

Εθνικό Μετσόβιο Πολυτεχνείο

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Ακαδημαϊκό Έτος: 2020-2021



Αναγνώριση Προτύπων

2η Εργαστηριακή Άσκηση

Θέμα: Αναγνώριση φωνής με Κρυφά Μαρκοβιανά Μοντέλα και
Αναδρομικά Νευρωνικά Δίκτυα

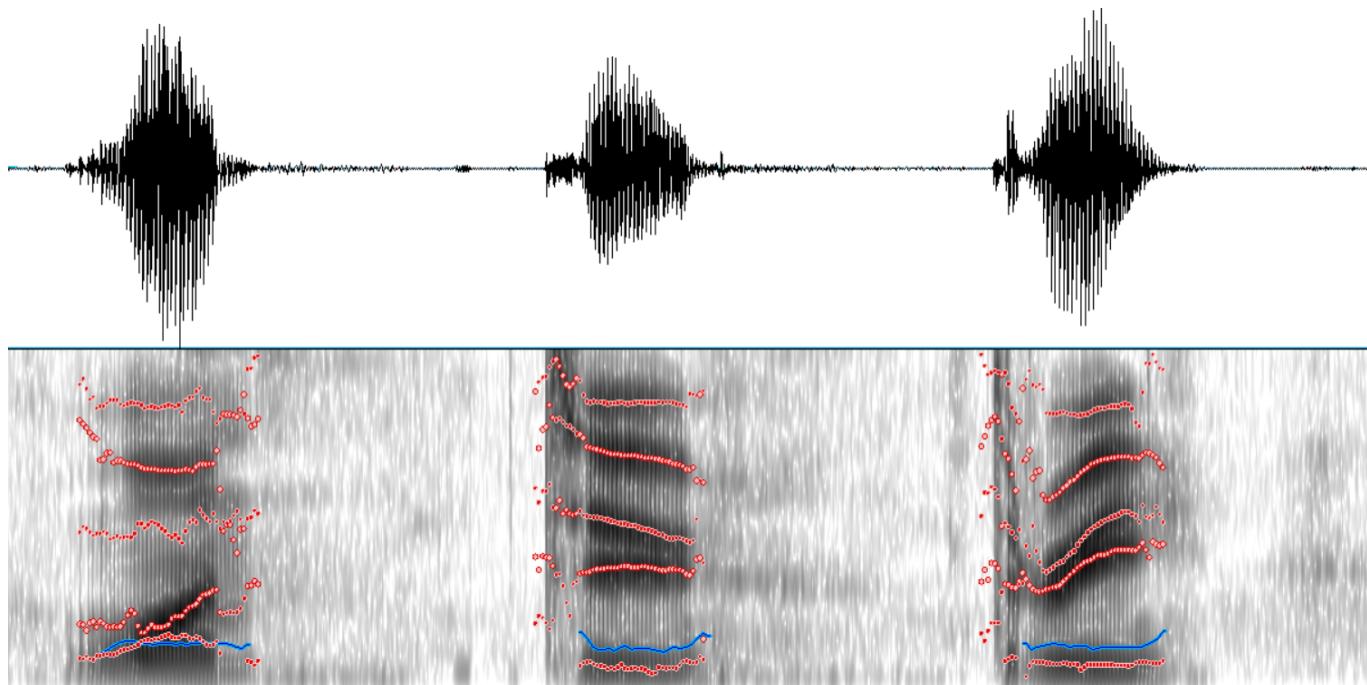
Τζε Χριστίνα-Ουρανία | 03116079
Ψαρουδάκης Ανδρέας | 03116001

Εισαγωγή

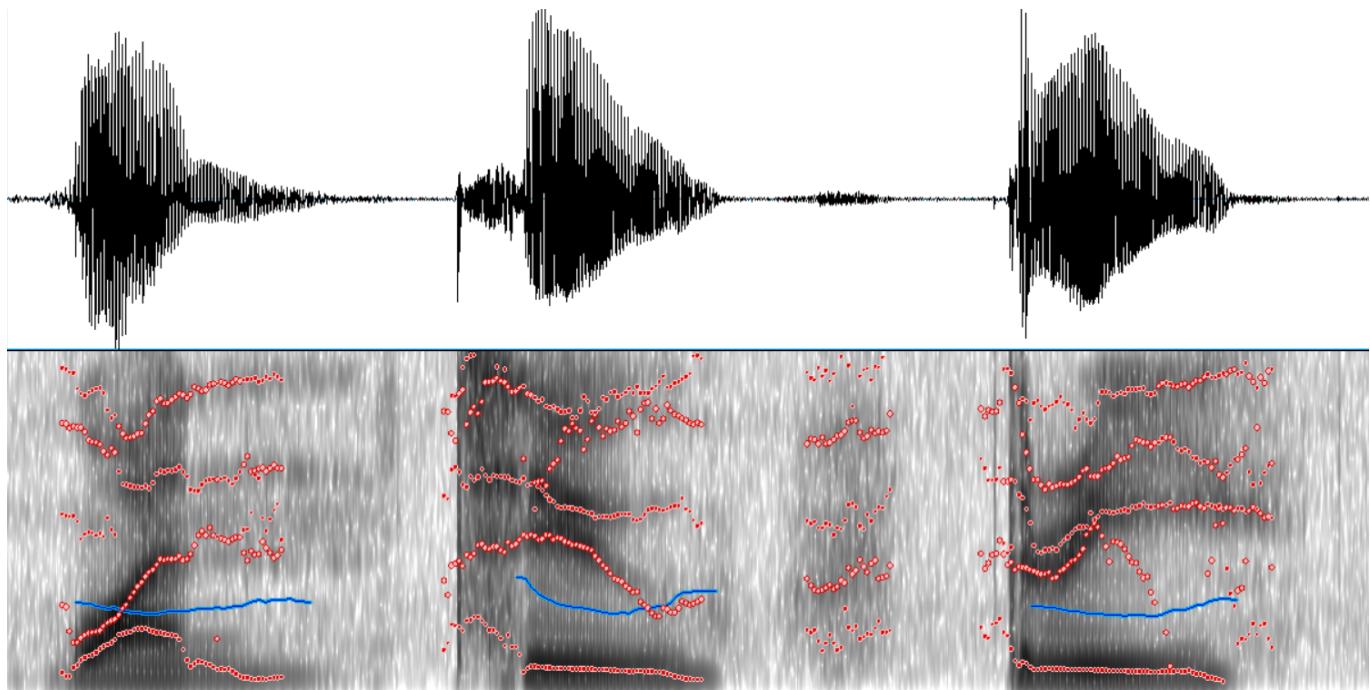
Σκοπός της παρούσας εργαστηριακής άσκησης είναι η υλοποίηση ενός συστήματος επεξεργασίας και αναγνώρισης φωνής, με εφαρμογή σε αναγνώριση μεμονωμένων λέξεων. Το πρώτο μέρος αποσκοπεί στην εξαγωγή κατάλληλων ακουστικών χαρακτηριστικών από φωνητικά δεδομένα, καθώς και στην ανάλυση και απεικόνισή τους, με σκοπό την κατανόηση και την εξαγωγή χρησιμών πληροφοριών από αυτά. Πιο συγκεκριμένα, το σύστημα που θα αναπτύξουμε αφορά σε αναγνώριση μεμονωμένων ψηφίων στα Αγγλικά. Τα δεδομένα που χρησιμοποιούμε περιέχουν εκφωνήσεις 9 ψηφίων από 15 διαφορετικούς ομιλητές σε ξεχωριστά .wav αρχεία. Συνολικά χρησιμοποιούμε 133 αρχεία. Τα ονόματα των αρχείων (π.χ. eight8.wav) υποδηλώνουν τόσο το ψηφίο που εκφωνείται (π.χ. eight), όσο και τον ομιλητή. Οι εκφωνήσεις έχουν ηχογραφηθεί με συχνότητα δειγματοληψίας ίση με $F_s = 16\text{kHz}$ και η διάρκειά τους διαφέρει.

