Computer Speech and Language 83 (2023) 101538

Contents lists available at ScienceDirect

Computer Speech & Language

journal homepage: www.elsevier.com/locate/csl

Trends and developments in automatic speech recognition research

Douglas O'Shaughnessy

INRS-EMT (University of Quebec), Montreal, Quebec H5A 1K6, Canada

A B S T R A C T

This paper discusses how automatic speech recognition systems are and could be designed, in order to best exploit the discriminative information encoded in human speech. This contrasts with many recent machine learning approaches that apply general recognition architectures to signals to identify, with little concern for the nature of the input. The implicit assumption has often been that training can automatically discover the useful properties that exist in signals, with minimal manual intervention. These approaches may be suitable for some tasks such as image recognition, where the diversity of visual input is vast; e.g., an image may be any (natural or synthetic) scene that a camera views. We first examine what makes speech special, i.e., a natural signal from a complex tube, driven by a source that is quasi-periodic and/or noisy, aiming to communicate a wide variety of information, using the different vocal systems of human speakers. Then, we view how pertinent features are extracted from speech via efficient means, related to the objectives of communication. We see how to reliably and efficiently identify the different units of oral language. We learn from the history of attempts to do ASR, e.g., why they succeeded and how improved methods exploited the increasing availability of data and computer power (in particular, deep neural networks). Finally, we suggest ways to render ASR both more accurate and efficient. This work is aimed at both newcomers to ASR and experts, in terms of presenting issues broadly, but without mathematical or algorithmic details, which are readily found in the references.

1. Introduction

Automatic Speech Recognition (ASR) converts speech signals to corresponding text via algorithms. This paper examines the history of ASR research, exploring why many ASR design choices were made, how ASR is currently done, and which changes may achieve significantly better results. ASR remains a very active area of work, as recognition accuracy for tasks other than clean unaccented speech, for example, remains significantly below that of human recognition (Spille et al., 2018). A benefit of suitable modifications to ASR approaches could be increased accuracy and more compact algorithms, more efficient than current cloud-based, computationally intensive methods. As many papers in the literature give ample computational details for the algorithms noted here, this paper will emphasize explanations rather than mathematical details. Sufficient citations are given so that one may readily find further detail elsewhere. The emphasis here is on acoustic aspects of *speech-to-text* (STT); section 6 looks briefly at related issues of language modeling and rescoring. Some applications of ASR are not STT, but seek actions, rather than textual output, for input speech; one may also ask whether (and when) a given keyword exists in a speech signal. Systems such as Alexa listen constantly to their environment, and activate when they hear a keyword or key phrase.

Artificial Intelligence (AI), including ASR, emulates human behavior (Rabiner and Juang, 1993). Most recent ASR uses *artificial neural networks* (ANNs) and specifically *deep neural networks* (DNNs) (LeCun et al., 2015, Hinton et al., 2012, Bishop, 2006). The general term ANN will be used to include all network-based *machine-learning* (ML) methods, with DNNs specifically having more than three layers. In the last two decades, the ASR field (for both research and applications) has had a major shift from the predominant use

E-mail address: dougo@emt.inrs.ca.

https://doi.org/10.1016/j.csl.2023.101538
Received 13 June 2022; Received in revised form 12 April 2023; Accepted 26 June 2023
Available online 20 July 2023

Table 1
Comparison of recognition tasks (rates show approximate bits/s for inputs).

Task	Success	Flaws	Challenges	Typical data rates (kbps)
Speech-to-text (ASR)	Good with clean speech	Weak for noisy or accents	Multiple levels of encoding	64
Images/video	Many practical applications	Less successful for complex items	Variations in illumination and objects	64000
Text translation	Widely used in internet	Still below human levels	Many languages	0.06
Autonomous driving	OK in good conditions	Poor in bad conditions	Integrating sources of data	1000000

of *hidden Markov models* (HMMs) to ML methods (Kamath et al., 2019, Li et al., 2015, O'Shaughnessy, 2019). Powerful network models, using computer tools such as Kaldi, PyTorch, TensorFlow, and data2vec (Shao et al., 2020, Povey et al., 2011, Abadi et al., 2016, Baevski et al., 2022), as well as access to huge amounts of text and speech data, have led this significant change. Some ASR still uses a hybrid ANN-HMM approach (Wong et al., 2020), where ANNs are used as acoustic models for HMMs.

Simple ASR tasks such as recognition of numbers have much fewer mistakes than more difficult tasks; e.g., Aurora digit strings can have *word error rates* (WERs) less than 1%. Speech from databases such as Switchboard and CallHome has WERs up to 8% (Tuske et al., 2020), comparable to what human listeners do (Saon et al., 2017), but only for clean speech. Switchboard has conversations among strangers, while CallHome conversations are among friends and family; people speak less formally in the latter context.

If input speech has distortions, the recording microphone is distant (e.g., in many audio conference situations), or speakers have accents, ASR accuracy is greatly reduced (Picheny et al., 2019, Zhang et al., 2018), while humans often understand such speech well. WER is the standard criterion for ASR success, but it does not distinguish word similarity, and tiny errors (e.g., "like" vs. "liked") are treated the same as more serious mistakes; WER also makes no distinction about the importance of words in utterances. *Character error rate*, measuring symbol-by-symbol, is an alternative accuracy metric; the *Levenshtein distance* is a similar string metric (Yujian and Bo, 2007). While ASR research uses WER as the main performance criterion, the reliability of estimates of WER need to be judged by confidence measures (Jiang, 2005). WER is easier to measure than a criterion such as sentence recognition or understanding. However, the concept of *words* varies greatly in the world's speech, e.g., agglutinative languages have long words formed by conjoined morphemes, and other languages are more complex, such as Mandarin. Some languages have longer sentences and shorter words, which may affect overall (absolute) values of WER.

The large majority of recent ANN ASR uses distortion-free speech from native speakers. However, one series of ASR *challenges* (sponsored competitions) called CHiME deals with noisy, far-field multi-speaker conversations, and often has WERs in excess of 50% (Sun et al., 2019). Human listeners understand well at low 0-dB *signal-to-noise ratio* (SNR) (Lippmann, 1987); on such noisy speech, ASR has WERs in excess of 15% (Mitra et al., 2017). This strongly suggests that current approaches to ASR need better guidance to approach human performance.

ASR has many successful commercial applications, e.g., Cortana, Siri, and Alexa. ASR research should nonetheless strive to approach (or even exceed) human-level speech recognition. (In the related field of speaker verification, for example, automatic systems can do better than humans (González Hautamäki et al., 2015).) A major reason for the gap between automatic and human SR lies in current choices for ASR methods that emphasize simple mathematical analysis and training that: 1) discard or ignore (rather than use) valuable information in speech signals and/or 2) seek to automatically train models with (relatively) little guidance. We explore what is useful information in speech and how current ASR treats it. For insight, we compare recognition of speech to that for signals other than speech (Table 1).

Speech is highly variable (across speakers and conditions), but has a specific origin within speakers as the acoustic output of a human vocal tract (or a synthetic copy). Pertinent audio information in speech is encoded in very complex ways, far different from what occurs for most physical signals. There are multiple layers of indirect nonlinear encoding for speech (brain message → neural commands to articulators → vocal tract motion → air flow → sound pressure); this complex encoding and speech's variability make ASR difficult.

Major problems for recognition are signal interpretation and variation. This paper examines how ASR handles the immense amount of variability in received speech signals. This challenge is evident if one tries to segment the dynamic speech signal into units smaller than a full utterance. Another issue is how to fuse multiple *streams* of information in pattern recognition. For a video example, autonomous vehicles need to integrate lidar, video, and other sensor signals to recognize crucial aspects of their environment. Recognizing objects in an image is potentially simpler than ASR, as most objects have inherent physical qualities well described by shapes, textures and colors, and well modeled by local spatial analysis. In contrast, the many levels of encoding in speech (semantics, syntax, acoustics, phonetics, psychology, articulation) make speech difficult to process with a single approach. Some ASR merges multiple streams of information (Lohrenz et al., 2021).

Until recently, the best ASR (with HMMs; Section 4.4) used an integration of an *acoustic model* and a *language model*, even though the former operates on the level of *frames* (brief time periods, e.g., 10 ms) and the latter on (much longer) words. Further, neither ANNs nor HMMs exploit speech *prosody* (durations and pitch), as the range of prosody goes well beyond individual words.

A motivation for this paper is that the literature lacks good descriptions of how and why ASR works. Choices for methods and algorithms are often simply stated with little explanation. Why numerical choices for many model parameters are made is often left unstated. This paper seeks to explain both how and why ASR functions. The focus here is on acoustic signal processing; we will not examine broader issues of *spoken language understanding*, where the system output is an action (Avila et al., 2017); this usually assumes STT as a first step that produces hypotheses of text transcriptions, with a *natural language understanding* (NLP) module then classifying transcriptions into intentions.

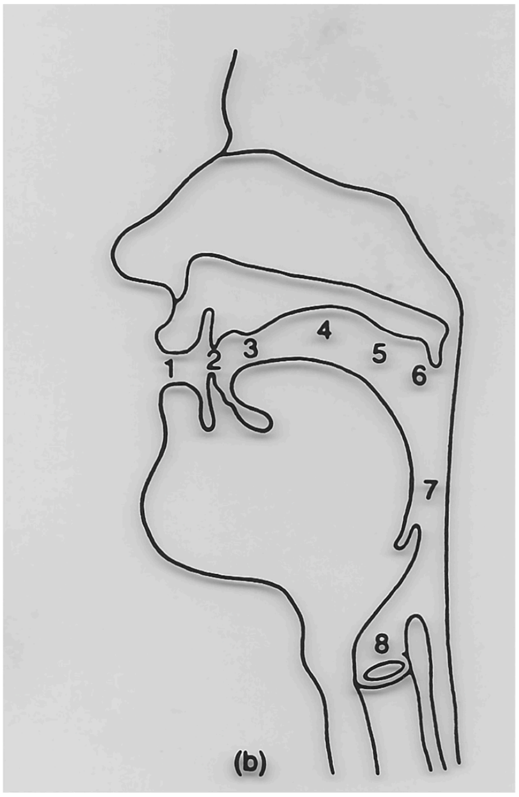


Fig. 1. Cross-section of a vocal tract: 1-lips, 2-teeth, 3-alveolar ridge, 4-palate, 5-velum, 6-uvula, 7-pharynx, 8-larynx. For a video, try www.youtube.com/watch?v=uTOhDqhCKQs.

ASR engineers are concerned with performance (i.e., accuracy, cost, delay, size), while speech scientists wish to understand *why* as well as *how*. Science can help engineering to select better methods. This paper focuses on choices made in ASR using either HMMs or ML, and specifically ANNs, including how such choices may be suboptimal and suggestions to improve such methods. We start with a discussion of what speech is, and proceed through the steps of analysis and decision processes for speech classification. We explain the choices made, and interpret methods in terms of accuracy, computation, storage, and delay.

Study of speech is useful for successful ASR. Many recent ASR methods derive directly from ML techniques learned from image recognition, but speech is vastly different from images. While the ever-increasing power of computers may be adequate for some recognition tasks using general ML tools, efficient methods targeted to speech are useful. It may be helpful to more closely emulate human intelligence; as an analogy, airplanes need wings, even if they do not flap them.

The paper is structured as follows. Section 2 examines the acoustic-phonetics of speech production and human speech perception. Section 3 briefly presents basic principles and methods of pattern recognition, with examples for ASR. Section 4 delves into methods for ASR; this includes spectral analysis such as the Fourier Transform and *Mel-frequency cepstral coefficients* (MFCCs), and methods for recognition such as HMMs. Section 5 examines neural methods for ASR; this includes basics of ANNs as well as specific architectures of *convolutional neutral networks* (CNNs), *recurrent neural networks* (RNNs) and *attention*. Section 6 briefly discusses language models for ASR. Section 7 deals with how prosody can be used for ASR. Section 8 presents ideas to improve ASR.

2. What is speech?

This section examines the nature of speech, its description in terms of phonetics, articulation, prosody, and acoustics. We note what is acoustically important for human speech production (Section 2.2) and perception (Section 2.4), including articulatory and phonetic aspects of production, showing collaboration between speakers and listeners. As speech results from excitation of a vocal tract (VT) filter and excitation involves non-segmental information, Section 2.3 examines prosody, which has been much neglected in ASR.

Speech is a complex signal with a wide range of information, communicating different aspects of the speaker and the speaker's ideas and physical status. Speech is encoded in complicated ways; speakers' ideas transform into neural commands to the VT (Fig. 1), in ways little understood. The VT moves in nonlinear fashion in response to the intention of articulating a sequence of *phonemes* (basic brief linguistic units of speech) (O'Shaughnessy, 2000). The laryngeal vocal cords vibrate at dynamic rates to inform the listener of a wide range of information (e.g., for syntactic and semantic purposes), distinct from phonemic information. In most cases, ASR sensors (microphones) do not capture this directly (e.g., via X-rays or cameras), but instead indirectly via pressure variations emitted from a

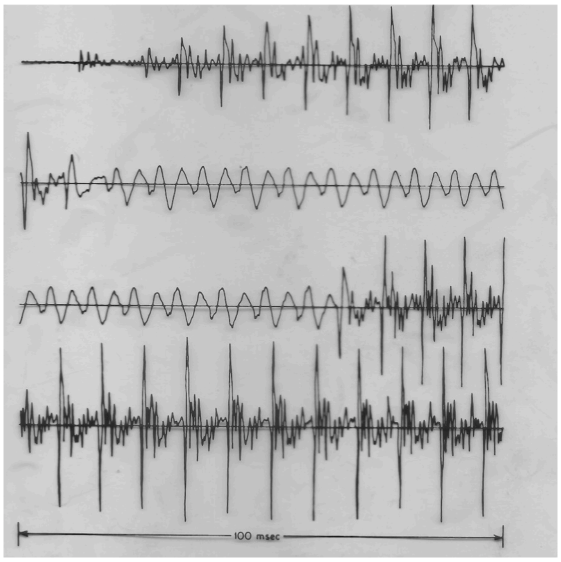


Fig. 2. Example of a speech waveform in time; four successive sections of 100 ms each. Two quasi-periodic strong vowels (10-115 and 260-400 ms) have a nasal (115-260 ms) between them (/ana/). Note the large dynamic range of amplitude between vowels and consonant.

speaker's mouth (Fig. 2). If we contrast these multiple complex nonlinear mappings for speech communication to the far more direct (and mostly linear) relationships of objects to their corresponding pixels in an image, we may appreciate why image ML has made more progress toward human-level performance than ASR has.

2.1. *Speech as communication*

For ASR, the pertinent aspects of speech relate to the ideas speakers wish to convey to listeners, but speech simultaneously also cues emotional and physical conditions of speakers (Avila et al., 2017), as well as language, dialect, and identity. All this information is highly encoded; i.e., the relationships between such information and corresponding physical aspects in speech are very complicated - much more than for many other signals, e.g., images and video. Speech is a natural system of communication that evolved in humans; it has analogs in writing and sign language, but is far more complex.

Much prior research in ML has dealt with image processing (Goodfellow et al., 2016). Relationships between objects (to be recognized) in an image and aspects of a video signal are often direct. Some *features* (e.g., contours, shapes, colors) directly correlate objects in a scene with their physical representation. This is also true for some human sounds (e.g., expressions of pain and emotion), but not for most speech, where utterances have very indirect links to the underlying ideas of the speaker. We define features as characteristics (labels) that result from a classification, while *parameters* are numerical values from a simple algorithm or process; e.g., energy is a parameter, whereas estimated resonances would be features.

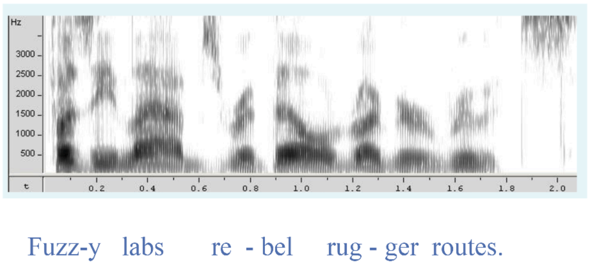
Speech occurs when sound emits from one's VT; an idea (message) in one's brain converts into a (discrete) sequence of phonemes, which is physically realized as a (continuous) sound pressure waveform. The acoustic units corresponding to phonemes are called *phones,* averaging about 80 ms in duration (O'Shaughnessy, 2000); most languages have approximately 30-40 phonemes, while the number of distinct phone realizations is far larger. (For this discussion, assume 32 phonemes per language, thus 5 bits to code each.)

The speaker's message typically concerns relationships or descriptions of objects or concepts, and is realized as a sequence of words, whose expression follows complex rules of syntax, semantics, and context. This task is simple for most humans who learn language readily, but aspects of speech production and perception are highly encoded and nonlinear, and emulations of such via *text-to-speech synthesis* (TTS) and ASR have required decades of research (Li et al., 2015, Sotelo et al., 2017, Arik, 2017).

Modern TTS and ASR have been successful for dozens of languages in commercial products (Ping et al., 2018). Yet, the quality of TTS for most utterances remains lower than that of human speech, and ASR accuracy still lags significantly what human listeners can do, especially in degraded conditions such as noise or reverberation. A major reason for weakness in ASR is the complex relationship between a speech signal and its information. In addition, most ASR methods largely ignore well-known aspects of human communication, instead treating speech as an arbitrary data sequence to which ML can assign a corresponding text. Incorporating knowledge about human speech production and perception into ASR has been difficult, owing to the practical choices made for ASR analysis. We examine the choices below.

2.2. *Human speech production*

Concepts in speech are usually examined in terms of sentences (sequences of words) that speakers utter. As each word consists of a sequence of phonemes, speakers move their VT (tongue, jaw, velum, lips) to create a series of sounds (phones) corresponding to



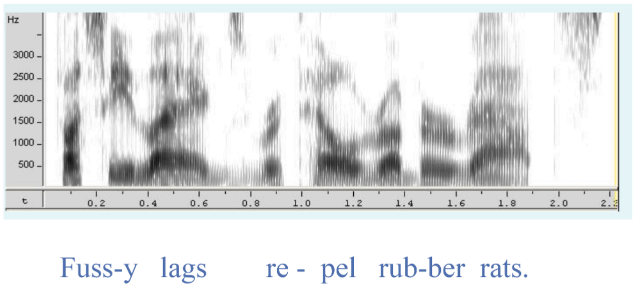


Fig. 3. Wide-band speech spectrograms (courtesy of Prof. Brian Hayes of UCLA). The darkness displays log energy. Nonsense texts: "Fuzzy labs rebel rugger routes." and "Fussy lags repel rubber rats." Vertical: Hz; horizontal: time (s). Text spacing aligns roughly with phonemes. Note: the voice bar in /z/ (top) vs. /s/ (bottom); F2-F3 joining in /g/; long VOT in /p/ (see section 2.4.2 for further discussion).

intended phonemes. Their durations vary with *speaking rate* (average rate of phones, or words, per second); this rate is a personal selection, and may slow to communicate ideas hard to understand. As speech varies greatly in VT motion, articulating some phoneme sequences is longer than others (e.g., consonants in clusters have shorter durations than in languages which have only one consonant after each vowel).

2.2.1. *Units of speech*

Each language has a different set of phonemes, but there is much overlap and several universal features. Phonemes are distinguished by: 1) *manner* of articulation, 2) *place* of articulation (Fig. 1), and 3) *voicing*. (Tone languages such as Mandarin also use pitch as a distinguishing feature; some languages also distinguish types of aspiration and clicks.) Manner of articulation has seven categories in two broad classes: a) **sonorants***: vowels, diphthongs* (e.g., "ow" and "aye"), *semivowels* (e.g., /w/), *nasals* (e.g., /m/, /n/), and *liquids* (/l/ , /r/), and b) **obstruents***: fricatives* and *stops* (the latter also called *plosives*). Vowels distinguish place by tongue height and lateral position, and by lip configuration (rounded or closed lips), while consonants have a constriction or a closure at a location in the VT: labial, alveolar, palatal, velar, pharyngeal, glottal (Fig. 1). Most phonemes can last as long as one has breath; for emphasis, speakers often elongate.

Voiced phones have quasi-periodic vocal cord vibration (Fig. 2), whose rate is the *fundamental frequency* (f0); its perceptual correlate is called *pitch*. Sonorants are voiced (except when whispered), the VT filter being excited by repeated puffs of air through the glottis. Obstruents may be voiced or unvoiced; their excitation is from the outlet of a VT constriction, where noise generated there may be modulated by vocal cord vibration. (All whispered speech is a glottal fricative, exciting the full VT with unvoiced broad-band noise). Another class of phonemes - implosive and ejective consonants (e.g., in some African languages) - abruptly suck or expel air, respectively, through a VT constriction, creating a brief, transient noise spectrally similar to stops.

2.2.2. *Vocal tract configurations and movement*

Each phoneme has a target VT shape, e.g., /i/ requires a raised and fronted tongue, while /t/ requires a complete closure at the alveolar region. These configurations are specified by positioning of the tongue and lips (for vowels), or by a VT constriction or complete occlusion. This creates a VT filter excited by periodic puffs of air (through vibrating vocal cords) or by a noise source from air exiting a VT constriction. When producing speech with VT movements, one controls this motion using auditory feedback (listening to one's own speech) to generate sounds of desired durations and spectral content, both to maximize communication with listeners and to minimize duration and effort.

