deep neural network with as many as six hidden layers; however, the choice of neural activation function, optimization procedure, and many other hyperparameters can affect how well the network learns. In particular, it does not seem as though the network has recurrent connections, which are commonly used in current state-of-the-art ASR systems.

Alternatively, the statistical approach used in most ASR systems may not be well suited to trajectories of production information. In order to broaden our search for other types of techniques, we investigated general solutions for trajectory classification which are used in applications such as gesture recognition and automatic video analysis.

4.4.1 Trajectory classification

Unsurprisingly, many of the existing solutions for trajectory classification are based on generative statistical models, particularly Hidden Markov Models (HMMs), as has been dominant in speech recognition (see Mlıch and Chmelar 2008; Nascimento et al. 2010 and Mitra and Acharya 2007; Weinland et al. 2011; Rautaray and Agrawal 2015 for reviews). Given the failure

of the statistical approach in Mitra et al. (2014), we focus here on systems that recognize trajectories through tracking the trajectory either in comparison to some known template, or as the state in a dynamical system. We identify three such systems that provide inspiration for the trajectory classification model described in Section 6.3.

Kılıboz and Güdükbay (2015) proposed a gesture recognition system for 2D trajectories using a finite state machine approach. Finite state machines are composed of discrete states and a set of state-specific functions that transition between states. During a learning phase, a target trajectory is played several times, and one or more finite state machines are constructed such that the target trajectory causes the state machine to transition to a final accepting state. During recognition, the continuous input trajectory is presented to all finite state machines in the system; the first to reach the accepting state is recognized. The system attains a 73% accuracy rate in a real-world user study. As the system operates continuously online, part of the system is applicable to Sermo; however, the discrete state space and gesture sequence representation cannot be easily adapted to speech. Each timestep in the gesture trajectory is represented by a string denoting whether the end effector has moved significantly in the x or y direction since the last frame; the overall trajectory is represented by a regular expression generalized from the strings seen in the learning phase. It is not clear that this representation scheme would scale to n-dimensional spaces, as is required for production information trajectories.

Quiroga and Corbalán (2013) proposed a system called competitive neural classifiers (CNC) that can recognize hand gestures using small training sets (three examples per gesture). CNC uses a collection of sub-classifiers that compete in order to collectively classify the overall trajectory. The trajectory is segmented into a set of subtrajectories, each of which is evaluated by a sub-classifier neural network whose output neural activations represent the probability that the subtrajectory is produced by the gesture associated with that output neuron. The overall classification aggregates the results across of all sub-classifiers, producing the correct classification in 98% of test cases. However, the system's impressive results are partly due to an elaborate preprocessing step in which a recorded trajectory is normalized and resampled such that the actual trajectory used as input has a fixed number of sample points uniformly distributed over the total length of the trajectory (see Figure 4.3). This type of preprocessing requires knowledge of the whole trajectory, and therefore could not be implemented in an online manner. Additionally, the operations done on the neural network outputs are difficult to implement with spiking neural networks (e.g., division, argmax), making this system unsuitable for use in Sermo.

Finally, Caramiaux et al. (2014) presented an algorithm called the Gesture Variation Follower (GVF), based on an online HMM-based trajectory tracking technique called Gesture Following (GF) (Bevilacqua et al., 2010, 2011). The goal of the algorithm is to match an input gesture

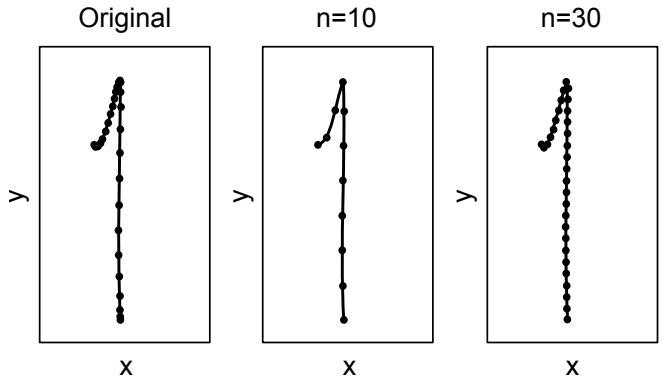


Figure 4.3: Example of the resampling procedure in Quiroga and Corbalán (2013). The original trajectory (left) has more sample points at the beginning and end of the trajectory. The resampled trajectories (middle, right) have sample points distributed to maximize the probability of a successful classification, which cannot be done in a continuous online system. Adapted from Quiroga and Corbalán (2013).

sequence with a set of predefined template gestures. Unlike the HMM-based GF algorithm, GVF views the trajectory as a dynamical system, and uses a particle filtering algorithm to learn a set of weights that denote the importance of each randomly generated particle (i.e., dynamical system state) to overall recognition accuracy.

The overall flow of the algorithm is as follows. First, a set of predefined example trajectories are defined in the system. Second, a set of particles are randomly generated in an n-dimensional space with a uniform distribution, and a set of weights are initialized with each particle weighted equally. In the main loop of the algorithm, a random sample is drawn according to each particle's position in state space, and the weights associated with each particle are updated based on the distance between the particle's sample and the observed input. If too many particles have small weights associated with them, then particles are resampled based on their current weights.

What differentiates this algorithm from other particle filtering algorithms is the structure of the state space. Briefly, the state space that each particle is in encodes the target gesture that this particle is associated with, and a probability distribution over the phase of the trajectory (i.e., how far along the trajectory we currently are) and the speed of the trajectory (i.e., how far along the trajectory do we expect to be on the next timestep). The state evolves over time according to predefined state transition functions, and on each timestep, each particle emits an observation according to a possibly nonlinear function.

A useful analogy for this algorithm is that it implements an online version of an HMM. Like an HMM, it maintains some internal representation that can be used to estimate the probability of a particular sequence by forming a prediction of the next observation. The system is queried by providing an observation, which changes the internal representation such that the next observation is processed in the context of the sequence of past observations.

GVF has been used successfully in music generation systems in which the music sample, volume, and speed are controlled by gestures that are tracked by the GVF algorithm. It achieves over 98% accuracy in a 2D gesture recognition task with 16 possible gestures, and was employed successfully in a 3D hand gesture user study with 10 participants (Caramiaux et al., 2014).

While GVF is one of the most promising algorithms for doing trajectory recognition in Sermo, it has two main limitations in its current formulation. First, while the classifier can be used online, it is formulated in discrete time; this weakness should be possible to overcome, however, as continuous time particle filtering has been done in the past (Ng et al., 2005). Second, while the dynamical system at the core of GVF should be implementable in spiking neurons, the resampling procedure is likely not. On each step of the algorithm, each particle represents a probability distribution over the system state, which is sampled (sampling has been shown to be possible with spiking neurons; see Buesing et al. 2011). However, when the importance weights of a sufficient number of particles are below a particular threshold, a new set of particles are randomly generated to replace those with lower importance weights. This procedure would translate to a significant reorganization of the neurons implementing the sampling procedure, which I believe would not have evolved if procedures that did not require reorganization existed.

I will propose a trajectory classification technique in Section 6.3 that shares many of the positive characteristics of the GVF algorithm, but can be implemented in a spiking neural network. Like GVF, it is also based on the idea of inferring internal dynamical system state based on observations; however, it performs the inference in a manner that we relate to DMPs (see Section 6.3.1).

Chapter 5

Methods

The three models implemented in this thesis employ methods for auditory processing and trajectory generation online and in continuous time, meeting Sermo's criteria outlined in Section 3.1. Additionally, we construct spiking neuron models that implement or interact with those methods using the Neural Engineering Framework and Semantic Pointer Architecture.

5.1 Auditory periphery models

In order to extract a feature vector from an audio signal in a biologically plausible manner, we use models of the auditory periphery to perform a continuous spectral analysis of the audio signal. We compare five existing auditory periphery model implementations in the model described in Section 6.1.

5.1.1 Gammatone filter

The Gammatone filter was proposed in the 1970s (see Johannesma 1972; De Boer 1975; Patterson 1976) and quickly became the most widely used auditory filter, as it emulates the basic function of the inner ear, yet can be implemented efficiently on general purpose computers and digital signal processors. It captures many of the psychoacoustical phenomena discussed in Section 2.1.2, but does not capture some basic phenomena; for example, the Gammatone auditory filter is symmetric. However, it can process audio online and in continuous time, making it applicable to Sermo.

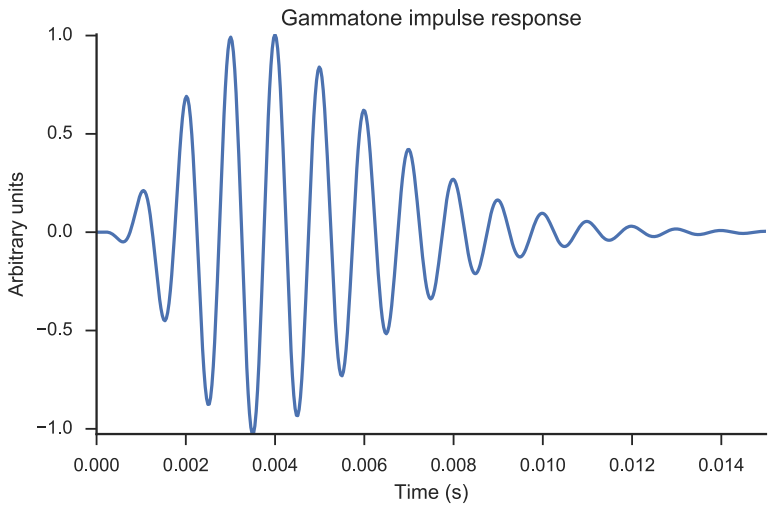


Figure 5.1: Impulse response of the Gammatone filter. It is clear from visual inspection that the Gammatone filter is a cosine wave convolved with a Gamma function. See text for equation.

As described in Patterson (1976), the Gammatone filter has an impulse response of the form

$$\mathrm{IR}(t) = t^{n-1}\cos(2\pi f t)\, e^{-2\pi b \mathrm{ERB}(f) t},$$

where n is the order of the filter, b is a parameter determining the filter bandwidth, and $\mathrm{ERB}(f)$ is the equivalent rectangular bandwidth of the filter centered at frequency f ($\mathrm{ERB}(f) = 24.7 + 0.108f$; see Figure 5.1). Intuitively, the impulse response is the convolution of a carrier signal (a cosine wave) and an envelope (a Gamma function, hence the name Gammatone). Fourth order Gammatone filters are the most common (Patterson et al., 1992).

Slaney (1993) proposed an efficient digital filter implementation of the Gammatone filter. The filter is an eighth order digital filter implemented as a cascade of four second order infinite impulse response (IIR) filters that can be realized with Direct Form II structure. The

Z-transforms of the second order filters are

$$H_1(z) = \frac{-2Tz^2 + z\left(\frac{2T\cos(2f\pi T)}{e^{BT}} + \frac{2\sqrt{3+2^{1.5}}T\sin(2f\pi T)}{e^{BT}}\right)}{\frac{-2}{e^{2BT}} - 2z^2 + \frac{4z\cos(2f\pi T)}{e^{BT}}}$$

$$H_2(z) = \frac{-2Tz^2 + z\left(\frac{2T\cos(2f\pi T)}{e^{BT}} - \frac{2\sqrt{3+2^{1.5}}T\sin(2f\pi T)}{e^{BT}}\right)}{\frac{-2}{e^{2BT}} - 2z^2 + \frac{4z\cos(2f\pi T)}{e^{BT}}}$$

$$H_3(z) = \frac{-2Tz^2 + z\left(\frac{2T\cos(2f\pi T)}{e^{BT}} + \frac{2\sqrt{3-2^{1.5}}T\sin(2f\pi T)}{e^{BT}}\right)}{\frac{-2}{e^{2BT}} - 2z^2 + \frac{4z\cos(2f\pi T)}{e^{BT}}}$$

$$H_4(z) = \frac{-2Tz^2 + z\left(\frac{2T\cos(2f\pi T)}{e^{BT}} - \frac{2\sqrt{3-2^{1.5}}T\sin(2f\pi T)}{e^{BT}}\right)}{\frac{-2}{e^{2BT}} - 2z^2 + \frac{4z\cos(2f\pi T)}{e^{BT}}},$$

where T is is the sampling interval, f is the characteristic frequency of the filter, and B controls the bandwidth of the filter ($B = 2\pi b\text{ERB}(f)$). See Slaney (1993) for the derivation of the Z-transforms.

Figure 5.2 shows the output of the auditory periphery model using the Gammatone filter for white noise, tone ramp, and speech inputs.

5.1.2 Log Gammachirp filter

A primary weakness of the Gammatone filter is that it responds symmetrically to power in frequencies above and below the characteristic frequency of the filter. In empirically determined auditory filters, portions of the basilar membrane are more sensitive to frequencies below the characteristic frequency, and have a sharp dropoff in sensitivity for power in frequencies above the characteristic frequency. The family of Gammachirp filters are infinite impulse response (IIR) filters that remain computationally efficient, like the Gammatone filter, but have an asymmetric frequency response. They are based on spectral masking experiments (see Section 2.1.2) in which the listener has to detect a brief sinusoidal signal among "notched" white noise, in which the power spectrum of the noise at frequencies just above and just below the frequency of the sinusoid are set to zero (Patterson, 1976). The Log Gammachirp filter, proposed by Unoki et al. (2001) is an improvement over the original Gammachirp filter (Irino and Patterson, 1997) in that it is more numerically stable.

The impulse response of the Log Gammachirp filter is

$$\text{IR}(t) = t^3\cos(2\pi(f\,t + c\ln(t)))\,e^{-2\pi b\text{ERB}(f)t},$$

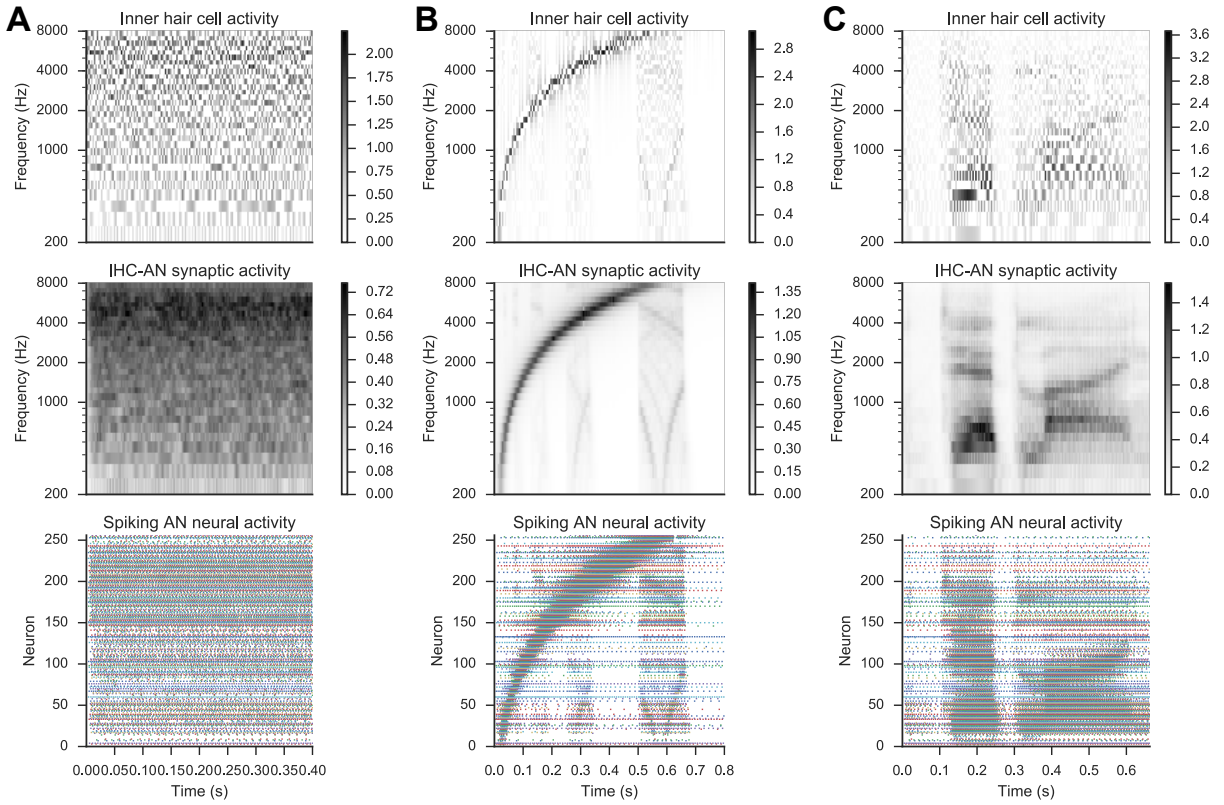


Figure 5.2: Auditory periphery model using the Gammatone filter. The top row plots the rectified and compressed output of the filter, which we take to be the activity of an inner hair cell (IHC). The middle row plots the activity across the IHC-auditory nerve (AN) synapse. The bottom row plots the spiking activity of neurons that project down the auditory nerve. The model is fed (A) white noise, (B) tone ramp, and (C) speech input.

where c is a "chirp" parameter that affects the asymmetry of the auditory filter; typically, c is set to a negative value, raising the response to lower frequency sounds and attenuating the response to higher frequency sounds. The Gammachirp impulse response (see Figure 5.4) is similar to that of the Gammatone, except that the frequency of the cosine wave carrier signal increases (for $c < 0$) or decreases (for $c > 0$) throughout the impulse response. The ability for the Gammachirp filter to change the frequency it is sensitive to (i.e., to perform "frequency sweeps" or "frequency glides") causes the asymmetry.

Gammachirp filters are closely related to Gammatone filters. In fact, the amplitude spec-

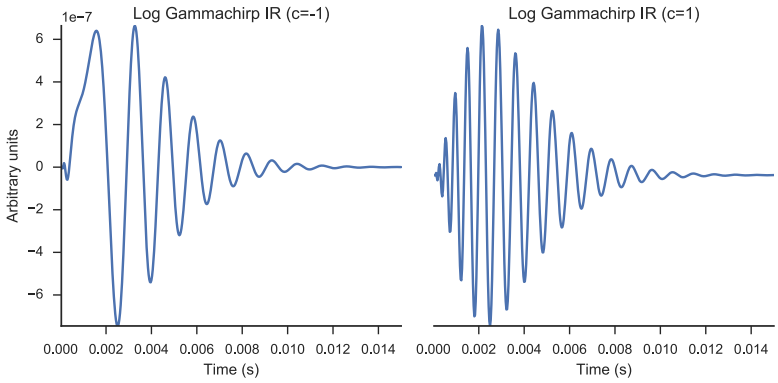


Figure 5.3: Impulse response of the Log Gammachirp filter. Increasing frequency can be seen when $c = -1$ (left); decreasing frequency can be seen when $c = 1$ (right). See text for equation.

trum of the Gammachirp is

$$|G_C(f)| = a_0 \cdot |G_T(f)| \cdot |H_A(f)|,$$

where a_0 is a scaling factor, $|G_T(f)|$ is the amplitude spectrum of the Gammatone filter, and $|H_A(f)|$ is the asymmetric function that depends on the parameter c. The final filter is designed by determining an asymmetric filter $H_C(f)$ that simulates the asymmetric function $H_A(f)$ (see Unoki et al. 2001 for details). The Log Gammachirp filter improves upon the original Gammachirp filter by identifying parameter values that yield discrepancies between $H_A(f)$ and $H_C(f)$, and using an alternative $H_C(f)$ with a parameter that is optimized to minimize the error between $H_C(f)$ and $H_A(f)$.

Figure 5.4 shows the output of the auditory periphery model using the Log Gammachirp filter for white noise, tone ramp, and speech inputs.

5.1.3 Dual resonance nonlinear filter

The filters discussed to this point are linear filters. However, as discussed in Section 2.1.4, there are many nonlinearities in the human auditory system, including at the level of the auditory periphery.

The Dual Resonance Nonlinear (DRNL) filter, proposed in Lopez-Poveda and Meddis (2001) based on earlier work by Meddis et al. (2001), consists of two parallel pathways that are summed together (see Figure 5.5). The linear path (top, Figure 5.5) scales the input by

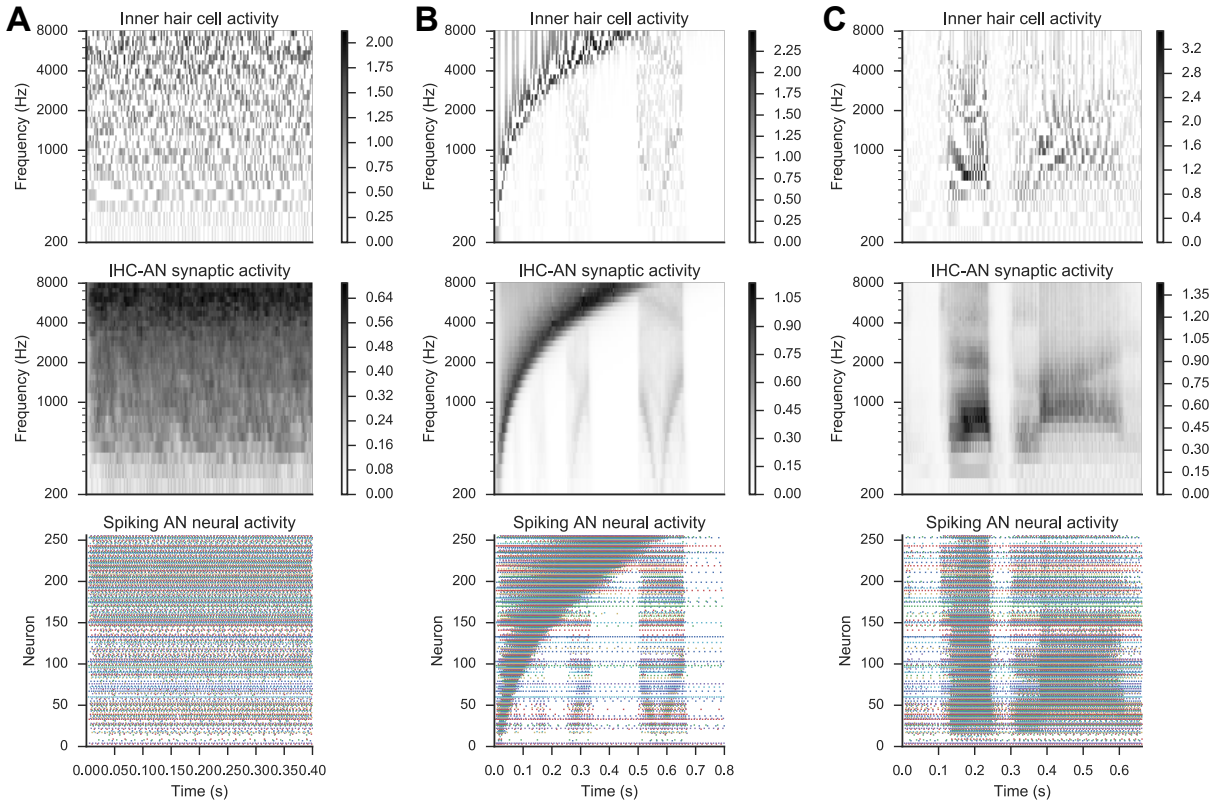


Figure 5.4: Auditory periphery model using the Log Gammachirp filter. The top row plots the rectified and compressed output of the filter, which we take to be the activity of an inner hair cell (IHC). The middle row plots the activity across the IHC-auditory nerve (AN) synapse. The bottom row plots the spiking activity of neurons that project down the auditory nerve. The model is fed (A) white noise, (B) tone ramp, and (C) speech input.

some gain factor, filters the signal through two or three first-order Gammatone filters, and then through a cascade of four second-order low-pass filters. The non-linear path (bottom, Figure 5.5) filters the signal through three first-order Gammatone filters, a nonlinear gain, and three more first-order Gammatone filters. The filters in the nonlinear path have the same frequency, which is the desired characteristic frequency of the filter, while the filters in the linear path have parameters fit to human data. The result of the linear and nonlinear paths are summed to yield the final result of the filter, which models the velocity of basilar membrane deflection.

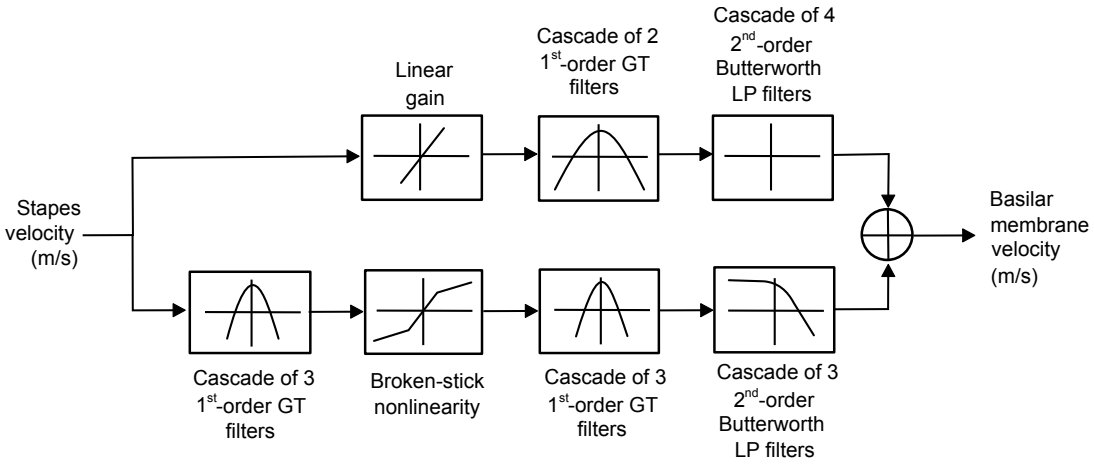


Figure 5.5: Signal processing pathways of the DRNL filter. The linear path is on the top; the nonlinear path is on the bottom. See text for more details. Adapted from Lopez-Poveda and Meddis (2001).

The nonlinear gain in the nonlinear path is a "broken stick" nonlinearity; i.e.,

$$y(t) = \text{sign}(x(t)) \min(a|x(t)|, b|x(t)|^c),$$

where $x(t)$ is the input signal, $y(t)$ is the output signal, and a, b and c are parameters. Like the parameters of the linear path's filters, these parameters are fit to human data. Detail on the fitting procedure and the digital filter implementation of the filter can be found in Lopez-Poveda and Meddis (2001).

Due to its complexity, the DRNL impulse response does not have a simple closed form. However, Figure 5.6 shows the output of the auditory periphery model using the DRNL filter for white noise, tone ramp, and speech inputs.

5.1.4 Dynamic compressive Gammachirp model

While the DRNL filter can be thought of as an nonlinear extension of Gammatone filters, the Dynamic Compressive Gammachirp model (DCGC) (Irino and Patterson, 2006) is a nonlinear extension of Gammachirp filters. It also consists of two pathways, which, unlike the DRNL filter, interact nonlinearly.

Figure 5.7 gives the overall structure of the filter. In the control (level estimation) pathway, the signal is filtered by a bank of passive Gammachirp filters, and then highpass filters. The

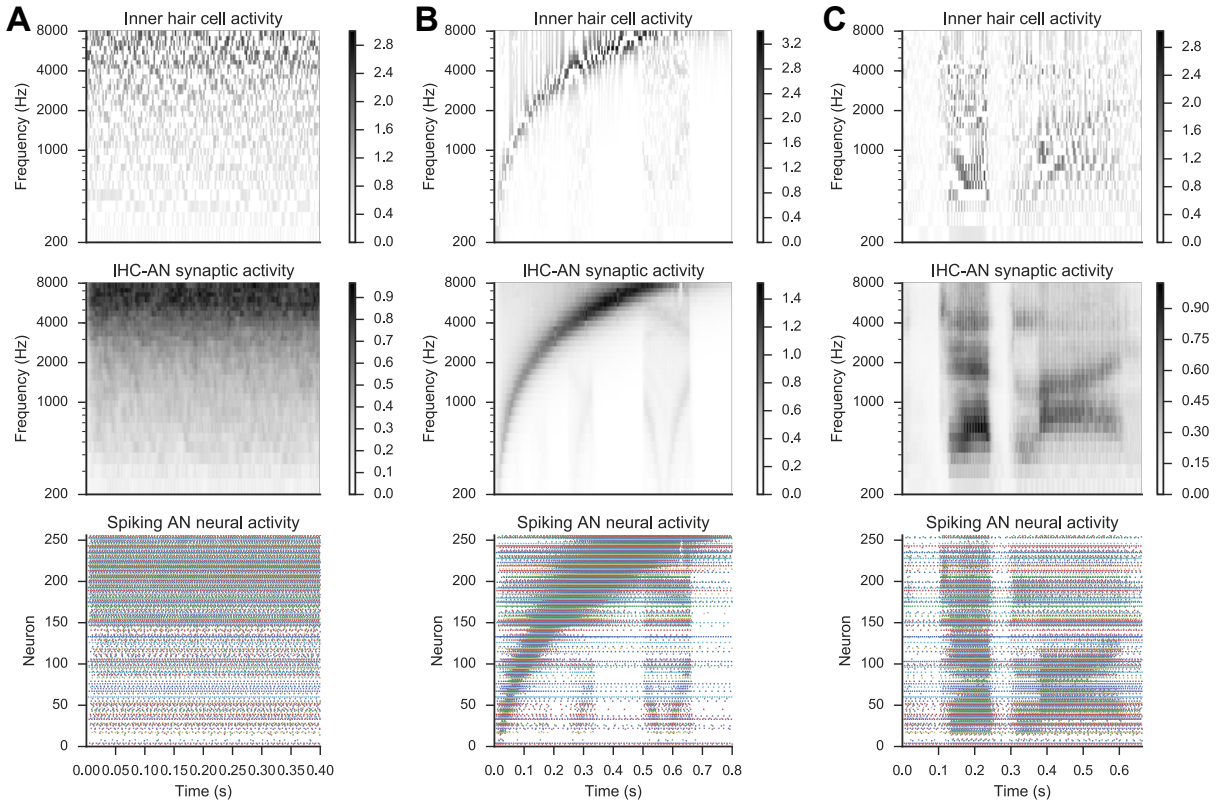


Figure 5.6: Auditory periphery model using the DRNL filter. The top row plots the rectified and compressed output of the filter, which we take to be the activity of an inner hair cell (IHC). The middle row plots the activity across the IHC-auditory nerve (AN) synapse. The bottom row plots the spiking activity of neurons that project down the auditory nerve. The model is fed (A) white noise, (B) tone ramp, and (C) speech input.

signal pathway also begins with a bank of Gammachirp filters, and then highpass filters, but the highpass filters in the signal pathway have variable cutoff frequencies depending on the output of the control pathway.

The filter simulates several psychoacoustical effects from masking experiments, and can be inverted in order to synthesize audio from transformed versions of filter output (Irino and Patterson, 2006). Like the DRNL filter, it is too complicated to express the impulse response in a simple closed form, but responses to input signals are shown in Figure 5.8.

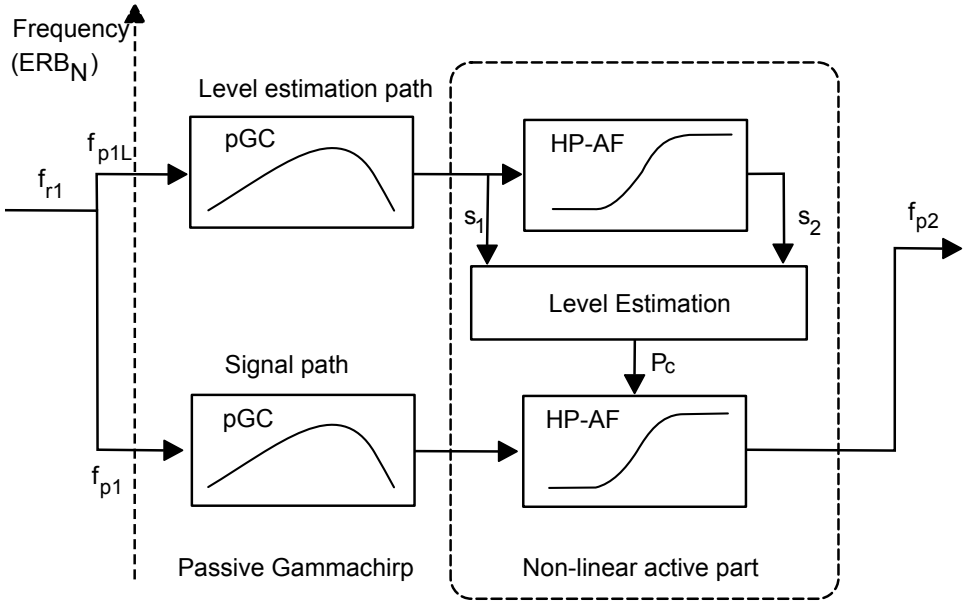


Figure 5.7: Signal processing pathways of the DCGC filter. See text for more details. Adapted from Irino and Patterson (2006).

5.1.5 Full auditory periphery modeling

The filters described up to this point emulate the deflection of the basilar membrane, given some audio signal. These filters are an important part of a full auditory periphery model, which should also include a method for transducing the basilar membrane deflections into inner hair cell activity, and a method for generating action potentials from the inner hair cell activity, mimicking the function of spiral ganglion cells.

For transducing the inner hair cell activity, we rectify and compress the output of the auditory filter model to account for two effects. First, the basilar membrane may deflect positively or negatively (as can be seen in the impulse response in Figure 5.1), but only positive deflections result in inner hair cell activity. Second, the amount of inner hair cell activity does not linearly increase with membrane deflection; the activity is "compressed" as the deflection increases. These effects are modeled by

$$f(x) = \begin{cases} x^{1/3} & \text{if } x \geq 0 \\ 0 & \text{if } x < 0. \end{cases} \tag{5.1}$$

The result of equation (5.1) is interpreted as the inner hair cell activity, which is the input to

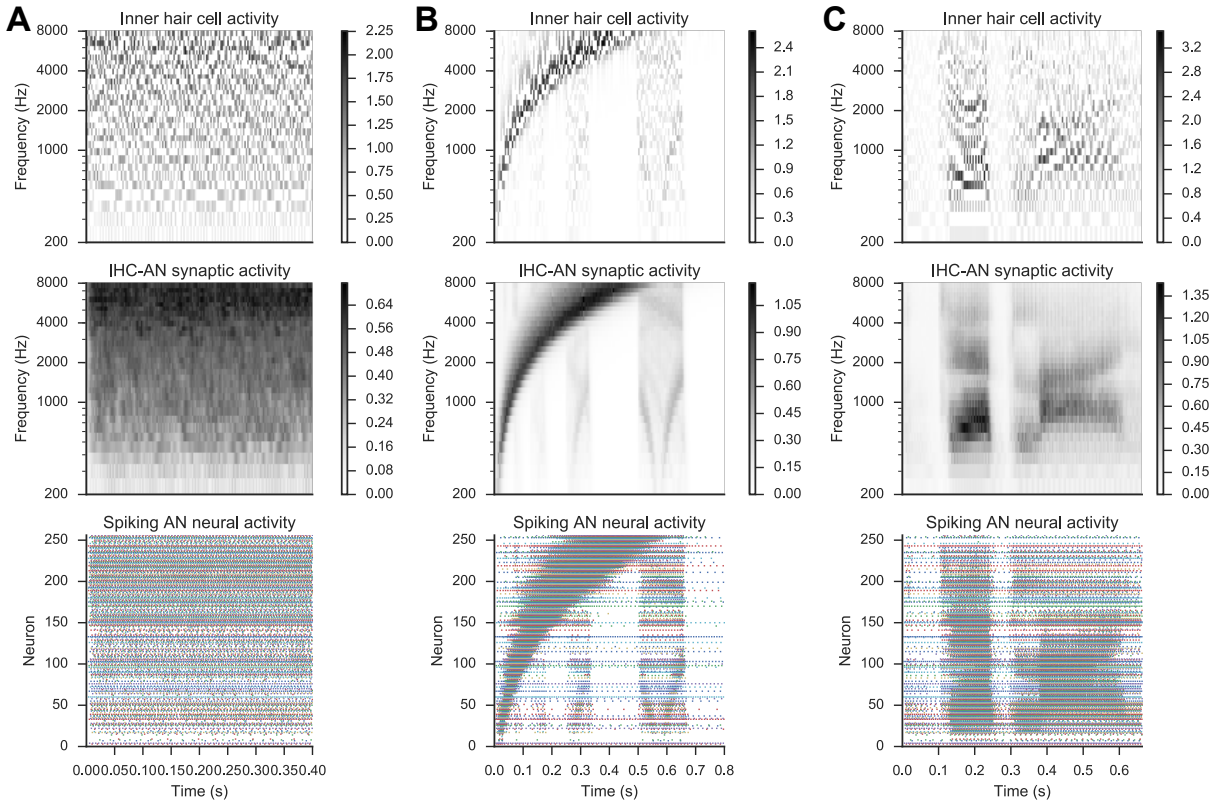


Figure 5.8: Auditory periphery model using the DCGC filter. The top row plots the rectified and compressed output of the filter, which we take to be the activity of an inner hair cell (IHC). The middle row plots the activity across the IHC-auditory nerve (AN) synapse. The bottom row plots the spiking activity of neurons that project down the auditory nerve. The model is fed (A) white noise, (B) tone ramp, and (C) speech input.

a simulated neuron. Synapses and neurons are simulated with Nengo; see Section 5.5 for details. Figures 5.2, 5.4, 5.6, and 5.8, include the compression and neural model for illustrative purposes.

5.1.6 Tan & Carney model

Finally, we also test the auditory periphery model presented in Tan and Carney (2003) which takes into account the full auditory periphery pathway.

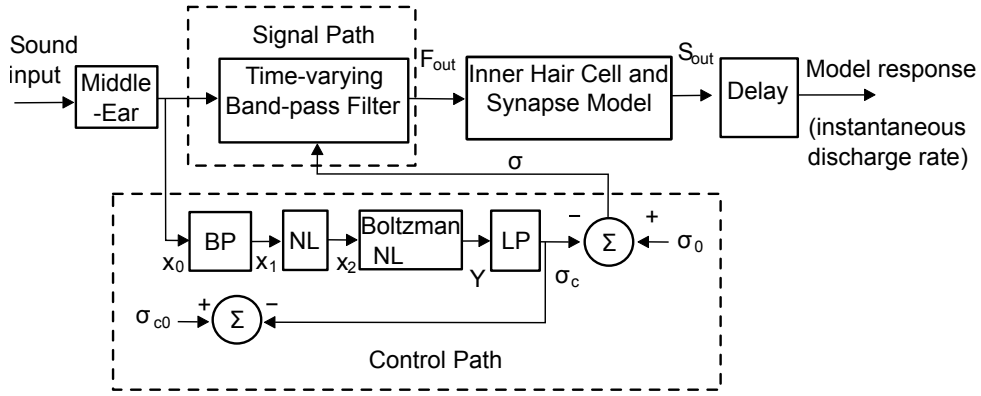


Figure 5.9: Signal processing pathways of the Tan Carney model. See text for more details. Adapted from Tan and Carney (2003).

The model, depicted in Figure 5.9, consists of a middle ear model, an auditory filter consisting of a control path and signal path, and an inner hair cell and synapse model. The output of the model is the instantaneous firing rate of spiral ganglion cells connected to inner hair cells sensitive to a particular characteristic frequency.

The signal path in the model consists of a linear bandpass filter with two eighth-order poles and one fourth-order pole, their complex conjugates, and a tenth-order zero on the real axis. Despite this complexity, Tan and Carney state that these poles and zero are the minimum number required to capture physiological effects like frequency glides and sharp frequency response curves.

The bandwidth and gain of the linear filter in the signal path is controlled by the output of the control path, which can be thought of as implementing the compression mentioned in the previous section. The control path cascade consists of a nonlinear wideband filter, a symmetric nonlinear function, an asymmetric nonlinear Boltzmann function, and a second-order lowpass filter with a cutoff frequency of 800 Hz. As mentioned above, the output of the control path controls the bandwidth of the signal path, but it also is fed back into the start of the control path to control the bandwidth of the initial wideband filter.

In the full model, the output of the signal path is used to simulate an inner hair cell and the synapse between an inner hair cell and a spiral ganglion cell (Zhang et al., 2001). The IHC model consists of a log-sigmoid transfer function and a seventh-order lowpass filter. The synapse model is a three-store diffusion model (Carney, 1993), which generates the instantaneous rate of a spiral ganglion cell. However, we will instead use a synapse and neuron model from Nengo to reduce computational complexity and make the five models more consistent.

Figure 5.10 shows the output of the auditory periphery model using the Tan Carney model (with Nengo neurons and synapses) for white noise, tone ramp, and speech inputs.