Βήμα 1: Ανάλυση αρχείων ήχου με το Praat

Εγκαθιστούμε το Praat και ανοίγουμε μέσω αυτού τα αρχεία ήχου **onetwothree1.wav** και **onetwothree8.wav**. Τα αρχεία αυτά περιέχουν την πρόταση “**one two three**” από τους ομιλητές 1 και 8, οι οποίοι είναι άντρας και γυναίκα αντίστοιχα. Οι κυματομορφές και τα αντίστοιχα spectrograms τους φαίνονται στα ακόλουθα σχήματα:



Σχήμα 1: Κυματομορφή και spectrogram για την πρόταση “one two three” από τον ομιλητή 1 (άνδρας)



Σχήμα 2: Κυματομορφή και spectrogram για την πρόταση “one two three” από τον ομιλητή 8 (γυναίκα)

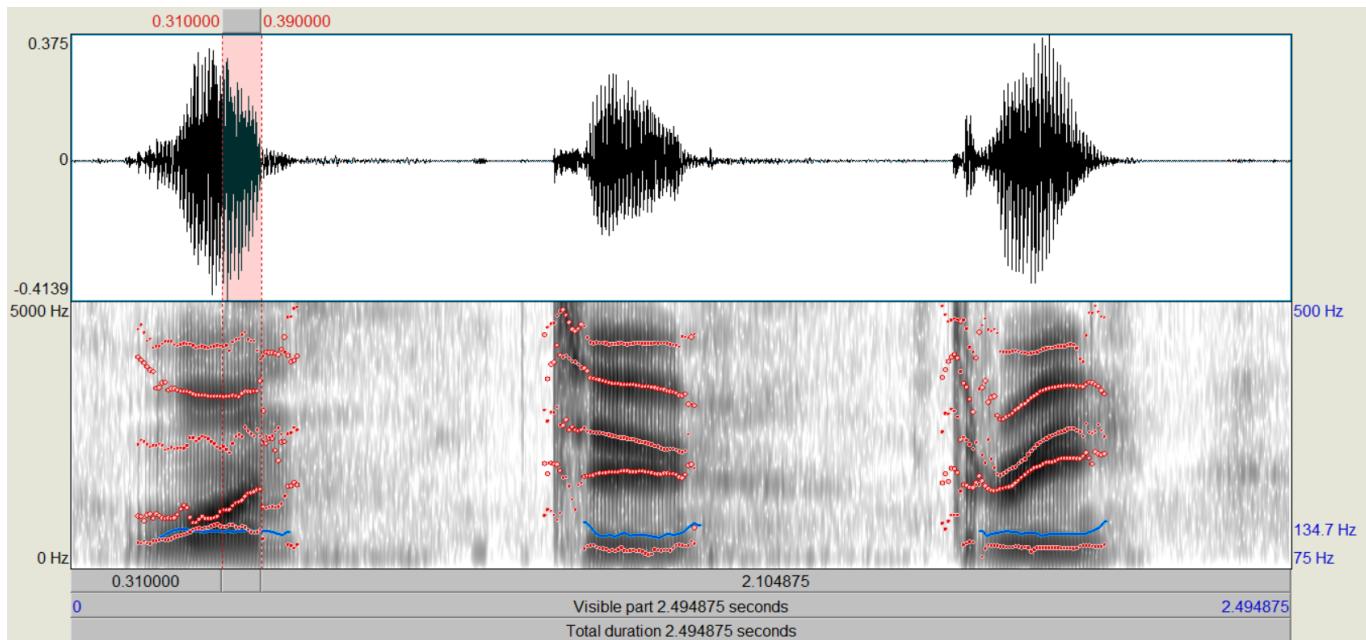
Οι μπλε γραμμές υποδεικνύουν το **pitch level** ενώ τα **χόκκινα σημεία** αναπαριστούν τα διάφορα **formants**, τα οποία αποτελούν συγκεντρώσεις ακουστικής ενέργειας γύρω από συγκεκριμένες συχνότητες.

Παρατηρούμε ότι το pitch της γυναικείας φωνής (ομιλητής 8) που απεικονίζεται στο Σχήμα 2 είναι αυξημένο σε σχέση με το pitch της αντρικής φωνής (ομιλητής 1) του Σχήματος 1, για όλα τα φωνήντα. Επίσης, τα formants της αντρικής φωνής φαίνεται να έχουν μια πιο συμπαγή και αυστηρά ορισμένη αναπαράσταση, σε αντίθεση με αυτά της γυναικείας φωνής. Τέλος, η κυματομορφή της γυναίκας ομιλήτριας απλώνεται περισσότερο στο χρόνο για κάθε φωνήν σε σχέση με αυτή του άντρα ομιλητή, παράγοντας έναν πιο καθαρό ήχο.

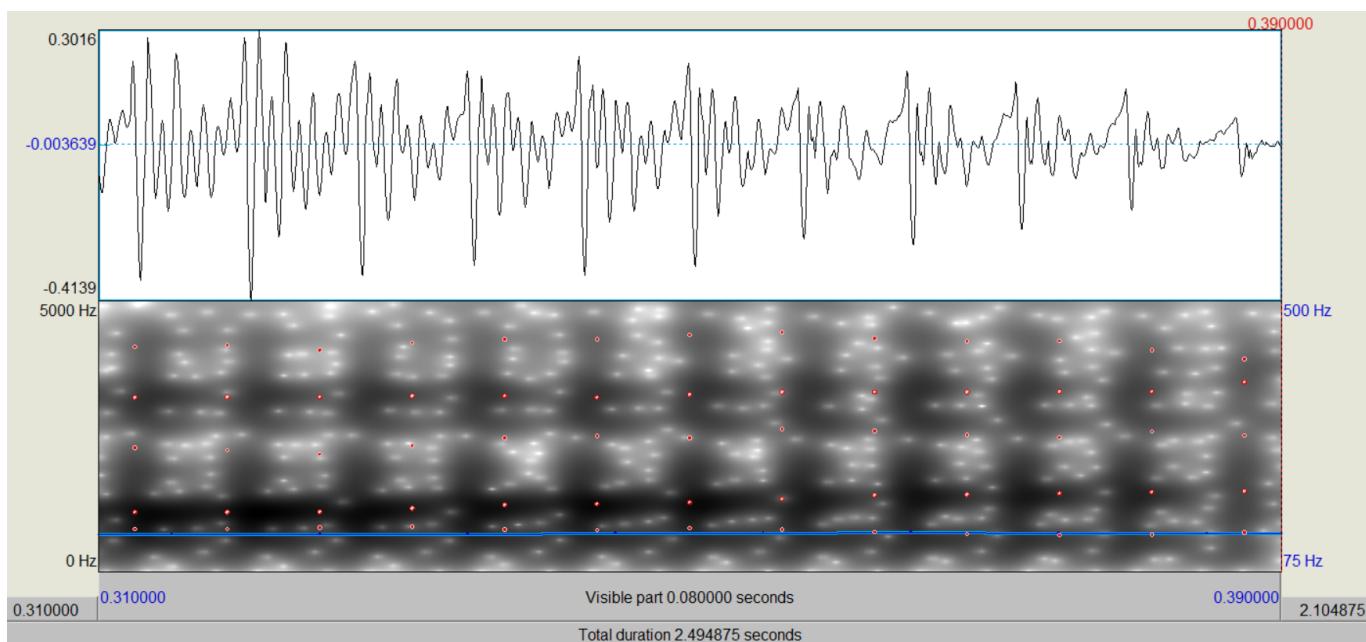
Θα εξάγουμε τώρα τη μέση τιμή του pitch στα φωνήντα “α”, “ου”, “ι” για τα 3 ψηφία και για κάθε ομιλητή καθώς επίσης και τα 3 πρώτα formants του κάθε φωνήντος. Έχουμε λοιπόν:

- Για τον άντρα ομιλητή (ομιλητής 1):

- Για το φωνήγεν “α”



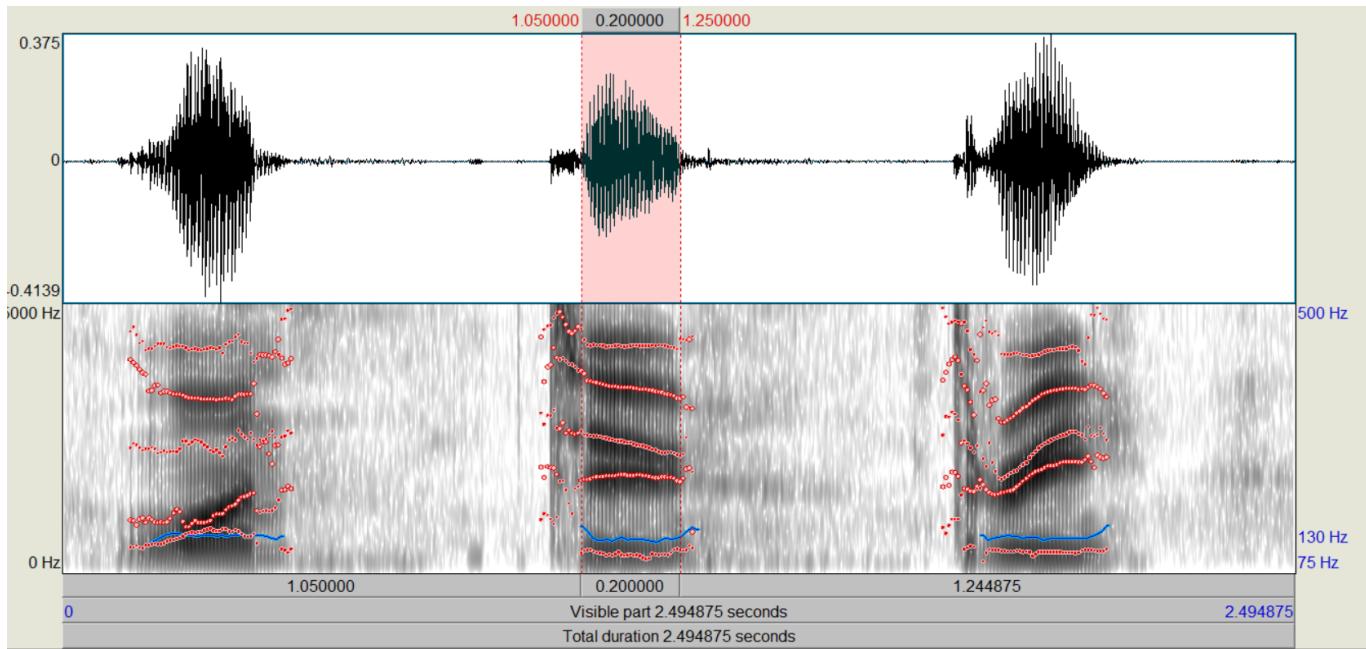
Σχήμα 3: Χρονικό διάστημα φωνήγεντος “α” για την πρόταση “one two three” από τον ομιλητή 1



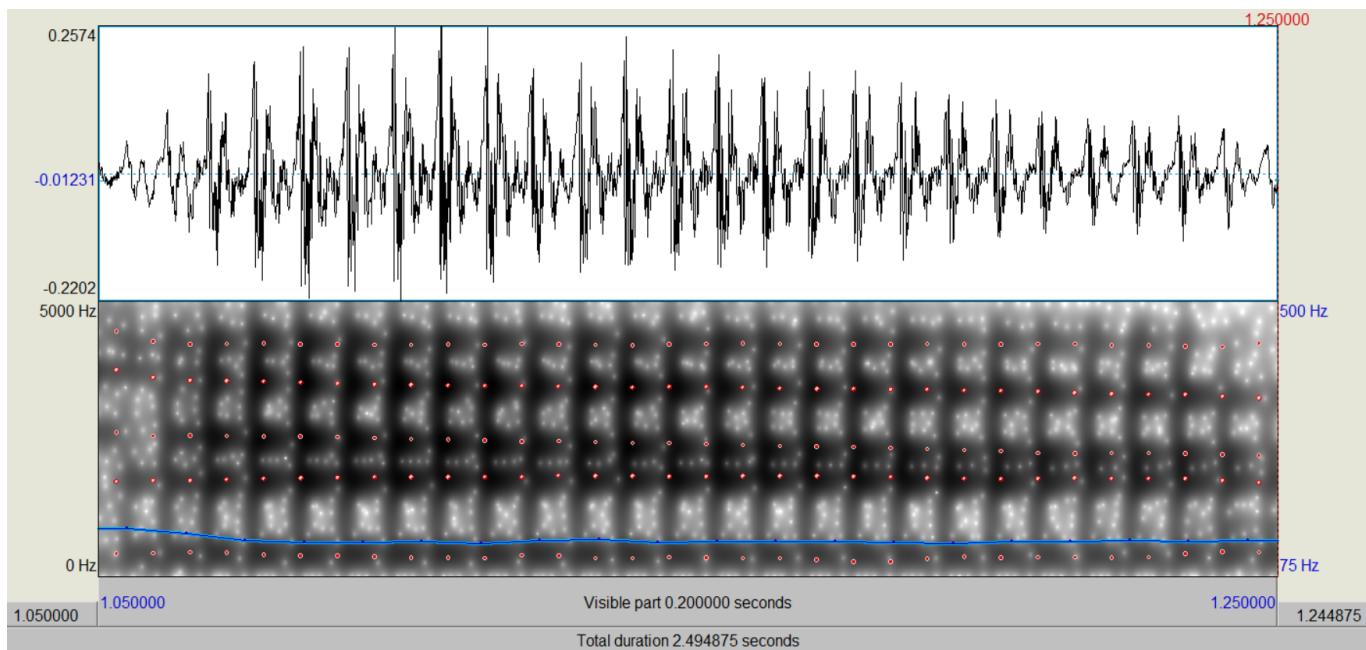
Σχήμα 4: Μεγέθυνση στο χρονικό διάστημα του φωνήγεντος “α” που απεικονίζεται με κόκκινο στο Σχήμα 3

