Conservatorio di Musica Alfredo Casella

Istituto Superiore di Studi Musicali

BIENNIO DI II LIVELLO IN NUOVE TECNOLOGIE E LINGUAGGI MUSICALI
MUSICA ELETTRONICA

INDIRIZZO REGIA E TECNOLOGIA DEL SUONO

Vowel phonemes Analysis & Classification by means of *OCON rectifiers* Deep Learning Architectures

RELATORE

Prof. MARCO GIORDANO

CANDIDATO

STEFANO GIACOMELLI

Matr. n°

812/II

ANNO ACCADEMICO 2022/2023

P.le Francesco Savini s.n.c. - 67100 L'Aquila - Tel.: 0862.22122 - Fax: 0862.62325 - e-mail:protocollo@consaq.it

Codice Fiscale 80007670666

...esiste un solo bene, la conoscenza,
e un solo male, l'ignoranza.

INDEX

ABSTRACT

In this work we introduce to a specific sub-field of *automatic speech recognition*: *vowel phoneme analysis* & *classification*. We approach such AI practice formulating a specific Deep Learning proposal: *OCON rectifiers* combined with vowel formant features. After we have analyzed related theories and state of the art solutions, we introduce to our proposal which aims to solve a *multi-label classification* problem by splitting it into a bank of *Multi-layer Perceptrons* binary classifiers (one for each phoneme class). Our research is structured on an *informed* hyper-parameters *grid-search method*: we draw an example of rigorous networks analysis, obtaining results in line with today's most complex architectures (90% - 93.7%) trying to optimize related learning phase, features estimation and computational distribution for *real-time* applications. Python-code and charts are respectively included in APPENDIX-A (external) and APPENDIX-B.

Keywords: Artificial Intelligence, Machine Learning, Deep Learning, Neural Networks, Digital Signal Processing, Vocal Acoustics, Speech Communication, Phonetics, Phonology

Scientific Disclaimer

Following technical results are not intended to be generalizable or production ready. Every DL technical proposal is necessarily dependent upon dataset features, optimization/initialization techniques and many other peculiarities employed. We do not directly aim to provide new human-related theories about vowel phonemes understanding.

Socio-Cultural Disclaimer

In line with fundamentals of inclusion and mutual respect of *gender theories* (LGBTQIA+ communities) the use of the terminology "*man/men*," "*woman/women*," and "*boy/girl/children*" here refers only to physiological and biological conditions which describe a vocal emission frame (in adulthood or school age).

Acknowledgements

A heartfelt thank you to my family and my better half for supporting and putting up with me during this (not easy) period of my growing up… you are and will forever be *The 3-Dimensions of My Space*.
Thanks to those who believed in this work and in my choices, and a special one to those who have sometimes doubted or still doubt it: *the doubt lays the foundation of certainty*…

INTRODUCTION

Vowel phoneme classification is a specific sub-field of a bigger branch of research defined *Speech Recognition* (SR) [1]. Speech recognition is in turn an interdisciplinary subfield of computer science, linguistics and engineering that develops methodologies and technologies to recognize[1] spoken (or even sung) communications with everyday devices, like computers, smartphones, smart assistant devices etc. It is also known as *Automatic Speech Recognition* (ASR).

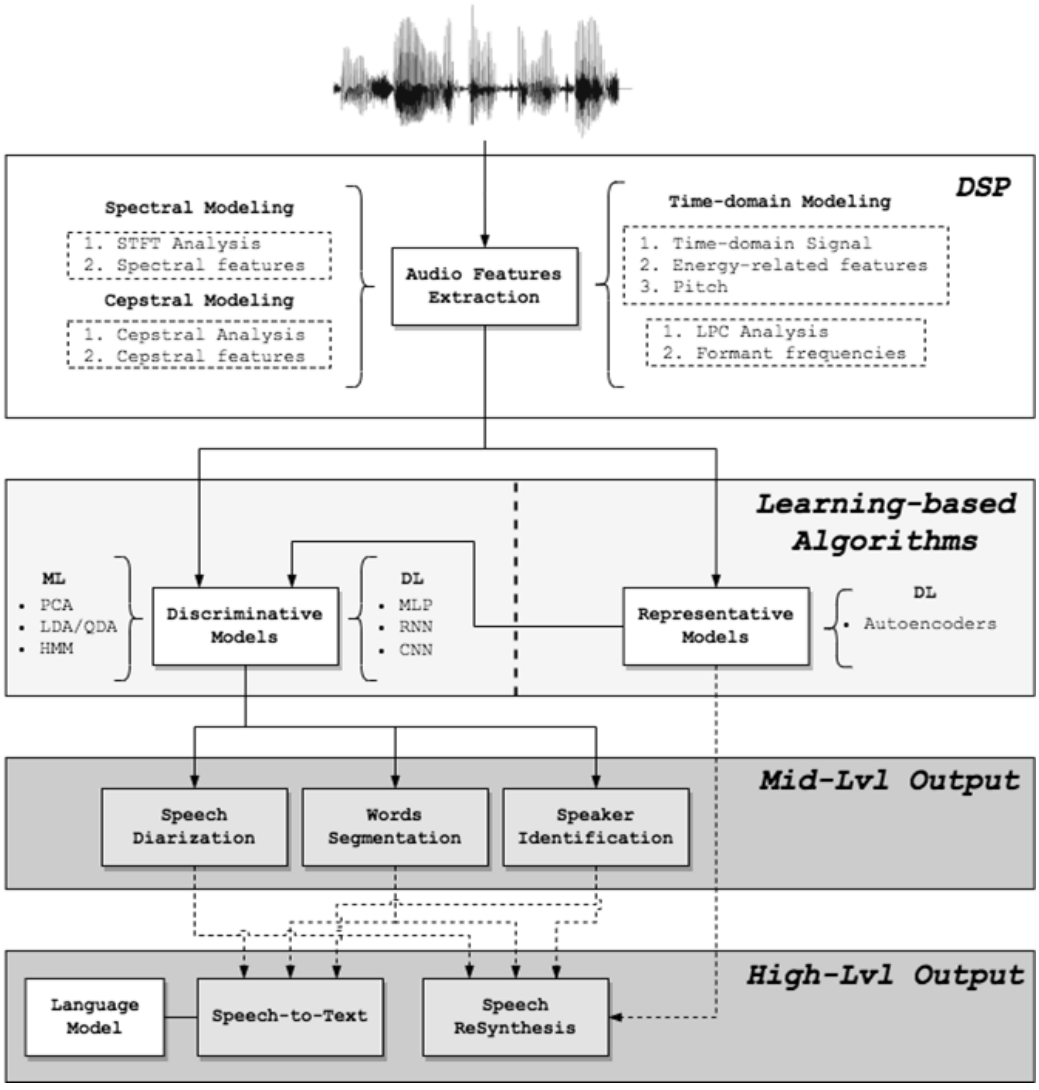


Figure 1 - Typical ASR processing chain

[1] Which is not directly synonymous with understanding.

Some ASR systems require a so-called *enrollment phase*, where an individual speaker needs to pronounce isolated vocabulary or short sentences and devices record and analyze resulting audio files: this is usually done to fine-tune *attention* and individual recognition, increasing system performance accuracy. Models that do not require this phase are defined *speaker-independent* while the others are *speaker dependent*.

The term *speaker identification* refers to identifying a specific person by key-aspects of his/her pronunciation, with the aim of improving speech recognition, *diarization* or even authentication/security processes. Speaker dependent trained models often couple previous tasks to achieve nowadays great results in *speech-to-text* (STT) applications [2].

There have been three main eras in the development of SR:

1) The *early period*: using either systems that matched speech against recorded templates or ones that supplied linguistic knowledge to pioneering rule-based AI systems;
2) *Statistically-driven period*: in which probabilities (rather than absolute rules) were used to estimate the likelihood of a match of strings of words;
3) The *modern period*: in which *deep neural networks*, big datasets and cloud computing concur in providing almost real-time speech recognition for mobile devices, with almost uncontrolled complexity solutions.

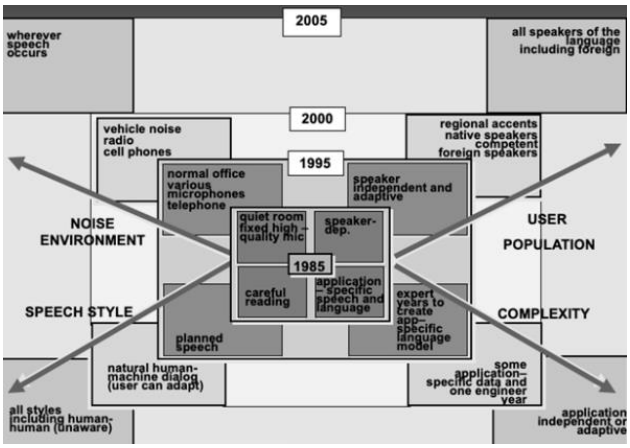


Figure 2 - ASR complexities and history

Closely according to technological eras[2], ASR complexities can be analyzed in terms of:

- *speech style*: even if context-independent isolated words analysis seems to be "as easy as useless", continuous speech recognition (inaugurated by Dr. Raj Reddy [3] at the end of 1960s) is now roughly approached by *End-to-End modeling* (E2E, *Large Language Models*), at the cost of losing contact with acoustical meanings.
  The first style represents a compulsory and maybe annoying way of vocal interaction with technology. The latter is the actual human approach [4], an intuitive as much as refined way of articulating *symbols* and *semantics*, to which machines are not accounted yet;

[2] https://en.wikipedia.org/wiki/Timeline_of_speech_and_voice_recognition.

- *environment*: human voice is recorded in *concrete* environments (i.e.: concurrent speakers, reverberant spaces, noisy conditions, topic specificity) and thereby advanced de-noising techniques are required to isolate speech signal in such a way that useful information is analyzed and extracted from the context too (when needed);
- *technological complexity*: topic complexity apparently results in technological complexity (in terms of computation but modeling too), through which we try to solve big data collection and analysis, advance features extraction, intuitive classification tasks and end-user interaction;
- *user population*: speech/speaker analysis can't be solved in an absolute general way, because it is radically marked by individual peculiarities and outliers, like physiological and psychological traits, age, culture [5] and education, languages [6], dialects, etc.

Unlike nowadays deep learning E2E models [7] [8] that jointly "learn" all the complexities of speech communication simplifying training and deployment technology processes, at the cost of an alienating methodology, the old (but still gold) machine learning and deep learning *acoustic modeling* approach, still represents an ongoing research area (for both academia and industry) due to its more robust connection with theoretical aspects of vocal physiology, phonemics, acoustics [9] [10] and digital signal processing (DSP) [11].

Figure 3 outlines a standard acoustic modeling chain, in which the speech audio signal is analyzed using DSP techniques, feed learning-based algorithms designed to recognize key-features (i.e.: audio/silence, consonants/ vowels, phonemes, words, sentences, phrases) or to encode information so that it can be efficiently transferred and decoded (*autoencoders*) [12].

The academic relevance of involving speech features analysis and modern classification technologies (mainly *neural networks*) lies in the importance of exploiting what are still called *black-boxes,* trough conscious intuition and scientific/artistic driven knowledge: a way of monitor these complex learning tools, finding a consistent interpretation of their workflow and internal states.

Due to the complexity of ASR highlighted above, in this work of thesis we decided to focus on a specific branch of *descriptive* speech modeling, which seems to show both scientific improvement potential and artistic spin-offs: vowel phoneme classification.

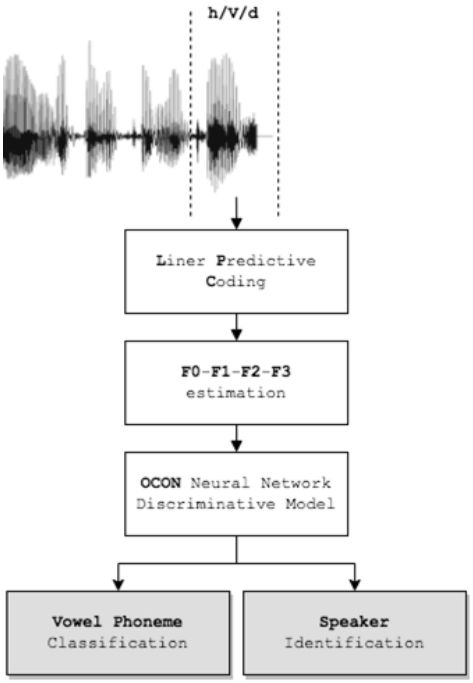


Figure 3 - Vowel phoneme analysis chain

Phoneme-based segmentation of non-silent audio frames is the common shared technique, between all kinds of speech recognition systems: phonemes are the fine-grained representation of speech content and analyzing signals by this unit allows a

better generalization of results, with a view to progress in *pronunciation modeling* [13] (phoneme concatenation) and *language modeling* (for word-chaining semantics-driven systems) [14] [15].

In the **Vocal Acoustics, Speech & Communication** chapter we will introduce vowel phonemes according to their physiological, linguistic and phonological correlates. To train and evaluate our discriminative model, we will adopt a reference dataset consisting of /*hVd*/ context-independent words recorded by a heterogeneous group of speakers, in a noise-controlled environment to minimize environmental redundancy and noisy contribution to the information transmission process.
Further insights about data structure and features will be provided in the **STATE-OF-THE-ART** and **Dataset Features Extraction & Conditioning** chapters.

Dataset of audio recordings has been processed to extract vowel nucleus segments and a smoothed estimate of the signal has been obtained by *Linear Predictive Coding/Analysis* approximation. We will introduce those time-domain and spectral-domain techniques in the **Audio Signal Analysis & Numeric Models** chapter.

After introducing **Deep Learning** and *Neural Networks* architectures, we will finally analyze the proposed discriminative model: The ***One-Class-One-Network*** classification model, which will be investigated for both vowel phoneme classification and speaker identification tasks. We introduce state of the art solutions and scoring and we will build our model by empirically evaluating the best subnetworks parametric setup: training and test evaluations will be held for each derived feature set, in order to discover possible biases and performance peaks.

In the **CONCLUSIONS & FURTHER WORKS** chapter we compare our results with references, discuss optimization headroom for outputs evaluation and further implications in real-time audio contexts.

Code scripts (*Python*) for experiments and model definition are provided in a separated documentation file [**APPENDIX-A (Code)**].

BACKGROUND

Vocal Acoustics, Speech & Communication

The fundamental purpose of speech is communication, that happens in the analog form (acoustic waveforms) of speech signals.

In order to achieve effective communication, a shared language code must be involved, which is used to convert textual messages in the form of articulated and prosodic sentences, by means of simple symbolic entries: consonants and *vowels* (arranged in so-called linguistic *phonemes*).

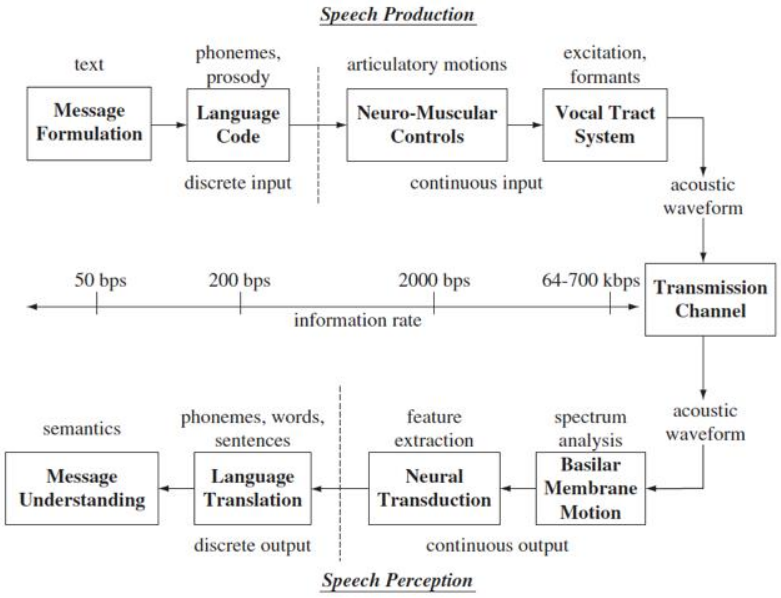


Figure 4 - Speech communication chain

These complex but spontaneous mechanism was developed by a unique combination of ourselves efforts, knowledge, culture/society and prior experiences and is established by coordinately articulating motions of our respiratory and mouth apparatus, ruled by our neuro-muscular system: discrete coded information is thus transduced in continuous neuro-electrical signaling and mechanical articulation, to obtain air-pressure variations [16] [17] and thus oral communication.

Speaking in terms of information theory (Figure 4), we can approximately assume a speaking emission rate of 10 symbols per second, with somewhat a total amount of 2^5 symbols for the standard English spoken language (including alphabet and simple punctuation). So, in the 1st stage of message formulation we can roughly estimate a rate of information flow:

$$5\ bits\ (per\ symbol) * 10\ symbols\ (per\ second) = 50\ \text{bps}$$

Entering in the "language coding" phase, phonemes and prosody markers increase the information rate by an average factor of 4: at the end of symbolic encoding we have already a data rate of 200bps. Estimates of bandwidth and accuracy [18] suggest that data sample rate for articulatory nervous signaling is around 2000 bps (an increasing factor

of 10). Distinguishable digitized speech signals require instead variable sampling rate: 8KHz (i.e.: for telephone transmission), 44.1KHz or more (i.e.: CD quality).
8 bits quantization in a log-scale disposition, impose a minimum of 64Kbs transmission (phone communication) or 705.6Kbps for standard audio quality reproduction: the total amount of increase can easily reach a factor of 10^3.
This raw estimate shows that a few bits of spoken information requires a huge amount of data sampling to be digitized and robustly decoded in a psycho-hearing fashion mechanism. Obviously, it was just an example provided to give a taste of the high complexity level of this topic, leading us to a fundamental remark: following theoretical topics, terminology and examples will be just streamlined to give a better understanding of one of these work subjects (*vowel phonemes*). Any further insight can be reviewed in cited **REFERENCES**.

Vowel emission and characteristics will be now introduced outlining some fundamental aspects of the complex *communication chain* [19] [20], starting with physiological ones and then approaching linguistics and acoustics correlates.

Vocal Tract & Speech Emission

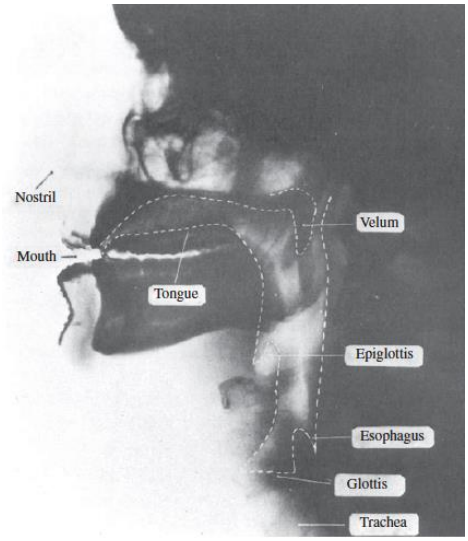


Figure 5 - Human X-ray head imaging

From an x-ray imaging (Figure 5) [18] it's possible to locate the essential physical features of human anatomy, which concur into the final stage of speech communication chain (mechanical emission).
The vocal tract is shown as a tube of non-uniform cross-sectional area (dotted) that is bounded at one end by vocal cords (delimiting *glottis*) and at the other by the *mouth* opening.
This tube which is composed of nasal and *oral cavities* and by the *pharynx* (which connects the *esophagus* to the mouth) serves as an acoustic transmitter (complex *resonant system*) for air flowing. In an average male the total length of the vocal tract is 17 - 17.5cm, while the effective cross-sectional area which is determined by the positions of tongue, lips, jaw and velum, varies from null (complete closure) to about 20cm^2.

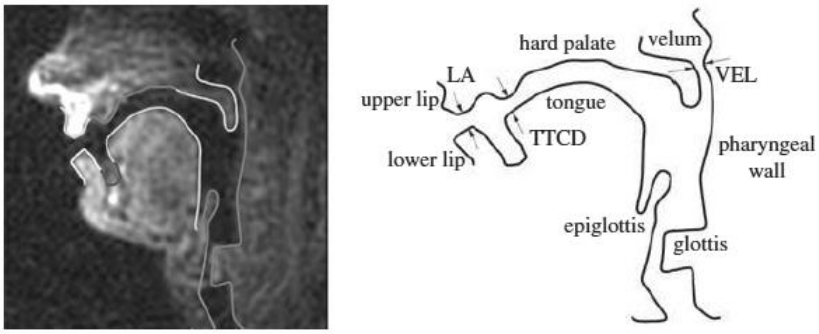


Figure 6 - Human MRI head imaging

Nasal tract is bounded by the *velum* and by the *nostrils*. The velum represents a trapdoor-like mechanism, located behind the mouth cavity and when lowered, foster the acoustical coupling with the vocal one (i.e.: to produce *nasal sounds*); the mouth cavity instead is composed by *tongue*, *lips*, *jaw* and mouth which are the main mechanical agents in our voluntary speech articulation movements.

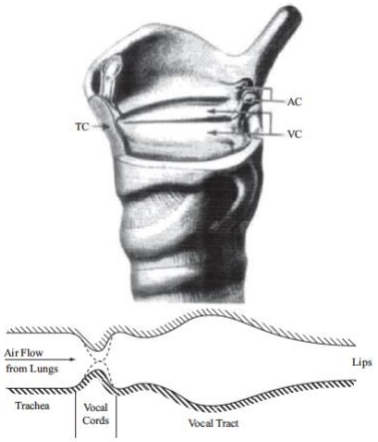


Figure 7 - Phonatory system artwork

The parts of the body involved in speech production (*source complex system*) are *lungs*, which contains air to excite the vocal tract resonances and the *chest* cavity which acts as a pressure agent to force the air emission from the lungs. The *trachea* conducts the air to the *vocal cords* that vibrate when tensed and pressed by air. Air flow is then chopped up by coordinated opening and closing of the glottal orifice into quasi-periodic pulses, that in turn enters inside the vocal tract (*phonation*) and undergoes spectral shaping phenomena, depending on the instantaneous morphological state of resonant cavities.

Vocal cords (or *membranes*) become tensed via appropriate musculature control of the cartilages, respectively arytenoid cartilage (AC) and thyroid cartilage (TC) that surround vocal folds (VC): when tensed a relaxation oscillator is formed. Air pressure continuously builds up behind the closed vocal cords, until they are eventually blown apart and let air-pulses flow in a quasi-periodic way.

This complex acoustic model was originally idealized [21] by Carl Gunnar Michael Fant (1919 – 2009) is often referred as the speech "source - filter" model: a mechanical abstraction useful to explain physical activity of our speech processes but for surgical & engineering developments too, as in the emblematic case of the *Artificial Larynx*: a device patented [22] for the first time in 1929 by Bell Telephone Laboratories (AT&T Corporation) and effectively involved in surgery by 1959 [23].

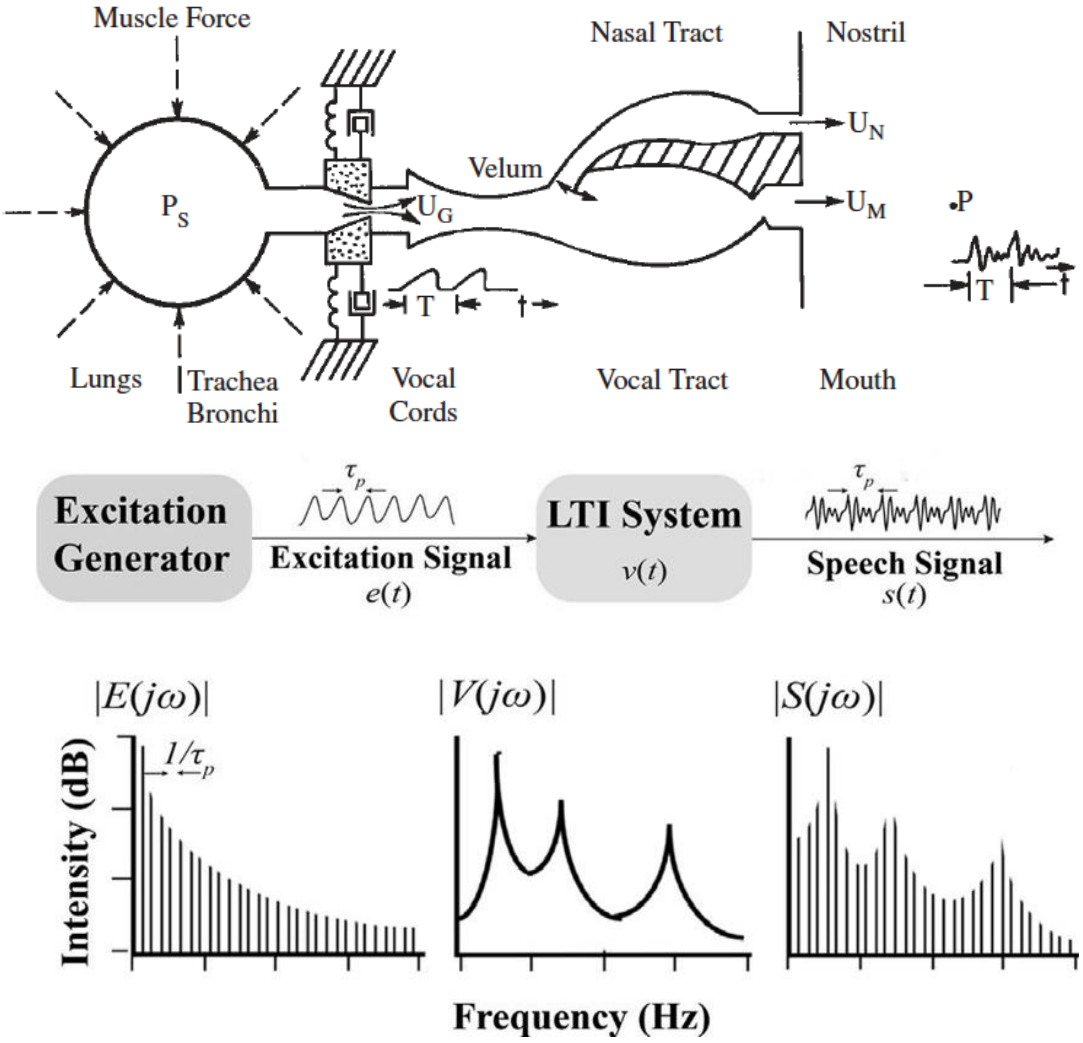


Figure 8 - Vowel emission: mechanical and digital features

In the **Audio Signal Analysis and Numeric Models** chapter will be clarified how engineers easily transpose this mechanical abstraction to functional equations and system graphs (Figure 8) for DSP applications [24].

Vowel Phonetics and Phonology Fundamentals

Speech can be represented by a finite set of typical language symbols called *phonemes*, with quantity varying upon the language and the refinement of the linguistic analysis, but for most, ranging between 32 and 64.

The emblematic case of *vowels* includes two complementary definitions:

- *phonetic*: a vowel is a sound produced with an open vocal tract, it is *median* (the air escapes along the middle of the tongue), oral (at least some of the airflow must escape through the mouth), *frictionless* (no turbulent airflow) and *continuant* (no complete closure of vocal tract). It does not entail significant build-up of air pressure at any point above the glottis.
- *phonological*: a vowel is defined as a syllabic sound in the peak of a syllable (also *nucleus*), whereas consonants usually identify relative *onsets* and *codas*.

The International Phonetic Association [25] [26], founded in 1886 by Paul Passy (1859 – 1940) tries to establish unambiguous ruling systems for *sound-to-symbol* encoding, for most of the world's languages.

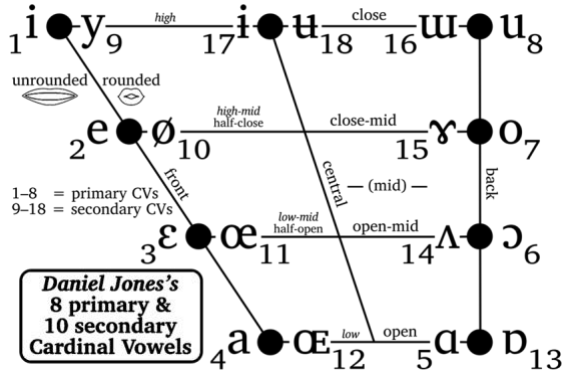


Figure 9 - Daniel Jones vowels cardinal system

IPA vowels system[3] inherits from phonetician Daniel Jones (1881 – 1967) [27] researches, about *cardinal vowels* quadrilateral set-up. Jones' English-vowels schematics, first presented in *An English Pronouncing Dictionary* [28], tries to organize vowel hierarchy by means of lips/tongue mutual positioning during speech emission.

The vowel [i] is recognized to be produced with the tongue as far forward and as high in the mouth as possible, producing no friction with spread lips. The vowel [u] is produced with the tongue as far back and as high in the mouth as possible, with protruded lips while the [ɑ] is produced with the tongue as low and as far back in the mouth.

Other vowels were "auditorily equidistant" placed between previous three corners, at four degrees of oral opening or height: *close* (high tongue positioning), *close-mid*, *open-mid* and *open* (low tongue positioning).

With a *front-back* separation, the schematics marks eight reference vowels also known as the eight *primary cardinal vowels*, considered to be the most common in the world's languages.

During vowel emission, lips position can be reversed with the one of the opposite front-back side, generating *secondary cardinal vowels* (less common), often characterized by fast *diphthongal* structures.

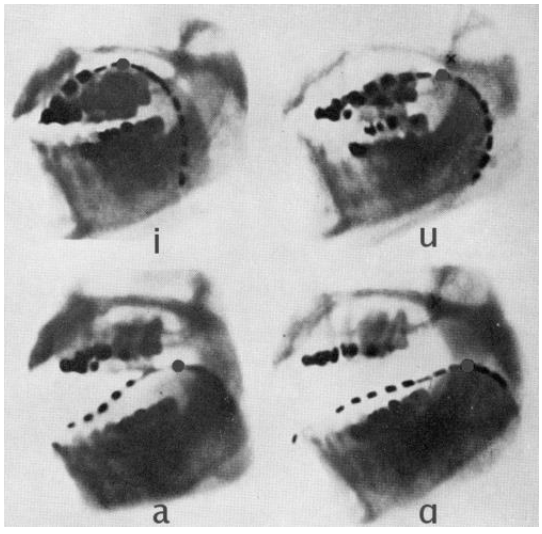


Figure 10 - MRI vowel emission (mouth articulation)

[3] See **APPENDIX-B** for the latest IPA Alphabet revised list.

From the aural-linguistic perspective, we can identify primary cardinal vowels as pure *monophthongs*, whose sound articulation at both beginning and end is relatively fixed and does not glide towards new position. Secondary vowels instead can also assume the form of *diphthongs* (or gliding vowels): commonly described as a vowel sound with two vocalic targets, a starting and an ending one.

This articulatory schematization has been known to be partially inaccurate and misleading, in fact the linguist Peter Ladefoged (1925 – 2006) in *A Course in Phonetics* [29] [30] and *The Sounds of The World's Languages* [31] states that early phoneticians misunderstand articulatory tongue's location recognition, with respective *formant frequencies*.

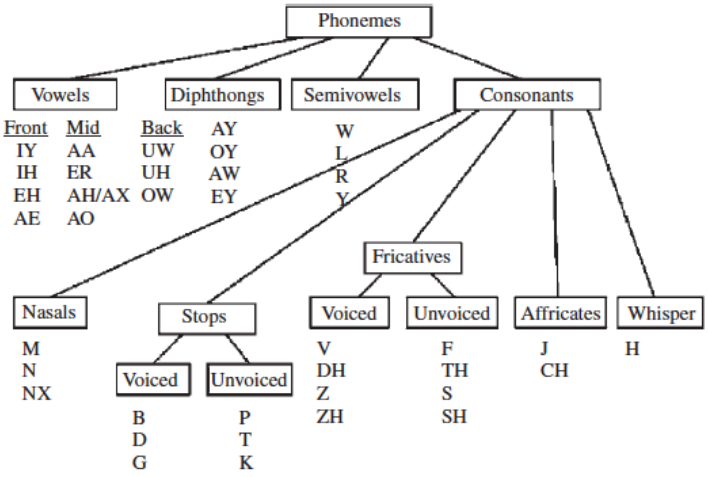


Figure 11 - Vowels phonetics/ARPABet classification

ARPAbet Vowels

	b_d	ARPA		b_d	ARPA
1	bead	iy	9	bode	ow
2	bid	ih	10	booed	uw
3	bayed	ey	11	bud	ah
4	bed	eh	12	bird	er
5	bad	ae	13	bide	ay
6	bod(y)	aa	14	bowed	aw
7	bawd	ao	15	Boyd	oy
8	Budd(hist)	uh			

Figure 12 - /bVd/ vowel ARPAbet transcription

In the field of digital phonology, the Defense Advanced Research Projects Agency (DARPA) [32] developed the ARPABET (also ARPAbet) [33]: a transcription code-style born as part of DARPA funded *Speech Understanding Research* project, in the 1970s.
It is still applied to digital encoding of General American English phonemes, with distinct sequences of ASCII characters, without special characters needed.
Two alternative coding styles were developed:

1) segment representation with one case-sensitive character (alternating upper- and lower-case letters)
2) segment representation with variable size encoding (case-insensitive).[4]

The ARPABET has been involved in several speech synthesizers project too, including *Computalker* for the S-100 system, *SAM* for Commodore 64, *SAY* for Amiga, *TextAssist* and *Speakeasy*, which used the *Votrax SC-01* dedicated speech synthesizer chip, for PC.
We will see that extended revisions of the ARPABET are involved in many speech (and singing) dataset corpus transcriptions, for DSP and statistical analysis.

[4] See **APPENDIX-B** for the complete ARPAbet characters list.

Articulation & Speech Formant Frequencies

Formant frequency concept is strictly related to articulation aspects of vowel emission. As previously seen, the concept that vowel qualities can be determined by tongue positioning and lip rounding continues to be used in speech pedagogy, as it provides an intuitive explanation of phonemic distinction, but nowadays we know that the involved portion of cross-sectional area of vocal tract plays a major role in the unique formant pattern composition, for the emitted sound.

The dependence of cross-sectional area upon distance along the vocal tract is defined as the *area function*, which can be estimated with combined MRI imaging and numerical techniques [34] [35]. It describes cavity surfaces that enforce the characteristic high-level shaping for vowel spectra (an example in Figure 13 [36]).

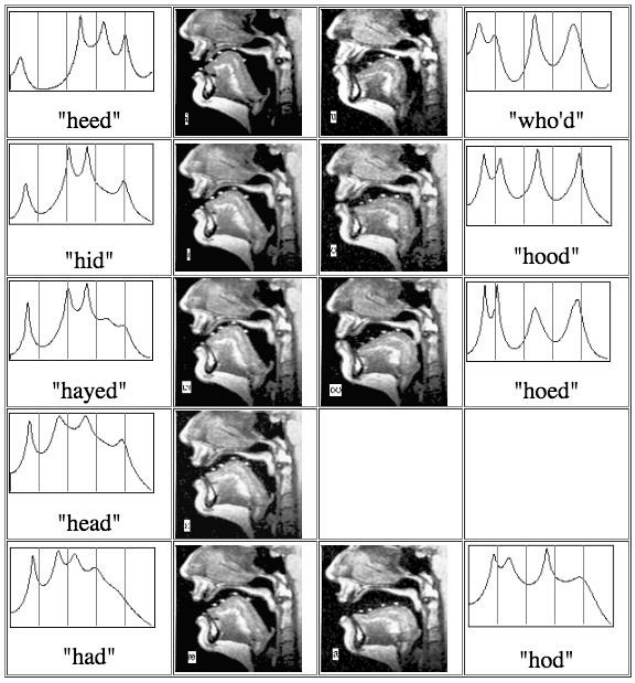


Figure 13 - /hVd/ vowel formants MRI

The 1st formant (simply F1) is the lowest independent vowel resonance and is strictly associated with the height of the tongue positioning.
In closed (high) vowels such as [i] and [u], F1 is consistent with the tongue being positioned close to the palate, high in the mouth; whereas in open (low) vowels such as [a], F1 is consistent with the jaw being open and the tongue being positioned low in the mouth.
Height is defined as inversely proportional to F1, so the higher F1 the lower (more open) the vowel: in Figure 10 we already observed a phonological 5-levels sampling of articulatory tongue heights:

- close (high)
- high-mid-close
- close-mid (also *true-mid*)
- low-mid-open
- open (low)

The vowel height (and so F1) appears to be the primary cross-linguistic feature for all spoken languages: for this reason, it is also defined as a *contrastive recognition feature*.

Vowel *backness* (more rarely *frontness*) is defined with the relative positioning of the tongue to the back of the mouth and the 2nd formant (F2) is the specific resonance linked with this articulatory aspect.
In front vowels such as [i], where F2 is relatively high, generally we can observe tongue positioning forward in the mouth, whereas in back vowels such as [u], F2 is generally low and consistent with the tongue being shifted towards the back of the mouth. In Figure 10 we observed a raw 3-levels sampling of tongue backness:

- front
- central[5]
- back

Roundedness is defined thanks to the peculiar lips rounding, typical of certain vowel emission. Rounded vowels can be identified by a consistent decrease on expected F2 (with respect to unrounded cases) and a slight one, on F1. This vowel property is better identified as *labialization*: in mid-to-high rounded back vowels, lips are generally protrude outward (*endolabial* rounding) with externally visible insides of lips; whereas in mid-to-high rounded front vowels, lips are generally compressed with margins pulled in and drawn towards each other (*exolabial* rounding).

The classification by "tongue direction" is not completely supported by articulatory evidence. John Esling (1949 -) author of *The Laryngeal Articulator model* [37] [38] [39] (accepted by IPA too) defines that vowels may be characterized by three tongue movement directions (from its rest positioning):

- front (forward),
- raised (upward and back)
- retracted (downward and back).

and front vowels should be qualified as close or open referring to the jaw opening rather than tongue positioning.

In addition, rather than being a unitary category of back vowels, it should be possible to identify *raised vowels*, where the body of the tongue approaches the velum, and *retracted vowels* where the root of the tongue approaches the pharynx.

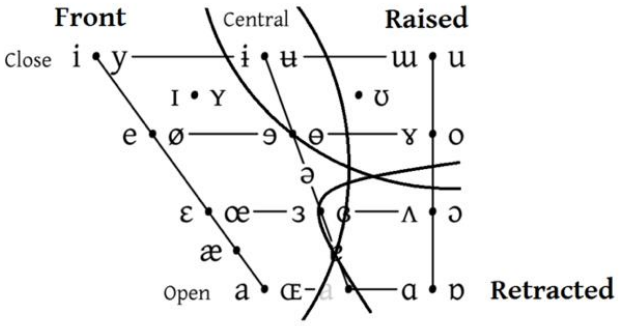


Figure 14 - IPA cardinal system

(post - *Laryngeal Articulation Theory*)

Nasal vowels (and consonants, more common) are produced by glottal excitation and concurrent vocal tract constriction at some point along the oral route. The velum is lowered so as to let air flow mainly through the nasal tract, with sound being radiated at the nostrils level. The oral cavity, although frontally constricted, remains acoustically coupled to the pharynx and serves as a resonant complex cavity that traps acoustic energy at certain frequencies, effectively acting as anti-resonances (zeros) in the complex filtering response.

Furthermore, nasal consonants and *nasalized vowels* (vowels preceding or following nasal consonants) are characterized by resonances that are spectrally broader than those of canonical vowels: broadening occurs because the

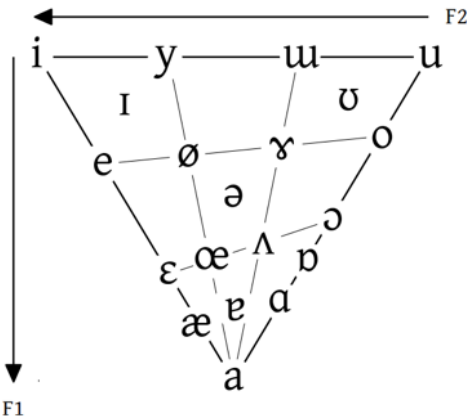


Figure 15 - Vowel triangular system

(formant frequency tendency)

[5] Can be further refined with a 3-level up-sampling: near-front, central, near-back.

inner surface of the nasal tract is convoluted so that the nasal cavity has a relatively large ratio of surface area to the total amount of cross-sectional area.
The peculiar case of *r-colored* (also *rhotic* or *retroflex*) vowels represents a modified vowel emission occurring with concurrent [r] consonants, that commonly results in a lowering of the 3rd formant frequency.
This kind of vowels can be articulated in various ways:

- the tip of the tongue may be turned up during (at least) part of the articulation (*retroflex articulation*)
- the back of the tongue may be bunched.

In addition, the vocal tract may often be constricted in the region of the *epiglottis*.

In the **STATE-OF-THE-ART** chapter we will see that many researchers spent their efforts in optimizing formants estimate, for vowel phonemes automatic recognition. Nowadays four to five formant frequencies seem to be sufficient for both vowel phonemes and unambiguous speaker recognition tasks, but a fine resolution in sound appraisal is often required, often undermining real-time availability.
An approximation (Table 1) [40] of vowel formant values (F1, F2 and respective distance) is given below for guidance purposes only[6].

Vowel (IPA)	F1 (Hz)	F2 (Hz)	F1 – F2	Vowel (IPA)	F1 (Hz)	F2 (Hz)	F1 – F2
a	850	1610	760	ø	370	1900	1530
ɑ	750	940	190	œ	585	1710	1125
ɒ	700	760	60	ɶ	820	1530	710
e	390	2300	1910	ɔ	500	700	200
ɛ	610	1900	1290	u	250	595	345
ɤ	460	1310	850	ɯ	300	1390	1090
i	240	2400	2160	ʌ	600	1170	570
o	360	640	280	y	235	2100	1865

Table 1 – IPA reference formant estimates

[6] We should have learned that formant frequencies suffer from high degrees of statistical variance.

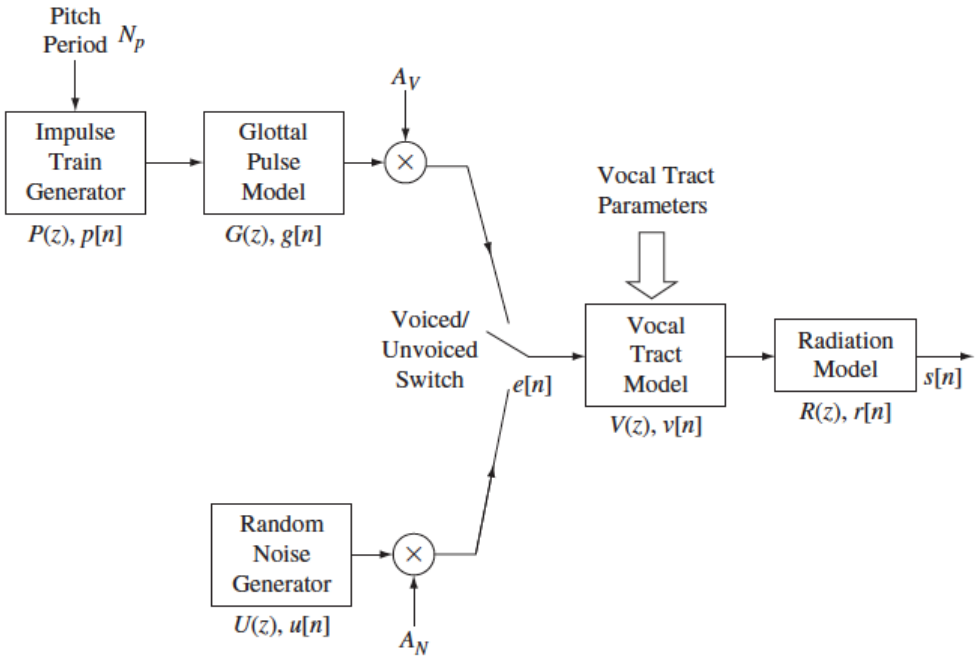


Figure 16 - DSP speech acoustic model

Most speech processing applications aim to obtain a convenient representation of the information carried by speech signals. This can be achieved by assuming that the discrete signal $s[n]$ is the output of a parametric synthesis model, as shown in Figure 16.
The model parameters sought are:

- the time-varying *pitch period* (N_p, in samples), or *pitch frequency* (F_p, in Hz with $F_p = \frac{SR}{N_p}$ and SR the signal sampling rate) for voiced speech regions;
- the glottal pulse model sequence, $g[n]$ [41];
- the time-varying amplitude envelope of voiced excitations, A_V;
- the time-varying amplitude envelope of unvoiced excitations, A_N;
- the time-varying excitation method: quasi-periodic pitch pulses for voiced sounds or pseudo-random noise for unvoiced sounds;
- the time-varying sequence out of convolution between excitation and vocal tract impulse response, $v[n]$
- the time-varying sequence from the convolution between the vocal tract filtered signal and the radiation model impulse response[7], $r[n]$.

We shall refer to any representation derived by DSP operations as a *set of parameters* of a speech model (or simply a *parametric representation*).
There are several domains for speech signal representation and we will focus on those involved in *spectral envelope* and fundamental/formant frequencies estimate: *time-domain* and *cepstral/LPA-derived*.

[7] For simplicity, it is assumed to be fixed over time.

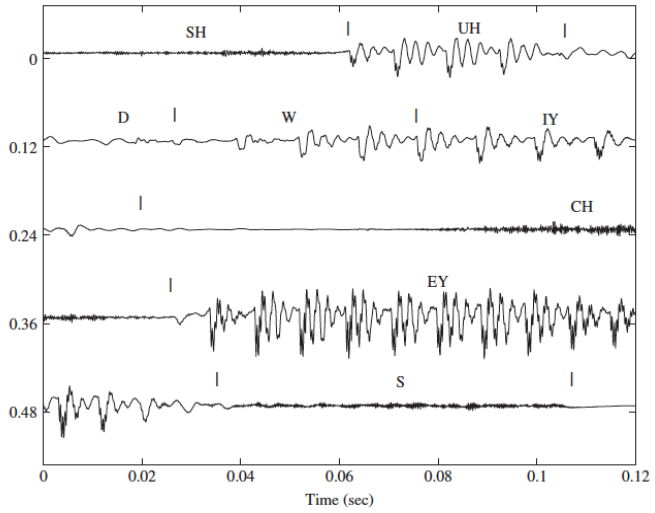


Figure 17 - Consonat/vowel phonemes waveform segmentation

For effective phoneme classification we need to define automatic methods for marking boundaries between voiced and unvoiced intervals, and also for determining the pitch and period within selected voiced segments.

The fact that these variations are often so evident in the vocal waveform structure (as in Figure 17) suggests that time-domain processing techniques should be able to provide useful estimates of these features, in particular: energy, vocal state (voiced or unvoiced) and pitch.

The usual underlying assumption is that the properties of the speech signal change relatively slowly with time, compared to the detailed sample-to-sample variation of the waveform): rates of change are in the order of 10–30 times/sec. This assumption leads to a variety of *short-time processing* methods in which short segments (analysis *frames*) of the speech signal are isolated by sliding *window functions* and processed as if they were pseudo-independent sustained sound, with isolated "fixed" properties.

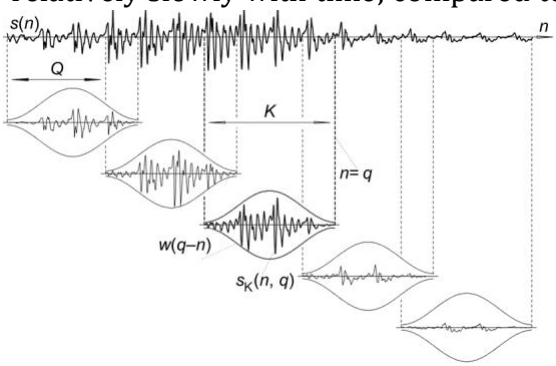


Figure 18 - Vowel phoneme windowing

A generic formulation for time-domain derived features is:

$$Q_n = \sum_{n=0}^{N} T(s[n])w[q-n]$$

where Q_n is the feature value/vector measured at time n overlapping frames (in Figure 18) obtained with a sliding window w, of length q. T represent an abstract placeholder for feature-specific computations.

There is always a degree of *uncertainty* in short-time measurements due to various levels in variations of actual speech. The combined effect of multiplying by the shifted window and summing the products, is equivalent to *low-pass filtering* the signal segment by an impulse response that is the window sequence itself. Often measurements are normalized by a factor of:

$$L = \sum_{n=0}^{N} w[q-n]$$

which is the *effective window length*, so as to transform simple measurements in weighted averages. A more comprehensive set of notions about windowing techniques and relative properties is provided in [42] [43] [44] [45].