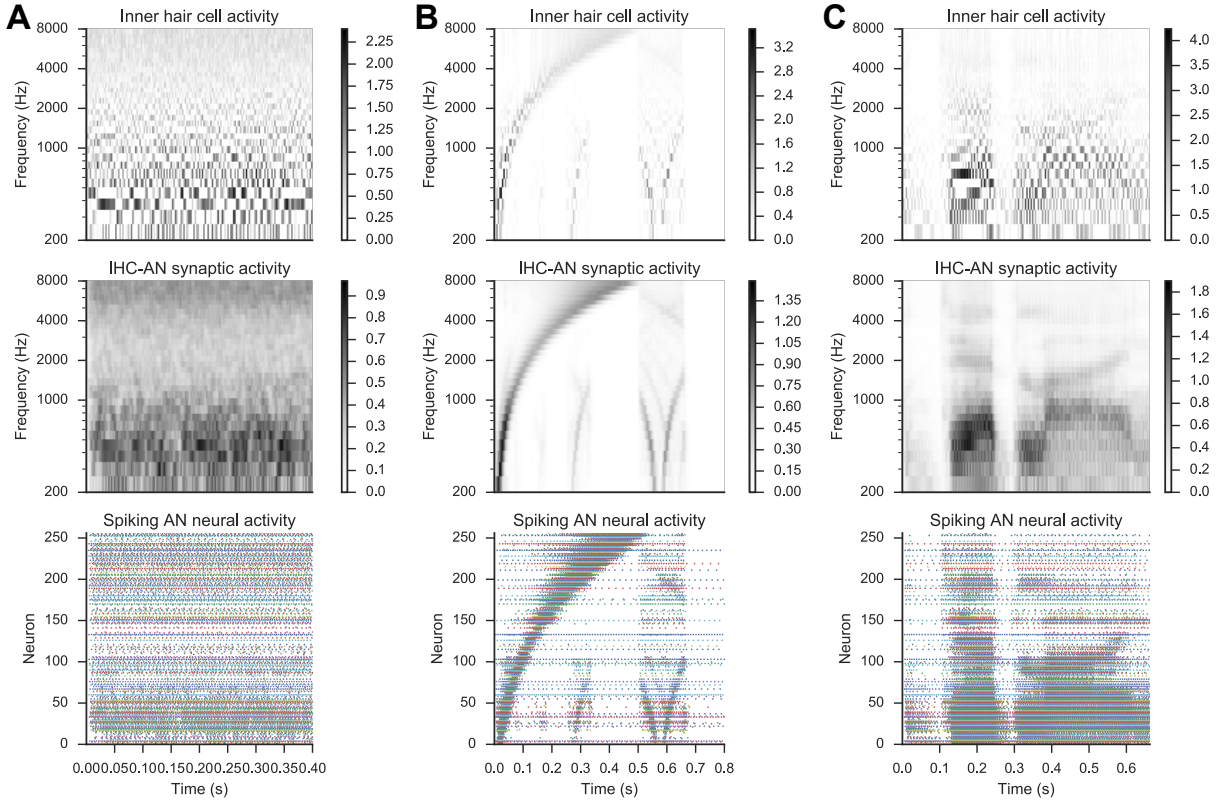


Figure 5.10: Auditory periphery model using the Tan Carney model. The top row plots the rectified output of the filter, which we take to be the activity of an inner hair cell (IHC). The middle row plots the activity across the IHC-auditory nerve (AN) synapse. The bottom row plots the spiking activity of neurons that project down the auditory nerve. The model is fed (A) white noise, (B) tone ramp, and (C) speech input.

5.1.7 Brian Hears software

The Brian Hears software (Fontaine et al., 2011) provides implementations of the five models described above. Brian Hears is part of the Brian package, which is a general purpose neural simulator capable of simulating spiking neural networks with relatively high precision but slow speed (Goodman and Brette, 2008; see Bekolay et al. 2013a for benchmark results). Due in

part to its slow speed, we use Brian only for the auditory periphery model implementations provided in the Brian Hears subpackage, and use Nengo (described below) for all other model implementations.

5.2 Dynamic movement primitives

The dynamic movement primitive (DMP) is a method for planning and controlling trajectories that requires little parameter tuning and is stable for trajectories with many degrees of freedom. DMPs were originally proposed by Schaal et al. (2005); Schaal (2006), reformulated and refined by Ijspeert et al. (2007), and simplified for use in spiking neural networks by DeWolf (2015). Here, we summarize the essential aspects of the DMP framework as described by DeWolf (2015), but invite interested readers to the previously cited works and Vijayakumar et al. (2005); Ijspeert et al. (2013) for further details and extensions.

DMPs use dynamical systems with specified, stable behavior to generate trajectories with more complex behavior. The main insight in the DMP framework is to define two separate systems, a point attractor, and a "canonical system." The point attractor pushes the system state to a goal, g, with dynamics

$$\tau \ddot{y} = \alpha_y(\beta_y(g - y) - \dot{y}) + f(x, g), \tag{5.2}$$

where y is the system state, $\dot{y}$ is the system velocity, and α_y and β_y are gain terms. τ is a scaling term that enables the system to operate at different timescales, and also appears in the definition of the canonical system.

$f(x, g)$ is a function of the canonical system called the "forcing function." In the discrete, one-time action case, the canonical system has state x that evolves with dynamics

$$\tau \dot{x} = \begin{cases} 1 & \text{if } x < 1 \\ 0 & \text{if } x \geq 1, \end{cases} \tag{5.3}$$

Typically, the initial value of x is set to 0, so the discrete forcing function is defined over the range $[0, 1]$. In the rhythmic case, the state evolves with two-dimensional oscillator dynamics

$$\begin{aligned} \tau \dot{x_1} &= -2\pi x_2 \\ \tau \dot{x_2} &= 2\pi x_1. \end{aligned} \tag{5.4}$$

The forcing function $f(x, g)$ is not directly computed; instead, it is computed as the weighted sum over some basis functions ψ_i (see Figure 5.11). In the discrete case, the basis functions are evaluated for the system state x directly. In the rhythmic case, the basis

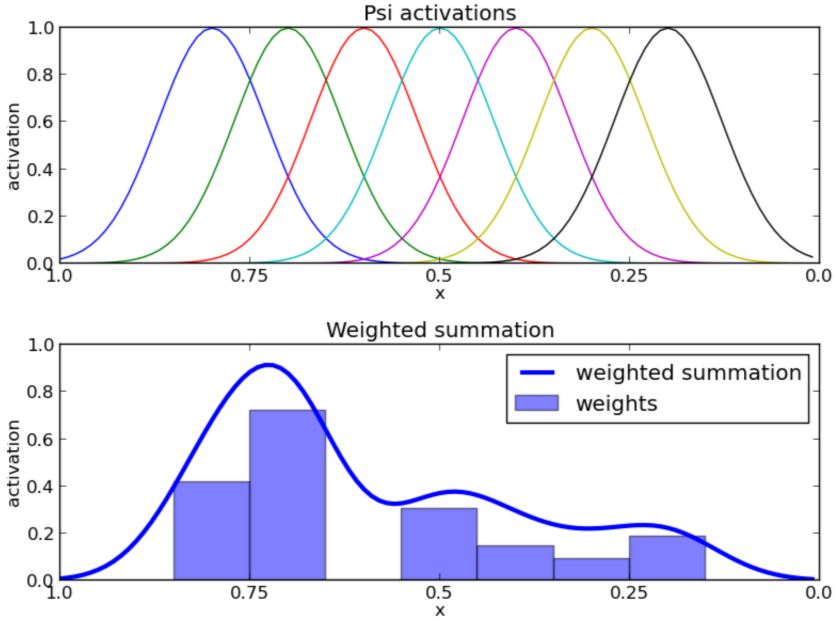


Figure 5.11: DMP forcing function, computed with evenly spaced Gaussians as basis functions, ψ. Reproduced from DeWolf (2015).

functions are evaluated for $\frac{1}{2\pi}\tan^{-1}\left(\frac{x_2}{x_1}\right)+\frac{1}{2}$, which maps the 2D oscillator state to the range $[0,1]$, allowing the forcing function to be defined as a function of x over the range $[0,1]$ in both cases. The goal g is used to scale the system state x depending on the initial distance to the goal. The final equation is

$$f(x,g)=\frac{\sum_{i=1}^{N}\psi_i w_i}{\sum_{i=1}^{N}\psi_i}x(g-y_0), \tag{5.5}$$

where w_i is the weight associated with basis function ψ_i and y_0 is the initial position of the system state.

Typically, Gaussian functions tiling the range of x with some overlap are used as the basis functions for the forcing function (as are used in Figure 5.11). DeWolf (2015) showed that the response curves of spiking neurons could be used as the basis functions instead (as is done for general function approximation in the Neural Engineering Framework; see Section 5.3).

5.3 Neural Engineering Framework (NEF)

The Neural Engineering Framework (NEF; Eliasmith and Anderson, 2004) provides a unified method of representing and transforming information in spiking neural networks such that they can implement arbitrary dynamical systems. The NEF is the most prominently used tool in this thesis, and greatly influenced the development of the conceptual Sermo model through the constraints and assumption inherent in the NEF.

The NEF defines three principles (representation, transformation, and dynamics) that can be used to build large-scale networks using any spiking or non-spiking neuron model.

5.3.1 Representation

The representation scheme in the NEF is based on population coding, which was first proposed by Georgopoulos et al. (1986) based on experiments in monkey cortex. Population coding is a type of distributed representation, theorizing that an ensemble of neurons collectively represents information. In the original Georgopoulos et al. (1986) experiment, the ensemble of neurons coded for the direction of a reach; in other words, a two-dimensional vector. The NEF generalizes this type of neural representation to n-dimensional vector spaces, and higher-level representations like functions and vector fields.

The representation principle in the NEF defines a nonlinear encoding process and a weighted linear decoding process. In encoding, we aim to distribute the representation of a vector, $\mathbf{x}$, by injecting current, J, in a neuron model, yielding neural activity $a(J)$. The exact neural activity depends on a nonlinear function, $G[\cdot]$, which is defined by the neuron model.

Encoding

In this thesis, we will primarily use leaky integrate-and-fire (LIF) neurons. LIF neurons are defined by the differential equation

$$\frac{dV}{dt} = -\frac{1}{RC}\left(V(t) - J(t)R\right),$$

where R is resistance, C is capacitance, and $V(t)$ is voltage at time t. When $V(t)$ reaches a threshold V^{th}, the model "spikes," meaning that the time of the spike, t_s, is recorded, and the voltage, $V(t)$, is reset to baseline.

In the general case, the neural activity $a(J)$ is determined by direct simulation. However, the LIF neuron is simple enough that the instantaneous activity rate can be solved for directly.

$$a(J) = G[J] = \begin{cases} \frac{1}{\tau^{ref} - \tau^{RC} \ln\left(1 - \frac{J^{th}}{J}\right)} & \text{if } J > J^{th} \\ 0 & \text{otherwise,} \end{cases} \tag{5.6}$$

where τ^{ref} is the refractory time constant, τ^{RC} is the RC time constant, and J^{th} is the current threshold above which the neuron will spike.

In order to represent $\mathbf{x}$, we must relate the desired vector $\mathbf{x}$ to the input current J for each cell. To do this, we first assign some attributes to each neuron in the ensemble that will represent $\mathbf{x}$.

- $\mathbf{e}$ is a unit length *encoder* that specifies the direction in vector space to which the neuron is sensitive.
- α is a gain term that scales incoming signals.
- J^{bias} is background current that is always present, biasing the cell to spike or not spike at varying input levels.

Then, the amount of current that is injected in that neuron is

$$J(\mathbf{x}) = \alpha \mathbf{e} \cdot \mathbf{x} + J^{bias}, \tag{5.7}$$

where $\cdot$ is the dot product. Since the dot product gives us the projection of one vector on another, input current is high when the two vectors are similar; effectively, the $\mathbf{e} \cdot \mathbf{x}$ term tells us that neurons fire more strongly when the input signal is similar to the part of the vector space that the neuron is sensitive to. That similarity is then scaled by α and biased by J^{bias}, giving us the input current for each neuron.

The parameters associated with each neuron can be chosen in several ways. The most common way is to randomly sample a distribution that is constrained by knowledge of the system being modeled. For example, if we are modeling an ensemble of Purkinje cells, then the gains should have the cell spike at between 1–150 Hz for normal input signals. If we are are modeling an ensemble of brightness sensitive cells, and there are usually twice as many 'on' neurons as 'off' neurons, then we would bias the random generation of encoders appropriately. Other times, the neuron parameters are chosen in order to better implement a particular function. For example, if we are attempting to implement a thresholding function (e.g., $f(x) = 1$ if $x > 0.9$) then we may use all positive encoders and biases such that the cells will only fire when $x > 0.9$.

Decoding

In decoding, the goal is to estimate the originally encoded vector $\mathbf{x}$. The NEF does this with a weighted sum of the neural activities with a set of decoding weights, $\mathbf{d}$. That is,

$$\hat{\mathbf{x}} = \sum_{i=0}^{n-1} \mathbf{d}_i a_i, \tag{5.8}$$

where $\hat{\mathbf{x}}$ is the decoded estimate of the encoded vector $\mathbf{x}$, $\mathbf{d}_i$ is the decoding weight for neuron i, and a_i is the activity of neuron i.

To determine the decoding weights $\mathbf{d}$ we minimize the reconstruction error $\mathbf{x} - \hat{\mathbf{x}}$ by setting up the relation

$$\mathbf{A}\mathbf{d} = \mathbf{X}, \tag{5.9}$$

where $\mathbf{X}$ is a set of sample points in the vector space, and $\mathbf{A}$ is a matrix with the activities of all neurons for all neurons; i.e.,

$$\mathbf{A} = \begin{bmatrix} a_0(X_0) & a_1(X_0) & \cdots & a_{n-1}(X_0) \\ a_0(X_1) & a_1(X_1) & \cdots & a_{n-1}(X_1) \\ \vdots & \vdots & \ddots & \vdots \\ a_0(X_{m-1}) & a_1(X_{m-1}) & \cdots & a_{n-1}(X_{m-1}). \end{bmatrix} \tag{5.10}$$

This equation is in the form of a linear least squares problem, which can be solved with standard methods (Lawson and Hanson, 1974). See Figure 5.12 for an illustration of how the weighted sum results in an estimate of an encoded scalar.

For neurons with instantaneous rate equations, like the LIF neuron (see Equation (5.6)), the description to this point is sufficient to encode and decode information. However, to deal with spiking neuron models, we must explicitly consider time, which to this point has been an implicit part of all of the equations. We model neural spikes as Dirac-δ functions, and define the time-varying neural activity as

$$a(t) = \sum_s \delta(t - t_s) * h(t) = \sum_s h(t - t_s), \tag{5.11}$$

where s is the set of spike times, $*$ is the convolution operator, and $h(\cdot)$ is a filter designed to model the change in a neuron's current as a result of an incoming spike of neurotransmitter.[1]

[1] Equation (5.11) is also how we determine the steady state activity of each neuron in equation (5.10) for neuron models that do not have an analytical instantaneous rate equation.

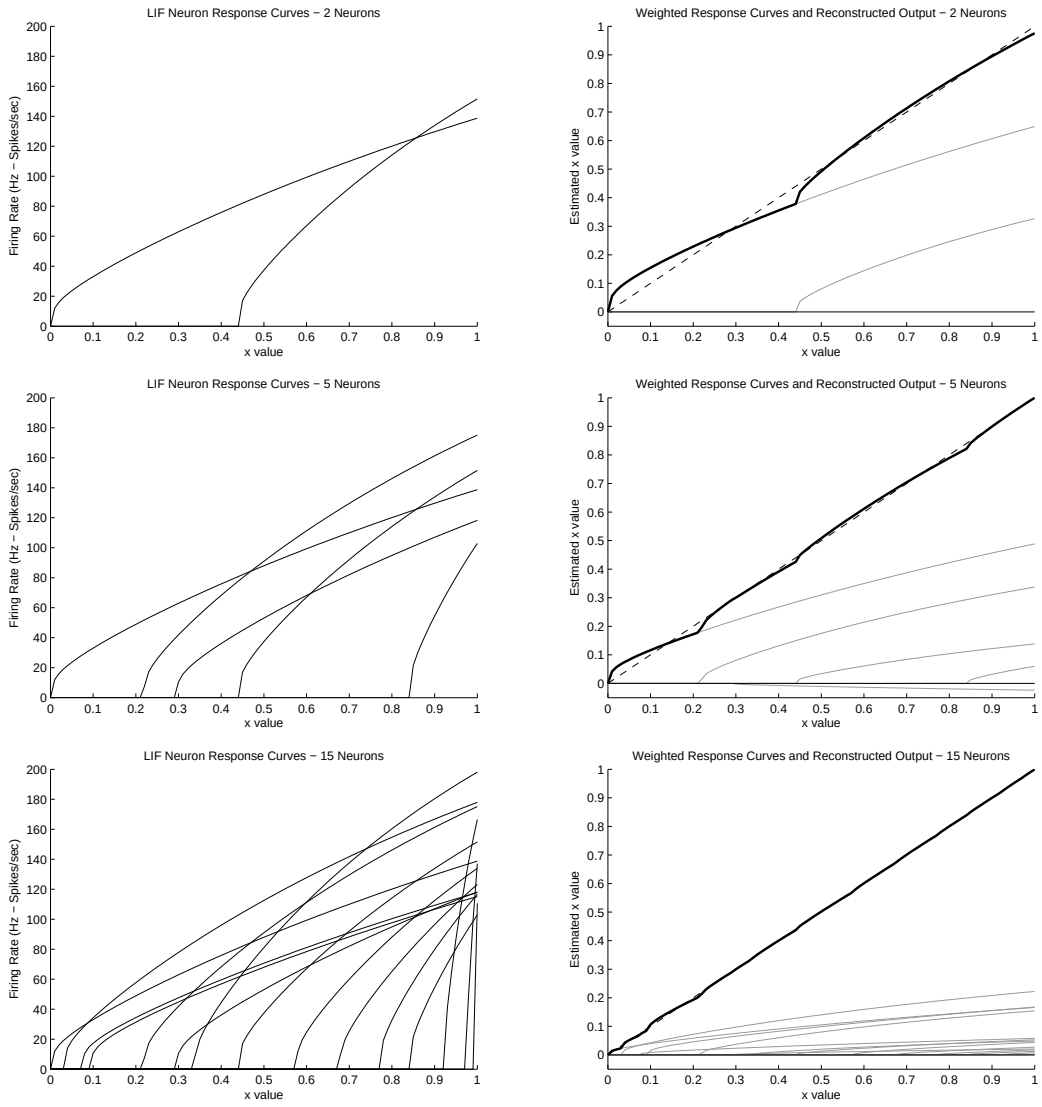


Figure 5.12: Illustration showing how the tuning curves of a population of LIF neurons can be linearly combined to estimate an input signal, x. Reproduced from Choo (2010).

Typically, we emulate post-synaptic current curves recorded empirically by using a decaying exponential; i.e.,

$$h(t) = e^{-t/\tau^{PSC}}, \tag{5.12}$$

where τ^{PSC} is the time constant of the exponential curve decay, designed to match a recorded

post synaptic current (PSC) curve.

Putting these equations together, we get the final expression for the decoded estimate of the input $\mathbf{x}$ using spiking neuron models,

$$\hat{\mathbf{x}}(t) = \sum_{i=0}^{n} \mathbf{d}_i \sum_{s_i} h(t - t_{s_i}). \tag{5.13}$$

See Figure 5.13 for an illustration of the temporal decoding process.

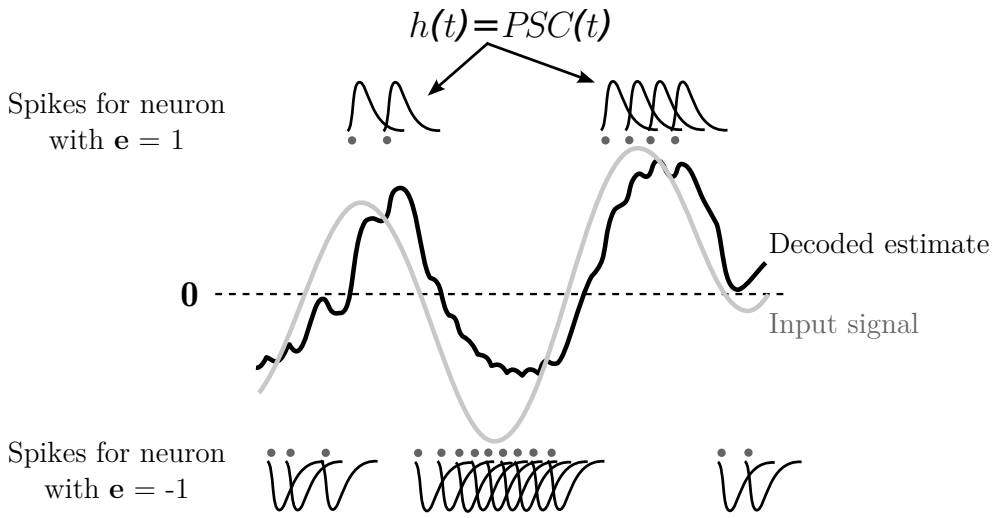


Figure 5.13: Example of decoding a scalar signal using a filtered spike train. For each spike, the filter $h(t)$ is pasted in, weighted by the decoding weight. Adapted from Eliasmith (2011).

5.3.2 Transformation

In the representation principle, we treated neurons as though it were possible to directly inject current to the somas of each neuron. While direct current injections can make sense for modeling sensory systems, the majority of neurons receive their input from other neurons. In the transformation principle, the NEF defines an alternately weighted linear decoding that can transmit functions of the vector represented by one ensemble to another ensemble through a connection weight matrix.

Given some neuron j in an ensemble downstream of an ensemble with neurons indexed

by i, the input current to neuron j is

$$J_j = \sum_{i=0}^{n-1} \omega_{ij} a_i + J_j^{bias}, \tag{5.14}$$

where ω_{ij} is the strength of the connection between neurons i and j.

Consider the case in which we want an ensemble to represent the same vector as the ensemble providing it input; i.e., we want to implement the function $f(\mathbf{x}) = \mathbf{x}$. Equation (5.8) tells us how we can approximate $\mathbf{x}$, so let us substitute that approximation into equation (5.7).

$$\begin{aligned} J_j &= \alpha_j \mathbf{e}_j \cdot \hat{\mathbf{x}} + J_j^{bias} \\ &= \alpha_j \mathbf{e}_j \cdot \sum_{i=0}^{n-1} \mathbf{d}_i a_i + J_j^{bias} \\ &= \sum_{i=0}^{n-1} \alpha_j \mathbf{e}_j \cdot \mathbf{d}_i a_i + J_j^{bias} \end{aligned} \tag{5.15}$$

Setting (5.14) equal to (5.15), we can rearrange terms to obtain an equation for ω_{ij}.

$$\begin{aligned} \sum_{i=0}^{n} \omega_{ij} a_i + J_j^{bias} &= \sum_{i=0}^{n} \alpha_j \mathbf{e}_j \cdot \mathbf{d}_i a_i + J_j^{bias} \\ \omega_{ij} &= \alpha_j \mathbf{e}_j \cdot \mathbf{d}_i \end{aligned} \tag{5.16}$$

Since the decoding process is linear, we can implement any linear function by introducing a matrix $\mathbf{L}_{ji}$ that performs a different combination of the downstream ensemble's encoders, $\mathbf{e}_j$, and the upstream ensemble's decoders, $\mathbf{d}_i$.

$$\omega_{ij} = \alpha_j \mathbf{e}_j \mathbf{L}_{ji} \mathbf{d}_i \tag{5.17}$$

Since the decoding process is a weighted summation, we can compute the sum of vectors of the same length by representing those vectors with separate ensembles and making a connection from each ensemble to the downstream ensemble.

$$\begin{aligned} J_j &= J_j^{\mathbf{x}} + J_j^{\mathbf{y}} + J_j^{bias} \\ &= \sum_i \omega_{ij}^{\mathbf{x}} a_i^{\mathbf{x}} + \sum_i \omega_{ij}^{\mathbf{y}} a_i^{\mathbf{y}} + J_j^{bias} \\ \omega_{ij}^{\mathbf{x}} &= \alpha_j \mathbf{e}_j \mathbf{L}_{ji}^{\mathbf{x}} \mathbf{d}_i^{\mathbf{x}} \text{ and } \omega_{ij}^{\mathbf{y}} = \alpha_j \mathbf{e}_j \mathbf{L}_{ji}^{\mathbf{y}} \mathbf{d}_i^{\mathbf{y}}, \end{aligned} \tag{5.18}$$

and so on for any number of input ensembles.

Nonlinear transformations of some vector $\mathbf{x}$ can be implemented by computing a new set of decoding weights, $\mathbf{d}^{f(\mathbf{x})}$, which minimizes the reconstruction error between the estimate $\hat{\mathbf{x}}$ and the result of applying the function, $f(\mathbf{x})$. That is, equations (5.9) and (5.10) become

$$\mathbf{A}^{f(\mathbf{x})} d^{f(\mathbf{x})} = f(\mathbf{X})$$

$$\mathbf{A}^{f(\mathbf{x})} = \begin{bmatrix} a_0(f(X_0)) & a_1(f(X_0)) & \cdots & a_{n-1}(f(X_0)) \\ a_0(f(X_1)) & a_1(f(X_1)) & \cdots & a_{n-1}(f(X_1)) \\ \vdots & \vdots & \ddots & \vdots \\ a_0(f(X_{m-1})) & a_1(f(X_{m-1})) & \cdots & a_{n-1}(f(X_{m-1})), \end{bmatrix}$$

and the remaining equations only change in the set of decoding weights used. See Figure 5.14 for an illustration of decoding a nonlinear function.

Note that this method of computing nonlinear transformations requires that all of the vectors participating in the transformation must be represented by a single ensemble. Therefore, in order to compute a function $f(\mathbf{x_1}, \mathbf{x_2})$, a new ensemble must be created that will represent a vector in a space that is the concatenation of the spaces in which $\mathbf{x_1}$ and $\mathbf{x_2}$ reside.

5.3.3 Dynamics

Dynamical systems have been used effectively for control problems in several domains. As animals are successful controllers of cognitive and motor actions, implementing dynamical systems with spiking neurons seems a natural fit. In the NEF, dynamics are implemented by interpreting the vector represented by some ensemble as the state of a dynamical system, which can implement differential equations through recurrent connections. No additional techniques are required to compute recurrent connections; the transformation principle is used as previously described. The dynamics principle is instead a type of network architecture (see Figure 5.15) that can implement differential equations as are described in dynamical systems and control theory.

The architecture depicted in Figure 5.15 implements a linear control system, and can be expressed as

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t),$$

where $\mathbf{A}$ is the dynamics matrix and $\mathbf{B}$ is the input matrix. Normally, transforming this equation into Laplace space results in dynamics scaled by the filter $H(s) = \frac{1}{s}$. However, the actual dynamics of the system are based on the synaptic dynamics, $h(t)$ from equation (5.12). The

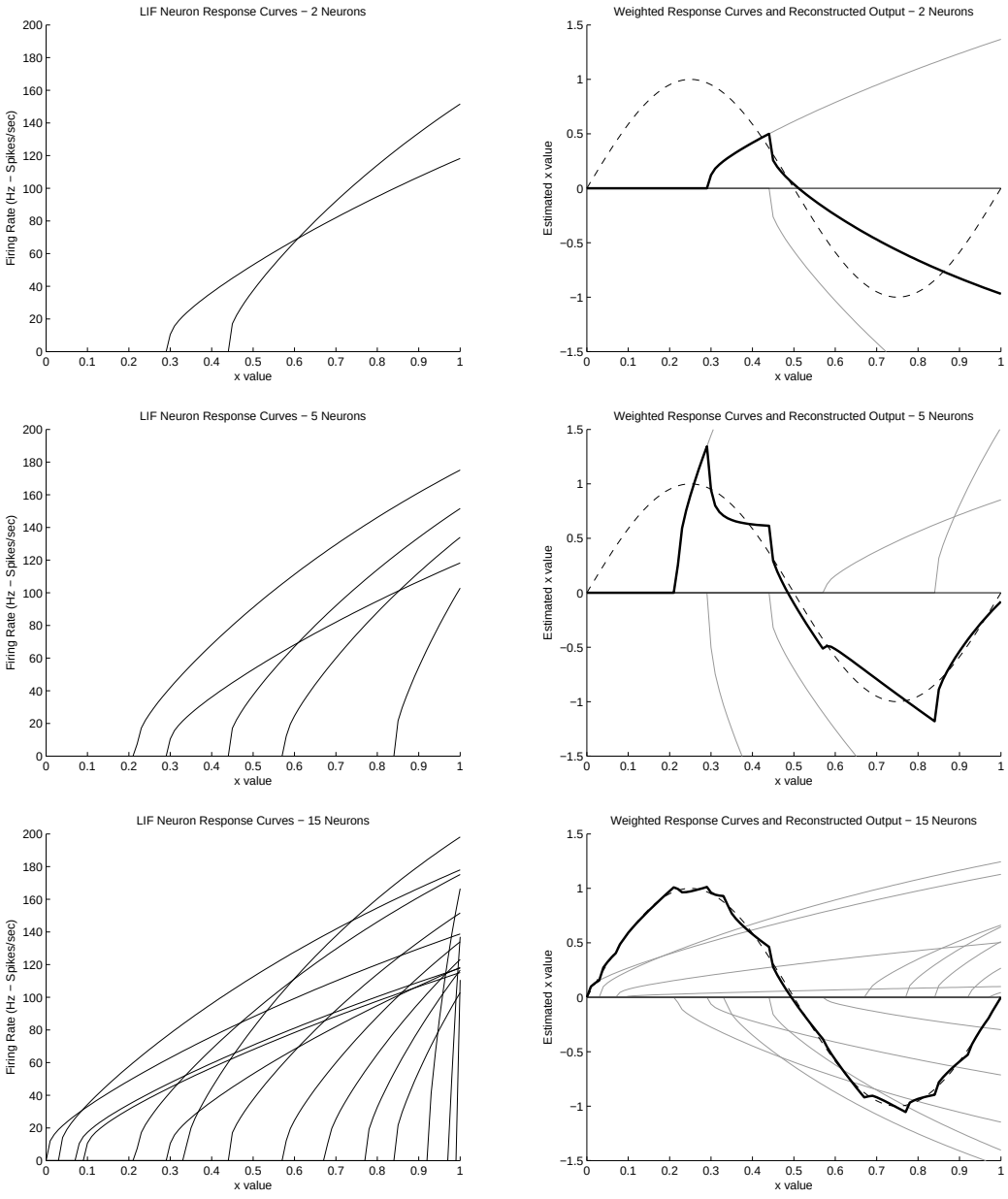


Figure 5.14: Illustration showing how the tuning curves of a population of LIF neurons can be linearly combined to estimate a nonlinear function; in this case, the nonlinear function being estimated is $\sin(2\pi x)$. Reproduced from Choo (2010).

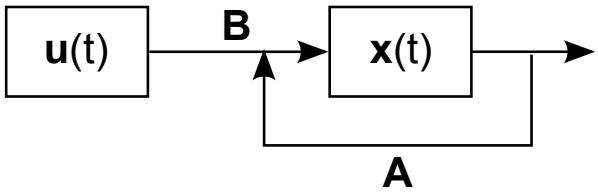


Figure 5.15: Network architecture for implementing linear control systems with the NEF. The input ensemble represents the input signal, $\mathbf{u}(t)$, which is scaled by the input matrix $\mathbf{B}$. The state ensemble represents the system state, $\mathbf{x}(t)$. The recurrent connection implements dynamics according to the dynamics matrix $\mathbf{A}$. Reproduced from DeWolf (2015).

Laplace transform of $h(t)$ is $H'(s) = \frac{1}{1+s\tau}$. By accounting for the difference between these two filters (see Eliasmith and Anderson 2004; Eliasmith 2013 for a full derivation), we get

$$\mathbf{A}' = \tau\mathbf{A} + \mathbf{I} \tag{5.19}$$

$$\mathbf{B}' = \tau\mathbf{B}, \tag{5.20}$$

where $\mathbf{I}$ is the identity matrix.

Equations (5.19) and (5.20) allow us to implement any linear dynamical system in spiking neural networks by setting up the recurrent connection from the ensemble representing the system state to compute the linear transform $\tau\mathbf{A}+\mathbf{I}$, and setting the transform on the connection from the input ensemble to the system state ensemble to have weight $\tau\mathbf{B}$.

Nonlinear dynamical systems can also be implemented with a similar approach; in essence, all that is required is to recognize that the synaptic filter $h(t)$ introduces an exponential "forgetting" of the system state which must be accounted for by scaling connections with the time constant τ. See Eliasmith and Anderson (2004); Eliasmith (2013) for more details.

5.4 Semantic Pointer Architecture (SPA)

The Neural Engineering Framework (NEF) allows us to manipulate dynamical systems in n-dimensional vector spaces with spiking neurons, which is essential for the continuous temporal tasks involved in speech. However, speech also connects to linguistic and cognitive systems that have traditionally been described with symbols and rule-based algorithms. The Semantic Pointer Architecture (SPA; Eliasmith, 2013) provides a method to implement symbol-like representations and transformations in spiking neural networks, building on the NEF.

The crux of the SPA is to implement the operations in a vector symbolic architecture using the NEF. While many vector symbolic architectures exist, the SPA specifically uses Holographic Reduced Representations (HRRs), developed by Plate (1994), as it scales well to high dimensional spaces, and can be implemented naturally with the NEF.

5.4.1 Representation

As in all vector symbolic architectures, we associate a real-valued vector with every "symbol" in the vocabulary associated with the problem domain. In the case of speech, the vocabulary depends on the specific subproblem, but could be composed of phoneme symbols, syllable symbols, word symbols, and so on.

In order to robustly implement the transformations discussed subsequently, we use vectors of unit length when possible. When we have no information about the structure of a particular pointer, we randomly generate it by sampling from a normal distribution for each dimension of the vector, then normalizing by the L2-norm of the vector.

When we do have information about the structure of vector, we construct them by using the transformations below with randomly generated vectors, or vectors learned through unsupervised methods. However, when doing these transformations, it is possible that we will obtain vectors that are no longer of unit length. To avoid this, we can use random *unitary* vectors (Plate, 1994).

A unitary vector is a vector with the same exact inverse as its approximate inverse. The exact inverse of a vector is sensitive to noise, which is prevalent in biological systems. Therefore, when a transformation needs to invert a vector, HRRs instead uses an approximation of the inverse called the *involution*. Given a vector $\mathbf{A}$, the involution of that vector, $\mathbf{A}'$, is a rearrangement of the vector dimensions; specifically,

$$\mathbf{A}' = A'_i = A_{-i \bmod n},$$

where n is the number of dimensions, and mod is the modulo operator. For example, if $\mathbf{A} = [a_0, a_1, a_2, a_3]$, then $\mathbf{A}' = [a_0, a_3, a_2, a_1]$. It is apparent that $\mathbf{A}'' = \mathbf{A}$, hence its usefulness as an approximate inverse.

The approximate inverse is equal to the inverse when the components of the discrete Fourier transform of the vector have unit magnitude. Therefore, to generate a random unitary vector, we generate a random unit vector, and then normalize the frequency components; specifically,

$$\mathbf{A}^1 = \mathrm{IDFT}\left(\frac{\mathrm{DFT}(\mathbf{A})}{\|\mathrm{DFT}(\mathbf{A})\|}\right), \tag{5.21}$$

where $\|\cdot\|$ is the L2-norm. Note that the a unitary vector generated with equation (5.21) is still a unit vector.

5.4.2 Transformation

Vector symbolic architectures define two transformations that are necessary to implement rule-like behavior: merging and binding. Merging takes two vectors and combines them such that the resulting vector is similar to both of the input vectors. Binding takes two vectors and combines them such that the resulting vector is dissimilar from either of the input vectors, but the input vectors can be recovered through an unbinding process.

In HRRs, the merging transformation, which we will denote with $\oplus$, is vector superposition.

$$\mathbf{A} \oplus \mathbf{B} = [A_0 + B_0, A_1 + B_1, ...]$$

Since the new representation is a linear combination of the two vectors, it is similar to both, as long as the two vectors are not too dissimilar. Note that similarity in SPA can be computed either by the dot product between two vectors,

$$\mathbf{A} \cdot \mathbf{B} = \sum_{i=0}^{n-1} A_i B_i,$$

or the cosine similarity, which is the dot product normalized by the length of the two vectors; i.e.,

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=0}^{n-1} A_i B_i}{\sqrt{\sum_{i=0}^{n-1} A_i^2} \sqrt{\sum_{i=0}^{n-1} B_i^2}}.$$

The binding transformation in HRRs, which we will denote with $\circledast$, is circular convolution. While circular convolution is defined as

$$\mathbf{A} \circledast \mathbf{B} = \sum_{i=0}^{n-1} A_{i \bmod n} B_{j-i \bmod n}, \text{ for } j = 0, \ldots, n-1,$$

we can reduce the computational complexity of the transform by computing it as the product of discrete Fourier coefficients. Specifically,

$$\mathbf{A} \circledast \mathbf{B} = \text{IDFT}\left[\text{DFT}(\mathbf{A}) \circ \text{DFT}(\mathbf{B})\right],$$

where $\circ$ is the Hadamard (element-wise) product. In addition to lowering the computational complexity, the DFT and IDFT functions are linear, making them easy to implement in a spiking

neural network. Also, we can observe in this formulation that convolving with a unitary vector (which has DFT coefficients of unit magnitude) does not change the magnitude of the resulting element-wise product. Therefore, when convolving a unit vector with a unitary vector, we obtain another unit vector, which is a useful property in many situations.

In order to recover inputs, we can "unbind" the bound vector to retrieve one input vector by binding with the approximate inverse (involution) of the other input vector.

$$(\mathbf{A} \circledast \mathbf{B}) \circledast \mathbf{A}' \approx \mathbf{B}$$

Note that unbinding is approximate; since the binding operation mixes the two vectors together, they cannot be recovered exactly. However, vectors can be subsequently "cleaned up" to known vocabulary items using an autoassociative memory (Stewart et al., 2011).

It is important to note that the HRR definitions for merging and binding do not change the dimensionality of the vector space, unlike other vector symbolic architectures (e.g., tensor products Smolensky (1990)). One can therefore think of the merging and binding operators as compressing the information in the input vectors. The Semantic Pointer Architecture gets its name from the idea that the vectors used in any reasonably large SPA network are made up of compressed versions of other vectors in vocabularies distributed among many interacting networks. The vectors in these vocabularies have meaning. Since the compressed representations can be decompressed through unbinding, we say that they "point" to deeper meanings, analogous to how a pointer in computer science points to an area of memory. Semantic pointers, however, also contain shallow meanings, so that the compressed representations themselves can be used to implement cognitive behaviors. From here on, we will refer to the vectors in a conceptual vocabulary as semantic pointers.

The SPA has been used to implement models of large-scale knowledge representation using WordNet (Crawford, 2014), serial working memory (Choo, 2010), and other cognitive tasks. An integrated model called Spaun combined several previous models in a single unified model that performed eight cognitive tasks with no reorganization. In these tasks, input was provided through a simulated visual system, and output was produced as motor commands driving a three link arm model (Eliasmith et al., 2012; Eliasmith, 2013).

5.5 Nengo software

Nengo is a piece of software for designing and simulating models using the representations, transformations, and dynamics of the NEF and SPA (Bekolay et al., 2013a). The NEF provides a level of abstraction that allows modelers to generate spiking neural networks that implement

dynamic transformations in vector spaces. The SPA provides a level of abstraction on top of that to allow modelers to generate NEF networks to perform cognitively relevant operations on symbol-like representations. Nengo provides modelers a level of abstraction to define and simulate NEF and SPA models using the Python programming language.

Nengo defines a set of six objects used to construct a spiking neural model. The *Ensemble* is a population of spiking neurons that represents a vector over time. Ensembles present a simplified interface, allowing modelers to specify details important for modeling, rather than manually specifying all model parameters; for example, instead of specifying the gain and bias (see Equation (5.7)), one can specify the desired x-intercepts and maximum firing rates of the ensemble's tuning curves. Additionally, one can specify these parameters in terms of probability distributions to be sampled, rather than specifying a value for each neuron.

Nodes provide outputs to the model, or gather inputs from the model, but do so without using spiking neurons. Nodes allow models to interact with environments. Connecting from a node provides input to the model, which could be a direct current injection, a list of vectors to be encoded, or data coming from a sensor like a spiking camera. Connecting to a node allows it to do arbitrary processing on model outputs; nodes can record model outputs, update plots in real time, or drive a physical robot arm.

Connections encapsulate the flow of information between two objects. For connections between ensembles, connections can be specified in vector space as a function of the input ensemble, or in neuron space by fully specifying the connection weight matrix. When functions are provided, the decoder solving process (see Equation (5.10)) is done automatically. Parameters affecting the decoder solving process can also be specified.

Probes specify quantities in the model that should be recorded for analysis and visualization. Nengo by itself does not impose any constraints on how that data is used; for the models in this thesis, the data was analyzed with NumPy (Van Der Walt et al., 2011) and SciPy (Jones et al., 01) and visualized with Matplotlib (Hunter, 2007) and Seaborn (Waskom et al., 2014). However, other analysis and plotting packages could be used.

Networks are a collection of the other five objects (and possibly other networks), and provide the primary abstraction for defining large models in few lines of Python code. The SPA transformations described in the previous section are implemented as several interacting networks. The models described in Chapter 6 consist of some instances of these SPA networks, and several networks that will be explained in Chapter 6.

Python scripts using Nengo to construct the models discussed in the next chapter are available at `https://github.com/tbekolay/phd`.

5.5.1 Integrating Brian into Nengo

As discussed in Section 5.1.7, we use the Brian neural simulator's Brian Hears package to simulate auditory periphery models. As our models are implemented in Nengo, it is necessary for Brian and Nengo to interact in some way. The naive approach, which is necessary in many simulation environments, is to manually step through both simulators, copying the data produced by Brian to Nengo when the Nengo simulator advances. A faster approach would be to pre-generate all of the data produced by Brian and feed that in as static input. However, then the Brian simulation could not use input from the Nengo simulator; a bidirectional interface is important for future closed loop models.

The integration is instead done through a Nengo node. The function attached to the node contains the state of the Brian simulation, which can be advanced at a different rate as the Nengo simulation if desired. In addition to collecting data from the auditory filters, the node can update the sound associated with the auditory filter, allowing for a bidirectional interface. Updating the sound would normally clear the state of the auditory filter, so we instead subclass Brian's Sound object and provide samples from Nengo when needed.

5.6 VocalTractLab articulatory synthesizer

One of the most successful articulatory synthesizers[2] is the VocalTractLab synthesizer (available at http://www.vocaltractlab.de/; Birkholz, 2013). VocalTractLab uses an analog electrical circuit transmission line for generating acoustics, a geometric three-dimensional model for the vocal tract, and contains several glottis models. We will use VocalTractLab for articulatory synthesis in this thesis due to its high-quality synthesis and gestural control model. We will use the default JD2 parameters for the vocal tract model and the default modified two-mass model of the glottis (Birkholz et al., 2011).