Φωνήγεν “α”			
Μέση τιμή pitch	1o formant	2o formant	3o formant
134.7 Hz	749.91 Hz	1283.25 Hz	2434.89 Hz

– Για το φωνήγεν “ou”



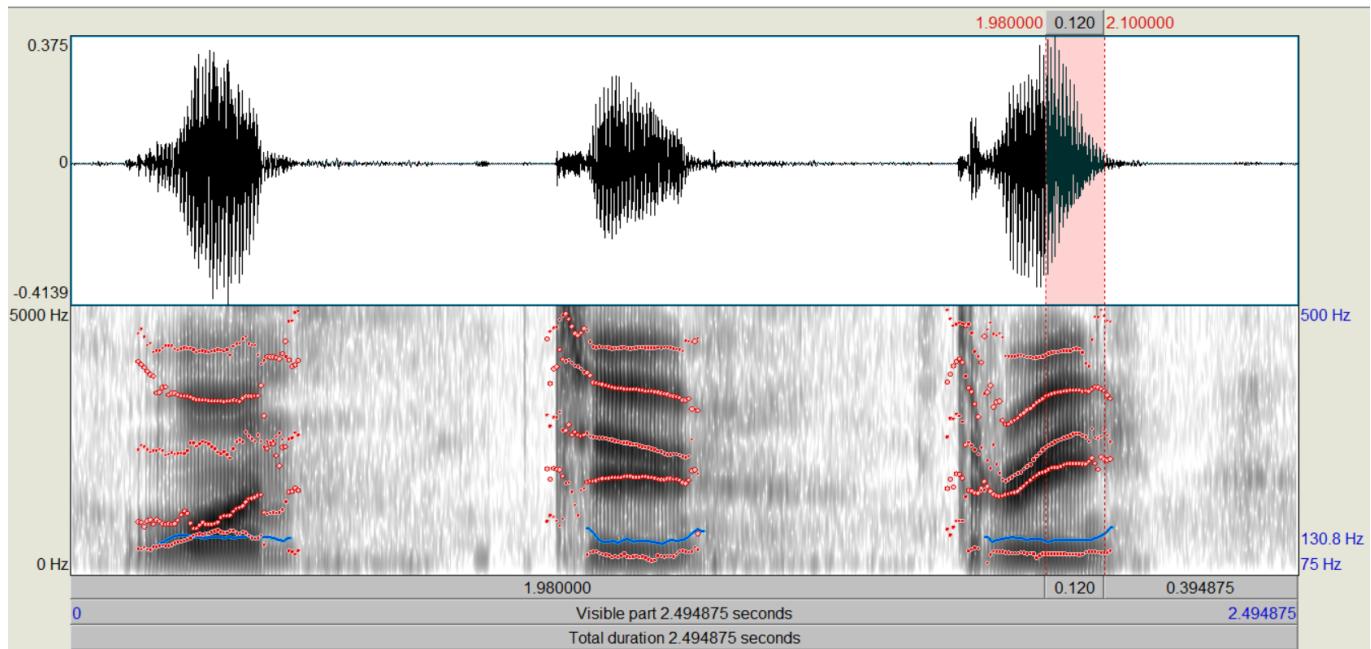
Σχήμα 5: Χρονικό διάστημα φωνήγεντος “ou” για την πρόταση “one two three” από τον ομιλητή 1



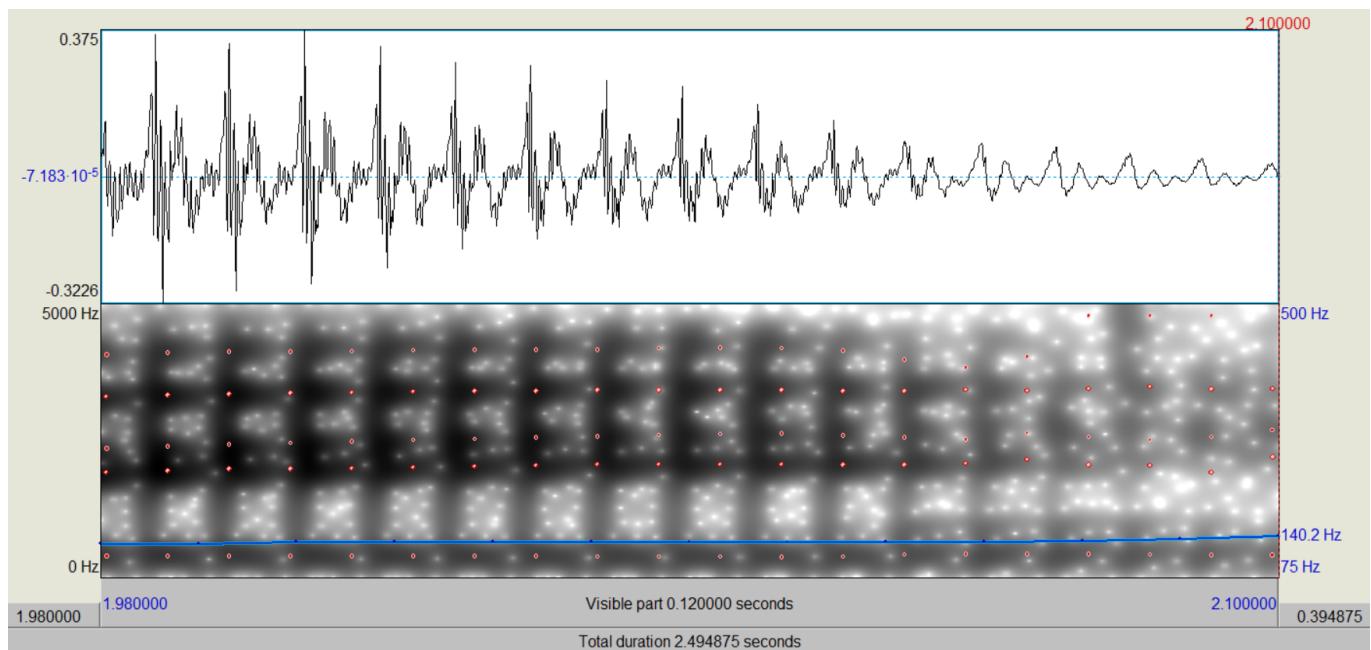
Σχήμα 6: Μεγέθυνση στο χρονικό διάστημα του φωνήγεντος “ou” που απεικονίζεται με κόκκινο στο Σχήμα 5

Φωνήγεν “ou”			
Μέση τιμή pitch	1o formant	2o formant	3o formant
129.96 Hz	352.95 Hz	1779.09 Hz	2389.21 Hz