Speech is dynamic (non-stationary): the VT moves among targets for phonemes in succession (Fig. 2). With faster speaking rate, the VT often undershoots targets, resulting in *coarticulation* - the continuous motion of the VT for a sequence of discrete phonemes. Context greatly affects articulation and the resulting speech sounds; e.g., in the word "strew," lips are rounded during the initial /s/, anticipating final /u/; as lip rounding lowers resonant frequencies (section 2.4.2.2), the spectrum for /s/ is different from in the word "spree." Thus, ASR can use context-dependent models for phonemes, and such acoustic context can extend several phonemes before and after. Despite the simplicity of only needing to discriminate among (approximately) 32 phonemes at any one moment, the number of models for ASR can expand greatly with context (e.g., in HMM ASR, up to 32ˆ3 *triphone* models and more if one includes wider context, although models are often shared among similar contexts) (Rabiner and Juang, 1993). Speakers typically have a preferred rate of speech (e.g., words/s), but often deviate: slower (hyper-articulation) or faster (hypo-articulation). ASR is often trained for an average articulation rate, but speech often does not match that.

In noisy conditions, speakers often alter their voice to improve communication, e.g., speak louder (*Lombard effect*) or more clearly. Unless properly addressed, this can create a mismatch between training and text conditions.

2.2.3. Speakers' focus to distinguish phonemes

Speakers vary VT shape, glottal airflow rate, and f0 dynamically to communicate. Primary information in speech resides in peaks of energy called *formants*, resonances (F1, F2, ..) at low audio frequencies (0-4 kHz) (de Wet et al., 2004). Voiced speech spectra consist of harmonics, spaced every f0 Hz (f0 is not a formant - section 2.3), whose intensity varies with the VT transfer function, which consists of these formant resonances. In Fig. 3, F1-F3 are readily seen as horizontal bands, while f0 is the inverse of the time spacing between the vertical lines, which correspond to closures of the vocal cords. (These abrupt closures incite the harmonics (O'Shaughnessy, 2000).)

Via articulation, speakers aim for a series of intended VT positions, which correspond to spectral peak patterns that listeners interpret to determine phoneme identity. For example, /i/ has a large separation between the two lowest-frequency formants (F1 and F2); the high, front tongue positioning for /i/ causes the resulting VT shape to produce this spectral pattern that both speakers and listeners associate with /i/. (Further detail on phone spectra is in section 2.4.2, which describes how listeners use spectra to perceive sounds.)

2.2.4. What do speakers NOT control?

Speech waveforms have much temporal detail related to airflow and losses (thermal and friction) that neither conveys useful linguistic information nor is intended by speakers. In their VT motion, speakers aim for targets in spectra, not to achieve any specific detail in the time waveform. Thus, speech pressure waveforms (Fig. 2) provide much less direct information to speakers' intentions than spectra do. For speech production and perception, it is far more useful to examine spectra than time patterns. This holds also for ASR, which generally uses spectral amplitude representations as input.

Phase is the angle of a spectral representation. Phase is heavily affected by minute effects of VT motion and airflow, unintended by speakers; so, phase has little use in ASR. Phase is so poorly understood that most bits in cell-phone transmission (where full signal reconstruction, including phase, is essential) are dedicated to phase. Basic *linear predictive coding* (LPC) speech at 2.4 kbps uses zero phase; the added bits in higher-quality CELP are for phase (Backstrom, 2017). For speaker verification, which is a very different application from ASR (nonetheless both currently use the same analysis methods), phase may be useful: one can recognize a person's voice from a single steady singing note; spectrally such a sound has relatively little information in its amplitude. (However, to date, phase has not been used for speaker verification.) Both ASR and low-rate speech coding rarely exploit phase [32, but 24], and our suggestions for better ASR (Sect. 8) do not include phase.

2.3. Prosody (suprasegmentals)

The previous section focused on how the identities for phonemes are cued in speech. These are the focus of VT motion, the major objectives in acoustic models for both HMM and ANN ASR, and are affected by speaker aspects (gender, size, health, language). However, speech also conveys important aspects of semantics (which words are emphasized), syntax (structure of sentences), and speaker state (emotion and health) (Yang et al., 2014, Shatzman and McQueen, 2006); those are cued by prosody: amplitude, durations, and f0 (Goldwater et al., 2010). Aspects of prosody vary with phonemic identity, e.g., vowels more intense and longer than consonants, and /i/ with higher f0 than /a/. However, primary variations of prosody relate to non-phonemic factors (uttering /i/ with highly varying f0, duration and amplitude, the sound still is perceived as /i/).

Speech often emphasizes important words (generally nouns, verbs, adjectives) vs. function words (e.g., articles). This is accomplished by the acoustic correlates of stress: dynamic f0, longer durations, and more intensity (usually a combination of all three). In many languages, including English, each word has one or more syllables that are *stressed*, as compared to other syllables in the word. This is so important to English speech that misuse of stress by foreign speakers often greatly hinders comprehension.

Languages often use word order to cue syntactic structure (e.g., subject-verb-object; noun phrases and verb phrases), but most (also) employ prosody to help listeners group local sequences of words into syntactic units. f0 has wide use; e.g., rising at the end of yes/no questions, emphasis on important words, rising at the ends of major syntactic phrases, phonemic use in tone languages, and varying in emotions (O'Shaughnessy, 2000, Kaur et al., 2021).

Speakers clearly exploit prosody, as that helps understanding. To date, ASR has largely ignored prosody (except for tone languages, where f0 is used to signal phonemic distinctions (Kaur et al., 2021)) and yet has achieved good (but sub-optimal) results. Here are examples of the utility of prosody (O'Shaughnessy, 1979):

a) "George came to yesterday" versus "George came to the city." (In the first, George has presumably fainted, then woke up.) In the first case, "to" is heavily stressed (long, with dynamic falling f0), as it acts as an adverb; in the second case, it has no stress (short and weak), being a preposition. A major difference occurs here in prosody, but these two sentences would not occur in the same conversational context, and context has major effects on expectations in communication.
b) "My brother, George, and his sister" would have three different f0 patterns on "George" depending on whether George is one of three distinct people, the brother's name is George, or one speaks to George.

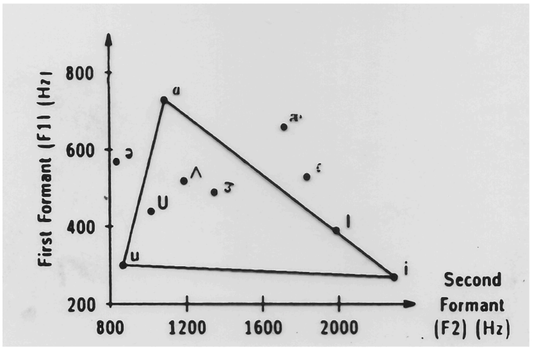


Fig. 4. Vowel triangle: average F1, F2 for English vowels.

c) In "The good flies quickly past," "good" is a noun; in "The good flies quickly passed," an adjective, but phonemically identically in both cases. To distinguish, prosody on "good" is very different in the two cases.

As further examples of the impact of prosody, important words in speech are longer, speakers slow down immediately before major syntactic boundaries, vowels are generally longer before unvoiced consonants, syllable-initial consonants tend to be longer than final ones, and *voice-onset times* (VOTs) are longer for unvoiced than for voiced stops (O'Shaughnessy, 2000). Speakers thus explicitly vary prosody in speech as an aid in communication, and human listeners pay attention to this, which can greatly improve understanding.

2.4. *Human speech perception*

Since ASR emulates human speech interpretation, we examine how audition works. For hearing, the basilar membrane inside the inner ear acts as a spectral analyzer, sending neural firings along the auditory pathway to the brain in an organized spatial fashion. Neurons linked to hair cells at the base of this membrane inform the brain about high frequencies, while neurons at its apex respond to low frequencies (O'Shaughnessy, 2000). Approximately 30000 hair cells in the cochlea of the inner ear have stochastic binary *firings*, in numbers proportional to the rectified amplitude of the membrane vibration, driven by sound stimuli to the ears (O'Shaughnessy, 2000). Thus, the brain receives information about when and at which frequencies sound energy occurs, but this data is highly encoded by the complex natural neural network.

Audition is highly nonlinear: 1) membrane spacing correlates (above 1 kHz) with log frequency (the *mel* scale, also viewed as 24 *critical bands* covering the full auditory range); 2) firings are both random and rectified; 3) after each firing, neurons do not fire for a brief latency period; 4) active neurons inhibit their neighbors (in both place and time) from firing (phenomenon called *masking*); 5) the ear has nonlinear amplitude dynamics (in a form of automatic gain control).

Details of audition are hard to link to speech perception; so we examine what acoustic information is needed for ASR. To understand speech, listeners focus on: 1) frequency ranges of prominent energy (e.g., *formants*), 2) presence/absence of *periodicity* (voicing), 3) f0 patterns (when voiced), 4) sound *durations*. These are all interpreted contextually; values for energy levels, formants, f0, and durations all have communicative utility in terms of values expected by listeners.

2.4.1. *Perceptual adaptation*

Hearing speech from an unknown person, one rapidly determines an internal *model* of the speaker, e.g., estimates of gender, age, size, dialect, and attitude. For example, children have short VTs and small vocal cords, and so more energy occurs at higher frequencies and with higher f0. Auditory *normalization* allows listeners to interpret prominent spectral peaks as being above/below expected average frequencies and relative to adjacent sounds (and similarly for f0 and speaking rate).

How this normalization occurs is uncertain (Johnson, 2020). One presumes that a listener, upon hearing an initial vowel in a speech signal, estimates an approximate range for f0 and VT resonances, and then adjusts expectations and refines these estimates as speech progresses. Listeners may attend to relative spacing of formants, which occur approximately every 1 kHz for men (a figure that scales inversely with VT length). In ASR, a speaker adaptation method is VT length normalization.

For ASR, the need to adapt to different people's speech led in the recent past to the use of *speaker-dependent* models tuned to subscribing speakers, with parameters specified through training from each speaker (Bell et al., 2021). For more general speaker-independent ASR (e.g., in most commercial services), one can use a single model designed to handle all speaker-related variability (e.g., as in much ANN ASR) or have multiple models (e.g., for men, women, children, dialects). A vast literature exists for speaker adaptation for HMMs (Bell et al., 2021, Gales, 1998). See more detail in section 4.3.2.

2.4.2. *Acoustic-phonetic features in speech*

Speakers articulate to achieve VT shapes, but we describe their results spectrally, as that is how listeners perceive. Most speech is sonorants; listeners distinguish these by the frequencies of their strong quasi-periodic resonances, primarily F1, F2, and F3.

2.4.2.1. Sonorant perception. A neutral prototype VT shape of uniform cross-sectional area (which is typical for a *schwa* vowel, used in most hesitation sounds - "uhh") has F1 = 500 Hz, F2 = 1500 Hz, and F3 = 2500 Hz (odd multiples of 500 Hz), using 340 m/s as the speed of sound and 17 cm as a typical VT length (O'Shaughnessy, 2000). F1 varies inversely with tongue height (e.g., 250 Hz for high vowels and 750 Hz for low ones); F2 varies mostly with lateral position of the tongue (850 Hz for far back and 2200 Hz for far front) (Fig. 4). Thus, F1 has a range of 500 Hz vs. 1350 Hz for F2 (e.g., using a distance in F1-F2 space would need normalization). F3 has small use perceptually (high for /i/ and low for /r/, but close to 2500 Hz for many other sounds). Upper formants in vowels are weaker and have little effect on sound discrimination (but their presence in wide-band speech adds to naturalness). For example, F4 typically only deviates from 3500 Hz by +/- 100 Hz (lower for back vowels, higher for front vowels), and is fully suppressed by communication links in telephone speech.

As sonorants use a complete VT, their perception directly exploits the strong energy at low-frequency resonances, which are readily detected in band-limited (and/or noisy) speech. Relatively simple ASR methods could apply to such long, steady, strong patterns. Obstruents, however, are more diverse and often transient, and present a more complicated recognition task, for both humans and machines, despite the fact that most ASR uses the same analysis for all speech. This detailed discussion of speech perception may motivate more diverse analysis for future ASR (section 8).

Listeners pay far more attention to detail at lower than higher frequencies. Small displacements of F1 and F2 (e.g., > 50 Hz) cue sonorant distinctions, while the categorization of obstruents tends to use larger spectral shifts, e.g., the difference between /s/ and /sh/ has more than a 300-Hz shift of a high-band cutoff.

2.4.2.2. Obstruent perception. Obstruents generate broad-band noise at the outlet of a VT constriction, which excites the oral cavity in front of the orifice. As that cavity is much shorter than the full VT (except for glottal /h/), energy is only found in spectra well above 2 kHz, and listeners attend to energy, randomness (periodic pulses or not), and what frequency range is used. Strong fricatives (those with place of articulation further back in the VT, thus exciting a longer cavity and so energy at lower frequency) are discerned by energy in higher spectral ranges than sonorants. While formants exist at high frequency, most fricatives are perceived simply by the lower-frequency cutoff of their high-band energy, rather than by specific spectral peaks. Telephone speech attenuates energy above 3.2 kHz, thus making recognition more difficult for obstruents; in such cases, context is more important: one judges telephone fricatives by their effect on adjacent sonorants.

The weakest consonants, stops and fronted fricatives, have much less energy than other phonemes, and thus are more prone to errors by ASR or by humans perceptually. Indeed, unvoiced stops /p,t,k/ have no energy at all during VT closure, and all stops cue their place of articulation via brief offset bursts and spectral transitions at their onset and offset (i.e., in adjacent sounds). Acoustic cues for this are more easily obscured via environmental noise, especially for unvoiced stops, where the energy at release is noisy aspiration, which is much weaker than sonorants (Table 3).

2.4.2.3. Perception of place of articulation. For labials, lip constriction causes all formants to decrease in frequency from normal positions (500, 1500, 2500 Hz), as the VT changes from a quarter-wavelength resonator (e.g., for a schwa vowel) to a half-wavelength resonator for labial /p,b,m/ (O'Shaughnessy, 2000). Constrictions further back in the VT have varying effects on formants, depending on the balance of kinetic and potential energy.

Besides spectral transitions in adjacent sonorants, a stop is perceived by a burst release cue, as when the mouth re-opens, a rush of air creates aspiration noise, exciting the VT in front of its place. For labials, a very short tube at the lips is very inefficient, creating a weak burst; velar /k,g/ have stronger noise (F2 and F3 together) near 2 kHz. To distinguish voicing in stops, the delay between burst onset and the ensuing start of voicing (VOT) for /p,t,k/ is > 40 ms, as compared to as little as 10 ms in /b,d,g/. During the VOT, weak aspiration noise is present. Cues for these weak obstruents are highly localized in a few frames, before and after VT closure. For vowels, cues to identity reside in the much stronger resonance energy of F1-F3 (such peaks stand out amid background noise even at 0 dB SNR).

2.4.2.4. Voicing and nasality. Phonetic feature *voicing* implies detecting vocal cord vibration, but its perception correlates also with presence of strong energy at low frequencies (in sonorants) and with VOT (in stops). For voiced fricatives, the envelope of the noise is periodic, whereas sonorants have quasi-periodic sets of waveform samples. This diversity in acoustic correlates of voicing may be why most ASR does not estimate either voicing or f0 (section 4.3.5).

For nasal consonants, the velum lowers and air exits the nostrils, not the closed mouth, which leads to wider resonances and lower energy than in vowels (owing to soft nasal tissue). While the VT transfer function for nasals has spectral zeros (unlike vowels other than nasalized vowels, e.g., as in French), listeners detect nasals primarily by their steady pattern of slightly weaker formants than vowels and by jumps in spectra when the velum opens or closes. Spectra for nasals vary slightly with place, but the main distinction for place is in spectral transitions of adjacent sonorants. This is similar to the case for weak obstruents, as well as for all fricatives in telephone speech. Thus ASR for these phones must attend to their adjacent sounds, while concentrating more on the phones themselves for phones with strong energy, which depend less on context for their identity. This is another example of the diversity of ways that phones are perceived.

2.4.2.5. Impact of phonetic diversity for ANN ASR. The diverse features for speech perception combine, in complex and nonlinear fashion, to inform listeners about the identity of phonemes, stress patterns, syntax, etc. All these complex processes in human speech communication pose a daunting task for ANNs, which use linear combinations of data, to sort all this out with simple nonlinear activations and losses (and a huge amount of training). ANNs have great power, but have difficulty in generalizing. It may be prudent to add structure to network-based ASR, to exploit our knowledge about how speech works (Section 8).

2.4.3. What is less important for recognition

To communicate information, the following aspects of speech are less important perceptually: a) relative bandwidths of spectral resonances (though nasality is cued, in part, via wider bandwidths), b) degree of tilt in the spectrum (whisper has a steeper fall-off with higher frequency than does shouted speech), c) frequency ranges with weak energy, and d) phase. Most spectral parameters used for ASR (e.g., MFCCs and *filter-bank energies* (FBEs)) (Section 4.3) are affected greatly by the first three properties (phase is discarded with use of spectral amplitude).

MFCCs and FBEs represent speech energy in broad measures, either smoothing across frequency or dividing the spectrum into many separate (highly correlated) channels. Their values change with spectral slope. They are also heavily affected by reverberation and environmental noise, and do not exploit aspects of masking. They are simple, easily specified and relevant spectral measures, but altered by acoustic channels and by aspects of speech that are largely irrelevant for communication. Alternative measures that instead focus on center-frequency values of some major spectral resonances could be far more relevant (Section 8).

2.5. Final thoughts on speech production/perception

We highlighted the complex nature of speech, how it is produced, and which features are pertinent. If considering ASR design, one should be aware of all this, to aim toward better ASR performance. The following contribute to the highly nonlinear and complex nature of audition: use of log scales (in both amplitude and frequency), discrete firings of neurons, their limited dynamic range, masking phenomena of hearing, and complex perception aspects. Many non-linearities occur in both speech production and speech perception, and all of this contributes to the complexity of speech.

3. Pattern recognition

Section 2 examined speech to review its distinct character and note pertinent features. Next, we examine how to recognize speech automatically, but first consider general recognition of signals. Like many signals, speech is readily viewed as a value (sound pressure amplitude) as a function of one dimension (time). One can feed these data values (or a transformed set), in sequence, as input to the first layer of an ANN, or to a HMM. This section examines such pattern recognition, starting with general approaches. We can map signals to classes using: a) direct comparison of simple models by their similarity, b) models that maximize likelihood, or c) parametric approaches.

Classification of patterns (e.g., speech, images) maps received signals (from sensors) to labels (names) corresponding to objects/ideas (classes) in the signals (Jurafsky and Martin, 2000). One selects output(s) from among a (discrete) set of known classes, given a (usually continuous, then sampled) input signal. The recognizer input is typically a sequence of N data values, which can be viewed as a point (or vector) in an N-dimensional parameter or feature space. For images, this input generalizes to a *tensor* (multiple dimensional matrix), such as a time series of three (RGB) 2-D sets of illuminated pixels. For speech, these values are usually samples of sound pressure at a microphone, uniformly obtained at an appropriate sampling rate (8 kHz for telephone speech (Backstrom, 2017), 16 kHz for wide-band applications such as VoIP, 32 and 48 kHz for Enhanced Voice Services, and 44.1 kHz for compact disks). The spectrum above 4 kHz has only minor additional information for ASR (e.g., distinguishing /f/ from /s/); applications that use information at higher frequencies have more complex models for small performance gain. Traditional telephony circuits do not transmit above 4 kHz to minimize cost, and listeners seem to compensate. Modern compression algorithms can compress speech much more efficiently.

3.1. General approaches

The *training* stage for recognition usually requires data collection of many typical elements of each class (e.g., words in speech, or objects in images) to be recognized. In such *supervised* training, each input (or *token*) to the recognizer provides a point in this representation (parameter or feature) space. The set of all such items for a specific class (e.g., token repetitions of a given phoneme, word, or sentence) may form regions (e.g., *manifolds*) in the space, if the space is suitable (Makhoul et al., 1989). Such regions vary enormously, depending on the consistency of the class (e.g., words from a single speaker in quiet conditions, versus from different speakers over various communication channels), and on the choice of parameters (dimensions). This variability is a huge task for recognition, as difficulty in both training and modeling increases greatly for complex spaces, which are common in most applications.

Recognition is often done by methods of maximizing likelihood or classifying in vector space (minimizing distance or maximizing correlation). If using supervised training, each input tensor has an appropriate label (class name); thus, a simple approach to recognition for a (test) input could be to locate the closest point in the model space for the input (this assumes use of a suitable distance measure in the space), and assign the name of that point.

3.2. Why not just use a table look-up?

For ASR, one might imagine storing every possible utterance and its corresponding textual label, then searching this huge "dictionary" for any given input utterance, and outputting the text found for that entry. This is not feasible, as the number of potential audio signals is immense, and there is no metric able to sort rich-variation signals into audio sets that associate with specific words.

The input for ASR is sampled audio; so one needs to consider representing a general utterance as any bit sequence (number of seconds of speech x L bits/second, e.g., using 8000 samples/s and 8 bits/sample for telephone speech), which leads to a huge memory of theoretically-possible signals, greatly exceeding future computer facilities. The same holds even if one codes each theoretically possible utterance efficiently, e.g., using 10 kb/s for typical cell phone speech, or 2.4-kb/s LPC (Backstrom, 2017). Almost all such bit sequences would be noise, not speech. However, there seems no way to prune this immense theoretical set of signals to retain only speech, nor conversely to obtain all possible utterances that could ever be spoken. Hence, table look-up is not feasible for ASR. For parallel reasons, the same holds for image recognition.

3.3. *Exemplar methods*

Versions of examples of signals can be stored as models or *templates*. This approach may require a large memory, and much computation during testing to search among all the stored models. It also assumes that the parameter set (input data samples or some processed version) is appropriate, e.g., that distance in its representation space has a reasonable interpretation of similarity. For this, we will study the parameters that have been used for ASR in Section 4. ASR in the 1970s was dominated by such exemplar approaches, most notably d*ynamic time warping* (DTW) (Rabiner and Juang, 1993, O'Shaughnessy, 2000).