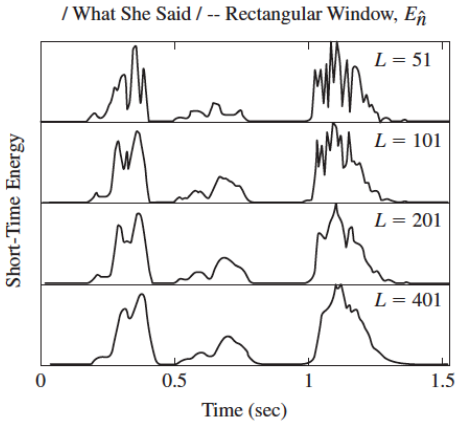


Figure 20 - E_n voiced segments

Segmenting voice and non-voiced fragments of a speech signal requires an evaluation that is sensitive to the time-varying changes in amplitude, like the *short-time energy* function:

$$E_n = \sum_{n=0}^{N} (x[n])^2 w[q-n]$$

As can be seen in Figure 19, the values of E_n for unvoiced segments are significantly lower than those for voiced segments and this difference in amplitude can be enhanced by the squaring operation. An example of application of the short-time energy representation is the *automatic gain control* (AGC) mechanism, integrated in some speech waveform coding (Figure 20), also defined as the *exponential windowing samples variance* estimate.

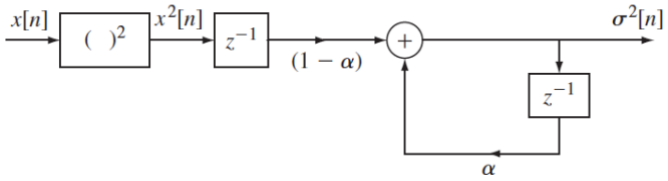


Figure 19 - AGC (exponential windowing) flowchart

One problem with this estimate is that it is very sensitive to large signal levels (since they are directly squared) thereby emphasizing large sample-to-sample variations, in $x[n]$.
One way to alleviate this problem is to define a *short-time magnitude* function:

$$M_n = \sum_{n=0}^{N} |x[n]| * w[q-n]$$

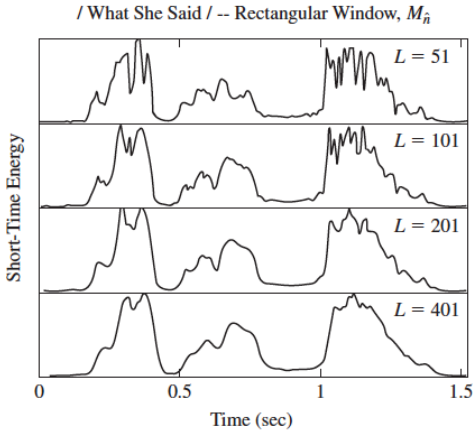


Figure 21 - M_n voiced segments

where the weighted sum of squared values is replaced with absolute values of the signal. Resulting differences are particularly noticeable in unvoiced segments (Figure 21).

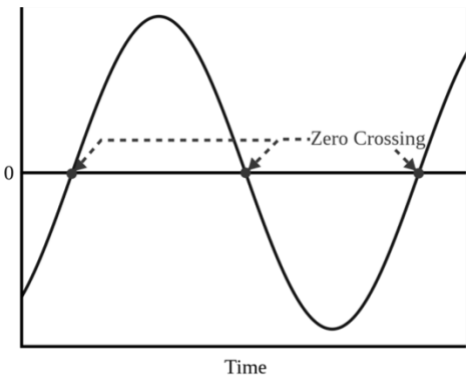


Figure 22 - Zero Crossing points

In the context of discrete-time signals, a *zero-crossing* is said to occur when successive waveform samples have different algebraic signs. The *rate* at which zero-crossings occur is a simple and often highly reliable measure (if properly processed) of the periodicity content of a signal: this is particularly true for narrowband signals.
Actual speech signals are broadband signals and the interpretation of the *short-time zero-crossing rate* is therefore much less reliable, but rough estimate of spectral energy disposition of the signal can be obtained using a sort of representation based on the

short-time average zero-crossing rate, defined as the *average number of zero-crossings*, in a block of L samples.

$$ZC_n = \frac{1}{2L}\sum_{n=0}^{N} |sgn(x[n]) - sgn(x[n-1])| * w[q-n]$$

where L is the effective window length, and $sgn(\cdot)$ "signum" operator is defined as:

$$sgn(x[n]) = \begin{cases} -1, & x[n] < 0 \\ 1, & x[n] \geq 0 \end{cases}$$

with w is usually a *rectangular window*.

Figure 23 shows that to extract a useful measure of frequency content, the input signal waveform must be *high-pass filtered* to ensure that no DC-component exists, prior to zero-crossing rates (ZCR) computation.

For a sampling rate of $SR = 1/T$ and a segment duration of τ seconds, corresponding to an interval of $M = SR \cdot \tau = \tau/T$ samples, we can modify the rates unit of Z_n multiplying its definition by M, thus obtaining a measurement in Hz.

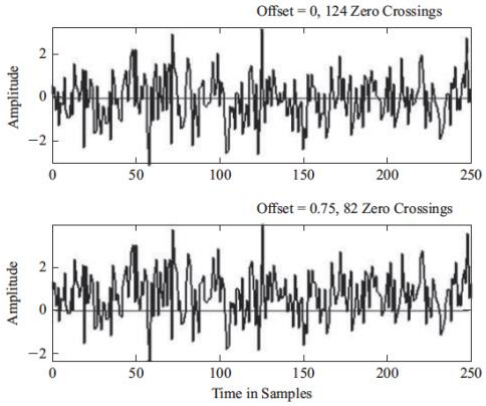


Figure 23 - DC side-effects on ZCR

The total energy of voiced speech is considerably greater than that of unvoiced speech. Therefore, voiced segments should be characterized by relatively high energy and relatively low zero-crossing rate, while unvoiced speech will have a relatively high zero-crossing rate and low energy. It is not possible to be absolutely precise about what *low and high values* mean but statistics can help us: Figure 24 shows an example of a distribution of zero-crossings in case of a vowel sound followed by a consonant one.

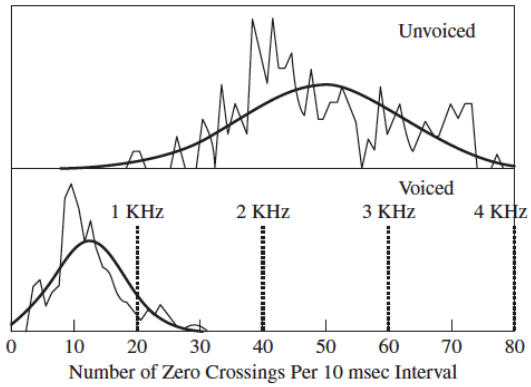


Figure 24 - Zero Crossings distribution

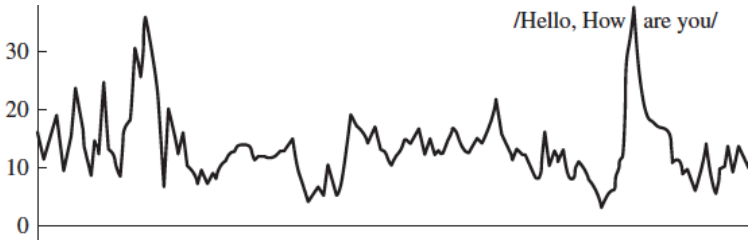


Figure 25 - sentence ZCR envelope

The *autocorrelation function* representation of a discrete signal is a convenient and alternative way of displaying and investigating periodicities. The computation of the

short-time autocorrelation function is usually carried out on a frame-by-frame basis too, with finite-length windows:

$$AC_n[k] = \sum_{m=0}^{M} (x[n+m] * w[m]) * (x[n+m+k] * w[m+k])$$

where k is an arbitrary *lag* (in samples).

The autocorrelation function of a periodic signal is also periodic with the same period and reaches a maximum at samples $[0, \pm Np, \pm 2Np, \ldots]$. This means that regardless of the time origin of the periodic signal, the period can be estimated by finding the location of the first maximum in the autocorrelation function (within a predefined range). An extended definition of the *modified auto-correlation function* and related algorithms can be found in [9] [10] [17] [16] [46].

The computation of the autocorrelation function involves considerable arithmetic, but a technique that eliminates the need for multiplications is based upon the idea that for a truly periodic input (of period Np), the sequence:

$$d[n] = x[n] - x[n-k]$$

would be zero for $k = [0, \pm Np, \pm 2Np, \ldots]$ and for short segments of actual voiced speech it is reasonable to expect that $d[n]$ will be relatively small (behind a certain threshold value) at multiples of that period. The *short-time average magnitude difference function* (AMDF) [47] is thus defined as:

$$AMDF_n[k] = \sum_{m=0}^{M} |x[n+m]w_1[m] - x[n+m-k]w_2[m-k]|$$

with w_1 and w_2 two arbitrary windowing functions (better if of the same length) inspired by the *modified* formulation of the autocorrelation function. In fact, it's quite noticeable how similar this implementation is to that of autocorrelation, optimized for real-time contexts (by replacing multiplications with subtractions).

Further insights about this topic and time-domain representations can be found in [48] [49] [50].

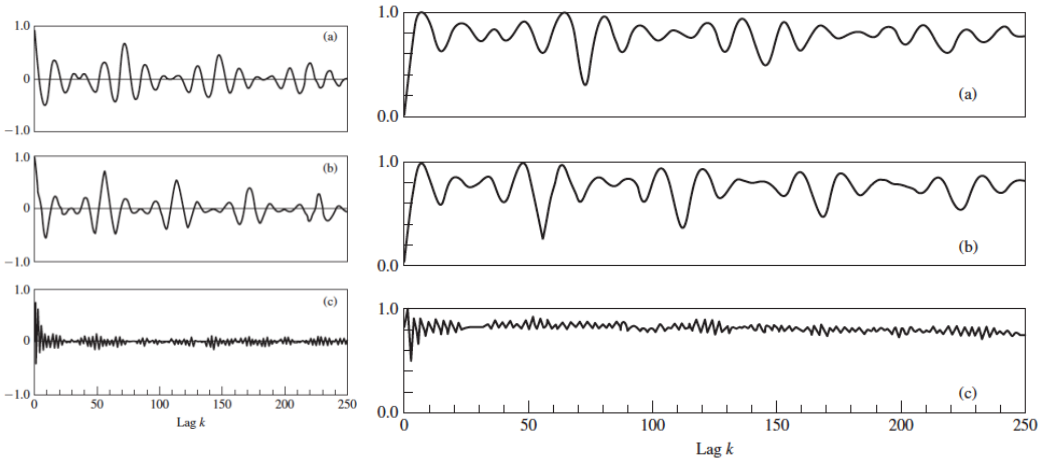


Figure 26 - ST-Autocorrelation vs AMDF

Cepstrum & LPC Methods: Spectral Envelope & Formants Estimate

In 1963 an iconic paper was published with one of the most unusual titles in the engineering literature: *The Quefrency Alanysis of Time Series for Echoes* [51]. In this paper, the authors observed that the logarithm of the Fourier spectrum (*log-spectrum*) of a signal plus an echo (a delayed and scaled replica) consists of the logarithm of the signal spectrum plus a single periodic component (due to the echo). To emphasize that the log-spectrum should be considered as a waveform (*spectral envelope* [52]), to be subjected to Fourier analysis, they defined the term *cepstrum* of a signal: the power spectrum of the logarithm of the power spectrum of a signal.

For discrete-time signals, a definition that captures the essential features of the original definition is that the cepstrum of a signal is the *inverse discrete-time Fourier transform* (IDTFT) of the logarithm of the magnitude of the *discrete-time Fourier transform* (DTFT) of that signal:

$$c[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} log\left|X(e^{j\omega})\right| e^{j\omega n} d\omega$$

where the DTFT $X(e^{j\omega})$ of the signal is:

$$X\left(e^{j\omega}\right) = \sum_{n=0}^{N} x[n] e^{-j\omega n}$$

Low quefrency values will correspond to slowly varying (in frequency) components of the log magnitude spectrum, while high quefrencies correspond to rapidly varying components. Isolated peaks, at multiples of a quefrency (N_p), correspond to a periodic component in the log magnitude spectrum, with $T = \frac{2\pi}{N_p}$ in rad/sec., or $\frac{SR}{N_p}$ in Hz.

Simultaneously with the introduction of the cepstrum, Alan Victor Oppenheim (1937 -) developed a new theory in which classes of nonlinear systems could be defined on the basis of a generalized *principle of superposition*, based on the mathematical theories of linear vector spaces [53] [54] [55]: he called those systems *homomorphic systems*. The principle of superposition states that if an input signal is composed of an additive (linear) combination of elementary signals, then the output is an additive combination of the corresponding outputs. If this system is linear and time-invariant, the output can be expressed as the convolution sum:

$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k] = x[n] * h[n]$$

where $h[n]$ is the impulse response of the system.

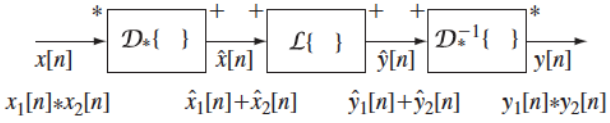


Figure 27 - Homomorphic filter (linear formulation)

Homomorphic filters represent easy and efficient tools to separate

convolved excitation $x_1[n]$ and vocal tract $x_2[n]$ components of the speech model (see Figure 27).

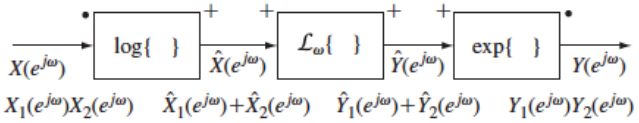


Figure 28 - Homomorphic filter (complex formulation)

An alternative representation of the *canonical form* of convolution can be inferred by DTFT application, which turns signal convolution into spectra multiplication: an operation that is commutative and associative too.

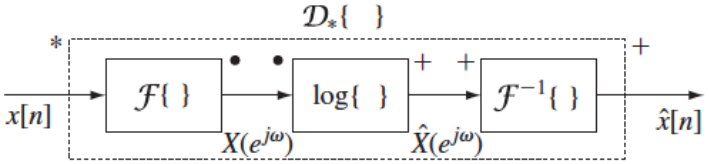


Figure 29 - Homomorphic filter (IDFT formulation)

According to this reasoning, to represent signals as sequences (rather than in the frequency domain) the *characteristic system* can be represented as in Figure XX, where the log function (a linear operator) is surrounded by the DTFT operator and its inverse:

$$\mathcal{F}\{\ \} = X(e^{j\omega}) = \sum_{n=0}^{N} x[n]e^{-j\omega n}$$

$$log\{\ \} = \hat{X}(e^{j\omega}) = log|X(e^{j\omega})|$$

$$\mathcal{F}^{-1}\{\ \} = \frac{1}{2\pi}\int_{-\pi}^{\pi} \hat{X}(e^{j\omega})e^{j\omega n}d\omega$$

thus, providing that cepstrum is a de-convolution homomorphic system.

In the acoustic model (Figure 16), the pressure signal at the lips ($s[n]$) for a voiced section of speech is represented as the convolution $s[n] = p[n] * h_V[n]$, where $p[n]$ is the quasi-periodic voiced excitation signal and $h_V[n]$ represents the combined effect of the vocal tract impulse response $v[n]$, the glottal pulse $g[n]$, the radiation load response at the lips $r[n]$, and the voice gain A_V:

$$h_V[n] = A_V(g[n] * v[n] * r[n])$$

And similarly, for unvoiced signals:

$$s[n] = u[n] * h_U[n]$$

and

$$h_U[n] = A_U(v[n] * r[n])$$

where $u[n]$ is the pseudo-random excitation signal.
Following a synthetic example of a vowel emission (from [56]) with relative cepstral alanysis [57] [58].

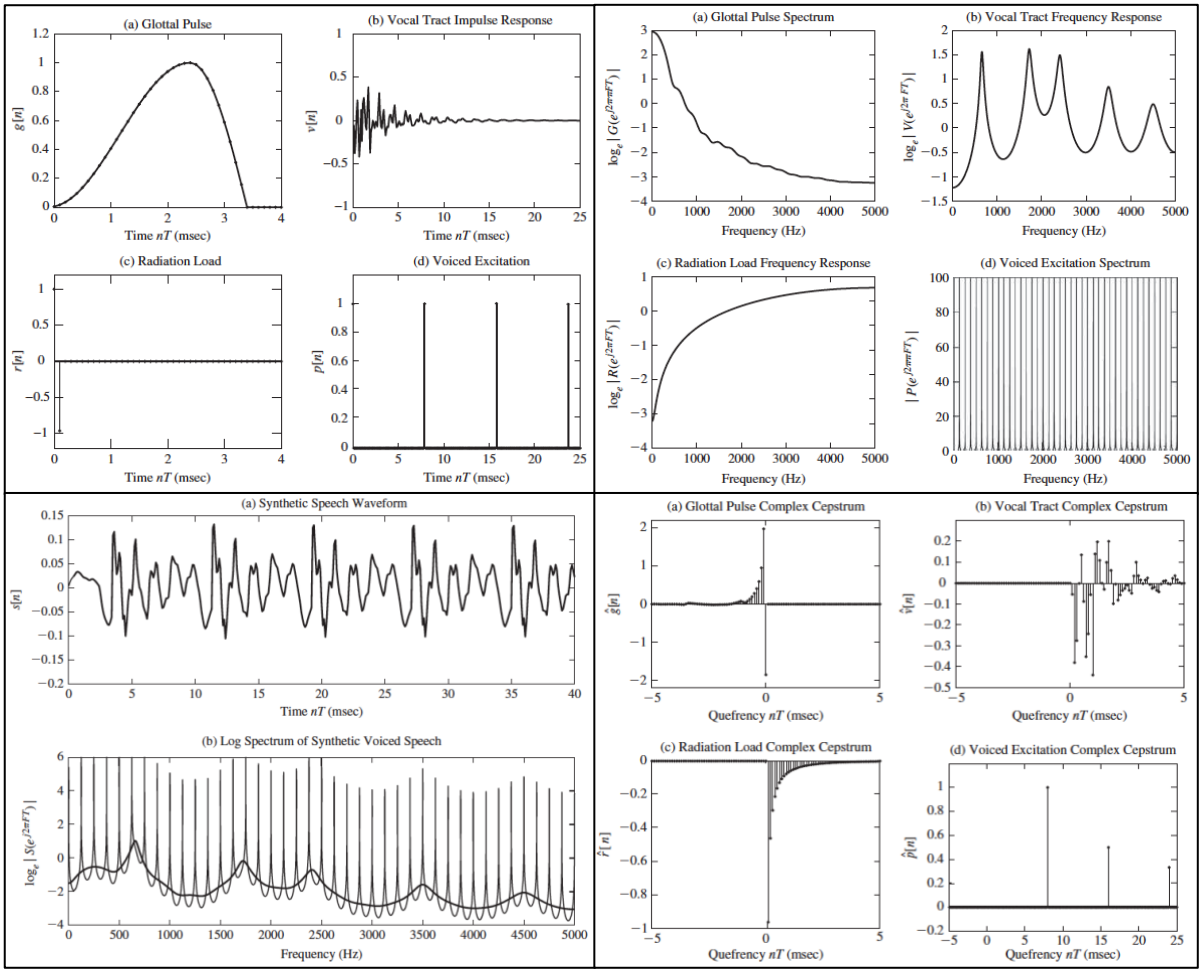


Figure 30 - Synthetic speech vowel cepstral alanysis example

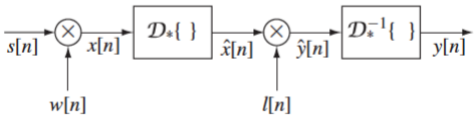


Figure 31 - Real-time ST-Cepstral Alanysis

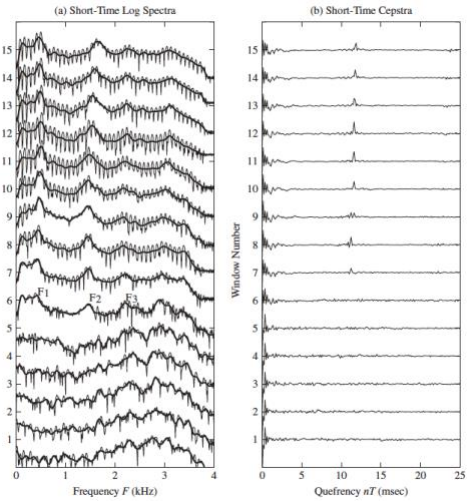


Figure 32 - Vowel ST-Cepstra tracking example

The approximation of sustain regions in vowel emission, for cepstral alanysis, is possible thanks to appropriate window-framing (also called *liftering* in cepstral terminology). Figure 31 shows a block-chain of *Short-Time Cepstral Alanysis* of vowel diphthong emission.

Linear Predictive Analysis (LPA) is maybe the most common and powerful speech analysis technique, used in spectral smoothing and formant frequencies estimation.

The idea is that a speech signal sample can be approximated by a linear combination of p previous speech samples. By locally minimizing the sum of the squared differences (MSE) between the actual speech samples and the linearly predicted samples, a unique set of predictor coefficients can be determined.

Linear prediction has historically [59] been used in speech coding applications, hence the term *Linear Predictive Coding* (LPC) [60]. As LPA

methods became more widely used in speech analysis the term "LPC" persisted, and now it is used as a comprehensive synonym of LPA techniques in general. Although multiple LPA methods exist, we will focus on a specific one: the *autocorrelation formulation* [61].

As we can assume from the introductory model (Figure 16) the input and resonant spectral composition can be represented by a time-varying digital filter, whose steady-state system function is:

$$H(z) = \frac{S(z)}{GU(z)} = \frac{S(z)}{E(z)} = \frac{1}{1 - \sum_{k=1}^{p} a_k z^{-k}}$$

where $H(z)$ is *the vocal tract system function*, even though it represents not only the effects of the vocal tract resonances but also the effects of radiation at the lips and, in the case of voiced speech, the spectral effects of the glottal pulse shape. G is the overall vocal *gain* factor. Although this seems to be a reasonable way of representing only non-nasal voiced sounds, if digital filter order p is high enough, this function provides a good approximation to any kind of speech sound.
In this system, the model speech samples $s[n]$ are related to the excitation signal by:

$$s[n] = \sum_{k=1}^{p} a_k s[n-k] + Gu[n]$$

with $u[n]$ the glottal excitation and a_k the effective coefficients of the filter. A linear predictor can therefore be defined as a system whose input is $s[n]$ and whose output is an approximation signal $\tilde{s}[n]$:

$$\tilde{s}[n] = \sum_{k=1}^{p} \alpha_k s[n-k]$$

with α_k the *prediction coefficients*. The prediction error is defined as the difference between the predicted signal and the original:

$$e[n] = s[n] - \tilde{s}[n] = s[n] - \sum_{k=1}^{p} \alpha_k s[n-k]$$

and thus, the prediction error sequence is the output of a system whose input is $s[n]$ and whose characteristic function is:

$$A(z) = \frac{E(z)}{S(z)} = 1 - \sum_{k=1}^{p} \alpha_k z^{-k}$$

also called the *prediction error polynomial* (or simply the *LPC polynomial*). The prediction error filter can thus be an inverse filter for *the vocal tract* (or LPC) *model* system $H(z)$:

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 - \sum_{k=1}^{p} \alpha_k z^{-k}}$$

Due to the time-varying nature of the speech signal, the analysis must be performed by a short-time framing, based on finding the set of predictor coefficients that minimizes the *mean-squared prediction error* over a short segment of the speech waveform. The resulting parameters are then *assumed* to be the parameters of the resulting system function $H(z)$.

The *short-time total squared prediction error* is thus defined:

$$\varepsilon_n = \sum_{m=0}^{M} e_n^2[m] = \sum_{m=0}^{M} (s_n[m] - \tilde{s}_n[m])^2 = \sum_{m=0}^{M} \left(s_n[m] - \sum_{k=1}^{p} \alpha_k s_n[m-k] \right)^2$$

where $s_n[m]$ is a fragment of speech signal, selected to be centered around the sample n. To find the values that minimize ε_n, we can set $\frac{\partial \varepsilon_n}{\partial \alpha_k} = 0$ with $k = 1, 2, \dots, p$, obtaining the equation:

$$\sum_{m=0}^{M} s_n[m-i]\tilde{s}_n[m] = \sum_{k=1}^{p} \hat{\alpha}_k \sum_{m=0}^{M} s_n[m-i]s_n[m-k], \quad 1 < i < p$$

where $\hat{\alpha}_k$ are values of α_k that minimize the ε_n.

To establish an appropriate value for M we can assume to take a waveform segment $s_n[m]$ that is null outside an interval $[0, L-1]$, where L is the effective length of a windowing function (i.e.: *Hamming window*) [62]. As can be seen in Figure 33, the prediction error is likely to be relatively large at the beginning of the interval $(0 \le m \le p-1)$ because the predictor has to predict the (non-zero) signal samples from previous samples that have been arbitrarily set to zero as a result of windowing. Likewise, the error is likely to be relatively large at the end of the interval $(L \le m \le L-1+p)$ because the predictor must predict p zero-valued samples outside the window, from samples that are non-zero and inside the window:

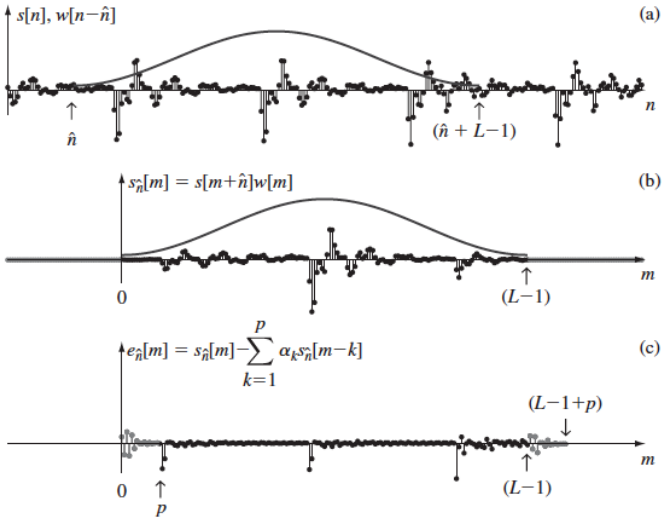


Figure 33 - LPA *Hamming* windowing

$$\varphi_n[i,k] = \sum_{m=0}^{L-1+p} s_n[m-i]s_n[m-k]$$

$$= \sum_{m=0}^{L-1+(i-k)} s_n[m-i]s_n[m+(i-k)], \quad \begin{cases} 1 \le i \le p \\ 0 \le k \le p \end{cases}$$

In this case the signal prediction is identical to the short-time autocorrelation function computation (hence the *autocorrelation method* name), and thus the prediction MSE:

$$\varepsilon_n = AC_n[0] - \sum_{k=1}^{p} \alpha_k AC_n[k]$$

All the auto-correlation values can then be expressed with vector notation as:

$$\begin{bmatrix} AC_n[0] & AC_n[1] & \cdots & AC_n[p-1] \\ AC_n[1] & AC_n[0] & \cdots & AC_n[p-2] \\ \vdots & \vdots & & \vdots \\ AC_n[p-1] & AC_n[p-2] & \cdots & AC_n[0] \end{bmatrix} \times \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_p \end{bmatrix} = \begin{bmatrix} AC_n[1] \\ AC_n[2] \\ \vdots \\ AC_n[p] \end{bmatrix}$$

with the resulting *Toeplitz* matrix, symmetric being positive definite, symmetric and with all the values on a chosen diagonal being equal. By exploiting the nature of the coefficient matrix, several efficient recursive methods have been developed to solve this system of equations. The most popular and well-known is the *Levinson–Durbin* recursive algorithm [63] [64] presented here in pseudocode notation.

```
def Levinson-Durbin_algorithm
    input_1: p                                  # Filter order (Coeff. num.)
    ε^(0) = AC[0]                               # 1st segment Autocorrelation
    for i = 1, ..., p do

        k_i = (AC[i] − Σ_{j=1}^{i−1} α_j^(i−1) AC[i−j]) / ε^(i−1)   # i-PARCOR coefficient computation
        α_i^(i) = k_i                           # actual scale coefficient assign.

        if i > 1 then                           # Row coefficients update
            for j = 1, ..., i-1 do
                α_i^(i) = α_j^(i−1) − k_i α_{i−j}^(i−1)
            end for
            ε^(i) = (1 − k_i^2) ε^(i−1)         # Compute actual pred. error
        end if
    a_j = a_j^(p)                               # Store coefficient
    end for
```

In the **STATE-OF-THE-ART** chapter we will see a different approach to solving LPC computation with a single-layer *neural network* implementation.

The most useful application of LPA to speech is in the extraction of spectra for formants estimation. In fact, applying the *short-time discrete Fourier transform* (STFT) model, we have:

$$|S_r(k)| = \left| \sum_{m=0}^{L-1} s[rR+m] w[m] \, e^{-j\left(\frac{2\pi}{N}\right)km} \right|$$

where the m-esimal magnitude STFT frame of the predicted signal $S_r(k)$ is computed at the set of times $t_r = rRT$ and the set of frequencies $F_k = kSR/N$, where R is the time shift (in samples) between adjacent frames, T is the sampling period, $SR = 1/T$ is the sampling rate (in Hz) and N is the size of the DFT.

Figure 34 shows an example of the p-order incidence over spectral resolution quality (against a standard short-time DFT spectrum) for a vowel utterance.

Comparison of results for the same STFT and LPA-STFT (Figure 35) parameters set, with low-order LPC approximation (p=12), shows finer energy bands due to the short-time approximation windowing, which is more useful for effective formants tracking (due to robust rejection of harmonic emphasis).

In Figure 36 a DFT spectra comparison between (a) a narrow-band standard DFT, (b) a wide-band standard DFT, (c) a cepstral alanysis derived spectrum and (d) an LPA derived spectrum.

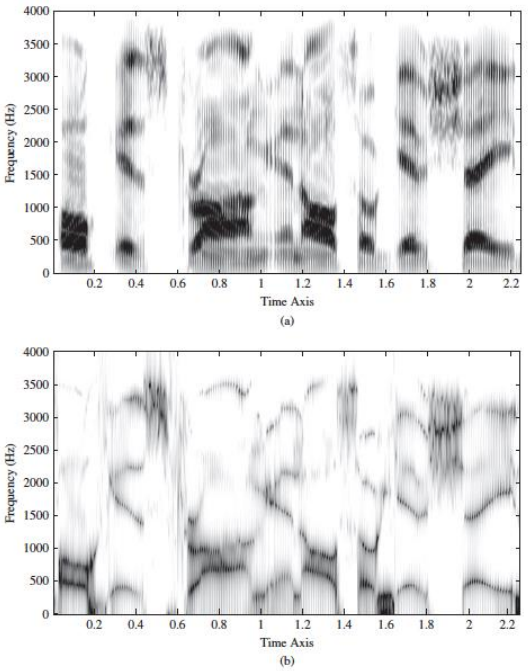


Figure 36 - STFT vs LPA spectral estimates

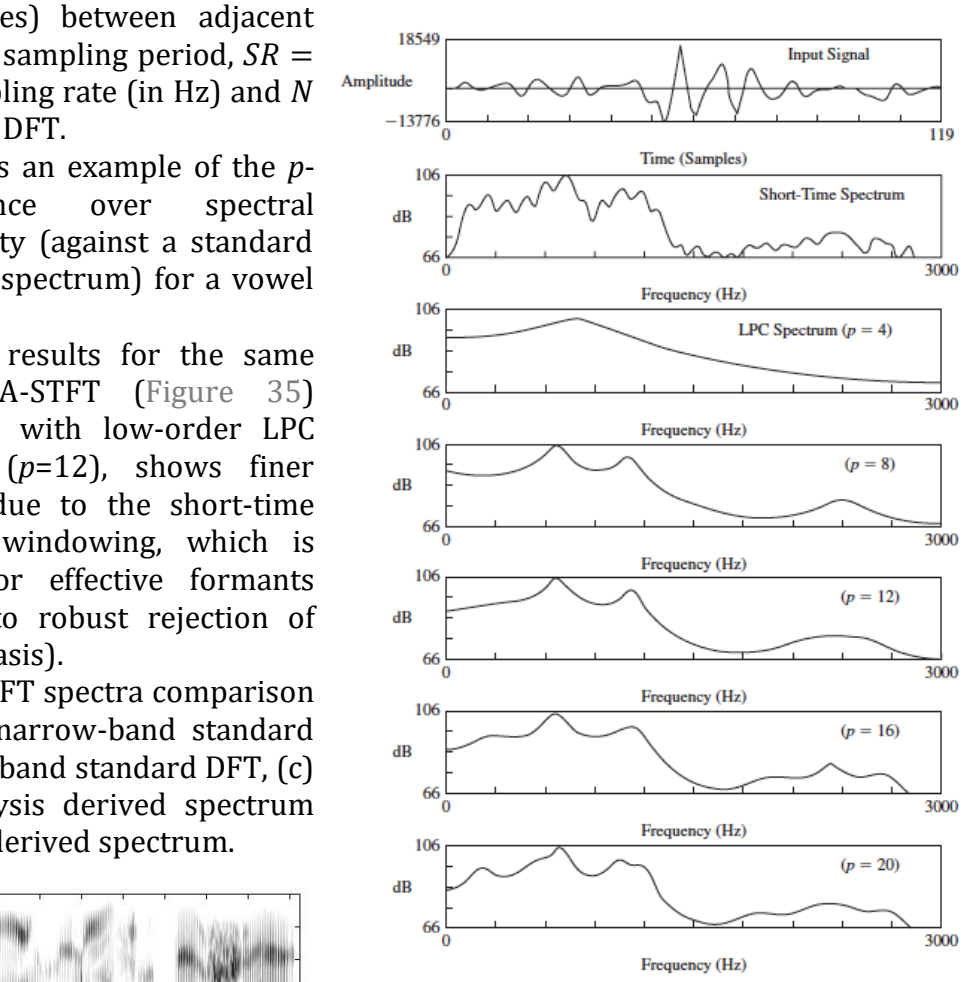


Figure 35 - LPA approximation (p) order incidence

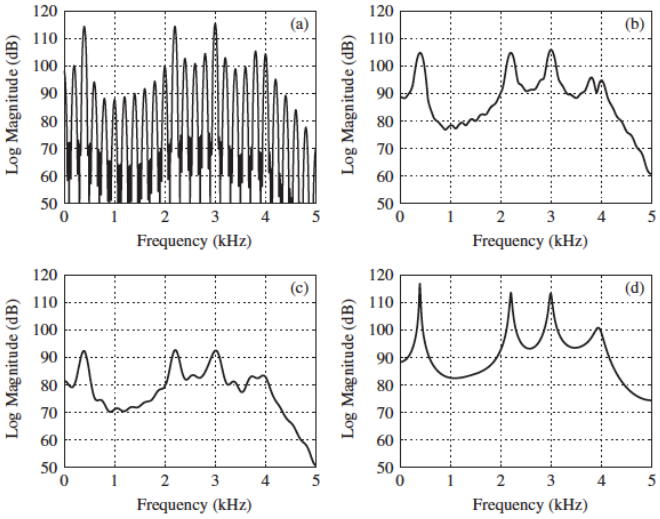


Figure 34 - (a) DTFT hi-res, (b) DTFT smooth, (c) Cepstral Alanysis, (d) LPA estimate

Deep Learning

Machine Learning (ML) can be defined as the field of Artificial Intelligence (AI) dedicated to providing performant and viable ways to approach complex real-world problems, like *pattern-matching*, *classification* or *regression*.
Thanks to its statistical foundation, ML provides solutions to allow computers to learn from "data experiencing" so as to infer numeric relationships in terms of hierarchies of concepts (feature abstraction). By gathering and processing experimental "knowledge" (*knowledge-based approach*), ML algorithms try to avoid the need for human operators to formally specify iterative sub-steps for subjective and intuition-based problems, which tends to be generally difficult to express in formal way [65] [66] [67].

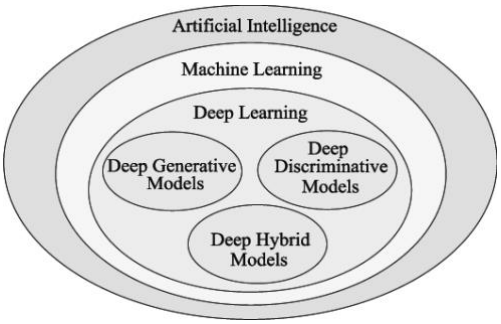


Figure 37 - AI sub-fields

Extracting patterns from raw data is a hard-task, so expertise is required in field-related data analysis and features extraction (i.e.: *specific factors of variation* disentangling).
One solution to this problem is to use ML algorithms itself to discover not only the desired mapping to output(s), but also the representation itself: this approach is known as *representation learning*. If we could take a look at these kinds of learning graphs, we will notice how deep and stratified they are: hence the *Deep Learning* (DL) [68] sub-field naming.

DL actually dates back to 1940s but often appears to be a novelty, because of its previous unpopularity and because it has gone through many different names.
Speaking in terms of technology ages, there have been three "waves" of DL development:

1) the *cybernetics phase*, 1940s – 1960s;
2) the *connectionism phase*, 1980s – 1990s;
3) the *deep learning resurgence*, 2006 – until now.

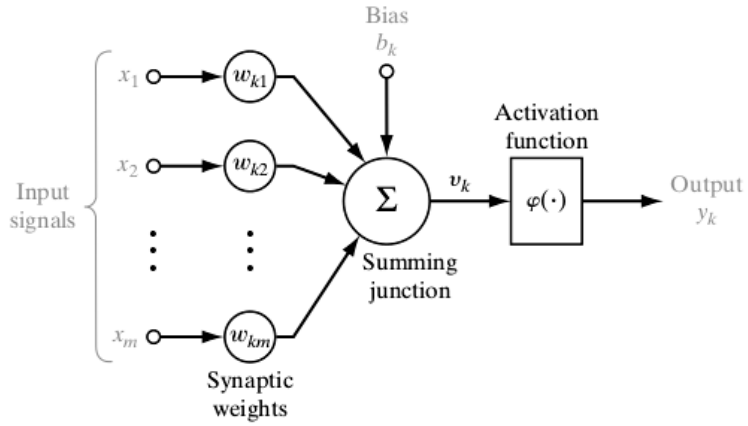


Figure 38 - Perceptron model

Because of some of the earliest DL algorithms we know today were intended to be computational models [69] that mimic biological learning, one of the names that DL has gone by is *artificial neural networks* (ANNs), with the corresponding erroneous expectation that those models are intended to act as biological brains, whereas they were not designed to do

that at all. They initially were simple linear models that take a set of n input values (x_1, …, x_m) and associate them with an output (y_k).

The *Perceptron* [70] [71] (also *artificial neuron* [72]) was the iconic result of computational neuroscience researches, conducted by Warren S. McCulloch (1898 – 1969) and Walter Pitts (1923 – 1969) during early '40s and can be considered the core-unit of modern DL neural networks architectures. Its mathematically defined as:

$$y_k = \varphi\langle x, \omega_k \rangle = x^T \omega_k = \varphi\left(\sum_{k=0}^{K} x_n \omega_k\right)$$

where x is the input features vector, ω_k a set of weights which serve as scaling coefficients at the summation entry (*net input*) and φ a nonlinear function (also *activation function*). x_0 is commonly set equal to $+1$ and its weight is brought out of the summation, signed as b_k (also *bias* or *intercept*). The original idea of McCulloch and Pitts was also called *Threshold Logic Unit* [73](TLU) because of the use of a *Heavy-Side* (or *step*) *function* as non-linear activation:

$$\varphi = \begin{cases} x & if\ x \geq \theta \\ 0 & elsewhere \end{cases}$$

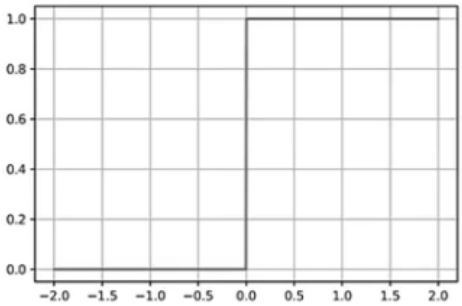
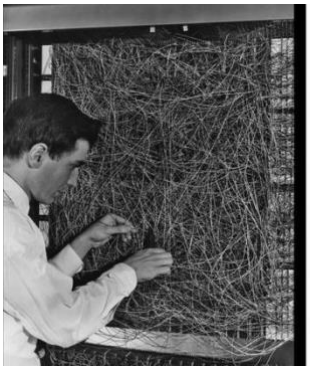


Figure 39 - Threshold unit function

(Perceptron activation function)



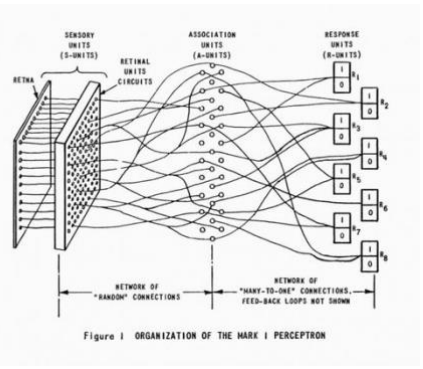


Figure 40 - F. Rosenblatt & Mark I - Perceptron (w. scheme)

where θ is an arbitrary threshold value. This linear model was capable of simple classification tasks (linear separable problems) in a 2D space.

In 1958 the psychologist Frank Rosenblatt (1928 – 1971) built the *Mark I Perceptron* [74] [75]: a machine for image recognition, with an array of 400 photocells connected to transistorized neurons, with weights encoded by potentiometers.

Learning phase was performable manually (changing pots states) or by means of electric motors.

The first major climax of DL coincided with previously mentioned disregarded expectations of "neural networks humanization" and with the 1969 Marvin L. Minsky (1927 – 2016) and Seymour A. Papert (1928 – 2016) critique: inside their publication *Perceptrons: an Introduction to Computational Geometry* [76] they enumerated lots of limitations for these models (being the "XOR" one the most famous).

In the 1980s, the second wave of neural network research arose with *connectionism* movement (also called *parallel distributed processing*): the core idea was that a large number of simple computational units could achieve a more effective and

intelligent behavior when networked together. The quintessential example of DL models finally raised to the surface: *Feedforward ANN*, or *Multi-Layer Perceptron* (MLP), that will be analyzed in more detail in the following paragraph.
Its feature abstraction capabilities were exponentially exaggerated: Perceptrons vertical layering (*shallow neural networks*) evolved in horizontal layering of computational units (*deep neural networks*) and weights adaptation became formally automatic and optimizable thanks to David Rumelhart (1942 – 2011) and Yan LeCun (1960 -) pioneering works in *back-propagation* algorithms [77] [78].
MLP can represent multivariate functions that could succeed in complex classification tasks by *hyperplane* separation of input features. Differentiate multivariate function is achievable by means of *gradient* computation (further insights in the *Back-Propagation* paragraph) and Rumelhart and LeCun works were just focused on scripting of gradient descent techniques, for errors minimization, learning scattering and coordinated weights adjustment.
Increasing computational complexity and the lack of commercial consistent results signed the second DL wave climax, which lately came back into fashion in the early 2000s, when affordable hardware performances increased, *big data*set became more easily obtainable and deep networks architectures became deployable with affordable commercial resources.

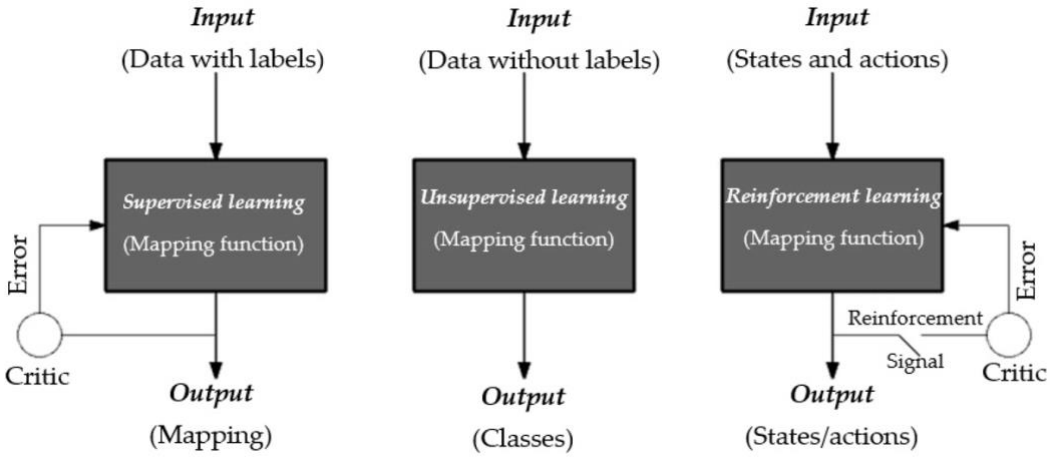


Figure 41 - ML/DL Learning types

For this work we are going to explore a specific DL/ML paradigm: *supervised learning*. In this context, network algorithms learn patterns from human pre-labeled data. Every sample pushes the algorithm to optimize its internal states, in order to output regression values/class probabilities which tend to be equal to provided *labels*. This field of DL is in charge of investigating ANNs defined as *Deep Discriminative Models*.

Now the MLP standard architecture will be introduced, directly contextualized in classification tasks. Learning techniques and optimization strategies will be then discussed in order to approach potentialities for real-time vowel phonemes recognition.

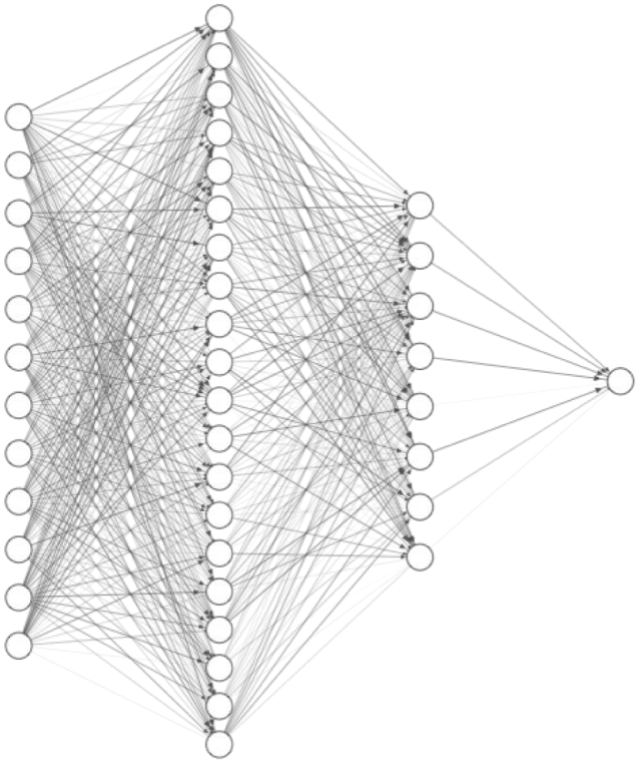


Figure 42 - MLP architecture <12/20/8/1>

MLPs are also called *feedforward neural network* models, because information flows (*forward propagation*) in a single direction through their *input layer*, to internal/intermediate layers (*hidden layers*) and finally to the *output layer*.
The model is associated with a *directed acyclic graph* (Figure 42) which describes how different functions are typically composed together within it. n functions $f^{(n)}$ are connected to form:

$$f(x) = f^{(n)}\left(f^{(n-1)} \dots \left(f^{(0)}(x)\right)\right)$$

where $f^{(0)}$ is the *input layer*, $f^{(n-1)}$ the 1st *hidden layer* and $f^{(n)}$ the *output layer*.
The number of layers is also referred to as the *depth* of the architecture, while dimensionality of hidden layers determines the *width* of the model. Rather than thinking of layers as representing a single vector-to-vector function, it's simpler and reasonable to generalize layers as consisting of many interconnected *units* that act in parallel, each representing a simple vector-to-scalar function.

One of the most common tasks of DL is *classification*, where computer programs are asked to specify which one of k categories (or *classes*) some input belongs to. To solve this task, the learning algorithm usually produce a function:

$$f(x) : \mathbb{R}^n \to \{0, \dots, k\}$$

that assign a discrete class ID to every sample x (deterministic classification). In real-world applications, discrete IDs assignment are usually substituted by probability belongings (probabilistic classification) and the output can then be considered a vector y_n (of length k) which contains normalized k-class probabilities listing for each input sample. This linear model is also called *logistic regression* [79] and is appealing because can be easily and efficiently fitted (either in closed form or with convex optimization) but it also has the obvious defect of approximation capability limited to linear functions.
To overcome approximation limits we can apply the previous linear transform to an *activated* input $g(x)$, where g is our *activation function*:

$$h_i = g(x_i^T W_i + b_i)$$

and h_i is the i-esimal *hidden node* (also *hidden feature*) of k-esimal layer, and W_i and b_i weights and biases respectively associated with x_i node. In modern neural networks $g(x)$ is often a *Rectified Linear Unit* (or ReLU), defined:

$$g(z) = \max\{0, z\}$$

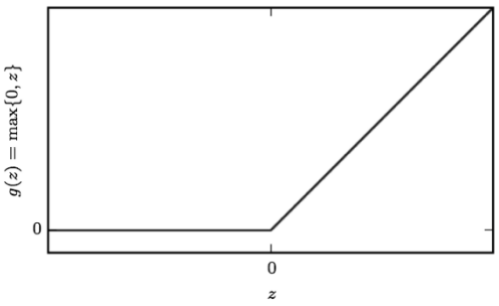


Figure 43 - ReLU activation function

turning k-layer formulation in:

$$f(x, W, b) = W_{i+1}^T max\{0, x_i^T W_i + b_i\} + b_{i+1}$$

where i is the layer index.