— Για το φωνήγεν “ι”



Σχήμα 7: Χρονικό διάστημα φωνήγεντος “ι” για την πρόταση “one two three” από τον ομιλητή 1

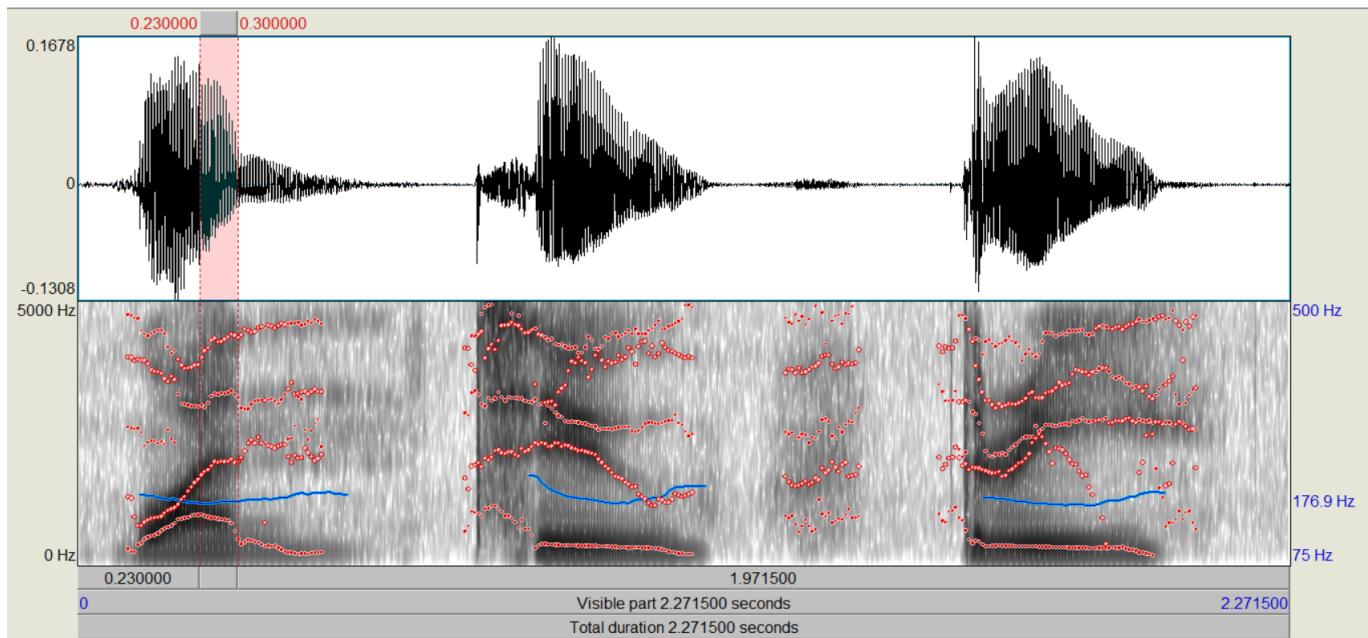


Σχήμα 8: Μεγέθυνση στο χρονικό διάστημα του φωνήγεντος “ι” που απεικονίζεται με κόκκινο στο Σχήμα 7

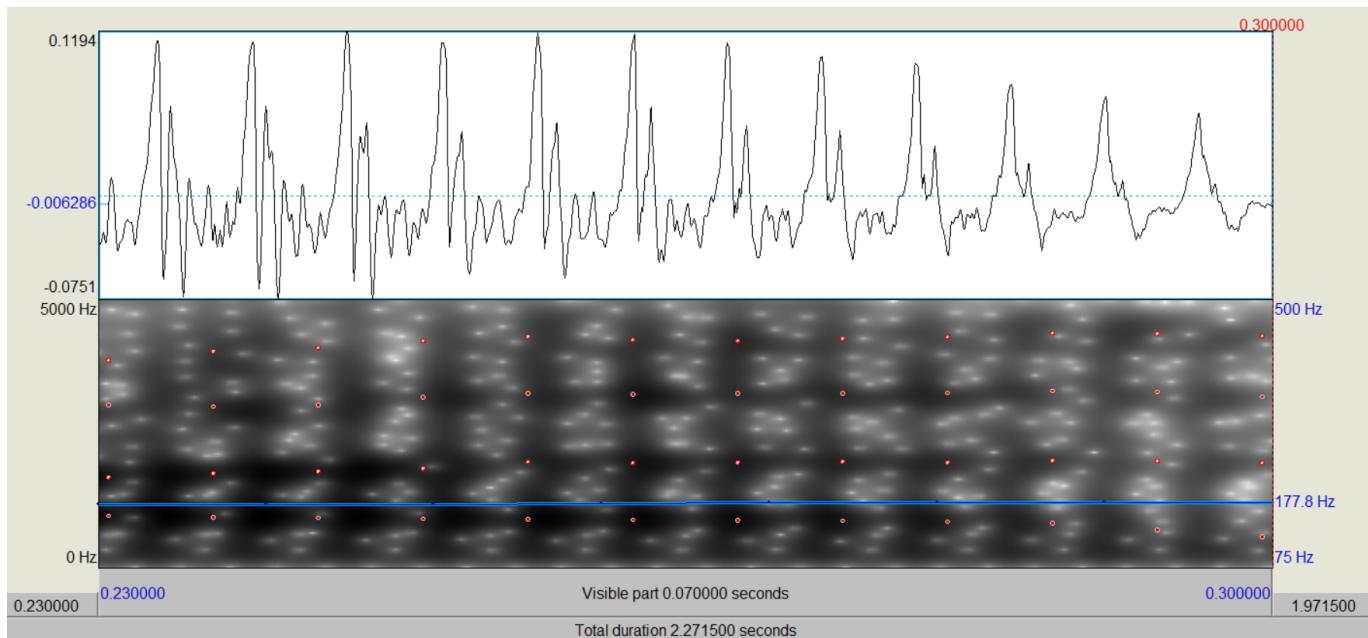
Φωνήγεν “ι”			
Μέση τιμή pitch	1o formant	2o formant	3o formant
130.79 Hz	393.09 Hz	2039.69 Hz	2533.54 Hz

- Για τη γυναίκα ομιλητή (ομιλητής 8):

– Για το φωνήν “α”