DTW is a simple way to align test and reference data. As speech varies greatly in use of durations, different utterances for a given sentence have different lengths. Linear mapping of one time signal with respect to another could directly allow comparison of two speech signals via decimation/interpolation, but this alignment fails owing to variable phone durations. DTW allows local deviations in aligning, while minimizing an accumulated distance measure. DTW can be expensive in terms of both storage and run-time computation. As a result, practical exemplar ASR is limited to small-vocabulary applications, e.g., when making commands to machines (autos, appliances, handicap aids), or in some speaker verification applications.

3.4. *Maximizing likelihood*

An alternative to finding proximate models by minimizing distance in N-dimensional space is instead to maximize probability (via Bayesian learning). The set of training elements for each class would then be represented by one or more *probability density functions* (pdf's), which are often multivariate Gaussians. While such could be estimated from histograms of training data, histograms have sample complexity exponential in the feature dimension. Thus, Gaussian parameters are often estimated from sufficient statistics using the *Estimation-Maximization* (E-M) algorithm (O'Shaughnessy, 2000).

For each test input, one calculates its output class likelihood using the pdf of each of the pertinent models, and chooses the model with the highest probability. Compared to the exemplar approach, this *maximum likelihood* method has the advantage of reducing memory and computation; e.g., a simple model for each class with a mean and a variance (vs. hundreds of exemplars), requiring only one pdf evaluation, rather than many exemplar distance or similarity calculations. Difficulties with this approach include inadequate modeling by use of simple pdf's (e.g., Gaussians, to represent complicated distributions), as training data are often highly variable (e.g., various speakers, channels, and contexts). For such stochastic models, one usually presumes consistent data (e.g., independence and identical distributions), which is often not the case. For speech, there is also the major issue of timing variability, which led, in the 1980s, to the stochastic approach of HMMs replacing DTW for ASR.

For stochastic processes such as speech, we could use sequences of pdf's (e.g., HMMs) to model the variability of speech. In training, we observe, say, J *tokens* (sets of samples) from a class (e.g., repetitions of a word spoken by one or several speakers). This creates a histogram (assuming some reasonable quantization of the input parameters, into a set of L bins; L is usually much less than J, but large enough to allow adequate detail in the pdf). In modeling speech, ASR often assumes that histograms can be well represented by mixtures of Gaussians, i.e., a weighted sum of Gaussian pdf's (GMMs) (Rabiner and Juang, 1993). The motivation here is to reduce cost (less memory to store the weights, means, and variances than the L parameters of a histogram) and to improve the generalization ability of the model. Another approach is to smooth the model, as histograms usually have many random fluctuations, owing partly to inadequate amounts of training data.

It is common to estimate a pdf using a low-dimensional model (e.g., Gaussian), based on a large number of samples (i.e., high number of dimensions). Histograms are very inefficient. ASR often assumes that (class-conditional) feature distributions can be modeled by mixtures of Gaussians, which are efficiently parameterized and provide smooth estimates of likelihoods that improve generalization. While common in HMM ASR, GMMs were popular for speaker verification until i-vectors and ANNs dominated that field (González Hautamäki et al., 2015). However, GMMs have also found use in the Dirichlet process GMM, used to train *zero-resource* ASR (Heck et al., 2018, Hermann et al., 2020) (Section 5.12.1).

Traditional HMM ASR has been based on maximizing probability of a series of 10-ms frames, but more recent *End-to-End* (E2E) training approaches use *sequence training*, which directly optimizes for a criterion such as *Connectionist Temporal Classification* (CTC) (Section 5.4.1). Discriminative training can focus on distinguishing the different potential outputs, rather than maximizing likelihood.

3.5. *Parameters: data reduction*

This section examines the importance of choice of parameters for pattern recognition. Each input tensor (i.e., observed data for members of classes to recognize) requires a mapping to the name of its class. Most recognition employs some data compression or coding that transforms the *raw* (unprocessed) data input into a more compact set of parameters or features (e.g., a mapping from the high-dimensional input speech signal to a lower-dimensional *latent* set). The objective here is to be more efficient, by eliminating irrelevant information and focusing on pertinent aspects of the signal, e.g., ASR could focus primarily on frequencies of spectral peaks that correspond to VT shapes (which have clear links to phones), while speaker verification seeks features that distinguish instead among people (who may have similar VTs).

If we compare pattern recognition to coding, the latter is an application that needs efficient transmission of data, e.g., speech coding, which employs parameterization such as adaptive differential pulse-code modulation (ADPCM), log-PCM, sub-bands, or LPC analysis (Rabiner and Juang, 1993, O'Shaughnessy, 2000). Such lower-bit-rate representations can also be useful for recognition (e.g., parameters for speech related to vocalic resonances), as they allow large data reductions and characterize signals efficiently. With moderate reductions to (still relatively) high bit rates (e.g., ADPCM), the parameters chosen can remove or exploit certain redundancies in the signal (e.g., repetitions of periods in voiced speech, or predictable patterns in the spectra). An objective here is to compress high-dimensional data into a more compact latent *embedding* space where conditional predictions are easier to model. Embedding models do a parsimonious encoding mapping data into a fixed dimensional tensor.

Let's examine the sampling precision needed to describe speech. The precision for ASR, as opposed to (simpler) coding applications, is often lower, as ASR does data reduction, but not signal reconstruction. For basic telephone speech (5G systems will have wider bandwidth), one needs 8000 samples/s with 8-bit log PCM, but one could use much lower rates for ASR, where phase can be discarded. The accuracy needed in specifying features for ASR need not be more precise than can cause a perceptual change in listeners. Just-noticeable differences are typically 1-3% for amplitudes and f0, and 10% for formant frequencies (O'Shaughnessy, 2000).

4. Automatic speech recognition

We now discuss how speech is recognized, examining choices of parameters for analysis and then the HMM approach, which dominated ASR for decades. How each choice may handle the huge variability found in speech is examined. We examine models for speech that can be both efficient and intuitive; they reduce the number of bits used to represent the data. As ANN ASR now dominates the field, it will be discussed in much greater detail in Section 5.

4.1. Parameter space for speech analysis

For some complex tasks (e.g., disease identification), relevant information is hard to measure; speech, however, has very useful models. For efficient analysis, non-stationary signals such as speech are usually divided into short sections called *frames*. Ten ms is the most common frame update (*stride* or *hop*) duration for all types of speech processing (O'Shaughnessy, 2000). Generating parameters or features from speech at 100/s is adequate to represent coarticulation. At a typical speaking rate of 12 phones/s, VT motion is represented well by 100 frames/s. However, useful information in speech is not uniformly distributed in time. Vowels are often much longer than consonants, which leads to typical vowels having many (e.g., > 7) frames to convey their identity, while stops have primary discriminating data in as few as 1-3 frames (e.g., at the stop release).

A major issue for both speech perception and ASR is *segmentation*. Units in speech (phones, words) vary greatly in duration, and many unit boundaries do not have clear acoustic cues. Abrupt spectral changes signal nasal and obstruent edges, but many phone sequences are very hard to segment. In addition, word edges have no clear cues. Listeners likely do not identify individual phonemes or words, but understand globally with context. As a result, ASR avoids explicit segmentation.

Consider a set of M successive speech frames in time, $X(n)$, $n = 1,\ldots M$, where each frame consists of N parameters (e.g., MFCC or LPC values, where $N = 13$ or 10, typically; Section 4.3.2). This can form a M-by-N matrix, effectively a transformation of a spectrogram. (For image recognition, such analysis could apply to an image of MN pixels.) Some techniques transform these data into more useful representations. *Principal components analysis* (PCA) seeks eigenvectors in N-dimensional space that efficiently model the variances (Kutner et al., 2004); however, such approaches are rarely used for ASR, owing to cost and limited benefit. The *autoencoder* approach (Section 5.12.3) for ASR approximates some aspects of PCA.

4.2. Normalization for speakers and environment

Items in classes to recognize vary in ways that should not affect classification; e.g., objects in an image may vary in size, orientation, and illumination. For speech, the same word/sentence (e.g., a class to recognize) is uttered in many ways by different people, and in varying acoustic environments. To help normalize data (reduce *mismatch* between prior trained models and new test input), one may use *affine* transformations (i.e., rotation, shift, and scaling in space). These are often used to handle different VT lengths (which scale vowel spectra); they automatically seek orientations in a representation space to reduce mismatch.

Recognition benefits from several types of normalization, as trained models should handle many aspects of variability. Normalization tries to reduce effects in signals not directly related to those causing differences among classes to identify. These include channel effects, e.g., an inverse filter that models a slowly changing transmission link. For HMMs, *cepstral mean subtraction* is often used with MFCCs, where the parametric vector averaged over all frames is subtracted from each individual vector, on the assumption that such a long-term average contains irrelevant data (e.g., channel or speaker effects) for phonetic classification of each frame (O'Shaughnessy, 2000). (For ANNs, *batch normalization* adjusts outputs of previous activation layers by subtracting the batch mean and dividing by its standard deviation (Sun et al., 2018); this speeds training via use of greater learning rates by normalizing input ranges to the next layer.)

For ASR, a major normalization is *speaker adaptation*. Each person's VT is different, which significantly affects speech spectra. Resonant frequencies scale inversely with VT length, but there are significant second-order effects. Spectral mappings between speakers are non-linear; e.g., when people mature, the size ratio of lower to upper VT changes (e.g., children's pharynxes are small compared to their oral cavities (O'Shaughnessy, 2000)). An ASR model based on one set of speakers needs adjustment(s) to better match other speakers' output.

ASR often uses VT *length normalization* (VTLN) (Johnson, 2020). The simplest mappings are linear, e.g., *maximum-likelihood linear regression* (MLLR) and fMLLR (feature-space) (Gales, 1998), which use simple affine transformations. Such mappings can apply either to speech features, e.g., scaling the frequency axis for input amplitude spectra directly, or to model parameters (if these have simple frequency interpretation). *Linear discriminant analysis* is another common transformation to minimize intra-class discriminability and maximize inter-class discriminability of speech features (Gales, 1998). The 1990s saw much research to adapt speaker-independent HMMs using small amounts of data from each new speaker.

4.3. Analysis for ASR

Is speech analysis truly needed? Some recent ASR rejects signal processing (Ravanelli and Bengio, 2018), instead feeding speech signal samples directly as input to an ANN. The motivation there is to avoid choosing any specific processing, as it might distort or remove relevant information. Instead, this leaves all analysis to the automatic ML of the ANN. Such approaches can be more expensive and/or have convergence problems. Since most ASR methods do use signal processing, we examine this now.

4.3.1. Parameter/Feature Selection for ASR

Preprocessing input data increases ASR accuracy and makes training more reliable. Most methods of speech analysis were developed for the related application of speech coding, whose objective differs from that of ASR (ASR *classifies*, whereas coders *reconstruct*, but both analyze to extract relevant information). A speech coder transforms an input signal into a compressed representation, e.g., 10 LPC coefficients/frame, and then decodes this (e.g., after transmission or storage) back into speech. For ASR, this *encoder/decoder* structure is also common, using ML to automatically learn hidden (*latent*) features in compressed representations.

A major early technique (still in wide use) is spectral analysis based on the *Fourier Transform* (FT). There is no inherent gain in terms of simplicity or in transmission bit rate by use of the FT, as the basic discrete version (often realized as a *fast* FT) transforms *L* time samples of speech to *L* spectral samples, or vice versa (*L* is typically 256 or 512, the size of a *window* - Section 4.3.4). The benefit of the FT is that speech production and perception both are more readily understood (and exploited for applications) spectrally than in the time domain. Band-pass filter outputs are common parameters for ASR; these effectively approximate the spectrum, reducing high-dimensional FT to (typically) dozens of filter outputs. Truncated sinc functions can be used as responses of band-pass filters (Backstrom, 2017). ASR in 1980 used (automatic) LPC analysis, which is still used for modern cell phones (Backstrom, 2017). Unlike speech coding, ASR abandoned LPC parameters in the mid-1980s, with the advent of MFCCs (Davis and Mermelstein, 1980) (which remain in significant use for ASR and other applications well beyond speech) (Table 4). Another spectral measure is *zero-crossing rate*, which counts how often a signal changes sign (positive/negative values) - a rough estimate of its dominant frequency; this simple measure can help in detecting fricatives, which is useful in hearing aid design.

4.3.2. Mel-frequency cepstral coefficients

MFCC combines spectral analysis with aspects of audition. It transforms *N* speech samples (time waveform) into spectral amplitude (discards phase), weights all values by about 26 triangular-shaped "filters" spaced by the mel-scale, takes a logarithm, and finally inverse transforms back to the time domain. This allows some nonlinear aspects of audition (i.e., log scales, for both amplitude and frequency) to be exploited for ASR, resulting in lower error rates. (MFCC is not used in LPC, as LPC equations to minimize mean-square error do not allow nonlinear mel scaling.)

While MFCCs have been in common use since 1985 for both ASR and speaker verification, simpler logarithmic *band-pass filter energies* (BFEs) (basically MFCC, but without the final inverse frequency transform step) have been increasingly used in ANN ASR. The final MFCC step is purely mathematical, to yield uncorrelated parameters, and has no counterpart in human speech perception. Only the lowest-order MFCC parameters allow intuitive interpretation: the initial coefficient is simply total energy, and the next coefficient shows the balance between low and high frequency ranges: high for vowels and low for obstruents. Alternative filters called Gammatone and Gabor (Dutta et al., 2019) are also used to generate similar ASR parameters.

Typical HMM-ASR uses 13 MFCCs as the set of parameters to characterize each frame of speech data, often augmented by another 13 delta values and often 13 delta-delta values (Section 4.4), in a 39-dimensional vector. More recent ANN-ASR uses 40-100 BFEs.

It is useful to examine why such numbers of parameters are used. In speech production, human speakers position their VT so as to allow listeners to distinguish among phones. Vowels differ primarily in F1 and F2 by center-frequency differences of at least 100 Hz. Some languages have only 3-5 vowels, which thus spread widely in F1-F2 space, but English has 11 vowels, with some close vowel pairs such as /i/-/I/ separated by as little as 100 Hz in F1 and/or F2. Both FBEs and MFCCs use the mel-scale, which deforms the frequency axis, but roughly 40 equally-spaced frequencies yield about 100-Hz spacing for 4-kHz bandwidth (range for 8000 sample/s telephone speech), and 200-Hz spacing for 8-kHz wide-band speech.

Far fewer (13) parameters suffice for MFCCs as their values spread spectral information across all coefficients, whereas BFEs are local in frequency. For the 13$^{\text{th}}$ coefficient, the final MFCC frequency-transform step can be viewed as multiplying the mel-deformed spectrum by a 13-period sinusoid, which discriminates at approximately the 100-Hz level.

4.3.3. Formants for ASR

MFCC and BFE parameter sets are more complete sets of acoustic correlates for speech than formants (Zahorian and Jagharghi, 1993), but as few as 3-4 formant frequencies could be equivalent in utility for ASR; however, formants are hard to determine reliably.

Recent usage of up to 100 BFEs is motivated to retain much spectral detail, but this needs more computation and can make training harder, as information pertinent for phonetic discrimination is spread among more parameters.

ASR can use fewer parameters (than samples in a frame of speech, e.g., 256) because few dimensions may describe each VT configuration (as few as five binary ones, for 32 phonemes). So, the range of sizes for parameter vectors to represent frames of speech varies enormously; the smallest are difficult to determine reliably. Recent trends towards larger vectors are also found in speaker verification (O'Shaughnessy, 2000), where what distinguishes speakers is far more difficult to interpret than what is important to discriminate phones for ASR.

As spectral amplitude is of prime importance, most ASR is based on FT amplitude or a related spectral version. ASR in the 1970s (using *expert-system* approaches) tried to automatically estimate formant frequencies. However, for a given VT shape, the energy in each formant varies greatly as a function of time, owing to differences in airflow through the vocal cords, whether the cords are vibrating, and other dynamic factors. This variability makes it difficult to estimate formants automatically (Nagamine et al., 2015, Dissen et al., 2019). Modern ASR avoids preliminary classification, as such decisions risk major errors. Early cognitive ASR methods relied unduly on human design of *if–then* rules based on expert knowledge. Errors in such decisions (e.g., phone boundaries) tend to accumulate, owing to the need for many successive choices, often with little feedback to correct errors. Thus, formants as such have little use in ASR.

4.3.4. Short-time speech analysis (windowing)

A major challenge for ASR is the dynamic behavior of speech. Speech has about 12 phones/s, in which coarticulation blends sounds in complex ways. This phenomenon is far different from what occurs for images or video. An image has no time dynamics, and shows objects in a scene; objects are diverse, but usually characterized by textures and shapes inherent to the nature of the objects. Video has much continuity in image sequences, and physical laws apply to motion. Of course, objects in images vary with movement, illumination, and occlusion, but such are more readily modelled than is coarticulation in speech.

In speech, phones "morph" over time from one to another: there are no constant speech "objects." Speakers spend varying amounts of time on individual phones according to complex objectives, to facilitate communication. So we examine how to apply speech analysis dynamically. Common approaches (FT, LPC, MFCC, BFEs, Gamma-tone filters) transform a small (windowed) set of speech samples into a set of 10+ spectral parameters, on a uniform basis (e.g., 100 frames/s) during the signal.

As a major focus for ASR at any time depends on the VT configuration (and its excitation), one examines speech as a series of time *frames*, each with a limited temporal range. Frame-based ASR applies a temporal window to the dynamic speech signal, i.e., multiplies samples $s(n)$ by a window of L samples, excluding all samples outside the window. As this affects resulting spectra, we examine appropriate shape and duration for windows.

The two most common choices for ASR windows are *rectangular* and *Hamming*. The former simply gives equal weight to all L samples, i.e., the window has a rectangular shape (0 outside and 1 inside the window). The raised cosine shape of a Hamming window acts as a better low-pass filter and allows shifting of the window along the time axis of speech without abrupt edge effects. Window duration L is typically 20–30 ms, to contain at least two periods of speech (shorter windows risk unreliable features in noisy conditions; for unvoiced speech, choice of L is less important). Successive windows typically overlap 50%, to not overlook samples. L can be shorter than a pitch period, to save computation; e.g., the LPC *covariance* method uses shorter pitch-synchronous windows, while, for f0 estimation, windows include multiple periods (Wang et al., 2017).

A 100-frames/s rate (10-ms stride) is a compromise to minimize computation, while adequately tracking VT motion. This choice corresponds to a low-pass filter of 50 Hz (sampling rate of 100/s) on the time sequence of speech parameters. Having a rate well above 12/s is useful to handle abrupt VT movements, e.g., closures/openings occurring within 1–3 ms. Recent *low-frame-rate* ASR (Jiang et al., 2021) (30-ms stride) saves computation, but risks overlooking some critical brief events, such as stop bursts. Another example is wav2vec2.0 with a 20-ms stride (Baevski et al., 2020).

Most approaches to ASR apply the same analysis every 10 ms. However, useful information varies widely across time. Some vowels can range up to 200 ms, repeating many pitch periods, which contributes relatively little information. Function words, flapped /t/, and glottal stops can be very short, with rapid spectral and energy changes lasting as few as 1-2 frames.

General ML trends smooth parameters, which is not appropriate for rapid speech events. For all sensory input (including speech), people pay great attention to change. Treating each frame of speech equally, as in many approaches to ASR, does not correspond well to human perception. *Attention* mechanisms in some ANNs are one way to address this point.

4.3.5. Features and parameters

Speech *features*, e.g., voicing, formants, and f0, are exploited in human speech communication, but these have been avoided in ASR, due to the lack of reliable feature estimators, as well as trends toward fully automatic ML. (For f0, an additional reason is that f0 is difficult to use in frame-based ASR (Section 8.3).) Formant trackers and f0 detectors are never fully accurate, and errors can propagate in ASR. Thus, ASR has preferred use of only *parameters* that are calculated with no classification, such as LPC, MFCC, energy, Fourier transform, wavelets, or *Perceptual Linear Prediction* (a variant of LPC that exploits auditory factors (Hermansky, 1990)).

As f0 and voicing are not directly calculated in most of ASR, let's examine if the parameters used (MFCCs and BFEs) have sufficient information to substitute for use of voicing or prosody. MFCCs explicitly obscure f0 by averaging energy across harmonics in their triangular weightings of the speech amplitude spectra, in order to focus on VT behavior. With enough BFEs (e.g., 80-100), individual BFEs could track harmonics indirectly, but ASR does not do that. So, lacking data on f0, discriminating voicing in fricatives (e.g., /s/ vs. /z/) is difficult, as their spectral envelopes are virtually the same (a weak *voice bar* at very low frequencies can help, but that frequency range is absent in telephone speech); ASR may rely on context (e.g., language models) for discriminating voicing in fricatives. The problem could be less for voicing in stops, as the transition after burst release is weaker in voiced stops than unvoiced ones. This complexity in analysis (e.g., not being able to employ a single, uniform approach for all phones) renders ASR training more difficult.

4.4. Hidden Markov Models (HMMs)

Since the mid-1980s till recently, ASR was dominated by HMMs, which persist in hybrid approaches with ANNs (Shao et al., 2020). HMMs handle some aspects of timing variability in speech better than earlier approaches (e.g., DTW) and ANNs.

Why is temporal variability such a challenge and different for speech? Indeed, many signals to recognize vary dynamically in time, but most signals other than speech have major (physical) constraints that limit their range of temporal variability. Many diverse examples (odors, weather, sea level, gases, liquids) vary slowly, compared to the dynamics of speech.