In supervised learning, unique labels (per sample) are given, in order to train the model and try to minimize its statistical prediction errors, expressed by a *cost function*. The model tries to define a categorical distribution $p(y|x, \theta)$, with θ the MLP parameters (weights and biases) applying the principle of *maximum likelihood*: this means we adopt measures of the *cross-entropy* [80] between dataset *labels* and model's *predictions*, as our cost function.

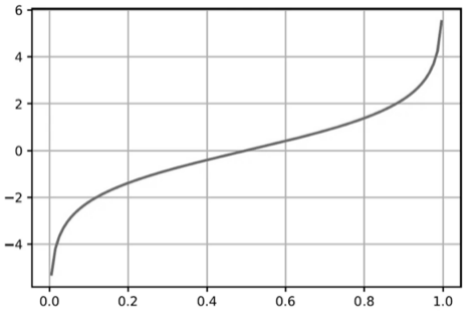


Figure 44 - Logit function (logistic distribution quantile)

In the context of binary classification, we compute class probability predictions $p(y = 1|x)$ setting a threshold value (typically 0.5) and then *dummy-evaluating* class with *booleans* 1 (if $p > threshold$) or 0 (if $p < threshold$). The basic assumption in logistic regression is that the *log-odds*[8] of the event that the sample belongs to the *True* class, is a linear combination of its features.

Logit (or *logistic unit*) [81] has been mathematically defined by Joseph Berkson (1899 – 1982) as the inverse of the *standard logistic function*:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

and so:

$$logit(p) = \sigma^{-1}(p) = \ln\left(\frac{p}{1 - p}\right)$$

This function is used to maps probability values defined in $[0, 1]$ to real numbers, in $(-\infty, +\infty)$. Assuming that log odds results from linear combination of the features, means that:

[8] Or also *logits*, logarithms of the odds ratio.

$$\ln\left(\frac{p_i}{1-p_i}\right) = h(w_0 x_0 + b_0) + \cdots + h(w_n x_n + b_n)$$

knowing that log-odds are expressed by means of logit equation, we can directly apply its inverse (the logistic function, also *sigmoid function*) on output dot-products, to convert results in probabilities regressions, expressed between $[0, 1]$.

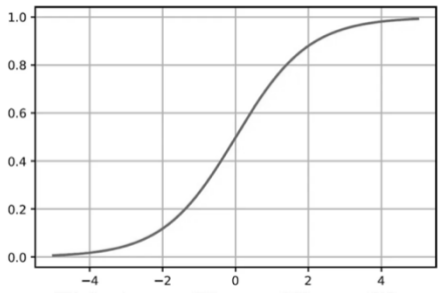


Figure 45 - Sigmoid function

The *loss function* will measure i-esimal prediction deviation from the provided *ground truth* (y_i) of a specific sample, differently from *cost functions* that measure model error rates averaging across entire feature sets:

$$\mathcal{J}(f(x), D) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(y_i, p_i)$$

where $\mathcal{J}$ is the cost function for a specific model $f(x)$ and an entire *dataset* of features, while the loss function ($\mathcal{L}$) for logistic regression case, is usually a *log loss* (logistic loss):

$$\mathcal{L}(y_i, p_i) = -y_i ln(p_i) - (1 - y_i)\ln(1 - p_i)$$

this expression is formally better known as the *cross entropy* of the predicted distribution from the actual: the total loss is then the overall *negative log-likelihood*, assuming that data labels are *Bernoulli distributed* [82].
In fact, Bernoulli distributions with positive class parameter p, are simply described as:

$$P(y = 1|p) = p$$

and so:

$$P(y = 0|p) = 1 - p$$

in a compact form:

$$P(y|p) = p^y (1 - p)^{1-y}$$

Then the log-likelihood of the outcomes, given the class probability is:

$$Log\big(P(y|p)\big) = yLog(p) + (1 - y)Log(1 - p)$$

and $\mathcal{L}(y_i, p_i)$ is proved to be the negative of this equation. The cost function can be than re-written as:

$$\mathcal{J}(f(x), D) = -\frac{1}{n}\sum_{i=1}^{n} [y_i Log(p_i) + (1 - y_i)Log(1 - p_i)]$$

and it is convex and has a unique global minimum. However, there is no closed-form solution for finding the optimal weights parameters to minimize it, due to the non-linearities introduced by the log function. Therefore, we need to use iterative and derivative-based optimization methods (such as *gradient descent*) to find the minimum

(further analysis in *Loss Minimization & Backpropagation* paragraph). In general, neural networks training algorithms do not to reach exactly a global minimum of the cost function, but under certain conditions, they can get close to a *local minimum*.

A ReLU activation function outputs zero across half its domain and this makes network nodes derivatives remain "large" whenever that node is active. The gradients are not only large but also consistent. We have seen how ReLUs are typically used on top of an affine transformation, like:

$$h = ReLU(x^T W + b)$$

this configuration is also called a *rectifier unit*, and when initializing the parameters for these units, it can be a good practice to set all *biases* to a small positive value (such as 0.01) to guarantee a gradient firing by each of them, especially if little is known about the input dataset. Otherwise you can force the hyperplane to start from the origin, placing *biases* to 0 (recommended only for positive-features sets).

Kaiming He [83] and colleagues, provided an efficient way to combine 0-biases with appropriate weights initializations, so as to minimize the risk of rising input feature magnitudes to exponential values during backpropagation, standardizing rectifier layers variance:

$$\frac{1}{2} n_l Var[w_l] = 1$$

where n_l is the number of nodes in a specific layer. To satisfy this condition they found that:

$$w_l = \mathcal{N}\left(0, \frac{2}{n_l}\right)$$

with $\mathcal{N}$ a 0-centered *gaussian distribution*, with a standard deviation of $\sqrt{2/n_l}$. This method drastically improves and speeds up *rectifiers* minimum cost-error convergence [84] [85].

A *Universal Approximation Theorem* [86] [87] [88] states that a feedforward network with a linear output layer and at least one hidden layer can approximate any *Borel* measurable function, from one finite-dimensional space to another, provided that the network is enough wide. Any continuous function on a closed and bounded subset of $\mathbb{R}^n$ is *Borel* measurable and therefore may be approximated by a NN, but we are not guaranteed that the training algorithm will be able to learn that function.
According to the previous theorem there exists a network large enough to achieve any degree of accuracy we need, but the theorem does not say how large this network must be [89].
Some references [90] provide bounds on the architectural size of a single layer network needed to approximate a broad class of functions. Unfortunately, in the worst case, an exponential number of hidden units may be required: one hidden unit for each input configuration to recognize. In the case of binary vectors of dimension n, $v \in \{0,1\}^n$, we would have a total amount of 2^{2n} possible binary functions and selecting one, would require 2^n bits and thereby $O(2^n)$ degrees of freedom.

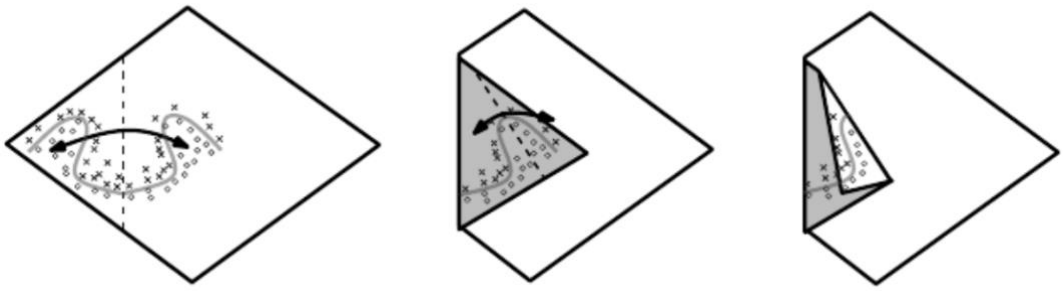


Figure 46 - MLP learning (geometric interpretation)

In Figure 46 is geometrically explained another main theorem about networks depth [91] that states that the number of linear regions carved out by a deep rectifier with d inputs, depth = l and width = n is equal to:

$$O\left(\binom{n}{d}^{d(l-1)} n^d\right)$$

this means that a sufficiently deep layering of rectifiers could approximate any complex symmetry of the required parameters function.

Backpropagation & Gradient Descent

The backpropagation algorithm [77] (often abbreviated *backprop*) allows the information accumulated through re-iterated training steps, to be propagated from cost evaluation to intermediate gradient computing, in order to better tune model parameters and minimize subsequent estimated errors.
This term is often used to refer to the whole learning algorithm, while actually backprop seems to represent only the method for computing the gradient, while other algorithms [such as *Stochastic Gradient Descent* (SGD)] are used to optimize network parameters using this gradient. In this case we are talking about the gradient of the cost function:

$$\nabla_\theta \mathcal{J}(\theta)$$

and from Gottfried W. Leibniz [92] (1646 -1716) studies about *chain rule of calculus*, if $x \in \mathbb{R}^m$, $y \in \mathbb{R}^n$, g is a function that maps from $\mathbb{R}^m$ to $\mathbb{R}^n$ and f maps from $\mathbb{R}^n$ to $\mathbb{R}$; if $y = g(x)$ and $z = f(y)$, then:

$$\frac{\partial z}{\partial x_i} = \sum_{j,i=0}^{+\infty} \frac{\partial z}{\partial y_j} \frac{\partial y_j}{\partial x_i}$$

in vector notation:

$$\nabla_x z = \left(\frac{\partial y}{\partial x}\right)^T \nabla_y z$$

with $\frac{\partial y}{\partial x}$ is the $n \times m$ *Jacobian* matrix of g.
This means that a variable gradient computation can be obtained by the above simple product and doing backprop consists of performing such products for each operation (function) in a pre-computed network graph [93].

The following pseudo-code algorithm illustrates *forward propagation*, that maps parameters to the supervised loss $\mathcal{L}(\hat{y}, y)$ associated with a single (input, target) training example (x, y). To obtain an effective total cost function $\mathcal{J}$, the loss must be summed to a *regularizing* coefficient ($\Omega(\theta)$ below), where θ is the model parameters matrix.

```
def forward_propagation_algorithm
    input_1: l,                          # Depth of the model
    input_2: W_i,  i ∈ {1, ..., l}       # Weight matrices of the model
    input_3: b_i,  i ∈ {1, ..., l}       # Bias parameters of the model
    input_4: x                           # Input sample (feature)
    input_5: y                           # Target output
        h_0 = x                          # Input layer pass
        for k = (1, ..., l) do           # Hidden layers pass loop
            a_k = b_k + W_k * h_k-1      # Net pass
            h_k = f(a_k)                 # Non linearity pass
        end for

        y_hat = h_(k = l)                # Predictions computation
        J = L(y_hat, y) + λΩ(θ)          # Cost (w. Regularization)
```

Now the backprop algorithm, referring to the previous *forward-prop*. This computation yields the gradients of the cost function for each layer `k`, starting from the output layer and going backwards to the first hidden one. These gradients can be interpreted as an indication of how each layer's output should change to reduce error (and thus how weights and biases must be updated).

def *backward_propagation_algorithm*

```
g ← ∇ŷJ = ∇ŷL(ŷ, y) + λΩ(θ)          # Output layer gradient
for k = (l, ..., 1) do               # Backward Gradient loop
    g ← ∇a_k J = g ∗ f′(a_k)         # Gradient to pre non-linearity
    ∇b_k J = g + λ∇b_k Ω(θ)      # Biases Gradient
    ∇W_k J = g h_{k-1}^T + λ∇W_k Ω(θ)    # Weights Gradient

    g ← ∇h_{k−1} J = W_k^T ∗ g       # Propagate Grad. to previous layer
end for
```

Optimization algorithms that use the entire available dataset to compute gradients are called *batch* or *deterministic gradient* methods, while algorithms that use only a single example at a time are called *stochastic* or *online gradient* methods. Most algorithms implied in DL use subsets of training examples: these are usually called *mini-batch* (or *mini-batch stochastic*) methods.

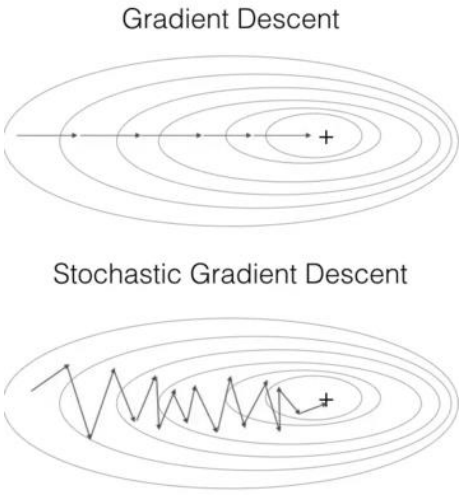


Figure 47 - Batch vs Stochastic Gradient Descent

Consistent batches generally offer regularizing effects [94] due to noise injection inside the learning process, but training with very small batch size often requires small *learning rates* to maintain stability, because of the high variance in the gradient estimate. The total learning runtime can rapidly increase because of more steps needing to observe the entire dataset, and for effective minimum reaching. Some kinds of hardware, especially GPUs, can grant high performance with power of 2 batch sizes: typical *mini-batch* size range from 16 to 256.

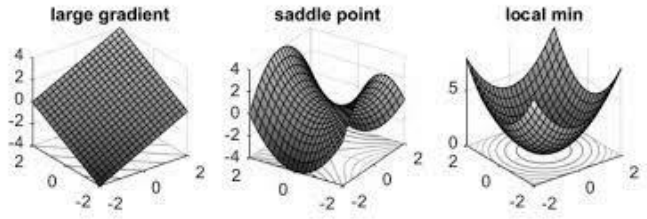


Figure 49 - Gradient's critical conditions

To compute unbiased gradient estimates it's crucial that mini-batches are selected randomly: for this reason, batch balancing and samples independence must be acquired in the dataset building phase.

Stochastic noise injection helps to reduce a non-trivial condition with high-dimensional non-convex functions (common in DL), where local minima turn out to be few compared to other zero-gradient conditions: *saddle-points*. Locations around a saddle point have greater cost than the saddle point itself, while others have a

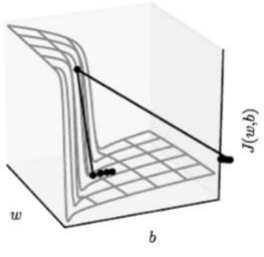


Figure 48 - Exploding gradient

lower cost, therefore it can be easily traded for a function minimum.
The opposite case is represented by *vanishing* and *exploding* gradients, where plane or cliff regions of high-dimensional function can enforce non-learning or extreme instability (negative jumps) due to gradient irregularities: a possible solution for those cases could be *gradient clip*, at fixed τ value.

Researchers have long realized that the *learning rate* (also LR) is one of the most difficult to set hyper-parameters, because it significantly affects any model performance. It represents a gradient scaling coefficient that defines how big the minimization error step will be. Standard SGD adopts constant LR values, but improved optimizers were developed to adapt the LR to gradient and parameters history.
The first was the *AdaGrad* [95] optimizer, that individually adapts LRs of all model parameters by scaling them inversely proportional to the square root of the sum of all the historical squared values of the gradient. Parameters with the largest partial derivative of the loss, have a correspondingly rapid decrease in their LRs while parameters with small partial derivatives, have a relatively small decrease.

The *RMSProp* algorithm [96] modifies AdaGrad to better perform in the non-convex setting by changing the gradient accumulation into an exponentially weighted moving average, so as to discard history from the extreme past and promote rapid convergence when a new convex bowl region is found.

def *RMSProp_algorithm*

input_1: ϵ # Global LR
input_2: ρ # LR Decay Rate
input_3: θ # Initial model parameters
input_4: δ = 10^{-6} # Division stabilizer
input_5: $(\{x_1, \dots, x_m\}, \{y_1, \dots, y_m\})$ # Mini-batch features & targets

r = 0 # Accumulation auxiliary variable
while criterion **do** # Criterion-based Training loop
g $\leftarrow \frac{1}{m} \nabla_\theta \sum_{i=0}^{m} L(f(x_i, \theta), y_i)$ # Mini-batch Gradient computation
r $\leftarrow \rho$r + (1 − ρ)g * g # Squared gradient (w. decay)

$\Delta\theta = -\frac{\epsilon}{\sqrt{\delta + r}}$ * g # Parameters variation
$\theta \leftarrow \theta + \Delta\theta$ # Update model parameters
end while

Adam [97] is another adaptive LR optimizer algorithm, whose name derives from "*adaptive moments*". In the context of the earlier algorithms, it can be seen as a peculiar combination of RMSProp and the physical concept of *momentum*. In Adam, momentum is incorporated directly as an estimate of the *first-order moment* of the gradient (with exponential weighting) and the algorithm includes bias corrections to the estimates of both the first-order (the *momentum term*) and the (*uncentered*) second-order moments.

def *Adam_algorithm*

input_1: ϵ = 0.001 # Global LR
input_2: ρ_1 = 0.9; ρ_2 = 0.999 # LR moments Decay Rate
input_3: θ # Initial model parameters
input_4: δ = 10^{-8} # Division stabilizer
input_5: $(\{x_1, \dots, x_m\}, \{y_1, \dots, y_m\})$ # Mini-batch features & targets

s = 0 # 1st Moment auxiliary accumulator
r = 0 # 2nd Moment auxiliary accumulator
t = 0 # Time step initialization
while criterion **do** # Criterion-based Training loop
g $\leftarrow \frac{1}{m} \nabla_\theta \sum_{i=0}^{m} L(f(x_i, \theta), y_i)$ # Mini-batch Gradient computation
t $\leftarrow$ t + 1 # Time step increase

s $\leftarrow \rho_1$s + (1 − ρ_1)g # Biased 1st moment est. update
r $\leftarrow \rho_2$r + (1 − ρ_2)g · g # Biased 2nd moment est. update
$\hat{s} \leftarrow \frac{s}{1-\rho_1^t}$ # 1st moment Bias Correction
$\hat{r} \leftarrow \frac{r}{1-\rho_2^t}$ # 2nd moment Bias Correction

$\Delta\theta = -\epsilon \frac{\hat{s}}{\sqrt{\hat{r}}+\delta}$ # Parameters variation
$\theta \leftarrow \theta + \Delta\theta$ # Update model parameters
end while

In this work we are going to test both RMSProp and Adam *optimizer* algorithms due to the lack of literature guidelines.

Regularization

Many strategies were deployed to reduce the *test error* of a model and improve its generalization, possibly at the expense of increasing *training error*. These strategies are commonly known as *regularization techniques*.

As introduced in the previous paragraph, many regularization approaches are based on limiting the model mnemonic capability by adding parameters norm penalty $\Omega(\theta)$ to the cost function, turning it into an *objective function*:

$$\tilde{J}(\theta;\, X, y) = J(\theta;\, X, y) + \lambda\Omega(\theta)$$

where $\lambda \in [0, \infty)$ is a hyper-parameter that scales relative contribution of norm penalty term Ω, usually applied only to the weights of a layer, leaving biases unregularized: in fact, biases typically require less data than weights to fit accurately.
Each weight specifies how nodes interact with each other and fitting weights well requires observing features in a variety of conditions. Each bias controls only a single node: this means that not too much variance is induced by leaving the biases unregularized. For these reasons θ will be substituted by W (a matrix of all regularizable parameters) in the following formulas.

Although many regularization techniques (and respective variants) are nowadays available, here we introduce only those effectively implemented for this research: *DropOut, L^2-Norm* and *Batch-Norm*.

The *L^2-Norm* penalty, commonly known as *weight decay*, is a regularization strategy which drives weights closer to the origin. Generally speaking, we could regularize parameters to be near any specific point in space, but better results will be obtained for a value closer to the effective one, with 0 being a default decision which makes sense when the effective value is unknown. The added penalty term is:

$$\Omega(\theta) = \frac{1}{2}\|w\|_2^2$$

In statistics, L^2 regularization is also known as *ridge regression* or *Tikhonov regularization*, after Andrey Nikholayevich Tikhonov (1906 – 1993) [98] [99] research. The objective function thus becomes:

$$\tilde{J}(W;\, X, y) = J(W;\, X, y) + \frac{\lambda}{2}W^T W$$

and parameter gradient:

$$\nabla_W \tilde{J}(W;\, X, y) = \lambda W + \nabla_W J(W;\, X, y)$$

a single regularization gradient step will be then:

$$W \leftarrow (1 - \epsilon\lambda)W - \epsilon\nabla_W J(W;\, X, y)$$

DropOut [100] provides a computationally inexpensive but powerful method of regularization. This method can be thought of as making *bagging* practical for ensembles of very large neural networks.

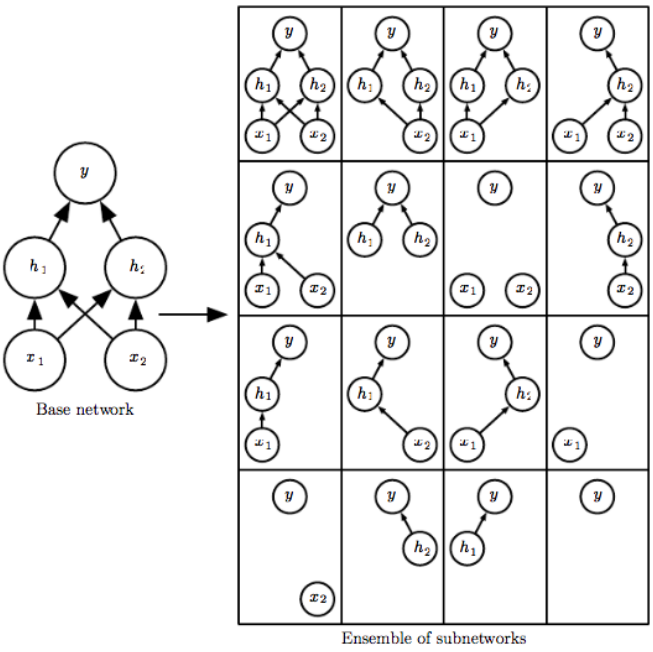


Figure 50 - DropOut subnets

Bagging involves training multiple models and evaluating them on each test example, which seems impractical when each model is represented by a large MLP and thus costs in terms of runtime and memory are really high.
In the context of a single MLP, DropOut trains the ensemble consisting of all sub-networks that can be realized by removing non-output units from an underlying base network: effectively achievable by multiplying their output value by 0.
In the example in Figure 50, a large proportion of the resulting networks have no input units or connections between inputs and outputs. This problem is proved to become insignificant for networks with wider layers, where the probability of dropping all possible paths from inputs to outputs, becomes smaller.
A common decision is represented by choosing two different *drop rates* for input layer (usually 0.8) and hidden layers (0.5 or more), establishing variable degrees of dependence between concrete dataset features and abstract ones.

Each sub-model defined by a so-called *masking vector* (μ) provides a specific sub-probability distribution $p(y \mid x, \mu)$. The resulting model probability will be therefore *inferred* applying arithmetic mean over all masks:

$$\sum_{\mu=0}^{n} p(\mu)p(y \mid x, \mu)$$

where $p(\mu)$ is the probability distribution used to sample μ during the training phase.

DropOut Training is not quite the same as *Bagging Training*: in the case of bagging, the models are all independent, while in dropout the models share parameters, with each model inheriting a different subset of parameters from the parent neural network.
This parameter sharing makes it possible to represent an exponential number of models with a condensed amount of memory. In the case of bagging, each model is trained to converge on its respective training set. In the case of dropout, typically most models are not explicitly trained at all, instead a tiny fraction of the possible sub-networks could be trained for just a single step and on the same training batch, and parameter sharing grants the remaining sub-networks to arrive at a good settings update too.

DropOut is computationally cheap: it requires only $O(n)$ computations per example per update, to generate n random binary values; depending on the implementation, it requires $O(n)$ memory too, to store those numbers until the backprop phase.

Batch normalization (also *Batch-Norm*) is a method [101] of adaptive model re-parametrization, motivated by the difficulty of training very deep models which involve composition of many functional layers. Actually, the gradient tells how to update each parameter under the assumption that the other layers do not change. In practice instead, when performing updates, unexpected results (*covariance shifting*) can happen due to many functions composed together that are changed simultaneously. The re-parametrization can be applied to any input or hidden layer in a network.

Let H be a mini-batch of activations of the layer to normalize, arranged as a design matrix with the activations (per example) appearing in the same row. To normalize H we apply *standardization* (or *Z-score*):

$$H' = \frac{H - \mu}{\sigma}$$

where μ is the vector containing the mean values and σ contains standard deviations (per unit). At training time:

$$\mu = \frac{1}{m}\sum_{i=0}^{m} H_i$$

and:

$$\sigma = \sqrt{\delta + \frac{1}{m}\sum_{i=0}^{m}(H - \mu)_i^2}$$

where δ is a small positive value (such as 10^{-8}) so as to avoid undefined values for the gradient (for $\sqrt{0}$). At test time, μ and σ can be replaced by running averages collected during training.

In practice, to maintain the informative expressive power of a specific network layer, it's common to replace the batch of hidden unit activations H with $(\gamma H' + \beta)$ where γ and β are learnable parameters that allow new layers to recover any required mean and/or standard deviation. Reference [101] suggests applying standardization to transformed layers (post non-linearity) only.

A special account should be kept for the *Early Stopping* technique.
When training large models with sufficient representational capability to *overfit* the task, we often observe that *training error* decreases steadily over time while *validation set error* (also *test error*) begins to increase (Figure 51).

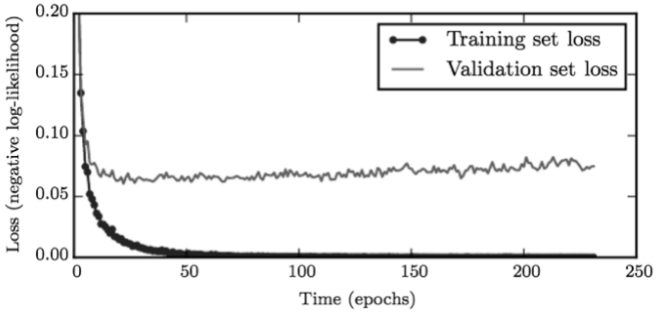


Figure 51 - Overfitting losses example

This means that a better validation model can be obtained by returning the parameter settings at the point in time (*epochs*) where the lowest validation set error occurred.
There are at least two viable ways to implement it:

1- Train the model until loss function is sufficiently minimized and store parameters configuration at each epoch, so as to return the best validation setting after training completion;
2- Train model in an infinite loop, imposing (multiples) exit conditions.

In the 1st case an accelerated multiple hardware set-up is preferable, so as to perform independent training and parameters storing while the 2nd implies a possible unfinished training, due to exceptional (x, y) features conditions, still resulting in a more affordable implementation (especially on unique CPUs).
A pseudocode example of the 1st case implementation is provided, while the 2nd can be reviewed directly in the external **APPENDIX-A (Code)**.

```
def Early_Stopping_meta-algorithm

      input_1: n                              # Learning Steps before evaluation
      input_2: p                              # "Patience" factor
      input_3: θ_0                            # Initial model parameters
      input_4: ∇alg                           # Desired Backprop/Learn algorithm

            θ = θ_0                           # Actual parameters variable
            i = 0                             # Training steps accumulator
            j = 0                             # Patience steps accumulator
            v = ∞                             # Validation Error initialization
            θ* = θ                            # Best Parameters memory
            i* = i                            # Best Train. step memory
            while j < p do                    # Patience-based Training loop
                  θ ← ∇alg(n, θ)              # Update parameters ("n" steps)
                  i ← i + n                   # Time step increase
                  v' = Valid_Error(θ)         # Validation Error computation
                  if v' < v then
                        j ← 0                 # Null patience
                        v ← v'                # Update Validation Error
                        θ* ← θ                # Update Best Parameters
                        i* ← i                # Update Best Training step
                  i* = i
                  else
                        j ← j + 1             # Increment Patience variable
                  end if
            end while
```

Early Stopping is both an unobtrusive form of training and regularization because it requires almost no changes in the underlying training algorithm or the objective function. This means that it is easy to use early stopping without damaging the learning dynamics, in contrast to other regularization techniques which could trap the model in undesired local minima.

One-Class-One-Network & Multiple-Categories Classification

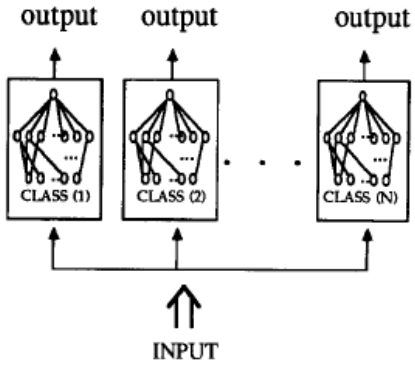


Figure 52 - The OCON model

In the context of computational parallel distributing era, the *One-Class-One-Network* [102] was developed to overcome the drawbacks of conventional backpropagation architectures, which needed to be completely re-trained whenever a class was added or removed.
This design can be described as a special case of architectural *ensembling*, because the problem of multiple outputs classification (also *softmax classification*) [103] is addressed by distribution over multiple independent subnetworks, each of which is focused on a much simpler binary classification task and expresses its unique class probability referred to a common input sample, concurring in a secondary complex multi-class output verdict.

Softmax classifiers represent the intuitive generalization of binary classifiers to multiple categorization tasks, where the training set is configured as $[(x_1, y_1), \dots, (x_n, y_n)]$, and $y_i \in [1, \dots, K]$ classes.
Under this condition, we can estimate the probability vector that a sample x belongs to each of K classes, by means of a unique NN architecture, implementing:

$$\begin{bmatrix} P(y=1|x) \\ \vdots \\ P(y=K|x) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} e^{(\theta_j^T x)}} \begin{bmatrix} e^{(\theta_1^T x)} \\ \vdots \\ e^{(\theta_K^T x)} \end{bmatrix}$$

The output is appropriately normalized so that the sum of all probabilities is 1. Similarly, the cost function will be a generalization of the *binary cross entropy*, also called *categorical cross entropy*:

$$\mathcal{J}(\theta) = -\left[\sum_{i=1}^{n} \sum_{k=1}^{K} 1\{y_i = k\} \log \frac{e^{(\theta_k^T x_i)}}{\sum_{j=1}^{K} e^{(\theta_j^T x_i)}} \right]$$

where $1\{\cdot\}$ is the *indicator function* [104] that nullifies false statements (category belongings). Nowadays it is very easy to build an MLP architecture for multiple class recognition, but training and production phases can be time-consuming due to problem complexities, at the cost of no modularity perspectives. In addition, technical infrastructures and theoretical perspective changes in big-data collection, can often be addressed as the main causes of new and "updated" dataset revisions: classes (labels) and sample features are usually affected too, at the risk of disrupting pre-trained model's efficiency and correct working.

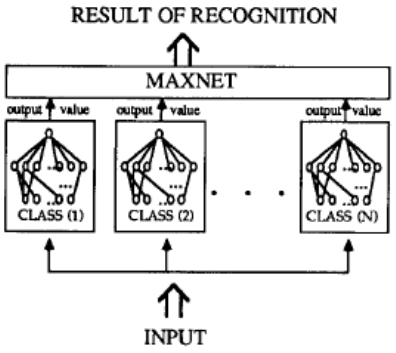


Figure 53 - The OCON model (w. MaxNet Algorithm)

As can be seen in Figure 52, the original OCON setup is a parallelization network composed of binary classifiers (*One-Class net*) each of which is a single output two-layer backpropagation, which can be trained and retrieved independently.

Every input sample is equally sent to every subnet input layer and processed traversing each class net. Every subnet will return single probability y_i which is not normalized to sum to 1 with the others.

To this extent a comprehensive output algorithm must be designed, in order to achieve a unique value categorization of the input, but at the same time the *unbalanced* output vector, consisting of the K non-normalized probabilities y_i (where K is both the number of classes and subnets) becomes a useful outcome to get insight in classification complexity and possible overlap of class boundaries in feature dataset (an example in Figure 54).

A unique complex architecture instead, would try to overcome this problem learning complex symmetries and tracing inaccurate but simpler non-overlapping class boundaries.

Figure 54 - HGCW vowel phonemes dataset (overlapping boundaries example)

Output Algorithms

The OCON model involves the use of a context-dependent output algorithm for unique class label inferring.
No clear literature references [102] [105] were found about this specific topic, but figures in [102] suggest a *MaxNet* algorithm implication.

In the *unsupervised learning* sub-field of ML, many algorithms were developed to autonomously discover distinctive patterns inside input data features (*data clustering*) and consequently, complex relationships between input(s) and output(s). The iconic computing principle *Winner-Take-All* [106] is at the core of the *competitive learning* paradigm, by which neurons compete with each other, inside the same architecture, for a unique activation: connections between them show the competition rules (subtractions and downscaling) in order to find the winner.

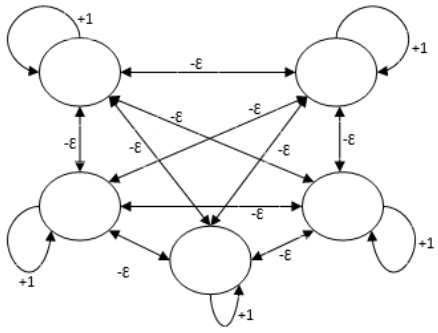


Figure 55 - MaxNet graph

The *MaxNet* is a *fixed weights recurrent recall network*, which usually serves as a sub-net for selecting the node having the highest input. All nodes are fully interconnected and there exists symmetrical weights in all weighted interconnections.
It uses an iterative process according to which all nodes receive inhibitory inputs from all other nodes through connections. The single node whose value is maximum would be finally active (or *win*) and the states of all others will result to be inactive. Max Nets use ReLU activation functions and its algorithm is enacted by the self-excitation weight (+1) and mutual inhibition magnitudes (ε), usually set to be positive and $< \frac{1}{m}$, where m is the total amount of nodes.

```
def Max_Net_algorithm
      input_1: f(·) = ReLU()                  # Activation function
      input_2: n                              # Nodes number
      input_3: ε = 1/n                        # Inhibition magnitude
      input_4: {y_1, …, y_n}                  # Network Outputs
      input_5: criterion                      # Winner-Take-All evaluation
            for k = (1, …, n) do              # Weights initialization
                  if k = n then
                        θ_k = +1
                  else
                        θ_k = ε
                  end if
            end for
            while criterion do                # Winner-Take-All loop
                  for k = (1, …, n) do
                        if i ≠ j then
                              y_k' = f(y_k - θ_k Σ_{i=1}^{n} y_i)
                        else
                              y_k' = y_k + θ_k
                        end if
                        y_k ← y_k'
                  end for
            end while
```

The weak aspect of this raw algorithm is traceable in rare occasions where multiple maxima occurs in the input vector: although it's usually uncommon to obtain multiple and equally high outputs from the OCON model, these exceptions could naturally imply an infinite looping condition inside the competitive algorithm.

In this case an effective alternative is represented by the *argument of the maxima* function [107] (also abbreviated *ArgMax*):

$$argmax\ f(x) \coloneqq \{x : f(s) \leq f(x), \forall\ s \in X\}$$

which returns a subset of discrete maxima indices of an input vector X.
The most common implementation of this function (for any programming language) returns a single value: the first occurrence of the maximum value (if more than once), so as to filter out eventual multiple occurrences.

We will test the *MaxNet* on our model and eventually substitute it with the ArgMax function call. In the **CONCLUSIONS** chapter hypotheses of theoretical more robust outputs evaluation will be discussed.

STATE-OF-THE-ART & RELEVANCE

The following pages attempt to summarize preliminary literature research carried out in order to effectively focus on the problem of vowel phoneme recognition, to understand its complexity, methodology and the results obtained.
Each figure in this chapter is directly derived from the cited references.

In terms of **datasets**, the first */hVd/* vowels database was originally developed by G.E. Peterson and H.L. Burney [108] [109] at Bell Telephone Laboratories (Murray Hill, New Jersey) coinciding with the first spectrographic possibilities unlocked by the *Sound Spectrograph* invention [110].

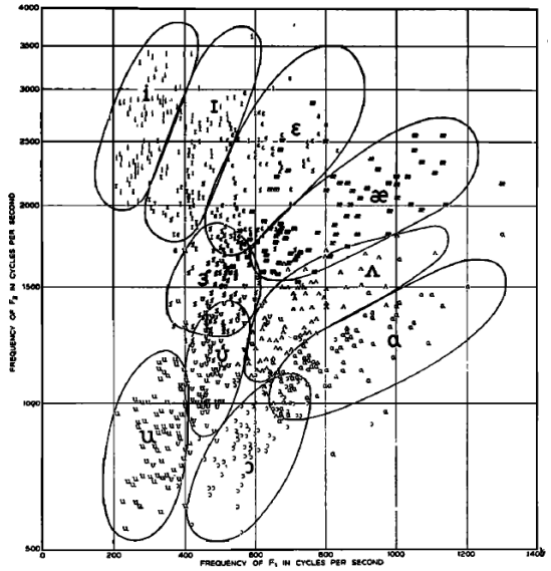


Figure 56 - PB dataset formants plot

A sequence of 10 vowels, spoken by 33 men, 28 women and 15 children were recorded and manually analyzed in order to compute narrow-band spectra, relative steady-state times and formant feature measurements: F_0, F_1, F_2 and F_3 with respective amplitudes. The so-called *PB database*[9] played a central role in formulation of vowel recognition theories [111] [112] [113] [114] [115] but it was concurrently addressed by strong critics too, mainly about its measurement limits: lack of temporal duration and spectral pattern changes [116] [117] tracking.

TABLE II. Averages of fundamental and formant frequencies and formant amplitudes of vowels by 76 speakers.

		i	ɪ	ɛ	æ	ɑ	ɔ	ʊ	u	ʌ	ɝ
Fundamental frequencies (cps)	*M*	136	135	130	127	124	129	137	141	130	133
	W	235	232	223	210	212	216	232	231	221	218
	Ch	272	269	260	251	256	263	276	274	261	261
Formant frequencies (cps)											
F_1	*M*	270	390	530	660	730	570	440	300	640	490
	W	310	430	610	860	850	590	470	370	760	500
	Ch	370	530	690	1010	1030	680	560	430	850	560
F_2	*M*	2290	1990	1840	1720	1090	840	1020	870	1190	1350
	W	2790	2480	2330	2050	1220	920	1160	950	1400	1640
	Ch	3200	2730	2610	2320	1370	1060	1410	1170	1590	1820
F_3	*M*	3010	2550	2480	2410	2440	2410	2240	2240	2390	1690
	W	3310	3070	2990	2850	2810	2710	2680	2670	2780	1960
	Ch	3730	3600	3570	3320	3170	3180	3310	3260	3360	2160
Formant amplitudes (db)	L_1	−4	−3	−2	−1	−1	0	−1	−3	−1	−5
	L_2	−24	−23	−17	−12	−5	−7	−12	−19	−10	−15
	L_3	−28	−27	−24	−22	−28	−34	−34	−43	−27	−20

Figure 57 - PB dataset statistics

[9] A scripted dataset revision is available at: https://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/areas/speech/database/pb/0.html [2023, June 24].

Prof. James Hillenbrand of Western Michigan University, provided an alternative solution with the HGCW database [118]: a set of 45 men, 48 women, 25 boys and 19 girls phonemes recordings, carefully screened by phonetician judgements during a 5/7 minutes individual interview selection and by examination of a 128-words passage reading.
Non-native English, evident speech/voice/language problems, respiratory infected and inadequate hearing capability speakers were carefully filtered out from the dataset.

Recordings were made with a Sony PCM-F1, a Shure 570-S dynamic microphone, with a 16KHz sampling rate. Every signal was then low-pass filtered at 7.2KHz and digitized (12-bit quantization) on a PDP-11/73 computer.

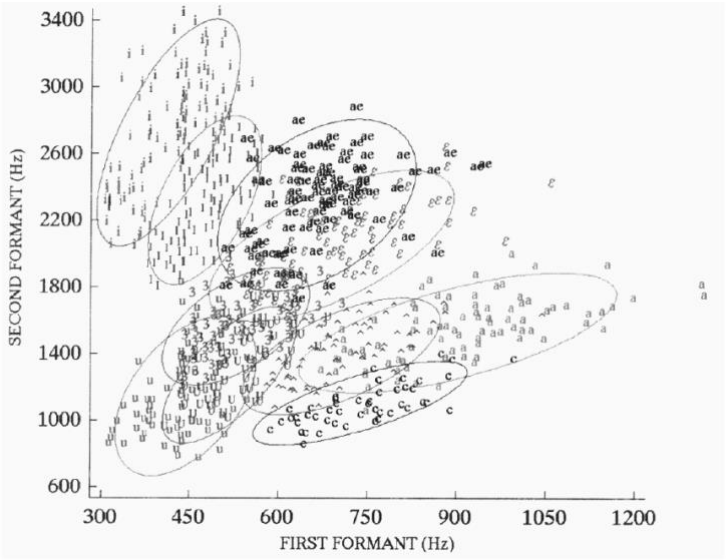


Figure 58 - HGCW dataset formants plot

Formants steady state analysis and fundamental contouring were programmatically structured:

1- LPC spectra (128-samples DFT) were computed by a 14-poles filter, with a 16ms Hamming window framing (50% of overlap = 8ms);
2- Formant frequency peaks were extracted by means of LPC spectral peaks retrieval; 3-points parabolic interpolated (61.5Hz resolution). Extracted trajectories were fed into an interactive editor for utterances discontinuity examination and manual correction;
3- F_0 contours were extracted by means of a 2-step autocorrelation pitch tracker with integrated pitch halving/doubling estimate sub-routines.

Durations and frequency features were judged by handcrafted measure/re-measurement technique, computing features variance for each sample group (statistics available in **APPENDIX-B**)
HGCW dataset[10] extended PB measurement limitations in terms of:

- *quantity*: doubled speech samples;
- *quality*: improved instruments and today availability of audio files;
- *complexity*: PB dataset shows few or none overlapping regions between adjacent phoneme families while almost all HGCW classes exhibit partially overlapped boundaries.

During the same years (1990 - 1993) the Texas Instruments/Massachusetts Institute of Technology (TI-MIT) (*corpus of read speech*) *dataset* was designed [119]: it was part of military research on telephone communications recognition systems development.

[10] A copy of the dataset can be requested at: https://homepages.wmich.edu/~hillenbr/voweldata.html.

This database contained samples from 630 different speakers, representing 8 different dialect subdivisions of American English (TI carefully assembled).
Its main breakthroughs are:

- included time-aligned orthographic, phonetic and textual transcriptions;
- structured speech waveform data, provided via CD-ROM resulting from the joint efforts of TI Research Department, MIT and SRI International (SRI), under sponsorship from the Defense Advanced Research Projects Agency - Information Science and Technology Office (DARPA-ISTO) and produced by the National Institute of Standards and Technology (NIST).

The speech corpus consists of 6300 utterances: 10 sentences spoken by 630 speakers, 70% of which are male and 30% female.
TIMIT prompts consist of 2 *dialect sentences* (designed by SRI), 450 *phonemically-compact* sentences (designed by MIT) and 1890 *phonetically-diverse* sentences (designed by TI), for approximately 5 hours of total speech materials.

Table 3.1: Dialect distribution of speakers

Dialect Region		# Male Speakers	# Female Speakers	Total # Speakers
Name	Code (dr)			
New England	1	31 (63%)	18 (27%)	49 (8%)
Northern	2	71 (70%)	31 (30%)	102 (16%)
North Midland	3	79 (67%)	23 (23%)	102 (16%)
South Midland	4	69 (69%)	31 (31%)	100 (16%)
Southern	5	62 (63%)	36 (37%)	98 (16%)
New York City	6	30 (65%)	16 (35%)	46 (7%)
Western	7	74 (74%)	26 (26%)	100 (16%)
Army Brat (moved around)	8	22 (67%)	11 (33%)	33 (5%)
Total # Speakers:		438 (70%)	192 (30%)	630 (100%)

Figure 59 - TIMIT speakers dialect distribution

Each speaker read the 2 dialect sentences, 5 of the phonemically-compact and 3 of the phonetically-diverse sentences. Recordings were made in a TI noise insulated booth. Binaural recordings were made using a Sennheiser HMD-414 headset microphone and a Brüel&Kjær 1/2" far-field pressure microphone (only the Sennheiser recordings were included on the commercial CD-ROM[11]). Audio recordings were then digitized ($SR = 20$KHz) using a Digital Sound Corporation SC-200 with LP anti-aliasing filter (10kHz). Files were finally de-biased and down-sampled to 16kHz.

Deeply involved in analysis tasks[12] TIMIT dataset is commonly pre-divided into training and test sets: the former contains 4620 utterances, the latter 1344, where 192 out of the 1344 are selected to form a *core-test* set (to increase data portability).

Microsoft Research Center, Georgia Institute of Technology, Carnegie Mellon, Princeton and Cornell universities developed a TIMIT spin-off dataset (538 sentences) for *vocal tract resonances* and formant frequencies estimate research: the VTR database [120].

VTR dataset consists of:

[11] There exist multiple online available revisions/variations: recording instruments selection, sentences filtering etc.
[12] A full version copy is available at: https://www.kaggle.com/datasets/mfekadu/darpa-timit-acousticphonetic-continuous-speech.

- 192 utterances from the core-test set and 346 utterances from train set;
- a total of 24 speakers from the *core-test* set (with 5 phonemically-compact and 3 phonetically-diverse sentences) and 173 speakers for the training set (with 1 phonemically-compact and 1 phonetically-diverse sentences each) to achieve quantitative balance.

Classes	Absolute Diff per frame (Hz)		
	F1	F2	F3
vowels	55	69	84
semivowels	68	80	103
nasal	75	112	106
fricatives	91	113	125
affricatives	89	118	135
stops	91	110	116

Table 1. Averaged pair-wise cross-labeling absolute difference per frame for F1, F2, and F3 and for each of the six broad phonetic classes (in Hz).

Classes	MSR			WaveSurfer		
	F1	F2	F3	F1	F2	F3
vowels	64	105	125	70	94	154
semivowels	83	122	154	89	126	222
nasal	67	120	112	96	229	239
fricatives	129	108	131	209	263	439
affricatives	141	129	149	292	407	390
stops	130	113	119	168	210	286

Table 2. VTR tracking errors measured by averaging absolute VTR difference per frame for the algorithm in [3] (MSR) and for the WaveSurfer algorithm against manually labeled data (in Hz). The results are listed for F1, F2, and F3 and for each of the six phonetic classes separately.

Figure 60 - VTR dataset formants statistics and estimates

This dataset was designed[13] to develop MATLAB (MathWorks©) tools to perform high-accuracy evaluation between three different formant tracking procedures: an ad-hoc LPC analysis algorithm (with 10ms Hamming window framing), "commercial" algorithms included in WaveSurfer[14] and Praat[15] software.

The only European effort in ASR dataset[16] creation, is linked to the *European Strategic Programme for Research and Development in Information Technology* (also known as ESPRIT FP2 1987-1992) [121] [122]. This is the *Robust Analytical Speech Recognition System*[17] (ROARS) project, which partly funded some French and Spanish university research centers, to develop respective idiom-based systems for phoneme recognition and automatic transcription [123].

The original dataset (not specifically found) was provided with *cochlear spectra*: the output of a filter bank with a constant $\Delta F/F_0$ and central frequencies distributed on a logarithmic Mel scale. Filter outputs were derived by 2ms and 8ms windowing (integrated on 4ms and 16ms) depending on the type of phoneme observed (stationary or transient). The *available* dataset revision is designed for the classification of nasal and oral vowels classification, with 1809 isolated syllabic features.

5 features were chosen to characterize each vowel phoneme: the amplitudes of the first five harmonics, normalized to the total spectral energy. Each amplitude/energy ratio was then signed positive, if it corresponded to a local maximum of the original spectrum and negative otherwise.

3-points tracking was performed, obtaining a total amount of 5404 samples:

---

[13] Available at: http://www.seas.ucla.edu/spapl/VTRFormants.html.

[14] https://wavesurfer-js.org/.

[15] https://www.fon.hum.uva.nl/praat.

[16] Unofficial repositories: https://datahub.io/machine-learning/phoneme, https://github.com/jbrownlee/Datasets/blob/master/phoneme.names, https://github.com/jbrownlee/Datasets/blob/master/phoneme.csv.

[17] Official reference: https://cordis.europa.eu/project/id/1736.

- 1 observation corresponding to the maximum energy peak
- 2 observations taken 8ms before and after the retrieved energy peak.

As we see, a kind of standardized set of data references exists and can be openly used to evaluate almost any formant-derived classification algorithm: something that generally does not seems to happen, because researchers tend to prefer ad-hoc sampling (especially in the context of DL algorithms literature) thus preventing comparative or generalization studies.

We decided to adopt the HGCW dataset for full availability of audio, tabular formants and temporal data: organized in a structured and multiplatform file format (*.data*). We hope that this could encourage literature adherence and simplify (at least) results evaluation.

In the field of **features extraction**, formant information continuity represents a crucial issue, [124] mainly for two reasons:

1) formant frequencies can change significantly within a given short time interval, such as at the boundary between colloquial hybrid nasalized/standard vocalization;
2) if a frame mistakenly labeled as voiced is included, it can lead to incorrect estimates for a large number of subsequent steps: common tracking algorithms, take into account previous states for successive computations (*confirmation based*, Figure 61 below show [125] technique).