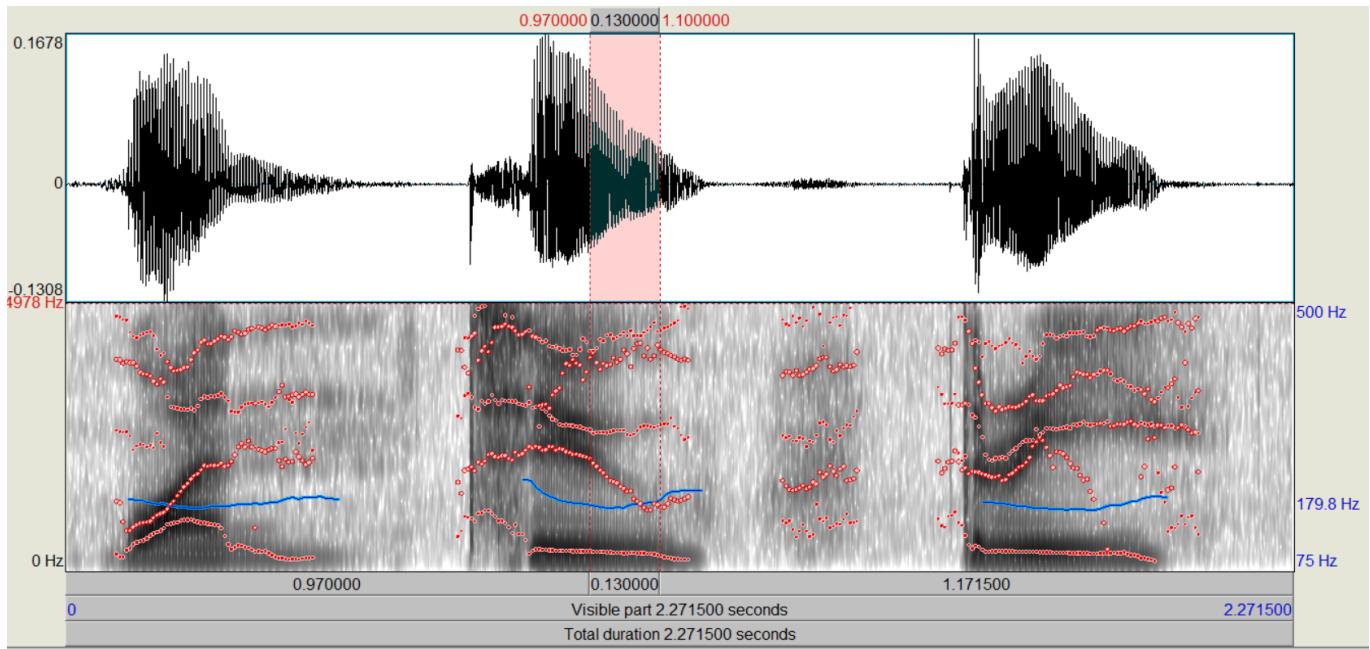
Σχήμα 9: Χρονικό διάστημα φωνήντος “α” για την πρόταση “one two three” από τον ομιλητή 8



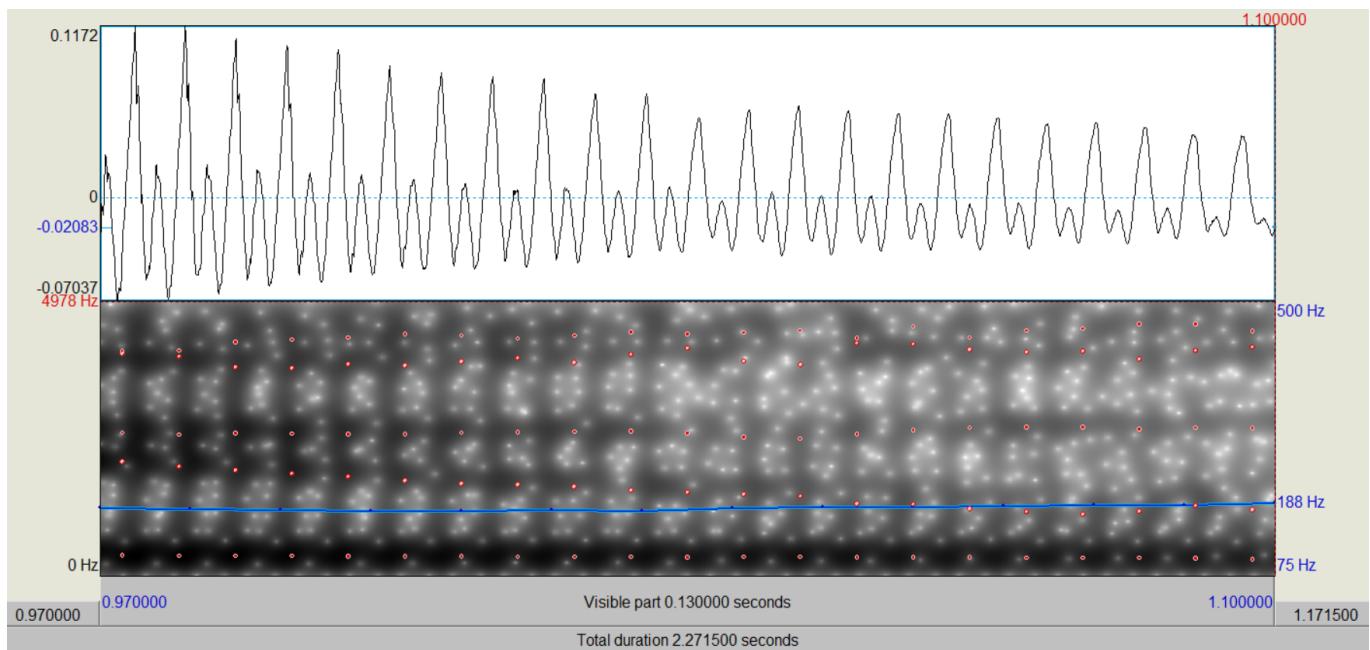
Σχήμα 10: Μεγέθυνση στο χρονικό διάστημα του φωνήντος “α” που απεικονίζεται με κόκκινο στο Σχήμα 9

Φωνήν “α”			
Μέση τιμή pitch	1o formant	2o formant	3o formant
176.89 Hz	854.29 Hz	1899.81 Hz	3183.09 Hz

– Για το φωνήγεν “ou”



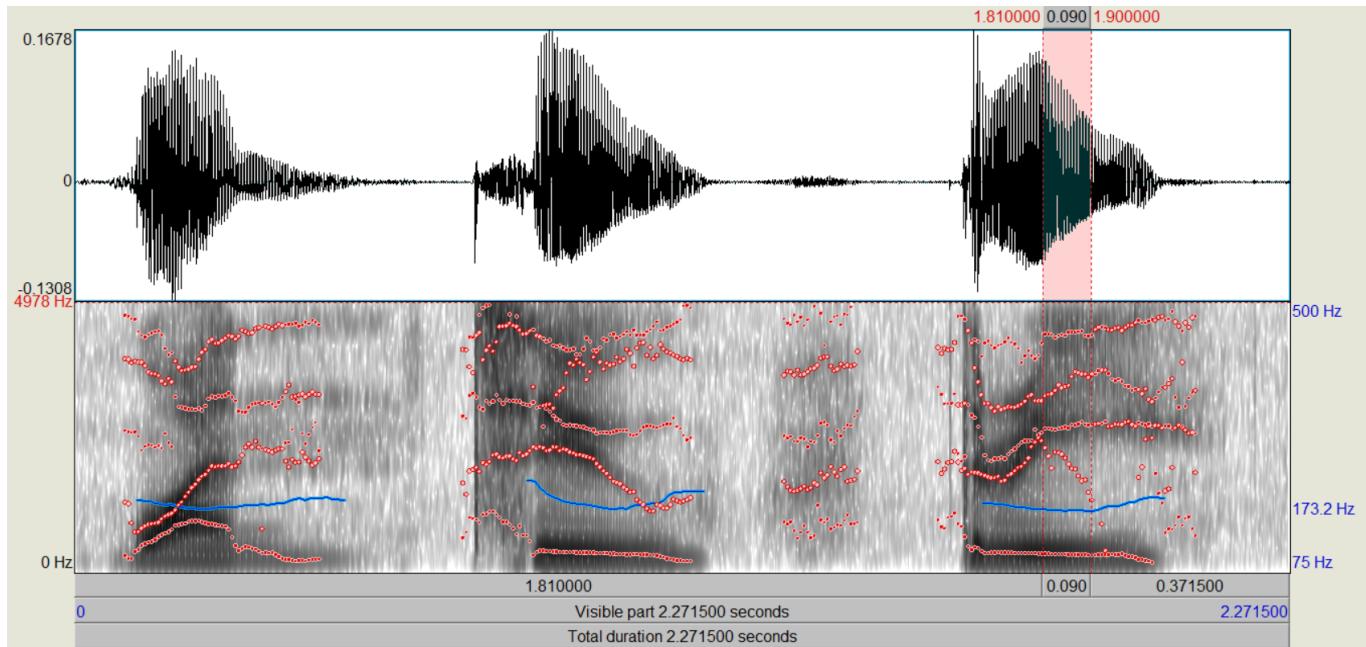
Σχήμα 11: Χρονικό διάστημα φωνήγεντος “ou” για την πρόταση “one two three” από τον ομιλητή 8