In the 1970s, HMMs were able to integrate spectral and timing modelling into a single approach far more efficient than DTW. In earlier expert systems, classification was made frame-by-frame, but HMMs use a global evaluation of full utterances for candidate models. HMMs use Bayes' theorem to select best candidate text T for a speech utterance S, i.e., max $P(T|S)$ = max $P(S|T)\ P(T)/P(S)$. (Both S and T are sequences.) $P(T)$ is a language model, i.e., the likelihood, a priori, of text T (section 6). (Each language has significant constraints of syntax and semantics.) $P(S|T)$ is the acoustic model of the sequence of speech parameter frames, for each possible text; this conditional pdf is the essence of an HMM. Each state-conditional distribution is often a mixture of Gaussians, as there is too much variability to assume a single Gaussian. To simplify, diagonal covariance matrices are often used; this assumes no correlation in parameters (one of many model assumptions that sacrifice performance to lower cost). Use of Gaussians is for efficiency and convenience in modelling the distributions of training data.

An HMM consists of a sequence of states, each for a model of a spectral pattern of speech (corresponding to a position of the VT), with transitions between states for successive frames of speech, allowing only forward progress (in time) or remaining in a state (a self-loop). The HMM is *hidden* because its states do not model direct observations, but rather the presumed status of the VT based on the directly observed speech.

As with many engineering methods, use of HMMs requires a major compromise to obtain an integrated time–frequency representation capable of efficient ASR, i.e., its first-order Markov assumption. The conditional probability of any transition in an HMM is based on knowledge of only the immediately prior state, and thus ignores the fact that speech results from production decisions (by the speaker) occurring over several seconds, and not just at each short 10-ms frame. The actual $P(S|T)$ for most utterances is a joint likelihood for many dozens of frames of S, which is far too complex for practical ASR; the Markov assumption immensely reduces calculations. The decrease in ASR accuracy by using this (tenuous) assumption has long been acknowledged, but rarely examined quantitatively (Gillick et al., 2012), as the assumption is essential to HMMs.

The choice to use short frames and conditional independence for HMM states is explicit; any alternative use of a wider time range of analysis is computationally unacceptable, as any potential gains in recognition accuracy would be overwhelmed by additional computation, as the search space would expand greatly.

As part compensation for the very limited analysis window duration, HMMs typically use an augmented spectral vector: the basic vector has 13–16 MFCC values, which describe adequately the spectrum of a static position of the VT; often added is a set of difference (*delta*) MFCCs, showing the change between successive frames (a measure of VT velocity), and a third set of *delta–delta* values, to model VT acceleration (Furui, 1981). This larger vector requires more storage and computation, but yields better accuracy in context-dependent phonemic HMMs. This delta approach extends the effective range of analysis beyond a typical 25-ms window, but far less than the duration of the short-term memory of listeners (or the planning range of speakers over time), nor enough to model coarticulation well.

Via training, HMM transitions are assigned probabilities (to model durations) and a pdf (e.g., mixture of Gaussians) is estimated for each state's spectrum. High probabilities are assigned to self-loops in HMM graphs, because the VT moves slowly relative to the frame rate. However, transition probabilities are often ignored in calculating ASR likelihoods, because state likelihoods are far more useful.

Phoneme HMMs are usually *context-dependent,* representing phonemes in the context of their immediate neighbors, have 2–4 states each, and a strong chance of several frames per state. (The consequent geometric pdf for duration, however, does not correspond well to actual duration densities, but it is inherent to the first-order Markov assumption.) A small likelihood may be assigned to the chance of skipping a state; i.e., training with slow speech may lead to including states that may not be needed in faster speech. The use of context here often leads to many hundreds of models, which are inter-related. Such models are often *tied* (e.g., sharing parameters for similar contexts), for efficiency. Modern connectionist systems do not need such *context decision trees*.

HMMs often train to maximize likelihood, but some use mutual information between observation data and the sequence of reference words. *Maximum mutual information* (MMI) is defined as the log likelihood of the ratio of the joint probability $P(S,T')$ and the sum of all $P(S,T\_i)$, where S is the speech input (usually in the form of features such as frames of MFCCs), T_i are all possible text transcriptions, and T' is the candidate text. ASR can estimate MMI using N-best lists and lattices. In Lattice-free MMI (Shao et al., 2020), a combined acoustic and n-gram phone model encodes all possible word sequences in a single HMM graph. There is far more to say about HMMs, but little research is now done in the area; so we limited this section to basics.

4.5. Evaluation databases

Utterances are non-stationary and may have long durations. Between pauses, utterances can range up to 10 s, but most ASR is tested on much shorter utterances (e.g., an average of 3 s in the Switchboard database (Tuske et al., 2020)). Evaluations rarely explore cases where prosody may be critical to understanding (Goldwater et al., 2010). Few (if any) databases for ASR are designed to test

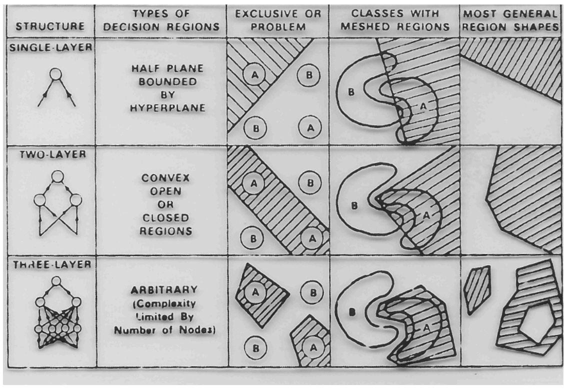

STRUCTURE
TYPES OF DECISION REGIONS
EXCLUSIVE OR PROBLEM
CLASSES WITH MESHED REGIONS
MOST GENERAL REGION SHAPES
SINGLE-LAYER
HALF PLANE BOUNDED BY HYPERPLANE
TWO-LAYER
CONVEX OPEN OR CLOSED REGIONS
THREE-LAYER
ARBITRARY (Complexity Limited By Number of Nodes)
A
B

Fig. 5. Possible classification regions for MLPs, for 1-3 layers (from (Lippmann, 1987)).

discriminability of either phones or words, e.g., to comprehensively examine how well ASR performs on speech with small phonetic differences (such as among /p,t,k/), or to discriminate among confusable words. Most databases instead focus on tasks (e.g., travel agents (ATIS), news broadcasts), informal conversations (CallHome, Switchboard), or voice commands (datasets used by private companies) (Table 6). TIMIT used hand-designed sentences to balance use of phonemes according to their average frequency in speech (e.g., /z/ less often than /t/) (Toledano et al., 2018), but databases rarely examine fine differences.

The popular read database LibriSpeech (Luscher et al., 2019) has a training subset of 360 hours of speech, from 921 speakers, i.e., approximately 1 million seconds of data. As ASR networks often have many millions of trainable parameters, this means many parameters utilize an average one second of speech; so, many parameters may derive from one phone's speech, on average.

The difficulty and complexity of tasks range widely. Simple digit sequences (e.g., telephone numbers: Aurora database) are easy to recognize and have very high accuracy (English 0-9 have diverse vowels and most are monosyllabic). Read speech (e.g., news) is more uniform than spontaneous speech. Practical applications for ASR are for spontaneous speech, which has much more disfluencies than read speech. Far-field and reverberant speech is more difficult. Of clean speech databases, CallHome and Switchboard have been traditional challenging spontaneous phone conversations. Few databases exist for degraded conditions; researchers often simulate such by adding noise or echo to clean speech, but speakers in actual bad conditions alter their voice to improve communication.

5. Artificial Neural Networks (ANNs) for ASR

We now discuss the predominant ASR method of ANNs: basic form and operation (sections 5.1-5.6), choices of architecture, e.g., CNNs, RNNs, and Attention (sections 5.7-5.9), E2E models (sections 5.10-5.11), and types of supervision (Section 5.12). Simple speech examples are given to illustrate ANN operation. Justification for three or more layers is discussed. We compare natural and artificial networks. *Activation functions*, the objective to minimize (*loss*), and *gradient descent* training are described. For generalization of models, many *regularization* methods are noted. The now popular E2E methods are justified based on simplicity of architecture, better performance, and a way to directly optimize for the final performance criteria. Finally, supervised vs. self-supervised training is examined.

An ANN does a nonlinear mapping of input to output (Bourlard and Morgan, 2012). It is roughly based on the action of natural neurons. In its simplest form, i.e., a *multi-layer perceptron* (MLP), a well-trained model optimizes the combined locations of hyperplane boundaries of (often complex) class regions in a representation space. To recognize data, an MLP defines boundaries for classes as portions of hyperplanes (Fig. 5). Each model neuron (perceptron node) determines one hyperplane. The simplest network consists of one or more *layers* composed of neurons.

A natural neuron has many dendrites feeding signals to an axon, whose output is binary. Each neuron *fires* (i.e., outputs a brief pulse ($<$ 1 ms), viewed as a binary value of 1) when the weighted sum of its inputs exceeds a specific *bias* threshold (otherwise, the output remains at 0). So, the model is $y = \varphi(w \cdot x + b)$, where w is a N-dimensional *weight* vector, b is a scalar bias vector, x is a N-dimensional vector of input parameters, $\varphi(.)$ denotes an element-wise nonlinearity such as the rectifier max(., 0), and y denotes the scalar output of the node. The nonlinear threshold operation φ is called an *activation function.* Human neural networks are massively complex (hundreds of billions of neurons), whereas ANNs are generally arranged in a small number of *layers* (sets) of nodes, where the outputs of each (lower) layer furnish the inputs to the next (higher) layer. Intermediate layers in an ANN are called *hidden* because they are neither observed as input data (lowest layer) nor provide output classes. Both artificial and natural neurons can have a much wider range of possible computations than the simple integrate-and-fire model.

ANNs are not *black boxes*, i.e., opaque systems that transform input data to output classification, with no access "inside." A designer can access all aspects of the network, just as one can for all parameters in an HMM. However, their massive size for many practical cases makes them extremely difficult to debug.

5.1. *ANN Operation*

Human NNs operate asynchronously, but each node in an ANN forwards its output in synchronous fashion to successive layers of nodes (networks with feedback would delay for one cycle). If a node has binary output, it can classify by assigning 0/1 to each side of a hyperplane in N dimensions, as the sum $w \cdot x + b$ forms a hyperplane in such space. This is a *correlation* filter, where each dot product of the input and weight vectors results in a number (feature) with potentially useful information, which may allow simple classification. Thus, we can view the output of each node as a feature of input data, deriving from a linear, weighted combination of its inputs. If using relevant inputs (e.g., F1 and F2 values for a vowel, to take a useful, but impractical, case), the node output could be a very helpful feature (e.g., indicating a high, front vowel). In practical ANNs, each node yields a tiny fraction of the needed information to classify, and many nodes yield useless information (the outputs of such nodes could be effectively discarded by use of very low weights).

5.1.1. *Simple example tasks*

Consider a brief speech signal with one word among M possible words. For very simple cases, e.g., where $M = 2$ spoken word classes could form two distinct regions in some suitable parameter space, a single perceptron could separate two classes with N features (e.g., using a binary classifier such as a *support vector machine* (SVM) (Fosler-Lussier et al., 2013)). Assume we use only the words YES and NO, and extract only one pertinent acoustic feature (i.e., $N = 1$); in this overly simple case, the feature could be an average (over the full duration of the word) of F2 (as YES has higher F2 than NO).

In more practical cases, we obtain many ($N >> 1$) parameters from input speech (e.g., 13 per frame, for 30-50 successive frames), and regions for YES and NO overlap in N-dimensional space, owing to variability in speech across speakers and environments. (A probabilistic approach (e.g., HMM) would choose the word-model yielding the higher value of two pdf's, and interpret any pdf overlap as the likelihood of ASR error.)

Models of classes of interest (obtained from training) for a practical recognition task rarely separate fully in N selected parameter dimensions. For applications to classify M words (e.g., $M = 10$ digits spoken in isolation, with intervening pauses to allow simple segmentation, to simplify this example), at least M-1 hyperplanes would be needed to assign M separate manifolds in the N-dimensional space. Owing to data variability and to choice of parameters for analysis, the M words have many more than M regions of complex shapes.

Another simple case would discriminate five vowels (/a,e,i,o,u/; many languages have these vowels). Assume we have estimated F1 and F2 features for such vowels (spoken in isolation). Three hyperplanes could partition F1-F2 space suitably, one for front-back (F2) and two for tongue height (F1). So, three nodes in the (only) hidden layer of the MLP would input values for F1 and F2, and emit one bit for each vowel candidate (e.g., 101 for /e/, as it is forward and medium-height). The output layer with five nodes, one for each vowel, could weigh the 3-bit pattern to yield a *one-hot* output (a sequence with one value non-zero), to select which side of the 3 hyperplanes each vowel corresponds to.

5.1.2. *Interpretation of MLP weights and layers*

MLPs have far more power than making simple decisions. While each node's output is binary, the choices of weights and bias control use of the node's inputs. Raising a weight indicates its input feature is more useful than another (assuming normalized inputs), and lowering a bias allows the node to require a (weighted) combination of the input features (but not all), in order to fire. So, higher bias means a node is less selective.

In our 5-vowel example, the weight on the first link (front-back feature) controls the importance of that link. If all three nodes' input features had equal utility, then equal weights (1,1,1) and a bias of -2 assures the output requires all three input features to be present. These parametric choices are made via training automatically (data-driven), with no human intervention.

Intuitively, training a node's weights displaces the hyperplane associated with that node so that the likelihood of a correct output label increases. Similar iterative processes are used when designing a codebook for CELP speech coding, as in modern cell phones (Goodfellow et al., 2016), or via estimation-maximization for HMM training. Thus, the architecture of a MLP is determined by the statistics of the training data, resulting in a deterministic model, whereas HMM ASR creates stochastic models.

For structure, a double-layer perceptron (one hidden layer and one output class set, not counting the input set of nodes) produces convex manifolds in the parameter space (each class could be bounded simply by all surrounding hyperplanes), which is only adequate for the simplest tasks. Adding a second, hidden layer of nodes to the network, for which the outputs of the first (input) layer feed into the second layer, allows the flexibility to generate non-convex regions for classes (boundaries at arbitrary angles) (Fig. 5). Each polygonal region for a class is formed by intersections of hyperplanes, with gaps allowed inside.

While the numbers of nodes in the input and output layers are directly specified by details of the recognition task, those of the hidden layers vary widely and are chosen empirically. It is assumed that lower layers develop low-level *latent* features (e.g., contour lines for image recognition; phonetic features for speech), while higher layers should form more complex and discriminative features that humans might interpret.

One may specifically limit the numbers of nodes in intermediate layers, trying to create *bottleneck* features (Bai et al., 2018): layer outputs with more relevance to improved classification (ten Bosch and Boves, 2018). The simple examples above (yes/no; 5 vowels) suggest that speech ANNs may generate features related to phonetics, but that rarely occurs for practical ASR. Even where research shows such features, this is rarely exploited to improve ANN design.

Consider a "perverse" set of units to classify, where items have no common features, or for which one uses features that have no utility. In such a case, to separate (i.e., classify) possible versions of M classes, given N training examples, one might require up to N separate manifolds, i.e., one manifold for each exemplar. ANNs have enough power for this case, but it would require an excessive number of parameters. Interpretation can be simple for very shallow networks (SVMs), but DNNs are very opaque (extremely difficult to debug).

5.2. Deep Neural Networks (DNNs)

Before 2008, most ANNs were MLPs with two hidden layers, as that is sufficient to classify very complex data, and efficiency in training DNNs is recent. There has been a great recent expansion of ANNs from three layers to several, often dozens—hence, *deep* NNs (DNNs) (LeCun et al., 2015, Goodfellow et al., 2016). Earlier, exceeding three layers was viewed as neither necessary nor useful nor practical, as computers were not as powerful, there were not enough available data, and training methods were elementary.

A typical ANN for ASR may have six or more layers and 512–1024 nodes in each of its hidden layers (Xiong et al., 2018, Qian et al., 2016). Below we examine more advanced architectures (CNNs, RNNs, and autoencoders). Accuracy often improves with more depth and complexity, rather than having more nodes in a shallow ANN; e.g., keeping the total number of nodes the same, accuracy is better in a MLP than with a broad one-layer (shallow) model.

5.3. Human and Artificial Neural Networks

ANNs differ greatly from natural NNs. Human neurons have binary output, i.e., have a baseline (viewed as 0) and jump to 1 briefly when their weighted inputs exceed a threshold. ANNs instead use an *activation* function (e.g., sigmoid, tanh, ReLU) that either smoothly (and monotonically) goes between two values (e.g., 0 to 1), or, for a ReLU (*rectified linear unit*), is 0 for low input and linear above a bias. Smooth activations allow calculation of derivatives for gradient descent in training.

Architectures for ANNs are often different from biological networks. Besides non-binary activation functions, nodes can vary significantly, e.g., use a *pooling* filter (linear average; nonlinear median smoother) or maximum output (*maxout*). Inputs could be modified to combine in ways other than a linear combination; the latter is perhaps most relevant to natural modes of perception, which focus on correlation and basic filtering, but other options could be explored to tease out features that could be useful in recognition.

5.4. ANN training

ASR should train models efficiently. ANNs generally use a *back-propagation* method: a network having an initial estimate of node multiplier weights and biases (usually specified at random, but sometimes via pre-training on unlabeled data) incrementally updates these parameters, minimizing a loss function. Supervised training uses the (correctly) labelled outputs to revise parameters so as to increase recognition likelihood for seen data.

5.4.1. ANN training criteria: minimize cost/loss

The most commonly used loss functions for training neural ASR models are: cross-entropy (in hybrid NN/HMM models and in attention-based E2E models), CTC (marginalizing over all alignments), *transducer loss* (for RNN-T and other transducer models), lattice-free MMI, and minimum expected WER (a word-level minimum Bayes risk loss).

In ML, the loss function can vary greatly, but is often *cross-entropy* (CE), which balances accuracy with efficient training (Xiong et al., 2018). CE uses the log-likelihood of the training data - a criterion similar to minimizing least square error (quadratic or L2 norm). In principle, one may prefer to train to directly minimize WER, the average probability of ASR error over all training for a model. WER is often not used for ML training because the objective function is not differentiable, thus not allowing gradient-based techniques. For estimation of discriminative models, one may minimize the classification error rate (a smooth approximation to WER) or MMI. (HMMs train their parameters in a similar way, called estimation-maximization.) However, WER is less convenient than a sequence-level criterion (e.g., state-level minimum Bayes risk (sMBR) (Wong et al., 2020)). ASR often compensates for not using WER during training via discriminative sequence training, which fine-tunes CE-trained models with criteria like sMBR.

Entropy can be measured by the number of bits to describe a random event; it increases with variance:
$H(X) = -sum[P(x) * log(P(x))]$. Thus, CE serves as a measure to minimize in ANN training; CE concerns the difference between two pdfs (e.g., where one might describe an optimal model for a stochastic source, e.g., a spoken word). A KL divergence calculates the relative entropy between two pdf's, whereas CE uses the total entropy between distributions. CE can be viewed as the average number of bits to encode data coming from a source having one pdf when using another model instead. KL divergence and CE are equivalent when using a fixed training data set.

Finding a suitable loss function for ASR is difficult.

A (perhaps unfair) comparison to speech coding shows that LPC uses very simple quadratic error, readily minimized by the solution to linear equations. This is due to the much simpler mean-square error to be minimized in LPC, whose derivative leads directly to a closed-form solution of linear equations, via inversion of an autocorrelation matrix. Shallow classifiers such as conditional random fields and SVMs also benefit from simple convex error functions (Fosler-Lussier et al., 2013), but they are much less general than DNNs. Whichever set of parameters is used in ASR, the loss function is complex and non-convex (even for simple tasks). There is much correlation among parameters, which ANNs have difficulty exploiting properly (Ghahremani et al., 2018).

5.4.2. ANN training: gradient descent

Training ANN parameters usually relies on derivatives of the loss function, e.g., gradients. A non-saturated activation such as ReLU facilitates this calculation. (Step functions, which transition abruptly from 0 to 1, would have an undefined derivative.) Loss gradients are back-propagated through the entire network. In DNNs, this requires examining long chains of network parameters (weights and biases), where effects are the product of many partial derivatives (via the chain rule). In many cases, such products can become very small, leading to the *vanishing gradient problem*, whereby relevant effects are lost owing to small values over many network layers. Especially in recurrent networks, relevant context may extend over many frames (in acoustic models) or words (in language models). The converse effect, i.e., *exploding* gradients (large derivatives in some nodes), is more easily solved by clipping excessive gradients. Calculating derivatives need not be very precise, as they merely control the rate at which the search proceeds; thus, one can save computation by examining only a mini-batch of samples, and/or use far less precision in calculations (Hubara et al., 2017).

In principle, ASR can be trained on all available suitable speech. In practice, data are often divided into separate batches. *Batch size* is the number of speech samples input to an ANN during each *epoch* (training cycle). In *full batch* mode, all learning data are evaluated each epoch, but in *mini-batch* mode, parameters are updated on smaller amounts of data, reducing computation. The use of mini-batches yields another example of "gradient noise" that generalizes ANN models.

5.4.3. *Variants of gradient descent*

Stochastic gradient descent (SGD) typically calculates model error and updates based on examples in the training dataset. Variants of SGD are *Adam* (Kingma and Ba, 2014), *Adagrad, RMSProp, Momentum* (Sutskever et al., 2013), and Nesterov's Accelerated Gradient (Sutskever et al., 2013). These all use the simple criterion of iteratively updating parameters to minimize loss, following a derivative of the loss as a function of the parameters. Update changes between epochs are controlled by the *learning rate* or step size. A small learning rate increases computation, as it takes longer to attain the final model, while too large a rate risks repeatedly overshooting a target. This rate is often reduced (*annealed*) in later epochs, as one approaches a target (i.e., a local minimum). Given the complexity of most ANNs, one never finds a global optimal model.