[2] An early version of the synthesizer came second in a singing challenge at Interspeech 2007; results available at http://www.interspeech2007.org/Technical/synthesis_of_singing_challenge.php.

Chapter 6

Implementation

In this thesis, we implement and evaluate three neural models that make up parts of the conceptual Sermo model. Each model provides an explanation for how the human brain might solve a task necessary for speech recognition or synthesis using the computational devices at hand, namely spiking neurons.

6.1 Neural cepstral coefficients (NCCs)

Problem statement

A raw audio waveform has high temporal resolution, but low spatial resolution, as it is a one-dimensional signal. Audio must therefore be preprocessed such that temporal information in the recent past is projected into some feature space at the current moment in time. Automatic speech recognition systems preprocess speech by extracting feature vectors containing cepstral coefficients. How can a neural model like Sermo preprocess speech?

The frontend of most automatic speech recognition systems extracts a feature vector from the incoming audio waveform. As reviewed in Section 4.1.1, Mel-frequency cepstral coefficients (MFCCs) have been used successfully in many ASR systems. To a certain extent, MFCCs are motivated by the organization of the human auditory system. We propose neural cepstral coefficients (NCCs) as an alternative feature vector representation for automatic

speech recognition and other tasks involving speech processing. Unlike MFCCs, NCCs are both motivated by the organization of the human auditory system and are implemented the same basic computational units as the human auditory system, spiking neurons.

6.1.1 Model overview

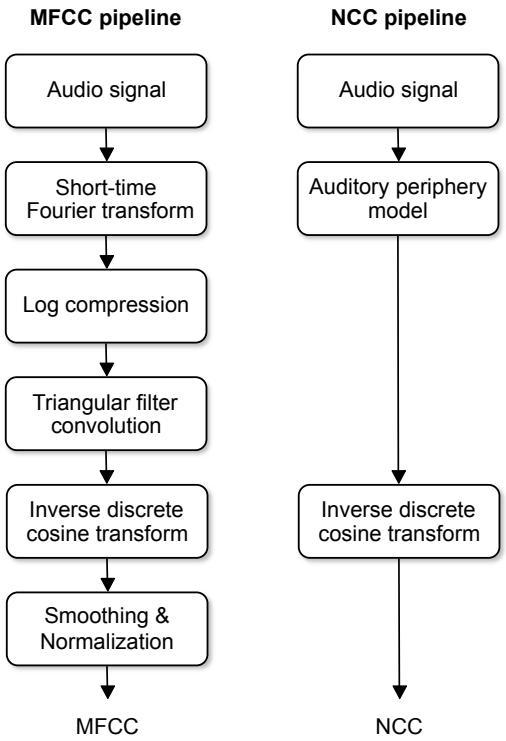


Figure 6.1: Pipeline for generating neural cepstral coefficients (NCCs), compared to MFCCs. While the NCC pipeline initially appears to be simpler, the missing components from the MFCC pipeline are instead incorporated into parts of the NCC pipeline; e.g., the logarithmic compression is part of the auditory periphery model. No explicit smoothing and normalization step is necessary, as signals will be smoothed and normalized in the NCC model by virtue of being represented by spiking neurons, and filtered by the synaptic filters between neurons.

Figure 6.1 shows the NCC processing pipeline, as compared to the MFCC processing pipeline. While the pipelines implement essentially the same computations, the manner in which those computations are performed differs significantly. The primary difference is that in

the NCC pipeline, incoming audio is processed in a continuous online fashion, rather than in a frame-based fashion. Typical MFCC extraction algorithms break the audio signal into overlapping fixed-length windows called frames, which are processed independently until the smoothing stage, where discontinuities due to frame boundaries are minimized. The NCC extraction algorithm, on the other hand, maintains internal state at each processing stage, and updates that internal state for each incoming audio sample. As each internal state update depends on the current state, changes occur smoothly through time, which obviates the need for an explicit smoothing step.

Another implementation difference is in the computation of the inverse discrete cosine transform. Typically, this computation is done as

$$y_k = \frac{x_0}{\sqrt{N}} + \sqrt{\frac{2}{N}} \sum_{n=1}^{N-1} x_n \cos\left(\frac{\pi}{N} n \left(k + \frac{1}{2}\right)\right) \text{ for } 0 \le k < N,$$

where y_k is the kth cepstral coefficient, x_n is the nth auditory filter output, and N is the number of auditory filter outputs. In matrix notation, this equation can be expressed as

$$\begin{aligned}
&\mathbf{k} = [0, 1, \ldots, N-1] && 1 \times N \text{ vector} \\
&\mathbf{s} = \left[\sqrt{2}, 1, 1, \ldots, 1\right] && 1 \times N \text{ vector} \\
&\mathbf{T} = \sqrt{2}N\,\mathbf{s} \circ \cos\left(\frac{\pi}{N}\left(\mathbf{k} + \frac{1}{2}\right) \otimes \mathbf{k}\right) && N \times N \text{ matrix} \\
&\mathbf{y} = \mathbf{T}\mathbf{x} && N \times 1 \text{ vector}
\end{aligned} \tag{6.1}$$

where $\circ$ is the Hadamard (element-wise) product, and $\otimes$ is the outer product. This rearranged equation allows us to precompute a matrix, $\mathbf{T}$, that maps auditory filter outputs directly to cepstral coefficients, enabling us to embed this matrix in the connection weights between two ensembles (see Figure 6.2). By restricting $\mathbf{T}$ to the first M rows, Equation (6.1) yields the first M cepstral coefficients; typically, ASR systems use $N \ge 26$ auditory filter outputs and $M = 13$ cepstral coefficients.

6.1.2 NCC neural model

The NCC pipeline is implemented in a Nengo network that uses the Brian Hears library of auditory filter models (Fontaine et al., 2011), and two layers of leaky integrate-and-fire (LIF) neuron ensembles connected in a feed-forward manner (see Figure 6.2). While Brian has the capability to simulate spiking neurons, we only use Brian's implementation of auditory

periphery models, which do not emit spikes. These models, summarized in Section 5.1, emulate the effect of the audio waveform on the basilar membrane, and the resulting inner hair cell activity arising from basilar membrane deflections. The auditory periphery model also includes rectification and compression, and therefore accomplishes a significant portion of the MFCC extraction pipeline, which is unsurprising as MFCC extraction is designed to emulate certain aspects of the human auditory system.

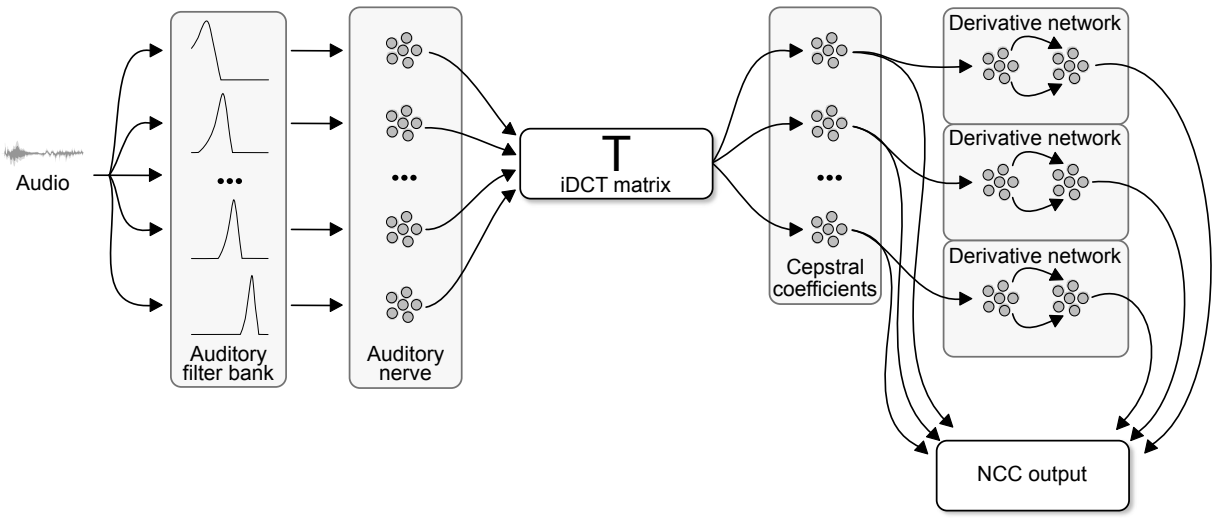


Figure 6.2: Network diagram for the neural cepstral coefficient model. A bank of auditory filters processes incoming sounds, projecting their output to an ensemble for each auditory filter. Those ensembles are connected to ensembles representing cepstral coefficients through the linear **T** matrix, which implements the inverse discrete cosine transform. Ensembles representing cepstral coefficients are projected to networks that compute the temporal derivative of the coefficient. Both the cepstral coefficient and derivative network outputs are concatenated together to form the final output feature vector.

The simulation timesteps for the Nengo model and the contained Brian auditory filter model are uncoupled. The auditory filter model is always run with the same timestep as the auditory stimulus; that is, its internal state is updated for each audio sample. The Nengo model, on the other hand, runs at a timestep independent of the auditory stimulus. On each Nengo timestep, therefore, one or more samples of the audio input is fed through the auditory filter model. The *auditory nerve* ensembles receive as input the current state of the auditory filter model, regardless of how many timesteps the Brian model has advanced.

The *auditory nerve* ensembles mimic the activity of spiral ganglion cells projecting down

the auditory nerve. For each characteristic frequency, a heterogeneous population of neurons is driven by the current output of the auditory filter model. Each characteristic frequency drives an independent neural ensemble, meaning that this layer of ensembles can be thought of as being cochleotopically organized, as the mean activity of an individual neuron only gives information about the output of a single auditory filter (although that filter may be modulated by power in nearby frequencies). Together, the decoded output of each ensemble gives an approximation of the compressed power in the part of the spectrum that the auditory filter is sensitive to. This quantity is essentially the output of the auditory filter (i.e., the first three steps in the MFCC pipeline), but now represented in cochleotopically organized spiking neurons. It is also the $\mathbf{x}$ vector in the inverse discrete cosine transform (Equation (6.1)).

The *cepstra* ensembles each represent one cepstral coefficient. The inverse discrete cosine transform for that coefficient is computed through the connections between the auditory nerve ensembles and the cepstral ensembles. Specifically, the connection between each auditory nerve ensemble and each cepstrum ensemble scales the decoded value of the auditory nerve ensemble by the corresponding value the $\mathbf{T}$ matrix. Since this is a linear transform, it is done by computing normal decoding weights according to equation (5.17) and scaling the decoders by $\mathbf{T}_{i,j}$. Since the currents coming from multiple connections to an ensemble are summed, the cepstral ensembles end up representing the dot product between the row of $\mathbf{T}$ that corresponds to its cepstrum and the $\mathbf{x}$ vector, which is collectively represented by the auditory nerve ensembles.

The end result of the networks is the decoded output of the *cepstra* ensembles, which is a continuously varying, automatically smoothed set of cepstral coefficients. We hypothesize that these coefficients can be used as the feature vectors in an automatic speech recognition system.

6.1.3 Delta neural cepstral coefficients

Another common technique in designing frontends for automatic speech recognition is to include temporal derivatives in the feature vector. Appending MFCC derivatives to the feature vector (also known as MFCC velocities, or Delta-Cepstral Coefficients [DCCs]) improves ASR accuracy. Appending derivatives of the MFCC derivatives (also known as MFCC accelerations, or Double Delta-Cepstral Coefficients [DDCC]) further improves recognition, though by a smaller margin than the improvement from the first derivative (Furui, 1986; Kumar et al., 2011).

Temporal derivatives in these systems are typically computed through straightforward

finite differences. That is, for an MFCC y_k at frame t, the corresponding delta is

$$\delta_k(t) = \frac{\sum_{n=1}^{N} y_k(t+n) - y_k(t-n)}{2\sum_{n=1}^{N} n},$$

where N is a small number of frames (typically between one and three). The double delta applies the same operation to the δ_k vectors.

6.1.4 Delta NCC neural model

Implementing temporal differentiation in spiking neural networks is not as straightforward as implementing Equation (6.1.3) in neurons, as the neural simulation operates online in continuous time, and therefore does not automatically have access to the information from previous timesteps. However, there are several ways to build networks that maintain information from the past and can therefore implement temporal differentiation.

Tripp and Eliasmith (2010) describes six such models, two of which use feedforward connections, two use adaptive neuron models, and two implement linear time-invariant (LTI) filters through recurrent connections. All six models successfully compute a temporal derivative, so for simplicity we implement the two feedforward temporal differentiation models, with the assumption that more sophisticated temporal differentiation models could be used if desired.

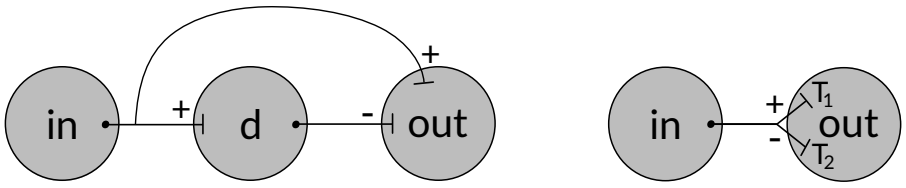


Figure 6.3: Two feedforward temporal differentiation models from Tripp and Eliasmith (2010). (Left) Implementing temporal differentiation with an intermediate ensemble. (Right) Implementing temporal differentiation with no additional ensembles, but two connections between two ensembles with different synaptic filters. See text for more details. Adapted from Tripp and Eliasmith (2010).

In the first feedforward differentiation model, an intermediate ensemble is introduced between the ensemble representing the input signal and the ensemble that we want to represent the derivative of the input signal (see Figure 6.3, left). The *out* ensemble receives direct

positively scaled input from the *in* ensemble, and delayed negatively scaled input from the *d* ensemble. The slight delay between the positive signal communicated from the *in* and the negative signal from the *d* ensemble results in the *out* ensemble representing the temporal derivative of the input signal.

Information transmitted between two ensembles is slightly delayed because changes to the input currents of the first ensemble must first be integrated for one or more timesteps before those changes can result in spikes that influence the second ensemble's input currents. The amount of time between when input signal changes are reflected in the first and second populations depends on several factors, including the simulation timestep, the neuron model, and the synaptic filter model. In the feedforward differentiation model with an intermediate ensemble, all connections use a lowpass filter with a relatively long time constant ($\tau = 0.1$ s) as the synaptic filter model.[1]

In the second feedforward differentiation model, two connections are made between the *in* and *out* ensembles. The first positively scaled connection uses a synaptic filter with a fast time constant ($T_1 = 0.005$ s), and the second negatively scaled connection uses a synaptic filter with a slow time constant ($T_2 = 0.1$ s).[2] Like the first model, this approach exploits the delays introduced by synaptic filtering, subtracting a more slowly changing signal from a signal changing quickly, resulting in a temporal derivative. While this network has the advantage of requiring fewer neurons as no intermediate connection is introduced, it adds the constraint that the neurons in the *out* ensemble contain both fast and slow neurotransmitter receptors.[3]

These differentiator networks are added to the NCC network (see Figure 6.2) to compute the desired derivative. The cepstral coefficients and derivatives are finally concatenated together in a final step to produce the NCC feature vector.

[1] Synaptic filter time constants of around 100 ms are consistent with NMDA glutamate receptors, which Young et al. (2003) have suggested are part of differentiator circuits underlying respiratory rhythm. Additionally, in terms of the biological plausibility of the network as a whole, Tripp and Eliasmith (2010) note that this form of network has been observed in *C. elegans* (Dunn et al., 2004) and may underlie chemotaxis.

[2] Synaptic filter time constants of 5 ms are consistent with some types of AMPA receptors (Jonas et al., 1993).

[3] Since AMPA and NMDA are both glutamate receptors, no constraint is placed on the *in* ensemble. Tripp and Eliasmith (2010) note that a derivative circuit in the vestibular system of oyster toadfish (Holstein et al., 2004) is likely to match the form of the second feedforward differentiation model.

6.2 Syllable sequencing and production

Problem statement

Once the intention to voice a syllable sequence has been formed, the motor system must translate a sequence of conceptual syllable representations to a continuous trajectory of motor commands that will cause muscle activations in the vocal tract. How can a neural model like Sermo generate a motor command trajectory from a symbolic syllable sequence representation?

We base our model on the linguistic theory that the speech production system is organized at the level of syllables, and that production information trajectories are explicitly stored for syllables that are frequently voiced. In this neural model, we describe how sequences of syllables are represented and iterated through with specific timing, and how those representations generate production information trajectories that can control an articulatory synthesizer.[4] We employ several modules in the semantic pointer architecture to iterate through the syllable sequence, and use DMPs to generate production information trajectories. We propose a novel combination of an associative memory and a set of DMP networks that we call a *temporal output associative memory*.

6.2.1 Model overview

In order to construct a network of modules from the semantic pointer architecture, we must first define the vocabulary of pointers used by the model, and how those pointers will be transformed. First, we randomly generate a pointer for each syllable in the model. In order to represent sequences of syllables, we generate semantic pointers to tag each syllable with its place in the list, as was done in Choo (2010). Specifically, a syllable sequence is defined as

$$\text{SEQ} = \text{SYLL1} \circledast \text{POS1} \oplus \text{SYLL2} \circledast \text{POS2} \ldots,$$

where each string of capital letters and numbers is a unique vector interpreted as a semantic pointer (see Section 5.4). For example, the word "Sermo" (/sɜrmoʊ/) would be represented as

$$\text{SERMO} = \text{S3R} \circledast \text{POS1} \oplus \text{MOU} \circledast \text{POS2}.$$

[4] Parts of the model described in this section were created in collaboration with Bernd Kröger. A preliminary version of this model was presented in Kroger et al. (2014).

Representing a syllable sequence by binding syllables to list positions allows us to query the sequence for its constituent syllables, as was done in Choo (2010); Eliasmith et al. (2012). For example, to query the model for the first syllable, we can bind the sequence semantic pointer with the inverse of the first position pointer.

$$\text{S3R} \approx \text{SERMO} \circledast \text{POS1}^{-1}$$

The semantic pointers representing syllable positions are constructed such that subsequent positions are the result of binding a position with a randomly generated NEXTPOS pointer. For example,

$$\text{POS2} = \text{POS1} \circledast \text{NEXTPOS}$$

The initial position, POS1, and semantic pointers for each syllable are also randomly generated.[5] Throughout the simulation, the current syllable to be voiced is computed by binding the sequence with the current position pointer; i.e.,

$$\text{SYLL} = \text{SEQ} \circledast \text{CURRPOS}^{-1},$$

where CURRPOS starts as POS1 and changes over the course of the simulation by binding with the NEXTPOS pointer when each syllable has been voiced.

Since the binding operation ($\circledast$) compresses multiple vectors into a single vector of the same length, the binding operation cannot be used to bind together an arbitrary number of semantic pointers. In Eliasmith et al. (2012), it was shown that a similar network could represent lists with six elements accurately; elements in the middle of lists with seven or more elements were sometimes unable to be queried. Our choice of syllable sequence representation, therefore, sets a soft upper bound of six on the number of syllables in a sequence. We therefore predict that the linguistic systems producing syllable sequences to be voiced will look ahead less than six syllables.

Each syllable semantic pointer is associated with a trajectory generation subsystem based on dynamic movement primitives (DMPs; Schaal, 2006; DeWolf, 2015). DMPs allow for reliable trajectory generation that can be temporally scaled, allowing for syllables to be voiced quickly or slowly. Each DMP generates a trajectory of vocal tract gestures, which are then mapped to motor commands that cause muscle activations in the vocal tract.

[5] Syllable semantic pointer could be constructed to reflect how the syllable is produced; e.g., it could have a representation like $\text{BA} = \text{B} \circledast \text{ONSET} \oplus \text{A} \circledast \text{NUCLEUS}$, where semantic pointers for the individual phonemes and syllable parts are randomly generated. However, as we will show in the remainder of this section, we do not need this information to produce syllables. Therefore, we deliberately do not impose any structure on the syllable semantic pointers in this model because we anticipate that higher-level linguistic subsystems in Sermo will dictate the features of the syllable that must be contained in the syllable semantic pointer representation.

Traditionally, DMPs can be either discrete or rhythmic, depending on whether the trajectory is followed repetitively. Since syllables are voiced one after another in typical speech, a discrete DMP would seem to be the obvious choice; however, it is possible for the same syllable to be voiced twice in a row, and some forms of singing feature rhythmic repetitions of the same syllable, necessitating the use of rhythmic DMPs.

Using rhythmic DMPs means that when a syllable is only voiced once, it must be explicitly stopped once the DMP traverses its limit cycle once. At the same time, the DMP for the next syllable must begin, or have already begun. We accomplish this precise timing by detecting when the current syllable is nearing completion and starting a control network that transitions smoothly between the currently active syllable and the next active syllable. When a syllable is repeated, the end result should appear as though the limit cycle continued undisturbed. The implementation of this network depends heavily on neural inhibition, and therefore will be discussed in Section 6.2.2.

As reviewed in Section 5.2, DMPs typically consist of a point attractor and a forcing function. Currently, we do not implement separate point attractors for each DMP, and instead assume that all DMPs have the same point attractor, whose stable point is to have no active vocal tract gestures. The forcing function for each DMP generates a sequence of temporally coordinated vocal tract gestures that are defined manually.[6]

6.2.2 Neural model

SPA modules implemented in Nengo are used to robustly iterate through the syllable sequence list. The input sequence is represented in a *state* module, which does not modify the sequence. The current and next position pointers is tracked by two interconnected *working memory* modules, as was done in Eliasmith et al. (2012). The current syllable is computed by binding the *state* with the inverse of the current pointer represented in *working memory*; similarly for the next pointer, which is stored in another working memory module. Both the current and next syllable representations are cleaned up with *associative memories*. While the cleaned up semantic pointer associated with syllables are available, they are not used; instead, the activity of the ensemble that is associated with the syllable disinhibits the oscillator associated with a DMP network whose forcing function follows the production information trajectory for that syllable. A timing network that we call a *sequencer* ensures that only one syllable DMP is active

[6] We assume that these vocal tract gesture sequences would be learned through babbling and imitation phases, as are modeled in DIVA and the Kröger model. We leave the incorporation of this learning process into the Sermo model as future work.

at one time, and that syllables transition immediately and smoothly. See Figure 6.4 for an illustration.

SPA modules

Networks for representing semantic pointers and doing common cognitively relevant manipulations on those pointers have been previously created for Spaun. This model uses those networks as described in Eliasmith (2013).

The binding modules compute circular convolution, implemented as multiplication in the Fourier domain. The design of these networks is described in more detail in Choo (2010).

The working memory modules maintain a memory trace through recurrent connections implementing an integrator. While the general design of the working memory is described in more detail elsewhere (Choo, 2010), how the two working memory modules interact warrants further explanation. In order to calculate the next position, we require the current position (see Equation (6.2.1)). The straightforward approach of computing this in a single recurrent loop from one working memory to itself fails because one of the two pointers being used in the computation is also the result of the computation. To remedy this situation, we exploit the fact that each working memory module is gated such that it operates in two regimes. In the "acquisition" regime, the memory module acquires a state to be remembered. In the "maintenance" regime, the memory module ignores incoming state information, and maintains a memory of the last state encountered in the first regime. The two working memory modules are connected to each other, and are always in the opposite regime. Initially, the first memory acquires the POS1 state from an external source. The gating signal turns on, which sets the first memory into the maintenance regime, representing the POS1 state. This simultaneously sets the second memory into the acquisition regime, where it acquires the state being maintained by the first memory (POS1 initially). The second memory is connected to the input of the first memory such that it computes the next position in the connection, and therefore is providing POS2 as input to the first memory, though it is currently ignoring it. When the gating signal is released, the two memories switch regimes. The second memory maintains its memory of POS1 and therefore projects POS2 to the first memory, while the first memory acquires the POS2 state from the second memory. This process repeats for each position in the syllable sequence; each time the gating signal is turned on and off, the position pointer in the first working memory increments.

This method of incrementing the position pointer is both robust and fast. It takes less than 50 ms to acquire a new memory state, and that state can be maintained for several seconds. Since syllables take on the order of hundreds of milliseconds to be voiced, the circuit can go

through a syllable sequence fast enough for natural conversation. However, one aspect of the circuit as described poses a timing challenge. As discussed in Section 6.1.4, there is a slight transmission delay when sending a signal between two ensembles. Yet, humans are capable of producing continuous speech with few to no discernible pauses between syllables. Therefore, the next syllable to be voiced must follow the current syllable very quickly, even overlapping when appropriate. The *sequencer* network, discussed below, facilitates this.

The associative memories map possibly noisy input pointers to clean output pointers. While the input pointers can be associated with arbitrary output pointers, in our case inputs and outputs are the same syllable pointers; this is called an autoassociative memory, or a cleanup memory. These memories contain an ensemble for each semantic pointer in the vocabulary; the ensemble is responsible for computing the similarity between that semantic pointer and some input signal. This computation is carried out through the connection between the ensemble representing the input signal and the ensemble representing the similarity to the desired pointer. The weights on this linear connection are set to be the semantic pointer itself; the encoding process (see Equation (5.7)) therefore computes the dot product between the desired semantic pointer and the input signal. Since semantic pointers are not necessarily orthogonal to each other, more than one ensemble may activate for a given input signal. In order to clean up to a single output semantic pointer, all of the ensembles representing semantic pointer similarity mutually inhibit one another. As long as the semantic pointers are sufficiently dissimilar to each other, this mutual inhibition implements a winner-take-all function, resulting in only a single ensemble being active: the ensemble representing similarity for the semantic pointer most similar to the input signal.

DMP networks

The rhythmic DMP networks generate repetitive production information trajectories. The networks are based on the general design presented in DeWolf (2015), with some modifications to enable interactions between the DMP networks and the SPA modules by way of the sequencer network.

The core of the network is an oscillatory ensemble that represents the system state. As the oscillator traverses the limit cycle, the multidimensional forcing function is decoded from the system state, resulting in the predefined production information trajectory playing out as the oscillator's angle moves from 0 to 2π. The ability to drive the oscillator at different frequencies enables the DMP to generate trajectories at different timescales.

The forcing function of each DMP interpolates a vector representation of a manually defined vocal tract gesture score over time. The gesture scores are designed to be used with

the VocalTractLab articulatory synthesizer (Birkholz et al., 2006; Birkholz, 2013). While a full speech production system would also include a mapping from the gesture score to vocal tract parameter changes (i.e., Sermo's motor expansion system), we leave the neural implementation of this mapping as future work, and instead automatically generate a gesture score from the neurally generated vector representation over time, which can then be synthesized by VocalTractLab.

We have made two modifications to the DMP network design in DeWolf (2015) to allow multiple DMPs to operate in quick succession. The first modification is that the oscillator is inhibited by default. Usually, the oscillatory ensemble has intercepts distributed away from zero, causing a "dead zone" in the part of the two-dimensional vector space represented by the ensemble. The dead zone forces the initial input to the oscillator to be strong enough to kick it out of its dead zone; however, it eliminates the problem of the oscillator randomly oscillating without receiving input. While the DMP networks used in this model still have a dead zone, we also include additional explicit inhibition because the signal that starts the oscillator is provided to all DMPs simultaneously. All DMPs receive this signal for efficiency and scalability reasons. The inhibitory signal is inhibited (and thus the oscillator is disinhibited) by the syllable similarity ensembles associated with the DMP in the two associative memories, which requires around 400 synaptic connections if the memory and inhibitory ensembles contain 20 neurons each. The alternative would be to set up a routing network that would route the oscillator starting signal to the correct oscillators depending on the state of the two associative memories. While such a routing network can be implemented, it would take more neurons and connections to implement than the approach used here, and may introduce delays that would be detrimental to smooth sequencing.

The second modification is a more robust method for starting the oscillator. Previously, we noted that the DMP's oscillator requires a start signal to begin the oscillation. Typically, this signal is a short pulse of driving force pushing the oscillator toward a part of the state space; in our case, we aim to push the oscillator to $[-1,0]$, or $\theta = \pi$. With a short pulse of driving force, the oscillator typically does not make it to $[-1,0]$ exactly, as the intrinsic dynamics of the oscillator quickly take over, beginning the oscillation. Additionally, the pulse must be presented for a sufficiently short time such that it does not affect the oscillator state as it traverse its limit cycle.

In order to overcome the limitations of the usual starting pulse, we instead continuously drive the oscillator to the target state until the starting pulse is released. To implement this continuous drive, we add an additional ensemble to the DMP which receives input from the oscillator and provides output to the oscillator; the output to the oscillator computes the difference between the current oscillator state and the target state, which in our case is $[-1,0]$. Like the oscillator itself, we only want this difference to be computed when the start pulse

is active. Therefore, we provide an inhibitory signal to the differencing ensemble, which is inhibited by the starting pulse, effectively disinhibiting the differencing ensemble and driving the oscillator to the $[-1, 0]$ state.[7] Since the differencing signal is active as long as the starting pulse is active, the start pulse effectively "resets" the oscillator to the target state until the pulse is released, at which point it starts traversing the limit cycle. We therefore overcome the issues with the starting pulse, as the oscillator will be driven to the target state even with the intrinsic dynamics, and the timing pulse can last as long as is needed, though its timing is still critical for natural speech trajectories.

Sequencer network

The structure of the SPA modules and the modifications to the DMP networks described to this point allow the two associative memories to disinhibit the rhythmic DMPs associated with the current and next syllables. All that remains is to generate the timing signals that switch the regimes of the working memories, and that drive any disinhibited DMPs to the target initial state. The sequencer network accomplishes these two functions.

The main idea of the sequencer network is that we keep track of how far along we are in the production of a syllable, and when we are sufficiently close to being done producing a syllable, we initiate the starting pulse and transition to producing the next syllable. Note that this assumes that only one syllable can be the "active" syllable at any one time, but syllables can still blend into one another during the time of the starting pulse.

The core of the sequencer network is a *timer* ensemble that receives input from all of the rhythmic DMP ensembles. If the assumption that at any given time only one syllable is active holds, then the timer ensemble effectively represents the state of the oscillator for the currently active syllable. The timer is used to switch the regimes of the working memories, and to initiate the starting pulse for the next syllable. As it turns out, the timing of these two signals is strongly linked; the ideal time to switch the regimes of the working memories is right after the timing pulse starts, and right after it ends. This timing is ideal because we want both associative memories to emit the "next" syllable during the starting pulse, but for the "current" and "next" syllables to be disinhibited for the rest of the time so that when the next starting pulse occurs, the next syllable is already disinhibited and ready to be voiced, rather than needing to be disinhibited before it can be driven by the starting pulse.

[7] Note that we set the goal state of the differencing ensemble to be the target state plus a small vector that counteracts the intrinsic dynamics of the oscillator. Since those dynamics depend on the frequency of oscillation, the vector added depends on the DMP's frequency (i.e., speed). Currently, we use oscillators with fixed frequency and therefore add a fixed vector to the target state. If controlled oscillators are used, it would be ideal to also modify the vector accordingly, though the gain in accuracy may not be significant.

Note, however, that we aim to have a slight delay between when the timing pulse starts and when the working memory switch occurs, as the switch will effectively inhibit the "current" syllable, which we wish to be active while the next syllable is being driven to the initial state. Since the current syllable's trajectory is bringing it near the initial state, the additional drive resulting from the starting pulse does not adversely affect the trajectory.[8] Similarly, we aim to switch the working memories again slightly before the starting pulse ends to limit the amount of time that the previous and current syllables overlap. Unlike phonemes, syllables are not typically coarticulated; the overlap that we achieve by the delay between the timing pulse and the first working memory switch is to reduce the delay between successive syllables, not necessarily to allow them to coarticulate, though some slight overlap is possible.

Temporal output associative memory

We refer to the combination of an associative memory, and DMP networks for each input-output pair in the memory as a temporal output associative memory. Since we use sequences of syllables as an input, we use the sequencer network to control sequence timing; however, temporal output associative memories can also be used in other situations, including those that allow for motor synergies to be flexibly weighted rather than using an all-or-nothing strategy as is done with syllables. The key innovation here is the association of a possibly noisy semantic pointer with an arbitrary time-varying trajectory. Many applications of this kind of association can be seen in human behavior; here, we associate static syllable representations with production information trajectories that will drive a motor system to produce that syllable over time. A similar mapping could produce other types of motor behavior (e.g., a reach) from static representations of cognitive intentions.

[8] If the trajectory is adversely affected, then the forcing function can be modified such that the predefined trajectory is warped in time to accommodate for the increased speed at the end of the trajectory.

6.3 Syllable recognition

Problem statement

Speech production information is decoded from integrated auditory and non-auditory sensory information. Yet, we perceive the continuous production information trajectory as a single speech unit. How can we recognize a trajectory of production information as having been produced by a syllable target?

Syllable recognition is in many ways the inverse of syllable production. Rather than expand a discrete, static representation to a continuous, dynamic one, we do the opposite and collapse a continuous, dynamic representation to a discrete, static one. As such, we theorize that the trajectory classification aspect of the sensorimotor integration system is also organized at the level of syllables. We propose a syllable recognition network using a novel technique based on DMPs, which we call inverse dynamic movement primitives (iDMPs). We use iDMPs to recognize production information trajectories, and then connect these iDMPs to associative memories in order to produce semantic pointer representations for recognized syllables. We call the novel combination of iDMPs and associative memories *temporal input associative memories* to emphasize the similarities between the syllable production and recognition systems.

6.3.1 Model overview

The first part of the model recognizes incoming production information trajectories with inverse DMPs (iDMPs). Recall that in the DMP framework, the state of the system is represented by $x(t)$, and the motor trajectory is generated by defining a forcing function $f(x(t)) = \mathbf{y}(t)$ that emits the motor trajectory for that point in the state space. In the discrete case, the intrinsic dynamics of the state act as a one-dimensional ramp; in the rhythmic case, the intrinsic dynamics of the state act as a two-dimensional oscillator.

In iDMPs, we aim to learn the inverse function, which maps from the motor trajectory back to the system state that generated that trajectory; i.e., we aim to approximate $g(\mathbf{y}(t)) = x(t)$. Unfortunately, we cannot do this approximation directly as there is no guarantee of a one-to-one mapping from $x(t)$ to $\mathbf{y}(t)$; i.e., the same motor command can occur at multiple points in

the trajectory. Therefore, we use an approach inspired by HMMs and GVF (see Section 4.4.1) to approximate this function.

The basic idea of iDMPs is that we keep track of our current estimate of the system state, $\hat{x}(t)$, and gate the dynamics of the system state estimate to progress only when the input (i.e., an observation, $\mathbf{O}(t)$) is sufficiently close to what we believe the observation should be (i.e., a prediction $\hat{\mathbf{y}}(t) = f(\hat{x}(t))$). We assume that the forcing function f is known. Then, we define a similarity function to compare the observation to the prediction; specifically,

$$s(\hat{\mathbf{y}}(t), \mathbf{O}(t)) = \frac{\hat{\mathbf{y}}(t) \cdot \mathbf{O}(t)}{\|\hat{\mathbf{y}}(t)\| \|\mathbf{O}(t)\|}, \tag{6.2}$$

where $\cdot$ is the dot product and $\|\cdot\|$ is the L2-norm.

Recall that the intrinsic dynamics of a discrete DMP are

$$\dot{x} = \alpha,$$

and the intrinsic dynamics of a rhythmic DMP are

$$\begin{aligned} \dot{x_1} &= -2\pi f x_2 \\ \dot{x_2} &= 2\pi f x_1. \end{aligned}$$

In the iDMP, we have estimates of the actual system state, and we gate the dynamics of the estimate by the similarity measure. Given some similarity threshold th_s, the discrete case of an iDMP has dynamics

$$\dot{\hat{x}} = \begin{cases} 0 & \text{if } s(\hat{\mathbf{y}}(t), \mathbf{O}(t)) < th_s \\ \alpha & \text{if } s(\hat{\mathbf{y}}(t), \mathbf{O}(t)) \geq th_s, \end{cases} \tag{6.3}$$

and the rhythmic case of an iDMP has dynamics

$$\begin{aligned} \dot{\hat{x}_1} &= \begin{cases} 0 & \text{if } s(\hat{\mathbf{y}}(t), \mathbf{O}(t)) < th_s \\ -2\pi f x_2 & \text{if } s(\hat{\mathbf{y}}(t), \mathbf{O}(t)) \geq th_s, \end{cases} \\ \dot{\hat{x}_2} &= \begin{cases} 0 & \text{if } s(\hat{\mathbf{y}}(t), \mathbf{O}(t)) < th_s \\ 2\pi f x_1 & \text{if } s(\hat{\mathbf{y}}(t), \mathbf{O}(t)) \geq th_s. \end{cases} \end{aligned} \tag{6.4}$$

Note that, like other trajectory recognition algorithms, each iDMP is associated with a specific trajectory, and therefore we require as many iDMPs as we have syllables in our vocabulary. Also like the GVF algorithm, iDMPs can be considered a "tracking" algorithm, as the underlying state is continuously estimated. Classification can therefore be done flexibly depending on

the characteristics of the trajectories in the vocabulary. Classification can be done when any iDMP's state passes some classification threshold, (e.g., when $\hat{x}(t) > 0.9$ in the discrete case). It can be done on an ad hoc basis, essentially querying the set of iDMPs for which is most likely at that moment in time (i.e., $\arg\max \hat{X}(t)$). It can also be done when one iDMP is sufficiently farther along than all other iDMPs.

It is important to note that the α and f parameters used in the iDMP cannot be chosen in the same way as in DMPs. In DMPs, the α and f parameters control how quickly the DMP is moving through the state space, and therefore controls the temporal scale of the trajectory. In the iDMP, however, α and f stay fixed, and instead represent the maximum speed that we expect the iDMP to move through the state space, or perhaps slightly higher than the maximum speed. If α and f are too low and the estimated state lags behind the actual state, then the predictions will diverge from the observations, leading to poor tracking (false negatives). If α and f are too high, then the iDMP may advance too far in the state space, making an improper classification (false positives). Choosing a good α or f is critical to the success of an iDMP.

While we believe that the iDMP is well suited to a wide range of trajectory classification problems, it is especially useful for syllables because it allows for the onset of the syllable to be the dominant factor in differentiating between another syllable with the same nucleus, but no onset. For example, if the input is a trajectory from the syllable [bɑr], the iDMP for [bɑr] will be further along in the state space than the iDMP for [ɑr]. However, an issue arises for codas, in that the iDMP for [bɑ] will be farther along than [bɑr] when the [r] is encountered. We will refer to this as the *trajectory subpath* problem. The best solution to the trajectory subpath problem depends on the characteristics of the trajectories in the vocabulary. In the case of syllables, we solve the trajectory subpath problem by introducing competition between iDMPs with common trajectory subpaths (see 6.3.2 for more details).

In the syllable recognition model, a discrete iDMP is associated with each syllable in the vocabulary. Each syllable is also associated with a randomly generated semantic pointer, which should be the same semantic pointer used by the syllable production system to produce that syllable.

6.3.2 Neural model

The spiking neural implementation of the syllable recognition model is organized much like the production model, but in reverse. Each syllable has an associated iDMP network that aims to recognize that syllable. The output of each iDMP is associated with an ensemble in an SPA associative memory, which produces a cleaned up version of the semantic pointer associated with the recognized syllable. The cleaned up version of the semantic pointer is projected to a

memory module, which remembers the recognized syllable for a short while in order to make it available to other networks. Finally, a temporal classification ensemble is used to monitor the state of the associative memory and reset the iDMP state ensembles once a classification has been made. See Figure 6.5 for the overall structure.

iDMP networks

The iDMP associated with each syllable is similar in structure to the DMPs described in Section 6.2.2. The core of the iDMP network is an ensemble that represents the estimated state of the dynamical system. A recurrent connection from the state to itself implements the dynamics that updates the estimated state based on the current observation and prediction (see Equations (6.3) and (6.4)).

Despite the production network using rhythmic DMPs, we use discrete iDMPs for syllable recognition. A significant amount of effort is taken in the production network to ensure that the 2D oscillators are reset at the correct time. With iDMPs, we cannot be as precise about time, as we are estimating the system state; it can even be advantageous to be less concerned about exact time, as it can allow the iDMP to classify a trajectory before it is complete. As such, we are essentially collecting evidence to increase our confidence that the observations come from the same syllable that the iDMP is created from; integrating that evidence over time is naturally done with a one-dimensional integrator, as is used in the discrete iDMP. Additionally, since we do not need to emit a classification immediately after the last one (we must first accumulate trajectory evidence) we can safely spend some milliseconds resetting all iDMP states once a classification is made.

We implement the discrete dynamics from Equation (6.3) in a single recurrent connection. Typically, Nengo models that compute the dot product (as is done in Equation (6.2)) do so by computing each 2D product separately, as the product is a nonlinear function that can be difficult to optimize in high-dimensional spaces. However, initial experimentation showed that the exact dynamics equation could be well approximated in a single connection as long as the observation and prediction were reduced in dimensionality. In general, this could be done with any number of statistical dimensionality reduction techniques; however, in the case of production information, the vector space already has orthogonal basis functions (there is no coupling between gestures), and each syllable uses relatively few dimensions (usually between four and seven). Therefore, for each syllable, we only explicitly represent the dimensions of the space that are actually used for that syllable. With this dimensionality reduction, we found that a recurrent connection specifying exactly Equation (6.3) was approximated well enough to do trajectory recognition in most cases.

The fact that we implement the dynamics in one recurrent connection implies that the ensemble represents both the system state and the observation dimensions of interest to that syllable. In other words, the state can have up to eight dimensions for complex syllables. In our case, even when high accuracy is required, the number of neurons needed is reasonable given some assumptions about adult vocabularies; see Section 7.2.3 for more details.