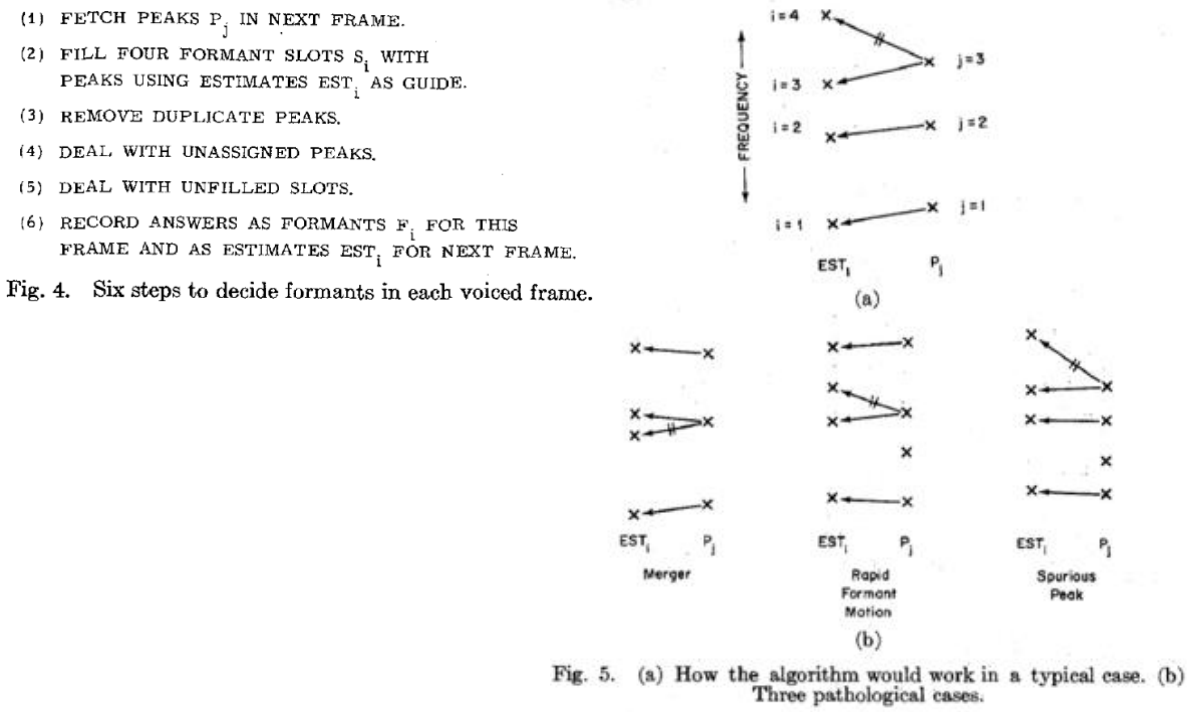


Figure 61 - Formant estimates tracking algorithm [116]

A good LPA tracking system should make continuity assumptions only for the initialization phase, possibly applying an *exception approach* later if continuity fails to

yield reasonable and predictable results: insights into peaks continuity solutions can be reviewed in [12] [125] [48].
All previous datasets (PB, HGCW and VTR in particular) are (potentially) characterized by uncertainty in formant tracking: sometimes "manually resolved", sometimes labeled (and eventually filtered out) by automatic exception handling.

Not surprisingly, most refined formant tracking techniques involve LPA(C) coefficients, computed with DL architectures or even NN *ensembling* [126].
In studying the Perceptron model, researchers [127] have concluded that a single-layer rectifier is capable of representing a standard LPA formula implementation, if:

1) Each unit input feature $(x_i, \dots, x_{i-p})$ is appropriately ordered to be the actual samples sequence of the input signal (possibly windowed): a p units input layer (where p is the order of the LPC filter);
2) The output $\hat{y}_i$ is thus predictable and compared to y_i (the $p+1$ samples of the input signal);
3) Training steps can follow a rectangular one-sample shifted order or a standard windowed overlapped shift;

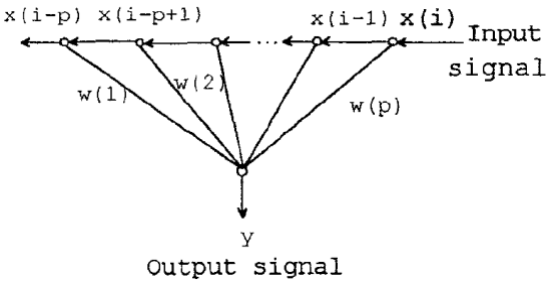


Figure 62 - LPA Perceptron implementation

4) MAE-based cost is computed at each training step and backprop/SGD optimize weights and bias until MAE drops below specified threshold (θ).

When the cost function is minimized (for each input signal) the rectifier Perceptron is able to linearly predict input signal values with w_i, which can be interpreted as the a_k coefficients of the LPA formula.

Formant frequencies studies show remarkable examples of auditory-based processing which deserve to be addressed, like the 3D vowels auditory "target zones" structure, defined by means of the logarithm of the formant distances [128] [129]:

$$\begin{gathered} Log(F_3) - Log(F_2) \\ Log(F_2) - Log(F_1) \\ Log(F_1) - sr \end{gathered}$$

where sr is the so-called *sensory reference*, empirically derived from dataset[18] speaker fundamental frequency:

$$sr = 168 \sqrt[3]{\frac{\sqrt{\prod_{n=1}^{N} F_0^{(k)}}}{168}}$$

where 168 (in Hz) is the geometric mean of fundamental frequencies across the entire dataset and the ratio numerator is the geometric mean of the speaker fundamental

[18] Research was conducted on PB dataset.

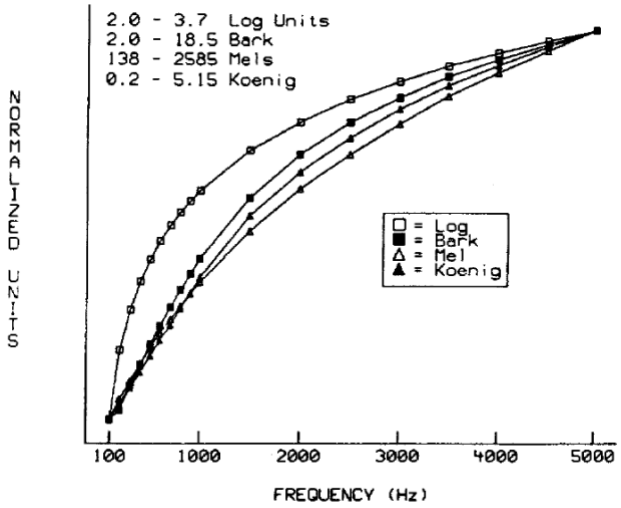


Figure 63 - Log, Bark, Mel, Koenig scales comparison

frequencies. Other researches [130] are deeply based on canonical auditory theories [48]. In that case, the Bark scale [131] [132], a "technical" Mel scale approximation [21] and the *lin-to-log* approximation [133] were respectively computed as:

$$f_{bark} = 13\arctan(0.76f_c) + 3.5\,arctan\left(\frac{f_c}{7.5}\right)^2$$

with f_c:

$$f_c = \begin{cases} f - 0.2(f - 150), & 150Hz < f < 200Hz \\ f - 0.2(250 - f), & 200Hz < f < 250Hz \\ f, & f > 250Hz \end{cases}$$

$$f_{mel} = \left(\frac{1000}{log2}\right)\log\left(\frac{f}{1000} + 1\right)$$

$$f_{Koenig} = \begin{cases} 0.002f, & f < 1KHz \\ (4.5logf) - 11.5, & 1KHz < f < 10KHz \end{cases}$$

We will avoid spectral and cepstral-derived descriptor literature (certainly the main part), although radically linked to actual CNN and RNN solutions for ASR. They represent a working but too broad *approach* to signal classification, while in this work we are interested in a more *vocalic approach* proposal.

As far as **classification** is concerned, *Linear Discriminant Analysis* (LDA) was one of the first supervised learning algorithms used in phoneme and speaker recognition. Historically developed by Sir Ronald A. Fisher (1890 - 1962) [134] in 1936, it is a statistical method that uses a linear combination of features for classification tasks, assuming that the data come from a multivariate gaussian distribution. According to the Bayesian classification rule, it is indeed possible to assign an input feature sample (x) to a class (y) if:

$$y = argmax_i \delta_i(x)$$

where:

$$\delta_i(x) = logf_i(x) + log\pi_i$$

with $\delta_i(x)$ the *discriminant function* (log-likelihood formulated), $f_i(x)$ the population density function and π_i the prior class-specific probability [135]. Decision boundaries are then traced where samples equally belong to two or more adjacent classes (same probability results). In the case of equal covariance between classes, LDA arises in:

$$\delta_i(x) = x^T \Sigma^{-1} \mu_i - \frac{1}{2} \mu_i^T \Sigma^{-1} \mu_i + log \pi_i$$

where Σ is the covariance matrix, and μ_i the mean value of class distributions. Otherwise:

$$\delta_i(x) = -\frac{1}{2} log|\Sigma_i| \mu_i - \frac{1}{2} (x_i - \mu_i)^T \Sigma_k^{-1} (x_i - \mu_i) + log \pi_i$$

and a curved discriminant function called *quadratic discriminant* (QDA).

LDA was applied on 10 vowel phonemes recognition tasks (on PB dataset) with *jack-knife* resampling technique [111] and formant distance features.

Formant feature Type	Task	Accuracy
Freq. - (in Hz)	Vowel phoneme Recognition	82.3%
Freq. - (in Bark)	Vowel phoneme Recognition	85.9%
Freq. - (in Hz)	Speaker Recognition	89.6%
Freq. - (in Bark)	Speaker Recognition	88%
Freq. distance (in Bark)	Speaker Recognition	41.7%

Table 2 - Syrdal et al. PB dataset LDA accuracy results

TABLE V. Classification matrix for ten vowels in hertz.

Group	Percent correct	Number of tokens classified into group /i/	/ɪ/	/ɛ/	/æ/	/ɝ/	/ʌ/	/a/	/ɔ/	/ʊ/	/u/
/i/	89.3	134	16	0	0	0	0	0	0	0	0
/ɪ/	80.7	13	12	6	0	0	0	0	0	0	0
/ɛ/	81.3	0	27	122	1	0	0	0	0	0	0
/æ/	85.3	0	1	19	128	0	1	1	0	0	0
/ɝ/	96.7	0	1	3	1	145	0	0	0	0	0
/ʌ/	82.7	0	0	0	0	0	124	19	7	0	0
/a/	77.3	0	0	0	0	0	26	116	7	1	0
/ɔ/	75.0	0	0	0	0	0	10	8	108	4	14
/ʊ/	75.3	0	0	0	0	0	3	0	5	113	29
/u/	78.7	0	0	0	0	1	0	0	2	29	118
Total	82.3	147	166	160	130	146	164	144	129	147	161
Jackknifed total	81.8										

Figure 64 - Syrdal et al. Hertz classification (confusion matrix)

TABLE VI. Classification matrix for ten vowels in bark differences.

Group	Percent correct	Number of tokens classified into group /i/	/ɪ/	/ɛ/	/æ/	/ɝ/	/ʌ/	/a/	/ɔ/	/ʊ/	/u/
/i/	95.3	143	7	0	0	0	0	0	0	0	0
/ɪ/	84.0	10	126	14	0	0	0	0	0	0	0
/ɛ/	86.7	0	20	130	0	0	0	0	0	0	0
/æ/	86.7	0	0	20	130	0	0	0	0	0	0
/ɝ/	94.0	0	2	6	1	141	0	0	0	0	0
/ʌ/	88.7	0	0	0	0	0	133	14	3	0	0
/a/	88.7	0	0	0	0	0	10	133	6	1	0
/ɔ/	79.9	0	0	0	0	0	3	11	115	4	11
/u/	77.3	0	0	0	0	0	4	0	3	116	27
/u/	77.3	0	0	0	0	0	0	0	3	31	116
Total	85.9	153	155	170	131	141	150	158	130	152	154
Jackknifed total	85.7										

Figure 65 - Syrdal et al. Bark classification (confusion matrix)

TABLE VII. Standard deviations of formants and bark-difference values for ten American English vowels (Peterson and Barney data).

Vowels	Standard deviations (Hertz) F1	F2	F3	Bark F1	F2	F3	F1–F0	Bark difference F2–F1	F3–F2
/i/	60	376	379	0.56	0.84	0.63	0.55	0.72	0.45
/ɪ/	75	339	426	0.66	0.90	0.80	0.52	0.76	0.38
/ɛ/	97	336	441	0.80	0.96	0.86	0.55	0.85	0.42
/æ/	172	290	413	1.21	0.92	0.83	0.85	1.01	0.52
/ɝ/	73	239	289	0.62	0.99	0.93	0.65	0.96	0.47
/ʌ/	114	191	411	0.85	0.95	0.86	0.53	0.71	0.73
/a/	147	158	378	1.02	0.85	0.79	0.68	0.61	0.81
/ɔ/	96	142	417	0.78	0.96	0.87	0.79	0.65	1.00
/ʊ/	71	196	457	0.62	1.11	0.99	0.59	0.85	0.87
/u/	76	222	454	0.69	1.38	1.00	0.56	1.14	1.40

Figure 66 - PB dataset statistics (standard deviations)

Generalized Linear Regression Models (GLM) [136] [137] have been shown [138] to be as much as efficient as LDA, when applied to a "normalized" PB dataset revision:

Formant feature	Model	Normalization (in sequential order)	Accuracy
Log freq.	GLM	None	87.4%
	LDA + MAE	F_0 geom. mean, 0.333	86.3%
	LDA + MAE	$-\overline{F_1}, -\overline{F_2}, -\overline{F_3}$	89.5%
Bark. freq.	GLM	None	86.2%
	LDA + MAE	F_0 geom. Mean, 1	85.3%
	LDA + MAE	$-\overline{F_1}, -\overline{F_2}, -\overline{F_3}$	88.3%
ERB freq.	GLM	None	86.8%
	LDA + MAE	F_0 geom. Mean, 0.5	87%
	LDA + MAE	$-\overline{F_1}, -\overline{F_2}, -\overline{F_3}$	88.8%

Table 3 - Nearey PB dataset GLM accuracy results

QDA applied to the same PB dataset scaled features [115] [118] didn't show improvements in overall accuracy for both vowel and speaker recognition (see Figure 67 & Figure 68 below, and **APPENDIX-B**).

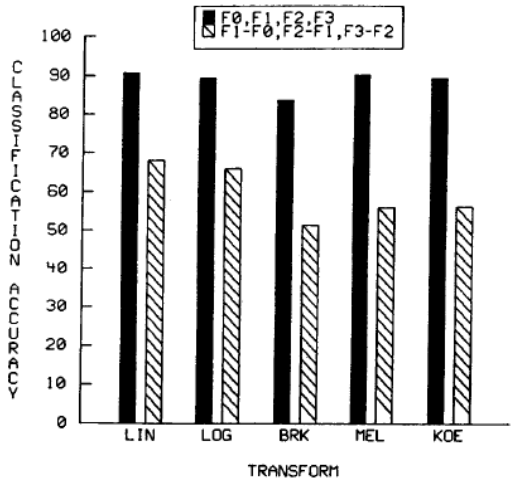


FIGURE 2. Overall accuracy of a quadratic discriminant classifier in identifying talker group (men vs. women vs. children). The classifier was trained on talker-group categories collapsed across all 10 vowels.

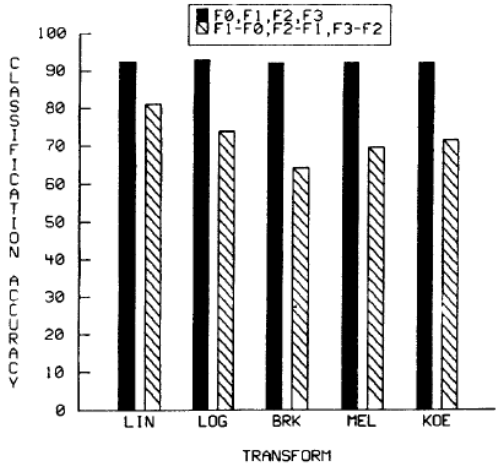


FIGURE 3. Overall accuracy of a quadratic discriminant classifier in identifying talker group (men vs. women vs. children). The classifier was trained on separate talker-group categories for each vowel. Results shown in the figure are averages across the 10 vowels.

Figure 67 - Hillenbrand et al. QDA PB dataset accuracy results

TABLE 1. Overall classification accuracy using various combinations of parameters.

Parameter Set	Transform				
	LINEAR	LOG	BARK	MEL	KOENIG
F1, F2	74.9	75.2	76.1	75.5	76.1
F1, F2, F3	83.6	83.6	83.6	83.5	83.6
F0, F1, F2	85.9	84.0	85.0	85.8	85.0
F0, F1, F2, F3	86.6	86.1	86.6	86.6	86.6
F1-F0, F2-F1, F3-F2	85.5	86.2	86.8	85.2	86.8

Figure 68 - Hillenbrand et al. QDA HGCW dataset accuracy results

The *hypercube* was the first NN architecture to be tested as a speech classifier [113]: with 2 hidden layers with fixed weights, 1 output layer with trainable weights. It was used as the core element of a more complicated MLP architecture, which comprised a *feature-map clustering* algorithm [139] [140] for unsupervised learning tasks.
Both models were trained and tested with PB dataset using the first two formant values only, showing:

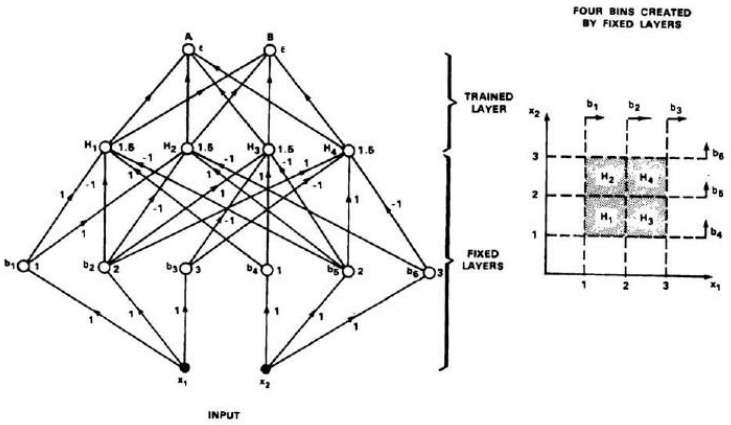


Figure 69 - Hypercube architecture

Architecture [w. 2 formants (in Hz)]	Accuracy
MLP (*hypercube*)	80.2%
MLP + *Kohonen* Feature Map	77.2%

Table 4 - Huang et al. Hypercube PB dataset accuracy results

From now on, NN experimentation seems to coincide with the increasing tendency in reference dataset abandonment, replaced by ad-hoc (sometimes "handmade") recordings solutions.

A standard 2 hidden layers (40/20 units) MLP architecture was evaluated [141] in the context of 11 long-stressed Swedish vowel recognition tasks. The ad-hoc dataset included 8 male and 4 female speakers (SR = 16KHz). Respective F_0, F_1, F_2 and F_3 frequencies were extracted by means of LPA (p = 19) estimate, on 6.4KHz band-limited spectra.

Input	Training time	Training iterations	Errors in test type	Errors in test speaker	Errors in test sex
F0-F3	10h 39m 57s	16 840 548	i:→y:		m
			y:→i:		m
			ɑ:→ɑ:		f
			ɯ:→ɛ:		f
			ø:→ɛ:		f
LPC samples	5m 2s	16 897	ɯ:→y:		m
			i:→y:		m
			ø:→ɯ:		m
			ɑ:→ɔ:		f
			i:→ø:		f

Figure 70 - Hult MLP training features, time & errors

Features	Accuracy
F_0, F_1, F_2, F_3	82%
19 LPC coefficients	73%

Table 5 - Hult MLP custom dataset accuracy results

One of the first MLP (9 units, 1 hidden layer) speaker recognition studies [142] involved directly LPC cepstral coefficients feature extraction. The training/test dataset

was composed of 9 male speakers (from New Zealand) and vowel samples were selected so as to be non-diphthongal extractions from multiple phonological contexts: */hVd/*, */bVd/*, */dVd/* and */gVd/*. This was done to achieve better coarticulatory integration of same class utterances: for a total of 44 utterances, recorded 8 times per subject. Although reported, the accuracy results were unfortunately biased by phoneme classes unbalancing.

VOWEL	Example word	1 mm	2 ph	3 ag	4 tr	5 ga	6 mn	7 mt	8 pm	9 fe
i*	heed	88	38	94	75	75	88	100	100	100
I	hid	94	19	94	63	50	81	100	100	50
ɛ**	head	94	63	88	56	75	88	100	100	100
æ*	had	56	56	100	31	75	69	75	100	100
a*	hard	100	63	81	88	75	63	100	25	100
ɔ	hoard	75	6	50	50	100	69	100	75	100
ɜ*	heard	56	25	56	75	25	50	100	75	75
u**	who'd	75	31	100	88	100	81	75	100	100
ʌ**	hud	69	56	100	94	75	38	100	50	75
o	hod	94	19	25	44	75	50	100	50	75
U*	hood	100	38	88	69	100	69	100	100	100
Average Rate		82	37	80	66	75	68	95	80	86

Table 1 : % Identification rates for each vowel for each of the nine subjects

*optimum vowel set using formant frequencies alone
**vowels that achieved average identification rates of > 75%

Figure 71 - Templeton et al. /hVd/ accuracy (confusion matrix)

Features	Accuracy
F_0, F_1, F_2, F_3	66.9%
14 cepstral coefficients	73.7%

Table 6 - Templeton et al. speaker recognition accuracy results

Other interesting research about MLP and LPC-derived features can be reviewed in [105] [143] [144] [145].

Since the second half of the 1990s, SR research has focused on more complex architectures (HMM [146], PNN [147], RNN, CNN): all share standardized feature extraction approach, the spectral one. We have chosen not to illustrate this dense part of the NN literature, but (to get a reference) these algorithms, which are today the state-of-the-art in complex audio SR complex networks, achieve an average 90% to 96% accuracy.

Based on this data review, we decided to focus on practicing MLP architecture algorithms with formant features. From our perspective, the reviewed literature cannot completely exclude MLP formants-based classifiers efficiency, due to following reasons:

1) none of the previous NN research included an evaluation of the HGCW dataset, despite its "more realistic" complexity;
2) none of the previous research comprehensively analyzed actual NN *regularizers/optimizers* in order to improve classification accuracy;
3) none of the previous research tried to effectively solve a multi-class problem with a multiple-subnets model (OCON): this could lead to parallel processing distribution and train/test phases modularity in case of different dataset phoneme availabilities;
4) formant estimation is still the most closely related vocal feature in phoneme recognition tasks: optimizing these classification methods (with actual computational capabilities) could lead to better theoretic generalization and more powerful embedded solutions;
5) We have already seen how NN provides a *multi-purpose* solution for different SR steps, in fact rectifier networks can be used efficiently in both the analysis and feature extraction phases (LPC prediction - LPA coefficients);
6) We can verify/outperform the minimum number of tracked formant features required to solve vowel phoneme recognition optimally for real-time applications.

THE OCON MODEL

In this chapter we describe our OCON proposal for phoneme vowel classification. We decided to evaluate OCON model efficiency because we did not find any previous experiments about it.

We started the practical side of this work, by defining our toolbox:

- *Python*[19]: a compiled and interpreted high-level programming language, intensively used in rapid application development and scripting data-science tasks. Python is totally C-optimized (its main implementation is *de-facto* CPython) and has a core structure developed in C while its interpreter and debugger are programmed in Python itself. It's a multi-paradigm scripting language that fully supports OOP (Object Oriented Programming), structured and functional programming and can be extended with a plethora of free-available extensions supported and developed by its heterogeneous community of programmers;
- *NumPy*[20]: one of the most famous Python external modules that innovates array processing including a specific-data class "`ndarray`": a n-dimensional data structure organized in stride views (contiguous elements) on memory. NumPy package contains mathematical functions, number generators, linear/complex algebra routines, transforms etc. using a mix of C and Fortran core pre-compiled library with high-level syntax;
- *MatPlotLib*[21]: a plotting library with an OOP Python API for embedding static, animated and interactive plots design using general-purpose open-source GUI toolkits such as *TkInter*, *QT* and *OpenGL*. MatPlotLib main classes are "`figure`" and "`axis`" (intended to be almost self-explanatory) and the entire `MatPlotLib.PyPlot` (2D plot sub-module) serves as an interface for class methods call and image properties changing;
- *PyTorch*[22]: is a Machine Learning/Deep Learning framework based on the Torch library, originally developed by Meta AI (and now part of the Linux Software Foundation) with a Python and a C++ interface. This package defines a class called "`torch.Tensor`" to store and operate on homogeneous multidimensional rectangular arrays of values.

[19] https://www.python.org.
[20] https://numpy.org.
[21] https://matplotlib.org/.
[22] https://pytorch.org.

PyTorch `Tensors` are similar to NumPy `ndarrays`, but can also be operated on a CUDA-capable NVIDIA GPU or other GPU platforms like AMD ROCm and Apple's Metal Framework.
We decided to use Google Colaboratory as reference scripting framework, for the following reasons:

- *Python Jupyter Notebooks*: *Project Jupyter* is a project to develop open-source software, open standards and services for interactive computing across three core programming languages: Julia, Python and R. Jupyter Notebooks (formerly *IPython Notebooks*) are web-based interactive computational environments for creating scripting documents, built using several open-source libraries including: *IPython*, *ZeroMQ*, *Tornado*, *jQuery*, *Bootstrap*, and *MathJax*. An *".ipynb"* is a structured file composed by: input/output cells which can contain Python code, GitHub flavored Markdown text cells, mathematics, plots and rich formats media;
- *cloud computing*: NN architectures typically require a large number of computations which are best distributed across high performance infrastructures. Google LLC is acknowledged as one of the best infrastructure and cloud services companies in the world and Colab is a practical example of this quality. A free user too can have access to a pre-compiled and custom extensible Python library set, which can be used on multiple runtime-core topologies: CPUs, GPUs as much as TPUs. Their cores certainly perform better than those available to the average user (including researchers).
- *multiple state-saved Outputs*: each Notebook is state-saved so as to be easily readable by third parties (if allowed to). Users can store Notebooks existing outputs (in `.ipynb`, `.py` or `.pdf`), reset and manage script states version control.
- *shared files*: like any other software of the Google Suite, Colab creates shareable files (inside the administrator Google Drive) for collaborative working environments;

Figure 72 - Our Python framework

We introduce our experimental research by describing dataset pre-processing and feature extraction first and then by presenting MLP architecture and OCON analysis.

In this experiment, each NN training procedure includes a performance and runtime evaluation: these are embedded code cells of which we can measure CPU-time (via ad-hoc scripting), isolating commands from the infrastructure communication process. Datasets, experiment results and MLP pre-trained architecture states are stored in an open-source cross-platform file format (`.npz`) for sharing and versioning purposes[23].

[23] https://drive.google.com/drive/folders/1JTsDqLAxzfosuuWrTxg_ELYYaF9ZZS3c?usp=drive_link.

Dataset Features Extraction & Conditioning

One of the main reasons for choosing the HGCW dataset is its *dual release* format: audio and textual. All */hVd/* utterance files are appropriately named and clipped so as to be easily indexable and analyzable.
Each audio *wave* file (*.wav*) is 1 sec. long, with a SR of 16KHz and 16bit of dynamic resolution. Filenames structure is as follows:

Ch. IDX-1	Ch. IDX: 2-3	Ch. IDX: 4-5	Example
m = *man*	nn = speaker n° (50 tot.)	xx = ARPABet phoneme	m10ae
b = *boy*	nn = speaker n° (29 tot.)	xx = ARPABet phoneme	b11ei
w = *woman*	nn = speaker n° (50 tot.)	xx = ARPABet phoneme	w49ih
g = *girl*	nn = speaker n° (21 tot.)	xx = ARPABet phoneme	g20oo

Table 7 - HGCW dataset filenames structure

For statistical tasks, original authors provided reliable formant analysis results (see **APPENDIX-B** and [118]) in a structured format, organized in 3 files:

1) *vowdata.dat - bigdata.dat*: those files contain formant estimates and tracking features: *vowdata* steady state measurements and *bigdata* formants tracking;
2) *timedata.dat*: contains vowel nucleus temporal features as: start, duration, tracking time-steps and *steady-state* perceptual estimates;
3) *vowdata.ds*: contains formant estimate descriptive statistics (see **APPENDIX-B**).

Based on reference quality, we decided to use pre-extracted features so as to focus only on the solution of the problem of classification.

Analyzing textual data by means of scripted *phoneme grouping, speaker grouping* and *null elements* filters, we found that:

Phoneme ARPABet	Elements	Boys	Girls	Men	Women	Encoded Label
æ = /ae/	134	25	17	45	47	0
ɑ = /ah/	135	24	19	45	47	1
ɔ = /aw/	133	24	18	45	46	2
ɛ = /eh/	139	27	19	45	48	3
ɜr = /er/	118	26	18	37	37	4
eɪ = /ei/	126	25	17	43	41	5
I = /ih/	139	27	19	45	48	6
i = /iy/	124	20	18	43	43	7
o = /oa/	136	25	19	45	47	8
ʌ = /oo/	139	27	19	45	48	9
U = /uh/	138	26	19	45	48	10
u = /uw/	136	25	19	44	48	11
TOTAL	1597	301	221	527	548	12

Table 8 - HGCW dataset phoneme/speaker classes statistics

only 1597 elements could be effectively involved in classification applications, compared with the total file size of 1668 samples. Samples with one or more *null* formant estimates were completely excluded in order to avoid miss-classification due to analysis uncertainties: this implies an additional level of under-representation for certain

phoneme and speaker families. We can confirm that not all the original recordings were effectively analyzed and tabulated.

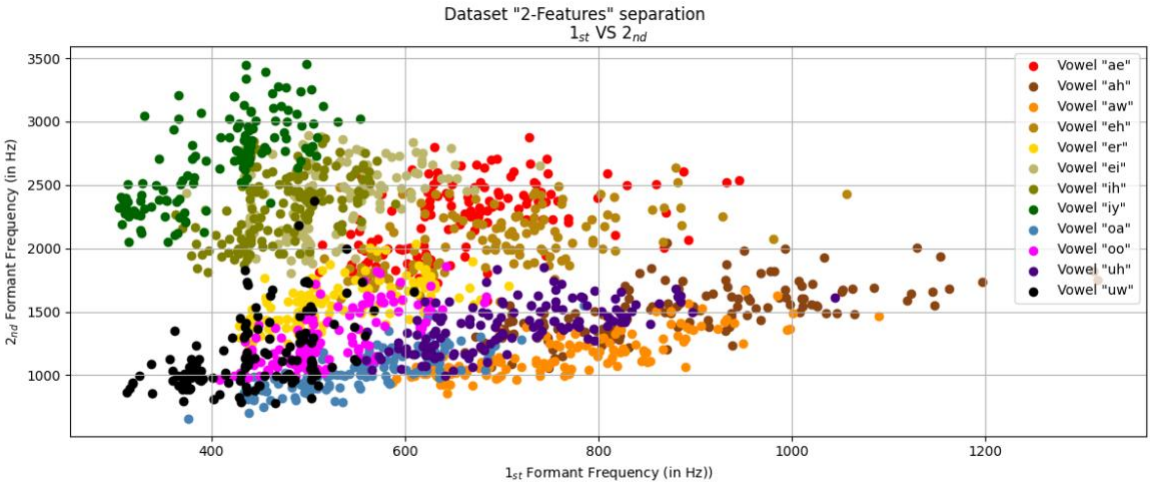


Figure 73 - Giacomelli et al. HGCW dataset formants plot

According to different tasks of the experiment, we derived multiple dataset variants, where each sample is described by:

1) fundamental frequency + 3 formants features (steady state estimates) (Figure XX)
2) dataset (1) speaker-based filtered: *men* only, *women* only, *children* only
3) fundamental frequency + 12 formants features (first 3 formants sampled at steady state, 10%, 50% and 80% of effective vowel nucleus duration).

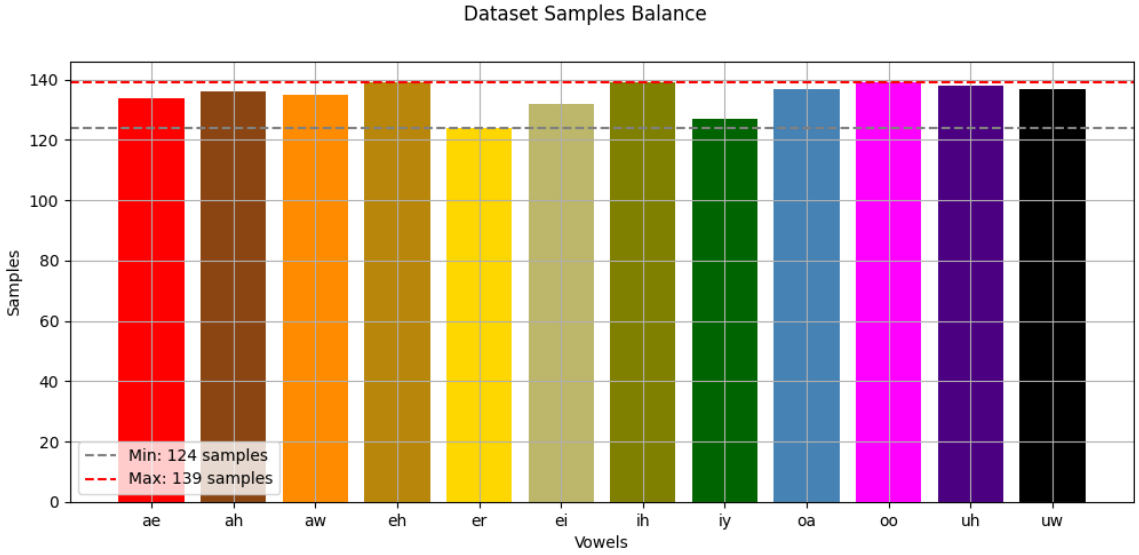


Figure 74 - Giacomelli et al. HGCW dataset phoneme classes

Considering that the frequency of the fundamental can vary considerably within the sample of speakers for physiological reasons, and that the pitch varies within the prosody, we decided to normalize the value of the formants with respect to that of the fundamental:

$$formant\_ratio = \frac{f_{formant_i}}{f_{pitch}}$$

where f_{pitch} is the fundamental frequency computed by double-pass autocorrelation and hysteresis-ZCR algorithms (see **STATE-OF-THE-ART** and [118]).
This normalization technique seems to ensure roughly better phoneme segregation (Figure 75 and **APPENDIX-B**).

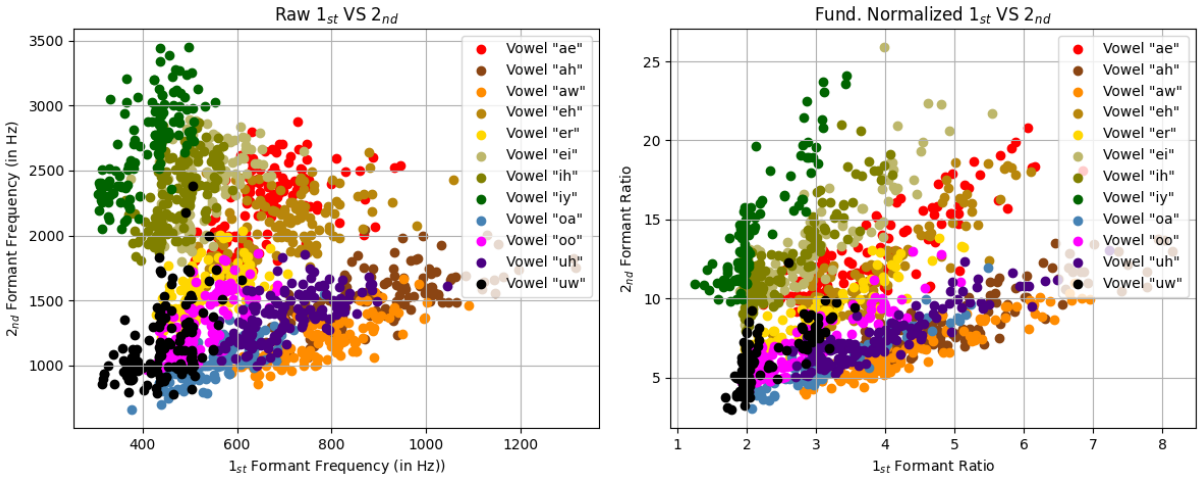


Figure 75 - Giacomelli et al. Raw vs Normalized HGCW dataset formants plot

The formant-to-fundamental ratio should be able to express formant location in a pitch-independent way and we will evaluate its quality with our model experiments.

To improve stability in neural networks learning, filtered data were normalized via *min-max* scaling:

$$\hat{x} = \frac{x_i - \min(x)}{\max(x) - \min(x)}$$

where x_i is the i-esimal sample feature and x the reference group, for that feature.

In ML (and DL) data pre-processing [148] features are disparate and usually belong to different domains, so min-max scaling is easily achievable normalizing each feature (usually columns arranged) by the minimum and maximum value of that column. In our study, features are structured in the following way:

Sample IDX	F_0	F_1 Ratio (steady state)	F_2 Ratio (steady state)	F_3 Ratio (steady state)
1	$F_0(x_1)$	$F_1(x_1) \,/\, F_0(x_1)$	$F_2(x_1) \,/\, F_0(x_1)$	$F_3(x_1) \,/\, F_0(x_1)$
2	$F_0(x_2)$	$F_1(x_2) \,/\, F_0(x_2)$	$F_2(x_2) \,/\, F_0(x_2)$	$F_3(x_2) \,/\, F_0(x_2)$
...				
n	$F_0(x_n)$	$F_1(x_n) \,/\, F_0(x_n)$	$F_2(x_n) \,/\, F_0(x_n)$	$F_3(x_n) \,/\, F_0(x_n)$

Table 9 - Giacomelli et al. dataset structure

and each formant belongs to the same domain, the frequency indeed. Instead of min-max scaling in a column-wise way, we decided to normalize ratios considering x as the entire formants dataset, excluding F_0 column (Table 9).

Previous tests with column independent min-max scaling proved that a standard MLP architecture was able to achieve sufficient results in multi-label classification (82-88%) but ratio features were theoretically unexplainable.

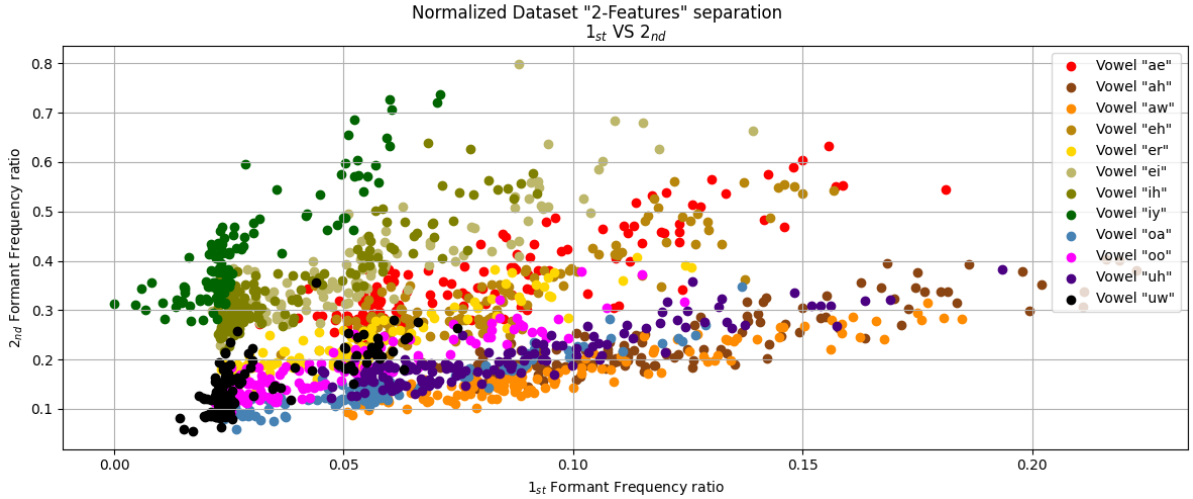


Figure 76 - Giacomelli et al. HGCW dataset Min-Max

We also evaluated normalized ratios PMD (*probability mass distribution*) to ensure that *Z-scoring* (standardization) was not a valuable normalization alternative (generally suitable for normally distributed data, Figure 77):

$$x_z = \frac{x_i - \bar{x}}{\sigma_x}$$

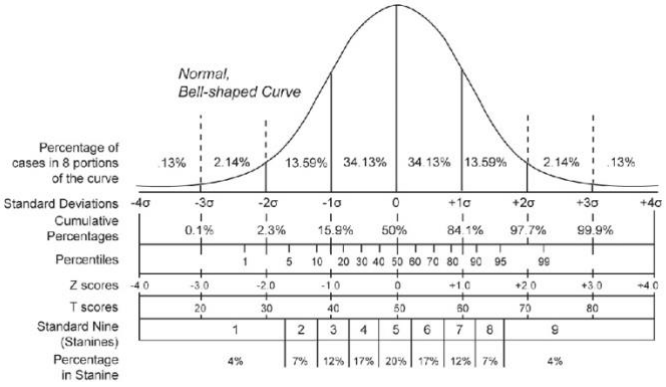


Figure 77 - Gaussian probability distribution (w. statistics)

where $\bar{x}$ is the mean value of the feature group and σ_x its standard deviation.

A phoneme-specific PMD evaluation (**APPENDIX-B**) proved how ratios possess common tendency in *Poisson* [149] or Log-*Normal* [150] distribution.

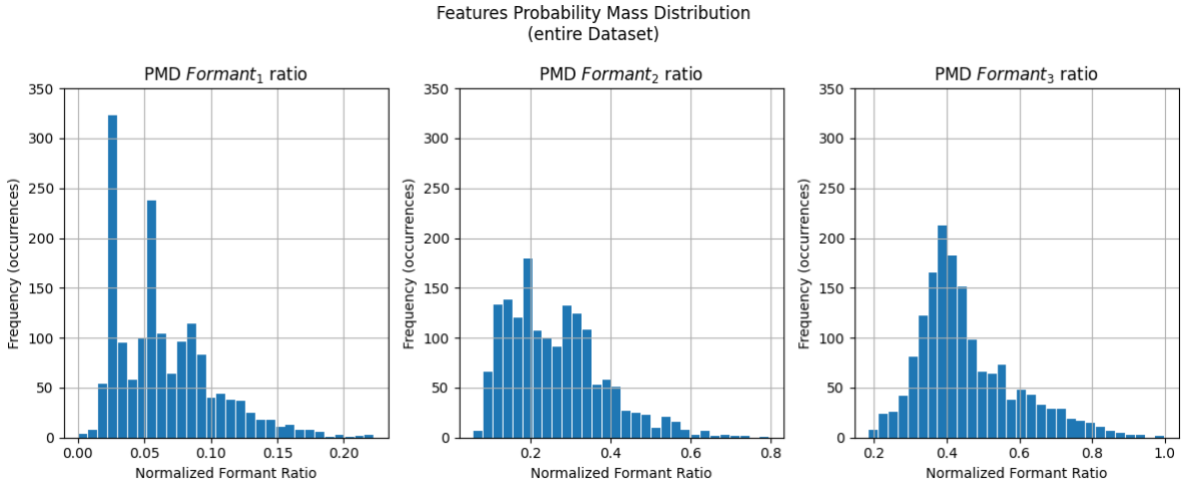


Figure 78 - Giacomelli et al. Formant ratios PMD analysis

No augmentation or other data-processing techniques were used in order to preserve authentic dataset complexities.

To promote data portability and usability we encoded all feature matrices in an open-source *NumPy* compressed data format (*.npz*) so as to preserve numeric resolution due to binary data encoding. The same data format will be adopted for NN experiment results storage.

Following a structure caption for all feature matrices we set in our study:

Matrix "*name*"	Rows (Samples)	Features[24]	Columns
4_features_complete	1617	F_0, F_1, F_2, F_3 (*SS*)	4
4_features_men	532	F_0, F_1, F_2, F_3 (*SS*)	4
4_features_women	558	F_0, F_1, F_2, F_3 (*SS*)	4
13_features_complete	1597	F_0, [F_1, F_2, F_3 (*SS*)], [F_1, F_2, F_3 (10%)], [F_1, F_2, F_3 (50%)], [F_1, F_2, F_3 (80%)]	13
13_features_complete_ (w_speaker)	1597	F_0, [F_1, F_2, F_3 (*SS*)], [F_1, F_2, F_3 (10%)], [F_1, F_2, F_3 (50%)], [F_1, F_2, F_3 (80%)]	14 (speaker label included[25])

Table 10 - Giacomelli et al. features dataset variants

additional dataset variants are obtainable by data sub-filtering (-) and/or juxtaposition (+):

- *n_features_train_set*: "*n*"*_features_*"*mm*" - F_0;
- *4_features_adult* (or *no_children*): *4_features_men* + *4_features_women;*
- *4_features_children*: *4_features_complete* - *4_features_adult*.

[24] Ratio min-max scaling included. *SS* = *steady state*.
[25] Speakers grouping: *men*, *women*, *children*.

Subnetworks Architecture & Hyper-parameters estimate

Our model proposal is based on parallel simplification of the multi-label classification problem by means of an OCON architecture approach.
In this case we are interested in splitting a 12 phoneme classification task on a 12 parallel binary classifiers network, where each subnet is focused on recognizing its unique phoneme class by associating samples to a label equal to 1 or 0 when the recognition has a positive or negative outcome.
Before studying subnet properties, we need to formulate a *binarizing* technique to transform sample labeling according to our approach.

One-Hot encoding (or *binarization*) is a statistics derived technique to transform features or category labels into so-called *dummy variables* (0 or 1) to ensure that the model doesn't assign different importance depending on the value of the ordinal of the class. In previous DL chapters we have seen how MLP architectures tends to approximate good decision boundaries as much as the training dataset is quantitatively and qualitatively balanced: for the qualitative balance we decided to rely on the complexity and variability due to the *sampling effort* performed by the original authors, but the quantitative balance is strictly required.

Due to Table 8 observations, we are forced to re-filter the number of samples to effectively one-hot encode the HGCW dataset, thus creating a further under-represented sub-set. The following pseudocode explains how each phoneme class is processed to achieve quantitative class balance.

```
def one-hot_encoding
      input_1: c                                  # 1-Class ID
      input_2: s                                  # Phoneme groups size
      input_3: x                                  # Samples dataset
      input_4: y                                  # Original labels

            one_class = x(y_c)                    # Initialize 1-class subset
            size = length(one_class)              # 1-class size
            zero_class = []                       # Initialize 0-class subset
            sub_size = int(size / 11)             # 0-class sub-groups size

            for k in y do
                  if y_k ≠ c then                 # For each 0-label...
                        zero_class =              # Take a few random samples
                              rand(x_k, sub_size) # from other classes
                  else
                        pass
                  end if
            end for

            y1 = array(1, size)                   # 1_class labels
            y0 = array(0, size)                   # 0-class labels
```

This procedure creates multiple variants of the original dataset whose dimension depends directly on the true-class samples size, trying to achieve correct balance by reserving the same number of samples for the false-class (equally re-distributed among the remaining 11 phoneme classes). True-class sizes may not be divisible by 11: this leads

to a 1 to 3 samples variability between True and False classes but we can accept it if compared with the starting inequality.
One-hot encoding function will be invoked once for each training cycle, before creating train and test splits and mini-batch partitioning. Speakers based binarization is achievable in the same way, just relying on $y_{speaker}$ labels array (Figure 79).