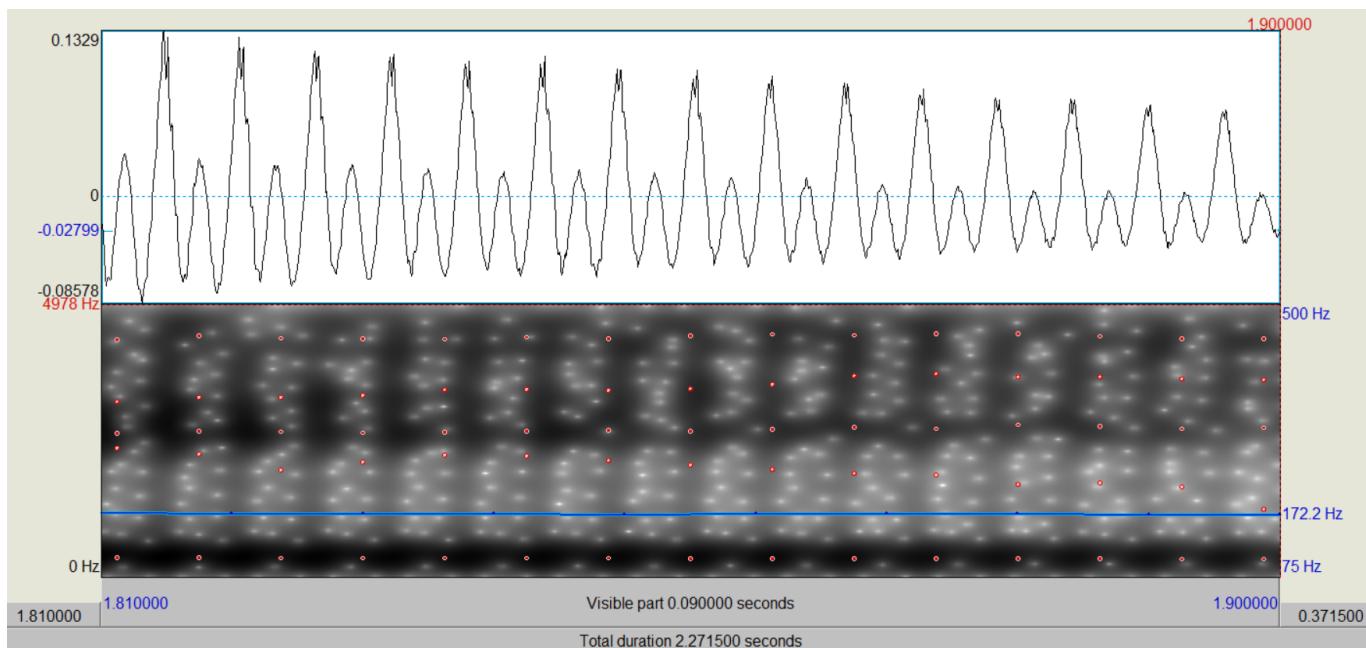
Σχήμα 12: Μεγέθυνση στο χρονικό διάστημα του φωνήγεντος “ou” που απεικονίζεται με κόκκινο στο Σχ. 11

Φωνήγεν “ou”			
Mέση τιμή pitch	1o formant	2o formant	3o formant
173.17 Hz	337.27 Hz	1957.64 Hz	2686.25 Hz

— Για το φωνήγεν “ι”



Σχήμα 13: Χρονικό διάστημα φωνήγεντος “ι” για την πρόταση “one two three” από τον ομιλητή 8



Σχήμα 14: Μεγέθυνση στο χρονικό διάστημα του φωνήγεντος “ι” που απεικονίζεται με κόκκινο στο Σχήμα 13

Φωνήγεν “ι”			
Μέση τιμή pitch	1o formant	2o formant	3o formant
134.7 Hz	749.906 Hz	1283.253 Hz	2434.893 Hz

Σχολιασμός

Παρατηρούμε αρχικά ότι όλα τα φωνήσαντα χαρακτηρίζονται από **σχεδόν περιοδικές κυματομορφές**, κάτι το οποίο αναμέναμε. Επίσης, διαχρίνουμε μια αυξημένη τιμή τόσο στη μέση τιμή του pitch όσο και στην τιμή του 3ου formant στη γυναικεία φωνή (ομιλητής 8). Σχετικά με την διάκριση των φωνηέντων μεταξύ τους, από το μέσο pitch δεν μπορούμε να βγάλουμε κάποιο συμπέρασμα, καθώς οι τιμές των 3 φωνηέντων έχουν πολύ μικρή απόκλιση. Ωστόσο, τα formants μπορούν να συνδράμουν σε αυτή την κατεύθυνση. Συγκεκριμένα, παρατηρούμε μια αισθητά μεγαλύτερη τιμή του 1ου formant του “α”, σε σχέση με τα άλλα φωνήσαντα, της τάξης των 400 Hz. Τέλος, με βάση τις τιμές του 2ου formant του ομιλητή 1 φαίνεται εύχολη η διάκριση των φωνηέντων. Να σημειωθεί πως τα χρονικά σημεία έναρξης και λήξης των φωνηέντων “α”, “ου” και “ι” αποτελούν υποκειμενικές μας εκτιμήσεις, με βάση τα αντίστοιχα ακουστικά σήματα και ενδέχεται να μην είναι απόλυτα εύστοχες.

Βήμα 2: Υλοποίηση data parser

Υλοποιούμε μια συνάρτηση **data parser** που διαβάζει όλα τα αρχεία ήχου που δίνονται μέσα στο φάκελο digits και επιστρέφει 3 λίστες Python, που περιέχουν:

1. το wav που διαβάστηκε με librosa
2. τον αντίστοιχο ομιλητή
3. το ψηφίο.