In the previous section, we discussed the trajectory subpath problem, which occurs when two syllables start with the same trajectory but diverge near the end (e.g., [ba] versus [bar]). In order to handle this problem, we introduce an additional connection in the model that explicitly punishes gestures that are not part of the iDMP's syllable. One method to do this would be to negatively weight any gestures not part of the current syllable in the recurrent connection; however, reducing the dimensionality of the observation space makes this impossible. Instead, we introduce a small *reset* ensemble in the iDMP that drives the system state back to the initial state whenever a gesture that is not part of the syllable is active.[9] This ensemble is normally inhibited, but an inhibitory connection is made from the input signal to the reset ensemble such that any gesture not part of the syllable will disinhibit the reset ensemble, pushing the iDMP state back to the initial state. As long as the target trajectories are normalized to the same length, punishing these gestures should mitigate the trajectory subpath problem in most cases.[10]

As an added benefit, punishing gestures not in the syllable helps differentiate between syllables with similar trajectory suffixes; e.g., [bar] versus [dar]. Without punishing extra gestures, it is more likely that the iDMP associated with [dar] will catch up to the [bar] iDMP and result in a misclassification if the parameters in the model are not well-tuned.

SPA modules

The associative memory is a standard SPA associative memory, mapping from a set of input semantic pointers to a set of output semantic pointers. In the recognition model, however, we do not provide any semantic pointer input; instead, we use the iDMP networks to drive the ensembles associated with each syllable semantic pointer directly. Specifically, the dimension or dimensions representing the system state are connected to the similarity sensitive ensembles.

[9] The reset ensemble is similar to the reset ensemble in the syllable production model in its implementation and use, in addition to its name.

[10] It should be noted that humans are not necessarily adept at disambiguating trajectory subpath issues; consonant phonemes can be interpreted as being part of the coda of one syllable or the onset of the subsequent syllable in many syllable sequences. The form of syllable recognition explained by this model would be influenced by linguistic systems, or used primarily in sensorimotor situations like learning to voice new syllables and recognizing infrequently heard syllables in a foreign language.

One benefit of using the associative memory is that it separates the state tracking function of the iDMP networks from the winner-take-all mutual inhibition of the associative memory, which makes the iDMP network easier to implement. Another important benefit is that it provides a location for sensorimotor information (provided by the iDMPs) to integrate with top-down linguistic information. Since neurons naturally sum input from multiple sources, the ensembles representing syllable similarity can receive input from both sources, and achieve faster or more correct classification as a result.

The mutual inhibition and thresholding options are critical to the proper functioning of the associative memory in the recognition model. Figure 6.5 shows the mutual inhibition in the associative memory network. When mutual inhibition and thresholding are used, the second layer of ensembles in the associative memory only become active if the first layer of ensembles (those that represent semantic pointer similarities) are sufficiently active. Thresholding is implemented by setting ensemble properties in the second layer to have encoders of [1], and x-intercepts that start around or above the chosen threshold. In doing so, the second layer of ensembles will only become active if the first layer ensemble associated with it crosses that threshold.

In addition to the associative memory module, which identifies a unique syllable semantic pointer once the iDMP associated with it has crossed threshold, the model contains a memory module that maintains a decaying memory of that pointer. The memory is implemented as an ensemble for each set of 16 subdimensions of the semantic pointer (which is 64 dimensional in this model). Each ensemble is recurrently connected with a weight less than 1 so that the syllable most recently classified is remembered for a few hundred milliseconds, which is around the amount of time that we expect for the next classification to take place. Using a recurrent weight with weight less than 1 is critical in this case, as if the memory used recurrent connections with weight 1, all of the syllable pointers encountered in the past would be superimposed on one another. Therefore, we assume that only the most recent syllable classified is of interest to downstream systems.

Temporal classification

The final part of the model is a small *classifier* ensemble that resets the iDMP states when a classification is made. It receives input from all of the second (thresholded) layer of ensembles in the associative memory. When the classifier ensemble is active, it directly inhibits all of the iDMP state ensembles.

Like the thresholded layer, the classifier ensemble's x-intercepts are distributed near or above a certain threshold; critically, this threshold is slightly above the threshold in the associa-

tive memory. The difference between these two thresholds results in a slight delay between when the associative memory output becomes active, and when the classifier ensemble becomes active. The associative memory must be active long enough to provide semantic pointer input to the memory module, which will maintain a memory of that pointer once the iDMP states are inhibited by the classifier ensemble.

Since the classifier directly inhibits all of the iDMPs, any neural activity in this population will have a large deleterious effect on the classification of the current syllable, and so we aim to completely inhibit the iDMP once the classifier becomes active. However, since the classifier inhibits an ensemble that provides it input, its activity will quickly cease. In order to ensure that its short burst of activity is sufficient to completely reset the iDMP states, we change two parameters from their defaults. First, we use synaptic time constants that are longer than usual on the connections from the associative memory to the classifier, and from the classifier to the iDMPs; specifically, the time constant is 100 ms, which is consistent with some NMDA receptors (Sah et al., 1990). Second, we use a different neuron model. All other ensembles mentioned in this thesis are composed of leaky integrate-and-fire neurons. In the classifier ensemble, we use an adaptive leaky integrate-and-fire neuron model (Koch, 1998). While a true bursting neuron model would be better, the adaptive LIF at least will produce more spikes given sporadic input compared to constant current.

It is worth pointing out that the classifier ensemble does not use the iDMP reset mechanism to reset the state, unlike the connection punishing syllables that are not part of the gesture. The primary reason is that the reset mechanism is too slow to fully reset the state in the span of time that the classifier ensemble is active. The punishment mechanism cannot use inhibition because it must only affect the system state, not the representation of the observations, which are represented in the same ensemble. When resetting the iDMP, however, the observations are ignored anyhow, so inhibiting the ensemble is acceptable.

It is also worth explicitly stating that this method of temporal classification accumulates experience and classifies once the estimated state of one iDMP passes a threshold.[11] While this classification method is straightforward to implement, it may make arbitrary choices when two syllables are similar, as the representations resulting from spiking neural networks are noisy. More sophisticated classification methods could be employed using iDMPs to track estimated system states.

[11] This type of decision making is reminiscent of the well known drift diffusion model that has been used to model human behavior in two-alternative forced choice tasks (Bogacz et al., 2006).

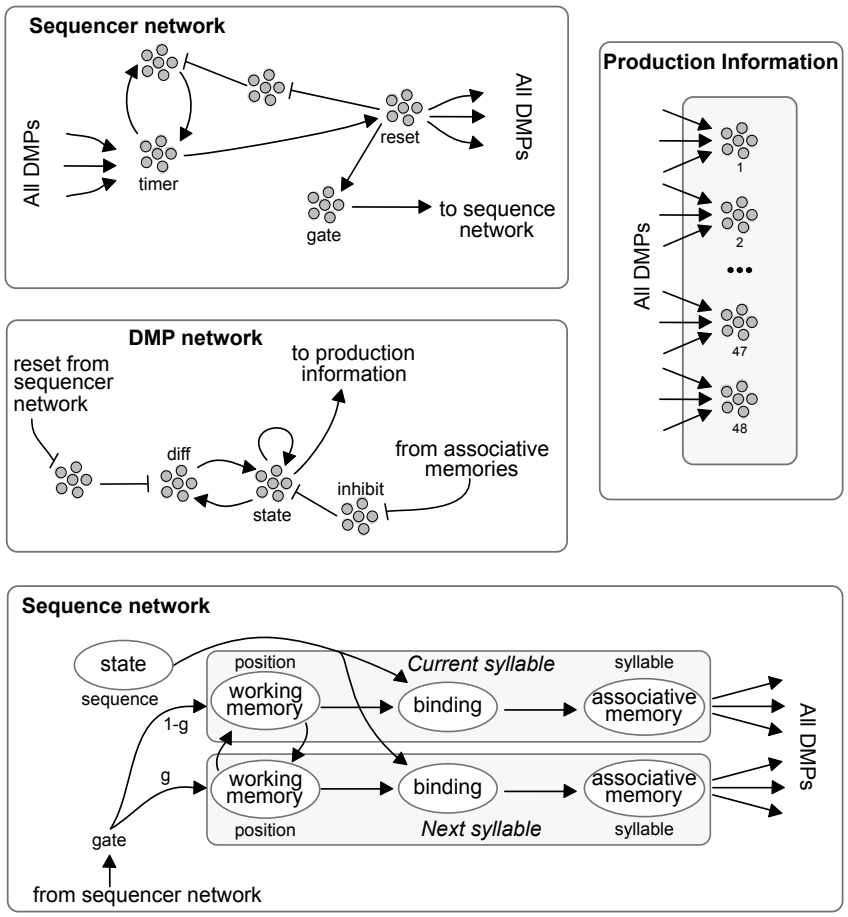


Figure 6.4: Network diagram for the syllable sequencing and production model. Lines terminated with lines, rather than arrowheads, represent inhibitory connections. The **sequence network** iterates through syllable positions in interconnected working memory modules (ovals represent SPA modules), which are switched through a connection from the sequencer network. Syllable positions are bound with the sequence and cleaned up through associative memories to represent the current and next syllable targets. DMPs associated with those syllable targets are disinhibited by the associative memories. The **sequencer network** receives input from all DMPs in order to keep track of the timing of the current syllable. When the current syllable is nearly finished, the reset ensemble is activated, resetting all disinhibited DMPs to their initial position. Working memory gates are switched when the reset ensemble activates and deactivates. When the reset ensemble is active, the timer is advanced through a recurrent connection with a separate ensemble, which is normally inhibited. **DMP networks** are primarily composed of an oscillator which represents the two-dimensional canonical system state. Production information trajectories are decoded from the system state and projected to the production information network. Input from associative memories disinhibit the oscillator; input from the reset ensemble in the sequencer reset the oscillator to an initial position. The DMP must be both disinhibited and driven to the initial position to generate a production information trajectory. The **production information network** aggregates information from all DMP networks.

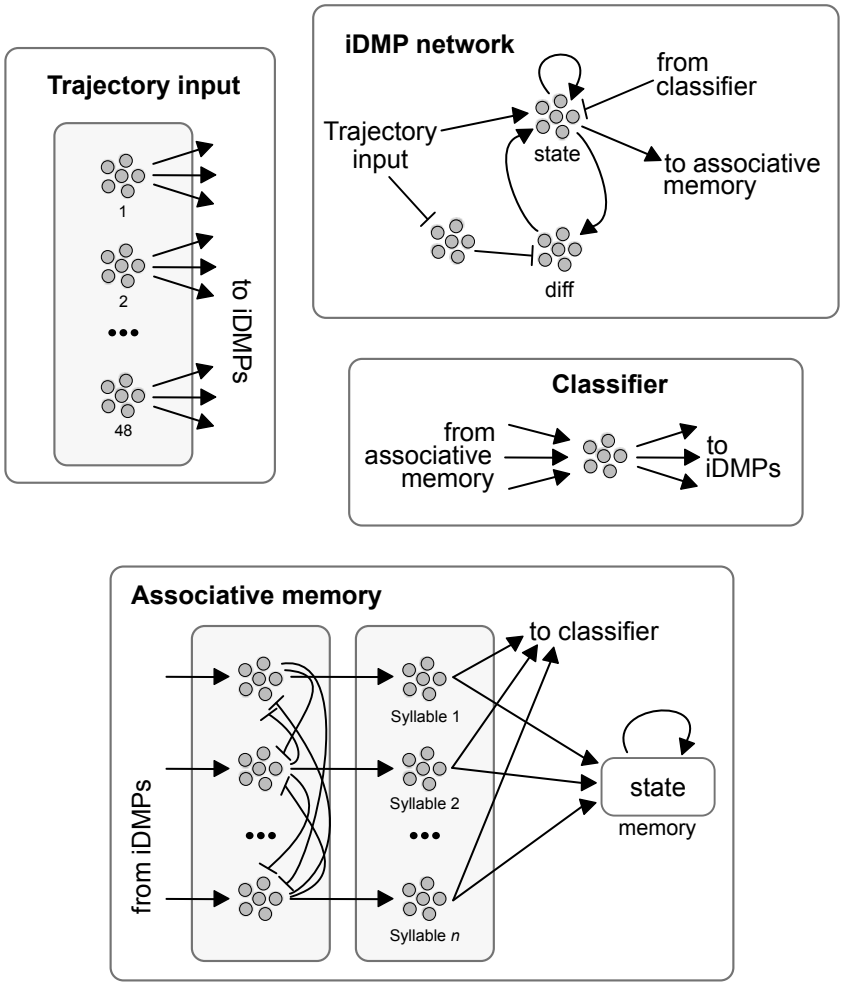


Figure 6.5: Network diagram for the syllable recognition model. Lines terminated with lines, rather than arrowheads, represent inhibitory connections. The **trajectory input** consists of an ensemble for each possible gesture. One connection is made from each ensemble to each iDMP state ensemble. If the iDMP corresponds to a syllable that uses that gesture, then it projects to a dimension of that iDMP. If the iDMP corresponds to a syllable that does not use that gesture, then it projects to the system state of the iDMP with a negative weight. Each **iDMP network** is primarily composed of a large recurrently connected state ensemble. The state ensemble's recurrent connection predicts an observation given the current system state, and advances the state if the actual observation is sufficiently close to the prediction. The system state can be pushed toward its initial state if incorrect gestures are encountered. The classifier ensemble inhibits the state ensemble once a classification is made. State output is also projected to an associative memory. The **associative memory** takes input from all iDMPs in order to emit a semantic pointer corresponding to classified syllables. The associative memory projects to a recurrently connected memory module that attempts to remember the last syllable encountered.

Chapter 7

Evaluation and results

Here, I present evaluation metrics for the three models described in Chapter 6, gather those metrics for several experimental conditions, and present the results.

7.1 Neural cepstral coefficients

We hypothesize that neural cepstral coefficients (NCCs) are a suitable feature vector representation for traditional ASR systems, and biologically grounded systems like Sermo. In order to evaluate their suitability for speech recognition tasks, we compare NCCs to the most common feature vector used currently, Mel-frequency cepstral coefficients.

7.1.1 Evaluation

While different ASR systems use the feature vector in different ways, the most important quality of the feature vector is that audio samples corresponding to a particular label (e.g., a phone or word) take a similar trajectory in the feature vector space as other samples corresponding to the same label. All samples corresponding to other labels should take trajectories that are far enough away in the vector space to provide a basis for labeling a test sample correctly. As such, we compare MFCCs and NCCs on a purely statistical level, rather than using them as a frontend in an ASR system, which would bias the results toward the aspects of the feature vector trajectories that the ASR system's backend is most suitable for detecting.

We therefore compare MFCCs and NCCs using a linear support vector machine (SVM) that is trained on labeled examples of feature vector trajectories corresponding to phones in the

TIMIT training corpus, and tested on trajectories corresponding to phones in the TIMIT testing corpus.

TIMIT (Garofolo et al., 1993) is a corpus of read speech with time-aligned phone and word transcriptions. The corpus contains ten speech samples from 630 speakers across eight dialects of American English, totaling 6300 utterances (5.4 hours of continuous speech). The utterances are separated into a training set (4620 sentences) and a testing set (1680 sentences), allowing for standardized comparisons of ASR systems. TIMIT was released in 1990, and is still a standard data set used for testing ASR systems due to the high quality manual phone transcriptions, diversity of speakers, and convenient size, as it is small enough to be computationally tractable but large enough to provide a meaningful comparison between ASR approaches.

We will focus on classifying phones in the TIMIT data set. Phones are voiced for a shorter time than words, and have high variability as phone sounds can be affected by the previous and next phones voiced, making phone classification a more difficult statistical task than word classification. Since we are not solving the full speech recognition problem, and are instead classifying pre-segmented speech samples, we anticipate that word classification would be too simple to meaningfully compare MFCCs and NCCs.

Additionally, classifying phones is advantageous because similar phones are voiced for a similar length of time, unlike words. Another way in which our statistical comparison differs from the full speech recognition problem is that we must stretch feature vector trajectories such that all trajectories are the same length, due to the constraints of SVM classifiers; therefore, we aim to minimize the amount of stretching done.

We stretch feature vector trajectories to be as long as the longest trajectory in the training and testing samples using simple linear interpolation. An alternative method for stretching the trajectory would be to use dynamic time warping (Ratanamahatana and Keogh, 2004). However, dynamic time warping involves aligning each speech sample to a representative example, which would necessitate manually determining a suitable example for each phone for each experiment. Additionally, the manual effort may not provide any concrete benefits; Ratanamahatana and Keogh showed that linearly interpolating time series data produced classification accuracy rates statistically indistinguishable from accuracy rates on time series data aligned using dynamic time warping.

TIMIT defines a set of 61 phones, which includes vowels, consonants, consonantal closures, pauses and silences. Lee and Hon (1989) identified allophones and other situations where two phones are used interchangeably, collapsing the full set of phones to a set of 39 phones, which are typically used instead of the full set. Many studies improve error rates by further categorizing the phones with similar properties; e.g., grouping together all plosives, fricatives,

and so on (Lopes and Perdigão, 2011). We will use the reduced set of 39 phones.[1] Because vowels are typically voiced longer than consonants, we will investigate whether separating them is advantageous since we lengthen all samples to the longest sample.

Support vector machines are a standard supervised learning technique often applied to classification tasks, and are particularly well suited to high-dimensional data. The SVM algorithm learns a hyperplane that separates data in one class from other classes, which it can subsequently use to classify data by evaluating where the data point lies in vector space compared to the hyperplane. When dealing with more than two classes, hyperplanes can be learned to separate each combination of two classes ("one-against-one" classification) or each class from all other classes ("one-vs-the-rest" classification). When data is relatively low dimensional, the kernel trick can be applied, which projects the input data into a higher dimensional space based on a kernel function.

In the case of speech feature vector trajectories, we have relatively many samples, classes and features, so we use a one-vs-the-rest support vector machine classifier that does not use the kernel trick (i.e., a linear SVM). We use LibLinear's fast implementation of linear SVMs (Fan et al., 2008), exposed to Python through the scikit-learn package (Pedregosa et al., 2011).

The metric that we collect in the experiments below is classification correctness, either in absolute terms, or NCC classification correctness relative to the baseline correctness obtained with MFCCs. In the ASR literature, phone error rates are often reported on TIMIT. The phone error rate takes into account incorrect classifications (substitutions), a lack of classification when there should be one (deletions) and erroneous classifications where there should not be one (insertions). However, since we pre-segment speech into short samples corresponding to some phone, we cannot have deletions or insertions, only substitutions, which corresponds to classification correctness. Since we use correctness rather than accuracy, we can only compare our approach to some reported results (Lopes and Perdigão, 2011), but even then the comparison is flawed. Hence, we generate both MFCCs and NCCs for the same speech samples in order to verify that NCCs generate feature vector trajectories that are at least as statistically separable as the equivalent MFCC trajectories.

7.1.2 Experiments and results

In each experiment, we isolate the TIMIT speech samples corresponding to the phones of interest, and generate a feature vector trajectory (i.e., MFCC or NCC) for that sample. MFCCs

[1] The set of 39 phones includes a "silence" phone, which we include when using all phones, but do not include in either the vowel or consonant phone set when testing with only vowels or consonants.

are generated using frames of 25 ms, which advance 10 ms on each timestep, as is standard in many ASR systems. NCCs are generated by running the model described in Section 6.1.2 with a 1 ms simulation timestep. Because NCC generation takes several minutes, and because the SVM fitting procedure is quadratic in the number of samples, we limit our experiments to one of the eight geographical region-specific subsets of the TIMIT dataset (the "army brat" region, which consists of a training set of 220 utterances across 22 speakers, and a test set of 110 utterances across 11 different speakers), but run each experiment 10 times with different random seeds to generate confidence intervals for each experimental condition. In the MFCC case, the feature vectors are the same on every trial; the random seed controls the stochastic nature of the supervised learning procedure used by the SVM. In the NCC case, the random seed affects both the generation of model parameters that are sampled from random distributions, and the random seed used by the SVM.

All trajectories are made to be the same length, as is required by the SVM algorithm. The number of frames used for all samples is the number of frames in the longest MFCC trajectory in the training set. Samples that are too short (e.g., most MFCC trajectories) are lengthened to match the target number of frames using linear interpolations. Samples that are too long (e.g., all NCC trajectories) are shortened to match the target number of frames by splitting the trajectory into equally spaced time bins and taking the average for each bin. See Figure 7.1 for an illustration of temporal scaling, and example MFCC and NCC trajectories.

Each trajectory is then flattened to produce a vector of length $n_{\text{cepstra}} \times n_{\text{frames}}$. Each trajectory is associated with the phone label associated with the original speech sample, producing an input-output pair that is used to train the linear SVM.

Finally, the same procedure is done with the test set to produce another set of input-output pairs. The input is fed to the linear SVM to generate phone label predictions, which are compared to the actual phones to yield a correctness measure.

Wherever not specified, the parameters used for each model are as listed in Table 7.1. In most cases, we test for correctness only on the set of consonant phones for speed reasons (a more thorough justification is given later).

Derivatives and z-scoring

Two important choices for a feature vector are whether to include derivatives, and whether to normalize the resulting vector. Typically, ASR systems include at least one derivative, and perform normalization through z-scoring (i.e., removing the mean and normalizing to unit variance). We investigated whether these choices are important for NCCs.

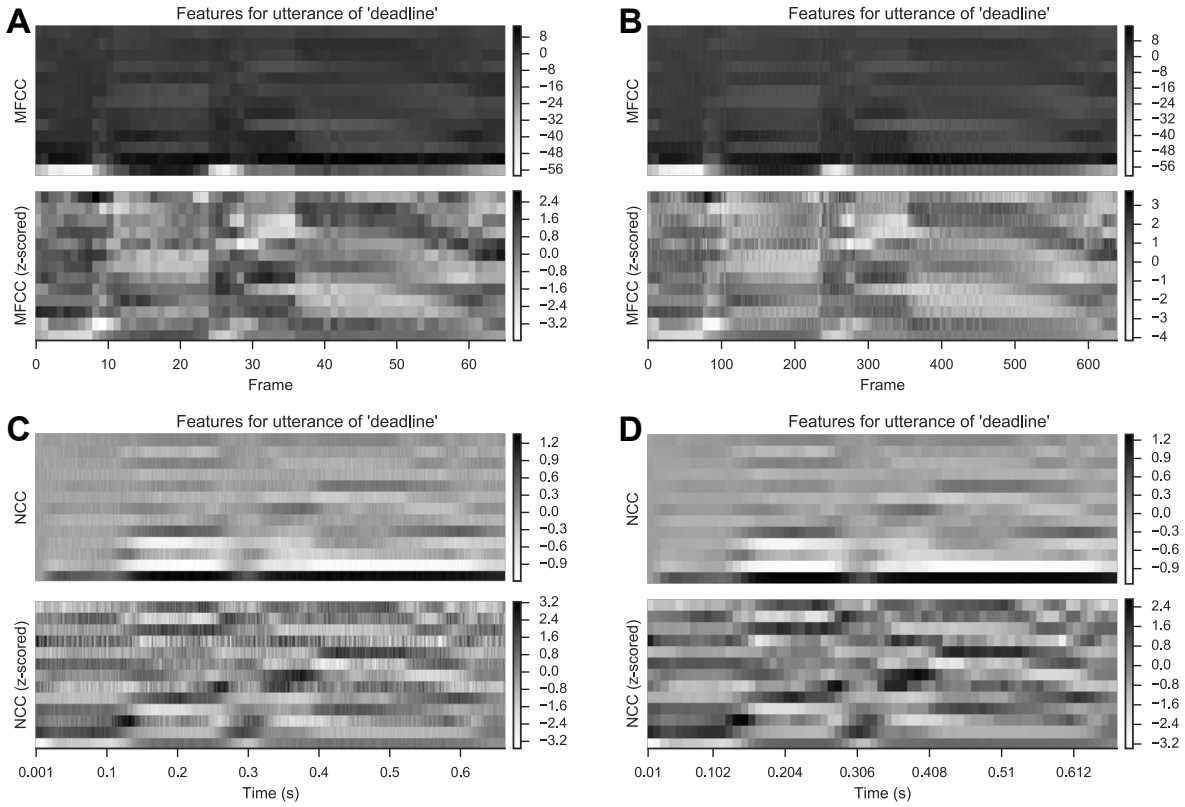


Figure 7.1: Example MFCC and NCC trajectories and temporally scaled versions. (A) MFCCs generated from an utterance of the word "deadline" from the TIMIT speech corpus with 25 ms frames advanced 10 ms per frame. Both the raw MFCC and z-scored MFCC are shown. (B) MFCCs lengthened through linear interpolation to match the same number of frames as the NCC network. (C) NCCs generated from the same utterance with 1 ms timestep. Both the raw NCC and z-scored NCC are shown. (D) NCCs shortened through neighborhood averaging to match the same number of frames as the MFCC model.

Figure 7.2 shows the training and testing correctness for ten trials with and without z-scoring. For MFCCs, z-scoring greatly reduces the variance of the training and testing correctness; the mean correctness values may be slightly lower for consonant phones, but the high variance is concerning. For NCCs, z-scoring significantly raises training correctness, but lowers testing correctness. Variance is approximately the same regardless of whether the vectors are z-scored.

We interpret these results as meaning that z-scoring is essential for MFCCs because without

Parameter	Default value	Part of model affected
`MFCC dt`	10 ms	MFCC (frame advance)
`MFCC window_dt`	25 ms	MFCC (frame size)
`n_fft`	512	MFCC
`deriv_spread`	2	MFCC (derivative)
`n_cepstra`	13	MFCC and NCC
`n_filters`	32	MFCC and NCC
`minfreq`	0	MFCC and NCC
`maxfreq`	8000	MFCC and NCC
`n_derivatives`	1	MFCC and NCC
`auditory_filter`	Gammatone	Auditory periphery
`neurons_per_freq`	8	Auditory periphery
`adaptive_neurons`	False	Auditory periphery
`cepstra n_neurons`	20	NCC cepstral coefficients
`deriv_type`	Feedforward	NCC derivative nets
`deriv_n_neurons`	20	NCC derivative nets
`deriv_tau_fast`	0.005	NCC FF derivative
`deriv_tau_slow`	0.1	NCC FF derivative

Table 7.1: Parameters used in the NCC model.

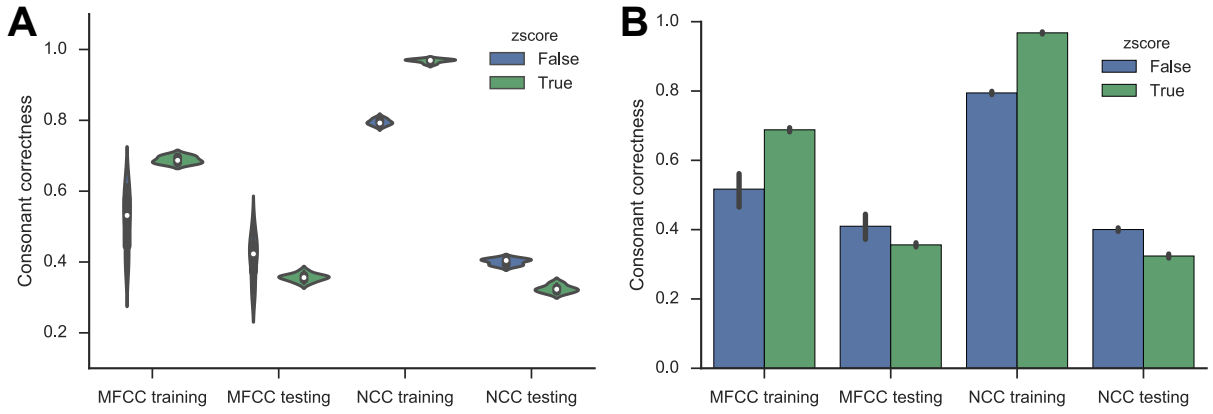


Figure 7.2: Results for the NCC experiment in which we vary whether the feature vector is z-scored or not. (A) Violin plot of the training and testing accuracy. A violin plot shows the kernel density estimation for each column, which visualizes the entire distribution. (B) Bar plot of the training and testing accuracy. Error bars represent bootstrapped 95% confidence intervals from the 10 trials.

it, the SVM fitting procedure varies significantly for different random seeds (hence the high variability without z-scoring). NCCs, on the other hand, have low variance in either condition. Since a higher test correctness indicates better generalization, we believe that z-scoring does more harm than good for NCCs. Therefore, for future simulations, we will z-score the MFCC feature vector, but not the NCC feature vector.

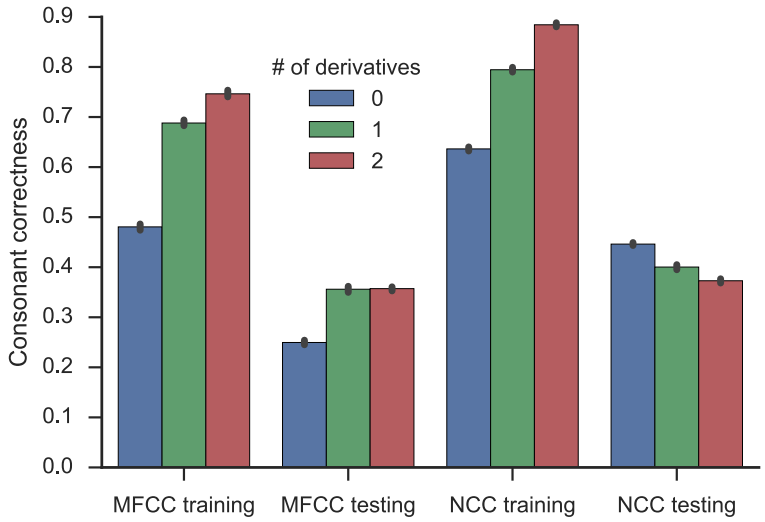


Figure 7.3: Results for the NCC experiment in which we varied the number of derivatives used in both the MFCC and NCC vectors. Error bars represent bootstrapped 95% confidence intervals from the 10 trials.

Figure 7.3 shows results varying the number of derivatives used while always z-scoring MFCCs, and never z-scoring NCCs. Note that using one derivative means that the feature vector has length 26, where the first 13 elements are cepstral coefficients, and the last 13 are derivatives of the cepstral coefficients; using two derivatives means that the feature vector has length 39, with the derivative of the derivative of the cepstral coefficients appended to the vector with 26 elements.

For MFCCs, adding a single derivative improves testing correctness significantly (from 25% to 36% correct), but adding a second derivative has very little effect (36% correct). For NCCs, adding derivatives improves training correctness, but slightly reduces testing correctness. Unlike with z-scoring, we believe it is important for the MFCC and NCC features to be the same length, so for subsequent experiments, both feature vectors will contain the first derivative (and therefore have length 26).

Number of neurons

Ideally, we would like to reduce the amount of time that the NCC generation process takes. Reducing the number of neurons used in the NCC model can speed it up, but at the cost of performance. In the next two experiments, we vary the number of neurons used in different layers of the model in order to see their impact on speed and performance.

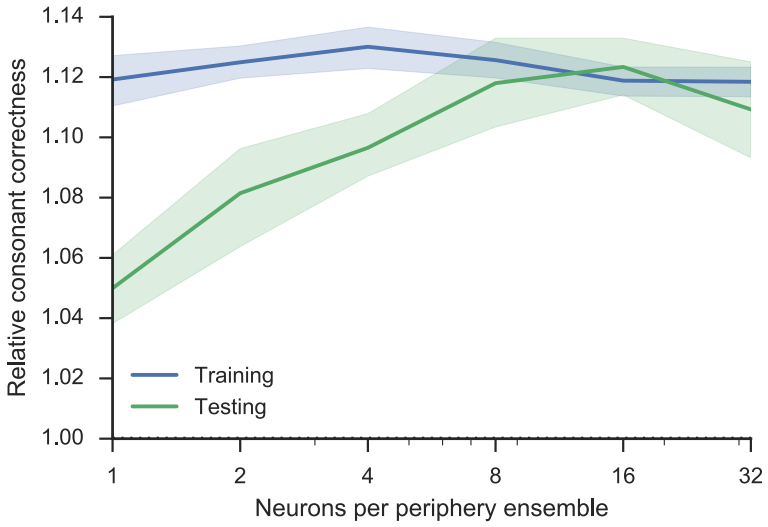


Figure 7.4: Results for the NCC experiment in which we varied the number of neurons in the auditory periphery. Shaded regions represent bootstrapped 95% confidence intervals from the 10 trials. Note that a logarithmic scale is used on the x-axis.

Figure 7.4 shows the results of varying the number of neurons in the auditory periphery layer. In this layer, each auditory filter projects to an ensemble of neurons of the size varied in this experiment. Since the representation here is a simple one (a positive scalar value), the number of neurons required should not be very large, so we vary between 1 and 32 neurons with steps at each power of two. The number of neurons used in the feature layers is fixed at 20 neurons per feature. Instead of plotting the absolute correctness, in this and subsequent experiments, we plot the correctness of the NCC model relative to the mean correctness of the same number of MFCC model instances.

As seen in Figure 7.4, training correctness does not change significantly whether we use 1 or 32 neurons neurons per filter. Testing correctness, however, improves up to using 8 neurons per filter, but then plateaus, or may even decline (possibly due to overfitting). Therefore, in all subsequent experiments, we use 8 neurons to represent the output of each auditory filter.

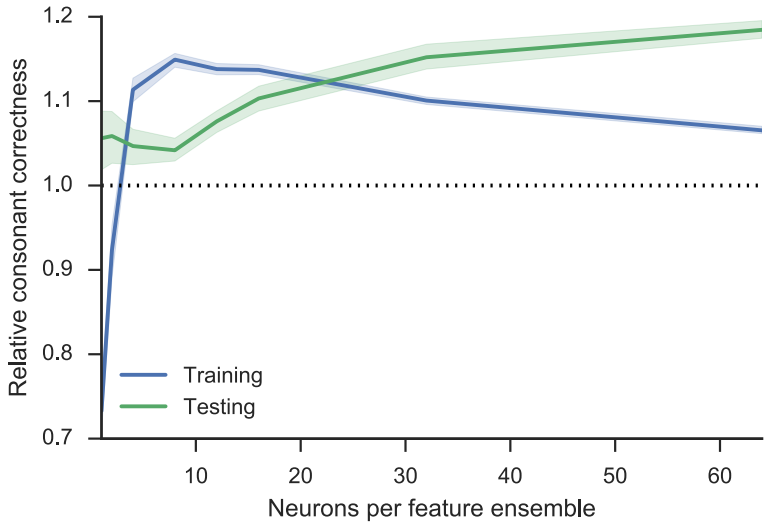


Figure 7.5: Results for the NCC experiment in which we varied the number of neurons in the ensembles representing elements of the feature vector (i.e., cepstral coefficients and the first derivative of the cepstral coefficients). Shaded regions represent bootstrapped 95% confidence intervals from the 10 trials.

Figure 7.5 shows the results of varying the number of neurons in the feature layer, which includes ensembles representing the cepstral coefficients, the ensembles representing the derivatives of those coefficients, and any additional ensembles required to compute the derivative. The number of neurons for each filter in the periphery is fixed at 8, as per the previous experiment.

A very different trend is evident when varying the number of neurons used in the feature layer. For very small number of neurons, training correctness is worse than MFCC training correctness. As the number of neurons increases, correctness improves up to using 8 neurons per ensemble, then steadily worsens. Testing correctness, however, is highly variable until we use 8 neurons per ensemble, at which point performance steadily increases up to the highest number of neurons tested, 64 neurons per ensemble.

The number of neurons in the feature layer has a big impact on the amount of time each experiment takes; see Figure 7.6. The time taken for each trial is the same up to 12 neurons per ensemble, at which point the time taken sharply increases. Since the correctness using 12 neurons per ensemble is significantly higher than the mean correctness of the equivalent MFCC feature vector, we believe that using 12 neurons per ensemble is sufficiently accurate.

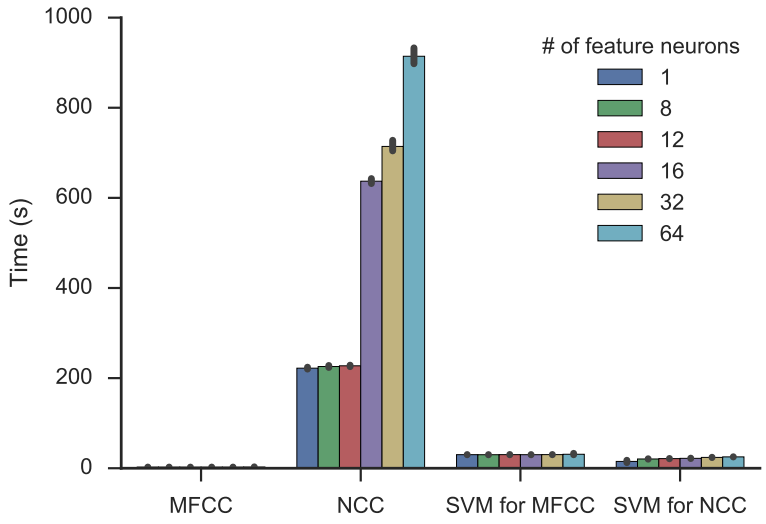


Figure 7.6: Time taken to run a trial for the NCC experiment varying the number of feature neurons. Error bars represent bootstrapped 95% confidence intervals from the 10 trials.

NCC averaging

Before the eventual goal of varying the auditory periphery model used, we address concerns about the validity of our comparison.

One concern is that the NCC model runs at a lower timestep, giving it some numerical advantages over the MFCC model; similarly, the shortening and lengthening procedures may affect the results. To examine the validity of these concerns, we performed an experiment with a fixed number of frames per audio sample, and varied the amount that the frame window advanced on each timestep. When the frame window moves at the same rate as the Nengo simulation timestep, the two models are manipulated in the same way by the shortening algorithm, which averages over a time window.

Figure 7.7 shows the consonant correctness as a function of the frame window advance in seconds. While using a lower timestep results in significantly better training correctness, testing correctness is actually better with a longer timestep, suggesting that the increased temporal resolution may cause overfitting. It is possible that this overfitting also occurs with the NCC model, but running with a high simulation timestep would result in an abundance of synchronous spiking activity and lowered decoding accuracy. We therefore conclude that the differing timesteps and temporal scaling do not introduce overt biases in favor of the NCC

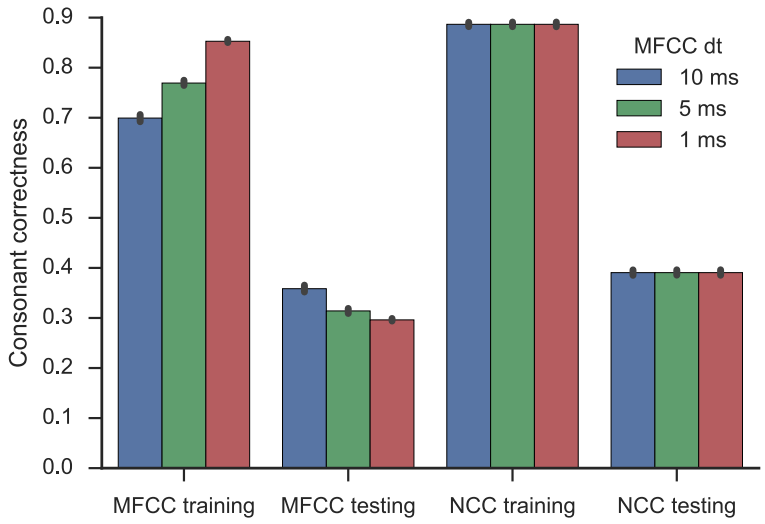


Figure 7.7: Results for the NCC experiment in which we varied the MFCC frame advance step. Error bars represent bootstrapped 95% confidence intervals from the 10 trials.

model.

Phone groups

To this point we have evaluated all of the experiments using the consonant phones under the assumption that they are more difficult to classify than vowel phones. However, it may be the case that using all phones together is a more advantageous situation for MFCCs compared to NCCs. For that reason, we compared the correctness of MFCCs and NCCs when using vowel phones, consonant phones, and all phones together (including the silent phone).

Figure 7.8 presents the results. Focusing on absolute correctness, we observe that using MFCCs yields slightly worse correctness when using all phones compared to vowel or consonant phone sets. Using NCCs yielded the same or slightly worse correctness for consonants compared to all phones. Using only consonants, therefore, is more advantageous for MFCCs than for NCCs. Looking at the relative correctness values, it can be seen that for all phone sets, NCCs perform significantly better than MFCCs.

As seen in Figure 7.9, using all phones increases the amount of time to run experiments significantly. Not only do NCCs take a longer time to generate, the increased number of samples causes the SVM fitting procedure to take a long time for both MFCCs and NCCs. Interestingly,

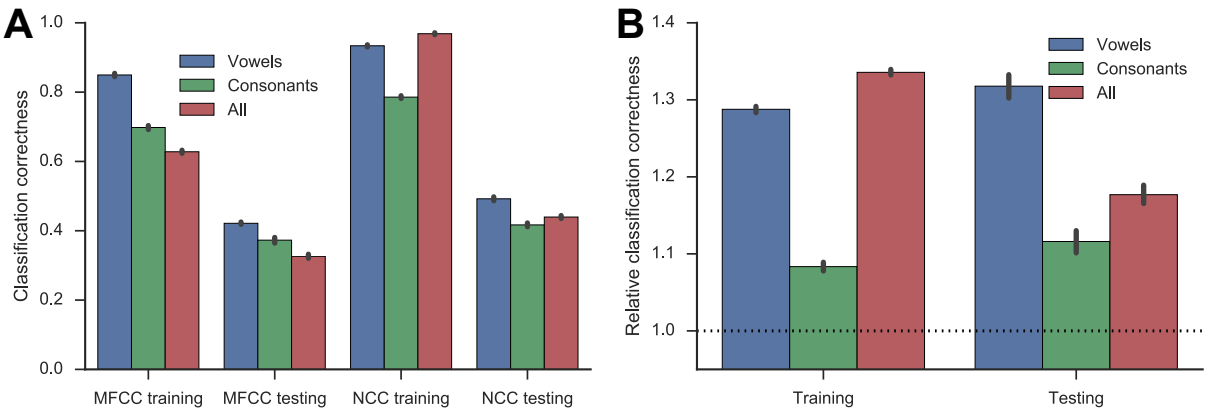


Figure 7.8: Results for the NCC experiment in which we varied the set of phones used in both training and testing. (A) Consonant correctness values for both MFCC and NCC models are shown. (B) The correctness of the NCC model relative to the mean of the MFCC models is shown. In all cases, error bars represent bootstrapped 95% confidence intervals from the 10 trials.