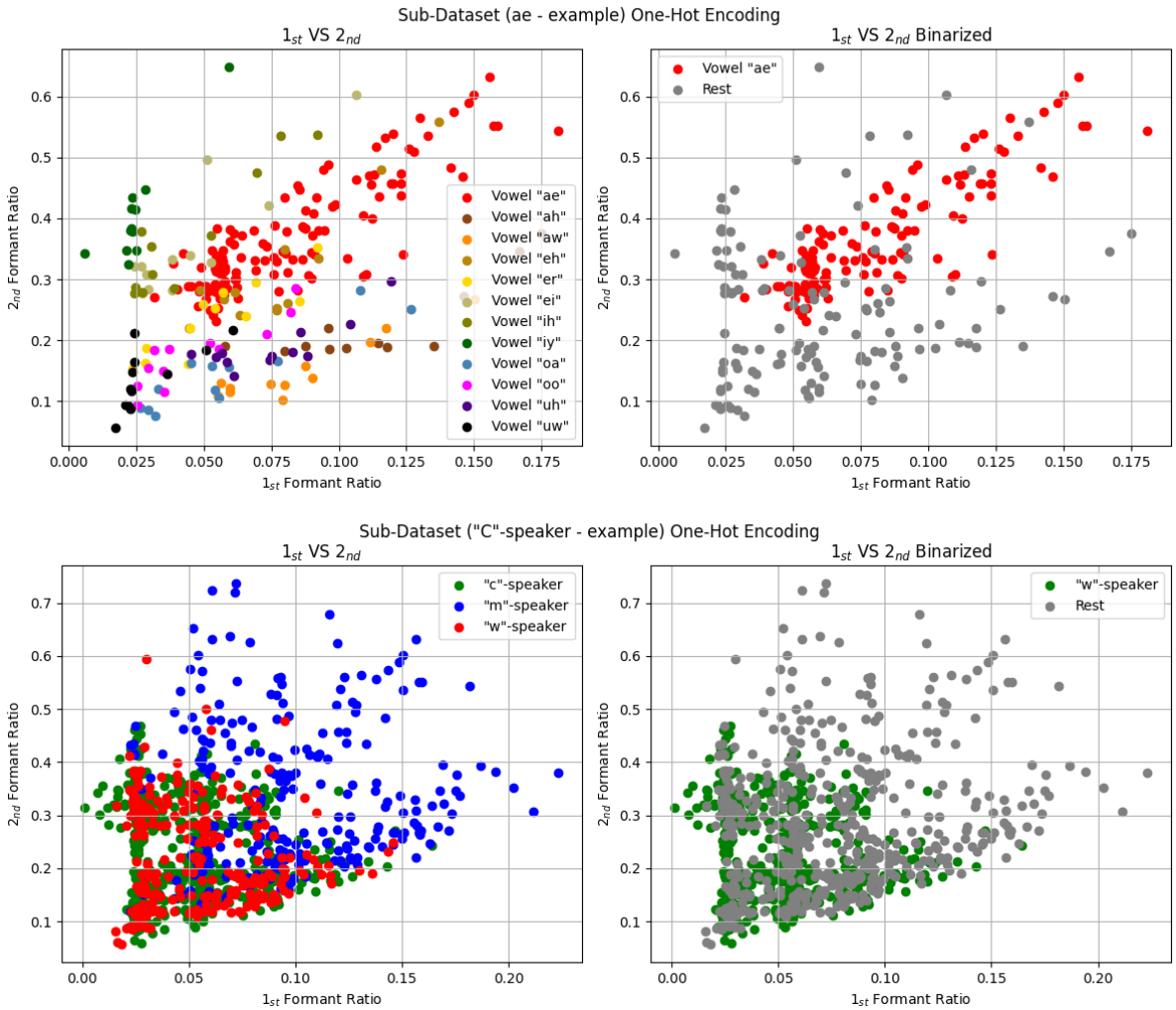


Figure 79 - Giacomelli et al. HGCW one-hot encoding (phoneme & speaker examples)

Figure 80 - Hyper-parameters grid-search (3D example)

It has been decided that each True-class subnet would be a complex MLP neural network, which in turn needs to be studied to optimize its binary classification capabilities. For this purpose, we adopted a *Grid-based* hyper-parameters *search*.

Grid-based search is an estimation method in which all possible parametric combinations are sampled on a multi-dimensional system (one axis for each parameter) stabilizing common or variable-independent resolutions (for each axis).

Hyper-parameters grid-search can easily become a time-consuming task, especially if the tested combination being tested causes a noticeable increase in the amount of required computation: this becomes even more important in our time-constrained framework. A good trade-off has been achieved by applying *informed* grid-based search methodology, summarized as follows:

1) define a parametric combination;
2) sample each parameter with an ad-hoc resolution;
3) sort hyper-parameters combination by ordered indexing;
4) best combination estimate can be:
    a. inherited for subsequent experiments: will be consider the *actual optimum estimate*;
    b. used to narrow resolution quantity for a new search experiment on the same hyper-parameters set;
5) go to step (2) and re-iterate as much as you need.

We decided to train and evaluate MLP architectures relying only on binary phoneme classification performance (accuracy). After achieving a good hyper-parameter estimate, we will test the same architecture and learning algorithms for speaker recognition tasks too, just reducing the number of one-class subnets: from 12 to 3 (*men*, *women* and *children* classes).

MLP training follows a standard *train/eval/test* procedure, schematically as follows:

1) extract hyper-parameters combination;
2) create a repeatable experiment: each random initialization is ruled by a fixed numeric *seed* (42);
3) create the defined MLP binary classifier architecture;
4) start an experiment timer;
5) one-hot encode the input features and labels dataset;
6) randomly split the dataset into balanced *training* set (70%), *evaluation* set (15%) and *test* set (15%);
7) compute a mini-batch (32 samples) training/eval/test procedure and store resulting *test accuracy* (in %);
8) stop the experiment timer and store result *training time* (in sec.);
9) go to step (1) and repeat until all combinations are tested.

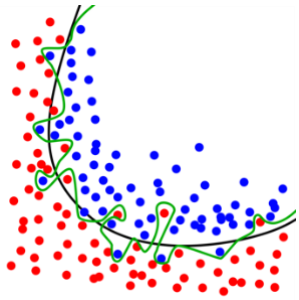


Figure 82 - Overfitting classification boundaries

Figure 81 - Dataset Train/Eval/Test split

Train/eval/test is the *gold-standard* training procedure for NNs and consists of splitting datasets in 3 subsets in order to increase resistance to data and research overfitting.

Data overfitting [151] is a major scourge in DL algorithms: in classification tasks it is represented by boundaries that are too much accommodating with respect to training data variability.

Overfitting boundaries leads to non-generalizable solutions but are hard to detect, especially if *test* accuracies are computed on the same training set.
For these reasons we compute a double independent sub-set partitioning in *eval* and *test* set (reserving most of the samples for the training set) in order to increase the variability and thus the independence between accuracy results, which also increases the quality and generalizability the evaluation.

Our first heuristic search comprised some fixed and testing *architecture hyper-parameters*, namely:

- fixed number of input units: 3 (steady state formant estimates)
- fixed number of hidden layers (HL): 1
- fixed activation function type: ReLU (Kaiming He standard initialization)
- fixed number of training epochs: 1000 (for training batch set)
- hidden nodes (HN): 10, 50 and 100
- backpropagation optimizers: Adam and RMSProp
- learning rate (LR): 0.001, 0.0001 and 0.00001

each training cycle is repeated 3 times in order to account for dataset random partitioning: True-class-specific results are averaged across iterations and experiment results are averaged across all 12 trained subnets. We have a total of:

$$18\ combinations \times 3\ iterations \times 12\ subnets = 648\ training\ cycles\ (1000\ epochs\ each)$$

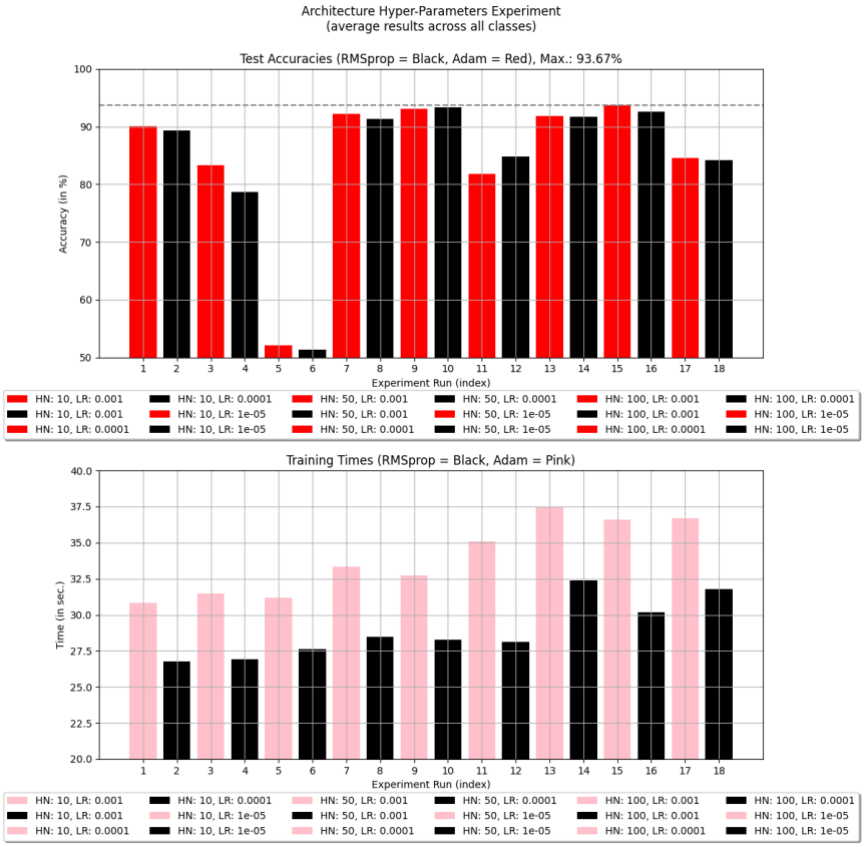


Figure 83 - Giacomelli et al. one-class architecture analysis

The accuracy plot (Figure 83) shows that two architecture combinations, regardless of the backprop algorithm, lead to a similar average recognition (around 93.67%):

1) HN: 50, LR: 0.0001;
2) HN: 100, LR: 0.0001;

but looking at the average training times, we see that:

1) RMSProp: 28sec. , Adam: 32.5sec.;
2) RMSProp: 30sec., Adam: 36.6sec.;

This confirms an increasing tendency in learning time as the size of the architecture increases (with some exceptions) and that RMSProp optimizer better slows this trend than Adam does.

We decided to inherit the best accuracy configuration: HL = 1, HN = 100, LR = 0.0001, optimizer = Adam (with average training time per class: 36.60 sec.) due to the fact that we observe a delta average training time of 6sec. only, prioritizing accuracy criterion.

Following experiments are structured to search for best regularization hyper-parameters estimate, in order for:

1) DropOut
2) Batch Normalization
3) L2 Norm

with fixed-0 architecture biases initialization.

DropOut heuristic set is:

- fixed number of training epochs: 3000 (for training batch set);
- previous experiment best run parameters;
- input layer DropOut rates: 0.8 and 0.9;
- HL DropOut rates: 0.5, 0.6, 0.7, 0.8, 0.9 and 1

Each training cycle is repeated 6 times now (on the same architecture topology), with the same averaging methods. Here we have a total of:

$$12\ combinations \times 6\ iterations \times 12\ subnets = 864\ training\ cycles\ (3000\ epochs\ each)$$

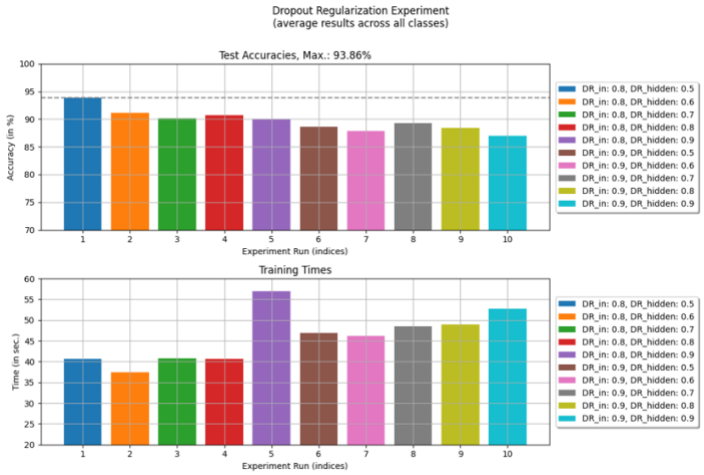


Figure 84 - Giacomelli et al. one-class DropOut analysis

Fortunately, we find that the best run (the 1st) is even one of the fastest: we obtain an average accuracy increase of 0.19% at the cost of around 3.4sec, with 0.8 DropOut probability for input layer nodes and 50% probability for hidden nodes.

Batch normalization heuristic set is:

- fixed number of training epochs: 1000 (for training batch set);
- architectural experiment best run parameters;
- DropOut experiment best run parameters;
- LR: 0.001, 0.0001 and 0.00001

each training cycle is repeated 10 times now (on the same architecture topology), same averaging methods. We re-tested some learning rates due to the intrinsic relationship between LR and *Batch-norm* methodology. Here we have a total amount of:

$$3\ combinations \times 10\ iterations \times 12\ subnets = 360\ training\ cycles\ (1000\ epochs\ each)$$

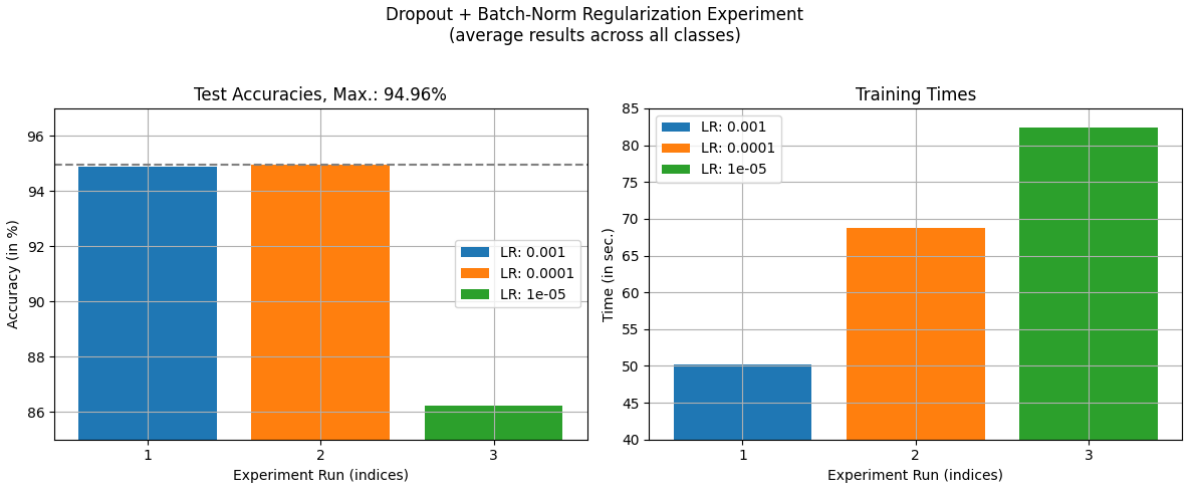


Figure 85 - Giacomelli et al. one-class Batch-norm analysis

The previously estimated LR is confirmed to be the best one and Batch normalization yields an increase of 1.1% in test accuracy at the cost of average training time of 68 sec. (almost doubled): we still accept this increasing amount.

Finally, *L2 (Ridge) penalty* heuristic set is:

- fixed number of training epochs: 1000 (for training batch set);
- all previous experiments best run parameters;
- λ penalty scale coefficient: 0.01, 0.001 and 0.0001

every training cycle is repeated 10 times (on the same architecture topology), same averaging methods. Here we have a total of:

$$3\ combinations \times 10\ iterations \times 12\ subnets = 360\ training\ cycles\ (1000\ epochs\ each)$$

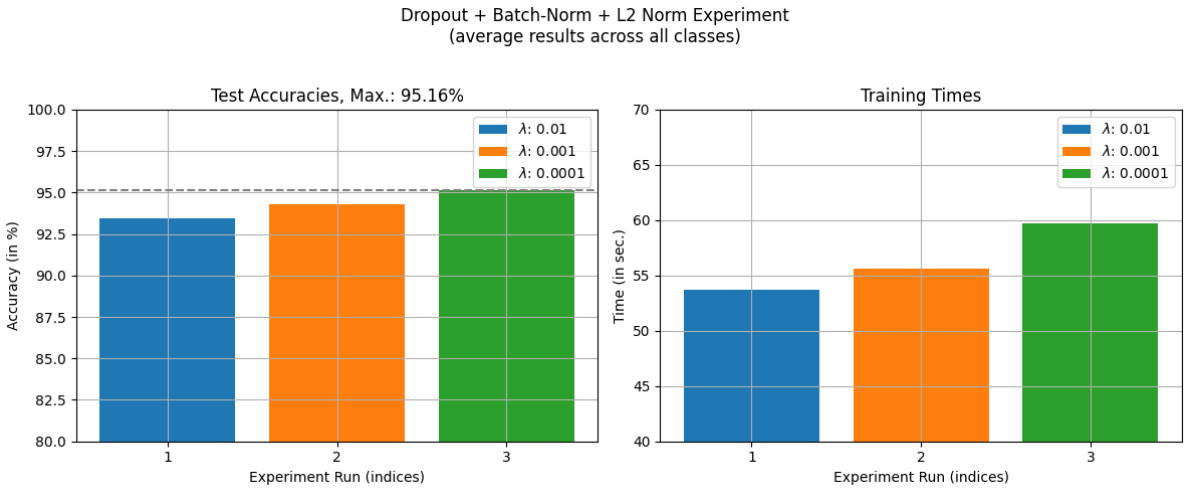


Figure 86 - Giacomelli et al. one-class L2 analysis

and best λ (also *weight decay*, 0.0001) yields better average accuracy (+0.19%) with concurrent optimization (decreasing) of training times: dropped below 60 sec.

At the end of a unique informed grid-search experiment we obtain a MLP binary classifier descriptive estimate as follows:

- Input layer: 3 nodes = 3 features (formant-to-fundamental ratios, min-max norm.)
- Hidden layer: 100 nodes
- Output layer: 1 node (normalized probability)
- Activation function: ReLU
- States initialization: Kaiming He standard distribution (weights), 0 (biases)
- Backprop optimizer: Adam
- Learning rate: 0.0001 (10^{-4})
- Mini-Batch training
    - Re-iterated sub-dataset shuffling (one-hot encoding)
    - Mini-batch size: 32 samples
- Regularizations
    - Weights L2 norm: 0.0001 (10^{-4})
    - Input layer DropOut rate: 0.8 (norm. probability for each node)
    - Hidden layer DropOut rate: 0.5 (norm. probability for each node)
    - Batch normalization

This MLP architecture is our one-class proposal to be used inside the OCON model.

OCON Network Training & Evaluation

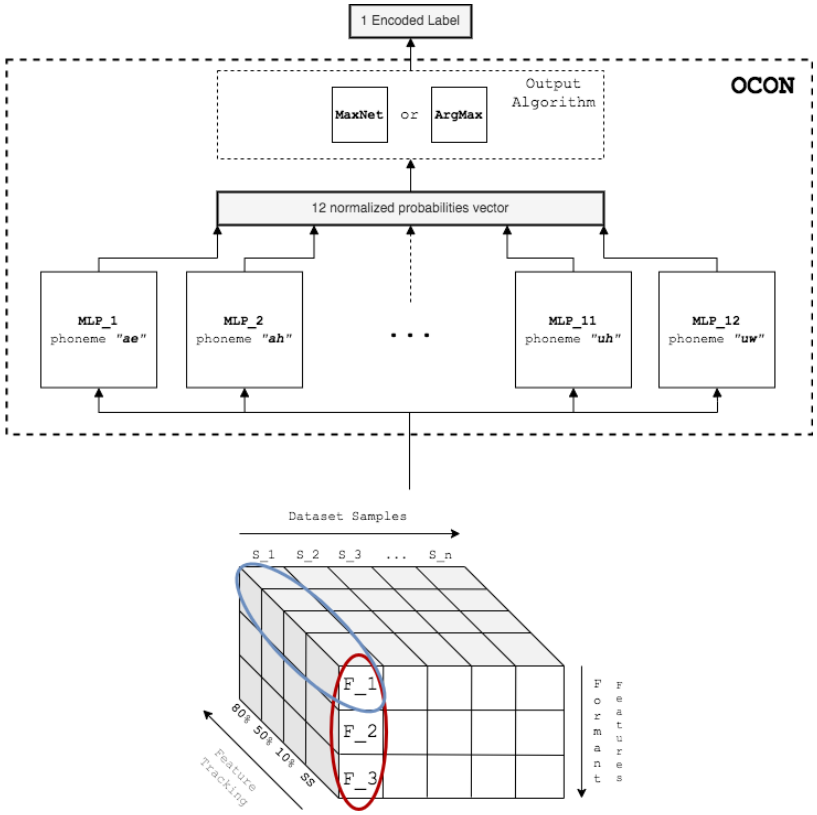


Figure 87 - Giacomelli et al. Deep Learning OCON proposal

Figure 87 shows our OCON model (a parallelized bank of MLP binary classifiers) and its forward propagation routine, which can be summarized as:

1) extract a sample features array (with relative encoded label) from the dataset and send it equally to all the one-class subnets;
2) compute 12 parallel `one-hot encodings` (one for each subnet) with respect of the MLP True-class;
3) trained MLPs will output normalized probabilities (a total of 12) which correspond to respective True-class belongings. Send the resulting vector to the desired `Output Algorithm`;
4) The output algorithm searches for the maximum value of the input vector and outputs its related index (starting from 0): this value is the OCON model prediction label.

We created a scripted bank of MLP architectures training each one sequentially. One of our deploying framework benefits is represented by custom selection of runtime infrastructures: CPUs (central processing units), GPUs (graphic processing units) or TPUs (tensor processing units). Glossing over the TPUs availability, GPU selection can obviously speed up NNs training phase by parallelizing each architecture learning loop

on distinctive physical cores. We decided to adopt CPUs sequential approach to easily measure isolated training loops for report and assessment needs.
The OCON model, although its NN definition, doesn't have a backprop algorithm *per sé*. The process of learning of this model relies on each MLP backprop algorithm.

Phoneme recognition experiments review follows, where we tested each input dataset variant previously introduced. We remember that MLP estimate is based on the simplest dataset structure: 3 formant features, estimated at the steady state portion of vowel utterances. Test accuracies and training times (for each subnet) will be analyzed in order to understand OCON output label results. The most performant OCON model will be then evaluated in a *speaker recognition* task too.

In this series of experiments, *train/eval/test* procedure remains basically the same one implemented for architecture study, but we tried to refine Early stopping algorithm defining a 2-variables match escaping condition:

- 1st variable: a loss threshold value (evaluated averaging last 50 samples results) ;
- 2nd variable: a test accuracy threshold (evaluated on the last batch-set).

These two variables are empirically assessed in order to guarantee best results within a learning time frame not exceeding 25/30 min. (for each architecture)[26].
This means that the architecture learning phase cannot be considered optimal: our purpose is to evaluate benefits and potentials of our approach, not commissioning a pre-trained model.

[26] In free Google Colab we have runtime limitation (around 3 consecutive hours).

Phoneme Recognition

In the 1st experiment we tested the OCON model against the *steady state* dataset variant. Heuristic setup is:

- 3 input features: F_0, F_1, F_2, F_3 ratios (SS) + 1 encoded label (for each sample)
- intermediate MLP output vector: 12 normalized True-class probabilities
- 1 output prediction: encoded label
- Training epochs: 1000, for each one-hot encoded batch-set (32 samples mini-batch training)
- Data Batch samples balancing tolerance: 0.01
- Loss/Cost threshold: 0.2
- Accuracy threshold: 90%

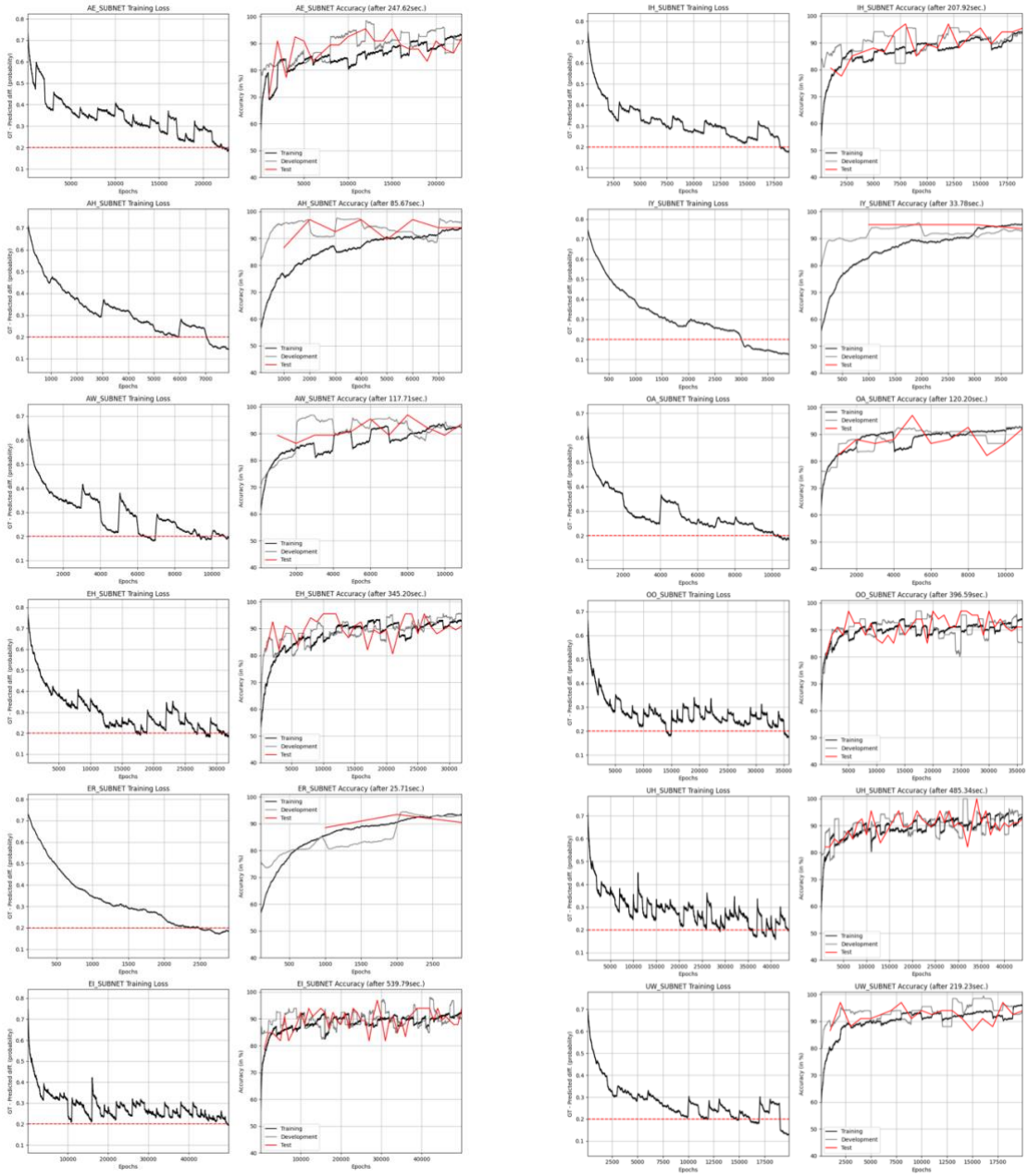


Figure 88 - Giacomelli et al. 3 features OCON model training (losses)

Only a few of MLPs loss functions reach a visible plateau (we can assume that those architectures cannot learn much more from this data) and the same apply for some training accuracy curves.

We observe some peculiar periodicities artifacts in almost any loss curve: taking into account that these plots are graphically smoothed, we can easily locate batch data re-shuffling epoch indices, coinciding with irregularity spikes. The corresponding loss increment or decrease is symptomatic of batch size containing False-labeled samples which are effectively pushing NNs learning (weights update).

The *er* phoneme is unexpectedly so well represented that any one-hot encoding or re-shuffling causes no change in the overall trend of the curve, same for *iy* class: this means that the most under-represented phoneme classes (118 and 124 samples) are instead the best represented: we can confirm that by looking at respective training times.

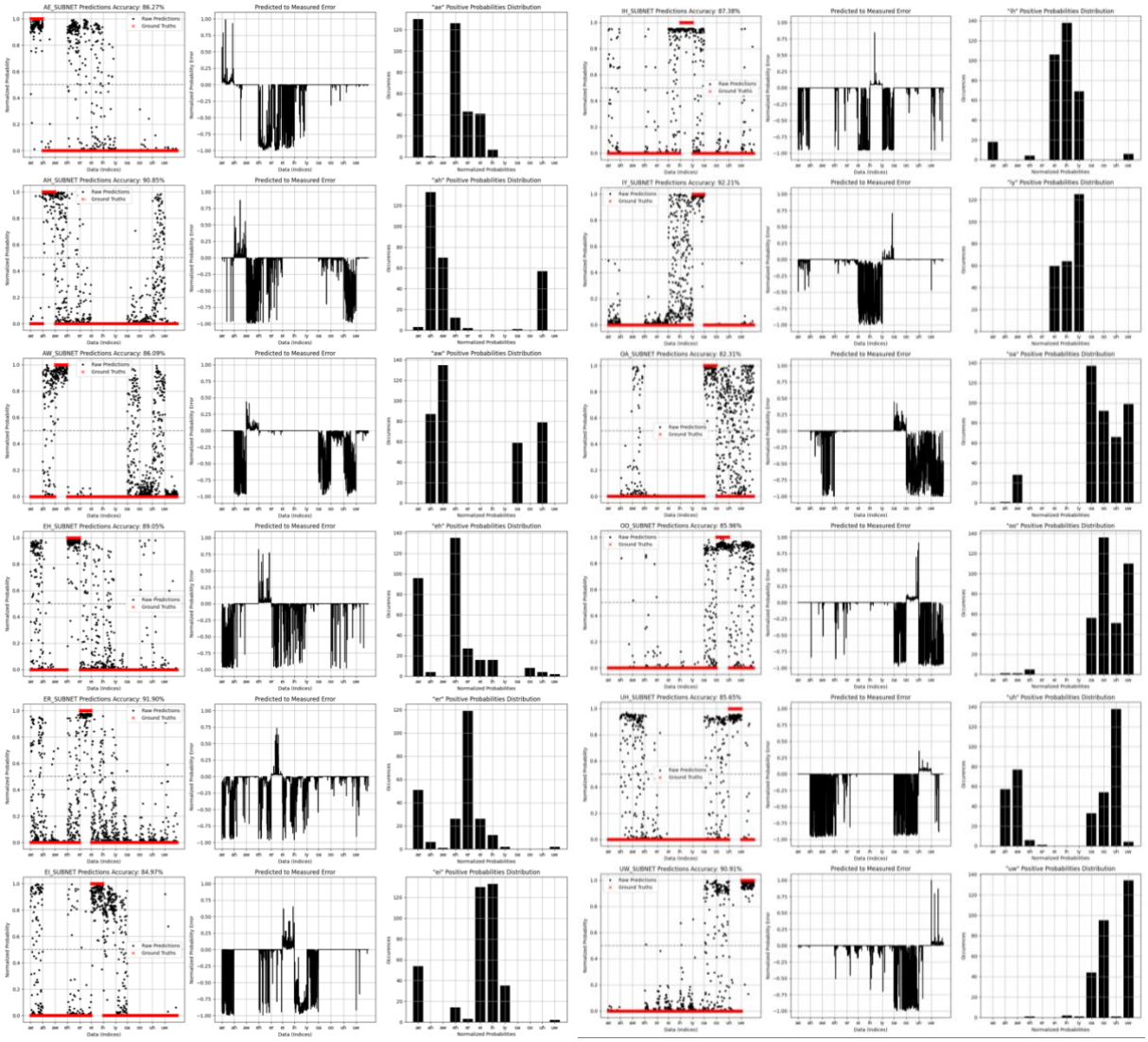


Figure 89 - Giacomelli et al. 3 feature OCON model evaluation

We applied an entire dataset accuracy evaluation to pre-trained one-class architectures, measuring error distance between predicted labels and *ground truths*, binarizing results according to a 0.5 threshold. We can easily see how certain MLPs reached a good boundary segregation, while others didn't complete learning (all True-predictions above 0.95).

The highest error rate is located in classes that are very close from an aural standpoint, for example *ae* and *eh*, and to a lesser extent in *er* or *ei*.
Reported test accuracies are:

Phoneme Class (MLP)	Test Accuracy	Training Time
æ = /ae/	86.27%	247.62 sec.
ɑ = /ah/	90.85%	85.67 sec.
ɔ = /aw/	86.09%	117.71 sec.
ɛ = /eh/	89.05%	345.20 sec.
ɜr = /er/	91.90%	25.71 sec.
eɪ = /ei/	84.97%	539.79 sec.
I = /ih/	87.38%	207.92 sec.
i = /iy/	92.21%	33.78 sec.
o = /oa/	82.31%	120.20 sec.
ʌ = /oo/	85.96%	396.59 sec.
U = /uh/	85.65%	485.34 sec.
u = /uw/	90.91%	219.23 sec.
Average	87.79%	235.40 sec.
OCON Model	70%	

Table 11 - Giacomelli et al. 3 features OCON model accuracies

Our OCON model accuracy is measured per sample, applying MaxNet to intermediate MLP output vectors, evaluating label/*ground truths* identity. We obtained 70% of accuracy indicating that even in the case of actual True-class recognition there exist multiple false positives with respectively higher probabilities, erroneously selected by the output algorithm.

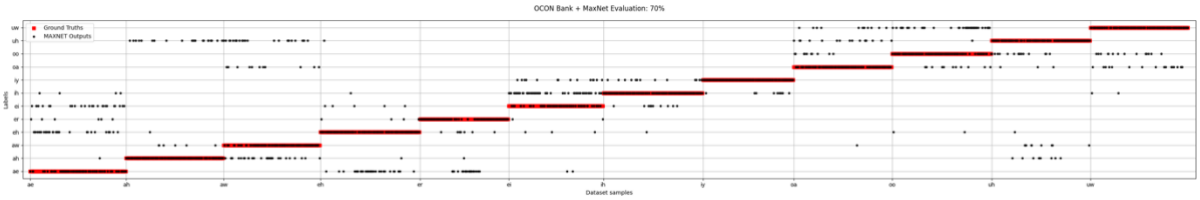


Figure 90 - Giacomelli et al. 3 features OCON model output

We already knew that steady state formants could not lead to great results, but before introducing temporal tracking, we tested for hidden speaker dataset biases. In fact, we know that blind recognition of children's vowels could lead to high error rates and in addition, we know nothing about the age (thus vocal maturity) of children's sub-set.

The 2nd experimental setup is:

- 3 input features: F_1, F_2, F_3 ratios (SS) + 1 encoded label (for each non-child sample, Figure 91);
- intermediate MLP output vector: 12 normalized true-class probabilities
- 1 output prediction: encoded label
- Training epochs: 1000, for each one-hot encoded batch-set (32 samples mini-batch training)
- Data Batch samples balancing tolerance: 0.01
- Loss/Cost threshold: 0.15 (lowered)
- Accuracy threshold: 90%

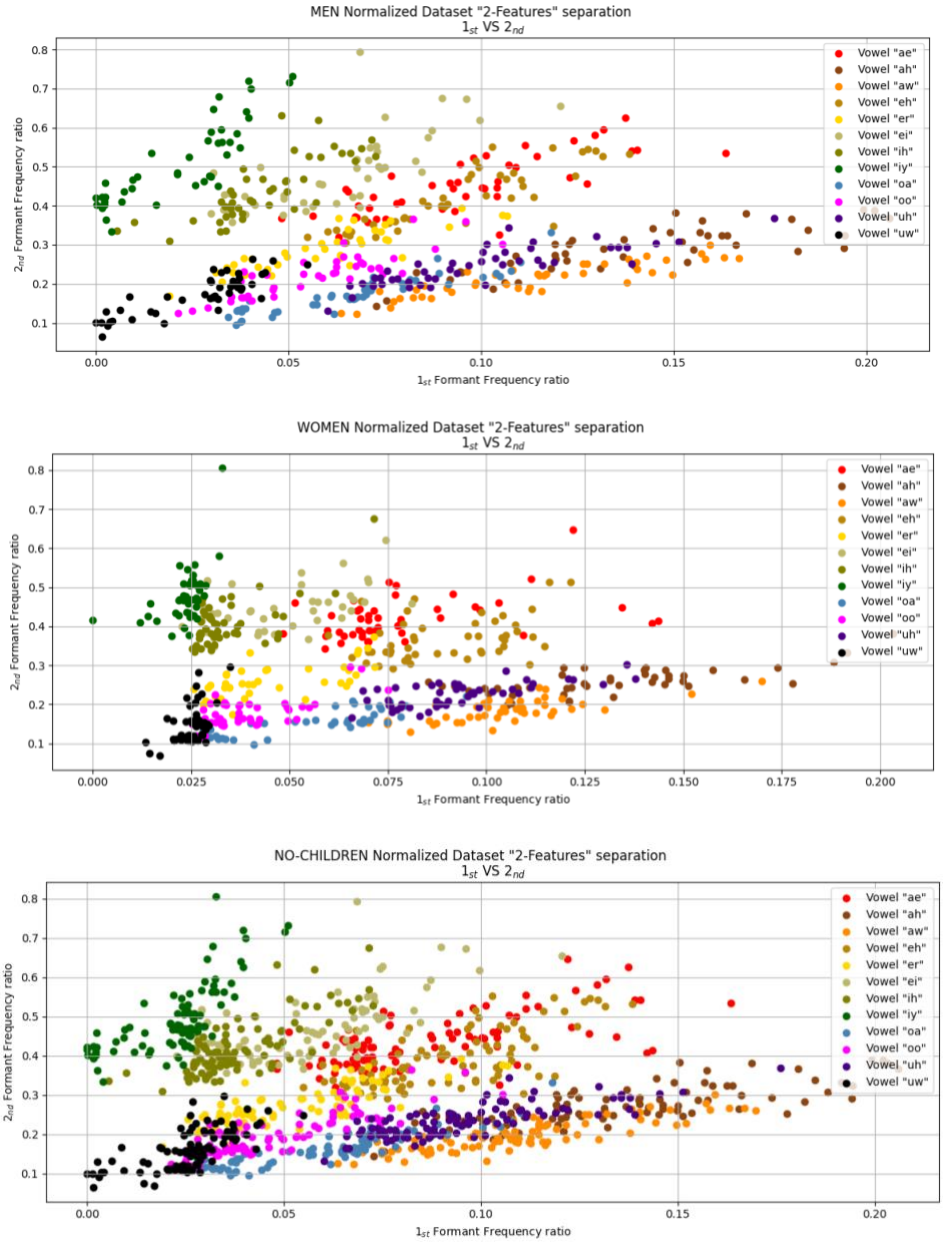


Figure 91 - Giacomelli et al. HGCW dataset (speaker groups)

MLP test accuracies show a small but widespread increase in boundaries efficiency (learning and evaluation graphs in **APPENDIX-B**):

Phoneme Class (MLP)	Test Accuracy	Training Time
æ = /ae/	88.44%	147.16 sec.
ɑ = /ah/	91.83%	82.46 sec.
ɔ = /aw/	86.70%	95.39 sec.
ɛ = /eh/	90.09%	246.88 sec.
ɜr = /er/	94.86%	24.51 sec.
eɪ = /ei/	82.48%	215.12 sec.
I = /ih/	89.17%	139.19 sec.
i = /iy/	94.31%	48.78 sec.
o = /oa/	85.05%	140.39 sec.
ʌ = /oo/	86.79%	124.50 sec.
U = /uh/	85.23%	223.83 sec.
u = /uw/	90.64%	61.63 sec.
Average	88.80%	129.15 sec.
OCON Model	74%	

Table 12 - Giacomelli et al. 3 features (speakers filtered) OCON model accuracies

Training times exception (increasing) could be addressed to lowering of the loss threshold and concurrent insufficient sample representation (due to speaker filtering).

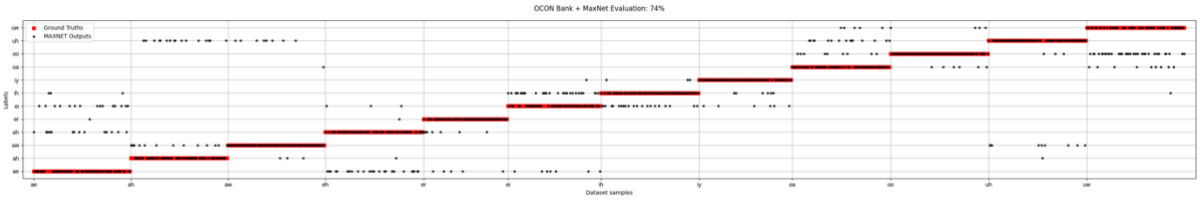


Figure 92 - Giacomelli et al. 3 features (speakers filtered) OCON model output

We can generally confirm that the "children aspect" is an additional difficulty for phoneme recognition (we will examine it deeply in *Speaker Recognition* paragraph) but we can't completely generalize the efficiency increment (except for the model accuracy) because some training times increase and one-class accuracies decrease. We still suffer from too many false positives.
We then tested steady states formant measurements re-adding F_0 features, hoping that it might be enough to restore a proper phonemes categorization, lowering speakers' misunderstandings.

The 3rd experimental setup is:

- 4 input features: F_0 (column min-max normalized) and F_1, F_2, F_3 ratios (SS) + 1 encoded label (for each sample)
- intermediate MLP output vector: 12 normalized true-class probabilities
- 1 output prediction: encoded label
- Training epochs: 1000, for each one-hot encoded batch-set (32 samples mini-batch training)
- Data Batch samples balancing tolerance: 0.01
- Loss/Cost threshold: 0.20
- Accuracy threshold: 90%

Unfortunately, this input dataset didn't improve neither model accuracy nor MLP one-class parameters.

The most performative features set variant is composed by rough formant tracks performed at 10%, 50% and 80% of vowel utterance durations.
The 4th experimental setup is:

- 12 input features: F_1, F_2, F_3 (SS + 10%, 50% and 80% time tracking) + 1 encoded label (for each sample)
- intermediate MLP output vector: 12 normalized true-class probabilities
- 1 output prediction: encoded label
- Training epochs: 1000, for each one-hot encoded batch-set (32 samples mini-batch training)
- Data Batch samples balancing tolerance: 0.01
- Loss/Cost threshold: 0.15 (lowered)
- Accuracy threshold: 95% (increased)

which leads to a significant improvement in learning phase: almost all loss and accuracy curves reach obvious plateaus, thus optimizing classification boundaries; early stopping and re-shuffling side-effects seems to decrease, although still present.

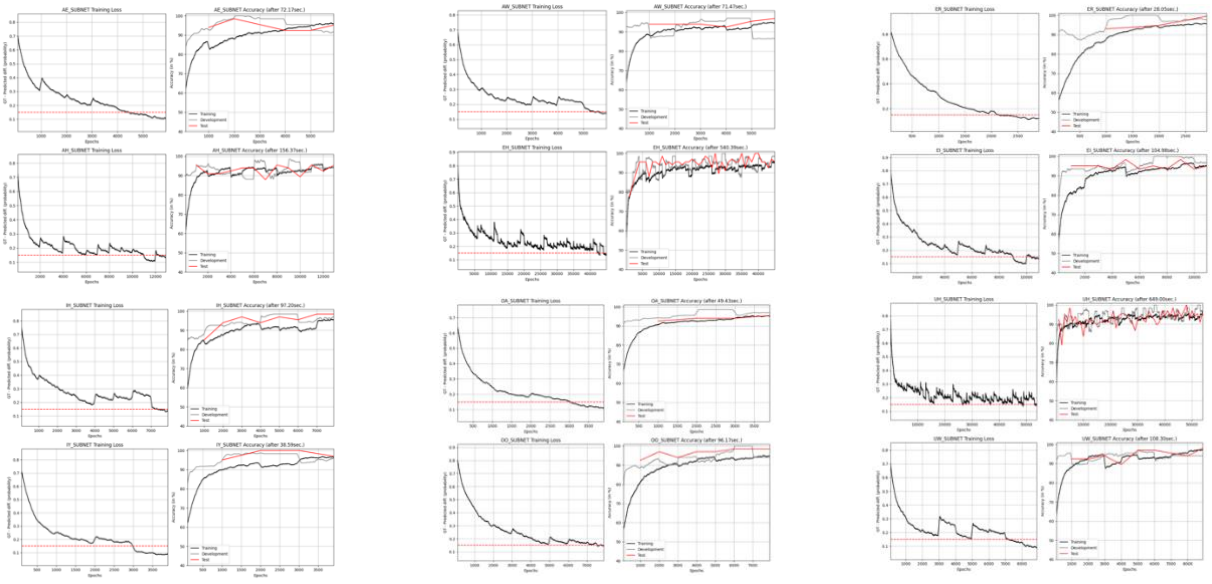


Figure 93 - Giacomelli et al. 3 features (temporal tracking) OCON model training (losses)

The number of false positives dramatically decreases although some uncertainties remain (especially with aurally assimilating classes).
The *eh*-class MLP drastically increases learning time because of the difficulties in reaching the threshold value for cost minimization (although maximizing accuracy fast) while elsewhere we noticed a slight reduction in training times.

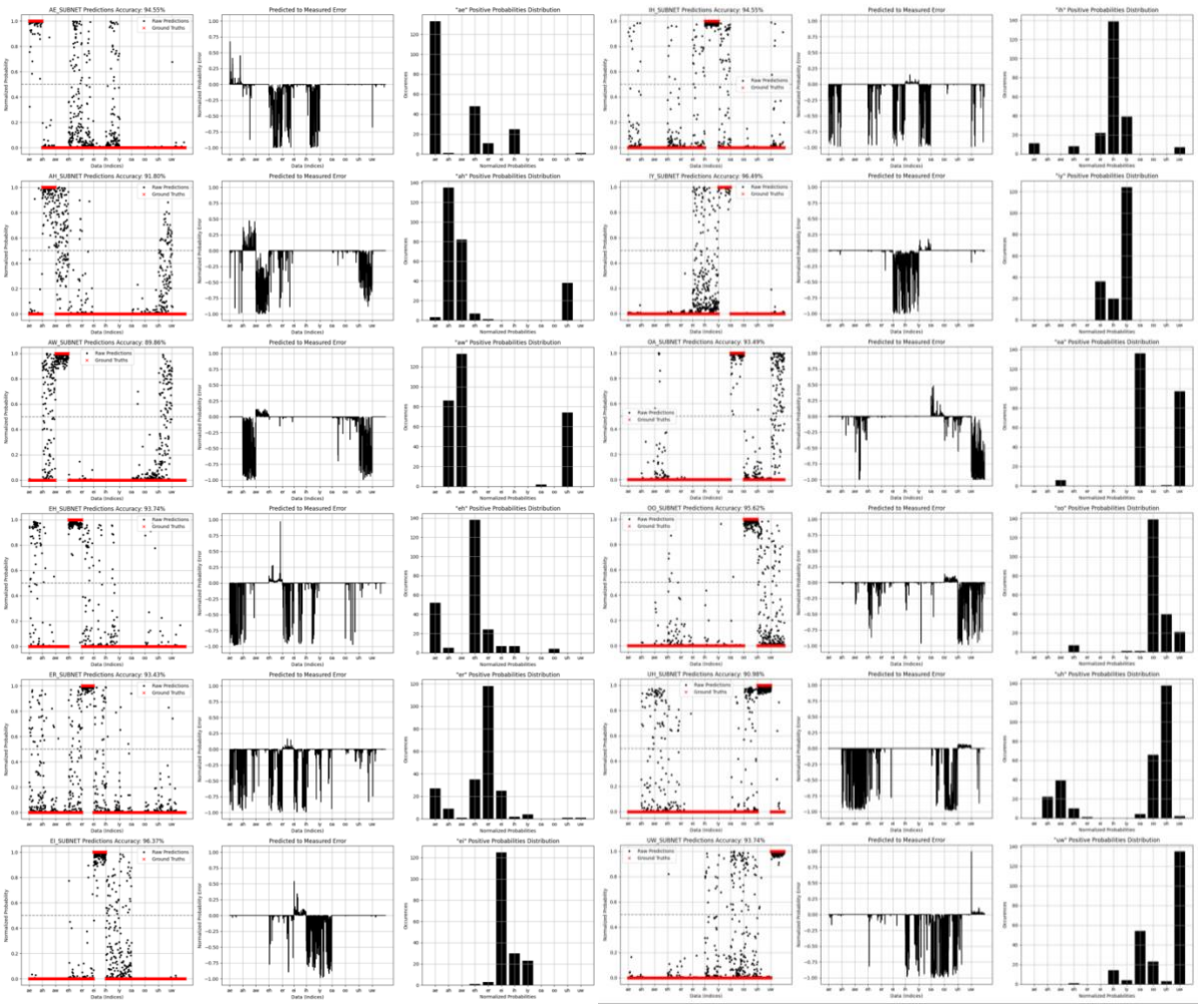


Figure 94 - Giacomelli et al. 3 features (temporal tracking) OCON model evaluation

Resulting test accuracies are:

Phoneme Class (MLP)	Test Accuracy	Training Time
æ = /ae/	94.55%	72.17 sec.
ɑ = /ah/	91.80%	156.37 sec.
ɔ = /aw/	89.86%	71.47 sec.
ɛ = /eh/	93.74%	540.39 sec.
ɜr = /er/	93.43%	28.05 sec.
eɪ = /ei/	96.37%	104.98 sec.
I = /ih/	94.55%	97.20 sec.
i = /iy/	96.49%	38.59 sec.
o = /oa/	93.49%	49.43 sec.
ʌ = /oo/	95.62%	96.17 sec.
U = /uh/	90.98%	649 sec.

u = /uw/	93.74%	108.30 sec.
Average	93.72%	167.68 sec.
OCON Model	90%	

Table 13 - Giacomelli et al. 3 features (temporal tracking) OCON model accuracies

Our OCON model finally exceeds 90% of accuracy, confirming [117] [116] theories about formant tracking requirement for effective speech recognition.