There are many variations on SGD, to make the search more efficient and to improve the model. In each epoch, all weights and biases are changed a small amount, proportional to both learning rate and the gradient (slope of the loss as a function of the parameters). For example, Adam (*adaptive moment estimation*) does first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments (Kingma and Ba, 2014). It uses little memory, and is invariant to diagonal rescaling of the gradients.

In comparison, the method RMSProp with momentum generates parameter updates using a momentum on the rescaled gradient, while Adam updates directly from running averages of the first and second moments of the gradient. They differ also in use of a bias-correction term. For momentum methods, each parameter is updated based on velocity, not position. Gradient methods explore different efficiencies, much like one might descend a hill, e.g., with switchbacks, if beneficial. Larger initial learning rates tend to yield better generalization, if they do not cause divergent models.

5.4.4. *Output layer*

The final (upper) layer of an ANN can have binary outputs (to classify), or instead have a *softmax* output, rather than 0/1. Softmax applies the exponential function to each element of its input vector, and then normalizes these values by dividing by the sum of all these exponentials; this ensures that the sum of the components of the output vector is 1. This allows the output to be viewed as a likelihood, not a class choice.

The output layer of ANNs can specify recognition classes, or, alternatively, produce other representations that facilitate classification (hybrid HMM-ANN ASR; e.g., scaled likelihoods for *senones*—context-dependent phonemic HMMs). For the first case, ideally, the set of outputs should have one node with an output of 1 (for the class with the desired classification), and all others not firing (i.e., 0), i.e., a one-hot vector. For the latter (hybrid) case, the outputs could be likelihoods associated with different VT positions, which then function as state probabilities when using HMM for the actual classification. A softmax output layer can be used for recognition at the phoneme sub-state level. A major motivation for such hybrid ANN-HMM ASR is that ANNs often work best for static patterns, while HMMs have a direct way to handle timing variability.

5.5. *ANN model generalization*

For better training in ANNs, the loss function to minimize is usually augmented with a nonlinear *regularization* term, to reduce *overfitting* (this is where the model becomes too close to the limited data seen in training, and thus less general and more subject to error on future data) (Sun et al., 2018, Tuske et al., 2019). Overfitting occurs in many optimization problems with small amounts of training data and models with many parameters. For speech and many other signals, the ANN objective function is highly complex (e.g., has an immense number of local minima). Given the complexity of modern models, use of only a simple error criterion could force the models to focus overly on the "seen" training data, and not generalize enough for future, "unseen" data. Such cases have low bias, but high variance.

5.5.1. *Regularization*

Regularization perturbs or diversifies training, to limit excess model flexibility. Its analog is constrained optimization, as occurs in linear programming, where one searches for a point in a space that maximizes a performance function, while remaining within the confines of certain constraints. One common form of regularization randomly omits consideration of nodes during ANN training (Srivastava et al., 2014). Other methods delete parts of training data and/or add distortion.

Regularization is a partial solution to a general recognition problem, i.e., *mismatch* between training and test data. One can never access enough relevant training data to anticipate all possible future inputs, including deteriorations such as noise and reverberation.

Recognizers often train on artificially modified data via *data augmentation:* modifying actual training samples, in an attempt to generalize ANNs (Sun et al., 2018). For example, for images, one can translate (shift), rotate, reflect, skew, scale, crop, and/or occlude, as well as alter illumination. For speech, one can add other sounds (e.g., noise) or reverberation, and/or randomly omit portions of data in both time and frequency (Ravanelli et al., 2020).

Dropout is the most common form of regularization (Srivastava et al., 2014). During training, individual ANN nodes are randomly deactivated with a probability (e.g., 50%). Each iteration of back-propagation is run on many different randomly modified versions of the original network, and the final network effectively uses an average. *DropConnect* is similar to dropout, but deactivates network connections rather than nodes by temporarily zeroing out weights (Wan et al., 2013). In *Zoneout*, some hidden states in RNNs (notably LSTMs) are forced to keep their previous values (Krueger et al., 2017).

For *Weight decay*, the additional component added to the loss function is the L2 norm of the trainable parameters; this tends to make the weights remain small (Srivastava et al., 2014). The regularization term is the sum of the squares of all the link weights in the ANN, multiplied by a hyperparameter (Section 5.6).

Common distance metrics are the L1 and L2 norms. Both L1 and L2 regularizations lead to a reduction in the weights with each iteration. For L2, the *weight reduction* is multiplicative and proportional to the value of the weight. In L1 regularization, weights shrink by a fixed amount each cycle. L1 regularization leads to ANNs in which the weights of most of the connections tend towards zero, i.e., a sparse network.

Early Stopping ceases further updating when performance on a held-out training set stops improving (Bishop, 2006). Similarly, one can reduce the learning rate when held-out performance ceases to improve.

Label smoothing interpolates output class targets with a uniform distribution during training (Goodfellow et al., 2016).

Scheduled sampling feeds the previous label prediction during each training epoch, rather than the correct one (*ground truth*) (Goodfellow et al., 2016). It introduces regularization by stochastically departing from *teacher forcing* (training RNNs with output-to-hidden recurrence at each time step (Kanda et al., 2013)), and instead using the predicted output of an autoregressive model as the input.

Batch normalization speeds up training by standardizing the input distribution to each layer in an ANN (Goodfellow et al., 2016, Sun et al., 2018). The output of each node (before application of the activation) is normalized by the mean and standard deviation of the outputs calculated over the examples in the mini-batch. (As this method needs running averages of summed input statistics, it is less easy for RNNs, which have varying sequence ranges.)

In *weight normalization*, each weight vector w is re-parameterized in terms of a parameter vector v and a scalar g, and SGD uses those parameters (Srivastava et al., 2014):

$w = g\, v \,/\, ||v||$, where $||v||$ is the Euclidean norm of v.

Curriculum learning presents training examples in a specific order, often in increasing order of difficulty (Goodfellow et al., 2016). *Tempo perturbation* artificially slows or accelerates the input training speech signals, in a form of data augmentation (Tjandra et al., 2018). *Sequence noise injection* adds noise to the input speech, while *weight noise* adds noise directly to the network parameters (Goodfellow et al., 2016).

SpecAugment removes sections of frequency and time, and/or warps time for data augmentation (Park et al., 2019).

Residual layers (ResNets) are ANN variants with skip connections between layers (Tang et al., 2018). By making direct connections from block inputs to outputs, such connections minimize vanishing gradient problems.

5.5.2. Speaker adaptation

Mean and variance normalization often occurs prior to applying inputs to an ANN (LeCun et al., 1998). Given enough speech (from a single speaker), one calculates statistics for each parameter, and then normalizes each parameter's values to zero mean and unit variance. This is effectively linear scaling. For ANNs, such normalization is not theoretically needed, as models can (in principle) handle all types of variability, given enough training data and a large enough network. However, this is an example of how prudent prior processing can help to reduce training complexity.

Model-based adaptation relies on a direct update of ANN parameters (Bell et al., 2020). Such direct action for a large ANN can result in very large speaker-dependent parameter sets, a computationally intensive adaptation process, and overfitting. An alternative method here is *speaker-adaptive training*, which appends speaker-specific *auxiliary features* to the usual network inputs (e.g., 40-dimensional fMLLR-adapted features), computed for each speaker at both training and test stages. One typical set is *i-vectors* (e.g., another 100 parameters), which can be regarded as basis vectors that span a subspace of speaker variability; they are common for speaker verification applications (González Hautamäki et al., 2015), and help for adaptation in ASR (Saon et al., 2013). Such large features sets may be reduced by PCA.

In the dynamic version of *Layer Normalization* (Kim et al., 2017), one adapts ANNs to different speakers and/or environments by use of dynamic scaling and shifting parameters based on the input sequence. A feed-forward neural network operates on each input speech signal to produce an utterance summarization feature vector.

5.6. Hyperparameters

Besides learning rate, ANNs have other *hyperparameters* - design choices to improve accuracy and accelerate processes, e.g., *momentum*, numbers of *epochs*, numbers of nodes and layers, and choice of activation function. Until recently, hyperparameters were chosen empirically and not learned via ML. However, given that their choices greatly affect performance, recent attempts use ML to train hidden layer dimensionality and connectivity as well (Kim et al., 2020).

5.7. Convolutional Neural Networks (CNNs)

Many architectures exist for ANNs. DNNs largely use the elementary MLP structure described above, but with more than three layers and with many variations: we will study CNNs, RNNs, and attention. Basic DNNs are *fully-connected feedforward (FFNN),* i.e., all nodes in each layer feed into all nodes in each successive layer. This is not ideal, as network complexity increases with more nodes, and patterns often have many localized aspects, which do not require such general structures. For example, an object may occupy a small portion in an image, and identifying it may need no input from other regions; identifying a vowel using BFEs may only need small subsets of them.

As relevant information for recognition is often local, CNNs process data locally. Data values are assumed to be in a tensor form, the simplest being a vector (e.g., series of speech samples, for which we use a 1-D convolution). More common is two dimensions (2-D): a matrix of pixels for gray-scale images, or a wide-band spectrogram for speech. (For color signals or video, we can extend to higher dimensional tensors.) The data pass through 2-D filters, before further processing at later layers (Qian et al., 2016). Here, *convolution* is similar to filtering (e.g., the output of a linear time-invariant filter is a convolution of input and impulse response), but simply means multiply-and-add: in 2-D, convolution usually multiplies a small square weight matrix (e.g., 3×3) with data in every neighborhood, summing results spatially.

The filters often use averaging over small ranges of pixels, called *receptive fields*; the range is called a *kernel*. In some cases, called *dilation*, the kernel inserts empty values to save computation and to regularize. These trainable convolutional kernels may be low-pass, high-pass, or more complex; e.g., *pooling* yielding the maximum product in the kernel. The main idea for CNNs is that most data are relevant locally; fully-connected networks allow all nodes to contribute to every other node, but the nearest ones provide far more utility than distant nodes. This localizing benefit is less clear for speech, where pertinent information for words often extends widely in time and frequency.

CNNs were first developed for image recognition, where one objective is to enhance edges and contours. Often, in images, what should look like smooth boundaries instead appear ragged, owing to sampling quantization and/or other distortions. CNNs have translation equi-variance and assume stationary statistics, which are appropriate for images. When applied to speech, CNNs also use an image: smoothing data in a spectrogram of the speech (Fig. 3). However, edges in spectrograms are less relevant as appropriate features for ASR (vs. object edges, which are salient features for images). Nonetheless, CNNs in ASR can smooth small variations in local data sequences (both in time and in frequency); such deviations are likely unintentional by the speaker and/or related to artifacts of the data acquisition or the acoustic channel environment.

CNNs try to summarize key features at each layer through nonlinear activation following each affine operation, with small patches at lower layers and large patches at higher layers. The final layer is usually fully connected to do classification, which needs a global operation. In image recognition, kernels may vary in size within a layer (*inception*) to handle varying sizes of objects, but that seems inappropriate for speech.

CNNs typically alternate convolutional layers with pooling layers that replace each 2-D set of nodes with single nodes, using either a (linear) average or a (nonlinear) maximum value (unlike in decimation, there is little concern about aliasing). This down-sampling reduces the size of the network, accomplishes a smooth learning, and can reduce effects of overfitting.

CNN is one of several ANN versions that modify the basic model to avoid overly specific training. An extension of such ideas is the *maxout* layer in ANNs, where the usual thresholding function is replaced with a simple maximum (Sun et al., 2018). Another option is a *vector-quantized* (VQ) layer in the ANN, where the output is from a VQ codebook, as in CELP speech coding. When using non-smooth activations, which are not differentiable, gradients are approximated using the *straight-through* estimator (Bengio et al., 2013) in the backward pass of training. Such pragmatic choices are designed to modify the basic NN architecture to enhance performance.

Section 5.8 deals with flexible exploitation of time sequencing in ASR, e.g., as HMMs use self-loop transitions to allow for variable phoneme durations. A type of CNN that predates recent neural developments is time-delay NN (TDNN), which applies 1-D convolution in time (e.g., a trainable nonlinear finite-impulse-response filter). For each time step, a TDNN receives inputs from past and future frames (usually regular dilations); these models have no recurrence (feedback) (Peddinti et al., 2015). Each higher layer in a TDNN is a weighted sum of nonlinear transformations of a window of lower layer frames. The idea is for lower layers to determine simple local patterns, while higher layers try to find complex patterns over broader contexts. As frames in a range of speech vary greatly in utility, a variable weighting scheme called attention (Section 5.9) can allow ASR to focus in non-uniform fashion.

5.8. Recurrence

CNNs exploit local correlations in data, while recurrence is useful for longer-range patterns. Pertinent information in speech is distributed unevenly in both time and frequency; e.g., phone durations are highly variable. (Indeed, most pitch periods within vowels repeat very similar information each period; speakers may decrease phone durations in situations where listeners can be expected to anticipate aspects of what may be said.) Sections of speech with low energy are far less useful than portions with strong formants, and coarticulation and prosody affect speech over tens and hundreds of frames, respectively. Both ANNs and HMMs have difficulty to exploit this non-uniform information. Most ANNs do best with static patterns, as in images, where the input data are often constrained by relatively simple physical laws, e.g., one could linearly expand or contract the space to zoom in or out.

5.8.1. Recurrent Neural Networks (RNNs)

Recurrent neural networks (RNNs) have architectures with feedback, and are thus more complicated than simple MLPs (Graves et al., 2013). They use distributed hidden states that store much information about the past input, and have non-linear dynamics

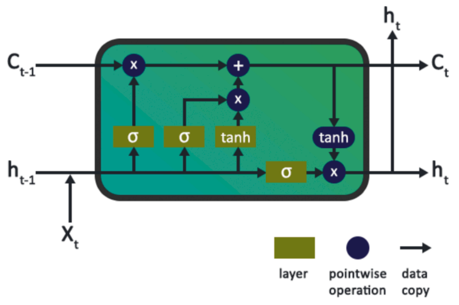


Fig. 6. LSTM cell (from C. Olah, http://colah.github.io/posts/2015-08-Understanding-LSTMs/)

allowing model updates in complicated ways. The RNN input is a sequence of N feature vectors ($x_1, \ldots, x_N$), one per time frame. The RNN computes a corresponding sequence of hidden state vectors ($h_1, \cdots, h_N$) using

$$h_t = g(W_x x_t + W_h h_{t-1} + b_h),$$

where W_x and W_h are weight matrices and b_h is a vector of biases (all consisting of parameters obtained by training). The function $g(\cdot)$ is a non-linear *squashing* function (e.g., hyperbolic tangent), to compress the values to a simple, finite range such as (0,1). Thus, the output probabilities for an RNN are conditioned on all h_t, allowing recognition decisions to use all the past.

5.8.2. Long short-term memory (LSTM) RNNs

One recurrent method for ASR is *long short-term memory* (Sainath and Li, 2016). It has been used for both acoustic modeling and language modelling. Why "short"? ASR often emulates short-term memory in perception, whereby listeners internalize significant temporal portions of speech (many frames, up to a few seconds of speech) in some analyzed form in their brain; such serves well when a listener has difficulty to decipher a complex utterance in real time. The "long" (of LSTM) means that the RNN's range of analysis can extend widely, e.g., beyond the limited scope of kernels or context-dependent HMMs (even with delta parameters, HMMs only consider a few successive frames of speech).

LSTMs have recurrent connections to allow for long retention of cell state information, with sigmoid gates called *input, output*, and *forget* (Graves et al., 2013) (Fig. 6), like state transitions in HMMs. LSTM employs forget gates to allow large variations in phone durations (via *highway* connections (Xiong et al., 2018)). Like HMMs, such methods try to accommodate timing. A more direct use of timing in ASR would base classifications partly on durational factors. RNNs accommodate different timings, interpreting words spoken at different speeds as corresponding to the same text. The information in each cell in an LSTM resides in a state, whose inputs pass through input gates, which control if data may enter. The state unit has a linear self-loop with weight controlled by a forget gate. The output of each cell can be shut off by an output gate.

Bidirectional LSTMs transform input feature vector sequence x_t and previous hidden state h_{t-1} into the next hidden vector sequence h_t, forward and backward in time. This recursive operation allows retention of features far into the past. They allow flow backward and forward. When used in language models, conditional likelihoods of sequences of words need not follow the unidirectional flow of time in speech (this, however, requires examining full utterances, i.e., not for real-time ASR).

Unlike in other ANN architectures, self-loops in LSTMs generate network paths where the gradient can flow for many frames, thus allowing long-term context handled better than in HMMs. The weights on self-loops are conditioned on context, rather than remaining fixed; thus time scales can be changed dynamically (because time constants are output by the model itself). Each cell state connection to previous cell states has only linear operations, which allows LSTMs to retain short-term memories over long durational contexts.

A *Gated Recurrent Unit* (GRU) is a simplified version of LSTM that combines forget and input gates into a single update gate, and merges the cell and hidden states (Cho et al., 2014); it handles long-term dependencies, has a simple architecture, and is easy to implement.

5.9. Attention

Despite the diverse distribution of information in speech, much of ASR uses localized analysis, e.g., windows examining a few brief frames of data at a time, HMMs with first-order memory, and CNNs with small kernels. To attempt to allow wider-range analysis for ASR, RNNs allow ASR to focus on relevant speech segments over a much longer time range than do HMMs, but the weights of mechanisms in RNNs usually decay with longer ranges of time. As well, RNNs have as many hidden state vectors as the length of the input sequence, which increases computation.

A recent recognition method called *attention* allows much freedom to choose various data to focus on, to better exploit informational distribution (Chan et al., 2016, Vaswani et al., 2017). It selects aspects in an input sequence (speech/image/text) that are more useful to update hidden model states, to make a prediction for the next output value. Attention is interpreted as a *correlation* of relevant information, and is calculated via matrix operations (e.g., dot products) that combine several terms: *queries* (inputs), *keys* (features), and *values* (desired outputs, weighted by the attention), with a softmax function to obtain normalized values for attention weights. Parallel (multiple) attention layers are called *heads*. Queries come from an earlier decoder layer, and memory keys and values come from the output of an encoder.

5.9.1. *Attention Mechanisms*

Attention is usually expressed as: Attention(Q, K, V) = softmax $(QK^T)V$, i.e., choose a value (among the elements of V) based on relative correlations of the keys, K, to the queries, Q. Matrix Q is typically input data multiplied by a learned matrix W, with similar learned matrices for K and V. The input is usually an embedding matrix (e.g., in NLP, stacked codes for each token in a sentence, where the vector length may be the maximum length of a sentence). The equation above effectively averages products (correlations) of all three terms over a local neighborhood. The product QK^T correlates Q and K, and softmax selects the desired element from among the V values.

This allows selecting (emphasizing) from among all network nodes in a given ANN layer, without applying a forgetting factor that decays with time (as is often assumed in many cases of temporal recognition). This has found success in machine translation, where word positions vary greatly across languages (e.g., unlike English, German has sentence-final verbs, which relate to distant sentence-initial subjects). The hidden state in attention carries information about all nodes, and so can have great flexibility. This general approach to attention is expensive, as all states are compared; a windowing approach can be more efficient (Cho et al., 2014).

Attention may combine three sources of information (Chan et al., 2016): decoding history from the previous state, current context, and attention weight from the previous step. For speech, attention may help with the segmentation problem of ASR, by grouping together spectrally similar frames of speech. Here, keys might be phoneme-like features. Durational effects of this approach are more valid than assumed geometric pdf's for durations that are inherent in HMMs. For time alignment, attention acts in a way similar to DTW, but avoids DTW's huge disadvantages of storage and computation.

5.9.2. *Self-attention*

If one uses a *self-attention* layer, the keys, values and queries all come from the output of the previous layer in the encoder (Pham et al., 2019). Self-attention can be efficiently implemented with batched tensor multiplication, and allows direct conditioning on short- and long-range context, without passing information through many intermediate states (unlike RNNs). An advantage of self-attention is its ability to scale with the input sequence length. In CNNs, on the other hand, analysis range is directly limited by small kernel size. For LSTMs, range is limited by the vanishing gradient problem. Self-attention is computationally advantageous compared to recurrent structures because intermediate states are no longer computed serially.

As self-attention computes the similarity of each pair of inputs, memory requirements grow quadratically with sequence length. For efficiency, this may require using a frame rate lower than the standard 100/s, e.g., by doing decimation (*downsampling* among the sets of nodes). If we compare to decimation for coders, where prior lowpass filtering is needed to avoid aliasing, note that ASR is not concerned about signal reconstruction, and so a filter is not needed. ASR may simply delete frames or stack them with the retained frames.

RNNs directly handle phoneme sequences, but self-attention has no direct way to model sequence position. For NLP applications (e.g., machine translation), self-attention can use trigonometric position encoding (Vaswani et al., 2017) for learned word embeddings; these are simply representations (embeddings) of absolute position within an input sequence. For speech positioning, basic attention lacks a way to handle temporal relationships. One approach is relative positional encodings, e.g., Conformer (Gulati et al., 2020).

Self-attention in ASR helps to group similar successive frames into phone-like units. This is what self-loops in HMMs do, but HMMs impose a geometric pdf on durations, which is a poor model; there is no such imposition for attention. Attention seeks to model context relevance (Sperber et al., 2018); in NLP, words with significant semantic and syntactic content are the input units, and thus broad interpretations of linguistic correlations are viable for NLP. For ASR, attention may serve a role that both CNNs and RNNs try to do: it helps group spectrally-similar frames (as CNNs average), while also handling variable durations (as RNNs do). This may be another case where applying ML techniques from one domain (e.g., images, NLP) to another (frame-based acoustics) may have fewer benefits, as the models only exploit correlation information, ignoring many other aspects of speech. Few ASR papers dealing with attention explain intuitively what attention is doing.