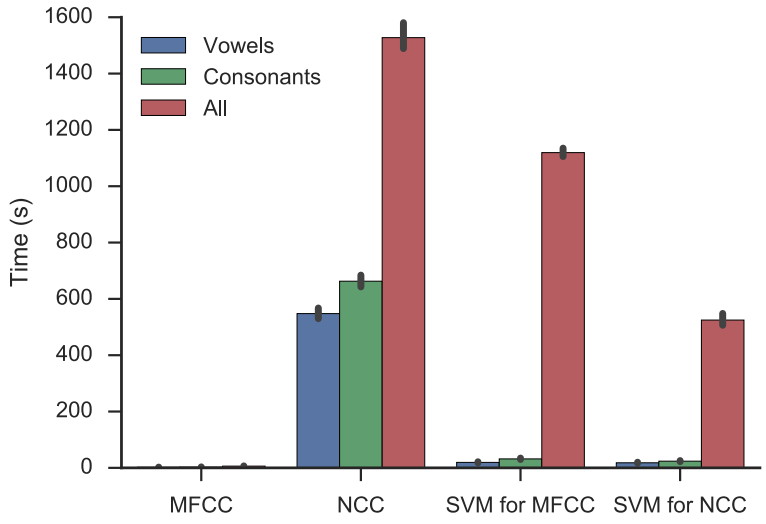


Figure 7.9: Time taken to run a trial for the NCC experiment varying the set of phones used. Error bars represent bootstrapped 95% confidence intervals from the 10 trials.

fitting for NCCs takes slightly above half the time as fitting for MFCCs (approximately 520 seconds compared to 1110 seconds for MFCCs), despite the number of samples and number of features being identical. This result indicates that it is easier to find linearly separating hyperplanes for NCCs than MFCCs.

In sum, we believe that our choice to test with consonant phones is justified, and provides the most difficult situation for NCCs compared to MFCCs.

Auditory periphery model

The final experiment varies the auditory periphery model used at the beginning of the NCC pipeline. We looked at two choices in the auditory periphery and how those choices impact classification correctness.

The first choice is the auditory filter model. We ran 20 trials with each of the five auditory filter models presented in Section 5.1 (10 with adaptive neurons, 10 with normal LIF neurons). While most of the parameters are the same as those in Table 7.1, the current implementation of the Tan Carney model can only be used for audio signals with a 50 kHz sampling rate; for this reason, all TIMIT samples in these experiments are upsampled from the recorded 16 kHz to 50 kHz, and all models (both the MFCC and NCC models) use the upsampled audio. Additionally, the Tan Carney model has numerical instabilities for auditory filters with low characteristic frequencies, so all models use frequencies evenly distributed over the Mel scale from 200 Hz to 8000 Hz.

The results are plotted in Figure 7.10. The Gammatone and Tan Carney models have the highest correctness values, with around 35% better relative testing correctness than the MFCC model. The compressive Gammachirp model also performed well, though statistically worse than the Gammatone and Tan Carney models. The Log Gammachirp and Dual Resonance models both perform significantly worse than the other three, achieving only around 20–24% more correctness than the MFCC model.

As can be seen in Figure 7.11, the compressive Gammachirp and Tan Carney models are more computationally expensive (3–4 times slower) than the other three models. As the least expensive model, the Gammatone filter provides by far the best tradeoff between classification correctness and computational cost.

We also varied whether we used normal LIF neuron or adaptive LIF neurons, as spiral ganglion cells have facilitating and depressing behavior that is partly captured in the adaptive LIF neuron. As shown in Figure 7.12, using adaptive LIF neurons had no impact on classification correctness. This result holds true even when evaluated for each auditory periphery model individually (not pictured).

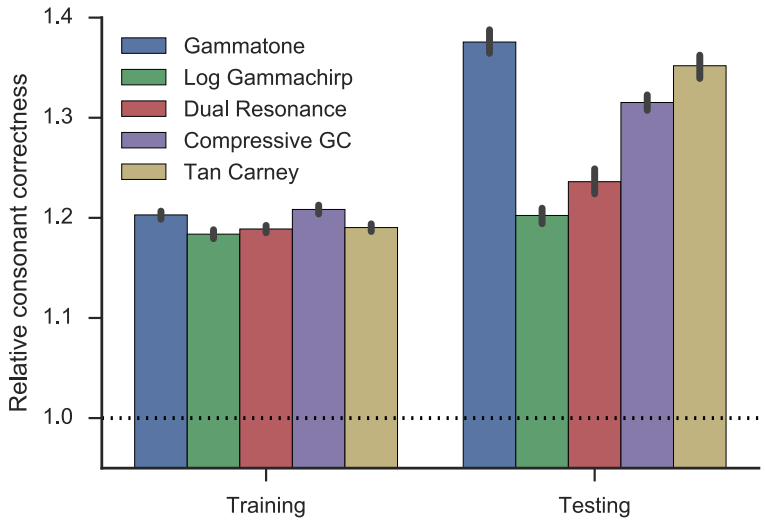


Figure 7.10: Results for the NCC experiment in which we varied the auditory periphery model, relative to the mean MFCC correctness. Compressive GC refers to the dynamic compressive Gammachirp filter. In all cases, error bars represent bootstrapped 95% confidence intervals from the 20 trials.

7.1.3 Scaling

The model used in these experiments represents a small portion of the human auditory system. The number of auditory filters and cepstral coefficients are matched to the parameters commonly used in ASR systems. However, since our model is implemented with biologically realistic parts, it is important to determine if a scaled up version of the model matches known neuroanatomical constraints.

The number of spiking neurons used in the model is

$$N = N_f|f| + N_c|c| + \sum_{i=1}^{|d|} 2N_d|c|,$$

where N_f is the number of neurons per frequency, $|f|$ is the number of frequencies modeled, N_c is the number of neurons per cepstral coefficient, $|c|$ is the number of cepstral coefficients, $|d|$ is the number of derivatives, and N_d is the number of neurons per derivative.

With the parameters used in the majority of the experiments above, N=1036 neurons. According to the BioNumbers database, the human cochlea has 3500 inner hair cells (Milo

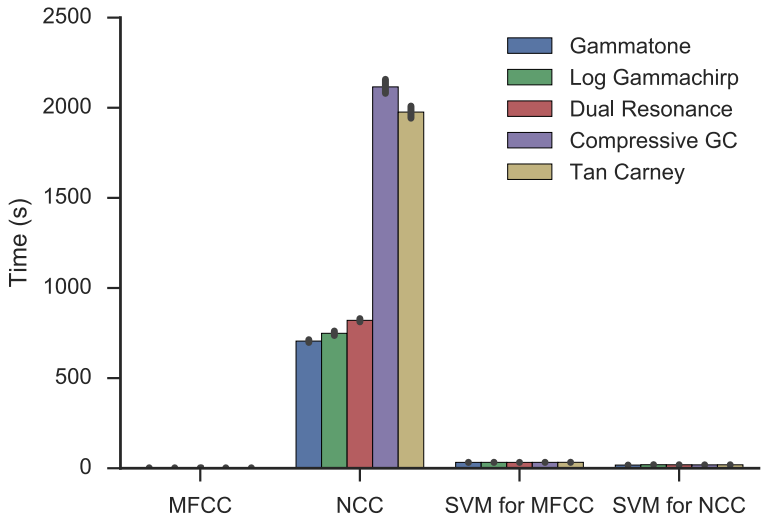


Figure 7.11: Time taken to run a trial for the NCC experiment varying the auditory periphery model. Compressive GC refers to the dynamic compressive Gammachirp filter. Error bars represent bootstrapped 95% confidence intervals from the 20 trials.

et al., 2010, BNID 100697). Assuming that each inner hair cell represents a separate auditory filter (which is not necessarily true, but a worst case assumption), we conservatively estimate that a scaled up version of the NCC model would use 20 cepstral coefficients, 1 derivative, and 50 neurons per feature, resulting in N=7.3 × 10^4 neurons. If we are somewhat more generous and use 40 cepstral coefficients, 2 derivatives, and 200 neurons per feature, we get N=1.8 × 10^5 neurons. Smiley et al. (2013) estimated that human A1 alone contains 2×10^7–3×10^7 neurons, meaning that the NCC model fits well within human neuroanatomical constraints.

7.2 Syllable sequencing and production

The goal of the syllable sequencing and production model is to generate a continuous trajectory of production information that can be used to control an articulatory synthesizer, given a static representation of a sequence of syllables. As the quality of generated speech is difficult to quantify, we evaluate this model by generating a gesture score for the decoded production information trajectory, and comparing it to the gesture score that was provided to the model as input.

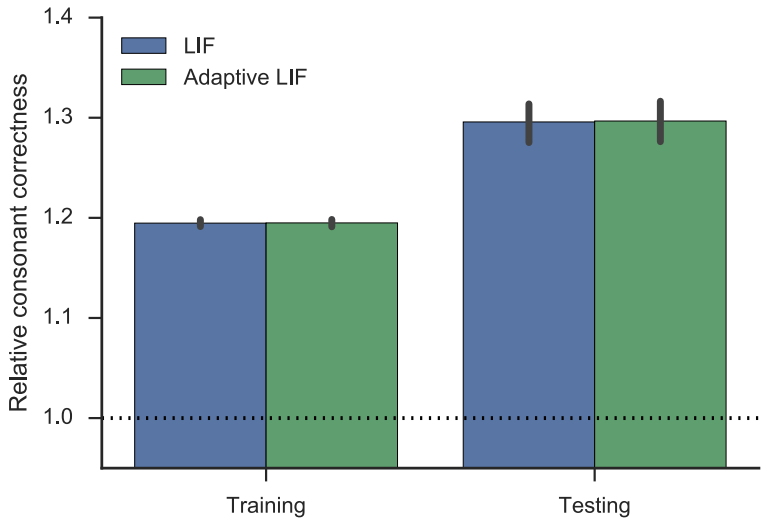


Figure 7.12: Results for the NCC experiment in which we vary the neuron type. Error bars represent bootstrapped 95% confidence intervals from the 50 trials (10 per periphery model).

7.2.1 Evaluation

While the syllable sequencing and production model can be used to control any articulatory synthesizer, we use the VocalTractLab synthesizer because of its high quality speech and available API (Birkholz, 2013). The vocal tract gesture representation in VocalTractLab is based on a set of predefined vocal tract shapes. Most gestures correspond to a particular shape, and the overall vocal tract shape is determined by linearly interpolating the shapes of all currently active gestures. Taking into account all of the possible shapes and glottal parameters, there are 48 possible vocal tract gestures (see Table 7.2) despite having only 7 articulator sets controlling 22 articulators.[2]

As discussed in Section 6.2.1, the input to the model is a semantic pointer representing a sequence of syllables. Each syllable has an associated DMP which is generated using a gesture score corresponding to a German syllable; the gesture scores are provided by Bernd Kröger.[3]

[2] There is also an f0 gesture and articulator set which is not included here, as it controls the pitch of the utterance. Since German is not a tonal language, the pitch is not required to voice a syllable, and instead is used for generating varied prosody; we do not consider prosody in this model, though including it is important future work.

[3] The gesture scores are freely available at http://www.phonetik.phoniatrie.rwth-aachen.de/bkroeger/research.htm. These German syllable gesture scores are among the few publicly available gesture scores; I was

Gesture	Value	Gesture type
a	Binary	Vowel
e	Binary	Vowel
i	Binary	Vowel
o	Binary	Vowel
u	Binary	Vowel
E:	Binary	Vowel
2	Binary	Vowel
y	Binary	Vowel
A	Binary	Vowel
I	Binary	Vowel
E	Binary	Vowel
O	Binary	Vowel
U	Binary	Vowel
9	Binary	Vowel
Y	Binary	Vowel
@	Binary	Vowel
@6	Binary	Vowel
aI-begin	Binary	Vowel
aI-end	Binary	Vowel
OY-begin	Binary	Vowel
OY-end	Binary	Vowel
aU-begin	Binary	Vowel
aU-end	Binary	Vowel
a-raw	Binary	Vowel
i-raw	Binary	Vowel
u-raw	Binary	Vowel
ll-labial-nas	Binary	Lip
ll-labial-stop	Binary	Lip
ll-labial-fric	Binary	Lip
tt-alveolar-nas	Binary	Tongue tip
tt-alveolar-stop	Binary	Tongue tip
tt-alveolar-lat	Binary	Tongue tip
tt-alveolar-fric	Binary	Tongue tip
tt-postalveolar-fric	Binary	Tongue tip
tb-palatal-fric	Binary	Tongue body
tb-velar-stop	Binary	Tongue body
tb-velar-nas	Binary	Tongue body
tb-uvular-fric	Binary	Tongue body
breathy	Binary	Glottal shape
pressed	Binary	Glottal shape
open	Binary	Glottal shape
stop	Binary	Glottal shape
modal	Binary	Glottal shape
slightly-pressed	Binary	Glottal shape
slightly-breathy	Binary	Glottal shape
fully-open	Binary	Glottal shape
velic	Scalar	Velic
lung-pressure	Scalar	Lung pressure

Table 7.2: Gestures defined by VocalTractLab. All gestures belong to a particular type, which determines the gesture sequence that they are a part of, and therefore the set of articulators that are modified by the gesture.

not able to find any gesture scores for English syllables.

Each gesture score in VocalTractLab is a collection of gestures that are parameterized by target value, onset time, gesture length, and a time constant that determines how quickly the gesture becomes active.

For each experiment described below, we chose a random set of syllables as the frequent syllables that make up the syllables in the mental syllabary of the model. A sequence of syllables of a given length are randomly chosen and assigned a speed sampled from $\mathcal{U}(0.8, 3.0)$ Hz, which corresponds to syllable trajectories of 333–1250 ms in length. Depending on the number of syllables and the randomly sampled lengths, in each trial we end up with a syllable sequence generating several seconds of production information trajectories, which in the case of these experiments, are 48-dimensional trajectories where each dimension is a gesture.

The model is provided the static syllable sequence input, and is run for the amount of time that the sequence should take to voice (determined by the randomly assigned syllable speeds), plus a small amount of time to account for variability in each model instance. The output of the temporal output associative memory and the production information ensembles are recorded.

That production information output is then converted into a gesture score by estimating the gesture parameters from the production information trajectories. Individual gestures are determined by taking the absolute value of the temporal derivative across each gesture; time slices which have high derivatives are indicative of the onset or offset of a gesture. We make a new gesture for each time slice with high derivative, using the midpoint between the start and end of the time slice as the beginning or ending time of the gesture. The last gesture ends at the end of the utterance. The time constant associated with the gesture is the length of the onset slice with high derivative. The value of scalar gestures is the average value between the two time slices with high derivatives. Every gesture has a neutral value representing that the gesture is not active; identified gestures with values sufficiently close to the neutral gestures are marked as neutral gestures.

The gesture score resulting from the model simulation is then compared to the original gesture score on three criteria. First, we calculate the accuracy of the reconstructed gesture score with the same equation as phoneme classification accuracy; specifically,

$$\text{Acc} = \frac{N_g - S - D - I}{N_g}, \tag{7.1}$$

where N_g is the number of gestures in the target gesture score, S is the number of substitutions, D is the number of deletions, and I is the number of insertions.

Therefore, we must find the number of substitutions, deletions, and insertions in our gesture score compared to a gesture score composed of the original gestures scores for the target

syllables concatenated together. In order to determine the number of insertions, deletions and substitutions, we first convert the original and reconstructed gesture scores to a sequence of characters by assigning a character to each gesture and determining their order within each articulator set, then concatenating them. Neutral gestures are recorded and denoted with the "`0`" character; numerical gestures are binarized and given either "`0`" or "`9`" depending on the numerical value relative to the gesture's range. We then do a global pairwise alignment[4] of the strings determined from the target gesture and the reconstructed gesture, which is then evaluated according to Equation (7.1).

In order to evaluate the model's accuracy scores (since we know of no comparable speech trajectory generation techniques) we compare the accuracy of the reconstructed gesture scores to a baseline calculated as the mean accuracy of all the non-matching syllables in the repository of syllables. In other words, we calculate the accuracy of all combinations of syllables in the syllable repository (except for matching syllables) and set the baseline as the mean of those accuracy measures. The result of this calculation is 0.5164. This baseline is relatively high (chance accuracy is likely much lower than 0.5164); however, consider that many syllables have the same form (e.g., CV or CVC) and are only differentiated by one or two gestures. Additionally, all syllables have a similar profile for the lung pressure gesture. If the model can perform better than this baseline, however, we can be confident in saying that it is following a similar trajectory as the target gesture score.

Since the accuracy measure only tells us that the gesture score contains the right gestures in the right order, we also aim to quantify how the temporal characteristics of the reconstructed gesture score differ from the target gesture score. For each correctly aligned gesture in the alignment computed for the accuracy measure, we record the difference between the gesture durations, and report both the mean and variance. The mean tells us about overall temporal difference, indicating that gesture durations differ in predictable ways; e.g., a high mean difference indicates that the reconstructed gesture score is slower than the target gesture score, but may still be able to produce intelligible speech. High variance, however, indicates that gesture durations vary in unpredictable ways, which is likely to be detrimental to intelligible speech.

It is critically important for speech that certain gestures either begin or end at the same time; we will call these co-occurring gestures, though they may end or start at different times. Therefore, for each pair of co-occurring gestures in the target gesture score, we record the difference in the reconstructed gesture score's corresponding co-occurrence pair, if those gestures exist. We report the average absolute difference between the start or end times of

[4] Using the Needleman-Wunsch algorithm (Needleman and Wunsch, 1970), implemented by the nwalign Python package available at `https://pypi.python.org/pypi/nwalign/`.

reconstructed gestures that co-occur in the target gesture, and compare to a baseline computed by shuffling all of the gestures and determining if two random gestures start or end at the same time.

Finally, we synthesize speech samples for each trial using VocalTractLab, and perform a qualitative evaluation of its intelligibility. Speech generated by the target gesture score is compared to that generated by the model. We do not obscure the label of the synthesized speech, as this measure is reported primarily to reinforce the other two measures; it is not the primary measure by which the model should be judged.

7.2.2 Experiments and results

Basic model operation

First, we examine the operation of a model instance in detail to ensure that the metrics we collect adequately characterize good and bad model trials.

Figure 7.13 shows a successful model trial sequencing and producing the syllables [bla ti das]; the [ti] syllable is voiced at around 2 Hz while the other two syllables are voiced around 1 Hz. The activities in various ensembles reflect the transformations and dynamics expected of the model, and result in a final production information trajectory that looks similar to the desired trajectory. Accuracy for this trial is 0.93 (2 insertions occur at the end of the trajectory); the co-occurrence metric is 0.5 (chance is 0.083).

Figure 7.14 shows an unsuccessful model trial sequencing and producing [bla ti das]. In this instance, the sequencing mechanism did not properly switch to the next syllable, resulting in a repetition of the [ti] syllable. Accuracy for this model is 0.76 (1 substitution, 2 deletions, and 4 insertions occur); the co-occurrence metric is 0.55 (chance is 0.083).

Determining model parameters

For each trial in the subsequent experiments, we randomly select a set of syllables and a syllable sequence from a repository of 417 German syllables. Of these syllables, 48 have CCV structure, 184 have CV structure, 162 have CVC structure, and 23 have V structure; we do not bias the random sample toward any of the structures, meaning that we will generally have more CV syllables than any other type in these experiments. Each randomly selected syllable has a semantic pointer, an ensemble in the associative memory, and a DMP network associated with it. The intrinsic frequency (speed) of each syllable is randomly assigned between 0.8

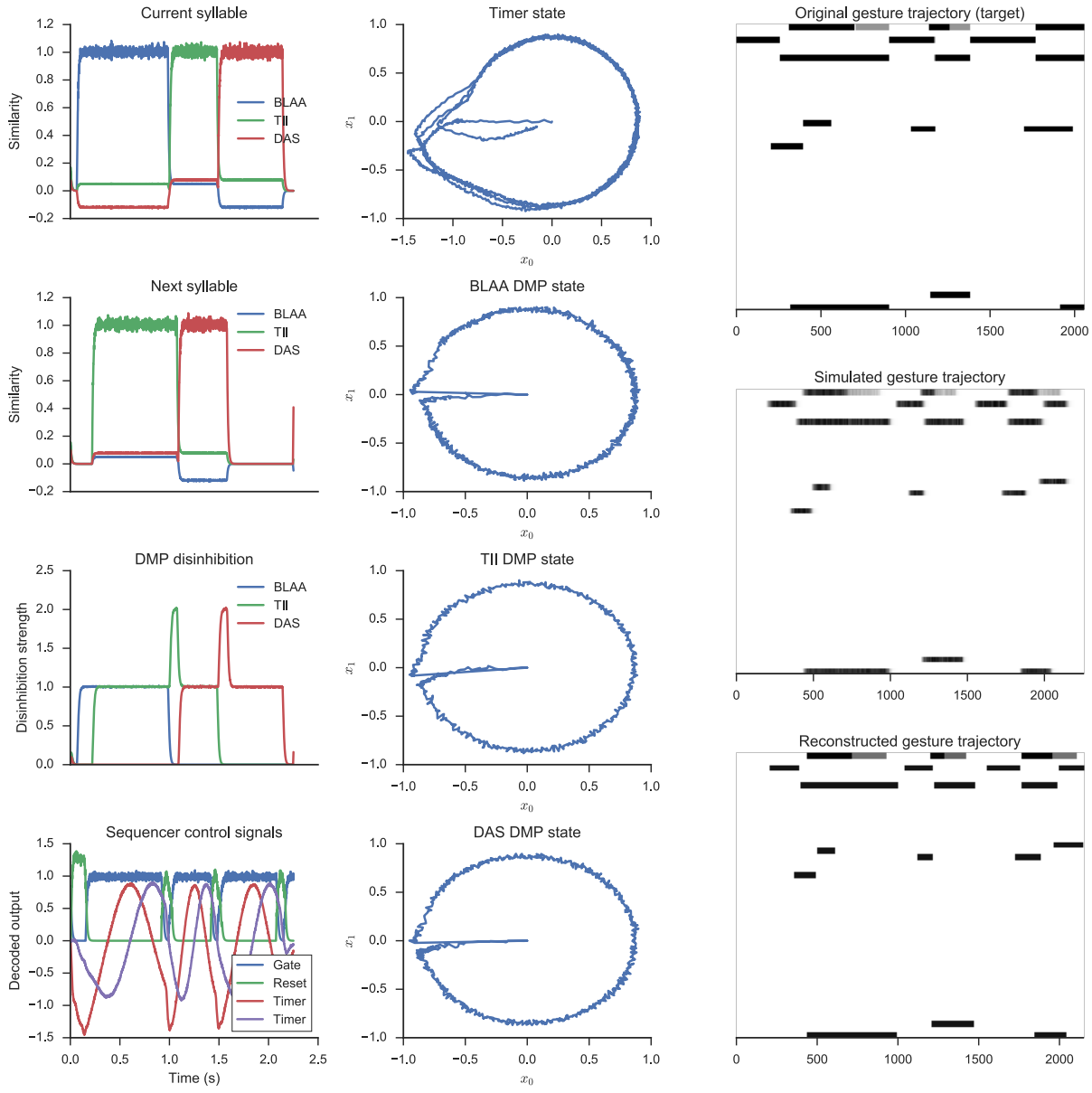


Figure 7.13: Successful example trial of the sequencing and production model. See Figure 6.4 for the network structure corresponding to the outputs plotted, and text for more details.

and 3.0 Hz, except where noted otherwise. For each trial, both the syllable repository and the

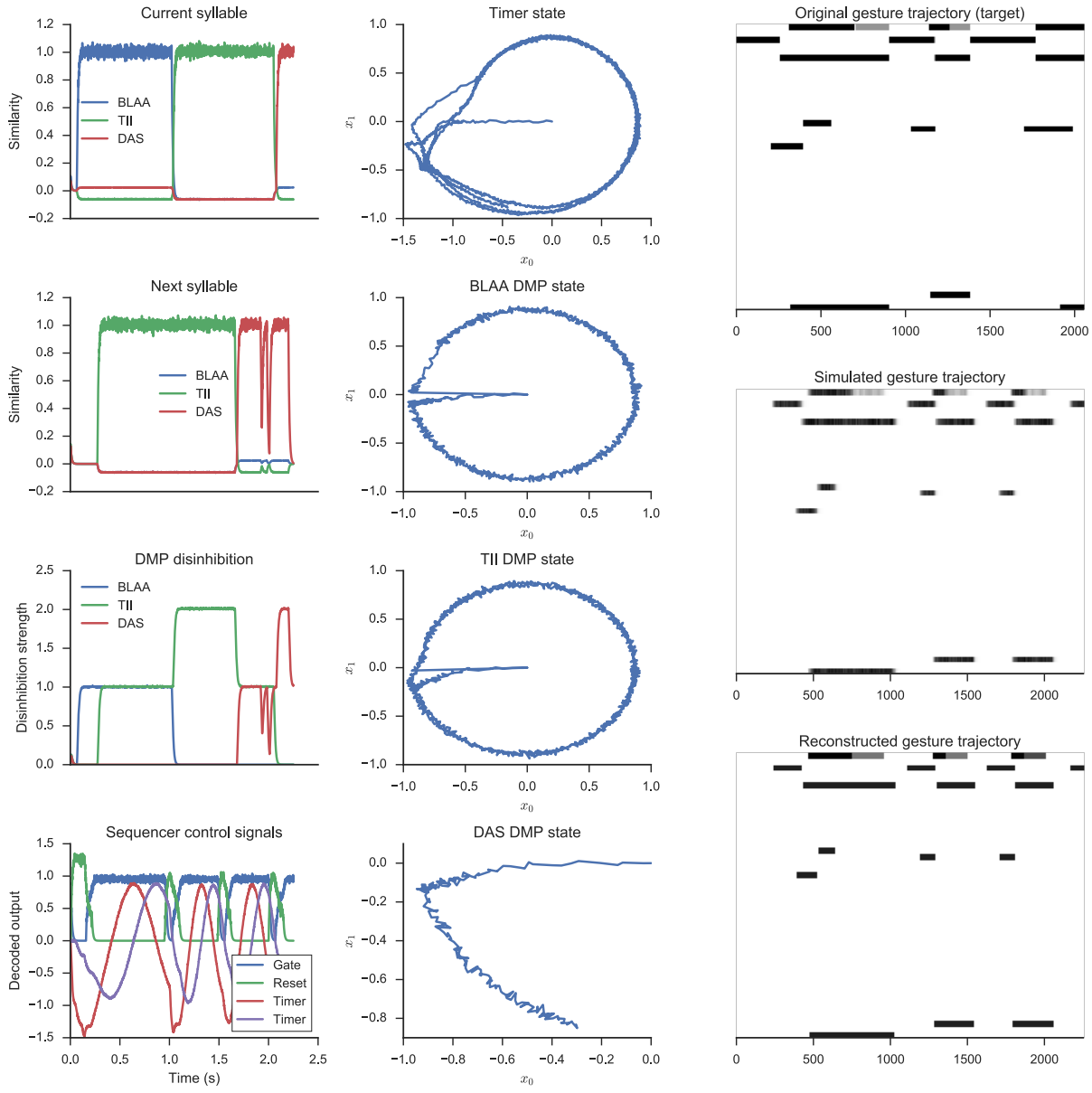


Figure 7.14: Unsuccessful example trial of the sequencing and production model. See Figure 6.4 for the network structure corresponding to the outputs plotted, and text for more details.

length and order of the syllable sequence to be produced are randomly generated, in order to obtain robust metrics not biased toward certain syllables or sequences of syllables. Other model parameters are as listed in Table 7.3, unless otherwise noted.

For each experimental condition, we ran 20 trials using a random sequence of three syllables from a repository of three random syllables, unless otherwise noted. We allow syllable repetitions to occur, but due to randomization, syllable repetitions are not guaranteed to occur.

Parameter	Default value	Part of model affected
`sequence n_per_d`	30	SPA modules in Sequence net
`syllable_d`	48	SPA modules in Sequence net
`n_positions`	7	SPA vocabulary
`difference_gain`	15	Working memories in Sequence net
`threshold_memories`	True	Associative memories in Sequence net
`sequencer n_per_d`	400	Ensembles in Sequencer net
`sequencer tau`	0.05	Timer ensemble in Sequencer net
`sequencer freq`	1 Hz	Timer ensemble in Sequencer net
`reset_time`	0.85	Timer to reset function in Sequencer net
`reset_threshold`	0.5	Timer to reset function in Sequencer net
`reset_to_gate`	0.65	Reset to gate function in Sequencer net
`gate_threshold`	0.4	Gate function in Sequencer net
`syllable n_per_d`	600	Syllable DMP ensembles
`syllable tau`	0.02	Syllable DMP recurrent connection
`syllable freq`	1 Hz	Syllable DMP recurrent connection
`prod_info threshold`	0.3	Production information net
`t_release`	0.14 s	Initial input length
`sequence`	No default	Syllable sequence
`repeat`	True	Syllable sequence generation

Table 7.3: Parameters used in the syllable sequencing and production model.

Since the dynamics of the DMP networks have a significant impact on the successful operation of the model, we begin by varying the time constant on the DMP state's recurrent connection.

As can be seen in Figure 7.15, accuracy is significantly above baseline until $\tau > 0.035$ s; poor accuracy is due to a proliferation of deletions, which are likely to due to some syllable DMPs not activating properly. Timing variance is generally low until $\tau \geq 0.04$ s. Timing means are negative, indicating that the model voices syllables faster than the original gesture score. Co-occurrence is significantly better than chance for all values of τ.

Next, as with the NCC model, we vary the number of neurons in different parts of the model

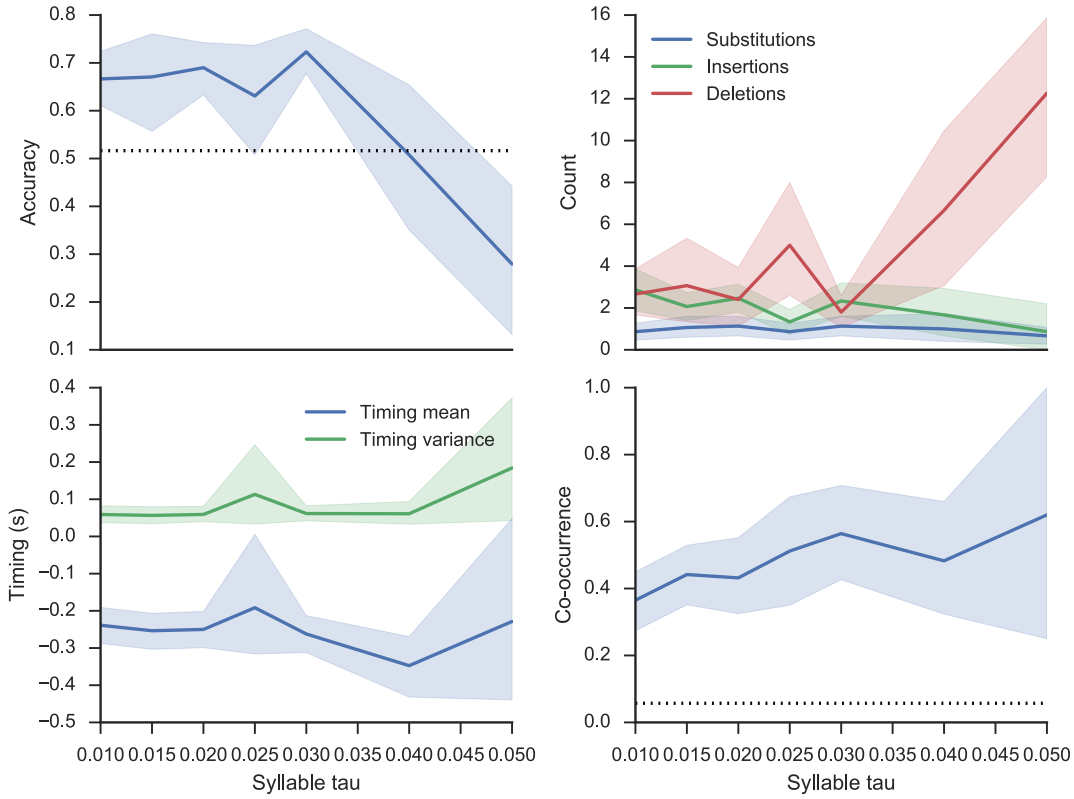


Figure 7.15: Results for the sequencing and production experiment in which we vary the τ parameter of each recurrent DMP state ensemble. Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and co-occurrence metric. The dotted line in the accuracy plot denotes the baseline accuracy value, which is the mean of the accuracy between all non-matched syllables in the repository of 417 syllables; the dotted line in the co-occurrence metric denotes the metric obtained through random chance. Shaded regions represent bootstrapped 95% confidence intervals from the 20 trials.

to speed up subsequent experiments. Specifically, we vary the number of neurons used in each DMP network and the number of neurons used in the timing ensemble in the sequencing network.

As shown in Figure 7.16, the number of neurons in each DMP network does not have an obvious effect on the model as long as at least 150 neurons per dimension are used. Since these networks implement the oscillatory dynamics, we hypothesized that using more neurons would increase accuracy; however, this hypothesis was not supported. If anything, accuracy

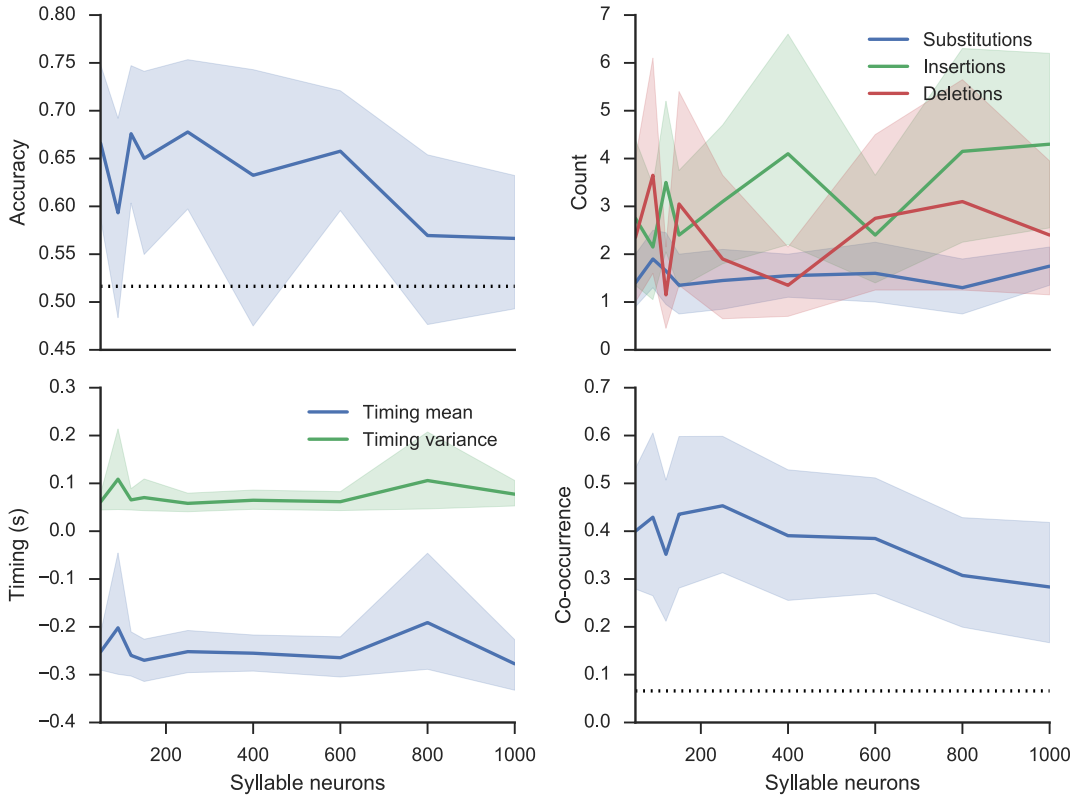


Figure 7.16: Results for the sequencing and production experiment in which we vary the number of neurons in each recurrent DMP state ensemble. Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and co-occurrence metric. The dotted line in the accuracy plot denotes the baseline accuracy value, which is the mean of the accuracy between all non-matched syllables in the repository of 417 syllables; the dotted line in the co-occurrence metric denotes the metric obtained through random chance. Shaded regions represent bootstrapped 95% confidence intervals from the 20 trials.

decreased with high numbers of neurons. For subsequent simulations, 600 neurons per dimension are used.

Figure 7.17 shows the results of varying the number of neurons in the timer ensemble in the sequencer network. Unlike the number of neurons in DMP networks, increasing the number of neurons in the timer ensemble has more clear results. Using very few neurons (less than 30 per dimension) results in performance significantly worse than baseline. Increasing the number of neurons generally increases accuracy, up to the maximum number tested, 1000 neurons per

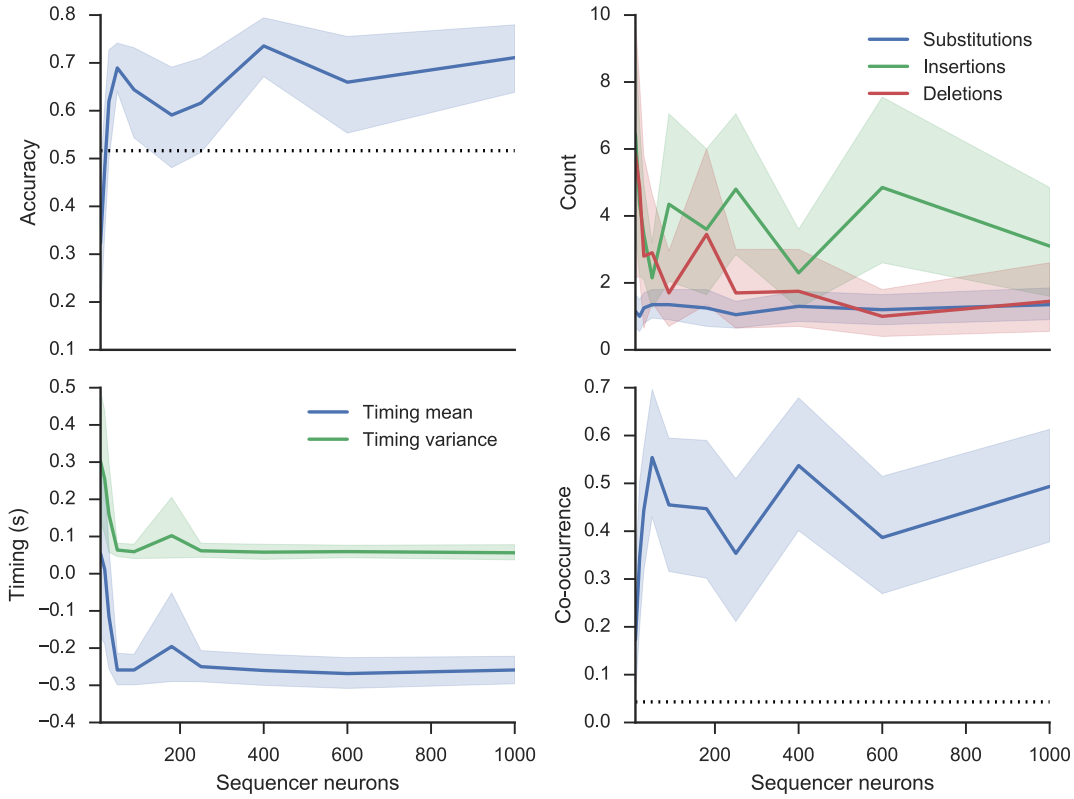


Figure 7.17: Results for the sequencing and production experiment in which we vary the number of neurons in the timing ensemble in the sequencer network (see Figure 6.4). Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and co-occurrence metric. The dotted line in the accuracy plot denotes the baseline accuracy value, which is the mean of the accuracy between all non-matched syllables in the repository of 417 syllables; the dotted line in the co-occurrence metric denotes the metric obtained through random chance. Shaded regions represent bootstrapped 95% confidence intervals from the 20 trials.

dimension, which is the value used in subsequent experiments.

Scaling to natural speech

With a reasonable set of parameters chosen, we focus on whether the model can scale to natural speech situations, which involve a large repository of syllables voiced rapidly.