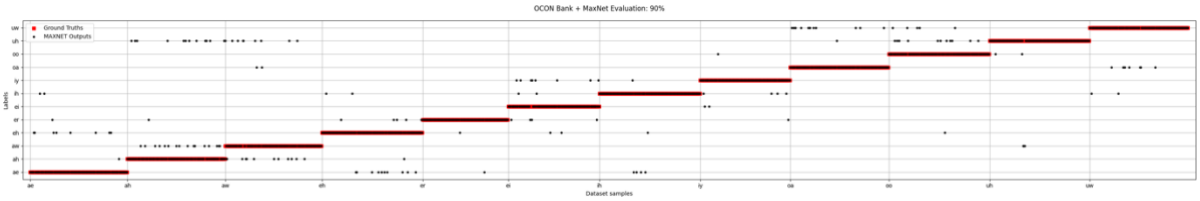


Figure 95 - Giacomelli et al. 3 features (temporal tracking) OCON model output

Speaker Recognition

Speaker recognition was outside the scope of our research but we wanted to test our best model estimate to verify the effective minimum number of formant features required for correct speaker identification.
In this context our OCON model is reduced to 3 MLP binary classifiers, one for each speaker group: *men*, *women* and *children*.
Taking into account previous children subset exclusion benefits (*Phoneme Recognition*) we decided to extend the input feature set, newly including F_0 to foster more effective classification.
The last (5^{th}) experimental setup is:

- 13 input features: F_0 (column min-max normalized) and F_1, F_2, F_3 ratios (SS + 10%, 50% and 80% time tracking) + 1 speaker encoded label (for each sample)
- intermediate MLP output vector: 3 normalized true-class probabilities
- 1 output prediction: encoded label

- Training epochs: 1000, for each one-hot encoded batch-set (32 samples mini-batch training)
- Data Batch samples balancing tolerance: 0.01
- Loss/Cost threshold: 0.36, 0.08, 0.45
- Accuracy threshold: 80%, 97%, 80%

In this test we have class-dependent Early stopping loss and accuracy thresholds, because it was empirically found that these values needed to be dynamically modulated in order for the classifiers to converge to effective solutions (avoiding infinite training loops).

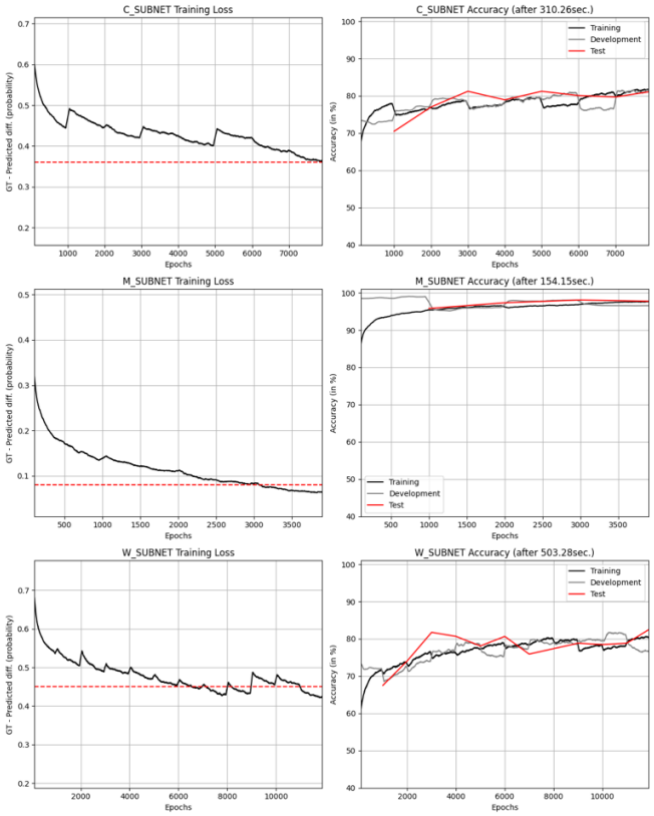


Figure 96 - Giacomelli et al. 3 features (temporal tracking) OCON model training (speaker recognition)

The training phase is visibly incomplete, but the accuracy curves seem to approach plateaus with varying degrees of affinity; *women* and *children* classes face greater difficulty in error minimization while *men* MLP rapidly converge to a low error, approaching 100% accuracy: this could be symptomatic of a better representation for male group but also of known difficulties in separating children's voices from certain adult female voices at blind-testing.

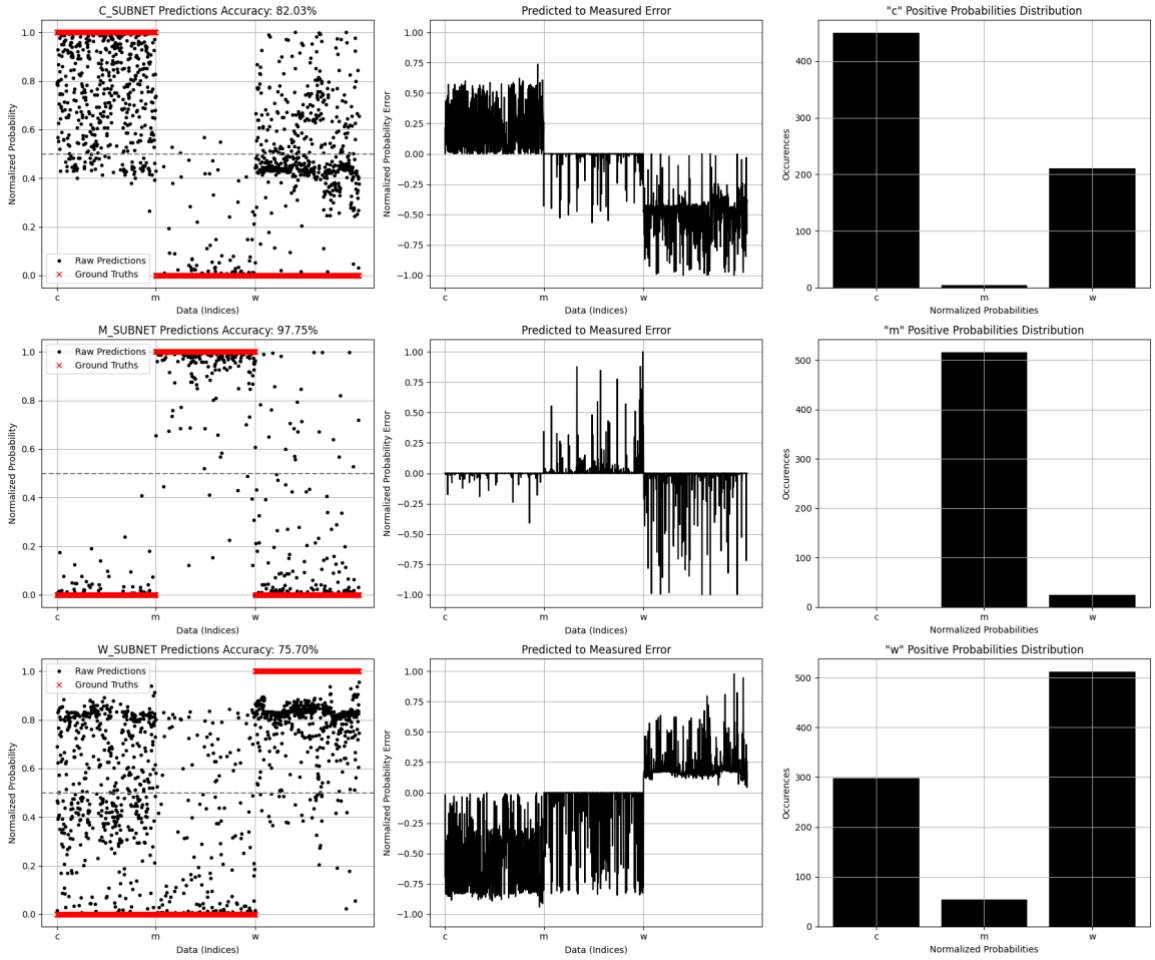


Figure 97 - Giacomelli et al. 3 features (temporal tracking) OCON model evaluation (speaker recognition)

Our assumptions are confirmed by evaluating the entire dataset: *male* class has the lower false positives rate and almost maximizes probability computation, while *children* and *women* are affected by higher false positive rates and are incapable of computing well-separated probabilities, even for samples closely true-class matching samples set.
Reported test accuracies are:

Speaker Class (MLP)	Test Accuracy	Training Time
Children	82.03%	310.26 sec.
Men	97.75%	154.15 sec.
Women	75.70%	503.28 sec.
Average	85.15%	322.56 sec.
OCON Model	80%	

Table 14 - Giacomelli et al. 3 features (temporal tracking) OCON model accuracies (speaker recognition)

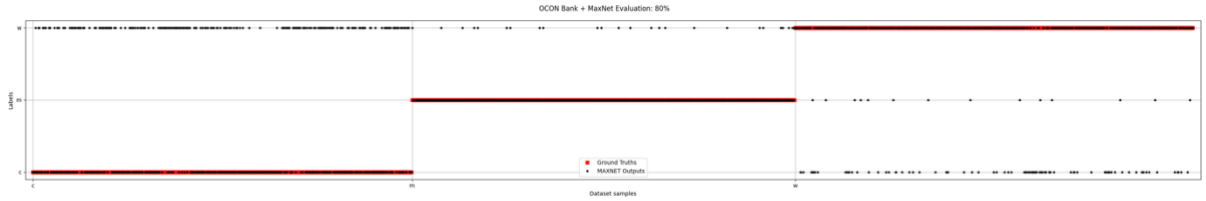


Figure 98 - Giacomelli et al. 3 features (temporal tracking) OCON model output (speaker recognition)

Our OCON model hovers around 80% to 85% of accuracy for *speaker recognition,* with 3 formants 3-points time tracking input features.
These results suggest that better accuracy could be obtained with the use of a larger number of formants, as, for instance, has long been done in the forensic field for speaker identification.

CONCLUSIONS & FUTURE WORKS

First of all, we would like to clarify some ethical, but also technological aspects that generally affect the understanding of DL research.

Big data availability is not to be taken for granted: we collected a fair amount of jagged and open available information about vowel datasets, due as much to a lack of prior knowledge (on our part) as to the structuring of data sharing. It is no coincidence that the best performing ASR systems belong to industries with dedicated and proprietary data infrastructures. Such expertise in data collection and processing is a key aspect of the success of their systems and now we can easier understand why. These companies often provide simplified tools or surrogate sets of their datasets, in order to apparently compensate for their lack of open contribution toward research, but likewise to acquire additional feedback and expand their datasets too. Big data are not open yet! For this purpose, we prefer to make a small contribution (to complement our work) including a short list of pointers to open (and non-open) data resources for ASR:

- https://www.openslr.org/index.html
- https://paperswithcode.com/search?q_meta=&q_type=&q=speech+dataset
- https://stagezero.ai/off-the-shelf-speech-datasets/
- https://www.magicdatatech.com/datasets/asr
- https://huggingface.co/datasets?task_categories=task_categories:automatic-speech-recognition&sort=trending
- https://www.openml.org/search?type=data&status=active
- https://www.kaggle.com/datasets?search=vowel
- https://github.com/Open-Speech-EkStep/ULCA-asr-dataset-corpus
- https://github.com/coqui-ai/open-speech-corpora
- https://github.com/jim-schwoebel/voice_datasets
- https://github.com/double22a/speech_dataset (*rev. 08/07/2023)*

Comparing best literature and our phoneme recognition results:

Authors	Algorithm	AVG. Accuracy (dataset + features)
Syrdal, A. K. et al.	LDA	**85.9%** (PB + 3 formants)
Hillenbrand, J. et al.	QDA	**91.8%** (HGCW – PB + 3 formant distances)
Nearey, T. M. et al.	GLM	**89.5%** (PB + 3 formants)
Bird, J.J. et al.	HMM	**86.23%** (Unknown + MFCC)
Giacomelli, S. et al.	MLP (OCON)	**93.7%** (HGCW + 3 formant ratios)
Sridhar, C. et al.	CNN + LSTM	**94% - 98.6%** (MNIST English digits + spectral/cepstral features)

Table 15 - Giacomelli et al. references accuracies review

We listed only the best retrieved result for each algorithm class. Our MLP-derived model seems to approach CNN-derived effectiveness, at the cost of more learning iterations but with clear computational distribution opportunities (software and hardware).

It is important to stress that our intent was mainly to approach DL research with a practical example of application, in order to better understand the complexity of the topic and the tools. We ended up with a performative solution which in turn describes an original *all-in-one* task approach, because formant estimation (LPA), features processing (normalization, abstraction) and classification algorithms can all be effectively implemented by similar MLP architectures and tensors processing.

What may really be of interest is not our result (which may still be biased, to certain extents) but our rigorous methodology, consisting of structured:

1) dataset/results/methods documentation;
2) problem-specific features retrieving;
3) data normalization and statistical analysis;
4) classifier and model studies and optimization;
5) computational optimization approach;

Each experiment is fully replicable and represents just a single run of a performant technique to find optimal parameter estimates for time-efforts limited framework contexts[27]. Repeating and/or narrowing parameters research would lead to better estimates and hopefully to better results.

For what we have learned about DL and NNs functioning [152], each auxiliary tool (dataset filtering, pre-processing, normalization factors etc.) must be effectively included in a production proposal, in order to hopefully approach the same development results[28] for changing data sources: as we can see, methods and decisions also represent computational biases for our solution, but the overall representation of the development context is the only way to effectively validate our tool. And we are still simply approaching biases relevancy:

- What do we know about dataset speaker's dialects/pronunciation peculiarities? And what about ethnic or socio-cultural diversification? Researchers and data collectors should account for this kind of ASR well-known biases, hence its interdisciplinary field definition;
- How many datasets and variants should be tested before being ready for a production phase submission or a scientific/artistic documentation? As already analyzed at the beginning of the chapter, we should gradually deal with the analysis and filtering of the available data in terms of features and labels structuration so as to speed up and focus research activities only on classification or regression problems solving: these are quite common and still unanswered issues in today's DL research;

[27] The most common working conditions.

[28] Or rather, to maintain the same operational basis.

Focusing on our experimental proposal, some technical improvements are feasible including:

- multiple Early stopping criteria refinement with MLP class-specific studies or dynamic threshold modulation via errors tracking and parameters adaptation;
- multiple dataset evaluation (PB, TIMIT, VTR-TIMIT, VocalSet etc.) with comparison results;
- more comprehensive output quality listing: *F1 scoring*, *Precision*, *Recall*, *Confusion Matrix* to better analyze false positives correlations and class-specific efficiency;
- output label selection algorithm could be improved too: instead of computing outputs by means of simply selecting the highest normalized probability, we could adopt a cost-derivative approach:
    1) evaluate multiple high-values probability conditions within a small tolerance arbitrary threshold (10^{-1} or 10^{-2});
    2) if (1) is True, evaluate training loss curve computing derivatives within a specified epochs interval.

The OCON model can choose the output with the highest probability corresponding to the MLP architecture that has minimized its associated cost function for the longest time ($d\mathcal{L} = 0$)[29]. This approach comes at the cost of additional memory to store training data with the required depth for each subnet.

Both *speaker* and *phoneme* classification outputs (12-valued intermediate vectors) could be finally coupled by means of another custom MLP architecture (to be studied [153]) in order to hypothetically learn high-level relationships between phoneme-speaker pairs, optimizing the results of both models.

We may update and extend our introductory audio processing chain (Figure 3) defining an *all-in-one* MLP-derived approach:

- LPA analysis performed with a single Perceptron algorithm, could feed both a formant estimation[30] and reflection coefficients (a_k) or signal predictions computation algorithms (both GPU-tensors optimizable [154]);
- *Autoencoders* may be involved for simple phonemic audio re-synthesis.

Autoencoders (AE) are just another MLP-variant architecture, commonly applied for data compression and transmission (*encoding* and *decoding*). Their structure can be generally assimilated to a standard

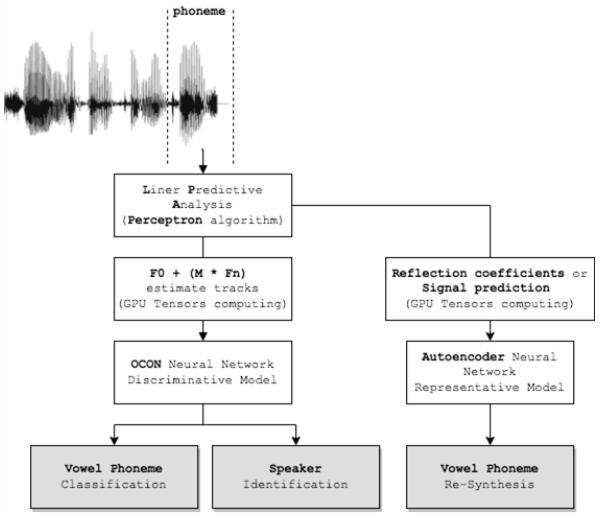


Figure 99 - Vowel phoneme processing chain

[29] As a synonym for stability achieved in the learning phase.

[30] Some references to Praat-to-Python scripting (formant estimation) are [158] [160] [159].

MLP (*encoder*) followed by an inverted MLP structure (*decoder*), fully connected in a so-called *latent* layer: usually designed to be composed of as few nodes as possible. Training and evaluation phases remain the same, except that we train the model to optimize regression operations (by MSE/MAE formulas) to reproduce exactly the input features.

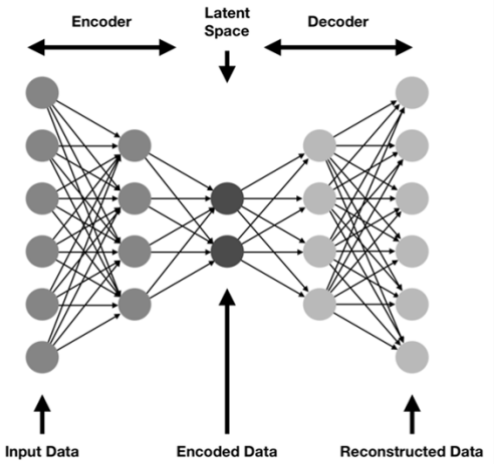


Figure 100 - Autoencoder architecture example

AE and probabilistic variants (Variational-AE) are proved to be efficient tools for audio and audio-video re-synthesis [155] [156], given spectral [157] or time-domain samples coefficients (LPCs too). In order to account for temporal information, finer features tracking would be required and features should be flattened vertically (Figure 100) and properly restored during re-synthesis (i.e.: spline or parabolic interpolation).

This chain could form a unified framework for the analysis and re-synthesis of vowel phonemes.

In the context of music applications, we are aware of methods of controlling speech spectra (plosives, fricatives, sibilants) by specific dynamics compression techniques, designed for consonants only (noise monitoring). Vowel "resonances" are the core elements on which singers articulate multiple embellishment techniques, that especially in *tenuto* conditions, can be traced to continuous vowels phonemic transitions.

In this context, we can easily imagine a dynamics-EQ profiler which holistically analyzes speech content to apply compression techniques, depending on consonant or vocalic nature of the signal. Vowel phonemes could be equalized by tracing their actual formants value, interpolating micro-articulations by their class formants "mean"[31] values.

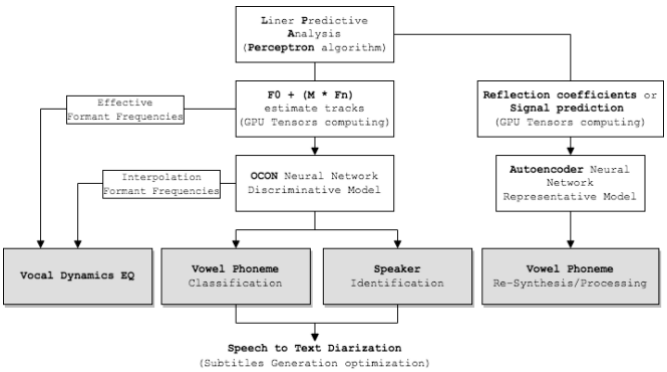


Figure 101 - Vowel phonemes unified analysis/re-synthesis framework

In *real-time* applications, the same algorithm could be applied (after proper optimization) for sound monitoring or to optimize pre-existing automatic subtitles generation processes (*speech-to-text diarization*).

Many contemporary artists could benefit from our methodology and tools to focus on creative applications of vowels resonance analysis (i.e.: alternative *score following* methods), in the same way as famous authors like John M. Chowning (1934 -), Barry Truax (1947 -) or Luciano Berio (1925 – 2003) among others already did in the past, with renewed technical tools and hopefully interests.

[31] Here we mean that: we can statistically compute actual effective interpolation values from recognized phoneme class. Same happens for transitions to assimilable phoneme classes.

REFERENCES

[1] Wikipedia, "Speech Recognition (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Speech_recognition. [Accessed 19 June 2023].

[2] Wikipedia, "Dynamic Bayesian Network (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Dynamic_Bayesian_network. [Accessed 20 June 2023].

[3] R. Reddy, "Raj Reddy Profile (Carnegie Mellon University)," [Online]. Available: http://www.rr.cs.cmu.edu/. [Accessed 20 June 2023].

[4] C. P. Browman and L. Goldstein, "Articulatory Phonology: An Overview," *Phonetica,* vol. 49, no. 3 - 4, pp. 155 - 180, 5 January 1992.

[5] J. Meyer, L. Rauchenstein, J. D. Eisenberg and N. Howell, "Artie Bias Corpus: An Open Dataset for Detecting Demographic Bias in Speech Applications," *Proceedings of the Twelfth Language Resources and Evaluation Conference,* pp. 6462 - 6468, 2020.

[6] K. Livescu and J. Glass, "Feature-based Pronunciation Modeling for Speech Recognition," *Proceedings of HLT-NAACL 2004: Short Papers,* pp. 81 - 84, 2004.

[7] R. Prabhavalkar, T. Hori, T. N. Sainath, R. Schlüter and S. Watanabe, "End-to-End Speech Recognition: A Survey," *IEEE/ACM Transactions on Audio, Speech, and Language Processing,* pp. 1 - 27, 03 March 2023.

[8] I. Papastratis, "Speech Recognition: a review of the different deep learning approaches," 2021. [Online]. Available: https://theaisummer.com/speech-recognition/. [Accessed 19 June 2023].

[9] J. O. Smith, Physical Audio Signal Processing, 2010.

[10] J. O. Smith, Spectral Audio Signal Processing, 2010.

[11] J. O. Smith, Introduction to Digital Filters with Audio Applications, 2007.

[12] R. J. McAulay and T. F. Quatieri, "Speech Analysis/Synthesis Based on a Sinusoidal Representation," *IEEE Transactions on Acoustics, Speech and Signal Processing (ASSP),* vol. 34, no. 4, pp. 744 - 754, 1986.

[13] C. A. Miller, "Pronunciation Modeling In Speech Synthesis," 1998.

[14] D. Jurafsky, Martin and J. H., Speech and Language Processing: An introduction to Natural Language Processing, UCSD EDU, 2007.

[15] D. Jurafsky and J. H. Martin, "Speech and Language Processing (3rd ed. draft)," [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/. [Accessed 20 June 2023].

[16] L. R. Rabiner and R. W. Schafer, Introduction to Digital Speech Processing, vol. 1, R. M. Gray, Ed., Hanover, MA: now Publishers Inc., 2007.

[17] L. R. Rabiner and R. W. Schaefer, "Introduction to Digital Speech Processing," *Foundations and Trends in Signal Processing,* vol. 1, no. 1-2, pp. 1-194, 2007.

[18] J. L. Flanagan, Speech Analysis, Synthesis and Perception, 2nd ed., Heidelberg, Berlin: Springer-Verlag, 1972.

[19] D. O'Shaughnessy, Speech Communications, New York, NY: IEEE Press Marketing, 2000.

[20] V. Pulkki and M. Karjalainen, Communication Acoustics: An Introduction to Speech, Audio and Psychoacoustics, John Wiley & Sons, 2015.

[21] C. G. M. Fant, Acoustic Theory of Speech Production, De Gruyter Mouton, 1971.

[22] R. R. Riesz and G. W. Burchett, "Artificial Larynx". USA Patent US1910966A, 23 05 1929.

[23] H. Barney, F. Haworth and H. Dunn, "An experimental transistorized artificial larynx," *The Bell System Technical Journal,* vol. 38, no. 6, pp. 1337 - 1356, November 1959.

[24] F. F. Li and T. J. Cox, Digital Signal Processing in Audio and Acoustical Engineering, T. &. Francis, Ed., Boca Raton: CRC Press, 2019.

[25] Wikipedia, "International Phonetic Alphabet," [Online]. Available: https://en.wikipedia.org/wiki/International_Phonetic_Alphabet. [Accessed 10 June 2023].

[26] IPA, "International Phonetic Association Homepage," [Online]. Available: https://www.internationalphoneticassociation.org/. [Accessed 11 June 2023].

[27] Wikipedia, "Daniel Jones (Phonetician)," [Online]. Available: https://en.wikipedia.org/wiki/Daniel_Jones_(phonetician). [Accessed 11 June 2023].

[28] D. Jones, An English Pronouncing Dictionary, 5th ed., New York: E. P. Dutton & Company, inc., 1943.

[29] P. Ladefoged and K. Johnson, A Course in Phonetics, 7th ed., Wadsworth, 2015.

[30] K. Johnson, "A Course in Phonetics (Homepage)," [Online]. Available: https://linguistics.berkeley.edu/acip/. [Accessed 11 June 2023].

[31] P. Ladefoged and I. Maddieson, The Sounds of The World's Languages, W. -. Blackwell, Ed., John Wiley and Sons Ltd, 1996.

[32] U. D. o. Defense, "DARPA.Researches," [Online]. Available: https://www.darpa.mil/our-research. [Accessed 11 June 2023].

[33] Wikipedia, "ARPABET," [Online]. Available: https://en.wikipedia.org/wiki/ARPABET. [Accessed 11 June 2023].

[34] I. Z. Skordilis, A. Toutios, J. Töger and S. Narayanan, "Estimation of vocal tract area function from volumetric Magnetic Resonance Imaging," *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP),* pp. 924 - 928, 2017.

[35] M. Echternach, J. Sundberg, T. Baumann, M. Markl and B. Richter, "Vocal tract area functions and formant frequencies in opera tenors' modal and falsetto registers," *Journal of Acoustical Society of America,* vol. 129, no. 6, pp. 3955 - 3963, June 2011.

[36] L. Goldstein, "Signal Analysis and Interpretation Laboratory (SAIL)," 28 November 2022. [Online]. Available: https://sail.usc.edu/~lgoldste/General_Phonetics/index.html. [Accessed 12 June 2023].

[37] J. H. Esling, "The IPA Categories “Pharyngeal” and “Epiglottal”: Laryngoscopic Observations of Pharyngeal Articulations and Larynx Height," *Language and Speech,* vol. 42, no. 4, pp. 349 - 372.

[38] J. H. Esling, "There Are No Back Vowels: The Larygeal Articulator Model," *Canadian Journal of Linguistics/Revue Canadienne De Linguistique,* vol. 50, no. 1 - 4, pp. 13 - 44.

[39] J. H. Esling, "The laryngeal articulator’s influence on voice quality and vowel quality," *Social and Biological Factors in Speech Variations (Studi AISV),* vol. 3, pp. 13 - 26, 31 December 2017.

[40] J. C. Catford, A Practical Introduction to Phonetics, New York, NY: Oxford University Press, 1988.

[41] Y. S. J. a. W. J. Swerdlin, "The Effect of Whisper and Creak Vocal Mechanisms on Vocal Tract Resonances," *Journal of The Acoustical Society of America,* vol. 127, pp. 2590 - 2598, 2010.

[42] F. J. Harris, "On the Use of Windows for Harmonic Analysis with The Discrete Fourier Transform," *Proceedings of The IEEE,* vol. 66, no. 1, pp. 51 - 83, January 1978.

[43] D. L. Jones and I. Selesnick, The DFT, FFT and Practical Spectral Analysis, OpenStax-CNX, 2010.

[44] A. H. Nuttall, "Some Windows with Very Good Sidelobe Behavior," *IEEE Transactions on Acoustics, Speech and Signal Processing (ASSP),* vol. 29, no. 1, pp. 84 - 91, February 1981.

[45] K. M. M. Prabhu, Window Functions and their Applications in Signal Processing, New York, NY: Taylor & Francis, CRC Press, 2014.

[46] D. M. Howard and J. Angus, Acoustics and Psychoacoustics, T. &. Francis, Ed., New York, NY: Routledge, 2017.

[47] M. J. Ross, H. L. Shaffer, A. Cohen, R. Freudberg and H. J. Manley, "Average Magnitude Difference Function Pitch Extractor," *IEEE Transactions on Acoustics, Speech and Signal Processing (ASSP),* vol. 22, no. 5, pp. 352 - 362, October 1974.

[48] S. Giacomelli, "Alcuni modelli e tecniche per l’analisi computazionale MIR," L'Aquila, 2022.

[49] S. Giacomelli, "Analisi comparativa dei modelli e dei descrittori documentati per la risposta acustica degli spazi di ascolto," L'Aquila, 2021.

[50] R. C. Maher and J. W. Beauchamp, "Fundamental Frequency Estimation of Musical Signals Using a Two-Way Mismatch Procedure," *Journal of Acoustical Society of America,* vol. 95, pp. 2254 - 2263, 1994.

[51] B. P. Bogert, J. R. Healy and J. W. Tukey, "The Quefrency Alanysis of Time Series for Echoes: Cepstrum, Pseudo-Autocovariance, Cross-Cepstrum, and Saphe Cracking," in *Proceedings of the Symposium on Time Series Analysis*, 1963.

[52] J. O. Smith, "Spectral Envelope Extraction," [Online]. Available: https://www.dsprelated.com/freebooks/sasp/Spectral_Envelope_Extraction.html. [Accessed 11 07 2023].

[53] A. V. Oppenheim, "Superposition in a Class of Non-Linear Systems," MIT Press, Cambridge, 1965.

[54] A. V. Oppenheim, R. W. Schafer and T. G. Stockham, "Nonlinear Filtering of Multiplied and Convolved Signals," *Proceedings of the IEEE,* vol. 56, no. 8, pp. 1264 - 1291, 1968.

[55] A. V. Oppenheim and R. W. Schafer, "From Frequency to Quefrency: A History of the Cepstrum," *IEEE Signal Processing Magazine,* pp. 95 - 106, September 2004.

[56] A. E. Rosenberg, "Effect of Glottal Pulse Shape on the Quality of Natural Vowels," *Journal of The Acoustical Society of America,* vol. 49, pp. 583 - 590, 1971.

[57] A. M. Noll, "Short-Time Spectrum and “Cepstrum” Techniques for Vocal-Pitch Detection," *Journal of The Acoustical Society of America,* vol. 36, pp. 296 - 302, 1964.

[58] A. M. Noll, "Cepstrum Pitch Determination," *Journal of The Acoustical Society of America,* vol. 41, pp. 293 - 309, 1967.

[59] B. S. Atal, "The History of Linear Prediction," *IEEE Signal Processing Magazine,* vol. 23, no. 2, pp. 154 - 161, March 2006.

[60] T. Bäckström, O. Räsänen, A. Zewoudie, P. P. Zarazaga, L. Koivusalo, S. Das, E. G. Mellado, M. B. Mansali and D. Ramos, "Introuction to Speech processing (online_book)," Aalto University, [Online]. Available: https://speechprocessingbook.aalto.fi/Representations/Linear_prediction.html. [Accessed 22 June 2023].

[61] J. Makhoul, "Linear Prediciton: A Tutorial Review," *Proceedings of The IEEE,* vol. 63, pp. 561 - 580, 1975.

[62] J. Markel and A. Gray, "On Autocorrelation Equations as Applied to Speech Analysis," *IEEE Transactions on Audio and Electroacoustics,* vol. 21, no. 2, p. 1973, 1973.

[63] Wikipedia, "Levinson recursive algorithm (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Levinson_recursion. [Accessed 22 June 2023].

[64] P. A. Morettin, "The Levinson Algorithm and Its Applications in Time Series Analysis," *International Statistical Review,* vol. 52, no. 1, pp. 83 - 92, April 1984.

[65] Wikipedia, "Tom Mitchell (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Tom_M._Mitchell. [Accessed 23 June 2022].

[66] C. M. University, "Tom Mitchell CMU Official Page," [Online]. Available: http://www.cs.cmu.edu/~tom/. [Accessed 23 June 2023].

[67] Wikipedia, "Arthur Samuel (Computer scientist)," [Online]. Available: https://en.wikipedia.org/wiki/Arthur_Samuel_(computer_scientist). [Accessed 11 07 2023].

[68] I. Goodfellow, Y. Bengio and C. A., Deep Learning Book, MIT Press, 2016.

[69] M. Nielsen, Neural Networks and Deep Learning, 2019.

[70] W. S. McCulloch and W. Pitts, "A Logical Calculus of The Ideas Immanent in Nervous Activity," *The bulletin of mathematical biophysics,* vol. 5, pp. 115 - 133, 1943.

[71] D. O. Hebb, The Organization of Behavior: a Neuropsychological Theory, P. Press, Ed., New York, NY: Taylor and Francis, 2002.

[72] Wikipedia, "Artificial Neuron (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Artificial_neuron. [Accessed 13 June 2023].

[73] Wikipedia, "Perceptron (Wikipedia)," Wikipedia (The Free Encyclopedia), [Online]. Available: https://en.wikipedia.org/wiki/Perceptron. [Accessed 14 June 2023].

[74] F. Rosenblatt, "The Perceptron: a Probabilistic Model for Information Storage and Organization in The Brain," *Psychological Review,* vol. 65, no. 6, pp. 386 - 408, 1958.

[75] F. Rosenblatt, "Principles of Neurodynamics: Perceptrons and The Theory of Brain Mechanisms," Buffalo (NY), 1961.

[76] M. L. Minsky and S. A. Papert, Perceptrons: an Introduction to Computational Geometry, MIT Press, 1969.

[77] D. E. Rumelhart, G. E. Hinton and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature,* vol. 323, pp. 533 - 536, 09 October 1986.

[78] Y. LeCun, "A Theoretical Framework for Back-Propagation," *Proceedings of the 1988 Connectionist Models Summer School (CMU),* pp. 21 - 28, 1988.

[79] Wikipedia, "Logistic Regression (Wikipedia)," Wikipedia, The Free Encyclopedia, [Online]. Available: https://en.m.wikipedia.org/wiki/Logistic_regression. [Accessed 14 June 2023].

[80] Wikipedia, "Cross-Entropy [Information Theory] (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Cross_entropy. [Accessed 14 June 2023].

[81] J. Berkson, "Application of the Logistic Function to Bio-Assay," *Journal of the American Statistical Association,* vol. 39, no. 227, pp. 357 - 365.

[82] Wikipedia, "Bernoulli Distribution (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Bernoulli_distribution. [Accessed 14 June 2023].

[83] K. He, X. Zhang, S. Ren and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," *2015 IEEE International Conference on Computer Vision (ICCV),* pp. 1026 - 1034, 2015.

[84] S. Yadav, "Weight Initialization Techniques in Neural Networks," [Online]. Available: https://towardsdatascience.com/weight-initialization-techniques-in-neural-networks-26c649eb3b78. [Accessed 11 07 2023].

[85] P. Ouannes, "How to initialize deep neural networks? Xavier and Kaiming initializatio," [Online]. Available: https://pouannes.github.io/blog/initialization/. [Accessed 11 07 2023].

[86] K. Hornik, M. Stinchcombe and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks,* vol. 2, no. 3, pp. 359 - 366, 09 March 1989.

[87] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems,* vol. 2, pp. 303 - 314, 1989.

[88] Wikipedia, "Universal Approximation Theorem (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Universal_approximation_theorem. [Accessed 15 June 2023].

[89] J. Brownie, "MachineLearningMastery.com," [Online]. Available: https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/. [Accessed 11 07 2023].

[90] A. R. Barron, "Universal Approximation Bounds for Superpositions of a Sigmoidal Function," *IEEE Transactions on Information Theory,* vol. 39, no. 3, pp. 930 - 945, May 1993.

[91] G. F. Montufar, R. Pascanu, K. Cho and Y. Bengio, "On the Number of Linear Regions of Deep Neural Networks," *Advances in Neural Information Processing Systems 27 (NIPS 2014),* vol. 27, 2014.

[92] Wikipedia, "Gottfried Wilhelm (von) Leibniz (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Gottfried_Wilhelm_Leibniz. [Accessed 15 June 2023].

[93] J. Schmidhuber, "Who Invented Backpropagation?," IDSIA.ch, [Online]. Available: https://people.idsia.ch/~juergen/who-invented-backpropagation.html. [Accessed 11 07 2023].

[94] D. R. Wilson and T. R. Martinez, "The General Inefficiency of Batch Training for Gradient Descent Learning," *Neural Networks,* vol. 16, no. 10, pp. 1429 - 1451, 2003.

[95] J. Duchi, E. Hazan and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research,* vol. 12, pp. 2121 - 2159, 2011.

[96] T. Tieleman and G. Hinton, "RMSProp - Divide the Gradient by a Running Average of Its Recent Magnitude," *COURSERA: Neural Networks for Machine Learning,* vol. 4, pp. 26 - 31, 2012.

[97] D. Kingma and L. Ba, "Adam: A Method for Stochastic Optimization," in *International Conference on Learning Representations (ICLR - 2015)*, 2015.

[98] Wikipedia, "Ridge Regression (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Ridge_regression. [Accessed 17 June 2023].

[99] A. E. Hoerl and R. W. Kennard, "Ridge Regression: Biased Estimation for Nonorthogonal Problems," *Technometrics,* vol. 12, no. 1, pp. 55 - 67, 1970.

[100] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: a Simple Way to Prevent Neural Networks from Overfitting," *The Journal of Machine Learning Research,* vol. 15, no. 1, pp. 1929 - 1958, 2014.

[101] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," *Proceedings of the 32nd International Conference on Machine Learning, PMLR,* vol. 37, pp. 448 - 456, 2015.

[102] J. I. C., Y. J. Tsay, S. C. Tsay, Q. Z. Wu and S. S. Yu, "Parallel Distributed Processing with Multiple One-Output Back-propagation Neural Networks," *Proceedings of The IEEE International Symposium on Circuits and Systems (ISCAS),* pp. 1408 - 1411, 11 - 14 June 1991.

[103] Wikipedia, "Multinomial Logistic Regression," [Online]. Available: https://en.wikipedia.org/wiki/Multinomial_logistic_regression. [Accessed 11 07 2023].

[104] Wikipedia, "Indicator Function (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Indicator_function. [Accessed 17 June 2023].

[105] S. J. Haskey and S. Datta, "A Comparative Study of OCON and MLP Architectures for Phoneme Recognition," in *5th International Conference on Spoken Language Processing (ICSLP 98)*, Sydney, 1998.

[106] Wikipedia, "Winner-Take-All (computing)," [Online]. Available: https://en.wikipedia.org/wiki/Winner-take-all_(computing). [Accessed 17 June 2023].

[107] Wikipedia, "Arg max (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Arg_max. [Accessed 17 June 2023].

[108] G. E. Peterson and H. Barney, "Control Methods used in a Study of the Vowels," *Journal of The Acoustical Society of America,* vol. 24, no. 2, pp. 175 - 184, 1952.

[109] R. L. Watrous, "Current Status of Peterson-Barney Vowel Formant Data," *Journal of The Acoustical Society of America,* vol. 89, no. 5, pp. 2459 - 2460, May 1991.

[110] W. Koenig, H. K. Dunn and L. Y. Lacy, "The Sound Spectrograph," *Journal of The Acoustical Society of America,* vol. 18, no. 1, pp. 19 - 44, 1946.

[111] A. K. Syrdal and H. S. Gopal, "A Perceptual Model of Vowel Recognition based on The Auditory Representation of American English Vowels," *Journal of The Acoustical Society of America,* vol. 79, no. 4, pp. 1086 - 1100, April 1986.

[112] R. Lippmann, "An introduction to Computing with Neural Nets," *IEEE ASSP Magazine,* vol. 4, no. 2, pp. 4 - 22, 1987.

[113] W. Y. Huang and L. R.P., "Neural net and traditional classifiers," *Proceedings of the 1987 International Conference on Neural Information Processing Systems (NIPS),* pp. 387 - 396, 1987.

[114] T. Nearey, "Applications of Generalized Linear Modeling to Vowel Data," *Proceedings of the Second International Conference on Spoken Language Processing (ICSLP-92),* pp. 583 - 586, 13 - 16 October 1992.

[115] J. Hillenbrand and R. T. Gayvert, "Vowel Classification based on Fundamental Frequency and Formant Frequencies," *Journal of Speech and Hearing Research,* vol. 36, no. 4, pp. 694 - 700, August 1993.

[116] M. G. Di Benedetto, "Vowel Representation: Some Observations on Temporal and Spectral Properties of The First Formant Frequency," *Journal of The Acoustical Society of America,* vol. 86, pp. 55 - 66, 1989.

[117] M. G. Di Benedetto, "Frequency and Time Variations of the First Formant: Properties Relevant to The Perception of Vowel Height," *Journal of The Acoustical Society of America,* vol. 86, pp. 67 - 77, 1989.

[118] J. Hillenbrand, L. A. Getty, M. J. Clark and K. Wheeler, "Acoustic characteristics of American English vowels," *Journal of The Acoustical Society of America,* vol. 97, no. 5, pp. 3099 - 3111, 1995.

[119] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett and N. L. Dahlgren, *DARPA TIMIT Acoustic Phonetic Continuous Speech Corpus,* D. L. D. Consortium, Ed., Philadelphia: DARPA, 1993.

[120] L. Deng, X. Cui, R. Pruvenok, J. Huang, S. Momen, Y. Chen and A. Alwan, "A Database of Vocal Tract Resonance Trajectories for Research in Speech Processing," *Proceedings of 2006 IEEE International Conference on Acoustics Speech and Signal Processing,* 14 - 19 May 2006.

[121] Wikipedia, "Framework Programmes for Research and Technological Development (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Framework_Programmes_for_Research_and_Technological_Development. [Accessed 24 June 2023].

[122] v. P. Hove, "ESPRIT, The European Strategic Programme for Research and Development in Information Technology," 1987 - 1992.

[123] Wikipedia, "SAMPA - Speech Assessments Methods Phonetic Alphabet (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/SAMPA. [Accessed 24 June 2023].

[124] B. Atal, "Effectiveness of Linear Prediction Characteristics of The Speech Wave for Automatic Speaker Identification and Verification," *Journal of The Acoustical Society of America,* vol. 55, no. 6, pp. 1304 - 1312, 1974.

[125] R. C. Snell and F. Milinazzo, "Formant Location from LPC Analysis Data," *IEEE Transactions on Speech and Audio Processing,* vol. 1, no. 2, pp. 129 - 134, Aprile 1993.

[126] Y. Dissen, J. Goldberger and J. Keshet, "Formant Estimation and Tracking: A Deep Learning Approach," *Journal of The Acoustical Society of America,* vol. 145, no. 2, pp. 642 - 653, 2019.

[127] I. I. Ibrahim and M. I. El-Adawy, "Determination of the LPC Coefficients using Neural Networks," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth (AST), 1995.

[128] R. J. Lloyd, Some Researches into the Nature of Vowel-Sound, University of California Libraries, 2011, p. 40.

[129] J. D. Miller, "Auditory-perceptual interpretation of the vowel," *Journal of The Acoustical Society of America,* vol. 85, no. 5, pp. 2114 - 2134, 1989.

[130] A. K. Syrdal, "Aspects of a Model of the Auditory Representation of American English Vowels," *Speech Communication,* vol. 4, no. 1 - 3, pp. 121 - 135, 1985.

[131] E. Zwicker and E. Terhardt, "Analytical Expressions for Critical-band Rate and Critical Bandwidth as a Function of Frequency," *Journal of The Acoustical Society of America,* vol. 68, no. 5, pp. 1523 - 1525, 1980.

[132] H. Traunmüller, "Perceptual dimension of openness in vowels," *Journal of The Acoustical Society of America,* vol. 69, no. 5, pp. 1465 - 1475, 1981.

[133] W. Koenig, "A New Frequency Scale for Acoustic Measurements," *Bell Telephone Laboratory Record,* no. 27, pp. 299 - 301, 1949.

[134] Wikipedia, "Ronald Fisher (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Ronald_Fisher. [Accessed 24 June 2023].

[135] Wikipedia, "Prior Probability (Bayesian Statistics) (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Prior_probability. [Accessed 24 June 2023].

[136] Wikipedia, "Generalized Linear Model (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Generalized_linear_model. [Accessed 24 June 2023].

[137] R. Levy, "Generalized Linear Models (Chapter - 6)," in *Probabilistic Models in The Study of Language*, draft, 2012.

[138] T. M. Nearey, "Applications of Generalized Linear Modeling to Vowel Data," *Proceedings of 2nd International Conference on Spoken Language Processing (ICSLP 1992),* pp. 583-586, 1992.

[139] T. Kohonen, K. Mäkisara and T. Saramäki, "Phonotopic Maps - Insightful Representation of Phonological Features for Speech Recognition," in *Proceedings of the 6th International Conference on Pattern Recognition (ICPR)*, Montreal, 1984.

[140] Wikipedia, "Kohonen Feature Map (SOM) (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Self-organizing_map. [Accessed 25 June 2023].

[141] G. Hult, "Some Vowel Recognition Experiments using Multilayer Perceptrons," 1989.

[142] P. D. Templeton and B. J. Guillemin, "Speaker Identification based on Vowel Sounds using Neural Networks," *Proceedings of 3rd International Conference on Speech, Science and Technology,* vol. 39, pp. 280 - 285, 1990.

[143] W. G. W. Jassem, "Off-line Classification of Polish Vowel Spectra using Artificial Neural Networks," *Journal of the International Phonetic Association,* vol. 34, no. 1, pp. 37 - 52, 2004.

[144] R. P. V. Shinde, "Vowel Classification based on LPC and ANN," *International Journal of Computer Applications,* vol. 50, no. 6, pp. 27 - 31, 2012.

[145] D. Kristomo, R. Hidayat and I. Soesanti, "Syllables Sound Signal Classification using Multi-layer Perceptron in Varying Number of Hidden-Layer and Hidden-Neuron," *Proceedings of the 2nd International Conference on Engineering and Technology for Sustainable Development (ICET4SD 2017),* no. 154, 2018.

[146] L. Lugosch, "Hidden Markov Models (PyTorch Tutorial)," [Online]. Available: https://colab.research.google.com/drive/1IUe9lfoIiQsL49atSOgxnCmMR_zJazKI. [Accessed 11 07 2023].

[147] C. Spark, "Neural Networks in Python: From Sklearn to PyTorch and Probabilistic Neural Networks," [Online]. Available: https://www.cambridgespark.com/info/neural-networks-in-python. [Accessed 11 07 2023].

[148] Wikipedia, "Feature Scaling (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Feature_scaling. [Accessed 1 July 2023].

[149] Wikipedia, "Poisson Distribution (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Poisson_distribution. [Accessed 1 July 2023].

[150] Wikipedia, "Log-Normal Distribution (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Log-normal_distribution. [Accessed 1 July 2023].

[151] Wikipedia, "Overfitting (Wikipedia)," [Online]. Available: https://en.wikipedia.org/wiki/Overfitting. [Accessed 2 July 2023].

[152] V. L. -. P. B. (. Researcher), "The Essential Guide to Neural Network Architectures," [Online]. Available: https://www.v7labs.com/blog/neural-network-architectures-guide. [Accessed 11 07 2023].

[153] D. Röckmann and C. Claudio Moraga, "Using Quadratic Perceptrons to Reduce Interconnection Density in Multilayer Neural Networks," *International Workshop on Artificial Neural Networks (IWANN - 1991),* vol. 540, 1991.

[154] PyTorch, "PyTorch NumPy Extension," [Online]. Available: https://pytorch.org/tutorials/advanced/numpy_extensions_tutorial.html. [Accessed 11 07 2023].

[155] K. Deng, A. Bansal and D. Ramanan, "Unsupervised Audiovisual Synthesis via Exemplar Autoencoders," in *9th International Conference on Learning Representations (ICLR 2021) Poster*, 2021.

[156] H. C. Byung, J. Gonzalvo, Chun-an, C. Agiomyrgiannakis, V. Ping, L. Wan, R. Andrew, J. Clark and J. Vit, "Text-To-Speech Synthesis Using an Autoencoder". USA Patent US 10249289 B2, 2 April 2019.