5.10. *End-to-End ASR*

A major constraint in applying ANNs to many AI problems is that basic ANN design maps a fixed-length input sequence to a fixed-length output sequence. For image recognition, the input range is specified by the camera resolution, but speech varies greatly in duration, and so does the ASR textual output. This has led to a significant class of ML algorithms called *sequence-to-sequence* (S2S) models, using *end-to-end* (E2E) training, to explicitly handle variable-length input and output. We use S2S and E2E synonymously for ASR, although S2S is a general paradigm for many ML problems, while E2E ASR emphasizes a uniform approach without distinct acoustic and language models, nor rescoring. One major type uses RNNs (Tjandra et al., 2018, Inaguma et al., 2020, Sutskever et al., 2014) in an encoder-decoder architecture, for both NLP and ASR.

5.10.1. *Connectionist Temporal Classification (CTC)*

CTC is a particular way to relate variable length non-discrete sequences to fixed-length symbolic sequences. It maps an input $X = [x_1,x_2,\ldots,x_N]$ (e.g., series of N speech frames) sequence to a set of probabilities for all possible corresponding output sequences $Y=[y_1, y_2,\ldots,y_L]$ (e.g., a text of L symbols) (Graves et al., 2006). CTC has a disadvantage of assuming that successive output symbols are independent (HMMs assume a strong, conditional independence of input frames, whereas CTC symbols are conditionally independent given the latent state of the neural network, which can depend on the entire input sequence); its training also can show peaky activations over time. CTC learns its acoustic model while mapping a sequence of input frames to a same-length sequence of output labels. This allows training sequence models without needing a frame-level alignment of the target labels. This output set is allowed to repeat characters and have*blank* symbols; for the output sentence, all symbol repetitions and blanks are collapsed to yield the (much shorter) final character sequence.

Early E2E systems (Hannun et al., 2014, Miao et al., 2015, Soltau et al., 2016) show that ASR can be efficient without context-dependent sub-word phone units that require a pronunciation lexicon, and without any *language model* (LM). A related approach to CTC, but which has an LM and an autoregressive component, is the *recurrent neural network transducer* (RNN-T) (Li et al., 2019). It has a prediction network but, unlike CTC, does not assume conditional independence between predictions at different time steps. Attention-based E2E models (section 5.11), which do not use blanks, appear to have better accuracy (Inaguma et al., 2020) than CTC and RNN-T, but use global attention score normalization over full encoder memories.

5.10.2. Other aspects of E2E approaches

The difficulty of exploiting the diverse features of speech in one cohesive method for ASR has led to numerous approaches, including using ANNs to estimate HMM state models (*hybrid* ASR). The recent trend for ASR of E2E systems (Audhkhasi et al., 2017) transforms a sequence of waveform samples or (more often) spectral patterns (one per frame of speech) directly into text characters via ANNs (Graves et al., 2013, Chiu et al., 2017, Zhang et al., 2016), thus leaving the ANN training to automatically discover all the pertinent details of speech information. These models combine most calculations into a single uniform process, rather than have separate components such as acoustic and language models (Sutskever et al., 2014, Collobert et al., 2016). The output may be a transformed sequence, e.g., the text corresponding to a speech input, or a translation to another language.

One may classify E2E systems as: (1) CTC-trained models, which have no decoder component; (2) RNN-T models, with autoregression (prediction network); and (3) attention-based encoder-decoder (AED) models, which use attention to handle alignment between the input features and output labels. Each of these can have different network architectures: CNNs, (B)LSTMs, or self-attention models like *Transformer* and *Conformer* models (section 5.11), to act as model components. Conformer encoders employ self-attention and convolution instead of BLSTM.

5.11. Attention-based ASR

AED models use attention to interface between an encoder and a decoder (Prabhavalkar et al., 2017). The initial model encoding step learns an acoustic representation from input speech. Attention-based ASR lets the decoder adaptively select hidden states for focus (Chiu and Raffel, 2018, Chorowski et al., 2015). Such E2E (Nussbaum-Thom et al., 2016) is now a popular way to simplify the ASR process. These models avoid the conditional independence assumption of speech frames in HMMs, as well as avoiding having separate ASR components and rescoring (Graves et al., 2013). They may use an encoder (RNN layers acting as an acoustic model), an attender (for time alignment), and then a decoder (RNN layers also, but to predict output sub-word sequences, to which the input speech could be compared; i.e., acting as language model). The first component maps speech input to a higher-order feature representation (e.g., via a multi-layer LSTM), and the last one replaces pronunciation and language models, using an embedding of the previous predicted output unit (e.g., word) to make each successive prediction. Such models may be based either on RNNs or on self-attention models (transformer or conformer).

Like other E2E ASR, most attention models examine entire utterances, and thus not real time. This allows use of self-attention layers that process the entire input, rather than the usual unidirectional (forward in time) models of most ASR (Nussbaum-Thom et al., 2016). Real-time (*streaming* or online) applications constrain the range of input to examine. In comparison, speaker verification (and recognizing aspects such as health and emotion) may examine whole speech utterances, as only a single decision is needed.

E2E handles inputs and outputs of varying length (speech has many speech samples or frames *in*, few text characters *out*). Early E2E used an encoder RNN to map an input speech sample sequence into a fixed-length vector, which a decoder RNN then mapped into an output sequence, one token at a time. This approach has been supplanted by attention-based methods where the encoder produces a sequence of representations. The attention mechanism provides feedback: the hidden states of the decoder RNN generate an attention vector over the input sequence of the encoder. This behaves as skip connections that allow better gradient training in an RNN, and better handle variations in speaking rate. For streaming ASR, one needs to output hypothesized words as they are uttered (Sainath et al., 2019).

Basic RNNs preclude parallelization, which is crucial to real-time ASR; so some recent E2E methods called *Transformer* (Vaswani et al., 2017) do not use recurrence and rely only on attention, so as to focus on critical aspects of the decision process over a wide range. In recurrent networks (e.g., LSTMs and GRUs), the input is processed sequentially; the hidden state at position t+1 depends on the hidden state from position t. However, Transformer has no notion of token sequence, but instead uses positional encodings in a separate embedding table (*embeddings* are simply mappings for discrete units such as phones, words, or positions to codes). Transformer ASR requires more computation, in terms of optimization, network structure, and data augmentation (Karita et al., 2019).

Transformer uses E2E architecture. A stack of encoding cells, each with self-attention followed by a FFNN, transforms an input feature sequence x to a hidden vector h. With h, a related stack of decoders then generates an output sequence one element (e.g., word) at a time. At each step, the decoders use the previously emitted elements as additional inputs. Each decoder has an attention block between the self-attention and the FFNN. No recurrence is used. Both encoders and decoders use *multi-head attention* (MHA) and *position-wise* 1-D CNNs instead of RNNs. Encoder outputs h are attended by each decoder block. MHA has several foci (heads) of attention, where each head can generate a different attention distribution.

5.12. Types of supervision in ASR learning

Recognition is most accurate when trained from labeled data, i.e., *supervised* learning. ANNs have done well in supervised learning of high-level feature representations from labeled data by using layered differentiable models. However, manually-labeled datasets are limited and often expensive, and many languages have little available data. *Unsupervised* ASR trains only on speech with no corresponding text and on unspoken text. *Self-supervised learning* (SSL), on the other hand, exploits other information sources, while *semi-supervised* methods use small amounts of labelled data, to create *pseudo-labels* using initial seed models.

5.12.1. Unsupervised Learning

The *Zero Resource Speech Challenge* does ASR for five languages with no labeled data (O'Shaughnessy, 1979, Johnson, 2020). One task does unsupervised learning of phone-like unit representations and another task focuses on SSL spoken *term discovery* (UTD) (Thual et al., 2018). The former (Bengio et al., 2013) trains on unlabelled speech to find a mapping to a compact representation (presumably useful features) that helps discriminate among linguistic units (e.g., subwords). It searches for meaningful word- or phrase-like patterns.

UTD typically uses a *k-Nearest-Neighbor* search (Thual et al., 2018) or a variant of DTW, called *segmental* DTW (Park and Glass, 2008), that identifies similar acoustic sub-sequences within two local vector time series. Groups of matching speech segments are then clustered using graphs; such units may be syllables, words, or phrases. Heuristics limit the number of DTW alignments. Optimization simultaneously learns a segmentation and clustering solution that minimizes the joint cost of operations.

To get suitable acoustic models in low-resource languages, it helps to learn frame-level feature representations that can distinguish sub-word units of any language. This must be robust to non-linguistic factors, such as speaker change and channel distortions (Sun et al., 2018).

5.12.2. Self-supervised Learning

SSL pre-trains ANNs on unlabeled data to learn general representations, which are then used to improve ASR accuracy with further training on small amounts of labeled data for a target language. Models of lexical learning based on SSL divide data into phonetic units. They "discover" phonetic features, but rarely learn longer-term phonological processes. Averaged latent representations can correspond to relevant phonetic units. SSL clusters speech data automatically into acoustic units presumed to share some features relevant for ASR. Quality of these units is evaluated via *same-different tests* (e.g., DTW distance between pairs deemed "same" if distance is less than a threshold). Older methods stacked several one-hidden-layer *Restricted Boltzmann Machines*, undirected graphical models that can model dependency among a set of random variables using a two-layered architecture.

SSL often defines surrogate *pretext* tasks achievable using only unlabeled data. SSL can predict masked-out (SpecAugment) discrete speech encodings by a contextualized representation from Transformer (Sadhu et al., 2021). SSL can be contrastive (e.g., *contrastive predictive coding* - CPC) or reconstructive (*Autoregressive Predictive Coding* - APC).

E2E models convert variable-length speech input into a set of hidden context states, which are then decoded into a target sequence (Sutskever et al., 2014). Unlike Markov models that limit range to one prior state, *autoregressive* methods condition on all the past, i.e., the hidden context states retain information about all prior states. (The term "autoregressive" comes from filtering, where recursive (feedback) units in digital filters allow long responses, as in all-pole LPC.) In APC (Chung et al., 2019), one predicts the spectrum of a future speech frame from its history (optimize via L1 regression), and in CPC (Oord et al., 2018), MMI is used to learn to discriminate latent representations of true and impostor samples. CPC optimizes by discriminating future frames from negative samples using a noise-contrastive estimation loss.

5.12.3. Autoencoders

An *autoencoder* is an unsupervised ANN that learns to compress (encode) data and reconstruct the data from the reduced representation. Self-supervised encoding finds models (hidden vector representations called *encoder embeddings*) in a latent space, while the decoder step is supervised to match input and output data (the difference, or loss, may be mean square). The encoder often consists of bi- or uni-directional LSTM layers. The *decoder* step generates output as close as possible to the original input (so as to verify that the encoder captures enough feature information, in the content, to reconstruct the signal with good quality).

Correspondence auto-encoder (cAE) (Kamper et al., 2015, Last et al., 2020) is a related version of SSL that trains on pairs of speech segments likely to be instances of the same word or phrase, e.g., from UTD (Heck et al., 2018, Thual et al., 2018). Each pair's frames are aligned via DTW, and pairs of aligned frames are presented to the cAE. After training, a middle latent layer serves as the learned feature representation.

5.12.4. Other types of self-training

SSL can do *Acoustic Word Embedding* (AWE), i.e., seek a fixed-dimensional representation of a variable-length audio signal in an embedding space. This relates to textual word embeddings that create similar vector representations for semantically similar words (Kamper et al., 2015). AWE models acoustic similarity instead of semantic similarity. Comparisons in fixed-dimensional embedding spaces allow simpler distances (e.g., Euclidean).

Problem-agnostic speech encoder (Ravanelli et al., 2020) learns multi-task speech representations from raw audio by predicting a number of handcrafted features such as MFCCs, prosody and waveforms. It encodes speech into a representation that is fed to different regressors and discriminators. Regressors compute standard features such as spectra or LPC, whereas discriminators employ positive or negative samples, which are trained to separate by minimizing cross-entropy.

5.12.5. Generative adversarial networks

Contrasting true and false samples of data is common in deep generative models, which synthesize data such as images and audio, e.g., *variational autoencoders* (VAEs) and *generative adversarial networks* (GANs). GAN architecture (Goodfellow et al., 2016) uses two networks: *generator* and *discriminator*. The generator produces data from a low-dimension latent space; the discriminator learns to distinguish between "real" training data and "false" generator outputs. Generators train to maximize the discriminator's error rate, while discriminators minimize their error rate. GANs can represent phonetically or phonologically meaningful information.

On the other hand, a VAE is an autoencoder with encoding distribution regularized during training to ensure its latent space has good properties to generate reasonable speech samples (Bie et al., 2021). The VAE model uses a simple joint likelihood, $P(X,Z) = P(X|Z)\, P(Z)$, where X is the input vector and $P(Z)$ is the prior of Gaussian latent variable Z, with dimension much less than that of X. A decoder ANN designs $P(X|Z)$, which is not analytically tractable and approximated with a parametric variational distribution (inference model), with parameters obtained from another ANN (encoder). To simplify, the encoder pdf is usually Gaussian with a diagonal covariance matrix (as is common in HMMs).

Teacher-student models have also been explored for audio self-supervision where the trained model from a previous epoch acts as the teacher for the next epoch (Kumar and Ithapu, 2020). The name implies that a learning ANN (student) is guided by a training teacher to behave well. A teacher DNN trained with supervision using many samples can train a compressed network (student model) using a smaller labelled dataset.

5.13. Final notes for ANN ASR

To put E2E in perspective, we note that ASR emulates human speech perception. One model for the latter is *analysis-by-synthesis,* i.e., that human listeners decipher what they hear in terms of a latent representation of their auditory input. This is analogous to an autoencoder, which learns efficient data codings in a self-supervised manner.

A major issue for ANNs is their huge complexity. As full ANN searches need inordinate amounts of processing for large-vocabulary ASR (even when using fast *graphics processing units* (GPUs)), several ways can reduce computation, while minimizing corresponding losses in recognition accuracy. These include knowledge distillation, low-rank matrix factorization, sparsification, quantization, and model compression (Fasoli, 2021, Hinton et al., 2015). (Speech processing has many such compromises, for efficiency, e.g., vector quantization, first-order Markov models, assumption of diagonal covariance matrices, Gaussian pdf's, all-pole LPC.)

6. Language Models (LMs) for ASR

Early work in ASR used only acoustic information to select text output for input speech. Since textual redundancies in intended messages aid significantly in speech perception, in the 1980s it became obvious to use a LM for ASR. In some cases, speech may be a random sequence of words drawn from a small vocabulary (e.g., digits in telephone or credit-card numbers); such cases have few redundancies. More generally, however, speech follows syntactic, semantic, and other contextual rules. Given a history of prior words in an utterance, the number (P) of possible words that could ensue as the next word is far smaller than a general vocabulary size V (for digits, $V = 10$; in general text, $V =$ hundreds of thousands). P is the *perplexity* of a LM, which can reduce to far less than 100, vs. much larger numbers of words in languages.

A LM is a stochastic description of probabilities of sequences of words in training texts (Huang et al., 2017). *N-gram* models estimate the likelihood of each word given the context of the preceding *N-1* words, e.g., *bigram* models use statistics of word pairs and *trigrams* model word triplets. *Unigram* statistics are simply the a-priori likelihoods for each word, independent of context. Probabilities are estimated via analysis of much text, which automatically captures textual redundancies. As a LM grows exponentially with N, most applications use $N < 4$. Higher-order N-grams are nonetheless useful as context can extend over a wide range. However, such cases are often poorly modelled, even with massive training texts, owing to an insufficient number of examples. There are numerous modifications to this basic approach, including *back-off* methods, grammar constraints, models for specific conversation domains, and *out-of-vocabulary* detection (Egorova et al., 2021).

ASR may output a sequence of symbols representing phonemes, with associated likelihoods, e.g., a weighted list of boundary locations and phoneme candidates, in the form of a *lattice*. More commonly, word lattices use a post-processor *rescoring* algorithm to apply a LM. ANNs have been shown to outperform basic approaches to LMs (Huang et al., 2017). Large memory requirements typically block usage of LSTM for LM in low-resource devices (Sundermeyer et al.,). While most LMs use words as units, *word-piece* models may be more suitable than outputting characters (Chiu et al., 2017); i.e., parts of words are likely more useful to capture some redundancy in text.

A common approach in NLP is called BERT (*Bidirectional Encoder Representations from Transformers*), for learning contextual word representations. BERT randomly masks a fraction of data (Devlin et al., 2019) (main forms of regularization are dropout and label smoothing), whereas for acoustic data one randomly masks some frequency bands and/or segments in time (Wang et al., 2020). LSTM language models have improved over N-gram LMs in recent ASR; however, their use of long history often limits use to a second ASR pass for rescoring.

As interest in E2E has increased greatly, research has focused on ways to include external knowledge, i.e., LM fusion, e.g., shallow fusion with class-based Weighted Finite State Transducer (Miao et al., 2015), attention-based deep context, and trie-based deep biasing. These help for specific tasks, but do not generalize well.

This NLP section is intentionally brief, to allow more space for the signal processing focus of this paper. We also note that many S2S models implicitly learn an LM as part of their decoder.

7. Use of prosody in ASR

A major weakness for most ASR (both ANN and HMM) is use of f0 and timing in speech: variations are viewed as noise to be ignored, rather than useful features to exploit (Vergyri et al., 2003). As section 2.3 noted, speakers vary f0 and word durations as aids to communication, and listeners take advantage to help understanding (Vicsi and Szaszak, 2010, Shatzman and McQueen, 2006), yet this is very poorly exploited in ASR. Prosody can especially help in *speech understanding*, rather than simple ASR transcriptions; e.g., for some examples in section 2.3, the ASR phonemic output is insufficient for understanding.

The main reason for not using prosody in ASR concerns the major differences between (short) frame-based speech analysis methods and the (longer) domain of prosody in human speech. Durations, pitch, and amplitude vary widely for many linguistic reasons, at the broader level of words and phrases, and not at the local level of frames. Prosody relates to words, while speech analysis is done at 100 frames/s - a rate that tracks coarticulation and simplifies acoustic models. This rate is suitable for phonemic analysis, which has been the major focus for HMM state analysis and for ANNs, but not for wider phonological phenomena.

Consider adding f0 as an additional feature to the spectral vector for individual frames of speech (Liu et al., 2019, Magimai-Doss et al., 2003, Cambara et al., 2009). This helps for tone languages such as Chinese, where f0 has phonemic value. However, f0 should be used in context, not frame-by-frame, as f0's main use is in the broader context of words, whereas other spectral parameters vary directly with local VT positions.

For E2E ASR using waveforms as input, exploiting f0 can be difficult, as f0 is embedded quasi-periodically in the speech signal. The samples in voiced speech approximately repeat every pitch period. Given the large variability in pitch periods, it is unlikely to expect an ANN to optimize parameters for nodes spaced at the fixed intervals usually found in ANNs. Speech corresponding to the same text can readily have very different f0, and in ways more complicated than the variability of formants.

8. The Future for ASR

To improve ASR, many challenges remain, such as data efficiency, robustness and generalization. Much recent ASR progress has been driven by large increases in computer memory and power, as well as access to more data, both speech and text. There has been inadequate exploitation of the nature of the signal to recognize, i.e., speech. ANNs often have similar structures for many different input signals. The basic operation of MLPs derives from fundamental principles of neurons: simple weighted correlations and thresholding.

The use of gradient search may have some foundation in human learning and perception. However, ASR has taken much more advantage of computational power than exploiting physical principles of human communication. ANNs are opaque; it is very difficult to understand (or debug) how they make decisions. Some deeper levels of ANNs may organize around intuitive (e.g., phonetic) features (Nagamine et al., 2015, Nagamine et al., 2016, Nagamine and Mesgarani, 2017), but not in sufficient known detail to allow significant progress toward superior ASR. Another weakness of E2E is that attackers may infer private information by observing ANN output, as models are developed directly from data that are not always well anonymized.

ANNs will surely continue to be used for ASR, as their structure and training methods yield a suitable engineering approach, and accuracy has improved over earlier methods such as HMMs. Yet, they do not now provide a satisfactory scientific methodology. The principles of classification are well motivated, but the non-convex search space is so complex that use of a general loss function and regularization has been inadequate to approach human performance for ASR, especially for signals different from noise-free, native speech (Yin et al., 2015). There have been very few attempts to incorporate more speech-based structure into ANNs, and below we examine more direct ways that may improve both accuracy and efficiency in ASR.

8.1. Where to improve ASR

To see how ASR might improve further, let's examine current technology more closely. First, start with the "raw" speech data (e.g., 8-bit waveform samples at 8000 samples/s for typical telephone speech), which some E2E ASR now inputs directly, with no signal processing (Lam et al., 2021). Any specific choice to transform raw speech risks overlooking some information. However, as there is little evidence that ASR benefits from acoustic information beyond what is available in spectral amplitude, this risk is worth taking, given the weaknesses and complexity of ANN training.

Many speech production and perception studies have shown that the following are of essence: patterns of energy as functions of time and frequency (e.g., formants, as seen clearly even in wide-band spectrograms of distorted speech), f0, periodicity, and durations. For convenience, most ASR has ignored the latter three features (prosody), and concentrated on amplitude spectra, and has used versions of these (usually MFCCs or BFEs) that obscure what humans focus on. They are indirect measures of what listeners pay attention to, which ASR has used for convenience.