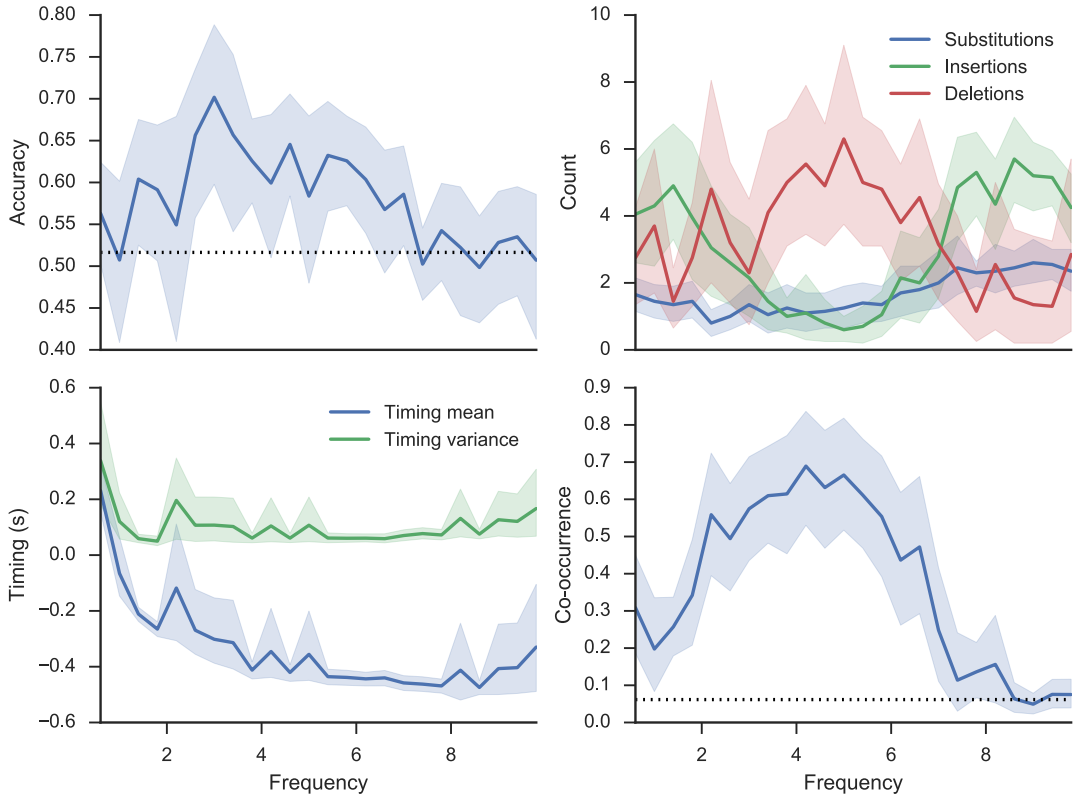


Figure 7.18: Results for the sequencing and production experiment in which we vary the frequency (i.e., speed) of syllables in the syllable repository. Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and co-occurrence metric. The dotted line in the accuracy plot denotes the baseline accuracy value, which is the mean of the accuracy between all non-matched syllables in the repository of 417 syllables; the dotted line in the co-occurrence metric denotes the metric obtained through random chance. Shaded regions represent bootstrapped 95% confidence intervals from the 20 trials.

First, we investigated the range of frequencies at which the model can produce accurate trajectories. In this experiment, we use a random sequence of three syllables from a repository of three random syllables with the same frequency. Figure 7.18 shows the accuracy of the model as a function of the frequency. Interestingly, the model is least accurate for both low and high frequencies. At low frequencies, accuracy is around the baseline, with relatively high numbers of insertions, suggesting that the DMP networks may be activated more than once when the sequencer network is operating at slow speeds. Accuracy is better than baseline from around

2.5–6.5 Hz (i.e., syllables of 150–400 ms in length), which matches the range of frequencies often used in normal speech. The network starts to break down at higher frequencies, again with a relatively high number of insertions.

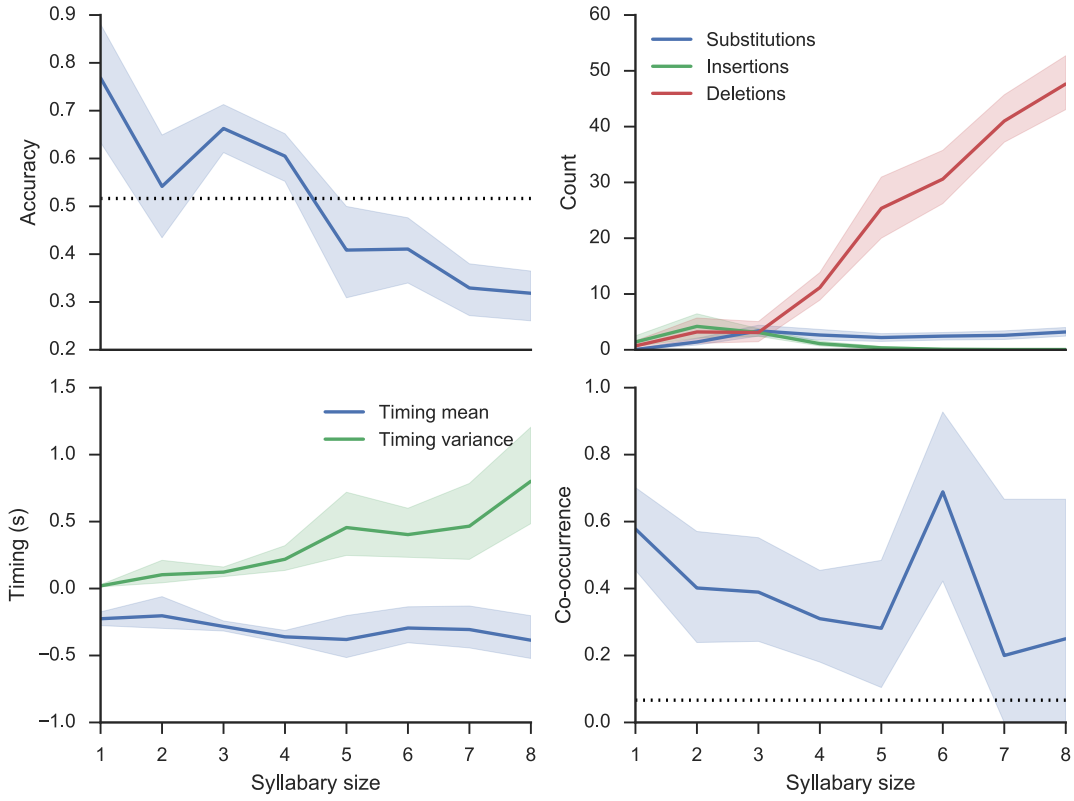


Figure 7.19: Results for the sequencing and production experiment in which we vary the syllabary size (i.e., the number of syllables in the syllable repository of the model). Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and co-occurrence metric. The dotted line in the accuracy plot denotes the baseline accuracy value, which is the mean of the accuracy between all non-matched syllables in the repository of 417 syllables; the dotted line in the co-occurrence metric denotes the metric obtained through random chance. Shaded regions represent bootstrapped 95% confidence intervals from the 20 trials.

Next, we investigated whether the model behavior changes when the size of the syllabary changes. Since adults have a repository of hundreds or thousands of commonly uttered syllables, the size of the syllabary should have little effect on the model. However, as is shown

in Figure 7.19, model accuracy decreases as the size of the syllabary increases, only being more accurate than baseline for up to four syllables, due to a sharp increase in the number of deletions. We have not yet determined the source of these deletions, but believe that the SPA modules are the source of poor performance, as we did not increase the dimensionality of the semantic pointers as the number of syllables increased.

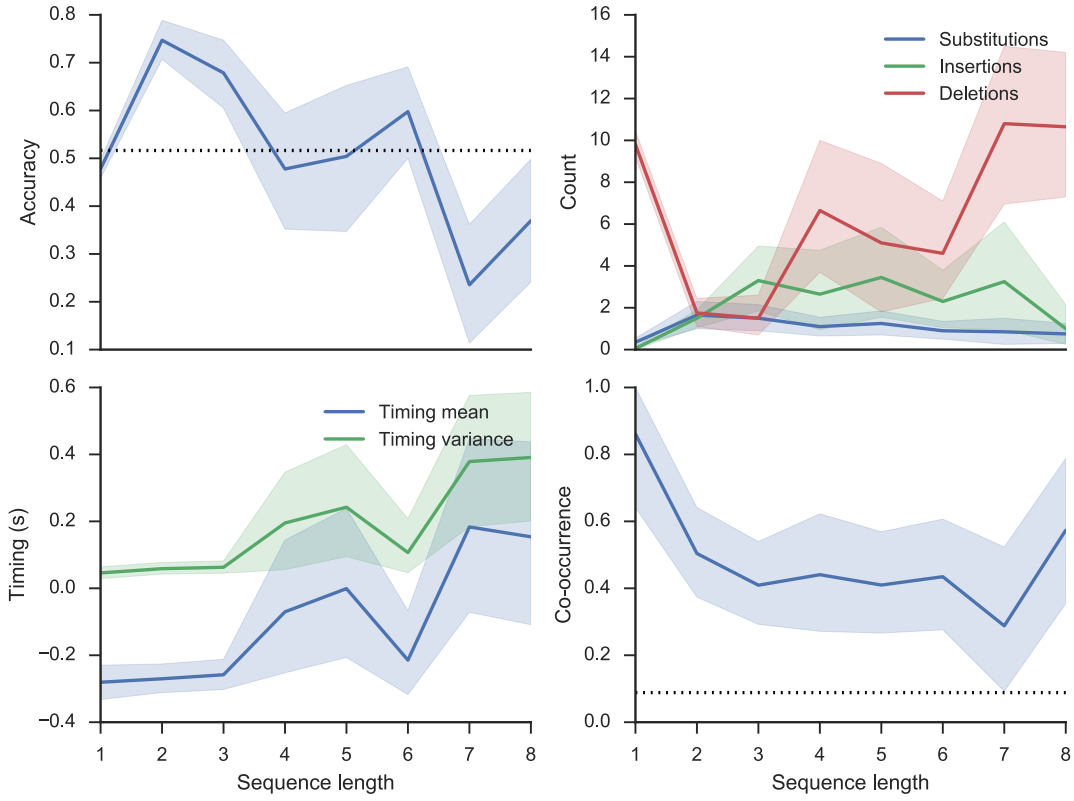


Figure 7.20: Results for the sequencing and production experiment in which we vary the number of syllables voiced in an experimental trial. Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and co-occurrence metric. The dotted line in the accuracy plot denotes the baseline accuracy value, which is the mean of the accuracy between all non-matched syllables in the repository of 417 syllables; the dotted line in the co-occurrence metric denotes the metric obtained through random chance. Shaded regions represent bootstrapped 95% confidence intervals from the 20 trials.

In a similar vein, we investigated whether model behavior depends on the length of the syllable sequence, with the size of the syllabary fixed. In this case, we know from Choo (2010)

that sequences of length five or greater tend to have difficulty with elements in the middle of the sequence. As shown in Figure 7.20, accuracy drops off after only four syllables. Interestingly, accuracy is also low for sequences of length one, due to increased deletions.

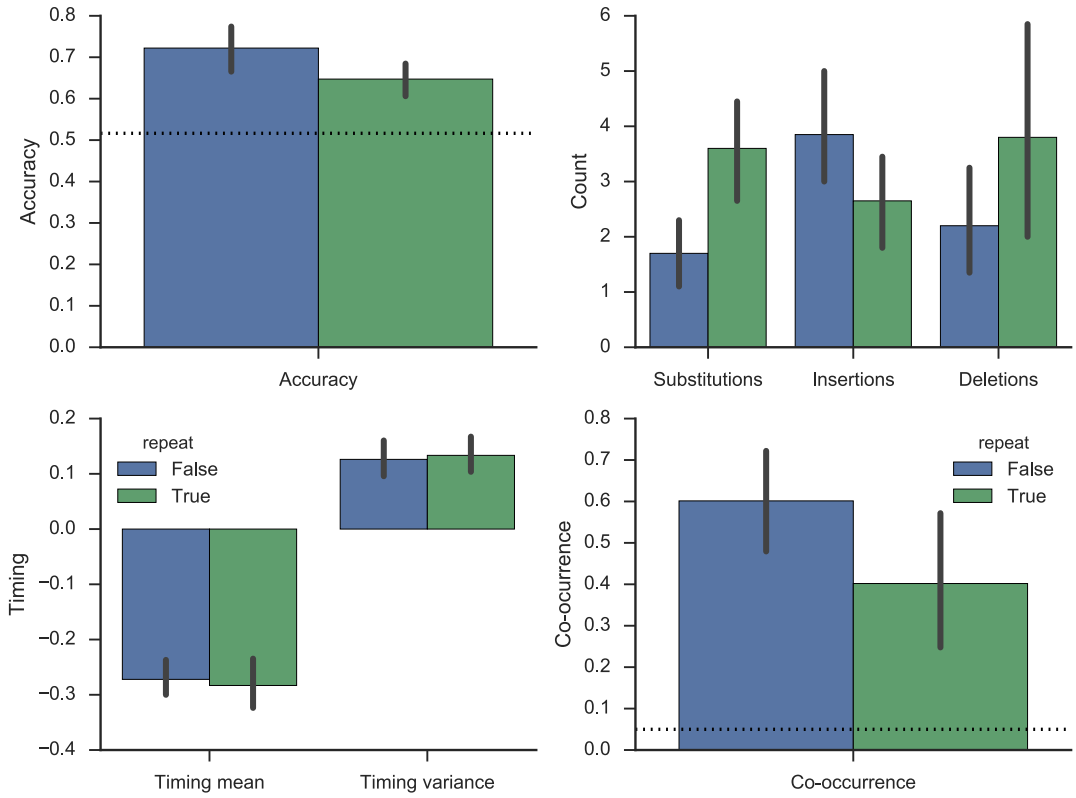


Figure 7.21: Results for the sequencing and production experiment in which we allow syllables to repeat, or constrain the sequence to contain only unique syllables. Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and co-occurrence metric. The dotted line in the accuracy plot denotes the baseline accuracy value, which is the mean of the accuracy between all non-matched syllables in the repository of 417 syllables; the dotted line in the co-occurrence metric denotes the metric obtained through random chance. Error bars represent bootstrapped 95% confidence intervals from the 20 trials.

Finally, to this point we have allowed syllables to repeat under the assumption that our network design (specifically the choice to use rhythmic DMPs) results in similar performance when repeating syllables compared to a sequence of unique syllables. Therefore, we investigated whether performance is impacted when sequences of length three are constrained to

be unique. As can be seen in Figure 7.21, the model performs the same whether syllables are allowed to repeat or not, verifying our assumption. The only statistically significant difference between the model with repetitions versus no repetitions is that substitution are more likely to occur when repetitions are allowed.

Speech intelligibility

While manually checking the synthesized speech for each experimental trial would take a significant amount of time, it is nevertheless important to ensure that the gesture scores reconstructed from the model outputs can yield intelligible speech.

To test this, we manually evaluated one set of experimental trials (specifically, those associated with the experiment varying the number of sequencer neurons, with 1000 neurons used). I and one other person evaluated the set of 20 utterances independently, and reevaluated utterances in which our evaluations differed. For the 20 trials, we found that almost all of them were cut off; that is, not all of the syllables in the sequence were present in the synthesized speech. This could be because the model was not run for a sufficient length of time, or because the last syllable DMP in the sequence did not activate despite the network running for a sufficient length of time. Regardless, since all trials contained sequences of three syllables, we compared the target synthesized syllables to the syllables that were present in the speech generated by the model. We found that

- 7 utterances (35%) were as intelligible as the target,
- 8 utterances (40%) were not as intelligible as the target, but still resembled it, and
- 5 utterances (25%) were not intelligible.

The model, therefore, is capable of producing intelligible speech with syllable sequences of varying speed.

7.2.3 Scaling

The number of neurons in the sequencing and production model is

$$N = N_{\text{SPA}} + N_S + N_{\text{DMP}} + N_O,$$

where these variables are the number of neurons in a subset of the model. N_{SPA} is the number of neurons for all SPA modules, which is

$$N_{\text{SPA}} = 5D(\text{SYLL})N_{D(\text{SYLL})} + |DFT_{\text{SYLL}}|N_{D(\text{SYLL})} + 2N_{\text{AM}}(3|\text{SYLL}| + 1),$$

where $D(\text{SYLL})$ is the dimensionality of syllable semantic pointers, $N_{D(\text{SYLL})}$ is the number of neurons per syllable dimension, $|DFT_{\text{SYLL}}|$ is the number of discrete Fourier transform coefficients used by the binding operator, N_{AM} is the number of neurons per associative memory ensemble, and |SYLL| is the number of syllables in the syllabary.

N_S is the number of neurons in the sequencer network, which is

$$N_S = 4N_{D(S)} + 140,$$

where $N_{D(S)}$ is the number of neurons per dimension in the sequencer network, and 140 neurons come from ensembles of fixed size.

N_{DMP} is the number of neurons in DMP networks, which is

$$N_{\text{DMP}} = 3|\text{SYLL}|N_{|\text{DMP}|} + 40,$$

where $N_{|\text{DMP}|}$ is the number of neurons per DMP dimension, and 40 neurons come from ensembles of fixed size.

N_O is the number of neurons in production information ensembles, which is

$$N_O = 48N_{|O|}$$

where 48 is the number of gestures, and $N_{|O|}$ is the number of neurons per production information ensemble.

With the parameters used in the majority of the experiments above, N=22180 neurons, with most of those neurons in the SPA populations (N_{SPA}=14200) and the DMP ensembles (N_{DMP}=5520). Fortunately, these parts of the model grow at a linear or sublinear rate as additional syllables are added to the syllabary. If we assume a conservative syllabary with around 1000 syllables, and increase the dimensionality of each syllable to 256, the model uses N=2.2×10^6 neurons. If we assume a more generous estimate with 2000 syllables and use 512 dimensions per syllable semantic pointer, then the model uses N=4.4×10^6 neurons. Given that there are approximately 2.7×10^4 neurons per mm^3 (on average) in human cortex (Milo et al., 2010, BNID 112050), a human scale syllabary would take up approximately 163.7 mm^3 of cortex (0.1637 cm^3) in the generous case, which is a very small portion of cortex, especially considering that this model would be distributed over several cortical areas.

7.3 Syllable recognition

The goal of the syllable recognition system is to temporally classify syllables given a continuous trajectory of production information. We will evaluate the system based on its ability to emit correct and timely syllable classifications, and whether those classifications are available in working memory between classifications.

7.3.1 Evaluation

The infrastructure for evaluating the recognition model is similar to that of the production model. We use the same library of German syllable gesture scores, from which we create 48-dimensional gesture trajectories. On each trial, we generate a trajectory by concatenating the trajectories of a randomly chosen sequence of trajectories together (i.e., we do not use the syllable production model to generate input for the model). That trajectory is provided to all of the iDMPs as input, and we record the output of the iDMP state ensembles and the memory module.

From the data collected, we compute three metrics to evaluate performance in different situations. First, we calculate classification accuracy using the same formula as is used to calculate accuracy in the production model; specifically, we generate a list of all of the syllables classified by the model, and compare that list to the originally specified list using the formula

$$\text{Acc} = \frac{N_S - S - D - I}{N_S},$$

where N_S is the number of syllables in the trajectory, S is the number of substitutions, D is the number of deletions, and I is the number of insertions. A substitution occurs when the syllable is misclassified. A deletion occurs when a syllable trajectory has been followed, but the model does not emit a classification. An insertion occurs when the model emits a classification when a corresponding trajectory has not been followed (e.g., it classifies the same syllable twice).

Second, we record the times at which syllable classifications occur, and compare those times to the end of the actual syllable trajectories. Here, we are attempting to determine if the model can classify syllables while they are being voiced, or if the entire syllable must complete before the classification can be made.

Third, we analyze the contents of the memory module in between syllable presentations. Specifically, we find the semantic pointer with the maximum similarity to the memory representation at the midpoint between the start and the end of a syllable for all syllables in

the trajectory (except the first, which cannot be remembered until it occurs). The memory representation of the trial is perfect if the previously classified syllable is in memory at the midpoint of the next syllable.

7.3.2 Experiments and results

Basic model operation

First, we examine the operation of a model instance in detail to ensure that the metrics we collect adequately characterize good and bad model trials.

Figure 7.22 shows a successful model trial recognizing the syllables [bla ti das]; all syllables are voiced at around 2 Hz. The activities in various ensembles reflect the transformations and dynamics expected of the model, and result in perfect classification and memory accuracy.

Figure 7.23 shows an unsuccessful model trial recognizing [bla ti das]. In this case, while the initial [bla ti] syllables are recognized successfully, an erroneous additional [ti] classification occurs. Because of this erroneous classification, the iDMPs are reset when the [das] syllable begins, causing it to not be classified. Therefore, there are two correct classifications and one substitution (the [das] is replaced with the erroneous [ti]) resulting in 66.7% accuracy.

Determining model parameters

The basic experimental setup in the following experiments is similar to that of the production model. However, instead of using a sequence of syllables as input and producing a production information trajectory, the recognition model uses a production information trajectory as input and produces a sequence of syllable classifications. Again, for each trial in each experiment, we randomly select a set of syllables and a syllable sequence from which to generate a trajectory. The random selection process is the same as in the production model. The randomly selected gesture scores are combined into a single trajectory, and that trajectory is provided to the network through a Nengo node. The intrinsic frequency (speed) of each syllable is randomly assigned between 1.3 and 1.5 Hz, except where noted otherwise. For each experimental condition, we ran 20 trials using a trajectory made from three random syllables in a repository of three syllables, meaning that syllable repetitions can occur, but are not guaranteed. Wherever not specified, we use parameters are defined in Table 7.4.

In the case of the recognition model, two parameters other than the number of neurons have the most significant effect on model behavior. First is the similarity threshold in Equation (6.3). As can be seen in Figure 7.24, classification accuracy peaks with similarity threshold

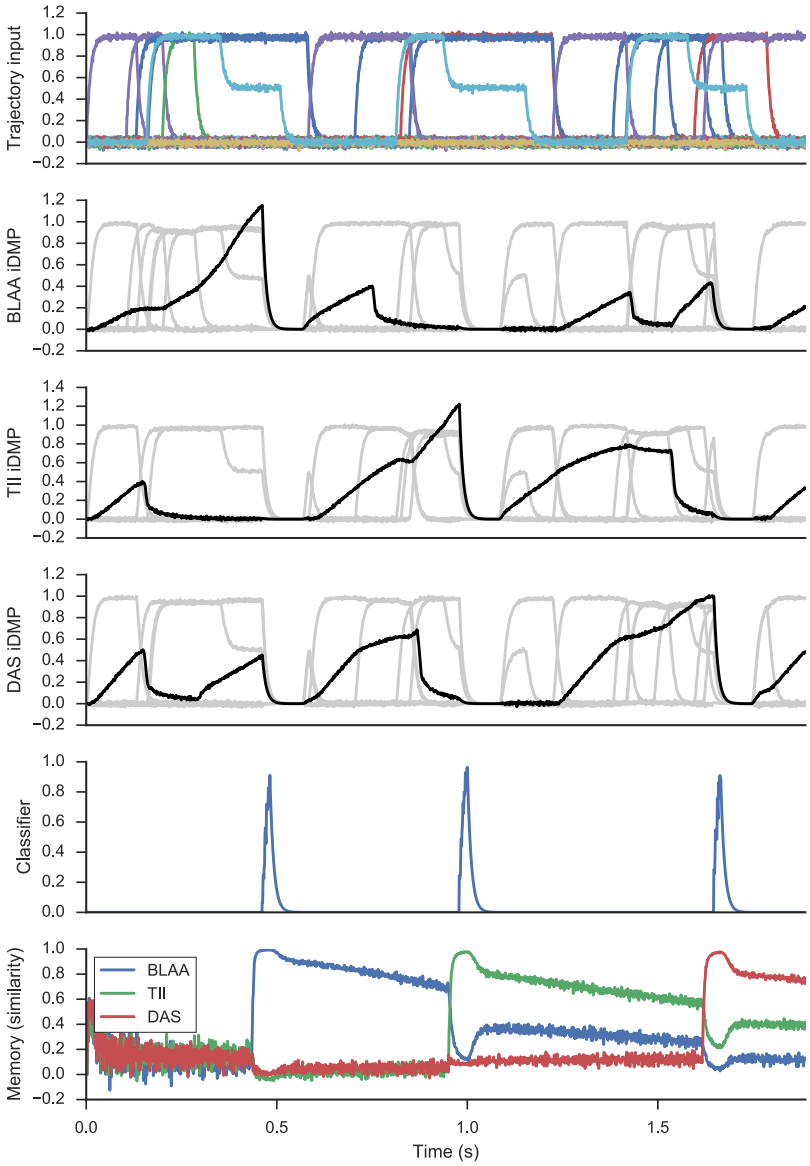


Figure 7.22: Successful example trial of the syllable recognition model. For iDMPs, the dimension representing the system state is emphasized to illustrate state tracking. See Figure 6.5 for the network structure corresponding to the outputs plotted, and text for more details.

around 0.8. Up to 0.8, the number of insertions is high, while after 0.8 the number of deletions

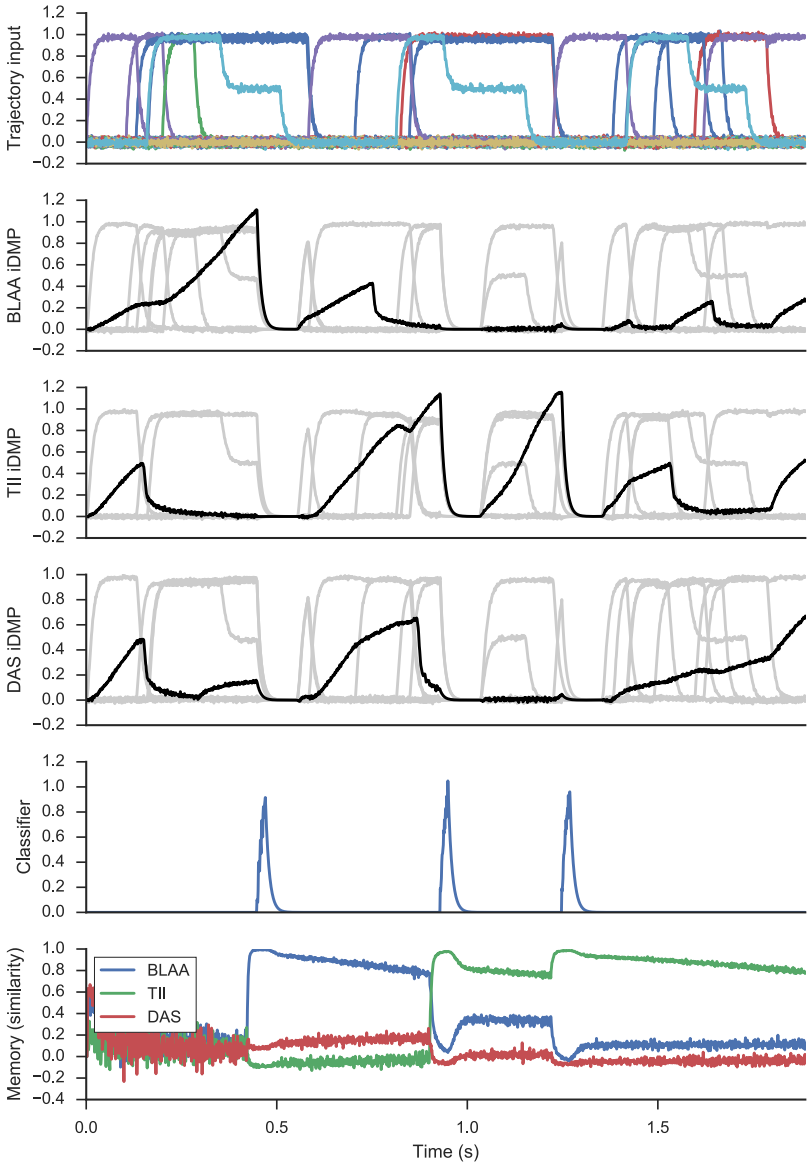


Figure 7.23: Unsuccessful example trial of the syllable recognition model. For iDMPs, the dimension representing the system state is emphasized to illustrate state tracking. See Figure 6.5 for the network structure corresponding to the outputs plotted, and text for more details.

increases. Since insertions do not always harm the memory representation, memory accuracy

Parameter	Default value	Part of model affected
`dimensions`	64	Associative memory
`threshold`	0.9	Associative memory
`wta_inhibit_scale`	3	Associative memory
`memory neurons_per_dimension`	60	Memory
`memory feedback`	0.83	Memory recurrent connection
`reset_th`	0.9	Classifier ensemble
`inhib_scale`	1	Classifier connections to iDMPs
`syllable n_per_d`	1000	Syllable iDMP ensembles
`syllable tau`	0.05	Syllable iDMP recurrent connection
`similarity_th`	0.82	Syllable iDMP recurrent connection
`scale`	0.71	Syllable iDMP recurrent connection
`reset_scale`	2.5	Syllable iDMP diff connection
`trajectory`	No default	Input trajectory
`repeat`	True	Input trajectory generation

Table 7.4: Parameters used in the syllable recognition model.

is high up to the peak of around 0.82, at which point it sharply decreases. Insertions are difficult to avoid; however, deletions can be dealt with through other parameters, so we choose a similarity threshold of 0.82 for subsequent experiments.

The next parameter of interest is the scale on the iDMP system state dynamics (α in Equation (6.3)). As can be seen in Figure 7.25, with similarity threshold of 0.82, the scale makes a smaller difference in classification and memory accuracy than does the similarity threshold. Deletions are likely up to a scale value to 0.8, after which insertions become more likely. Again, since deletions can be handled through other means, we err on the side of avoiding insertions, and choose a scale value of 0.71 for subsequent experiments.

Finally, we vary the number of neurons used in each iDMP network. As shown in Figure 7.26, the number of neurons in each iDMP network has a significant impact on the performance of the overall model. Classification accuracy is relatively low until around 300 neurons per dimension, and continues to increase to the maximum size attempted, 1000 neurons per dimension. As the number of neuron increases, the probability of deletions decreases, as an ensemble with more neurons can better approximate Equation (6.3). Memory accuracy with 1000 neurons per dimension is especially high (nearly 90%), so we use 1000 neurons per dimensions in subsequent experiments. Note that using more than 1000 neurons per dimension may give slight increases, but with 1000 neurons per dimension the number of deletions is already as low as the number of insertions and substitutions, implying that further improvements would be modest.

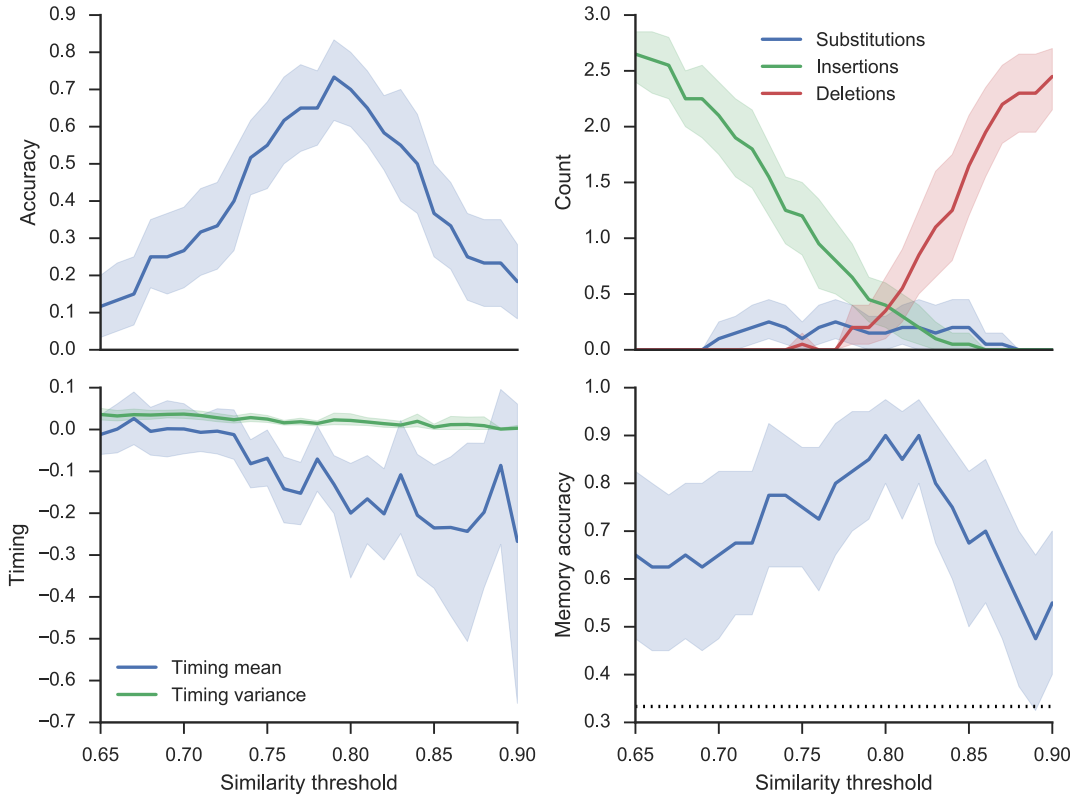


Figure 7.24: Results for the syllable recognition experiment in which we vary the similarity threshold in the function computed through the recurrent connections on iDMP state ensembles. Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and memory accuracy. The dotted line in the memory accuracy plot denotes the chance accuracy value, which depends on the syllabary size. Shaded regions represent bootstrapped 95% confidence intervals from the 20 trials.

Scaling to natural speech

As in the production network, we next vary experimental parameters to examine how the model scales to more realistic speech.

First, we investigated the range of frequencies in which the model can successfully classify trajectories. In this experiment, we use a random sequence of three syllables from a repository of three random syllables with the same frequency. Originally, we hypothesized that the model would be especially weak in this experiment, as the values for the similarity threshold and

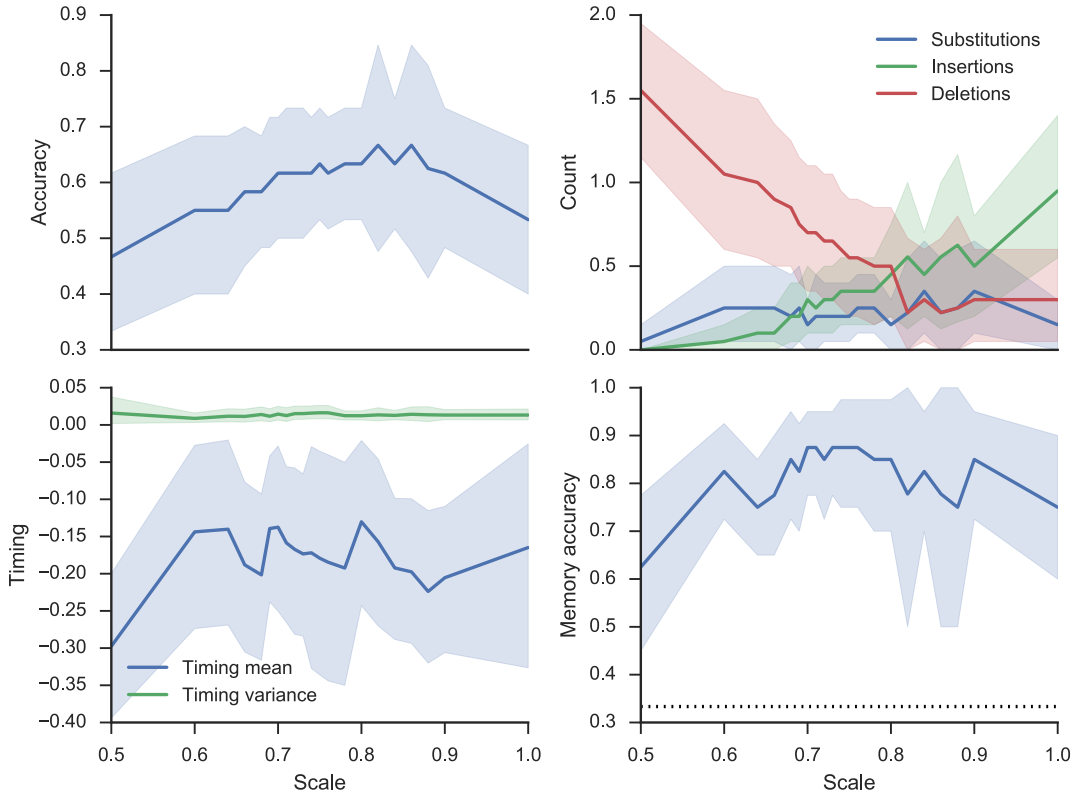


Figure 7.25: Results for the syllable recognition experiment in which we vary the scale term in the function computed through the recurrent connections on iDMP state ensembles. Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and memory accuracy. The dotted line in the memory accuracy plot denotes the chance accuracy value, which depends on the syllabary size. Shaded regions represent bootstrapped 95% confidence intervals from the 20 trials.

scale were chosen based on experiments with 1.3–1.5 Hz. As can be seen in Figure 7.27, our hypothesis was correct in the sense that the recognition model is more sensitive to frequency changes than the production model. Classification accuracy is initially poor; however, this is due to a large number of insertions (likely insertions of the correct syllable). Maximal accuracy around 75% occurs at 1.8 Hz, just above the frequency that we optimized for earlier. After 1.8 Hz, accuracy decreases as deletions become more common than insertions. The variance of the timing difference (i.e., when the classification occurs relative to the end of the syllable) is very small from about 1.4 Hz and above. The mean timing difference is negative for all

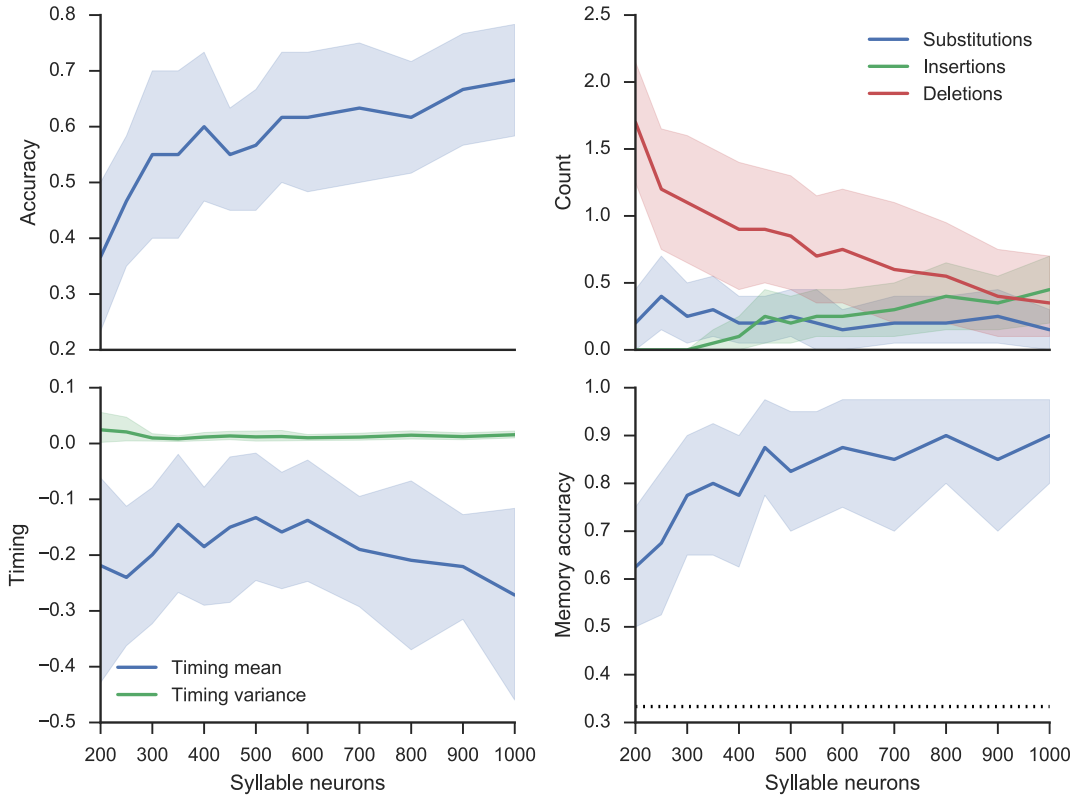


Figure 7.26: Results for the syllable recognition experiment in which we vary the number of neurons per dimension in the iDMP state ensembles. Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and memory accuracy. The dotted line in the memory accuracy plot denotes the chance accuracy value, which depends on the syllabary size. Shaded regions represent bootstrapped 95% confidence intervals from the 20 trials.

frequencies, indicating that classifications occur before the syllable trajectory is complete.

We then investigated whether the model behavior changes when the size of the syllabary changes. As shown in Figure 7.28, accuracy drops off quickly when more than five syllables are part of the syllabary. Unlike in the production system, this effect is predictable from the structure of the network, due to the competitive nature of the iDMPs; all iDMPs are constantly tracking the system state, except when a classification has occurred and the iDMP networks are inhibited. As the number of syllables in the repository increases, the chance of substitutions increases, as can be seen in Figure 7.28. Memory accuracy, however, remains significantly

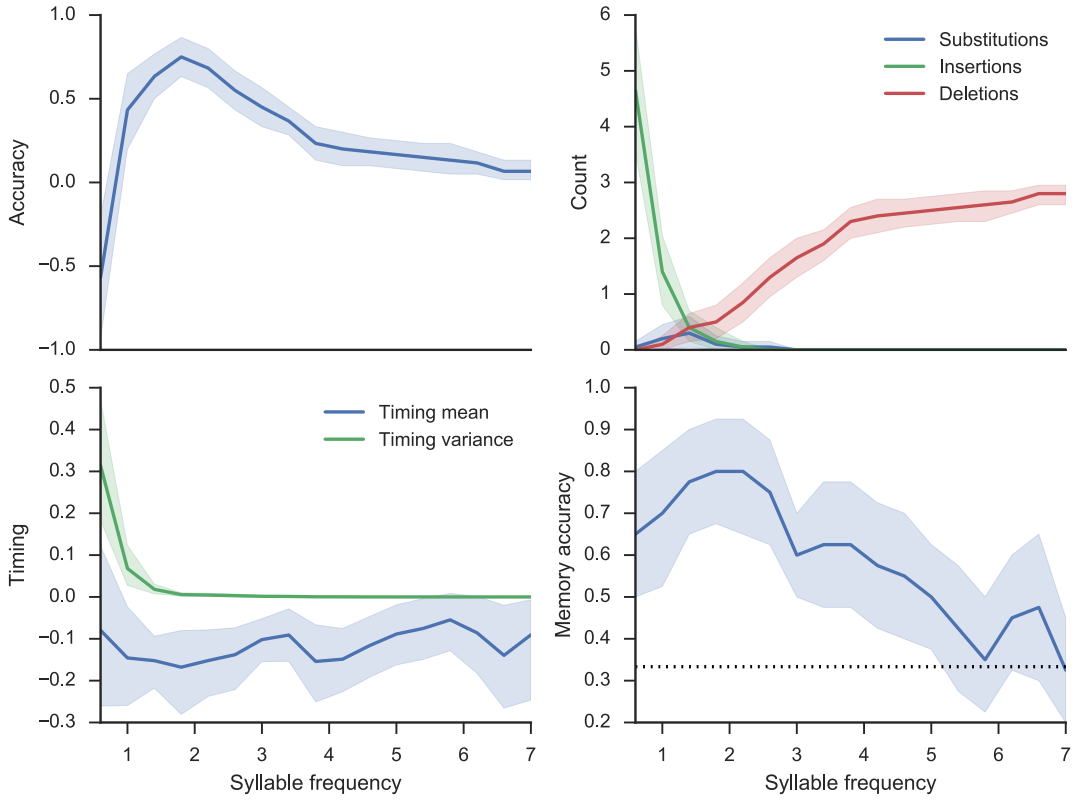


Figure 7.27: Results for the syllable recognition experiment in which we vary the frequency of the syllables that make up the input production information trajectory. Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and memory accuracy. The dotted line in the memory accuracy plot denotes the chance accuracy value, which depends on the syllabary size. Shaded regions represent bootstrapped 95% confidence intervals from the 20 trials.

above chance in all cases.