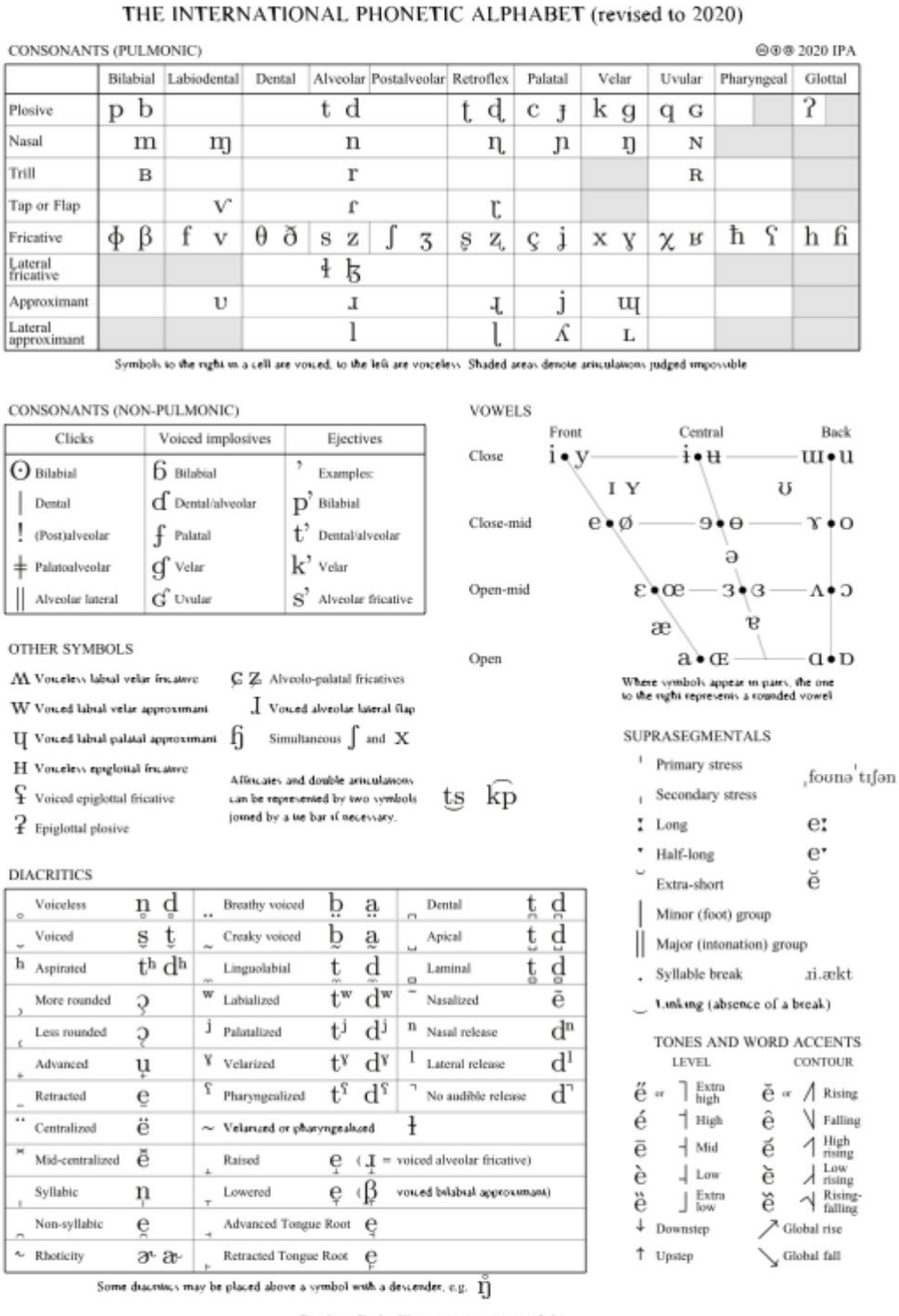
[157] A. M. Sarroff and M. Casey, "Musical Audio Synthesis Using Autoencoders Neural Nets," *Goldsmiths Research Online,* 1 October 2014.

[158] Parselmouth, "Parselmouth Documentation (Python)," [Online]. Available: https://parselmouth.readthedocs.io/en/stable/index.html. [Accessed 11 07 2023].

[159] audEERING, "openSMILE Python," [Online]. Available: https://audeering.github.io/opensmile-python/index.html. [Accessed 11 07 2023].

[160] blog.syntheticspeech.de, "How to Extract Formant Tracks with Praat and Python," [Online]. Available: http://blog.syntheticspeech.de/2021/03/10/how-to-extract-formant-tracks-with-praat-and-python/. [Accessed 11 07 2023].

APPENDIX-B (Graphs & Tables)



THE INTERNATIONAL PHONETIC ALPHABET (revised to 2020)

CONSONANTS (PULMONIC) ©℗® 2020 IPA

	Bilabial	Labiodental	Dental	Alveolar	Postalveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Glottal
Plosive	p b			t d		ʈ ɖ	c ɟ	k g	q ɢ		ʔ
Nasal	m	ɱ		n		ɳ	ɲ	ŋ	ɴ		
Trill	ʙ			r					ʀ		
Tap or Flap		ⱱ		ɾ		ɽ					
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	h ɦ
Lateral fricative				ɬ ɮ							
Approximant		ʋ		ɹ		ɻ	j	ɰ			
Lateral approximant				l		ɭ	ʎ	ʟ			

Symbols to the right in a cell are voiced, to the left are voiceless. Shaded areas denote articulations judged impossible

CONSONANTS (NON-PULMONIC)

Clicks	Voiced implosives	Ejectives
ʘ Bilabial	ɓ Bilabial	' Examples:
ǀ Dental	ɗ Dental/alveolar	pʼ Bilabial
ǃ (Post)alveolar	ʄ Palatal	tʼ Dental/alveolar
ǂ Palatoalveolar	ɠ Velar	kʼ Velar
ǁ Alveolar lateral	ʛ Uvular	sʼ Alveolar fricative

OTHER SYMBOLS

ʍ Voiceless labial-velar fricative
w Voiced labial-velar approximant
ɥ Voiced labial-palatal approximant
ʜ Voiceless epiglottal fricative
ʢ Voiced epiglottal fricative
ʡ Epiglottal plosive

ʗ ʖ Alveolo-palatal fricatives
ɺ Voiced alveolar lateral flap
ʃʃ Simultaneous ʃ and x

Affricates and double articulations can be represented by two symbols joined by a tie bar if necessary.

t͡s k͡p

VOWELS

	Front	Central	Back
Close	i • y	ɨ • ʉ	ɯ • u
	ɪ ʏ		ʊ
Close-mid	e • ø	ɘ • ɵ	ɤ • o
		ə	
Open-mid	ɛ • œ	ɜ • ɞ	ʌ • ɔ
	æ	ɐ	
Open	a • ɶ		ɑ • ɒ

Where symbols appear in pairs, the one to the right represents a rounded vowel

SUPRASEGMENTALS

ˈ Primary stress	ˌfounəˈtɪʃən
ˌ Secondary stress	
ː Long	eː
ˑ Half-long	eˑ
˘ Extra-short	ĕ
| Minor (foot) group	
‖ Major (intonation) group	
. Syllable break	ɹi.ækt
‿ Linking (absence of a break)	

DIACRITICS

◌̥ Voiceless	n̥ d̥	◌̤ Breathy voiced	b̤ a̤	◌̪ Dental	t̪ d̪
◌̬ Voiced	s̬ t̬	◌̰ Creaky voiced	b̰ a̰	◌̺ Apical	t̺ d̺
◌ʰ Aspirated	tʰ dʰ	◌̼ Linguolabial	t̼ d̼	◌̻ Laminal	t̻ d̻
◌̹ More rounded	ɔ̹	◌ʷ Labialized	tʷ dʷ	◌̃ Nasalized	ẽ
◌̜ Less rounded	ɔ̜	◌ʲ Palatalized	tʲ dʲ	◌ⁿ Nasal release	dⁿ
◌̟ Advanced	u̟	◌ˠ Velarized	tˠ dˠ	◌ˡ Lateral release	dˡ
◌̠ Retracted	e̠	◌ˤ Pharyngealized	tˤ dˤ	◌̚ No audible release	d̚
◌̈ Centralized	ë	~ Velarized or pharyngealized	ɫ		
◌̽ Mid-centralized	e̽	◌̝ Raised	e̝ (ɪ̝ = voiced alveolar fricative)		
◌̩ Syllabic	n̩	◌̞ Lowered	e̞ (β̞ voiced bilabial approximant)		
◌̯ Non-syllabic	e̯	◌̘ Advanced Tongue Root	e̘		
◌˞ Rhoticity	ɚ a˞	◌̙ Retracted Tongue Root	e̙		

Some diacritics may be placed above a symbol with a descender, e.g. ŋ̊

TONES AND WORD ACCENTS

LEVEL		CONTOUR	
e̋ or ˥	Extra high	ě or ˩˥	Rising
é ˦	High	ê ˥˩	Falling
ē ˧	Mid	e᷄ ˦˥	High rising
è ˨	Low	e᷅ ˩˧	Low rising
ȅ ˩	Extra low	e᷈ ˩˥˧	Rising-falling
↓ Downstep		↗ Global rise	
↑ Upstep		↘ Global fall	

Typefaces: Doulos SIL (metatext); Gentium (symbols)

THE IPA/ARPAbet OFFICIAL TABLE

		IPA Symbol	ARPAbet (SV)	ARPAbet (UV)	Examples
Vowels	Front	i	i	IY	beat
		ɪ	I	IH	bit
		e	e	EY	bait
		ɛ	E	EH	bet
		æ	@	AE	bat
	Back	ɑ	a	AA	Bob
		ɔ	c	AO	bought
		o	o	OW	boat
		ʊ	U	UH	book
		u	u	UW	boot
	Mid	ɝ	R	ER	bird
		ə	x	AX	ago
		ʌ	A	AH	but
Diphthongs		ɑɪ	Y	AY	buy
		ɑʊ	W	AW	down
		ɔɪ	O	OY	boy
		ɨ	X	IX	roses
Stop Consonants	Voiced	b	b	B	bat
		d	d	D	deep
		g	g	G	go
	Unvoiced	p	p	P	pea
		t	t	T	tea
		k	k	K	kick
Fricatives	Voiced	v	v	V	vice
		ð	D	DH	then
		z	z	Z	zebra
		ʒ	Z	ZH	measure
	Unvoiced	f	f	F	five
		θ	T	TH	thing
		s	s	S	so
		ʃ	S	SH	show
Semivowels	Liquids	l	l	L	love
		l	L	EL	cattle
		r	r	R	race
	Glides	w	w	W	want
		ʍ	H	WH	when
		j	y	Y	yard
Nasal	Non vocalic	m	m	M	mom
		n	n	N	noon
		ŋ	G	NX	sing
	Vocalic	m	M	EM	some
		n	N	EN	son
Affricates		tʃ	C	CH	church
		dʒ	J	JH	just
Others	Whisper	h	h	HH	help
	Vocalic	f	F	DX	batter
	Glottal stop	ʔ	Q	Q	

TABLE II. Average absolute difference between formant frequencies sampled at "steady-state" times determined by two judges. Figures in parentheses are differences as a percent of average formant frequency.

	Men	Women	Children	Overall
*F*1	7.5 (1.3%)	9.2 (1.5%)	10.7 (1.8%)	9.2 (1.5%)
*F*2	14.2 (0.8%)	20.0 (1.1%)	18.5 (1.1%)	17.6 (1.0%)
*F*3	18.6 (0.7)	21.2 (0.7%)	27.6 (1.0%)	22.5 (0.8%)
*F*4	20.6 (0.5%)	31.5 (0.8%)	36.4 (0.9%)	29.5 (0.7%)

TABLE III. Measurement–remeasurement reliability for formant frequencies obtained from a randomly selected 10% of the signals. Values are given as average absolute differences and average signed differences.

	Men		Women		Children		Overall	
	Abs	Signed	Abs	Signed	Abs	Signed	Abs	Signed
*F*1	8.1	−1.8	12.2	−3.3	14.2	−2.5	11.7	−2.6
*F*2	20.7	2.8	26.4	2.8	27.4	2.4	25.2	3.4
*F*3	23.1	1.2	28.2	1.2	34.0	8.9	28.7	2.5
*F*4	56.2	−3.1	50.7	−3.1	69.1	15.4	59.0	4.0

TABLE IV. Comparison of formant frequencies derived from LPC analysis and cepstral smoothing. Results are given as average absolute differences and as average signed differences (LPC-cepstrum). Figures in parentheses are differences as a percent of average formant frequency.

	Absolute	Signed
*F*1	53.5 (8.9%)	41.5 (6.9%)
*F*2	60.4 (3.5%)	12.7 (0.7%)
*F*3	74.0 (2.6%)	10.9 (0.4%)
*F*4	91.1 (2.3%)	7.4 (0.2%)

TABLE V. Average durations, fundamental frequencies, and formant frequencies of vowels produced by 45 men, 48 women, and 46 children. Averages are based on a subset of the tokens that were well identified by listeners (see text for details). The duration measurements are in ms; all others are in Hz.

		/i/	/ɪ/	/e/	/ɛ/	/æ/	/ɑ/	/ɔ/	/o/	/ʊ/	/u/	/ʌ/	/ɝ/
Dur	M	243	192	267	189	278	267	283	265	192	237	188	263
	W	306	237	320	254	332	323	353	326	249	303	226	321
	C	297	248	314	235	322	311	319	310	247	278	234	307
*F*0	M	138	135	129	127	123	123	121	129	133	143	133	130
	W	227	224	219	214	215	215	210	217	230	235	218	217
	C	246	241	237	230	228	229	225	236	243	249	236	237
*F*1	M	342	427	476	580	588	768	652	497	469	378	623	474
	W	437	483	536	731	669	936	781	555	519	459	753	523
	C	452	511	564	749	717	1002	803	597	568	494	749	586
*F*2	M	2322	2034	2089	1799	1952	1333	997	910	1122	997	1200	1379
	W	2761	2365	2530	2058	2349	1551	1136	1035	1225	1105	1426	1588
	C	3081	2552	2656	2267	2501	1688	1210	1137	1490	1345	1546	1719
*F*3	M	3000	2684	2691	2605	2601	2522	2538	2459	2434	2343	2550	1710
	W	3372	3053	3047	2979	2972	2815	2824	2828	2827	2735	2933	1929
	C	3702	3403	3323	3310	3289	2950	2982	2987	3072	2988	3145	2143
*F*4	M	3657	3618	3649	3677	3624	3687	3486	3384	3400	3357	3557	3334
	W	4352	4334	4319	4294	4290	4299	3923	3927	4052	4115	4092	3914
	C	4572	4575	4422	4671	4409	4307	3919	4167	4328	4276	4320	3788

TABLE IX. Quadratic discrimination results for the present data showing the effect of including duration and spectral change information on classification accuracy. The table shows overall classification accuracy using the "jackknife" method in which measurements for individual tokens are removed from the training statistics prior to classification. The one-sample results are based on a single sample of the formant pattern at "steady state;" the two-sample results are based on samples taken at 20% and 80% of vowel duration; the three-sample results are based on samples taken at 20%, 50%, and 80% of vowel duration. ("NoDur"=vowel duration not included; "Dur"=vowel duration included.) Entries under the heading of "All tokens" used the full database; entries under the heading "well identified tokens only" are based on a data set that did not include tokens with error rates of 15% or greater (11.5% of the tokens).

			All tokens			
	One sample		Two samples		Three samples	
Parameter set	NoDur	Dur	NoDur	Dur	NoDur	Dur
*F*1,*F*2	68.2	76.1	87.9	90.3	87.7	90.4
*F*1,*F*2,*F*3	81.0	84.6	91.6	92.7	91.8	93.1
*F*0,*F*1,*F*2	78.2	82.0	91.6	92.5	91.0	92.6
*F*0,*F*1,*F*2,*F*3	84.7	87.8	93.6	94.1	92.8	94.8
			Well identified tokens only			
	One sample		Two samples		Three samples	
Parameter set	NoDur	Dur	NoDur	Dur	NoDur	Dur
*F*1,*F*2	71.4	80.0	90.8	93.6	90.7	93.3
*F*1,*F*2,*F*3	85.3	89.1	95.4	96.2	95.3	95.8
*F*0,*F*1,*F*2	82.3	86.3	95.5	96.3	94.8	96.0
*F*0,*F*1,*F*2,*F*3	88.7	91.6	97.3	97.8	96.6	97.3

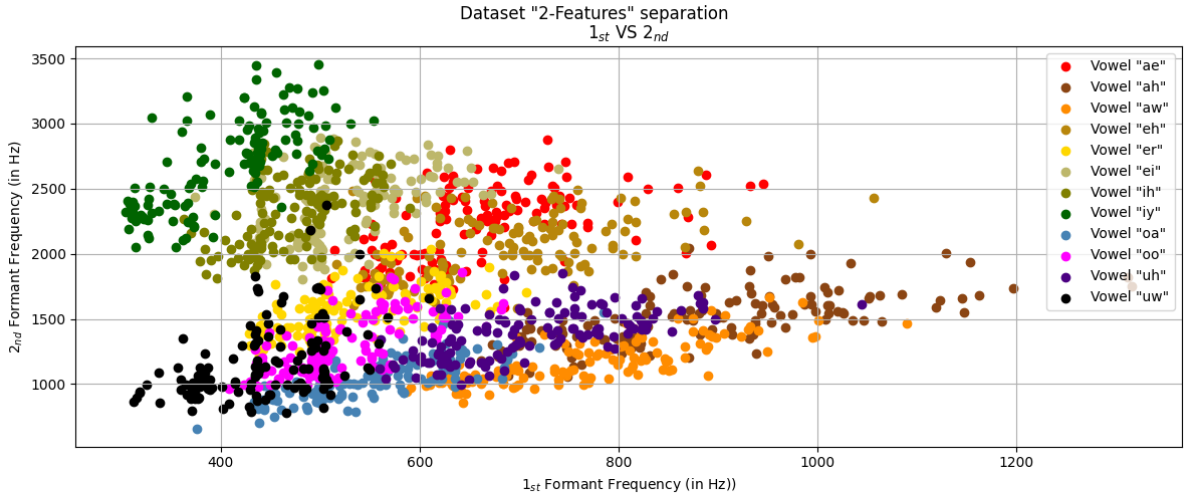
TABLE VI. Overall percent correct identification by vowel category. for the present study (HGCW) and for Peterson and Barney (PB).

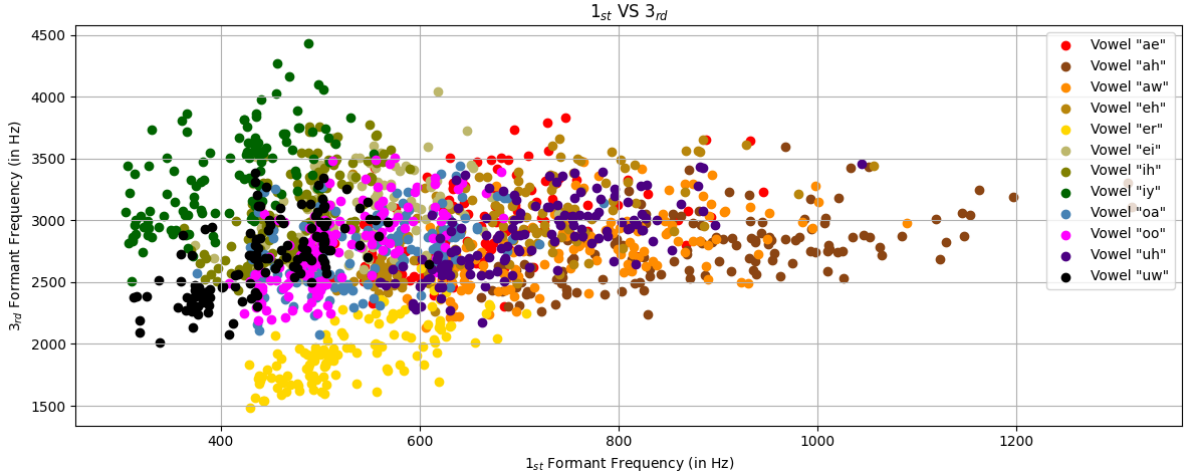
	HGCW	PB
/i/	99.6	99.9
/ɪ/	98.8	92.9
/e/	98.3	a
/ɛ/	95.1	87.7
/æ/	94.1	96.5
/ɑ/	92.3	87.0
/ɔ/	82.0	92.8
/o/	99.2	a
/ʊ/	97.5	96.5
/u/	97.2	99.2
/ʌ/	90.8	92.2
/ɝ/	99.5	99.7
Total:	95.4	94.4
Men:	94.6	b
Women:	95.6	b
Children	93.7	b

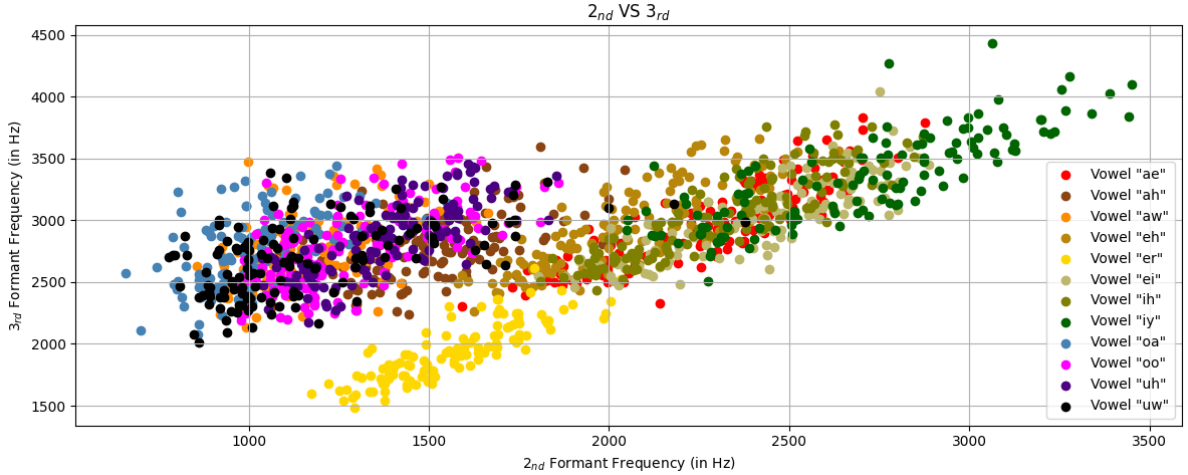
[a]These vowels were not recorded by Peterson and Barney.
[b]Peterson and Barney did not report results separately for men, women, and child talkers.

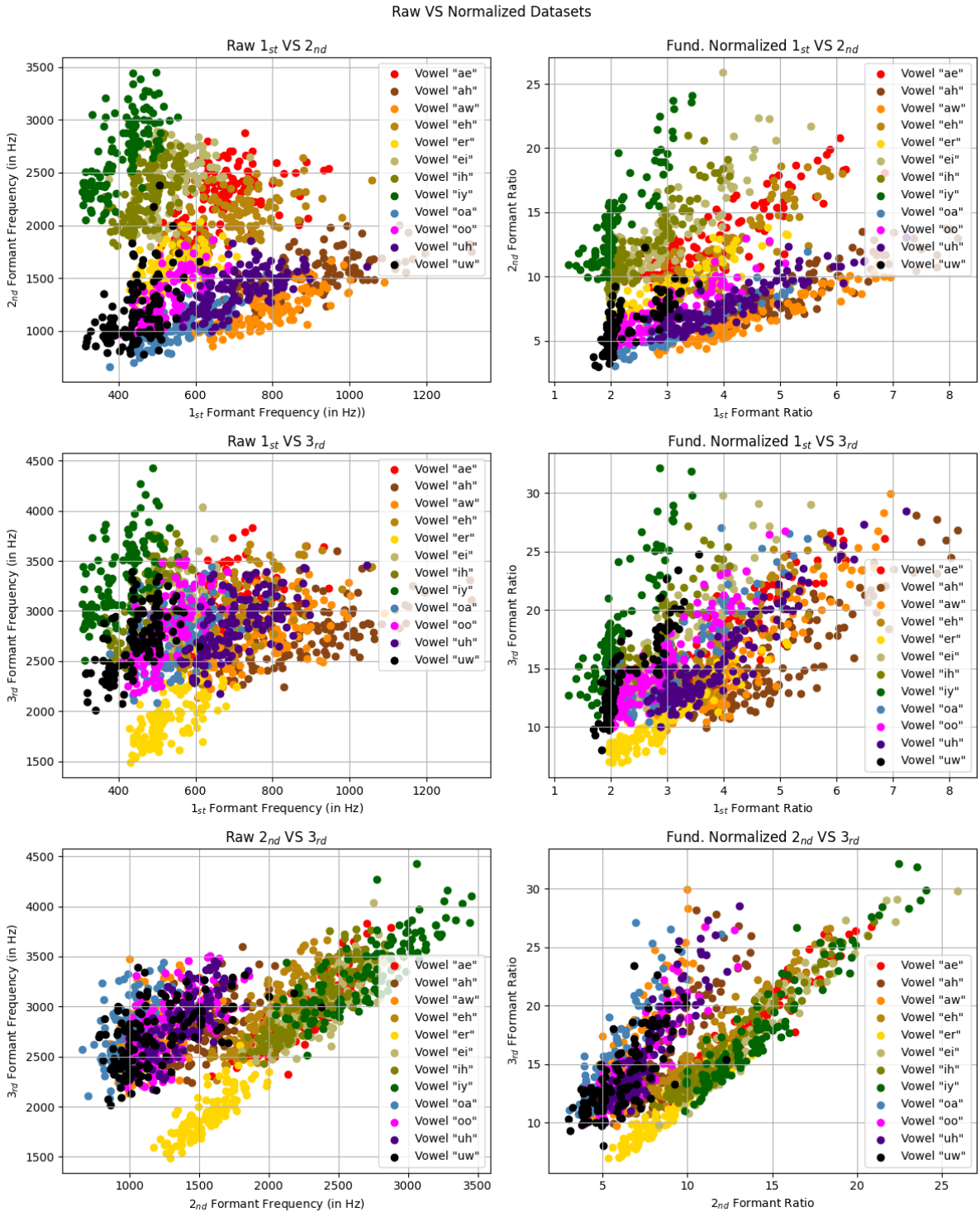
TABLE VIII. Quadratic discrimination results for the present data (HGCW) and for the PB data set based on a single sample of the formant pattern. The table shows overall classification accuracy using the "jackknife" method in which measurements for individual tokens are removed from the training statistics prior to classification.

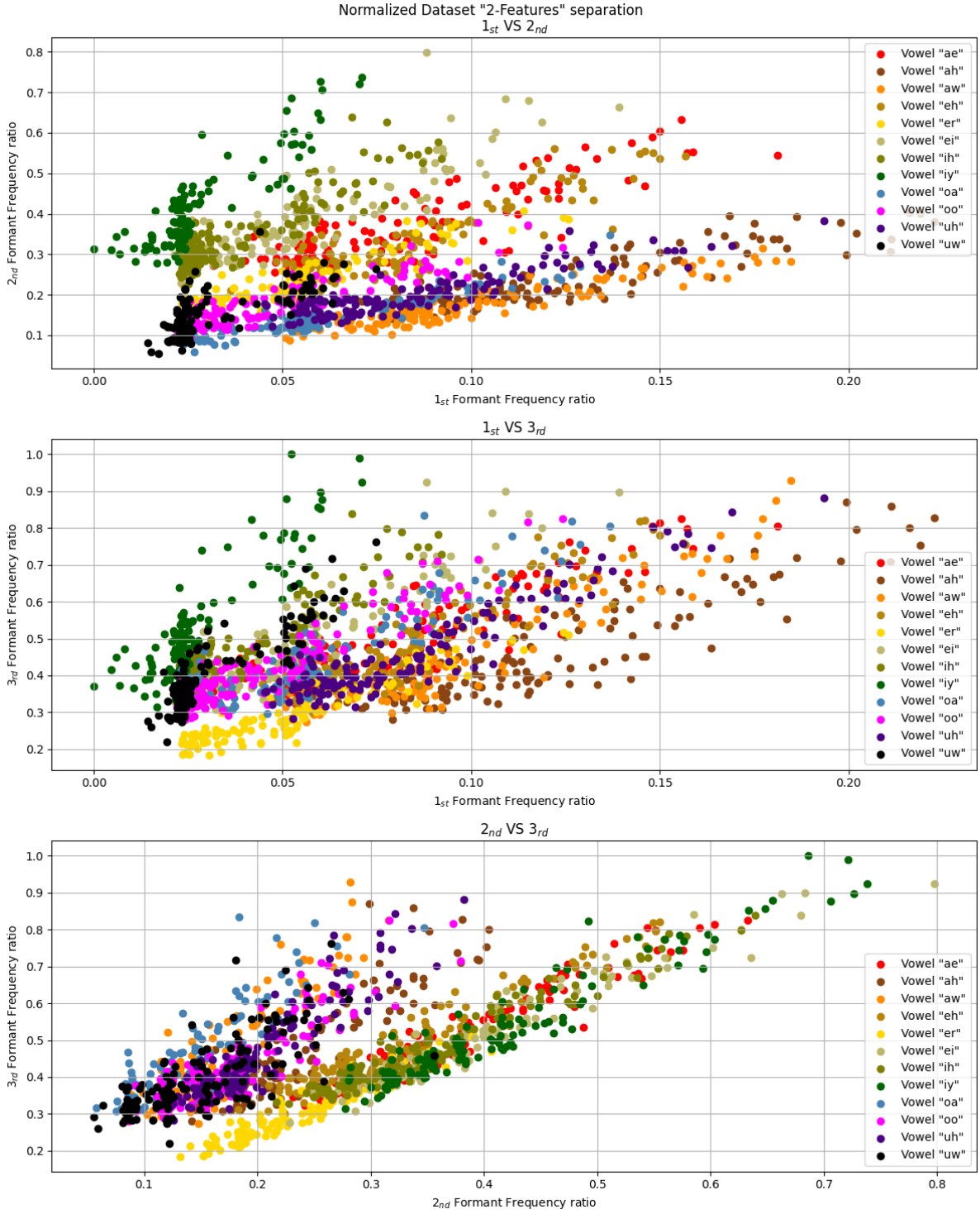
Parameter set	HGCW	PB
*F*1,*F*2	68.2	74.9
*F*1,*F*2,*F*3	81.0	83.6
*F*0,*F*1,*F*2	78.2	85.9
*F*0,*F*1,*F*2,*F*3	84.7	86.6

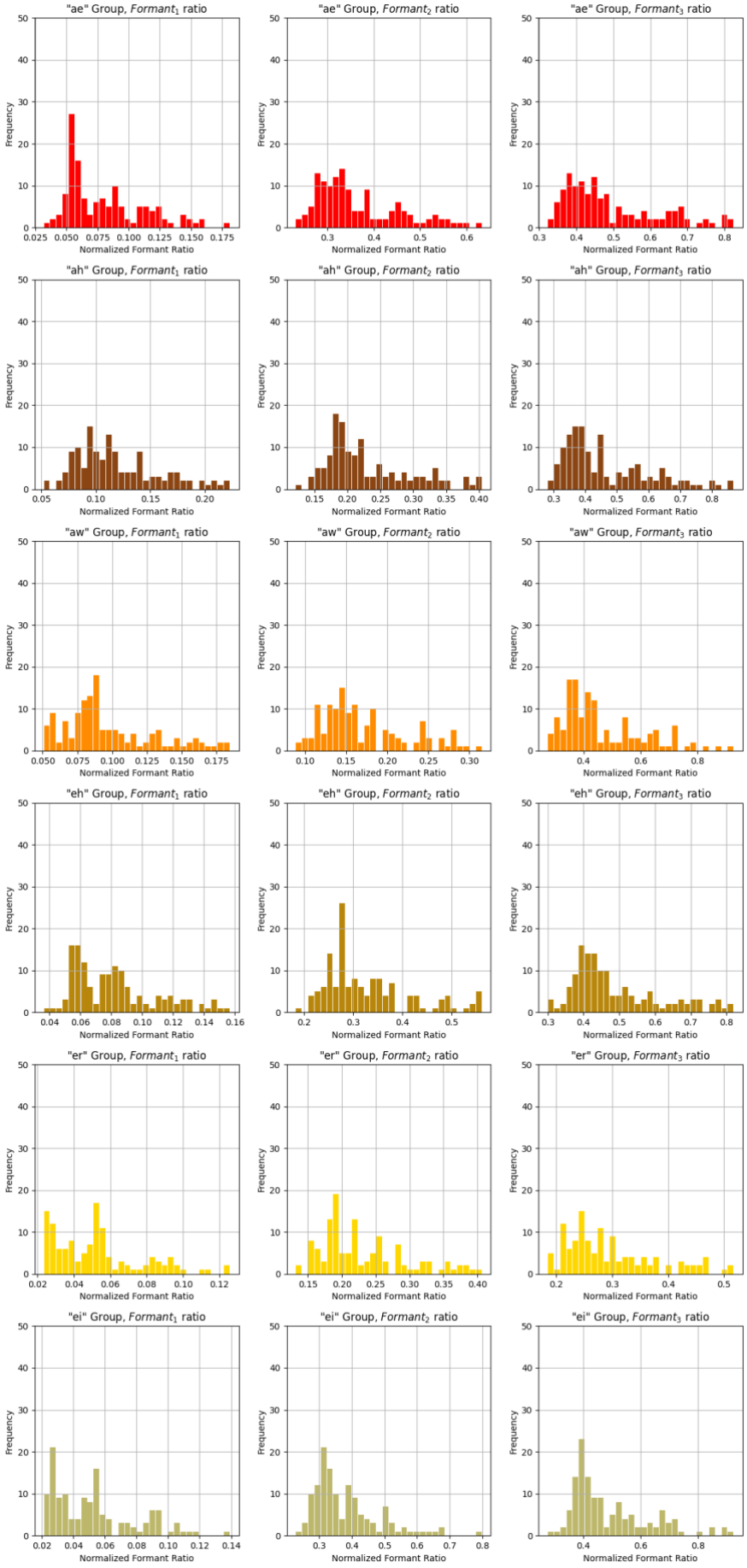

Dataset "2-Features" separation
1st VS 2nd
2nd Formant Frequency (in Hz)
1st Formant Frequency (in Hz))
Vowel "ae"
Vowel "ah"
Vowel "aw"
Vowel "eh"
Vowel "er"
Vowel "ei"
Vowel "ih"
Vowel "iy"
Vowel "oa"
Vowel "oo"
Vowel "uh"
Vowel "uw"


1st VS 3rd
3rd Formant Frequency (in Hz)
1st Formant Frequency (in Hz)
Vowel "ae"
Vowel "ah"
Vowel "aw"
Vowel "eh"
Vowel "er"
Vowel "ei"
Vowel "ih"
Vowel "iy"
Vowel "oa"
Vowel "oo"
Vowel "uh"
Vowel "uw"

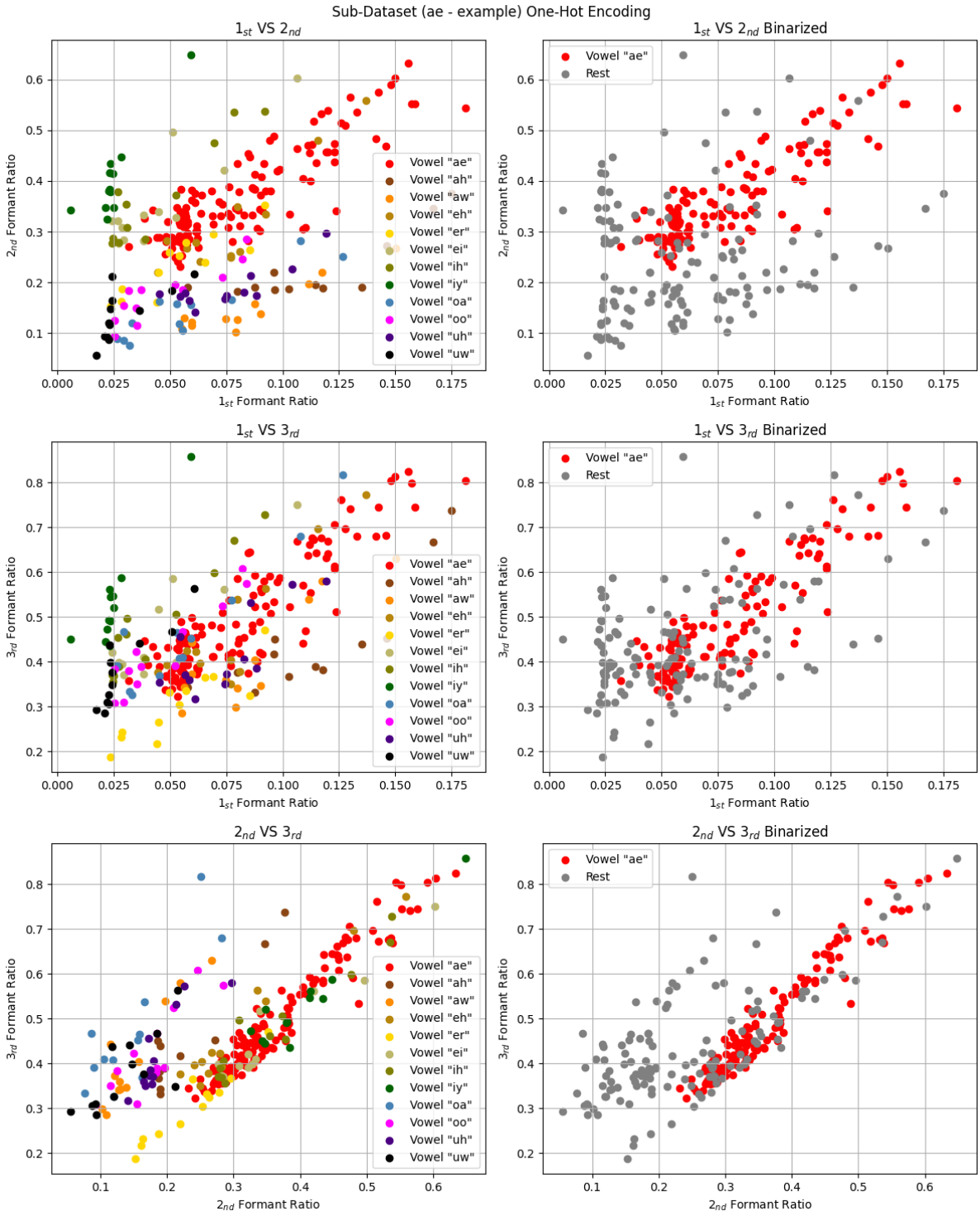

2nd VS 3rd
3rd Formant Frequency (in Hz)
2nd Formant Frequency (in Hz)
Vowel "ae"
Vowel "ah"
Vowel "aw"
Vowel "eh"
Vowel "er"
Vowel "ei"
Vowel "ih"
Vowel "iy"
Vowel "oa"
Vowel "oo"
Vowel "uh"
Vowel "uw"

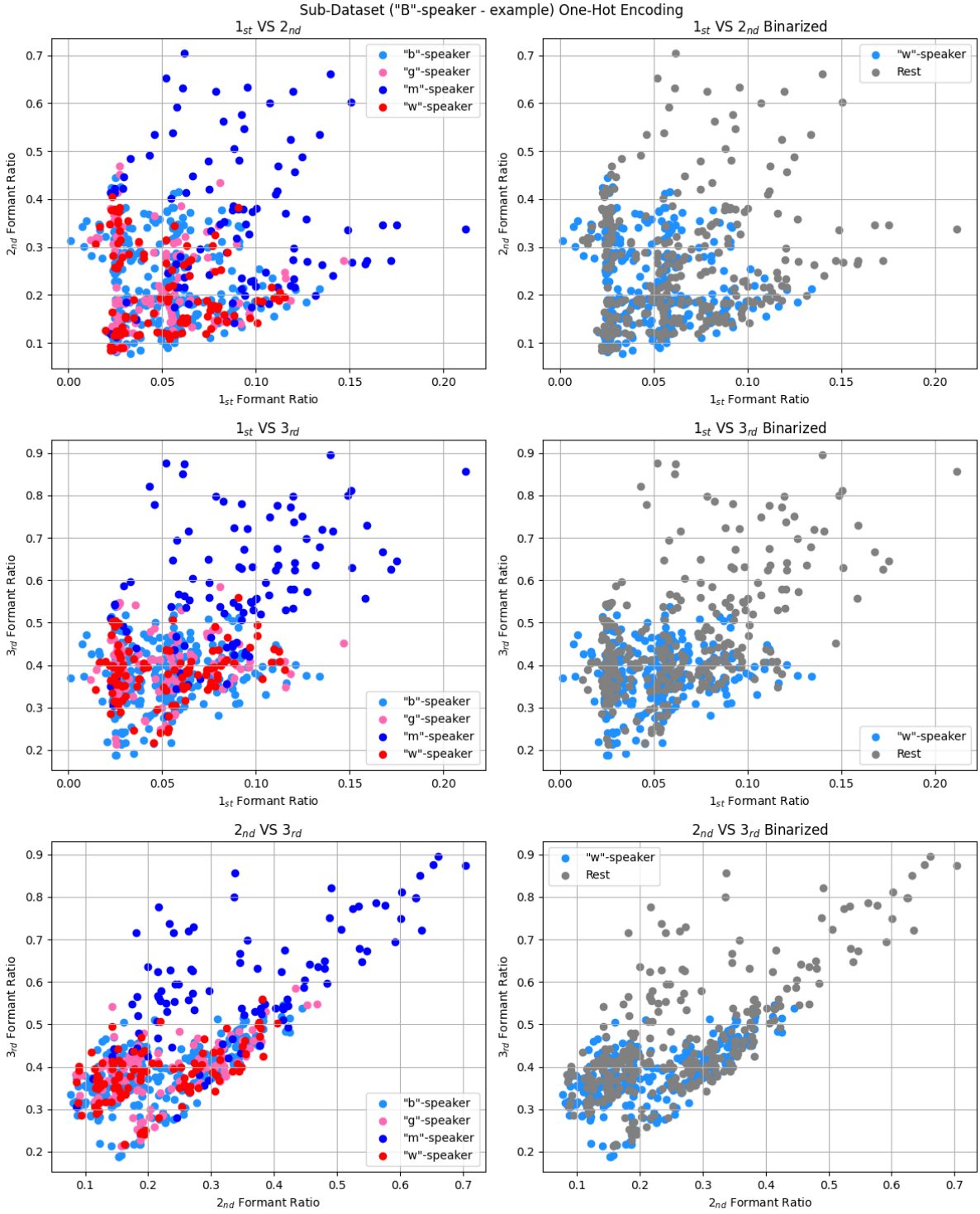

Raw VS Normalized Datasets
Raw 1st VS 2nd
2nd Formant Frequency (in Hz)
1st Formant Frequency (in Hz))
Fund. Normalized 1st VS 2nd
2nd Formant Ratio
1st Formant Ratio
Raw 1st VS 3rd
3rd Formant Frequency (in Hz)
1st Formant Frequency (in Hz)
Fund. Normalized 1st VS 3rd
3rd Formant Ratio
1st Formant Ratio
Raw 2nd VS 3rd
3rd Formant Frequency (in Hz)
2nd Formant Frequency (in Hz)
Fund. Normalized 2nd VS 3rd
3rd FFormant Ratio
2nd Formant Ratio
Vowel "ae"
Vowel "ah"
Vowel "aw"
Vowel "eh"
Vowel "er"
Vowel "ei"
Vowel "ih"
Vowel "iy"
Vowel "oa"
Vowel "oo"
Vowel "uh"
Vowel "uw"

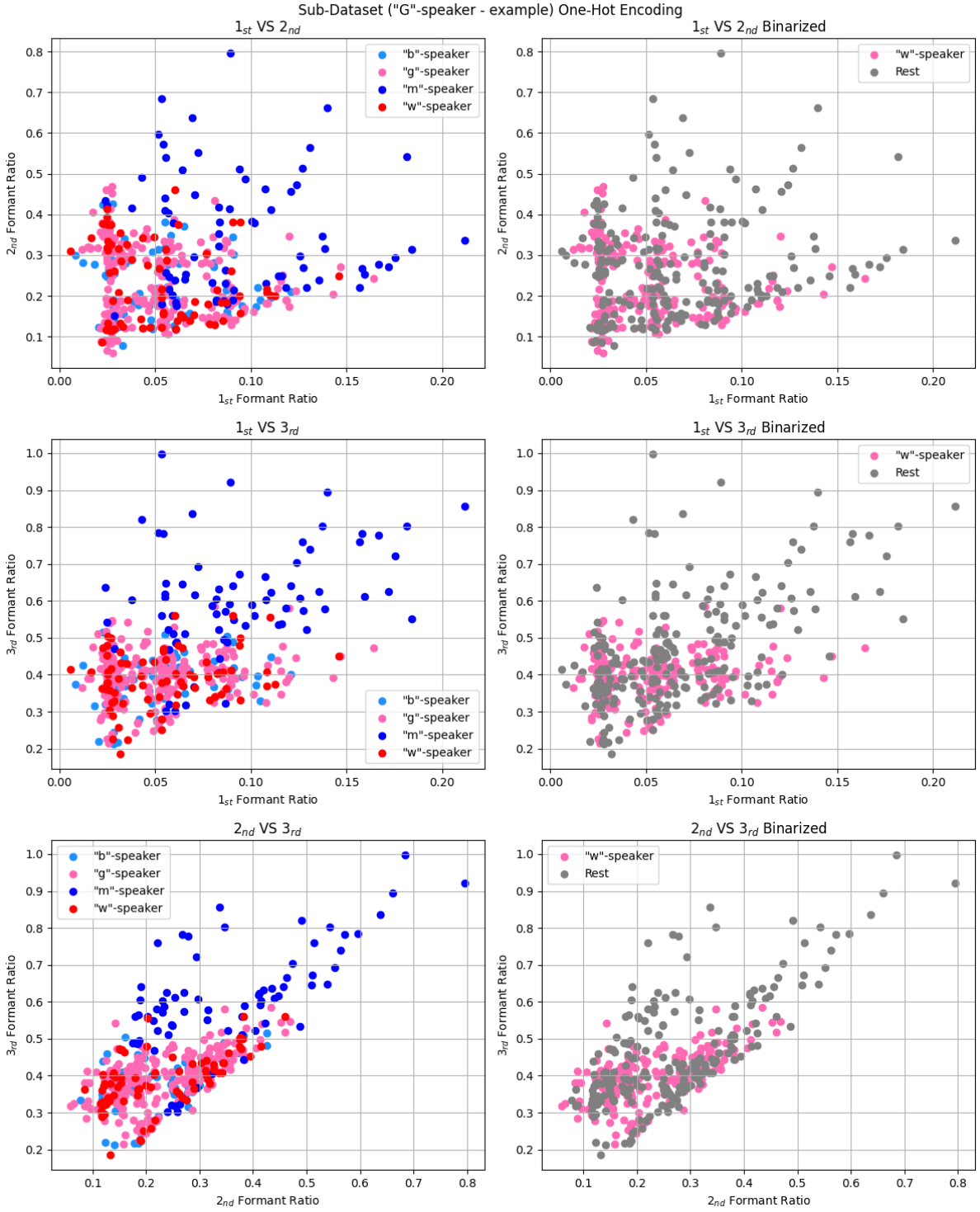

Normalized Dataset "2-Features" separation
1st VS 2nd
2nd Formant Frequency ratio
1st Formant Frequency ratio
1st VS 3rd
3rd Formant Frequency ratio
1st Formant Frequency ratio
2nd VS 3rd
3rd Formant Frequency ratio
2nd Formant Frequency ratio
Vowel "ae"
Vowel "ah"
Vowel "aw"
Vowel "eh"
Vowel "er"
Vowel "ei"
Vowel "ih"
Vowel "iy"
Vowel "oa"
Vowel "oo"
Vowel "uh"
Vowel "uw"

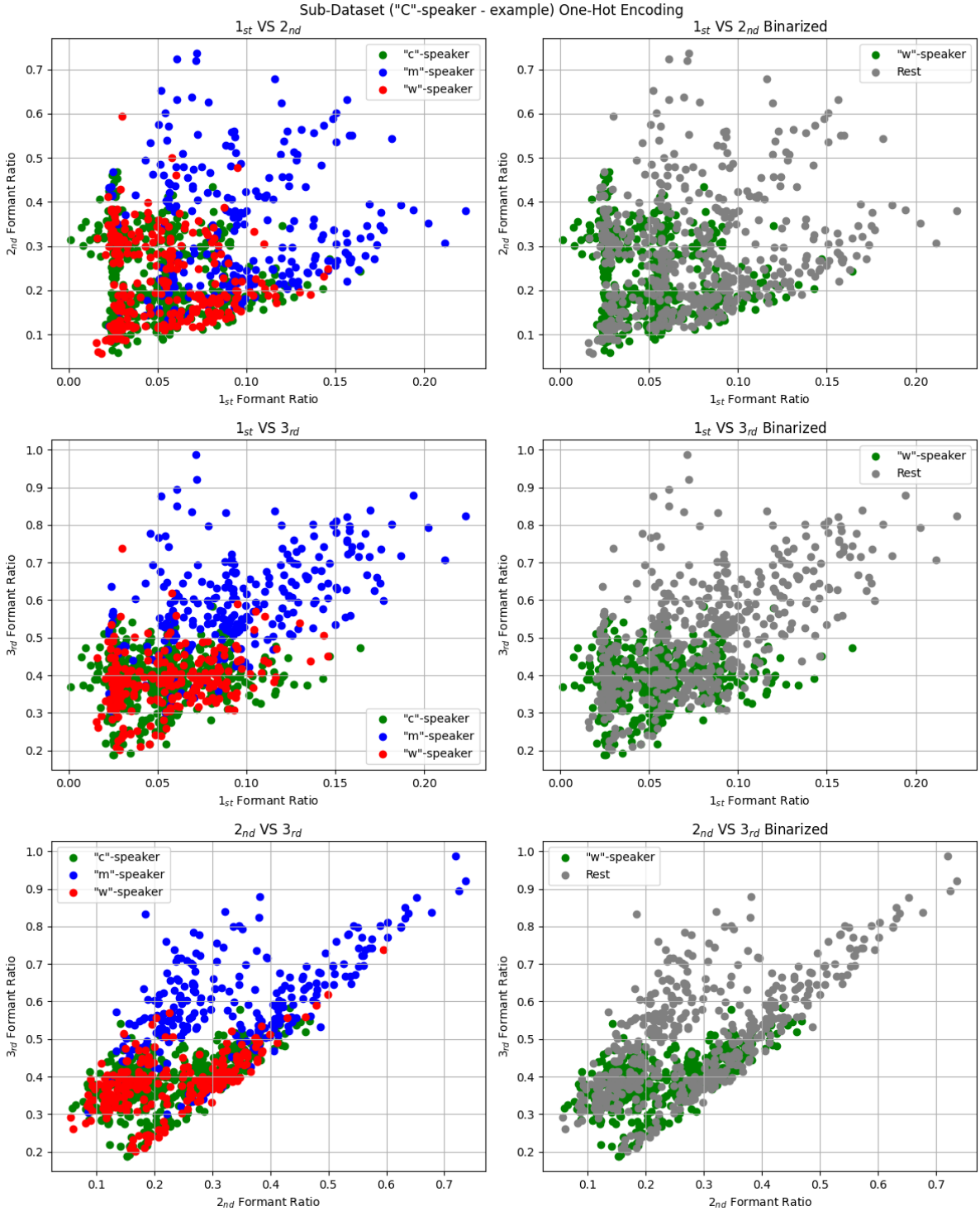

"ae" Group, Formant1 ratio
"ae" Group, Formant2 ratio
"ae" Group, Formant3 ratio
"ah" Group, Formant1 ratio
"ah" Group, Formant2 ratio
"ah" Group, Formant3 ratio
"aw" Group, Formant1 ratio
"aw" Group, Formant2 ratio
"aw" Group, Formant3 ratio
"eh" Group, Formant1 ratio
"eh" Group, Formant2 ratio
"eh" Group, Formant3 ratio
"er" Group, Formant1 ratio
"er" Group, Formant2 ratio
"er" Group, Formant3 ratio
"ei" Group, Formant1 ratio
"ei" Group, Formant2 ratio
"ei" Group, Formant3 ratio
Frequency
Normalized Formant Ratio