Most attempts to interpret internal operations of ANNs have found only tentative trends (Z et al., 2019). Given the power of ANNs to learn complex patterns, one could suggest to directly impose some structure that is known to be relevant for human speech production and perception, rather than have ML "discover" what researchers already know to be useful. Choices such as MFCCs clearly do discard useful information (e.g., f0), but MFCCs were selected primarily for computational simplicity, and it was never claimed that MFCCs retained all useful speech information.

ASR evaluations often report only average WER, and rarely discuss examples of errors. So we have few notions of why errors occur in ASR systems. As a result of this lack of knowledge and limited to techniques that avoid early decisions, ASR has been designed with general methodology, with powerful but broad approaches such as MLP, SGD, and regularization. Rather than focus on details of the signal to recognize, ways have been sought to generalize (e.g., via data augmentation) and smooth models.

To reduce the large amount of computation such models require, ASR sometimes tries approaches that appear to save calculations with little decrease in accuracy. However, lacking knowledge of how the system makes decisions may lead to choices that oversimplify, e.g., a current trend to use a low-frame rate (LFR), where the traditional 100 frames/s is lowered to 30/s via downsampling (Jiang et al., 2021). Speaking rates are approximately 12 phones/s, which may make LFR attractive. However, speech has rapid transients (e.g., stop bursts) that last as little as 1-2 frames, and these may not be handled well at LFR. (LFR stacks frames, which may preserve some information about rapid transients.) Lacking data on ASR errors, it is hard to judge the value and cost of LFR. However, a non-uniform downsampling may have benefit (see the next subsection).

8.2. Spectral primitives

Human speech production and perception both focus on frequency ranges where energy is strong. However, formant trackers are unreliable. Consider instead using simplified spectral-amplitude patterns (call them *primitives*) that display formant-like trajectories, over variable time sections of approximately 80 ms (e.g., typical phone durations). These should be more readily estimated from spectra than formants, yet directly relate to speech perception far better than MFCCs or BFEs. One might compare these to Gabor features used in image classification.

Explicit formant trackers have failed mostly because spectrograms often do not clearly display distinct formant tracks or because the VT is not always fully excited by glottal energy (Deng et al., 2006), e.g., when unvoiced, when noise is generated higher in the VT than at the glottis, or when the velum is lowered for nasal sounds. (Low energy in higher frequencies is another tracking problem.) Attempts to track unexcited formants have been futile (de Wet et al., 2004). Current research to track formants (Nagamine et al., 2015, Dissen et al., 2019) remains committed to getting a fixed set of resonances, while ASR work long ago abandoned this approach.

8.2.1. Examples of how to define primitives

A possible solution here is not to insist on tracking one formant per kHz (which 17-cm-long VTs have, on average). Indeed, listeners are not likely to do formant tracking; they instead pay attention to where and when energy is high. Wide-band spectrograms display dark patterns about 100 Hz wide (typical formant bandwidth) that come and go (increase and fade in amplitude), and rise and fall in frequency. It is these dynamic patterns that could be used as efficient inputs to ASR systems, including ANNs. Such spectral patterns would not be as simple as FBEs every 10 ms, as these primitives would have varying duration, but they would be far more relevant to (and efficient for) ASR than sets of MFCCs or FBEs every frame.

With adaptation such as VTLN (Section 4.2), primitives could be scaled in frequency to handle different length VTs (to accomplish speaker adaptation). Relative positions and combinations of these spectral trajectories are likely what cue the various phonemes of all languages. They have not yet been used in ASR owing to their inconvenience to frame-based systems, which have long dominated ASR. Modeling spectral peaks, they would be far more robust to noise and reverberation than most other measures; low-frequency energy peaks are the most robust aspects of speech spectra in harsh acoustic environments. In contrast, MFCCs and BFEs are much influenced by distortions.

Consider adding a bandwidth measure to each pattern. Most formants show as black bands on spectrograms of about 100 Hz in width, but these bands often appear to merge or split over time (due to VT motion). Rather than try to follow such changes, one might note an estimated bandwidth for each brief spectral pattern and use that as another parameter. In most cases (i.e., where the pattern effectively follows a formant), such a bandwidth parameter would have little effect on ASR. When formants appear to merge, the combined spectral peak would have a much larger bandwidth.

8.2.2. Evidence from example spectrograms

For examples where formant tracking fails, but primitives could succeed, the spectrograms in Fig. 3 show two utterances that are phonetically similar, and typify the sorts of acoustic features that ASR should distinguish. (In practical ASR, some of these distinctions would be resolved with language models, but we focus here on acoustic phenomena.) F3 is a hard feature to track, owing mostly to its weaker energy. For example, in "routes," F3 fades greatly (/u/ often has little energy above 1500 Hz). In "lags," F2 and F3 both have a ragged appearance, which suggests to use context and smooth spectral patterns, rather than estimate spectral patterns frame-by-frame. The phoneme /r/ is distinguished by F3 being below 2 kHz (for 17-cm VTs), and /r/ often appears as transitions to low values, rather than a steady strong formant. Several of the /r/ phones in Fig. 3 show little energy at the middle of /r/ except at low frequencies, and the cue to /r/ lies in transitions.

Another subtle and complex cue is for stop place of articulation; e.g., distinction between /b/ and /g/ ("rubber" - "rugger") shows as relatively small F2-F3 displacements adjacent to the stop closures. Nasals appear as abrupt spectral changes in both amplitude and spectra, but contain energy well above 400 Hz, unlike in voiced stops. Other sequences containing vowels, semivowels and liquids display smooth spectral patterns (e.g., /l/ and /r/ in Fig. 3). These all could be well and efficiently modelled by primitives.

Phone durations average 80 ms, but vary significantly (e.g., long in stressed vowels, as in the third vowels (200 ms) in Fig. 3; short in unstressed syllables, as in the first syllable of "repel"). The variability of durations is clear in Fig. 3, showing several cases where a formant can rise and then fall within either a brief or long portion of speech that might suggest as one or more sonorants. F2 rise+fall in the second syllable of "fuzzy" shows one vowel during 100 ms, and the same for F3 in "rugger." On the other hand, the long vowels may have only a smooth rise (e.g., F2 in "lags"), or have relatively steady formants with brief transitions at either end of the vowel (as for F1 in "rats").

8.2.3. *Definition of primitives*

Formally, one could partition continuous, strong spectral-peak speech sequences longer than 80 ms into shorter portions (aiming for an average of 80 ms), each thus being labelled as a primitive, with either a (roughly linear) rise or fall. Each of these should show approximately 100-Hz-wide spectral bands (bands in Fig. 3 appear wider, owing to smoothing). What would appear to be a single band wider than this would be labelled as two primitives, e.g., F2+F3 in the second syllable in "rubber" and in "rugger." When the pattern shows both rise and fall in frequency in a short portion of time, we would have two primitives, e.g., F2 in the second syllable of "fuzzy"; i.e., a phone may have two primitives. We do not suggest phone segmentation.

Thus, each speech section of approximately 80 ms with energy exceeding a certain threshold (determined by a running average of the speech) would have 2-4 primitives (e.g., 2 for phonemes such as /u/, which have very weak energy above 2 kHz, and 3-4 for sonorants with well-spaced peaks). Each primitive would roughly correspond to a portion of a formant rising, falling or flat. Segment boundaries would occur where one or more primitives change slope significantly at the same time. Segments could be as short as 20 ms, corresponding to very brief vowels, or as long as 200 ms (e.g., in cases where speakers elongate a syllable). Major energy changes could cue more obvious boundaries. In the next subsection, we see that such an asynchronous approach to speech analysis for phones also could work well more broadly in time.

8.2.4. *Integrating prosody into primitives*

The widely accepted practice of processing speech every 10-ms frame, in almost all of speech processing (coding, synthesis, recognition, enhancement), works well in most cases, but is less helpful to most ASR methodology. Both HMMs and ANNs accept variable time ranges of input data to process, and 10-ms frames are convenient choices for coarticulation and typical speech rates of 12 phones/s. Speakers and listeners are not constrained by such synchronous needs as are ASR processors. Speakers plan their speech at levels of phonemes, words and sentences, not 100 frames/s.

Long-range planning is used for prosody. While f0 can rise and fall within a single vowel, it more typically cues information to listeners over words, not frames. Individual values of f0 (per frame) are only useful in the context of their trends over multiple frames (i.e., words). This is somewhat similar to coarticulation (an ASR problem) for speech spectra, where HMMs store static MFCCs along with delta and delta-delta versions (i.e., position, velocity and acceleration). Because first-order Markov models have very short look-back history, they store longer-range information within each frame's data. However, such a mechanism is awkward and inefficient, tripling storage and computation, for only small gains in accuracy. In addition, the longer range that HMMs allow is only about nine frames, e.g., only one phone's duration, whereas the domain of prosody goes much further.

Nonetheless, prosody should be able to be included in ANN ASR. Data need not necessarily be fed into an ANN in 10-ms frame chunks (Ravanelli et al., 2020). One can use features of f0 rises and falls over longer durations in the range of phones (e.g., primitives, as discussed above), and include simple f0 parameters representative of such patterns as inputs to ANNs. These are relevant for how speakers signal stress and syntactic structure.

Something similar would be useful in ASR for durations. However, relevant durations would not be as simple to calculate as are f0 values (for which the literature has an immense number of estimation algorithms (Wang et al., 2017)). ASR has always viewed segmentation of speech signals into useful linguistic units (words, phones) as a huge challenge; this is implicit with the timing issues discussed earlier. Modern ASR avoids that issue, e.g., by conjoining short models in HMMs and delaying all decisions till all data are examined. Both HMMs and RNNs treat timing as an undesired variable in speech to overcome (it is considered as noise, not signal). Since humans exploit duration in speech communication, ASR should find a way to do so as well. Just because using prosody is inconvenient to modern ASR technology choices is little excuse to ignore it. As with f0, however, a frame-based approach is not best. Various approaches toward segment-based ASR have been tried (Abdel-Hamid et al., 2013), but usually in hybrid systems that often increase complexity.

8.2.5. *Using primitives in ANNs*

More simply and directly, what could work is to group frames that are similar spectrally into phone-like segments, and use their numbers of frames (i.e., segment durations) as parameters to feed an ANN. These units need not correspond to actual phones, but rather brief segments of speech with uniform spectral movement (e.g., primitives). Listeners pay attention to how much time is spent on individual syllables; any segmentation "errors" here (unlike in earlier expert-system ASR) would not necessarily be detrimental to ASR decisions with ANNs. These unit durations would be only additional (and useful) parameters to employ in an ANN's input. Segment boundaries (very roughly approximating phones, or parts of phones) could be determined by using a spectral derivative measure, aiming for units of 80-ms duration approximately.

Thus the suggestion here is to feed ANN ASR with parameters that are flexible both in nature and in timing. For convenience, the spacing between unit boundaries could be roughly 80 ms (typical phone durations), but would vary as needed (i.e., determined empirically by the patterns themselves). It may be good to divide longer spectral patterns into shorter patterns of durations closer to phone units, to synchronize with f0 and duration parameters. Thus, feed an ANN roughly every 80 ms with a set of, typically, 3-4 *pseudo-formants* (i.e., the endpoint frequencies of primitives) as well as start and end f0 values (or, instead, mean and change values), and the actual unit duration (which would deviate from an 80-ms average). Like other measures such as MFCCs, these data should be normalized, as in Section 4.2, for at least their means (to handle different VT lengths, vocal cord sizes, and speaking rates).

These spectral and prosody primitives could be determined themselves by an ANN, and then fed to another ANN for further speech classification. This is different from just using one ANN; the first ANN would be dedicated to efficient spectral and prosody representation, as such modeling (e.g., primitives) is not likely to be found well with a single, universal training (i.e., SGD with CE loss). Such features require more complex, structured search.

Table 2
Phonemes and some of their features.

Phonemes	Articulation	Spectrum	Excitation
i, I, e	High front vowel	Low F1, high F2	Periodic
E, ae	Low front vowel	High F1, high F2	Periodic
A, aw, o	Low back vowel	High F1, high F2	Periodic
U, u	High back vowel	Low F1, low F2	Periodic
l, r	Liquid	Low F1 and F2	Periodic
m, n, ng	Nasal	Steady F1, F2	Periodic
j, w	Glide	Low F1	Transient
f, s, sh	Fricative	Highpass	Aperiodic noise
v, z, zh	Fricative	Highpass	Periodic noise
p, t, k	Stop	silence+burst	Aperiodic transient
b, d, g	Stop	silence+burst	Periodic transient

Some ideas in this section resemble *segmental models* used to improve HMMs (Gillick et al., 2012). Such earlier approaches did not advance much owing to three interrelated problems: (1) Handling temporal variability with longer units. (2) Avoiding error propagation in models that make hard decisions at low levels. (3) Large search spaces with multiple segmentation hypotheses.

Unlike lattice approaches with multiple hypotheses, a single segmentation is advocated, estimating pseudo-phone boundaries based on the best available evidence. Missing, extra or misplaced boundaries (i.e., those not agreeing with expert linguists' manual labels) would not be crucial, given the ability of ML to handle great variability; boundaries need not correspond to actual phones. No hard decisions are needed.

8.3. *Efficiency*

Many commercial applications for ASR (e.g., Apple Siri, Microsoft Cortana, Amazon Echo) require major remote processing, owing to large models and much search computation (Xiong et al., 2018, Sainath, 2021, Zhang et al., 2018). Nonetheless, the size and power limitations of portable devices (*edge computing*) mean that efficient processing is still important (e.g., Google Assistant). The suggestions in this section could make ASR both more accurate and far more efficient than using as inputs either raw speech samples (as in some E2E ASR) or frames (as in LSTMs); e.g., 12 input sets of about 12 parameters/s, vs. 8000/s for E2E or 100 sets of 13 (or 39) MFCCs per second for LSTMs. An input data rate of 144 parameters/s is much more in line with what the human brain can handle for speech decisions. The actual information rate of speech, in terms of phonetic decisions, is far lower than the data flow used in modern ASR.

We have gotten used to powerful computers and cloud computation in modern ASR methods, but should always strive for more efficient systems, especially if we might wish to have ASR in edge computing, rather than remote servers (Wong et al., 2008). Recent ANN ASR systems employ many millions of parameters, take days to train on multiple GPUs, and employ hundreds of spectral features per frame (Tuske et al., 2020) (Table 8). To model vocabularies of a few thousand words, ANNs often use many millions of parameters, i.e., many thousands of parameters per word; this is not efficient. In addition, as the depth of DNNs increases, training by SGD is more difficult due to the high non-convexity of error surfaces.

Spectral energy patterns (pseudo-formants) would be each parameterized by their durations and end-points. Using inputs to ANNs that are not strictly synchronous to either sampling rate (as in current E2E systems) or to frame rate (as in LSTMs) is not a problem. ANNs have great flexibility to handle a large range of inputs.

9. Conclusion

The essential structure of an ANN has been (and remains) the basic MLP, and the standard training way remains SGD search. There have been numerous improvements to these basic methods, taking advantage of more computation power and massive increases in data, but increasingly complex ANN models remain opaque and difficult to decipher.

When one knows, generally, which features are important to classify a set of signals such as speech, why insist on having relatively brute-force training re-learn the features? To avoid discarding possibly useful information? We are much more likely to benefit by employing directly-useful spectral-amplitude patterns as input to an ANN. These would be resistant to distortion, and far more efficient than using many dozens of log energies and their derivatives.

ANNs are powerful tools, but can improve with more structure. Earlier ASR research rejected expert-system (ES) methodology, owing to the failure of ES to handle the large variability of speech and the lack of error correction feedback. Feature extractors (f0

Table 3
Summary of perception features

Phonemes	Manner feature	Primary feature	Secondary feature
Vowels	Strong formants	F1 frequency	F2 frequency
Fricatives	Random noise	Highpass cutoff	Voicing
Nasals	Steady weaker formants	Abrupt onset and offset	
Stops	silence+burst	F2-F3 transitions	Burst amplitude, frequency

Table 4
Summary of speech analysis methods.

Method	Properties	Advantages	Flaws
Fourier transform	Discards phase	Direct spectral measure	Many parameters
Linear prediction	all-pole model	10 coefficients	Narrow bandwidths
MFCC	Triangular mel filters	13 coefficients	No intuitive interpretation
Formants	Resonance frequencies	Few coefficients	Unreliable estimation
Band-frequency energies	Direct from FT	Intuitive interpretation	Coarse model

Table 5
Brief summary of major approaches used for ASR.

Method	Properties	Advantages	Flaws
HMM	State model with transitions	Handles both spectral and timing issues	Markov assumption; geometric duration model
ANN	Neural-based nonlinear data mapping	Great power; many architectures	Extremely complex network
DTW	Nonlinear sequence alignment	Simple	Expensive; slow
Expert system	If-then decisions	Intuitive; easy to modify	Poor error rates

Table 6
Frequently used English databases for ASR.

ASR task	Style	Content	Vocabulary size (words)	Comments
Aurora	read	digit sequences	10	English digits easier to recognize than letters
Resource Management	read	sentences	1000	Early official DARPA database for ASR
Wall Street Journal	read	actual news articles	5000-20000	Popular standard database
Broadcast News	read	actual news reports	Approximately 60000	Read by professionals on public media
Air Travel Information System	question and answer	user requests	Approximately 20000	Wizard-of-Oz requests about flights
Switchboard	spontaneous	conversations between strangers	Tens of thousands	Talk between strangers is more formal
Call Home	spontaneous	conversations between friends	Tens of thousands	Talk among friends can be very informal
Meeting	spontaneous	conversations among colleagues	Tens of thousands	European-funded AMI project
Librispeech	Read	Audiobooks	100000	Audio for entertainment

Table 7
Brief summary of ANN architectures.

Type	Properties	Advantages	Weaknesses
Feedforward FFNN	Fully connected	Simple structure	Many parameters
Convolutional CNN	Kernel filters	Simplifies locally	Local effects
Recurrent RNN	Input, output, forget gates	Handles distant effects	Vanishing gradient
Attention	Emphasis via correlation	Handles non-uniform distribution of information	Use of simple correlation as criteria
Transformer	Encoder/decoder with attention	Merges ideas from other methods	Alignment

Table 8
Comparison of ASR methods

Method	Complexity (parameters)	Advantages	Training	Flaws
Basic ANN	Millions	Great flexibility	Very data hungry	Very opaque
HMM	Hundreds of thousands	Handles time and spectra variations	Very data hungry	First-order Markov assumption
Structured ANN	Thousands	More efficient	Less needed	Uses prior assumptions

detectors and pseudo-formant trackers) have flaws, but may yield useful, efficient measures. With suitable methods, such features could be useful in the future. Table 2, Table 5, Table 7.

Declaration of Competing Interest

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

Data availability

No data was used for the research described in the article.

Acknowledgements

This work was funded by NSERC (Canada) (Grant No. 142610). I wish to thank Tiago H. Falk and John Hansen for their comments to help improve the paper.

References

Spille, C., Kollmeier, B., Meyer, B.T., 2018. Comparing human and automatic speech recognition in simple and complex acoustic scenes. Comput. Speech Lang. 52, 123–140.

Rabiner, L., Juang, B.-H., 1993. Fundamentals of Speech Recognition. Prentice-Hall, NJ.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–444.

Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., Kingsbury, B., 2012. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. 29, 82–97.

Bishop, C., 2006. Pattern Recognition and Machine Learning. Springer, Berlin).

Kamath, U., Liu, J., Whitaker, J., 2019. Deep Learning for NLP and Speech Recognition. Springer.

Li, J., Deng, L., Haeb-Umbach, R., Gong, Y., 2015. Robust Automatic Speech Recognition: A Bridge to Practical Applications. Academic Press.

O'Shaughnessy, D., 2019. Recognition and processing of speech signals using neural networks,". Circuits Systems Signal Process. 38, 3454–3481.

Shao, Y., Wang, Y., Povey, D., Khudanpur, S., 2020. Py Chain: A fully parallelized PyTorch implementation of LF-MMI for end-to-end ASR. In: Proc. Interspeech.

Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., et al., 2011. The Kaldi speech recognition toolkit. In: ASRU.

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., 2016. TensorFlow: a system for Large-Scale machine learning. In: Proceedings of the USENIX symposium on operating systems design and implementation (OSDI 16), pp. 265–283.

Baevski, A., Hsu, W.N., Xu, Q., Babu, A., Gu, J. and Auli, M., "Data2vec: A general framework for self-supervised learning in speech, vision and language." arXiv preprint arXiv:2202.03555, 2022.

Wong, J.H.M., Gaur, Y., Zhao, R., Lu, L., Sun, E., Li, J., Gong, Y., 2020. Combination of end-to-end and hybrid models for speech recognition. In: Proc. Interspeech.

Tuske, Z., Saon, G., Audhkhasi, K., Kingsbury, B., 2020. Single headed attention based sequence-to-sequence model for state-of-the-art results on Switchboard. In: Proc. Interspeech.

Saon, G., et al., 2017. English conversational telephone speech recognition by humans and machines,". In: Proc. Interspeech, pp. 132–136.

Picheny, M., Tuske, Z., Kingsbury, B., Audhkhasi, K., Cui, X., Saon, G., 2019. Challenging the boundaries of speech recognition: the MALACH corpus. In: Proc. Interspeech.

Zhang, Z., Geiger, J., Pohjalainen, J., Mousa, A., Schuller, B., 2018. Deep learning for environmentally robust speech recognition: an overview of recent developments. ACM Trans. Intell. Syst. Technol.

Yujian, L., Bo, L., 2007. A normalized Levenshtein distance metric. IEEE Trans. Pattern Anal. Mach. Intell. 29 (6), 1091–1095.

Jiang, H., 2005. Confidence measures for speech recognition: a survey. Speech Commun. 45 (4), 455–470.