The length of the sequence, on the other hand, has little effect on classification accuracy, as can be seen in Figure 7.29. In this case, the trajectories of the three syllables in the syllabary are repeated several times. The iDMP networks continue to operate at the same accuracy rate regardless of how long they have been active, which is a benefit of this method over HMMs, which must be renormalized and reset often. However, as seen in Figure 7.29, it may be the case that memory accuracy would decrease if the sequence length is increased further, due to the fact that the memory is implemented with a simple decaying integrator. If longer

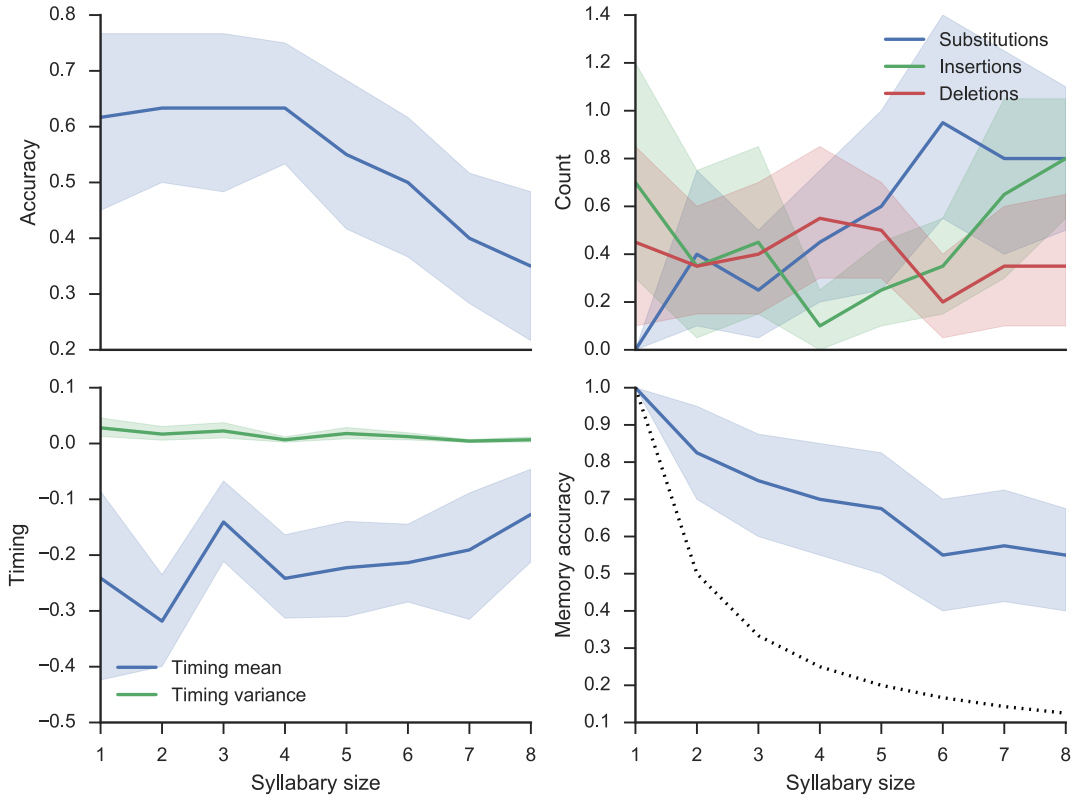


Figure 7.28: Results for the syllable recognition experiment in which we vary the size of the syllabary from which we select three syllables to recognize. Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and memory accuracy. The dotted line in the memory accuracy plot denotes the chance accuracy value, which depends on the syllabary size. Shaded regions represent bootstrapped 95% confidence intervals from the 20 trials.

sequences prove to be problematic, it may be necessary to replace the memory with an input-gated working memory module, as is used in the production system for representing syllable positions.

Finally, we verified that the model operates well in situations with sequences of unique or repeated syllables. As shown in Figure 7.30, the overall accuracy is indistinguishable whether repeated syllables are allowed or not. All other measure are within confidence intervals as well.

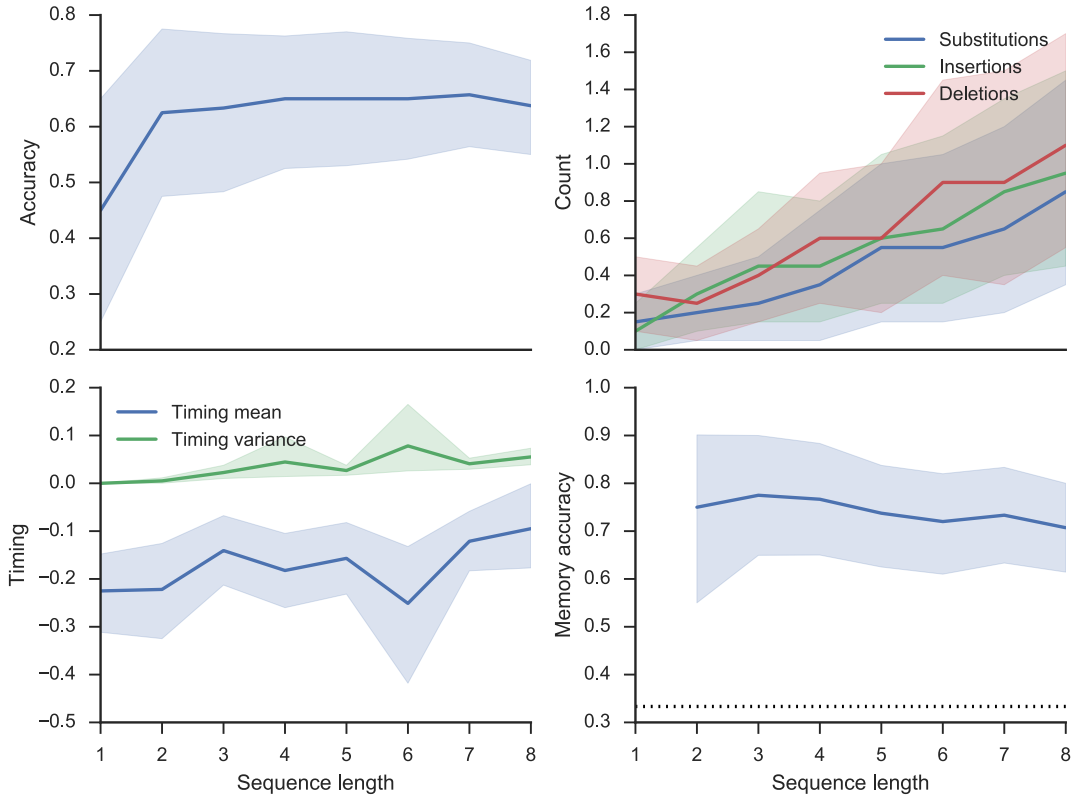


Figure 7.29: Results for the syllable recognition experiment in which we vary the number of syllables selected from the syllabary and concatenated together to make up the input trajectory. Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and memory accuracy. The dotted line in the memory accuracy plot denotes the chance accuracy value, which depends on the syllabary size. Shaded regions represent bootstrapped 95% confidence intervals from the 20 trials.

7.3.3 Scaling

The number of neurons in the syllable recognition model is

$$N = N_{\text{in}} + N_{\text{SPA}} + N_{\text{iDMP}} + N_C,$$

where these variables are the number of neurons in a subset of the model.

N_{in} is the number of neurons in trajectory input ensembles, which is

$$N_{\text{in}} = 48 N_{|\text{in}|}$$

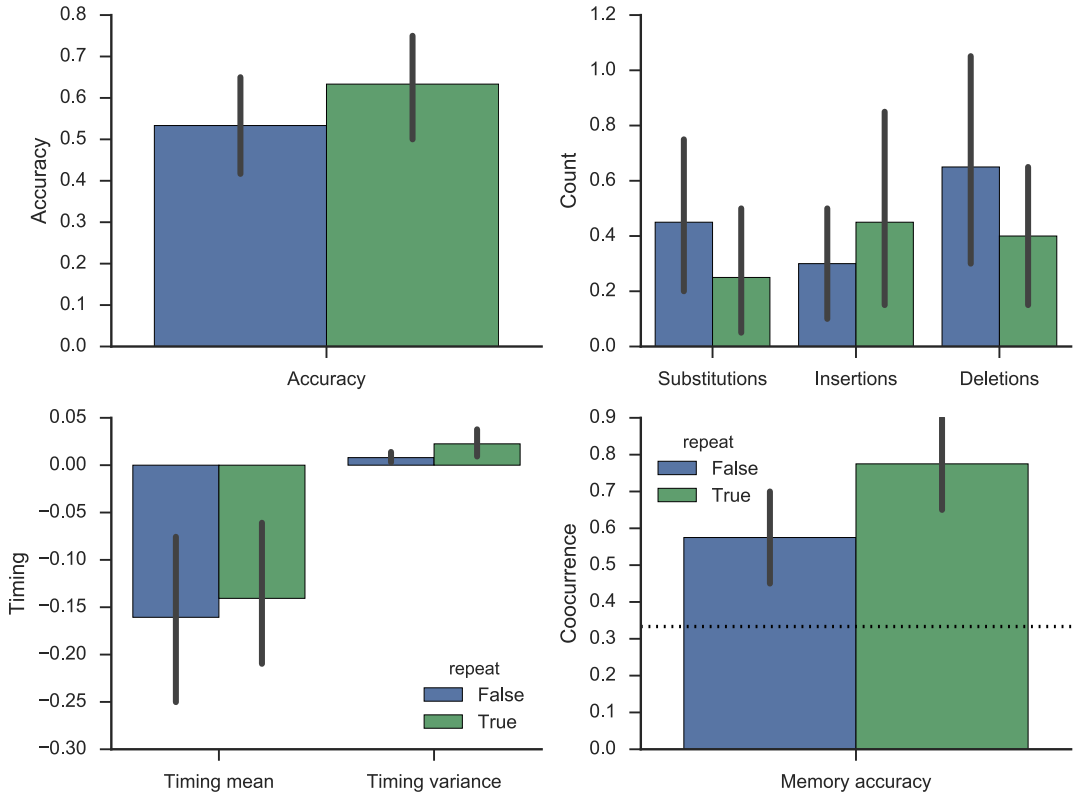


Figure 7.30: Results for the syllable recognition experiment in which we allow syllables to repeat, or constrain the trajectory to contain only unique syllables. Plots show the accuracy, number of substitutions/insertions/deletions, timing differences, and memory accuracy. The dotted line in the memory accuracy plot denotes the chance accuracy value, which depends on the syllabary size. Error bars represent bootstrapped 95% confidence intervals from the 20 trials.

where 48 is the number of gestures, and $N_{|\text{in}|}$ is the number of neurons per input ensemble.

N_{SPA} is the number of neurons for all SPA modules, which is

$$N_{\text{SPA}} = 3|\text{SYLL}|N_{\text{AM}} + D(\text{SYLL})N_{D(\text{SYLL})},$$

where |SYLL| is the number of syllables in the syllabary, N_{AM} is the number of neurons per associative memory ensemble, $D(\text{SYLL})$ is the dimensionality of syllable semantic pointers, and $N_{D(\text{SYLL})}$ is the number of neurons per syllable dimension.

N_{iDMP} is the number of neurons in iDMP networks, which is

$$N_{\text{iDMP}} = |\text{SYLL}| \left(\left(\overline{D}(\text{iDMP}) + 2 \right) N_{D(\text{iDMP})} + 20 \right),$$

where $\overline{D}(\text{iDMP})$ is the average number of gestures in each syllable, $N_{D(\text{iDMP})}$ is the number of neurons per iDMP dimension, and 40 neurons come from ensembles with fixed size.

N_C is the number of neurons in the classifier, which is $N_C = 20$.

With the parameters used in the majority of the experiments above, $N \approx 30000$ neurons, with most of those neurons in the iDMP populations ($N_{\text{iDMP}} \approx 22000$). Note that these values are approximate because the number of gestures per syllable varies depending on the syllable, which is randomly selected.[5] Like the production model, the number of resources grows linearly with the number of syllables. However, it should be noted that the linear growth is scaled by $N_{D(\text{iDMP})}$, the number of neurons per iDMP dimension, which we chose as 1000 through the experiments above. If we assume a conservative syllabary with around 1000 syllables, and increase the dimensionality of each syllable to 256, the model uses $N \approx 7.86 \times 10^6$ neurons, which is more than the generous estimate for the production model. If we assume a more generous estimate with 2000 syllables and 512 dimensions for syllable semantic pointers, then the model uses $N \approx 1.64 \times 10^7$ neurons. The estimate represents approximately 606 mm^3 of cortex (0.606 cm^3), which is still a small amount considering the size of the brain areas involved in sensorimotor integration, but represents poorer scaling than the other two models presented in this thesis.

[5] If we did not sparsify the syllable representation, then we would be using all 48 gesture dimensions. Typically, $\overline{D}(\text{iDMP})$ is around 6–7, which is a significant savings in terms of neural resources used.

Chapter 8

Discussion

8.1 Main results

8.1.1 Neural cepstral coefficients

The most important and somewhat surprising result from the NCC model is that NCCs can yield significantly better training and test correctness rates than MFCCs, which are the most commonly used feature vector in ASR systems. Our goal in these experiments was to show that NCCs are as good as MFCCs, as the primary motivation behind NCCs is to generate a feature vector entirely in spiking neurons, therefore making it suitable for use in Sermo and in neuromorphic systems using silicon cochleas. The fact that they yielded better accuracy rates is promising, though it should be kept in mind that these results are from classification through an ideal statistical linear SVM classifier, rather than through a full continuous ASR system. It remains to be seen whether NCCs will be successful in that more realistic setting.

One advantage of NCCs is that no explicit normalization is required, as shown in Section 7.1.2, where it was determined that NCCs are best used without z-scoring, unlike MFCCs. It is important to note, however, that NCCs do still employ a method of normalization; spiking neurons have a refractory period, meaning that they have a strict maximum firing rate, which puts a limit on the decoded output of an ensemble. Unlike MFCCs, however, this normalization is applied for every step in the pipeline, rather than just once at the end. It is possible that the normalization inherent in spiking neural networks is responsible for the improved performance compared to MFCCs.

While a similar argument could be made for why adding derivatives had a deleterious effect on NCC test accuracy, it is also possible that using offline SVM classification is responsible

for decreased performance. Having many more features may make the supervised learning procedure more difficult, or perhaps since we have the full history, the derivative is redundant and makes it more difficult to linearly separate data, or makes overfitting more likely.

We view the final NCC experiment, in which the auditory periphery model was varied, as being the main contribution of this model. Because the model bridges the gap between detailed auditory periphery models and automatic speech recognition systems, we are able to do apples-to-apples comparisons of how well the periphery models process realistic speech samples. In general, the more realistic the model, the better it is for speech, with the notable exception of the Gammatone filter, which is the least realistic, cheapest to compute, yet as accurate as the most complicated Tan Carney model.

There are two reasons why we might be skeptical of the impressive Gammatone results. For one, the determination of good parameter sets in the other experiments in Section 7.1.2 all used the Gammatone filter. It is possible that performing the same experiments with other filters would find parameters sets that allow those auditory periphery models to perform as well or better than the Gammatone filter. Additionally, arguably the most difficult part of continuous ASR systems is determining where phone boundaries lie, rather than classifying pre-segmented phones. While we believe that the experiments performed in Section 7.1.2 are a useful comparison between MFCCs and NCCs, some of the benefits of more sophisticated models like the Tan Carney model may lie in their ability to segment speech. The same argument holds for adaptive neuron types; while they may not be essential for differentiating between pre-segmented speech samples, they may be essential for segmenting continuous speech. Looking at the impact of these NCC representation choices in a continuous ASR system is therefore an important next step.

8.1.2 Syllable sequencing and production

The syllable sequencing and production model can generate production information trajectories of several seconds in length from a mental syllabary of size four, and those trajectories can result in intelligible speech when synthesized with the VocalTractLab articulatory synthesizer.

It is surprising that the model is able to perform at relatively high speeds (2.5–6.5 Hz), but has trouble with lower speeds. We believe the source of this difficulty is in the sequencer network. When a syllable is about to end, the sequencer network switches the gating signal and starts the reset signal, effectively starting the DMP network associated with the next syllable, while also pushing the current syllable's DMP network to the starting position. When frequencies are high, the drive to the starting position is approximately the same as the intrinsic

dynamics, so the system moves as expected. However, for low frequencies, the additional drive in the current syllable forces it to finish the current syllable too quickly.

The results in Figure 7.19 are surprising because syllables not taking part in a sequence should be completely inhibited, and therefore should not affect the operation of other syllable DMPs. Yet, a large number of deletions occur when the syllabary has four or more syllables. While we believe that the SPA modules are the source of these insertions (due to the relatively low dimensionality of syllable representations used in the experiment), further inspection of these models is crucial to scaling it to large vocabularies.

While the production network is able to produce discrete vocal tract gesture trajectories successfully, DMPs were created for continuous control problems, and deal with continuous trajectories more naturally. In Chapter 3, we noted that the choice between discrete and continuous production information is an empirical one. While we have not ruled out discrete vocal tract gestures as a production information representation, we should still investigate whether continuous trajectories are more natural for the methods employed in the syllable sequencing and production model.

The final neural model employs neural inhibition in several key areas (see Figure 6.4). While employing neural inhibition was not an explicit goal in the design process, inhibition provides a method for making qualitative behavioral changes rapidly, and therefore was used when such changes were necessary. Having neural inhibition as one of the tools at a modeler's disposal provides another concrete benefit of using spiking neural models over other modeling techniques.

One process in the model that required significant effort was the translation from a recorded production information trajectory to a gesture score. The approach using temporal derivatives works well, but is parameterized with a few parameters which were chosen through a manual process rather than through an automatic optimization process. It would be worth looking at the gesture score reconstruction algorithm in more detail to examine if an improved algorithm changes the accuracy results, or the quality of synthesized speech.

Finally, it is important to emphasize that this model captures the end result of some learning process, which continues to operate even in adulthood. One possible way forward for this model is to begin with its current state, and use NEF-based learning techniques (e.g., MacNeil and Eliasmith 2011; Bekolay et al. 2013b) to fine-tune its performance based on sensory feedback.

8.1.3 Syllable recognition

The syllable recognition model successfully addresses the main criteria influencing its design. The model is able to classify trajectories with relatively high degrees of freedom (at least seven) that advance non-uniformly through time, and that occur in a continuous stream with no segmentation signals.

One of the biggest limitations of the syllable recognition model in its current state is its sensitivity to the frequency of the syllable (see Figure 7.27). Like the production model, it performs poorly for slow syllables, yielding a large number of insertions (although it could be argued that repeated identification of the correct syllable is acceptable). Unlike the production model, performance is best at around 1.8 Hz, then monotonically worsens for higher frequencies. As mentioned previously, we believe that the decreased accuracy is due to a coupling between the speed of the syllable and other parameters in the model (e.g., the scale). In other words, the iDMPs must progress their state forward at a higher rate when the syllable is voiced at a higher rate. Methods for estimating speech rate should therefore be investigated and used to dynamically modify the effective scale depending on the overall speech rate. Interestingly, Pasley et al. (2012) also suggested that the human auditory system contains nonlinearities that depend on speech rates (see Section 2.1.4).

The poor results of the model with increased numbers of syllables in the syllabary indicate that the competition introduced to isolate a relatively small number of iDMPs is insufficient to allow only one syllable iDMP to be active. However, the model does operate well with few syllables, and many syllables are similar and would often be confused by humans if they did not use linguistic context. Therefore, we conclude that the iDMP networks are still suitable for trajectory classification, but may not necessarily be used in a drift-diffusion style classification system in which the first iDMP to reach a threshold value makes a classification. Instead, we aim to explore a model in which the sensorimotor integration system would allow the iDMPs to continuously operate, and integrate the information from those iDMPs as well as information from other sources (including top-down linguistic influences) to make a final determination of the uttered syllable (when necessary). Aside from sensorimotor learning, it is possible that this type of syllable classifier may only be used in cases where the linguistic systems disambiguate between two similar choices.

8.2 Comparison to existing models

Currently, it is difficult to make apples-to-apples comparisons between the models presented in this thesis and other models, especially those in the speech literature. The models that are

well known and discussed in speech literature are traditional connectionist models, which provide interesting theoretical predictions, but are not functional models that can replicate speech function. On the other hand, artificial intelligence has adopted statistical methods for speech recognition and synthesis, which work well in practice, but contribute little to speech theory. We hope that the neural models presented in this thesis bridge the gap between productive and theoretically interesting models by solving recognition and synthesis problems with methods that can be directly related to linguistic and neuroscientific theories.

Putting aside the general argument that all of the models in this thesis are implemented with spiking neurons, there are still useful comparisons to make with the previous work surveyed in Chapter 4.

For the NCC model, explicit comparisons were made with the commonly used MFCC feature vector as part of its evaluation. Surprisingly, NCCs are better suited than MFCCs for classifying pre-segmented phones. While there are many other variants of MFCCs using Gammatone and other auditory filters, we could not find any MFCC variants that gave state-of-the-art results on a large speech corpus like TIMIT. Instead, the motivation for using more biologically realistic auditory filters is that they are more noise robust, and so the studies using these filters use speech data sets that are either naturally noisy, or have noise added to it. As the NCC model is the first that I am aware of to show an improvement with a subset of pre-segmented TIMIT, it will be revealing to use NCCs in a continuous online ASR system.

For the syllable sequencing model, the syllable sequences are far more temporally flexible and large scale than the sequencing methods reviewed in Section 4.2, except for the OSE model, which we adapt for our sequencing model. Our additions to the sequencing model do not add anything useful for solving the general problem of serial list memory, but are useful in other situations that require rapid iteration through a list of symbols, which may be common in highly learned actions driven by cortical networks rather than the cortex-basal ganglia-thalamus-cortex loop. As such, we will extract this aspect of the model out in order to make it more approachable for other researchers using the SPA to generate symbol sequences.

For trajectory generation, it is not yet clear the exact comparison between DMPs and Task Dynamics. At their core, the two techniques are remarkably similar. In the trajectory generation model, we only use the canonical system and forcing function of the DMP; however, a more robust mapping from the production information trajectory to articulator positions in the VocalTractLab synthesizer would also take into account the DMP's point attractors.[1] Making a more robust motor expansion system and providing articulator positions to VocalTractLab, rather than a gesture score, could improve synthesized speech quality. Similarly, it would be

[1] In general, a tighter integration with VocalTractLab would improve aspects of the model significantly; unfortunately, VocalTractLab is currently not an open source program, making integration more difficult.

worth attempting a neural implementation of Task Dynamics in order to compare the two approaches using the same methods.

For trajectory classification, the iDMP is a novel trajectory classification technique which warrants further investigation outside of the context of a neural model. Some early experiments using a non-neural version of the model were successful in classifying toy problems; however, it remains to be seen whether it could be used in gesture recognition systems with high accuracy, like HMM-based methods and the GVF algorithm.

8.3 Contributions

As this thesis draws on theories and techniques from a myriad of fields, it also attempts to make contributions to many of those fields. While a general contribution to each field is relating the concepts in that field to concepts in other fields, we also note specific contributions for several fields.

8.3.1 Contributions to computer science

Progress in machine learning has been accelerated significantly through the existence of benchmark data sets with easily reproducible state-of-the-art results (e.g., the MNIST data set, which is now solved better by machine vision algorithms than humans). However, one issue with benchmarks is that they can lead to myopic advances which eke out one or two more accuracy points with little context of how the improvement might generalize to other problems, or how it could play a role in a larger integrated system.

I believe that we make two contributions to machine learning through providing a novel benchmark data set, and an integrated system that can be used to benchmark how algorithmic choices affect other systems. First, we intend to release a data set consisting of time-aligned vocal tract gesture trajectories and speech samples synthesized by the VocalTractLab synthesizer. As previously discussed, data sets like the X-ray microbeam database (Westbury et al., 1990) and MNGU0 (Steiner et al., 2012) provide data sets consisting of time-aligned vocal tract articulator positions and speech samples. However, I am aware of no available data set of time-aligned vocal tract gestures and speech samples. We theorize in Sermo that production information is decoded from auditory features. In order to determine whether that production information could take the form of vocal tract gestures, we will release the gesture trajectory data set so that machine learning researchers can apply statistical methods to determine if the acoustic signal has enough information to decode vocal tract gestures.

Additionally, we theorize in Sermo that the lexical decoding process often used in ASR systems is a necessary component of the speech system. By providing a conceptual framework through which to contextualize the lexical decoding process, machine learning researchers can see how an ASR system might fit into a biologically plausible speech system. As was shown in the comparison between auditory periphery models, integrated systems like Sermo enable comparisons between related techniques on the basis of its role in a larger system, rather than its ability to match an accuracy rating on a benchmark problem.

Outside of benchmarking, the NCC feature vector representation may be useful for general automatic speech recognition, a subfield of artificial intelligence. The iDMP algorithm introduced in this thesis is likely to be applicable to gesture recognition, a field of interest to human-computer interaction and machine learning researchers.

8.3.2 Contributions to linguistics

While Sermo summarizes a large body of linguistics research with terminology that should be understandable for computer scientists, the theories embodied in Sermo are not novel in the context of linguistics. However, it may be instructive to see linguistic theories discussed with computational and neurobiological terms.

Our primary contribution to linguistics is a method for connecting speech to higher-level linguistic concepts. Previously, Blouw and Eliasmith (2013); Blouw et al. (2015); Stewart et al. (2015) have explored solutions to syntactic and semantic problems using the Semantic Pointer Architecture (SPA). We propose that audio signals can be mapped to semantic pointers through an auditory feature extraction and linguistic decoding step. The linguistic decoding may also be informed by production information and syllable classifications, which are also represented with semantic pointers.

8.3.3 Contributions to neural modeling

We make two main contributions to large-scale neural modeling using the NEF and SPA: connecting spiking neural networks to the domain of speech, and associative memories using temporal inputs and outputs.

The models presented in this thesis are the first instances (to my knowledge) of incorporating the sensory and motor aspects of speech into the NEF through Nengo. The NCC model processes speech online using auditory periphery models constructed with Brian Hears. The

speech sequencing and production model produces audio speech samples using the VocalTractLab articulatory synthesizer. The addition of these sensors and actuators enables new avenues for large-scale neural modeling research. For example, one could envision a version of Spaun that uses audio inputs and outputs rather than vision and handwritten digits.

The SPA deals with symbol-like representations using high-dimensional vector spaces. Currently, the semantic pointers associated with concepts are static vectors that can be manipulated through the merging and binding operators, among other operations. However, the concepts represented in speech are not static. In order to deal with the temporal nature of speech, we presented temporal output associative memories and temporal input associative memories. Temporal output associative memories generate continuously varying output signals from semantic pointer representations using DMPs. They provide a robust link between the discrete, symbolic world of high-level representations that can be flexibly combined and the continuous, subsymbolic world of low-level representations that must be precisely timed. Temporal input associative memories map a continuously varying input signal to semantic pointer classifications using iDMPs. They provide an analogous link between continuously varying low-level sensory inputs and high-level symbolic representations. While we have introduced these networks to deal with speech, they can also be used for other situations where temporal information is important (e.g., video analysis, general motor control).

8.4 Predictions

Sermo and the systems of Sermo modeled in this thesis are in early stages, primarily informed by existing theories and models. As such, we embody many of the predictions made by these theories and models. While the predictive power of Sermo will develop as the models develop, we can nevertheless make some predictions based on the models presented in this thesis as they currently exist.

8.4.1 Role of the auditory periphery in speech perception

In Section 7.1.2 we showed that more complicated auditory periphery models achieved better phone correctness rates than simpler models (with the exception of the Gammatone filter). We therefore predict that the nonlinear effects modeled in the dynamic compressive Gammachirp and Tan Carney auditory filter models are important for perceiving speech.

Similarly, we showed that using adaptive LIF neurons did not affect the phone correctness rate. However, it is usually the case that adaptive neuron models fire fewer action potentials

than do non-adaptive models like the LIF neuron. The adaptive LIF fires more action potentials during the onset of a signal, but then fires fewer action potentials as it adapts to that signal. We therefore predict that the spiral ganglion cells in the inner ear are adaptive because it is more energy efficient, rather than because the responses of adaptive neurons are more effective for perceiving speech.

8.4.2 Syllabification limits

In Section 7.2.2, we showed that producing sequences of syllables tends to break down when the sequence length is four or above. While it is possible to increase the dimensionality of the semantic pointer representations to improve the accuracy of longer syllable sequences, we nevertheless predict that there is a limit to the speed of the linguistic system's prosodification process. Recall that the prosodification process organizes strings of phonemes into discrete syllables, which become syllable targets for the sequencing system. If speech production is suppressed, the prosodification process could plan the syllabification of an entire utterance. Our model predicts that the linguistic system cannot plan that far ahead; it can construct a sequence of four (or possibly up to five or six) syllable targets ahead of time, but in general, the syllabification of an utterance must be generated on the fly.

8.4.3 Mapping to brain areas

In Chapter 2 we reviewed the brain areas involved in speech recognition and production, in part to place connectivity constraints on the model as a whole. However, in order to generate testable predictions, we can also impose a mapping from the neural structures in the model to speech-related brain regions.

The NCC model is primarily a model of the auditory periphery, which is cochleotopically organized. The ensembles representing cepstral coefficients and derivatives aggregate information across all frequencies, so we predict that cepstral coefficients (or, in general, decorrelated acoustic information) would be represented in higher auditory areas in the superior and middle temporal gyri. More specifically, we predict that activity in the first layer of non-tonotopically organized auditory cortex would closely resemble the activity of the neurons in the cepstral coefficient ensembles in the NCC model. The fact that the human auditory system integrates information from many more frequencies than are integrated in the model indicates that additional hierarchical processing layers may be beneficial before combining all this information together, as is done in human auditory cortex. The NCC model should also include these hierarchical process layers when scaling up.

The sequencing and production model can be related to ventral sensorimotor cortex (vSMC) literature, as reviewed in Section 2.2.3. Specifically, we predict that the vSMC can be mapped to the production information network in our model. As such, vSMC activity should be very similar to the activity of production information network ensembles in the production model. Briefly, because the production information network assigns a separate ensemble to each vocal tract gesture, we predict that vocal tract gestures could be decoded from recordings of vSMC activity. If this turns out to not be case, then it would either indicate that the sequencing and production model should produce vocal tract articulator position targets rather than vocal tract gestures, or it may indicate that the representation of vocal tract gestures in the production information network is more distributed than is currently in the model. The model can be modified to use fewer multidimensional ensembles in the production information network to test these more specific predictions.

Finally, we believe that the syllable recognition model can be mapped to area Spt, which we reviewed in Section 2.3.2. We therefore predict that activity in Spt would be closely related to the activity in the syllable recognition model. Since the syllable recognition model is likely to be used in situations where a lexical decoding must be disambiguated, we propose an experiment to test the prediction that area Spt performs syllable recognition. In the experiment, subjects would watch videos of people voicing sentences with words that can be manipulated with the McGurk effect; i.e., one word would be perceived differently depending on the video clip associated with the audio. We predict that transient deactivation of area Spt would abolish the McGurk effect, making the sentences sound the same regardless of the video clip viewed. Unfortunately, this kind of manipulation is difficult to perform in humans; transcranial magnetic stimulation (TMS) may be an option if a subject's area Spt is not too deep in the Sylvian fissure.

Many more mappings can be made in future iterations of these models, and with a more thorough investigation of neurobiological literature.

8.5 Limitations

The primary limitation of these models is that they are still in early stages, and do not scale to the challenges of natural speech, either in terms of the speed of speech (as in the recognition model) or in the number of syllables in the mental syllabary (as in the production and recognition models). However, as we showed in Sections 7.1.3, 7.2.3, and 7.3.3, all of the models presented in this thesis scale well in terms of neural resources, meaning that adding additional neural structures to increase scaling performance will not result in biologically implausible neural models.

In terms of the algorithms used by these models, our treatment of DMPs is currently incomplete, as we do not consider the DMP point attractor. As mentioned previously, incorporating the point attractor in the production model should be possible with tighter integration between the model and the articulatory synthesizer. However, in the recognition model, the novel iDMP algorithm requires further development to estimate both the state of the canonical system, and the state of the system with respect to the point attractor.

Finally, the conceptual Sermo model only describes the computations necessary for recognizing and synthesizing syllables. However, speech involves much more than perceiving and producing syllables; even the perception and production of syllables is affected by prosody, which is not currently reflected in Sermo. Sermo is only a first step toward an integrated closed-loop model of speech.

8.6 Future work

While we have already mentioned many aspects of Sermo and the three models implemented in this thesis that can be improved in future work, there are nevertheless many more avenues of interesting future research that I wish to highlight.

8.6.1 Model extensions

Part of the motivation for the NCC model is that there are existing hardware devices called silicon cochleas which aim to replicate the function of the biological cochlea such that they can be implanted in the brain to partially recover the sense of hearing. As we have shown that the NCC model can be used to classify phones in pre-segmented speech, it would be useful to implement the NCC model using a silicon cochlea as the auditory periphery model and run the same experiment to determine if phones can be classified in pre-segmented speech. If so, then the NCC model could serve as a way of comparing different silicon cochlea implementations without having to implant them in human subjects. Additionally, the NCC model would be a useful tool in silicon cochlea design, as long as there is agreement between the results of the NCC model and the perceptual evaluations of human subjects with cochlear implants.

Currently, the syllable sequencing and production model can instantiate syllable DMPs with different intrinsic frequencies. However, syllable speed can also vary within syllables. The DMP framework makes this possible through dissociating the canonical system state from the forcing function itself. In order for the production model to take advantage of this dissociation, we would need to make the DMP system state oscillator a controlled oscillator. Fortunately,

controlled oscillators have already been implemented successfully with the NEF and Nengo (Bekolay et al., 2013a), so this modification should be straightforward.

In the syllable recognition model, the classifier ensemble uses adaptive LIF neurons in order to provide a slightly longer burst of activity when a syllable is classified, ensuring that the states of the iDMPs are cleared when a classification occurs. However, as can be seen in Figure 7.23, states are not always reset completely, resulting in erroneous syllable classifications. One way to mitigate this issue would be to use a neuron model that intrinsically bursts, rather than the adaptive LIF neuron which fires only slightly more action potentials than normal. Incorporating a bursting model into Nengo would be useful for other models as well; I anticipate that precisely timed models, including the production model, could benefit from bursting neurons in situations where neural inhibition plays an important role.

Finally, we currently use production information trajectories computed offline as the input to the syllable recognition model. Using actual trajectories, either from databases like MNGU0 (Steiner et al., 2012), or from a model that decodes production information from auditory features, would be a better test of the syllable recognition model.

8.6.2 Syllable consolidation

Currently, we have implemented models of sensorimotor integration, including a mental syllabary, as the endpoint of a learning process. However, with the recent development of synaptic plasticity rules for the NEF (MacNeil and Eliasmith, 2011; Bekolay et al., 2013b), it is possible to also model sensorimotor development, which is important because the role of sensorimotor integration is most prominent during speech development, not in adult speech.

A full picture of speech development involves learning vocal tract gestures through reinforced motor babbling, learning basic syllables through mimicry, and building a repository of learned syllable representations (i.e., a mental syllabary). While I believe that Sermo can be a useful starting point for a model of all developmental stages, adding elements to the mental syllabary would be possible with few modifications. We call this learning process "syllable consolidation," and provide a sketch for designing such a model in the remainder of this section.

Syllable consolidation involves learning a novel vocal tract gesture trajectory to voice a syllable that is initially infrequently encountered, but through repeated presentations, becomes frequent (i.e., is consolidated in the mental syllabary). We propose that it would be learned in three steps.

1. Initialize a new syllable from the most similar existing syllable. For example, when learning to voice the syllable [bɑ], the system should start from the syllable [gɑ] if it is known.

2. Swap compatible speech gestures. For example, vowel producing gestures would be compatible, allowing for modifying a [bɑ] to a [bu], and so on. The choice of which gesture to swap and how to swap it will be informed by the syllable recognition system.

3. Fine-tune the voiced syllable until it can be recognized as the syllable to be learned.

This type of learning system could be called "bootstrapped" because it assumes that a system with an existing repertoire of syllables already exists. These existing syllables would be used in learning the new syllable. Bootstrapped syllable learning contrasts with the type of syllable learning done as an infant and toddler, which uses reinforced speech babbling to learn novel syllables.

Bootstrapped learning would be accomplished with the NEF and Nengo by setting up a new DMP with a system state oscillator. Initially, the forcing function decoded from the state would always return zero. The prescribed error sensitivity learning rule (MacNeil and Eliasmith, 2011; Bekolay, 2011) would be applied to the connection decoding the forcing function. Initially, the error signal would be provided by the syllable DMP that is most similar to the syllable to be learned. Once that transfer learning has finished, the error signal would be provided by the production information decoding, which would swap the compatible gestures, and fine-tune the newly consolidated syllable.

These steps require several systems that have already been implemented in the recognition and synthesis systems separately; for example, the ability to compare voiced syllables to those already known is one of the primary goals of the recognition system itself, so it can be leveraged when trying to learn new syllables. However, these steps also point to new systems that must be implemented. First, the system requires a method to transfer a forcing function from one DMP to another. Second, the production information decoding process that we have hypothesized is done in the sensorimotor integration system would be required for gesture swapping and fine-tuning.

8.6.3 Prosody

A final area for fruitful future work is the incorporation of prosodic effects in Sermo and models of Sermo parts. Prosody is an important part of speech, and is notably missing from current state-of-the-art approaches to speech recognition and synthesis. Adding prosodic effects

to approaches to computational speech would go a long way toward making recognition and synthesis more natural. Unfortunately, most existing computational approaches make prosody difficult to add in *post hoc*. However, I believe that because Sermo is designed with highly parallel parts (i.e., spiking neurons), and uses biologically inspired components (e.g., an articulatory synthesizer), prosody will be natural to include in future work.

Prosody will involve additional theoretical concepts and Sermo subsystems. Theoretically, I believe that the linguistic concept of tone units will be central to incorporating prosody in Sermo. Briefly, tone units represent an additional level in the speech production hierarchy (see Figure 2.11) that is higher than syllables. A tone unit is made up of a serially ordered sequence of syllables.

The structure of a tone unit is similar to a syllable, except its component parts are syllables instead of phonemes, and its components are serially ordered. A tone unit must contain a tonic syllable (sometimes also called the nucleus), and can optionally contain one or more syllables in a pre-head, head, or tail section. The pre-head consists of all syllables before the first stressed syllable in a tone unit. The head consists of all syllables from the first stressed syllable to the tonic syllable. The tonic syllable is the most significant syllable in the tone unit because pitch changes in the tonic syllable will occur relative to the tonic syllable; the tonic syllable is not necessarily the loudest or most prominently stressed syllable in the tone unit, though it does always contain a stressed (and therefore heavy) syllable. The tail consists of all syllables following the tonic syllable.

Tone units are defined by a pitch trajectory that aligns with the component parts of the tone unit. There are a limited set of possible pitch trajectories in English, including falling, rising, fall-rise, and rise-fall trajectories. The same sequence of syllables can change its meaning dramatically by using a different pitch trajectory, or by changing the position of the tonic syllable within that pitch trajectory. The meaning of each pitch trajectory changes depending on the utterance in question.

Adding prosody to Sermo would require the perception and production of pitch trajectories, which would be aligned to syllables in the roles described above. Perceiving pitch trajectories would also require a system for determining the baseline pitch of a particular speaker. I believe that all of these systems could be designed and implemented with the NEF.

Chapter 9

Conclusion

In this thesis, I proposed a conceptual model called Sermo which provides a biologically constrained framework for a computational closed-loop speech system. I implemented three models with biologically plausible spiking neurons that span aspects of speech relating to perception, production, and the interface of the two, sensorimotor integration.

The first neural model extracts acoustic features from speech signals using auditory periphery models and connections between neural ensembles. The feature vector, called neural cepstral coefficients (NCCs), performs significantly better than a feature vector commonly used in automatic speech recognition on a phone classification task using pre-segmented speech. Additionally, the model is able to do apples-to-apples comparisons between five auditory periphery models in terms of how well they process pre-segmented speech samples for classification. We found that the most complicated (Tan Carney) model and the least complicated (Gammatone) model performed the best. The general trend is that more complicated models perform better (with the exception of the Gammatone model).

The second neural model sequences a list of syllables and generates production information trajectories from those syllables, which can be synthesized by an articulatory synthesizer. The trajectories generated were significantly closer to the target trajectories than random syllables, and synthesized speech was intelligible in 35% of cases. Scaling the model to the level of an adult vocabulary would take up only 0.1637 cm^3 of cortex, though in its current state, the model does not perform well when scaled up to the size of an adult syllabary.