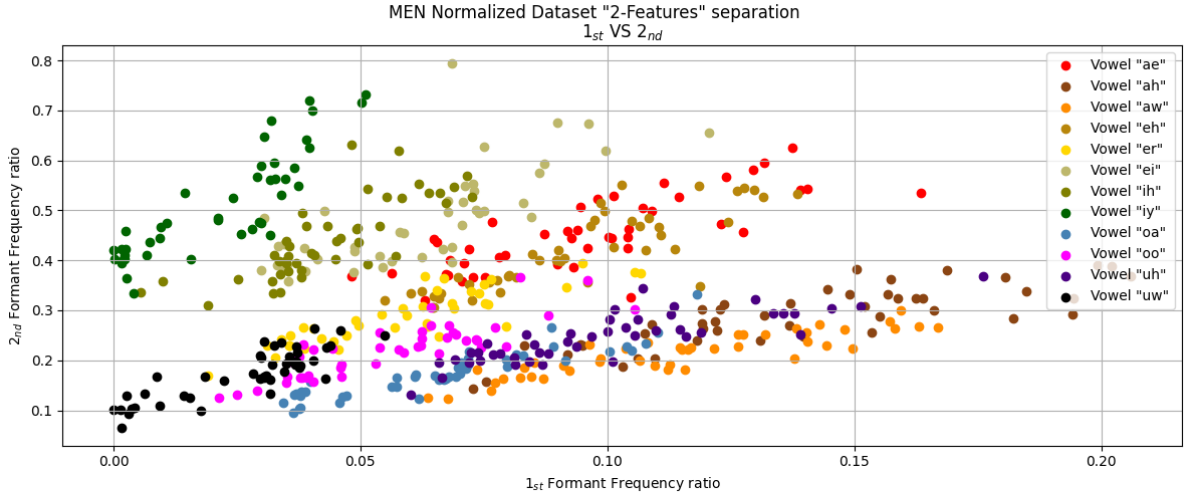

"ih" Group, Formant1 ratio
"ih" Group, Formant2 ratio
"ih" Group, Formant3 ratio
"iy" Group, Formant1 ratio
"iy" Group, Formant2 ratio
"iy" Group, Formant3 ratio
"oa" Group, Formant1 ratio
"oa" Group, Formant2 ratio
"oa" Group, Formant3 ratio
"oo" Group, Formant1 ratio
"oo" Group, Formant2 ratio
"oo" Group, Formant3 ratio
"uh" Group, Formant1 ratio
"uh" Group, Formant2 ratio
"uh" Group, Formant3 ratio
"uw" Group, Formant1 ratio
"uw" Group, Formant2 ratio
"uw" Group, Formant3 ratio
Frequency
Normalized Formant Ratio

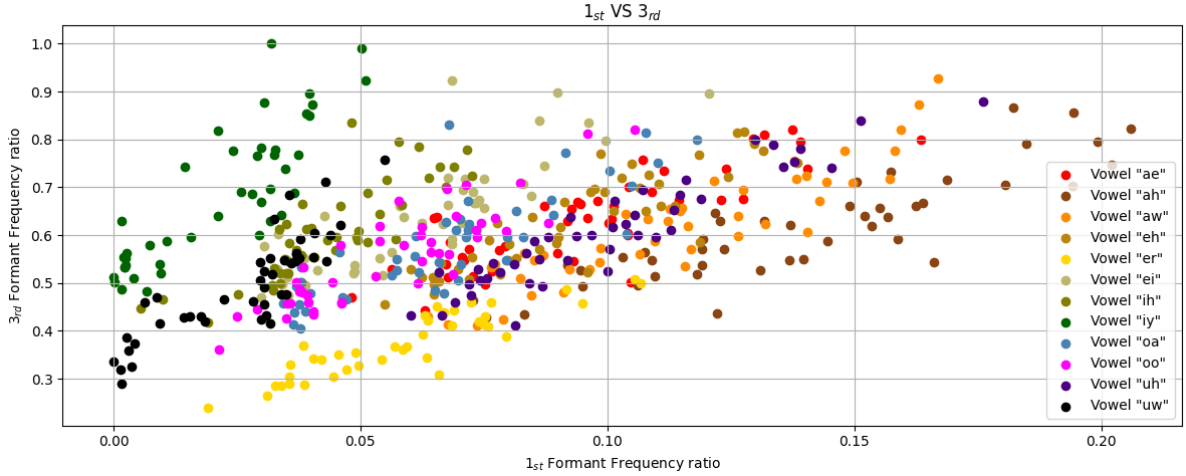

Sub-Dataset (ae - example) One-Hot Encoding
1st VS 2nd
1st VS 2nd Binarized
1st VS 3rd
1st VS 3rd Binarized
2nd VS 3rd
2nd VS 3rd Binarized
1st Formant Ratio
2nd Formant Ratio
3rd Formant Ratio
Vowel "ae"
Vowel "ah"
Vowel "aw"
Vowel "eh"
Vowel "er"
Vowel "ei"
Vowel "ih"
Vowel "iy"
Vowel "oa"
Vowel "oo"
Vowel "uh"
Vowel "uw"
Rest

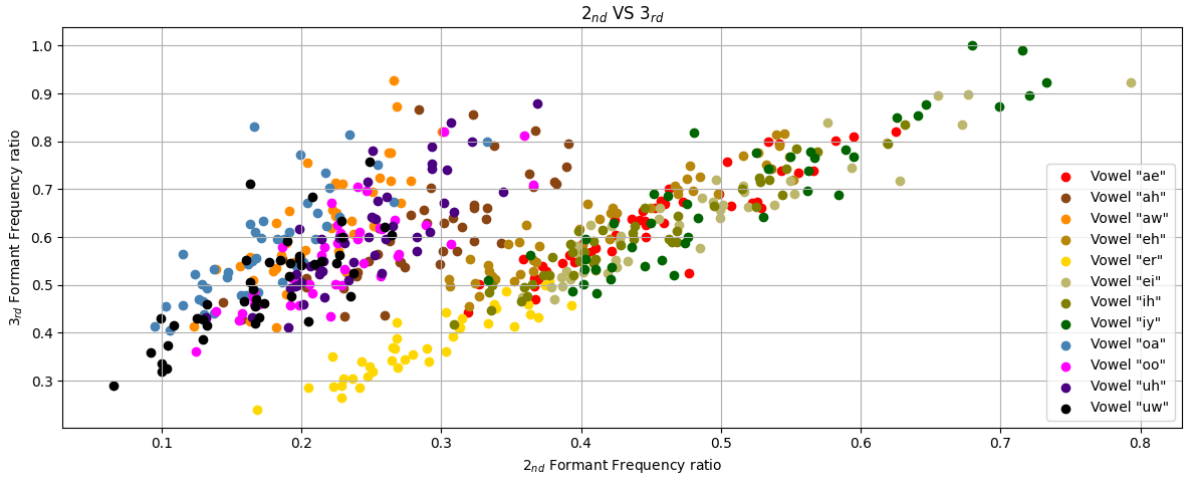

Sub-Dataset ("B"-speaker - example) One-Hot Encoding
1st VS 2nd
1st VS 2nd Binarized
1st VS 3rd
1st VS 3rd Binarized
2nd VS 3rd
2nd VS 3rd Binarized
"b"-speaker
"g"-speaker
"m"-speaker
"w"-speaker
"w"-speaker
Rest
1st Formant Ratio
2nd Formant Ratio
3rd Formant Ratio

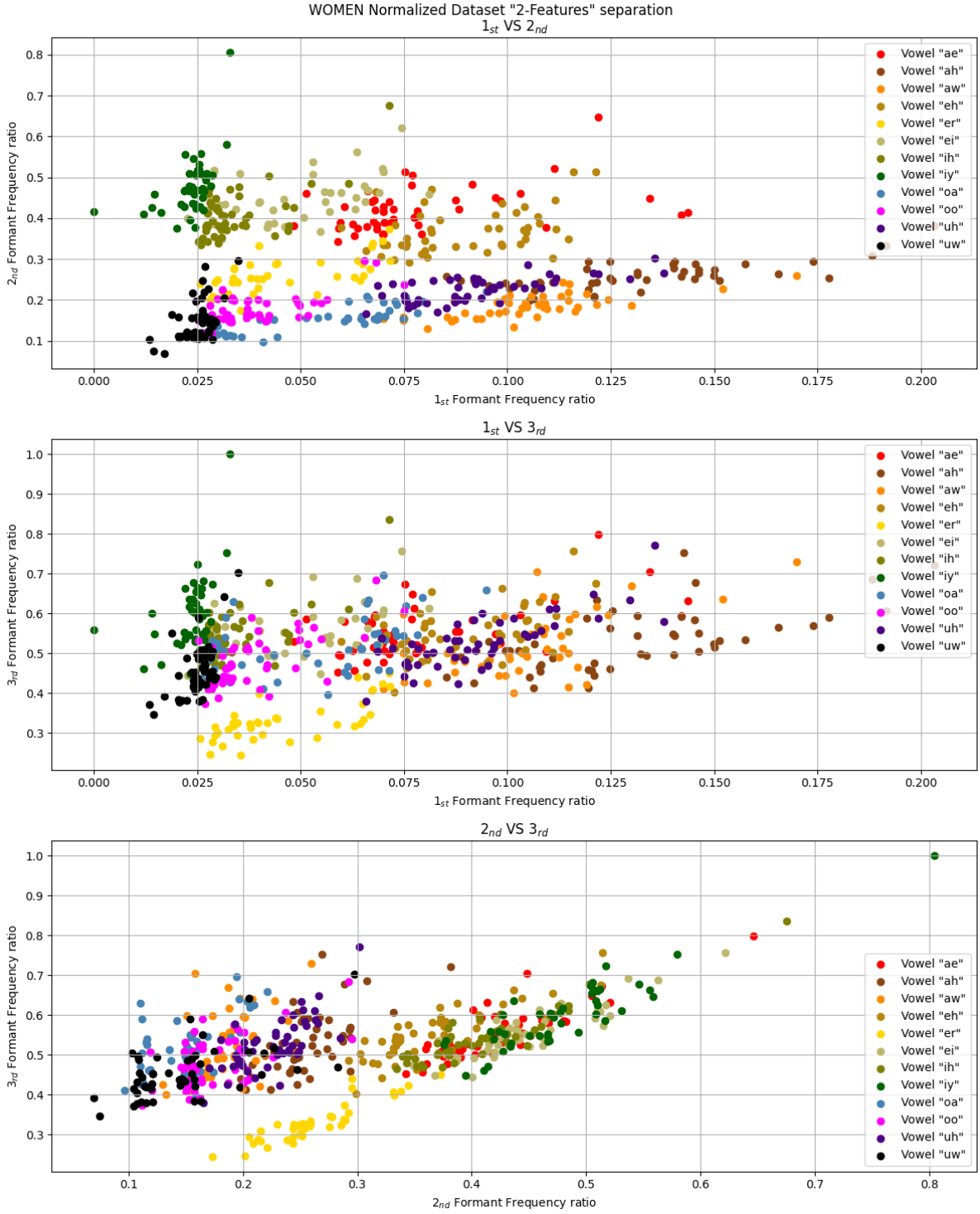

Sub-Dataset ("G"-speaker - example) One-Hot Encoding
1st VS 2nd
1st VS 2nd Binarized
1st VS 3rd
1st VS 3rd Binarized
2nd VS 3rd
2nd VS 3rd Binarized
"b"-speaker
"g"-speaker
"m"-speaker
"w"-speaker
Rest
1st Formant Ratio
2nd Formant Ratio
3rd Formant Ratio

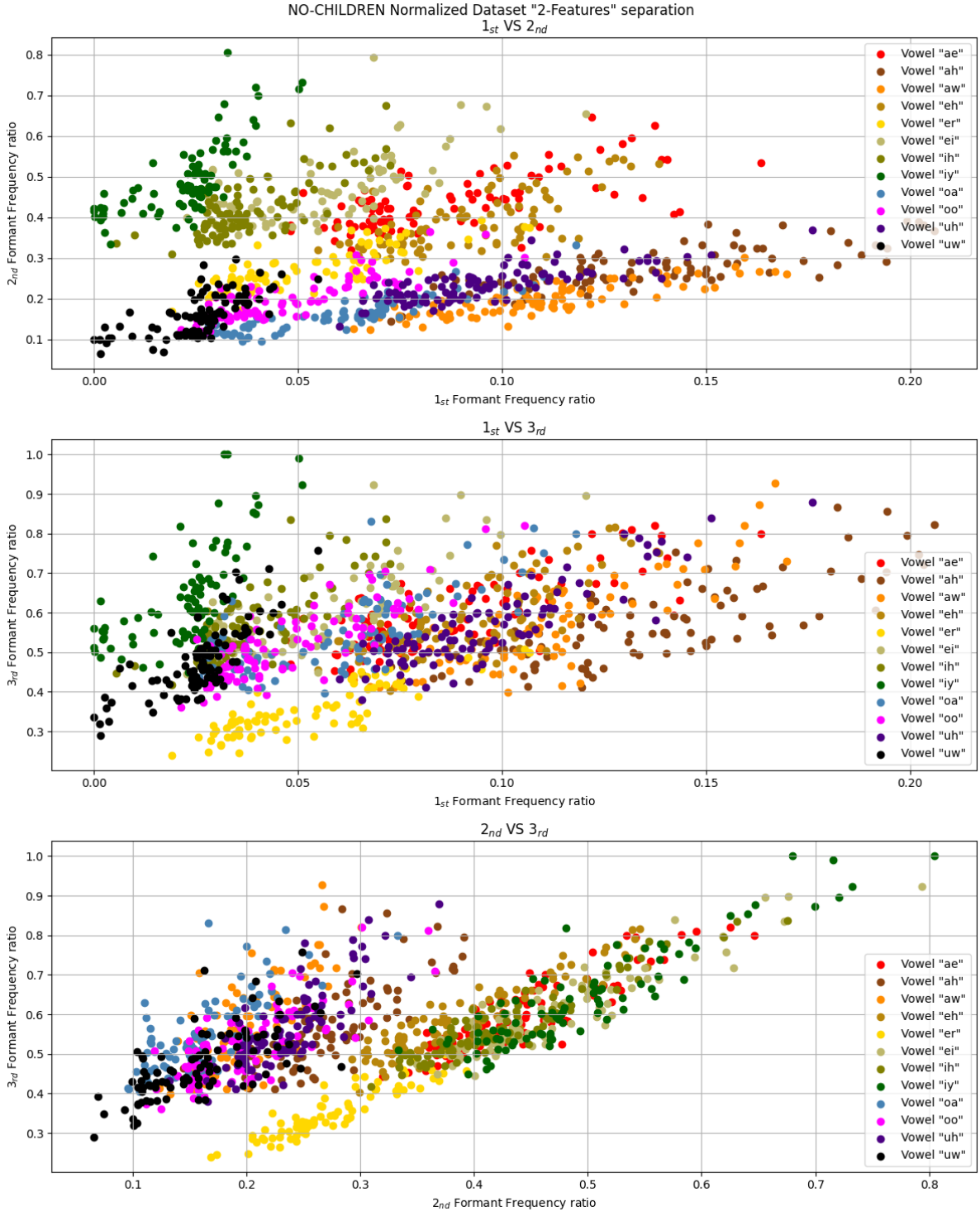

Sub-Dataset ("C"-speaker - example) One-Hot Encoding
1st VS 2nd
"c"-speaker
"m"-speaker
"w"-speaker
2nd Formant Ratio
1st Formant Ratio
1st VS 2nd Binarized
"w"-speaker
Rest
1st VS 3rd
3rd Formant Ratio
1st VS 3rd Binarized
2nd VS 3rd
2nd VS 3rd Binarized


MEN Normalized Dataset "2-Features" separation
1st VS 2nd
2nd Formant Frequency ratio
1st Formant Frequency ratio
Vowel "ae"
Vowel "ah"
Vowel "aw"
Vowel "eh"
Vowel "er"
Vowel "ei"
Vowel "ih"
Vowel "iy"
Vowel "oa"
Vowel "oo"
Vowel "uh"
Vowel "uw"

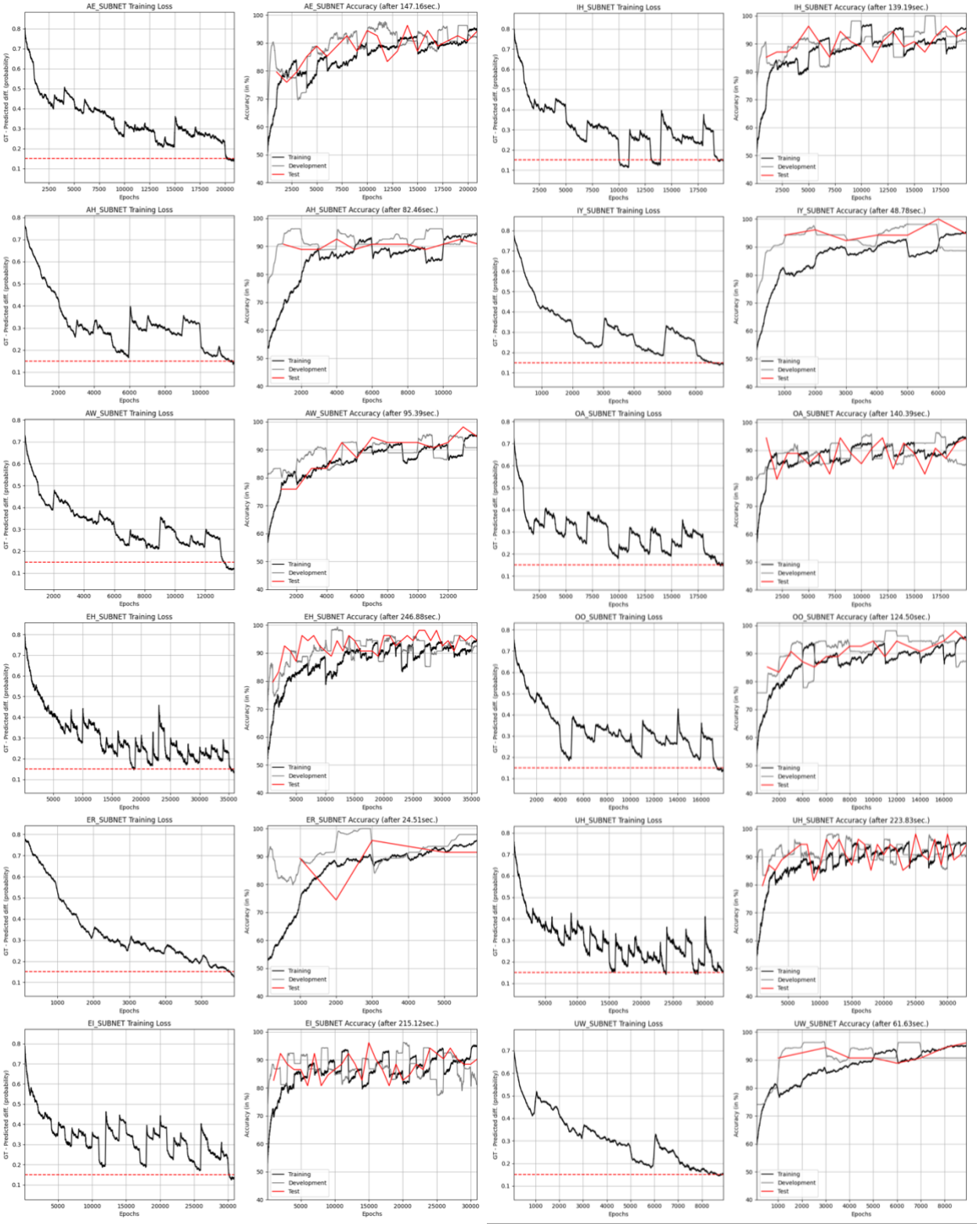

1st VS 3rd
3rd Formant Frequency ratio
1st Formant Frequency ratio
Vowel "ae"
Vowel "ah"
Vowel "aw"
Vowel "eh"
Vowel "er"
Vowel "ei"
Vowel "ih"
Vowel "iy"
Vowel "oa"
Vowel "oo"
Vowel "uh"
Vowel "uw"


2nd VS 3rd
3rd Formant Frequency ratio
2nd Formant Frequency ratio
Vowel "ae"
Vowel "ah"
Vowel "aw"
Vowel "eh"
Vowel "er"
Vowel "ei"
Vowel "ih"
Vowel "iy"
Vowel "oa"
Vowel "oo"
Vowel "uh"
Vowel "uw"

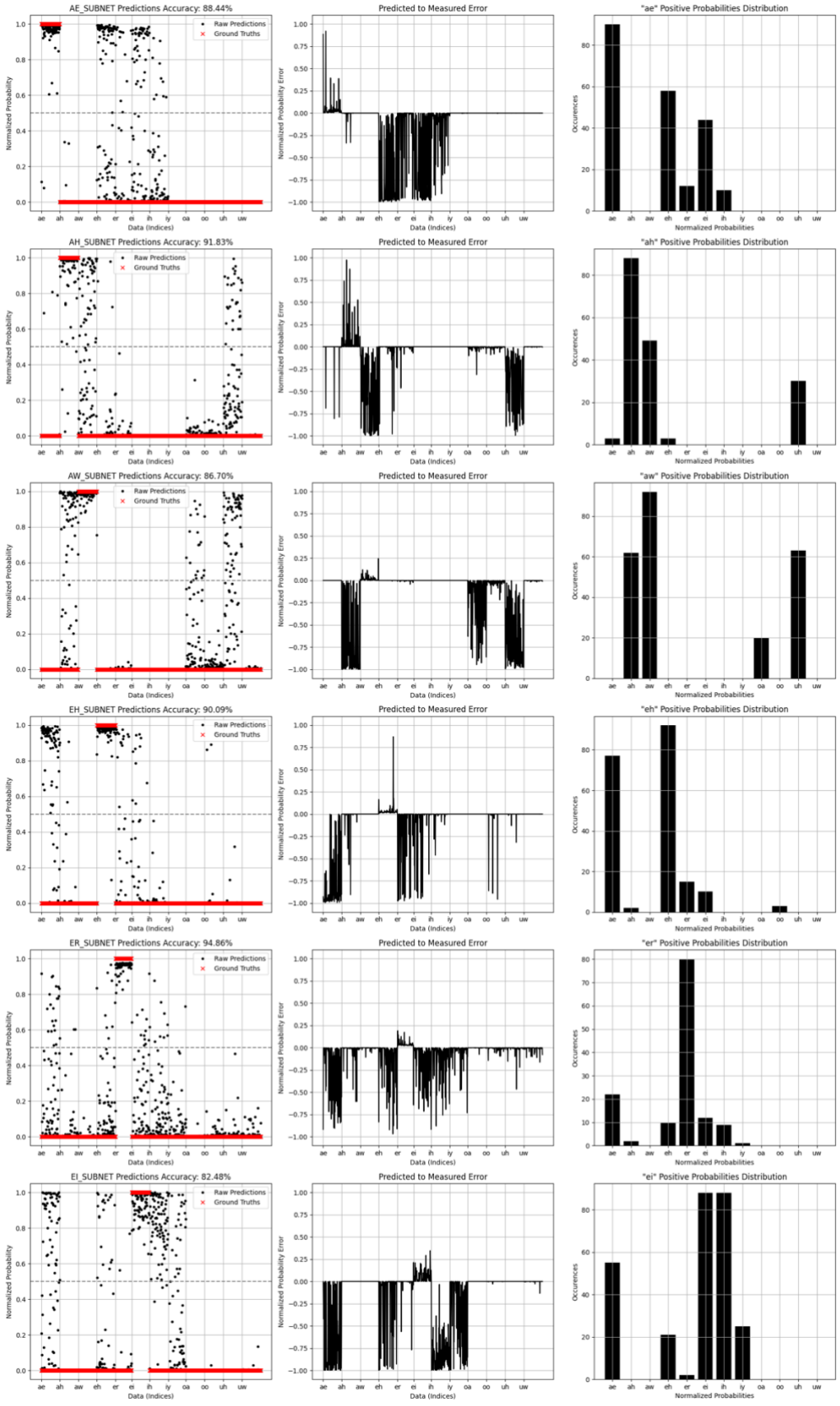

WOMEN Normalized Dataset "2-Features" separation
1st VS 2nd
2nd Formant Frequency ratio
1st Formant Frequency ratio
1st VS 3rd
3rd Formant Frequency ratio
1st Formant Frequency ratio
2nd VS 3rd
3rd Formant Frequency ratio
2nd Formant Frequency ratio
Vowel "ae"
Vowel "ah"
Vowel "aw"
Vowel "eh"
Vowel "er"
Vowel "ei"
Vowel "ih"
Vowel "iy"
Vowel "oa"
Vowel "oo"
Vowel "uh"
Vowel "uw"

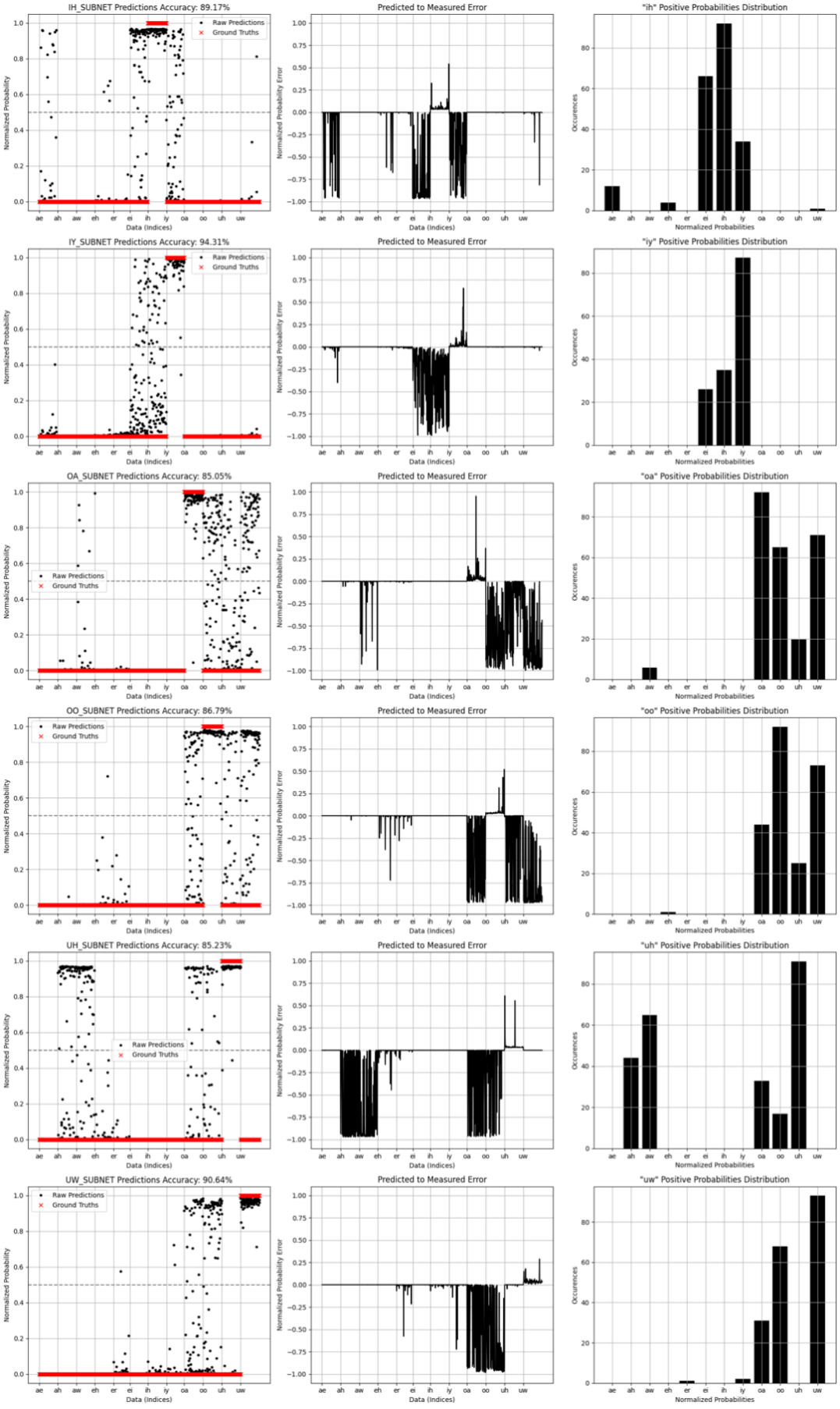

NO-CHILDREN Normalized Dataset "2-Features" separation
1st VS 2nd
2nd Formant Frequency ratio
1st Formant Frequency ratio
1st VS 3rd
3rd Formant Frequency ratio
1st Formant Frequency ratio
2nd VS 3rd
3rd Formant Frequency ratio
2nd Formant Frequency ratio
Vowel "ae"
Vowel "ah"
Vowel "aw"
Vowel "eh"
Vowel "er"
Vowel "ei"
Vowel "ih"
Vowel "iy"
Vowel "oa"
Vowel "oo"
Vowel "uh"
Vowel "uw"

OCON Training (3 Formant features *Steady State, No-Children Dataset*)

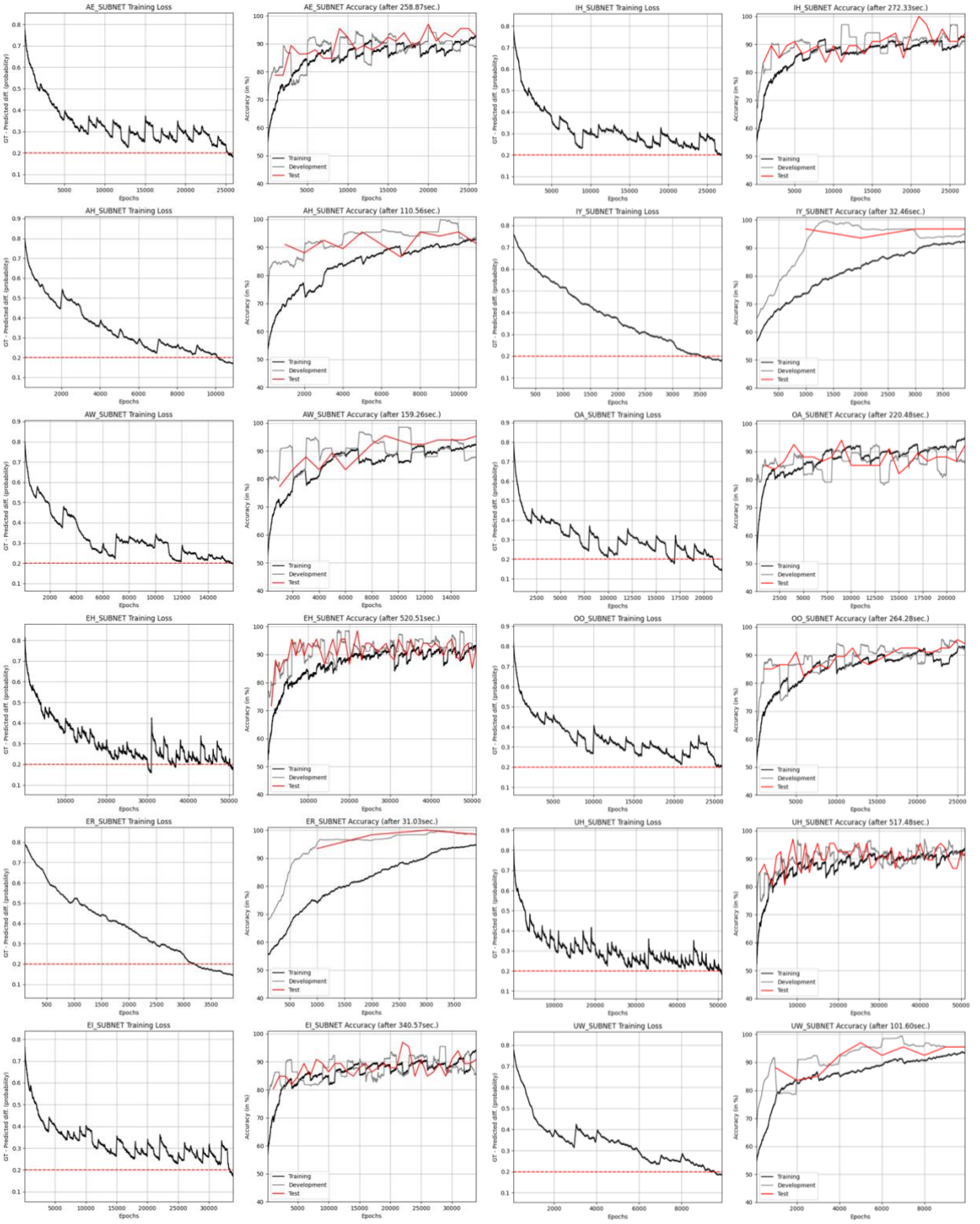


OCON Evaluation (3 Formant features *Steady State, No-Children Dataset*)

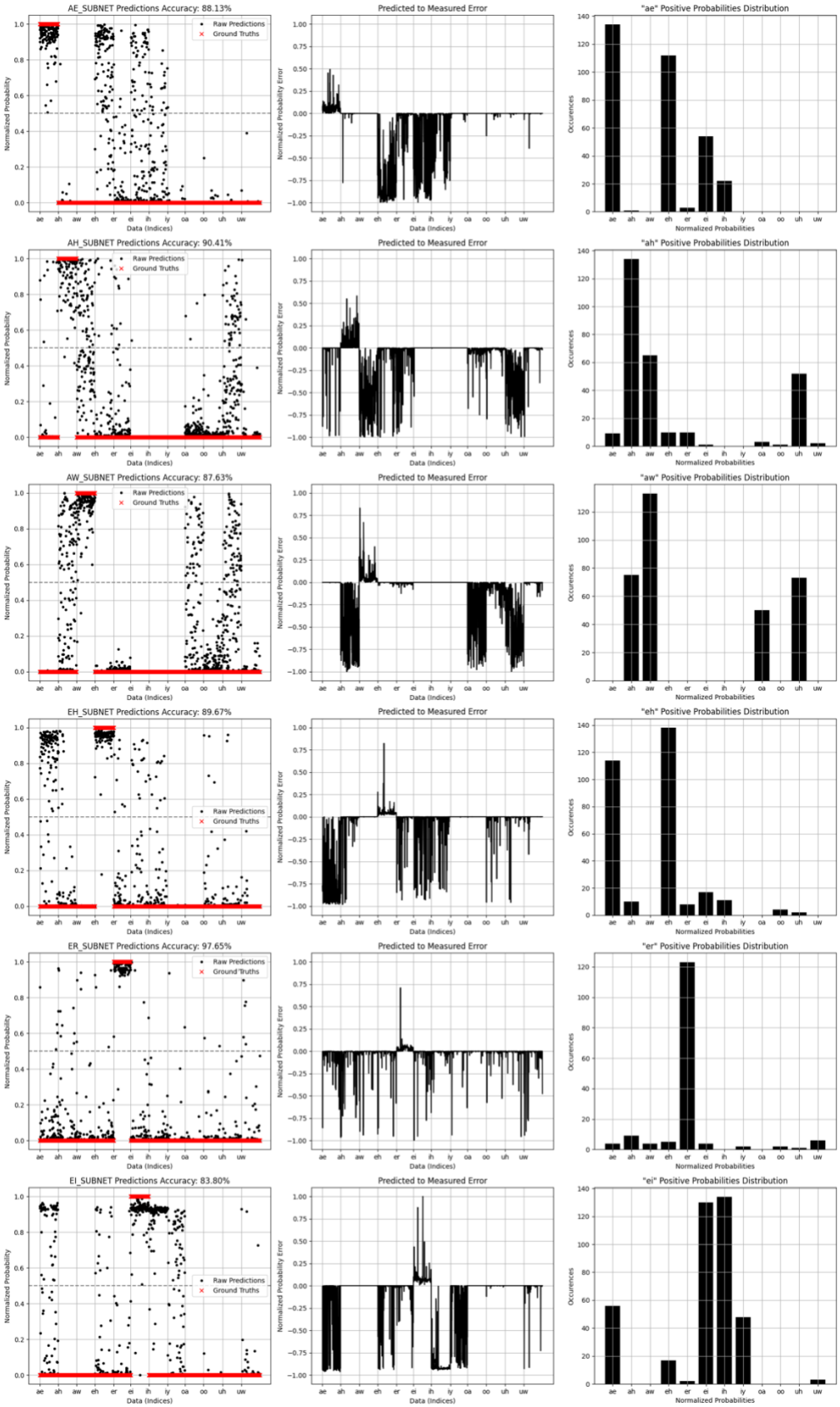


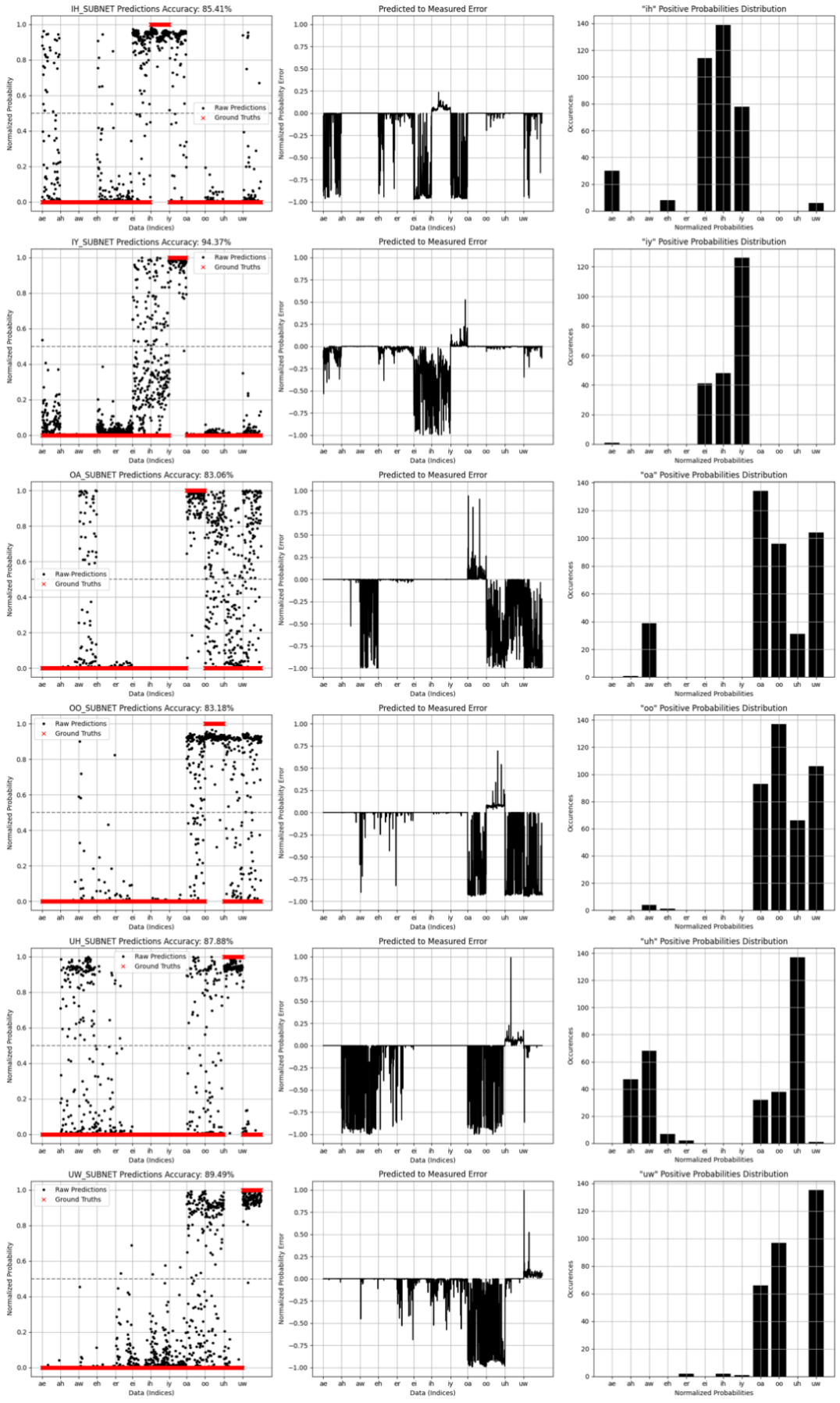

IH_SUBNET Predictions Accuracy: 89.17%
Predicted to Measured Error
"ih" Positive Probabilities Distribution
IY_SUBNET Predictions Accuracy: 94.31%
Predicted to Measured Error
"iy" Positive Probabilities Distribution
OA_SUBNET Predictions Accuracy: 85.05%
Predicted to Measured Error
"oa" Positive Probabilities Distribution
OO_SUBNET Predictions Accuracy: 86.79%
Predicted to Measured Error
"oo" Positive Probabilities Distribution
UH_SUBNET Predictions Accuracy: 85.23%
Predicted to Measured Error
"uh" Positive Probabilities Distribution
UW_SUBNET Predictions Accuracy: 90.64%
Predicted to Measured Error
"uw" Positive Probabilities Distribution
Raw Predictions
Ground Truths
Normalized Probability
Normalized Probability Error
Occurences
Data (Indices)
Normalized Probabilities
ae ah aw eh er ei ih iy oa oo uh uw

OCON Training (4 Formant features *Steady State, Fundamental Frequency*)

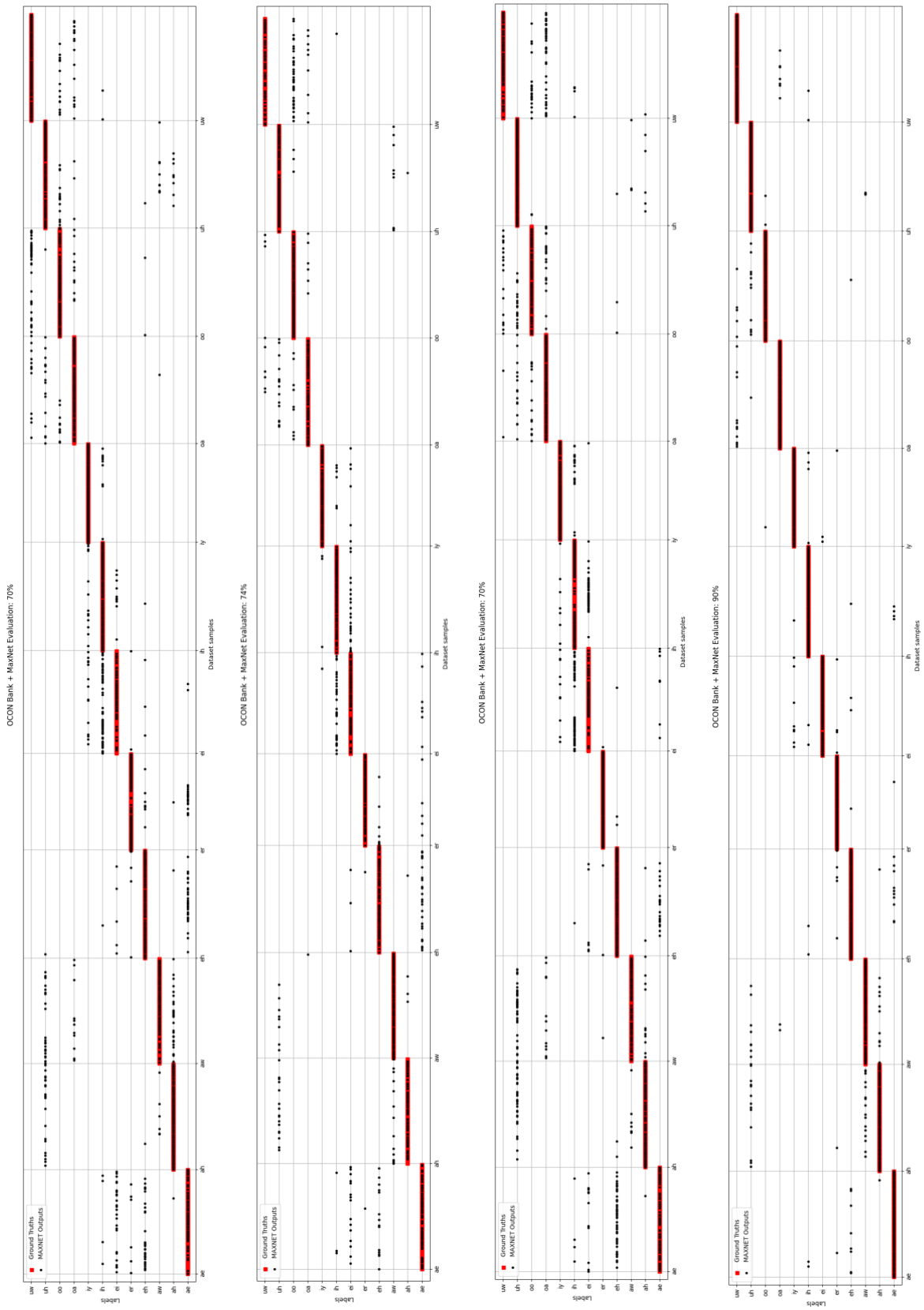


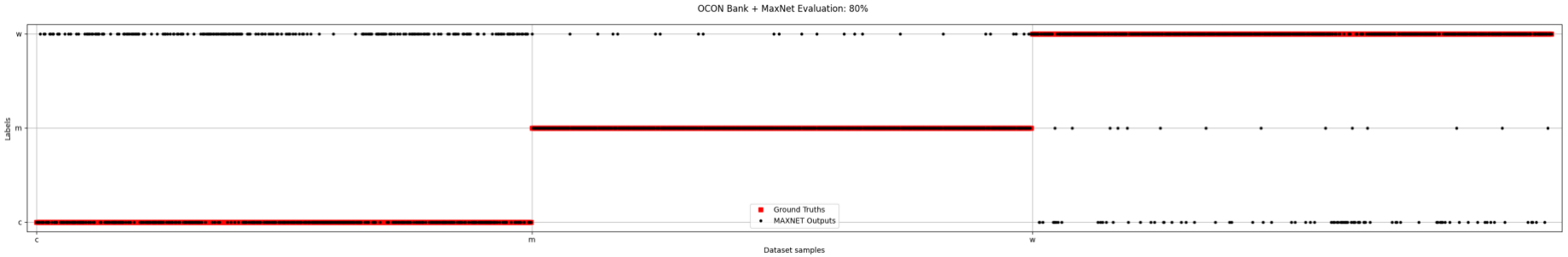
OCON Evaluation (4 Formant features *Steady State, Fundamental Frequency*)




IH_SUBNET Predictions Accuracy: 85.41%
Predicted to Measured Error
"ih" Positive Probabilities Distribution
IY_SUBNET Predictions Accuracy: 94.37%
Predicted to Measured Error
"iy" Positive Probabilities Distribution
OA_SUBNET Predictions Accuracy: 83.06%
Predicted to Measured Error
"oa" Positive Probabilities Distribution
OO_SUBNET Predictions Accuracy: 83.18%
Predicted to Measured Error
"oo" Positive Probabilities Distribution
UH_SUBNET Predictions Accuracy: 87.88%
Predicted to Measured Error
"uh" Positive Probabilities Distribution
UW_SUBNET Predictions Accuracy: 89.49%
Predicted to Measured Error
"uw" Positive Probabilities Distribution
Raw Predictions
Ground Truths
Normalized Probability
Normalized Probability Error
Occurences
Data (Indices)
Normalized Probabilities
ae ah aw eh er ei ih iy oa oo uh uw

OCON Model Output Accuracies




OCON Bank + MaxNet Evaluation: 80%
w
m
c
Labels
Ground Truths
MAXNET Outputs
c
m
w
Dataset samples

FIGURES (INDEX)