Sun, L., Du, J., Gao, T., Fang, Y., Ma, F., Lee, C-H., 2019. A speaker-dependent approach to separation of far-field multi-talker microphone array speech for front-end processing in the CHiME-5 challenge. IEEE J. Sel. Top. Signal Process. 13, 827–840.

Lippmann, R., 1987. An introduction to computing with neural nets. IEEE ASSP Mag. 4 (2).

Mitra, V., Franco, H., Bartels, C., van Hout, J., Graciarena, M., Vergyri, D., 2017. Speech recognition in unseen and noisy channel conditions. In: , pp. 5215–5219.

González Hautamäki, R., Kinnunen, T., Hautamäki, V., Laukkanen, A-M., 2015. Automatic versus human speaker verification: the case of voice mimicry. Speech Commun. 13–31.

Lohrenz, T., Li, Z., Fingscheidt, T., 2021. Multi-encoder learning and stream fusion for transformer-based end-to-end automatic speech recognition. In: Proc. Interspeech, pp. 2846–2850.

Avila, A., Monteiro, J., O'Shaughnessy, D., Falk, T., 2017. Speech emotion recognition on mobile devices based on modulation spectral feature pooling and deep neural networks. IEEE ISSPIT.

O'Shaughnessy, D., 2000. Speech Communication: Human and Machine. IEEE Press, New York.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press, Cambridge.

Sotelo, J., Mehri, S., Kumar, K., Santosy, J.F., Kastner, K., Courvillez, A., Bengio, Y., 2017. Char2wav: end-to-end speech synthesis. In: Proc. ICLR.

S.O. Arik et al, "Deep Voice: Real-time Neural Text-to-Speech," ArXiv, 2017.

Ping, W., Peng, K., Gibiansky, A., Arık, S.O., Kannan, A., Narang, S., 2018. Deep voice 3: scaling text-to-speech with convolutional sequence learning. In: Proc. ICLR.

de Wet, F., Weber, K., Boves, L., Cranen, B., Bengio, S., Bourlard, H., 2004. Evaluation of formant-like features on an automatic vowel classification task, full text links. J. Acoust Soc. Am. 116 (3), 1781–1792.

Backstrom, T., 2017. Speech Coding: With Code-Excited Linear Prediction. Springer, Berlin.

Yang, X., Shen, X., Li, W., Yang, Y., 2014. How listeners weight acoustic cues to intonational phrase boundaries. PlosOne.

Shatzman, K., McQueen, J., 2006. Segment duration as a cue to word boundaries in spoken-word recognition. Percept. Psychophys. 68 (1), 1–16.

Goldwater, S., Jurafsky, D., Manning, C., 2010. Which words are hard to recognize? Prosodic, lexical, and disfluency factors that increase speech recognition error rates. Speech Commun. 52, 181–200.

Kaur, J., Singh, A., Kadyan, V., 2021. Automatic speech recognition system for tonal languages: state-of-the-art survey. Arch. Comput. Meth. Eng. 28 (3), 1039–1068.

O'Shaughnessy, D., 1979. Linguistic features in fundamental frequency patterns. J. Phonetics 7 (2), 119–145.

Johnson, K., 2020. The ΔF method of vocal tract length normalization for vowels. Lab. Phonol. 11 (1), 1–16, 10.

Bell, P., Fainberg, J., Klejch, O., Li, J., Renals, S., Swietojanski, P., 2021. Adaptation algorithms for neural network-based speech recognition: an overview. IEEE Open J. Signal Process. 2, 33–66.

Gales, M., 1998. Maximum likelihood linear transformations for HMM-based speech recognition. Comput. Speech Lang. 12 (2), 75–98.

Jurafsky, D. and Martin, J.H., Speech and Language Processing, 2000.

Makhoul, J., El-Jaroudi, A., Schwartz, R., 1989. Formation of disconnected decision regions with a single hidden layer. In: Proceeding of the International Joint Conference on Neural Networks.

Heck, M., Sakti, S., Nakamura, S., 2018. Learning supervised feature transformations on zero resources for improved acoustic unit discovery. IEICE Trans. Inf. Syst. 205–213.

Hermann, E., Kamper, H., Goldwater, S., 2020. Multilingual and unsupervised subword modeling for zero-resource languages. Comp. Speech Lang.

Kutner, M., Neter, J., Nachtsheim, C., Wasserman, W., 2004. Applied Linear Statistical Models. McGraw-Hill, New York.

Sun, W., Su, F., Wang, L., 2018. Improving deep neural networks with multi-layer Maxout networks and a novel initialization method. Neurocomputing 278, 34–40.

M. Ravanelli and Y. Bengio,"Speech and speaker recognition from raw waveform with SincNet," arXiv:1812.05920, 2018.

Davis, S.B., Mermelstein, P., 1980. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. IEEE Trans. ASSP 28, 357–366.
Dutta, Anirban, Ashishkumar, G, Rao, Ch V-R., 2019. Auditory inspired acoustic model for hybrid ASR system using gammatone based gabor filters. In: Proceeding of the IEEE International Symposium on Signal Processing and Information Technology (ISSPIT).
Zahorian, S., Jagharghi, 1993. Spectral-shape features versus formants as acoustic correlates for vowels. J. Acoust. Soc. Am. 94 (4), 1966–1982. Oct.
Nagamine, T., Seltzer, M.L., Mesgarani, N., 2015. Exploring how deep neural networks form phonemic categories. In: Proc. Interspeech, pp. 1912–1916.
Dissen, Y., Goldberger, J., Keshet, J., 2019. Formant estimation and tracking: a deep learning approach. J. Acoust. Soc. Am. 145 (2), 642–653.
Wang, D., Yu, C., Hansen, J.H.L., 2017. Robust harmonic features for classification-based pitch estimation. IEEE/ACM Trans. Audio Speech Lang Process. 25 (5), 952–964.
Jiang, D., Zhang, C., Woodland, P.C., 2021. Variable frame rate acoustic models using minimum error reinforcement learning. In: Proc. Interspeech, pp. 2601–2605.
Baevski, A., Zhou, Y., Mohamed, A., Auli, M., 2020. Wav2vec 2.0: a framework for self-supervised learning of speech representations. Adv. Neural Inf. Process. 33.
Hermansky, H., 1990. Perceptual linear predictive (PLP) analysis of speech. J. Acoust. Soc. Am. 87 (4), 1738–1752.
Gillick, D., Wegmann, S., Gillick, L., 2012. Discriminative training for speech recognition is compensating for statistical dependence in the HMM framework. ICASSP.
Furui, S., 1981. Cepstral analysis technique for automatic speaker verification. IEEE Trans. Acoust. Speech Signal Process. 29 (2), 254–272. April.
Toledano, D., Fernandez-Gallego, M., Lozano-Dies, A., 2018. Multi-resolution speech analysis for automatic speech recognition using deep neural networks: Experiments on TIMIT. PlosOne.
Luscher, C., et al., 2019. RWTH ASR systems for LibriSpeech: hybrid vs attention. In: Proc. Interspeech, pp. 231–235.
Bourlard, H., Morgan, N., 2012. Connectionist speech recognition: a hybrid approach. Kluwer, 1994.
Fosler-Lussier, E., He, Y., Jyothi, P., Prabhavalkar, R., 2013. Conditional random fields in speech, audio, and language processing. Proc. IEEE 101, 1054–1075.
Bai, L., Weber, P., Jancovic, P., Russell, M., 2018. Exploring how phone classification neural networks learn phonetic information by visualizing and interpreting bottleneck features. In: Proc. Interspeech, pp. 1472–1476.
ten Bosch, L., Boves, L., 2018. Information encoding by deep neural networks: what can we learn?". In: Proc. Interspeech, pp. 1457–1461.
Xiong, W., Wu, L., Alleva, F., Droppo, J., Huang, X., Stolcke, A., 2018. The microsoft 2017 conversational speech recognition system. ICASSP.
Qian, Y., Bi, M., Tan, T., Yu, K., 2016. Very deep convolutional neural networks for noise robust speech recognition. IEEE/ACM Trans. Audio Speech Lang. Process. 24, 2263–2276.
Ghahremani, P., Hadian, H., Lv, H., Lovey, D., Khudanpur, S., 2018. Acoustic modelling from frequency domain representations of speech. In: Proc. Interspeech, pp. 1596–1600.
Hubara, I., Courbariaux, M., Soudry, D., El-Yaniv, R., Bengio, Y., 2017. Quantized neural networks: training neural networks with low precision weights and activations. J. Mach. Learn. Res. 18 (187), 1–30.
Kingma, D.P., Ba, J.Lei, 2014. Adam: a method for stochastic optimization. In: Proceedings of the International Conference on Learning Representations (ICLR).
I. Sutskever, J. Martens, G. Dahl, G. Hinton, "On the importance of initialization and momentum in deep learning," in Proceedings of the International Conference on Machine Learning, PMLR, vol. 28(3):1139-1147, 2013.
Sun, S., Yeh, C.-F., Ostendorf, M., M.-Y.Hwang, L.Xie, 2018. Training augmentation with adversarial examples for robust speech recognition. In: Proc. Interspeech, pp. 2404–2408.
Tuske, Z., Audhkhasi, K., Saon, G., 2019. Advancing sequence-to-sequence based speech recognition. In: Proc. Intersp., pp. 3780–3784.
Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1929–1958.
Ravanelli, M., Zhong, J., Pascual, S., Swietojanski, P., Monteiro, J., Trmal, J., Bengio, Y., 2020. Multi-task self-supervised learning for Robust Speech Recognition. ICASSP.
L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, R. Fergus, "Regularization of Neural Networks using DropConnect," Int. Conf. on Machine Learning, PMLR, 28(3): 1058-1066, 2013.
D. Krueger, T. Maharaj, J. Kramár, M. Pezeshki, N. Ballas, N. R. Ke, A. Goyal, Y. Bengio, A. Courville, C. Pal, "Zoneout: Regularizing RNNs by Randomly Preserving Hidden Activations," ArXiv 2017.
Kanda, N, Takeda, R, Obuchi, Y, 2013. Elastic spectral distortion for low resource speech recognition with deep neural networks. ASRU.
Tjandra, A., Sakti, S., Nakamura, S., 2018. Sequence-to-sequence ASR optimization via reinforcement learning. ICASSP.
Park, D.S., Chan, W., Zhang, Y., Chiu, C.C., Zoph, B., Cubuk, E.D., Le, Q.V., 2019. SpecAugment: a simple data augmentation method for automatic speech recognition. In: Proc. Interspeech, pp. 2613–2617.
Tang, J., Song, Y., Dai, L., McLoughlin, I., 2018. Acoustic Modeling with densely connected residual network for multichannel speech recognition. In: Proc. Interspeech.
LeCun, Y., et al., 1998. Efficient BackProp. In: Orr, Müller (Eds.), Neural Networks: Tricks of the Trade. Springer.
Bell, P., Fainberg, J., Klejch, O., Li, J., Renals, S., Swietojanski, P., 2020. Adaptation algorithms for speech recognition: an overview. EEE Open J. Signal Process.
Saon, G., Soltau, H., Nahamoo, D., Picheny, M., 2013. Speaker adaptation of neural network acoustic models using i-vectors. IEEE Workshop on ASRU 55–59.
Kim, T., Song, I., Bengio, Y., 2017. Dynamic layer normalization for adaptive neural acoustic modeling in speech recognition. In: Proc. Interspeech.
Kim, J., Wang, J., Kim, S., Lee, Y., 2020. Evolved speech-transformer: applying neural architecture search to end-to-end automatic speech recognition. In: Proc. Interspeech.
Bengio, Y., Léonard, N., Courville, A., 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. CoRR.
Peddinti, V., Povey, D., Khudanpur, S., 2015. A time-delay neural network architecture for efficient modeling of long temporal contexts. In: Proc. Interspeech.
Graves, A., Mohamed, A., Hinton, G.Geoffrey, 2013. Speech recognition with deep recurrent neural networks. ICASSP 6645–6649.
Sainath, T.N., Li, B., 2016. Modeling time-frequency patterns with LSTM vs. convolutional architectures for LVCSR tasks. In: Proc. Interspeech, pp. 813–817.
Cho, K., van Merrienboer, B., C.Gulcehre, F.Bougares, Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the Empirical Methods of Natural Language Processing.
Chan, W., et al., 2016. Listen, attend and spell: a neural network for large vocabulary conversational speech recognition. In: Proc. ICASSP, pp. 4960–4964.
A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, "Attention is all you need," Advances in neural information processing systems, 5998-6008, 2017.
Pham, N-Q., Nguyen, T-S., Niehues, J., Müller, Markus, Waibel, A., 2019. Very deep self-attention networks for end-to-end speech recognition. In: Proc. Interspeech.
Gulati, A., et al., 2020. Conformer: Convolution-augmented transformer for speech recognition. In: Proc. Interspeech.
Sperber, M., Niehues, J., Neubig, G., Stüker, S., Waibel, A., 2018. Self-attentional acoustic models. In: Proc. Interspeech.
Inaguma, H., Gaur, Y., Lu, L., Li, J., Gong, Y., 2020. Minimum latency training strategies for streaming sequence-to-sequence ASR. In: Proc. ICASSP.
I. Sutskever, O. Vinyals, Q.V. Le, "Sequence to sequence learning with neural networks," in Advances in Neural Info. Process. Cyst., 3104–3112, 2014.
A. Graves, S. Fernández, F. Gomez, J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in the Proc. Int. Conf. on Machine Learning, 2006.
Li, J., Zhao, R., Hu, H., Gong, Y., 2019. Improving RNN transducer modeling for end-to-end speech recognition. IEEE ASRU, pp. 114–121.
Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A. and Ng, A.Y., "Deep speech: scaling up end-to-end speech recognition." arXiv preprint arXiv:1412.5567, 2014.
Miao, Y., Gowayyed, M., Metze, F., 2015. EESEN: End-to-end speech recognition using deep RNN models and WFST-based decoding. In: Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), pp. 167–174.
H. Soltau, H. Liao, and H. Sak, "Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition." arXiv preprint arXiv:1610.09975, 2016.

Audhkhasi, K., Ramabhadran, B., Saon, G., Picheny, M., Nahamoo, D., 2017. Direct acoustics-to-word models for English conversational speech recognition. In: Proc. Interspeech, pp. 959–963.
Chiu, C.-C., et al., 2017. State-of-the-art speech recognition with sequence-to-sequence models. In: Proc. ICASSP.
Zhang, Y., Pezeshki, M., Brakel, P., Zhang, S., Laurent, C., Bengio, Y., Courville, A., 2016. Towards end-to-end speech recognition with deep convolutional neural networks. In: Proc. Interspeech, pp. 410–414.
R. Collobert, C. Puhrsch, G. Synnaeve, "Wav2Letter: an end-to-end ConvNet-based speech recognition system," arXiv:1609.03193, 2016.
Prabhavalkar, R., Sainath, T.N., Li, B., Rao, K., Jaitly, N., 2017. An analysis of "attention" in sequence-to-sequence models. In: Proc. Interspeech, pp. 3702–3706.
Chiu, CC, Raffel, C, 2018. Monotonic chunkwise attention. In: Proc. Int. Conf. on Learn. Representations.
Chorowski, J.K., Bahdanau, D., Serdyuk, D., Cho, K., Bengio, Y., 2015. Attention-based models for speech recognition. In: Proc. NIPS, pp. 1–16.
Nussbaum-Thom, M., Cui, J., Ramabhadran, B., Goel, V., 2016. Acoustic modeling using bidirectional gated recurrent convolutional units,". In: Proc. Interspeech, pp. 390–394.
Sainath, T.N., Pang, R., Rybach, D., He, Y., Prabhavalkar, R., Li, W., Visontai, M., Liang, Q., Stroh-man, T., Wu, Y., McGraw, I., Chiu, C-C., 2019. Two-pass end-to-end speech recognition. In: Proc. Interspeech.
Karita, S., et al., 2019. A comparative study on Transformer vs RNN in speech applications. ASRU.
Thual, A., Dancette, C., Karadayi, J., Benjumea, J, Dupoux, E., 2018. A k-nearest neighbours approach to unsupervised spoken term discovery. In: Proceedings of the IEEE Spoken Language Technology Workshop, pp. 491–497.
Bengio, Y., Courville, A.C., Vincent, P., 2013. Representation learning: a review and new perspectives. IEEE Tr. Patt. An. Mach. Intell. 35, 1798–1828.
Park, A., Glass, J., 2008. Unsupervised pattern discovery in speech. IEEE Tr. ASLP 16 (1).
Sadhu, S., He, D., Huang, C-W., Mallidi, S.H., M.Wu, A.Rastrow, Stolcke, A., Droppo, J., 2021. wav2vec-C: a self-supervised model for speech representation learning. In: Proc. Interspeech, pp. 711–715.
Chung, Y-A., Hsu, W-N., Tang, H., Glass, J., 2019. An unsupervised autoregressive model for speech representation learning. In: Proc. Interspeech.
Oord, A., Li, Y., and Vinyals, O. "Representation learning with contrastive predictive coding," arXiv:1807.03748, 2018.
Kamper, H., Jansen, A., Goldwater, S., 2015. Unsupervised word segmentation and lexicon discovery using acoustic word embeddings. In: Proc. ICASSP.
Last, P-J., Engelbrecht, H.A., Kamper, H., 2020. Unsupervised feature learning for speech using correspondence and siamese networks. IEEE Signal Process Lett. 27.
Bie, X., Girin, L., Leglaive, S., Hueber, T., Alameda-Pineda, X., 2021. A benchmark of dynamical variational autoencoders applied to speech spectrogram modeling. In: Proc. Interspeech, pp. 46–50.
Kumar, A, Ithapu, VK, 2020. A sequential self teaching approach for improving generalization in sound event recognition. In: Proceedings of the International Conference on Machine Learning (ICML).
Fasoli, A., et al., 2021. 4-bit quantization of LSTM-based speech recognition models,". In: Proc. Interspeech, pp. 2586–2590.
G Hinton, O Vinyals, J Dean, "Distilling the knowledge in a neural network," arXiv, 2015.
Huang, Y., Sethy, A., Ramabhadran, B., 2017. Fast neural network language model lookups at N-gram speed. In: Proc. Interspeech, pp. 274–278.
Egorova, E., Vydana, H.K., Burget, L., Černocký, J., 2021. Out-of-vocabulary words detection with attention and CTC alignments in an end-to-end ASR system. In: Proc. Interspeech, pp. 2901–2905.
Devlin, J., Chang, M-W., Lee, K., Toutanova, K., 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proc. NAACL-HLT.
Wang, W., Tang, Q., Livescu, K., 2020. Unsupervised pre-training of bidirectional speech encoders via masked reconstruction. In: Proc. ICASSP.
Vergyri, D., Stolcke, A., Gadde, V.R.R., Ferrer, L., Shriberg, E., 2003. Prosodic knowledge sources for automatic speech recognition. In: Proc. ICASSP.
Vicsi, K., Szaszak, G., 2010. Using prosody to improve automatic speech recognition. Speech Commun. 52 (5), 413–426.
Shatzman, K., McQueen, J., 2006. Segment duration as a cue to word boundaries in spoken-word recognition. Percep. Psychophys. 68 (1), 1–16.
Liu, S., Hu, S., Liu, X., Meng, H., 2019. On the use of pitch features for disordered speech recognition. In: Proc. Interspeech.
Magimai-Doss, M., Stephenson, T.A., Bourlard, H., 2003. Using pitch frequency information in speech recognition. In: Proc. Eurospeech.
G. Cambara, J. Luque, and M. Farrus, "Convolutional speech recognition with pitch and voice quality features," ArXiv, 2009.01309, 2020.
Nagamine, T., Seltzer, M.L., Mesgarani, N., 2016. On the role of nonlinear transformations in deep neural network acoustic models. In: Proc. Interspeech, pp. 803–807.
Nagamine, T., Mesgarani, N., 2017. Understanding the representation and computation of multilayer perceptrons: a case study in speech recognition. In: Proceedings of the International Conf. on Machine Learning, 70.
Yin, S., Liu, C., Zhang, Z., Lin, Y., Wang, D., Tejedor, J., Zheng, T., Li, Y., 2015. Noisy training for deep neural networks in speech recognition. EURASIP J. Audio, Sp., Music Process.
Lam, M.W.Y., Wang, J., Weng, C., Su, D., Yu, D., 2021. Raw waveform encoder with multi-scale globally attentive locally recurrent networks for end-to-end speech recognition. In: Proc. Interspeech, pp. 316–320.
Z, Tüske, Schlüter, R., Ney, H., 2019. Acoustic modeling of speech waveform based on multi-resolution, neural network signal processing. In: Proc. ICASSP.
Deng, L., Cui, X., Pruvenok, R., Huang, J., Momen, S., Chen, Y., Alwan, A., 2006. A database of vocal tract resonance trajectories for research in speech processing. In: Proc. ICASSP.
Abdel-Hamid, O., Deng, L., Yu, D., Jiang, H., 2013. Deep Segmental Neural Networks for Speech Recognition. In: Proc. Interspeech.
Sainath, T.N., et al., 2021. An efficient streaming non-recurrent on-device end-to-end model with improvements to rare-word modeling. In: Proc. Interspeech, pp. 1777–1781.
Zhang, Z., Geiger, J., Pohjalainen, J., Mousa, A., Jin, W., Schuller, B., 2018. Deep learning for environmentally robust speech recognition: an overview of recent developments. ACM Trans. Intell. Syst. Technol. 9 (5), 49.
A. Wong, M. Famouri, M. Pavlova, S. Surana, "TinySpeech: Attention Condensers for Deep Speech Recognition Neural Networks on Edge Devices," arXiv, 2008.04245, 2020.
M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling." Interspeech.