The third neural model recognizes syllables from a continuous trajectory of production information. Classification accuracy reached over 80% in some experiments, despite using no acoustic information. The model also produced semantic pointers for the recognized syllables; the accuracy of the memory representation reached 90% in some experiments,

though chance values were 33% due to a small set of possible syllables. The model is able to operate continuously without needing to be reset, but scales poorly to rapid utterances and large repositories of syllables.

The conceptual and neural models presented here represent the first steps in applying large-scale neural modeling techniques to speech. As such, we are optimistic that future models can overcome the scaling issues that these models currently face, and can result in a biologically constrained closed-loop model of speech, enabling computers to have natural interactions with humans.

References

Ackermann, H. and A. Riecker
2004. The contribution of the insula to motor aspects of speech production: a review and a hypothesis. *Brain and Language*, 89(2):320–328.

Aertsen, A. and P. Johannesma
1981. The spectro-temporal receptive field. *Biological Cybernetics*, 42(2):133–143.

Alam, M. J., P. Kenny, and D. O'Shaughnessy
2013. Low-variance multitaper Mel-frequency cepstral coefficient features for speech and speaker recognition systems. *Cognitive Computation*, 5(4):533–544.

Andersen, R. A.
1997. Multimodal integration for the representation of space in the posterior parietal cortex. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 352(1360):1421–1428.

Badin, P., G. Bailly, L. Reveret, M. Baciu, C. Segebarth, and C. Savariaux
2002. Three-dimensional linear articulatory modeling of tongue, lips and face, based on MRI and video images. *Journal of Phonetics*, 30(3):533–553.

Bahl, L. R., F. Jelinek, and R. L. Mercer
1983. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2:179–190.

Ballachanda, B. B.
1997. Theoretical and applied external ear acoustics. *The Journal of the American Academy of Audiology*, 8:411–420.

Bekolay, T.
2011. Learning in large-scale spiking neural networks. Master's thesis, University of Waterloo.

Bekolay, T., J. Bergstra, E. Hunsberger, T. DeWolf, T. C. Stewart, D. Rasmussen, X. Choo, A. R. Voelker, and C. Eliasmith
2013a. Nengo: a Python tool for building large-scale functional brain models. *Frontiers in Neuroinformatics*, 7.

Bekolay, T., C. Kolbeck, and C. Eliasmith
2013b. Simultaneous unsupervised and supervised learning of cognitive functions in biologically plausible spiking neural networks. In *Conference of the Cognitive Science Society*, Pp. 169–174.

Benjamin, B. V., P. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J.-M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. Merolla, and K. Boahen
2014. Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE*, 102(5):699–716.

Bevilacqua, F., N. Schnell, N. Rasamimanana, B. Zamborlin, and F. Guédy
2011. Online gesture analysis and control of audio processing. In *Musical Robots and Interactive Multimodal Systems*, Pp. 127–142. Springer.

Bevilacqua, F., B. Zamborlin, A. Sypniewski, N. Schnell, F. Guédy, and N. Rasamimanana
2010. Continuous realtime gesture following and recognition. In *Gesture in Embodied Communication and Human–Computer Interaction*, Pp. 73–84. Springer.

Birkholz, P.
2013. *VocalTractLab 2.1 User Manual*. Technische Universität Dresden.

Birkholz, P., D. Jackèl, and B. J. Kröger
2006. Construction and control of a three-dimensional vocal tract model. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 1, Pp. 873–876. IEEE.

Birkholz, P., B. J. Kröger, and C. Neuschaefer-Rube
2011. Synthesis of breathy, normal, and pressed phonation using a two-mass model with a triangular glottis. In *Interspeech*, Pp. 2681–2684.

Blouw, P. and C. Eliasmith
2013. A neurally plausible encoding of word order information into a semantic vector space. In *Conference of the Cognitive Science Society*, Pp. 1905–1910.

Blouw, P., E. Solodkin, P. Thagard, and C. Eliasmith
2015. Concepts as semantic pointers: a framework and computational model. *Cognitive Science*, Pp. 1–35.

Bogacz, R., E. Brown, J. Moehlis, P. Holmes, and J. D. Cohen
2006. The physics of optimal decision making: a formal analysis of models of performance in two-alternative forced-choice tasks. *Psychological Review*, 113(4):700.

Bolhuis, J. J., K. Okanoya, and C. Scharff
2010. Twitter evolution: converging mechanisms in birdsong and human speech. *Nature Reviews Neuroscience*, 11(11):747–759.

Bouchard, K. E., N. Mesgarani, K. Johnson, and E. F. Chang
2013. Functional organization of human sensorimotor cortex for speech articulation. *Nature*, 495(7441):327–332.

Breshears, J. D., A. M. Molinaro, and E. F. Chang
2015. A probabilistic map of the human ventral sensorimotor cortex using electrical stimulation. *Journal of Neurosurgery*, Pp. 1–10.

Broca, P.
1861. Remarks on the seat of the faculty of articulated language, following an observation of aphemia (loss of speech). *Bulletin de la Société Anatomique*, 6:330–57.

Browman, C. P. and L. Goldstein
1989. Articulatory gestures as phonological units. *Phonology*, 6(02):201–251.

Brown, S., A. R. Laird, P. Q. Pfordresher, S. M. Thelen, P. Turkeltaub, and M. Liotti
2009. The somatotopy of speech: phonation and articulation in the human motor cortex. *Brain and Cognition*, 70(1):31–41.

Buesing, L., J. Bill, B. Nessler, and W. Maass
2011. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLOS Computational Biology*, 7(11):e1002211.

Cao, M., A. Li, Q. Fang, E. Kaufmann, and B. J. Kröger
2014. Interconnected growing self-organizing maps for auditory and semantic acquisition modeling. *Frontiers in Psychology*, 5.

Caramiaux, B., N. Montecchio, A. Tanaka, and F. Bevilacqua
2014. Adaptive gesture recognition with variation estimation for interactive systems. *ACM Transactions on Interactive Intelligent Systems*, 4(4):18.

Carney, L. H.
1993. A model for the responses of low-frequency auditory-nerve fibers in cat. *The Journal of the Acoustical Society of America*, 93(1):401–417.

Chan, V., S.-C. Liu, and A. Van Schaik
2007. AER EAR: A matched silicon cochlea pair with address event representation interface. *IEEE Transactions on Circuits and Systems*, 54(1):48–59.

Chittka, L. and A. Brockmann
2005. Perception space–the final frontier. *PLOS Biology*, 3(4):e137.

Choe, Y. and R. Miikkulainen
1998. Self-organization and segmentation in a laterally connected orientation map of spiking neurons. *Neurocomputing*, 21(1):139–158.

Cholin, J., N. O. Schiller, and W. J. Levelt
2004. The preparation of syllables in speech production. *Journal of Memory and Language*, 50(1):47–61.

Choo, F.-X.
2010. The ordinal serial encoding model: serial memory in spiking neurons. Master's thesis, University of Waterloo.

Cogan, G. B., T. Thesen, C. Carlson, W. Doyle, O. Devinsky, and B. Pesaran
2014. Sensory-motor transformations for speech occur bilaterally. *Nature*, 507(7490):94–98.

Crawford, E.
2014. Biologically plausible, human-scale knowledge representation. Master's thesis, University of Waterloo.

Dallos, P.
1992. The active cochlea. *The Journal of Neuroscience*, 12(12):4575–4585.

Dang, J. and K. Honda
2004. Construction and control of a physiological articulatory model. *The Journal of the Acoustical Society of America*, 115(2):853–870.

Davis, K., R. Biddulph, and S. Balashek
1952. Automatic recognition of spoken digits. *The Journal of the Acoustical Society of America*, 24(6):637–642.

De Boer, E.
1975. Synthetic whole-nerve action potentials for the cat. *The Journal of the Acoustical Society of America*, 58(5):1030–1045.

Dell, G. S.
1988. The retrieval of phonological forms in production: Tests of predictions from a connectionist model. *Journal of Memory and Language*, 27(2):124–142.

DeWolf, T.
2010. NOCH: a framework for biologically plausible models of neural motor control. Master's thesis, University of Waterloo.

DeWolf, T.
2015. *A neural model of the motor control system.* PhD thesis, University of Waterloo.

Dimitriadis, D., P. Maragos, and A. Potamianos
2005. Auditory Teager energy cepstrum coefficients for robust speech recognition. In *Interspeech*, Pp. 3013–3016.

Doyon, J., V. Penhune, and L. G. Ungerleider
2003. Distinct contribution of the cortico-striatal and cortico-cerebellar systems to motor skill learning. *Neuropsychologia*, 41(3):252–262.

Drew, P. J. and L. Abbott
2003. Model of song selectivity and sequence generation in area HVc of the songbird. *Journal of Neurophysiology*, 89(5):2697–2706.

Dunn, N. A., S. R. Lockery, J. T. Pierce-Shimomura, and J. S. Conery
2004. A neural network model of chemotaxis predicts functions of synaptic connections in the nematode caenorhabditis elegans. *Journal of Computational Neuroscience*, 17(2):137–147.

Eide, E.
2001. Distinctive features for use in an automatic speech recognition system. In *Interspeech*, Pp. 1613–1616.

El-Masri, S., X. Pelorson, P. Saguet, and P. Badin
1996. Vocal tract acoustics using the transmission line matrix (TLM) method. In *International Conference on Spoken Language*, volume 2, Pp. 953–956. IEEE.

Eliasmith, C.
2011. Notes from SYDE750.

Eliasmith, C.
2013. *How to build a brain: A neural architecture for biological cognition*. Oxford University Press.

Eliasmith, C. and C. H. Anderson
2004. *Neural Engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press.

Eliasmith, C., T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen
2012. A large-scale model of the functioning brain. *Science*, 338(6111):1202–1205.

Ellis, A. W., D. Miller, and G. Sin
1983. Wernicke's aphasia and normal language processing: A case study in cognitive neuropsychology. *Cognition*, 15(1):111–144.

Escabı, M. A. and C. E. Schreiner
2002. Nonlinear spectrotemporal sound analysis by neurons in the auditory midbrain. *The Journal of Neuroscience*, 22(10):4114–4131.

Everest, F. A., K. C. Pohlmann, and T. Books
2001. *The Master Handbook of Acoustics*, volume 4. McGraw-Hill New York.

Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin
2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874.

Fee, M. S., A. A. Kozhevnikov, and R. H. Hahnloser
2004. Neural mechanisms of vocal sequence generation in the songbird. *Annals of the New York Academy of Sciences*, 1016(1):153–170.

Fee, M. S. and C. Scharff
2010. The songbird as a model for the generation and learning of complex sequential behaviors. *Institute for Laboratory Animal Research Journal*, 51(4):362–377.

Fernández, S., A. Graves, and J. Schmidhuber
2008. Phoneme recognition in TIMIT with BLSTM-CTC. *arXiv preprint arXiv:0804.3269*.

Fletcher, H. and W. A. Munson
1933. Loudness, its definition, measurement and calculation. *Bell System Technical Journal*, 12(4):377–430.

Flinker, A., A. Korzeniewska, A. Y. Shestyuk, P. J. Franaszczuk, N. F. Dronkers, R. T. Knight, and N. E. Crone
2015. Redefining the role of broca's area in speech. *Proceedings of the National Academy of Sciences*, 112(9):2871–2875.

Fontaine, B., D. F. Goodman, V. Benichoux, and R. Brette
2011. Brian hears: online auditory processing using vectorization over channels. *Frontiers in Neuroinformatics*, 5.

Fowler, C. A., J. M. Brown, L. Sabadini, and J. Weihing
2003. Rapid access to speech gestures in perception: Evidence from choice and simple response time tasks. *Journal of Memory and Language*, 49(3):396–413.

Furber, S. B., D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown
2013. Overview of the SpiNNaker system architecture. *IEEE Transactions on Computers*, 62(12):2454–2467.

Furui, S.
1986. Speaker-independent isolated word recognition based on emphasized spectral dynamics. In *International Conference on Acoustics, Speech, and Signal Processing*, volume 11, Pp. 1991–1994. IEEE.

Gaikwad, S. K., B. W. Gawali, and P. Yannawar
2010. A review on speech recognition technique. *International Journal of Computer Applications*, 10(3):16–24.

Galantucci, B., C. A. Fowler, and M. T. Turvey
2006. The motor theory of speech perception reviewed. *Psychonomic Bulletin & Review*, 13(3):361–377.

Gales, M. and S. Young
2008. The application of hidden Markov models in speech recognition. *Foundations and Trends in Signal Processing*, 1(3):195–304.

Garofolo, J. S., L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett
1993. DARPA TIMIT acoustic-phonetic continous speech corpus CD-ROM. NIST speech disc 1-1.1. *NASA STI/Recon Technical Report N*, 93:27403.

Georgopoulos, A. P., A. B. Schwartz, and R. E. Kettner
1986. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419.

Goldstein, L., D. Byrd, and E. Saltzman
2006. The role of vocal tract gestural action units in understanding the evolution of phonology. In *Action to Language via the Mirror Neuron System*, Pp. 215–249. Cambridge University Press.

Goldstein, L., H. Nam, E. Saltzman, and I. Chitoran
2009. Coupled oscillator planning model of speech timing and syllable structure. *Frontiers in Phonetics and Speech Science*, Pp. 239–250.

Goodman, D. and R. Brette
2008. Brian: a simulator for spiking neural networks in Python. *Frontiers in Neuroinformatics*, 2.

Graves, A.
2008. *Supervised sequence labelling with recurrent neural networks*. PhD thesis, Technische Universität München.

Graves, A., S. Fernández, F. Gomez, and J. Schmidhuber
2006. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *International Conference on Machine Learning*, Pp. 369–376. ACM.

Graves, A., A.-r. Mohamed, and G. Hinton
2013. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Pp. 6645–6649. IEEE.

Guenther, F. H.
1995. Speech sound acquisition, coarticulation, and rate effects in a neural network model of speech production. *Psychological Review*, 102(3):594.

Guenther, F. H.
2006. Cortical interactions underlying the production of speech sounds. *Journal of Communication Disorders*, 39(5):350–365.

Guenther, F. H. and S. S. Ghosh
2003. A model of cortical and cerebellar function in speech. In *International Congress of Phonetic Sciences*, Pp. 169–173.

Guenther, F. H., S. S. Ghosh, and J. A. Tourville
2006. Neural modeling and imaging of the cortical interactions underlying syllable production. *Brain and Language*, 96(3):280–301.

Guenther, F. H. and J. S. Perkell
2004. A neural model of speech production and its application to studies of the role of auditory feedback in speech. In *Speech Motor Control in Normal and Disordered Speech*, Pp. 29–49. Oxford University Press.

Haas, H.
1972. The influence of a single echo on the audibility of speech. *Journal of the Audio Engineering Society*, 20(2):146–159.

Hain, T., P. C. Woodland, T. R. Niesler, and E. W. Whittaker
1999. The 1998 HTK system for transcription of conversational telephone speech. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, Pp. 57–60. IEEE.

Hamilton, T. J., J. Tapson, C. Jin, and A. Van Schaik
2008. Analogue VLSI implementations of two dimensional, nonlinear, active cochlea models. In *Biomedical Circuits and Systems Conference*, Pp. 153–156. IEEE.

Hanson, H. M., R. S. McGowan, K. N. Stevens, and R. E. Beaudoin
1999. Development of rules for controlling the HLsyn speech synthesizer. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, Pp. 85–88. IEEE.

Hickok, G., K. Okada, and J. T. Serences
2009. Area Spt in the human planum temporale supports sensory-motor integration for speech processing. *Journal of Neurophysiology*, 101(5):2725–2732.

Hickok, G. and D. Poeppel
2007. The cortical organization of speech processing. *Nature Reviews Neuroscience*, 8(5):393–402.

Hinton, G., L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury
2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97.

Hogan, N. and D. Sternad
2009. Sensitivity of smoothness measures to movement duration, amplitude, and arrests. *Journal of Motor Behavior*, 41(6):529–534.

Holstein, G. R., R. D. Rabbitt, G. P. Martinelli, V. L. Friedrich, R. D. Boyle, and S. M. Highstein
2004. Convergence of excitatory and inhibitory hair cell transmitters shapes vestibular afferent responses. *Proceedings of the National Academy of Sciences*, 101(44):15766–15771.

Houde, J. F. and M. I. Jordan
1998. Sensorimotor adaptation in speech production. *Science*, 279(5354):1213–1216.

Huang, X., J. Baker, and R. Reddy
2014. A historical perspective of speech recognition. *Communications of the ACM*, 57(1):94–103.

Hunsberger, E., P. Blouw, J. Bergstra, and C. Eliasmith
2013. A neural model of human image categorization. In *Conference of the Cognitive Science Society*, Pp. 633–638.

Hunsberger, E. and C. Eliasmith
2015. Spiking deep networks with LIF neurons. *arXiv preprint arXiv:1510.08829*.

Hunter, J. D.
2007. Matplotlib: A 2D graphics environment. *Computing in Science and Engineering*, 9(3):90–95.

Hurford, J. R.
1991. The evolution of the critical period for language acquisition. *Cognition*, 40(3):159–201.

Ijspeert, A. J., A. Crespi, D. Ryczko, and J.-M. Cabelguen
2007. From swimming to walking with a salamander robot driven by a spinal cord model. *Science*, 315(5817):1416–1420.

Ijspeert, A. J., J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal
2013. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373.

Indefrey, P.
2011. The spatial and temporal signatures of word production components: a critical update. *Frontiers in Psychology*, 2.

Indefrey, P. and W. J. Levelt
2004. The spatial and temporal signatures of word production components. *Cognition*, 92(1):101–144.

Irino, T. and R. D. Patterson
1997. A time-domain, level-dependent auditory filter: The gammachirp. *The Journal of the Acoustical Society of America*, 101(1):412–419.

Irino, T. and R. D. Patterson
2006. A dynamic compressive gammachirp auditory filterbank. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(6):2222–2232.

Ishizaka, K. and J. L. Flanagan
1972. Synthesis of voiced sounds from a two-mass model of the vocal cords. *Bell System Technical Journal*, 51(6):1233–1268.

Iskarous, K., L. Goldstein, D. H. Whalen, M. Tiede, and P. Rubin
2003. CASY: The Haskins configurable articulatory synthesizer. In *International Congress of Phonetic Sciences*, Pp. 185–188.

Ivry, R. B. and L. C. Robertson
1998. *The two sides of perception*. MIT Press.

Johannesma, P. I.
1972. The pre-response stimulus ensemble of neurons in the cochlear nucleus. In *Symposium on Hearing Theory*, Pp. 58–69. IPO Eindhoven.

Jonas, P., G. Major, and B. Sakmann
1993. Quantal components of unitary EPSCs at the mossy fibre synapse on CA3 pyramidal cells of rat hippocampus. *The Journal of Physiology*, 472(1):615–663.

Jones, E., T. Oliphant, P. Peterson, et al.
2001–. SciPy: Open source scientific tools for Python. [Accessed 2015-12-14].

Kaas, J. H. and T. A. Hackett
2000. Subdivisions of auditory cortex and processing streams in primates. *Proceedings of the National Academy of Sciences*, 97(22):11793–11799.

Kaas, J. H., T. A. Hackett, and M. J. Tramo
1999. Auditory processing in primate cerebral cortex. *Current Opinion in Neurobiology*, 9(2):164–170.

Kandel, E. R., J. H. Schwartz, T. M. Jessell, et al.
2000. *Principles of Neural Science*, volume 4. McGraw-Hill New York.

Karuppuswamy, R., K. Arumugham, and S. Priya M.
2013. Folded architecture for digital Gammatone filter used in speech processor of cochlear implant. *Electronics and Telecommunications Research Institute Journal*, 35(4):697–705.

Khayam, S. A.
2003. The discrete cosine transform (DCT): theory and application. ECE 802 Information Theory and Coding Seminar.

Kılıboz, N. Ç. and U. Güdükbay
2015. A hand gesture recognition technique for human–computer interaction. *Journal of Visual Communication and Image Representation*, 28:97–104.

King, S., J. Frankel, K. Livescu, E. McDermott, K. Richmond, and M. Wester
2007. Speech production knowledge in automatic speech recognition. *The Journal of the Acoustical Society of America*, 121(2):723–742.

Kirchhoff, K., G. A. Fink, and G. Sagerer
2002. Combining acoustic and articulatory feature information for robust speech recognition. *Speech Communication*, 37(3):303–319.

Koch, C.
1998. *Biophysics of Computation: Information Processing in Single Neurons*. Oxford University Press.

Kohonen, T.
1982. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69.

Kohonen, T. and T. Honkela
2007. Kohonen network. *Scholarpedia*, 2(1):1568.

Kollmeier, B., T. Brand, and B. Meyer
2008. Perception of speech and sound. In *Springer Handbook of Speech Processing*, Pp. 61–82. Springer.

Kowalski, N., H. Versnel, and S. A. Shamma
1995. Comparison of responses in the anterior and primary auditory fields of the ferret cortex. *Journal of Neurophysiology*, 73(4):1513–1523.

Kröger, B. J.
1993. A gestural approach for controlling an articulatory speech synthesizer. In *3rd European Conference on Speech Communication and Technology*.

Kroger, B. J., T. Bekolay, and C. Eliasmith
2014. Modeling speech production using the Neural Engineering Framework. In *IEEE Conference on Cognitive Infocommunications*, Pp. 203–208. IEEE.

Kröger, B. J. and M. Cao
2015. The emergence of phonetic-phonological features in a biologically inspired model of speech processing. *Journal of Phonetics*, 53:88–100.

Kröger, B. J., J. Kannampuzha, and E. Kaufmann
2014. Associative learning and self-organization as basic principles for simulating speech acquisition, speech production, and speech perception. *EPJ Nonlinear Biomedical Physics*, 2(1):1–28.

Kröger, B. J., J. Kannampuzha, and C. Neuschaefer-Rube
2009. Towards a neurocomputational model of speech production and perception. *Speech Communication*, 51(9):793–809.

Kuhl, P. K., B. T. Conboy, S. Coffey-Corina, D. Padden, M. Rivera-Gaxiola, and T. Nelson
2008. Phonetic learning as a pathway to language: new data and native language magnet theory expanded (NLM-e). *Philosophical Transactions of the Royal Society B: Biological Sciences*, 363(1493):979–1000.

Kumar, K., C. Kim, and R. M. Stern
2011. Delta-spectral cepstral coefficients for robust speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*, Pp. 4784–4787. IEEE.

Kutas, M. and K. D. Federmeier
2000. Electrophysiology reveals semantic memory use in language comprehension. *Trends in Cognitive Sciences*, 4(12):463–470.

Kutas, M. and S. A. Hillyard
1980. Reading senseless sentences: Brain potentials reflect semantic incongruity. *Science*, 207(4427):203–205.

Lamel, L. and J.-L. Gauvain
1993. High performance speaker-independent phone recognition using CDHMM. In *European Conference on Speech Communication and Technology*, volume 93, Pp. 121–124.

Lawson, C. L. and R. J. Hanson
1974. *Solving least squares problems*, volume 161. SIAM.

Lee, K.-F. and H.-W. Hon
1989. Speaker-independent phone recognition using hidden Markov models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(11):1641–1648.

Levelt, W. J.
1992. Accessing words in speech production: Stages, processes and representations. *Cognition*, 42(1):1–22.

Levelt, W. J., A. Roelofs, and A. S. Meyer
1999. A theory of lexical access in speech production. *Behavioral and Brain Sciences*, 22(01):1–38.

Levelt, W. J. and L. Wheeldon
1994. Do speakers have access to a mental syllabary? *Cognition*, 50(1):239–269.

Liberman, A. M. and I. G. Mattingly
1985. The motor theory of speech perception revised. *Cognition*, 21(1):1–36.

Liu, S.-C. and T. Delbruck
2010. Neuromorphic sensory systems. *Current Opinion in Neurobiology*, 20(3):288–295.

Logan, B.
2000. Mel frequency cepstral coefficients for music modeling. In *International Symposium on Music Information Retrieval*, Pp. 1–11.

Lopes, C. and F. Perdigão
2011. Phone recognition on the TIMIT database. In *Speech Technologies*, volume 1, Pp. 285–302. InTech.

Lopez-Poveda, E. A.
2005. Spectral processing by the peripheral auditory system: facts and models. *International Review of Neurobiology*, 70:7–48.

Lopez-Poveda, E. A. and R. Meddis
2001. A human nonlinear cochlear filterbank. *The Journal of the Acoustical Society of America*, 110(6):3107–3118.

Lyon, R. F., A. G. Katsiamis, and E. M. Drakakis
2010. History and future of auditory filter models. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, Pp. 3809–3812. IEEE.

MacNeil, D. and C. Eliasmith
2011. Fine-tuning and the stability of recurrent neural networks. *PlOS ONE*, 6(9):e22885.

MacNeilage, P. F. and B. L. Davis
2001. Motor mechanisms in speech ontogeny: phylogenetic, neurobiological and linguistic implications. *Current Opinion in Neurobiology*, 11(6):696–700.

Maeda, S.
1982. A digital simulation method of the vocal-tract system. *Speech Communication*, 1(3):199–229.

Martin, A.
2007. The representation of object concepts in the brain. *Annual Review of Psychology*, 58:25–45.

McGurk, H. and J. MacDonald
1976. Hearing lips and seeing voices. *Nature*, 264:746–748.

Meddis, R., L. P. O'Mard, and E. A. Lopez-Poveda
2001. A computational algorithm for computing nonlinear auditory frequency selectivity. *The Journal of the Acoustical Society of America*, 109(6):2852–2861.

Mermelstein, P.
1973. Articulatory model for the study of speech production. *The Journal of the Acoustical Society of America*, 53(4):1070–1082.

Mesgarani, N., C. Cheung, K. Johnson, and E. F. Chang
2014. Phonetic feature encoding in human superior temporal gyrus. *Science*, 343(6174):1006–1010.

Meyer, P., R. Wilhelms, and H. W. Strube
1989. A quasiarticulatory speech synthesizer for German language running in real time. *The Journal of the Acoustical Society of America*, 86(2):523–539.

Milo, R., P. Jorgensen, U. Moran, G. Weber, and M. Springer
2010. BioNumbers–the database of key numbers in molecular and cell biology. *Nucleic Acids Research*, 38(suppl 1):D750–D753.

Mitra, S. and T. Acharya
2007. Gesture recognition: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 37(3):311–324.

Mitra, V., H. Franco, M. Graciarena, and A. Mandal
2012. Normalized amplitude modulation features for large vocabulary noise-robust speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*, Pp. 4117–4120. IEEE.

Mitra, V., G. Sivaraman, H. Nam, C. Espy-Wilson, and E. Saltzman
2014. Articulatory features from deep neural networks and their role in speech recognition. In *International Conference on Acoustics, Speech and Signal Processing*, Pp. 3017–3021. IEEE.

Mlıch, J. and P. Chmelar
2008. Trajectory classification based on hidden Markov models. In *International Conference on Computer Graphics and Vision*, Pp. 101–105.

Mohamed, A.-r., G. E. Dahl, and G. Hinton
2012. Acoustic modeling using deep belief networks. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):14–22.

Moore, B. C. and B. R. Glasberg
1983. Suggested formulae for calculating auditory-filter bandwidths and excitation patterns. *The Journal of the Acoustical Society of America*, 74(3):750–753.

Nam, H., L. Goldstein, E. Saltzman, and D. Byrd
2004. TADA: An enhanced, portable Task Dynamics model in MATLAB. *The Journal of the Acoustical Society of America*, 115(5):2430–2430.

Nam, H., V. Mitra, M. Tiede, M. Hasegawa-Johnson, C. Espy-Wilson, E. Saltzman, and L. Goldstein
2012. A procedure for estimating gestural scores from speech acoustics. *The Journal of the Acoustical Society of America*, 132(6):3980–3989.

Nam, H., V. Mitra, M. Tiede, E. Saltzman, L. Goldstein, C. Y. Espy-Wilson, and M. Hasegawa-Johnson
2010. A procedure for estimating gestural scores from natural speech. In *Interspeech*, Pp. 30–33.

Nam, H. and E. Saltzman
2003. A competitive, coupled oscillator model of syllable structure. In *International Congress of Phonetic Sciences*, volume 1, Pp. 2253–2256.

Nascimento, J. C., M. A. Figueiredo, and J. S. Marques
2010. Trajectory classification using switched dynamical hidden Markov models. *IEEE Transactions on Image Processing*, 19(5):1338–1348.

Needleman, S. B. and C. D. Wunsch
1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453.

Ng, B., A. Pfeffer, and R. Dearden
2005. Continuous time particle filtering. In *International Joint Conference on Artificial Intelligence*, volume 19, P. 1360.

Ni, G., S. J. Elliott, M. Ayat, and P. D. Teal
2014. Modelling cochlear mechanics. *BioMed Research International*, 2014.

Nieto-Castanon, A.
2011. *Simulink DIVA model*. Speech Lab at Boston University.

Nieto-Castanon, A., F. H. Guenther, J. S. Perkell, and H. D. Curtin
2005. A modeling investigation of articulatory variability and acoustic stability during American English /r/ production. *The Journal of the Acoustical Society of America*, 117(5):3196–3212.

Otake, T., G. Hatano, A. Cutler, and J. Mehler
1993. Mora or syllable? Speech segmentation in Japanese. *Journal of Memory and Language*, 32(2):258–278.

Pasley, B. N., S. V. David, N. Mesgarani, A. Flinker, S. A. Shamma, N. E. Crone, R. T. Knight, and E. F. Chang
2012. Reconstructing speech from human auditory cortex. *PLOS Biology*, 10(1):175.

Patterson, K., P. J. Nestor, and T. T. Rogers
2007. Where do you know what you know? The representation of semantic knowledge in the human brain. *Nature Reviews Neuroscience*, 8(12):976–987.

Patterson, R. D.
1976. Auditory filter shapes derived with noise stimuli. *The Journal of the Acoustical Society of America*, 59(3):640–654.

Patterson, R. D., J. Holdsworth, and M. Allerhand
1992. Auditory models as preprocessors for speech recognition. In *The Auditory Processing of Speech: from Sounds to Words*, Pp. 67–89. Walter de Gruyter.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and É. Duchesnay
2011. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830.

Plate, T. A.
1994. *Distributed representations and nested compositional structure*. PhD thesis, University of Toronto.

Pulvermüller, F.
2001. Brain reflections of words and their meaning. *Trends in Cognitive Sciences*, 5(12):517–524.

Pylkkänen, L. and A. Marantz
2003. Tracking the time course of word recognition with MEG. *Trends in Cognitive Sciences*, 7(5):187–189.

Quiroga, F. and L. Corbalán
2013. A novel competitive neural classifier for gesture recognition with small training sets. In *19th Argentine Congress on Computer Science (CACIC)*, Pp. 140–149.

Rabiner, L. R.
1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Ratanamahatana, C. A. and E. Keogh
2004. Everything you know about dynamic time warping is wrong. In *Workshop on Mining Temporal and Sequential Data*. ACM.

Rautaray, S. S. and A. Agrawal
2015. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review*, 43(1):1–54.

Richmond, K.
2009. Preliminary inversion mapping results with a new EMA corpus. In *Interspeech*, Pp. 2835–2838.

Rizzolatti, G., L. Fogassi, and V. Gallese
1997. Parietal cortex: from sight to action. *Current Opinion in Neurobiology*, 7(4):562–567.

Rizzolatti, G. and M. Matelli
2003. Two different streams form the dorsal visual system: anatomy and functions. *Experimental Brain Research*, 153(2):146–157.

Roach, P.
2010. *English Phonetics and Phonology Fourth Edition: A Practical Course*. Cambridge University Press.

Roelofs, A.
2000. WEAVER++ and other computational models of lemma retrieval and word-form encoding. In *Aspects of Language Production*, Pp. 71–114. Psychology Press.

Roelofs, A.
2008. Dynamics of the attentional control of word retrieval: Analyses of response time distributions. *Journal of Experimental Psychology: General*, 137(2):303.

Roelofs, A.
2014. A dorsal-pathway account of aphasic language production: The WEAVER++/ARC model. *Cortex*, 59:33–48.

Rosowski, J. J.
1996. Models of external- and middle-ear function. In *Auditory Computation*, Pp. 15–61. Springer.

Sah, P., S. Hestrin, and R. A. Nicoll
1990. Properties of excitatory postsynaptic currents recorded in vitro from rat hippocampal interneurones. *The Journal of Physiology*, 430(1):605–616.

Saltzman, E. and D. Byrd
2000. Task-dynamics of gestural timing: Phase windows and multifrequency rhythms. *Human Movement Science*, 19(4):499–526.

Saltzman, E. and J. Kelso
1987. Skilled actions: a task-dynamic approach. *Psychological Review*, 94(1):84.

Saltzman, E. L. and K. G. Munhall
1989. A dynamical approach to gestural patterning in speech production. *Ecological Psychology*, 1(4):333–382.

Schaal, S.
2006. Dynamic movement primitives–A framework for motor control in humans and humanoid robotics. In *Adaptive Motion of Animals and Machines*, Pp. 261–280. Springer.

Schaal, S., J. Peters, J. Nakanishi, and A. Ijspeert
2005. Learning movement primitives. In *International Symposium on Robotics Research*, Pp. 561–572. Springer.

Schiller, N. O., A. S. Meyer, R. H. Baayen, and W. J. Levelt
1996. A comparison of lexeme and speech syllables in Dutch. *Journal of Quantitative Linguistics*, 3(1):8–28.

Schluter, R., L. Bezrukov, H. Wagner, and H. Ney
2007. Gammatone features and feature combination for large vocabulary speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 4, Pp. 646–649. IEEE.

Schmidt, R. A.
1975. A schema theory of discrete motor skill learning. *Psychological Review*, 82(4):225.

Scott, S. K. and I. S. Johnsrude
2003. The neuroanatomical and functional organization of speech perception. *Trends in Neurosciences*, 26(2):100–107.

Semple, M. N. and B. H. Scott
2003. Cortical mechanisms in hearing. *Current Opinion in Neurobiology*, 13(2):167–173.

Shao, Y., Z. Jin, D. Wang, and S. Srinivasan
2009. An auditory-based feature for robust speech recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, Pp. 4625–4628. IEEE.

Slaney, M.
1993. An efficient implementation of the Patterson-Holdsworth auditory filter bank. Technical Report 35, Apple Computer.

Slotine, J.-J. E. and W. Li
1987. On the adaptive control of robot manipulators. *The International Journal of Robotics Research*, 6(3):49–59.

Smiley, J. F., T. A. Hackett, T. M. Preuss, C. Bleiwas, K. Figarsky, J. J. Mann, G. Rosoklija, D. C. Javitt, and A. J. Dwork
2013. Hemispheric asymmetry of primary auditory cortex and Heschl's gyrus in schizophrenia and nonpsychiatric brains. *Psychiatry Research: Neuroimaging*, 214(3):435–443.

Smith, A.
2006. Speech motor development: Integrating muscles, movements, and linguistic units. *Journal of Communication Disorders*, 39(5):331–349.

Smith, A. and H. N. Zelaznik
2004. Development of functional synergies for speech motor coordination in childhood and adolescence. *Developmental Psychobiology*, 45(1):22–33.

Smolensky, P.
1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1):159–216.

Sondhi, M. M. and J. Schroeter
1987. A hybrid time-frequency domain articulatory speech synthesizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(7):955–967.

Steiner, I., K. Richmond, I. Marshall, and C. D. Gray
2012. The magnetic resonance imaging subset of the mngu0 articulatory corpus. *The Journal of the Acoustical Society of America*, 131(2):EL106–EL111.

Stephenson, T., M. M. Doss, and H. Bourlard
2004. Speech recognition with auxiliary information. *IEEE Transactions on Speech and Audio Processing*, 12(3):189–203.

Stephenson, T. A., H. Bourlard, S. Bengio, and A. C. Morris
2000. Automatic speech recognition using dynamic bayesian networks with both acoustic and articulatory variables. In *Interspeech*, volume 2, Pp. 951–954.

Stern, R. and N. Morgan
2012. Hearing is believing: Biologically-inspired feature extraction for robust automatic speech recognition. *IEEE Signal Processing Magazine*, 29(34-43):170.

Stevens, S. S., J. Volkmann, and E. B. Newman
1937. A scale for the measurement of the psychological magnitude pitch. *The Journal of the Acoustical Society of America*, 8(3):185–190.

Stewart, T. C., P. Blouw, and C. Eliasmith
2015. Explorations in distributed recurrent biological parsing. In *International Conference of Cognitive Modeling (ICCM)*, Pp. 7–12.

Stewart, T. C., X. Choo, and C. Eliasmith
2014. Sentence processing in spiking neurons: A biologically plausible left-corner parser. In *Conference of the Cognitive Science Society*, Pp. 1533–1538.

Stewart, T. C., Y. Tang, and C. Eliasmith
2011. A biologically realistic cleanup memory: Autoassociation in spiking neurons. *Cognitive Systems Research*, 12(2):84–92.

Tan, Q. and L. H. Carney
2003. A phenomenological model for the responses of auditory-nerve fibers: II. Nonlinear tuning with a frequency glide. *The Journal of the Acoustical Society of America*, 114(4):2007–2020.

Tchorz, J. and B. Kollmeier
1999. A model of auditory perception as front end for automatic speech recognition. *The Journal of the Acoustical Society of America*, 106(4):2040–2050.

Theunissen, F. E., K. Sen, and A. J. Doupe
2000. Spectral-temporal receptive fields of nonlinear auditory neurons obtained using natural sounds. *The Journal of Neuroscience*, 20(6):2315–2331.

Tian, B. and J. P. Rauschecker
2004. Processing of frequency-modulated sounds in the lateral auditory belt cortex of the rhesus monkey. *Journal of Neurophysiology*, 92(5):2993–3013.

Titze, I. R.
1989. A four-parameter model of the glottis and vocal fold contact area. *Speech Communication*, 8(3):191–201.

Tresch, M. C. and A. Jarc
2009. The case for and against muscle synergies. *Current Opinion in Neurobiology*, 19(6):601–607.

Tripp, B. P. and C. Eliasmith
2010. Population models of temporal differentiation. *Neural Computation*, 22(3):621–659.

Troyer, T. W. and A. J. Doupe
2000. An associational model of birdsong sensorimotor learning II. Temporal hierarchies and the learning of song sequence. *Journal of Neurophysiology*, 84(3):1224–1239.

Ungerleider, L. G.
1982. Two cortical visual systems. In *Analysis of Visual behavior*, Pp. 549–586. MIT press.

Unoki, M., T. Irino, and R. D. Patterson
2001. Improvement of an IIR asymmetric compensation gammachirp filter. *Acoustical Science and Technology*, 22(6):429–433.

Uria, B., S. Renals, and K. Richmond
2011. A deep neural network for acoustic-articulatory speech inversion. In *Conference on Neural Information Processing Systems (NIPS)*.

Van Der Walt, S., S. C. Colbert, and G. Varoquaux
2011. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

Vijayakumar, S., A. D'souza, and S. Schaal
2005. Incremental online learning in high dimensions. *Neural Computation*, 17(12):2602–2634.

Waskom, M., O. Botvinnik, P. Hobson, J. B. Cole, Y. Halchenko, S. Hoyer, A. Miles, T. Augspurger, T. Yarkoni, T. Megies, L. P. Coelho, D. Wehner, cynddl, E. Ziegler, diego0020, Y. V. Zaytsev, T. Hoppe, S. Seabold, P. Cloud, M. Koskinen, K. Meyer, A. Qalieh, and D. Allan
2014. Seaborn: v0.5.0.

Weinland, D., R. Ronfard, and E. Boyer
2011. A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115(2):224–241.

Wen, B. and K. Boahen
2009. A silicon cochlea with active coupling. *IEEE Transactions on Biomedical Circuits and Systems*, 3(6):444–455.

Westbury, J., P. Milenkovic, G. Weismer, and R. Kent
1990. X-ray microbeam speech production database. *The Journal of the Acoustical Society of America*, 88(S1):S56–S56.

Wildgruber, D., H. Ackermann, and W. Grodd
2001. Differential contributions of motor cortex, basal ganglia, and cerebellum to speech motor control: effects of syllable repetition rate evaluated by fMRI. *Neuroimage*, 13(1):101–109.

Wrench, A. A.
2000. A multi-channel / multi-speaker articulatory database for continuous speech recognition research. *Phonus*, 5:1–13.

Young, D. L., F. L. Eldridge, and C.-S. Poon
2003. Integration-differentiation and gating of carotid afferent traffic that shapes the respiratory pattern. *Journal of Applied Physiology*, 94(3):1213–1229.

Zhang, X., M. G. Heinz, I. C. Bruce, and L. H. Carney
2001. A phenomenological model for the responses of auditory-nerve fibers: I. Nonlinear tuning with compression and suppression. *The Journal of the Acoustical Society of America*, 109(2):648–670.

Zhuang, X., H. Nam, M. Hasegawa-Johnson, L. Goldstein, and E. Saltzman
2009. Articulatory phonological code for word classification. In *Interspeech*, Pp. 2763–2766.

Zhuang, X., H. Nam, M. Hasegawa-Johnson, L. M. Goldstein, and E. Saltzman
2008. The entropy of the articulatory phonological code: recognizing gestures from tract variables. In *Interspeech*, Pp. 1489–1492.

Zlokarnik, I.
1995. Adding articulatory features to acoustic features for automatic speech recognition. *The Journal of the Acoustical Society of America*, 97(5):3246–3246.

Zwicker, E.
1961. Subdivision of the audible frequency range into critical bands (Frequenzgruppen). *The Journal of the Acoustical Society of America*, 33(2):248.

Zwicker, E., G. Flottorp, and S. S. Stevens
1957. Critical band width in loudness summation. *The Journal of the Acoustical Society of America*, 29(5):548–557.