Για την υπολοποίηση, αξιοποιούμε το γεγονός ότι τα ονόματα των αρχείων (π.χ. eight8.wav) υποδηλώνουν τόσο το ψηφίο που εκφωνείται (π.χ. eight), όσο και τον ομιλητή που το λέει (οι ομιλητές είναι αριθμημένοι από 1-15).

Υλοποίηση σε python

Η **data parser** δέχεται ως ορίσματα:

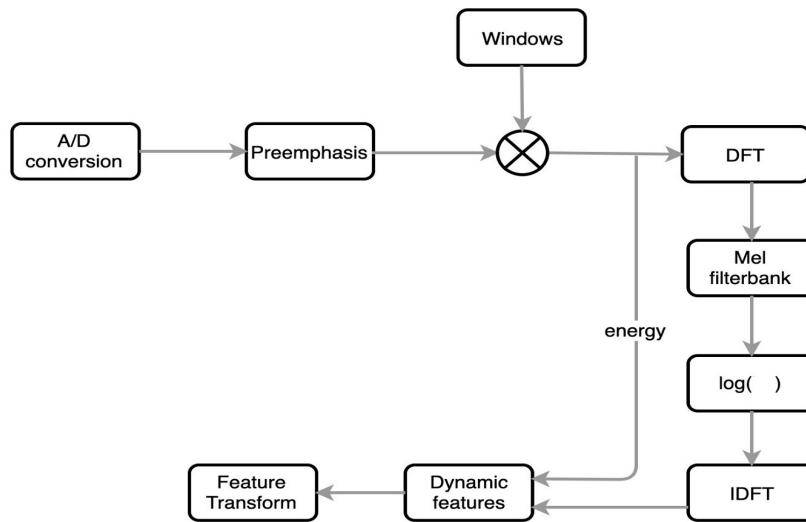
- το directory που βρίσκονται τα wav αρχεία (φάκελος digits)
- τη συχνότητα δειγματοληψίας (Fs)

Αρχικά, μέσω της **librosa.core.load**, η συνάρτηση διαβάζει τα αρχεία ήχου, υπό τη μορφή διανυσμάτων πραγματικών αριθμών (floating point vectors), και τα δειγματοληπτεί με την ίδια συχνότητα Fs. Κάθε διάνυσμα που προκύπτει γίνεται append στη λίστα wavs. Στη συνέχεια, εφαρμόζεται κατάλληλο split στο όνομα του εκάστοτε wav αρχείου, για τη διάκριση του filename (π.χ. eight8) από την κατάληξη (.wav). Ακολουθεί διαχωρισμός των γραμμάτων του filename (π.χ. eight) από το ψηφίο (π.χ. 8), έτσι ώστε να γίνεται αναγνώριση τόσο του ψηφίου που εκφωνείται όσο και του αντίστοιχου ομιλητή. Μέσω κατάλληλου λεξικού που έχουμε ορίσει, τα ψηφία μετατρέπονται από γράμματα σε αριθμούς (π.χ. το eight αντικαθίστανται από

το “8”). Τα ψηφία και οι αντίστοιχοι ομιλητές αποθηκεύονται στις λίστες digits και speakers αντίστοιχα. Η συνάρτηση αφότου ολοκληρώσει τη διαδικασία αυτή για όλα τα αρχεία, επιστρέφει τις λίστες wavs, digits και speakers.

Βήμα 3: Εξαγωγή Mel-Frequency Cepstral Coefficients (MFCCs)

Στο βήμα αυτό εξάγουμε τα **Mel-Frequency Cepstral Coefficients (MFCCs)** για κάθε αρχείο ήχου. Η διαδικασία που ακολουθείται για την εξαγωγή τους φαίνεται συνοπτικά στο ακόλουθο διάγραμμα:



Σχήμα 15: Στάδια εξαγωγής MFCCs από ακουστικά σήματα

Αρχικά, το σήμα φωνής διαιρείται σε έναν αριθμό από πλαίσια (frames). Έπειτα, σε κάθε ένα από αυτά εφαρμόζεται ένα παράθυρο (συνήθως χρησιμοποιείται το παράθυρο Hamming) με στόχο να ελαχιστοποιηθούν οι ασυνέχειες του σήματος στην αρχή και το τέλος του πλαισίου. Το επόμενο βήμα επεξεργασίας είναι ο Fast Fourier Transform (FFT) για την μεταφορά από το πεδίο του χρόνου στο πεδίο της συχνότητας και ο υπολογισμός του φάσματος. Επειδή σύμφωνα με το ακουστικό μοντέλο η ανθρώπινη αντίληψη για τις συχνότητες ενός σήματος δεν ακολουθεί γραμμική κλίμακα, χρησιμοποιούμε μια διάταξη τριγωνικών φίλτρων η οποία εφαρμόζεται στο πεδίο της συχνότητας και προσομοιώνει το υποκειμενικό φάσμα. Έτσι, για κάθε τόνο με πραγματική συχνότητα f σε Hz υπάρχει ένας υποκειμενικός τόνος στην κλίμακα mel. Για τον υπολογισμό σε mels μιας συχνότητας f χρησιμοποιείται η ακόλουθη προσεγγιστική σχέση:

$$mel(f) = 2595 \cdot \log_{10} \left(1 + \frac{f}{700} \right)$$

Για το φάσμα στην έξοδο του mel filter bank, φάσμα mel, υπολογίζουμε τον φυσικό λογάριθμό του (cepstrum). Οι συντελεστές MFCC's προκύπτουν λαμβάνοντας τον Inverse Discrete Fourier Transform (IDFT) του mel cepstrum και συγχεκριμένα τους πρώτους 12-13

συντελεστές. Σημειώνουμε πως επειδή ο λογάριθμος του power spectrum είναι πραγματικός και συμμετρικός, ο IDFT είναι **ισοδύναμος** με τον DCT.

Για την εξαγωγή των MFCCs από κάθε αρχείο ήχου χρησιμοποιούμε το **librosa**.

Συγκεκριμένα χρησιμοποιούμε την συνάρτηση **librosa.feature.mfcc**. Οι παράμετροι που ρυθμίζουμε είναι οι εξής:

1. **sr**: Είναι η συχνότητα δειγματοληψίας η οποία από τα δεδομένα της εκφώνησης δίνεται ίση με $F_s=16\text{kHz}$.
2. **n_mfcc**: Είναι ο αριθμός των χαρακτηριστικών (features) που θέλουμε να εξάγουμε. Εν προκειμένω, τίθεται ίσος με 13.
3. **n_fft**: Είναι το μήκος του παραθύρου και ισούται με τον ρυθμό δειγματοληψίας επί την διάρκειά του σε seconds. Συνήθεις τιμές για την διάρκεια του παραθύρου είναι τα 20-25 msec.
4. **hop_length**: Είναι το βήμα με το οποίο μετακινούνται τα κυλιόμενα παράθυρα κατά την παραθύρωση του σήματος. Τίθεται ίσο με 10msecs.

Στην ακόλουθη εικόνα δίνεται μία εποπτική ερμηνεία των δύο τελευταίων παραμέτρων. Με το γράμμα A συμβολίζεται η παράμετρος **n_fft** ενώ με το B η **hop_length**.

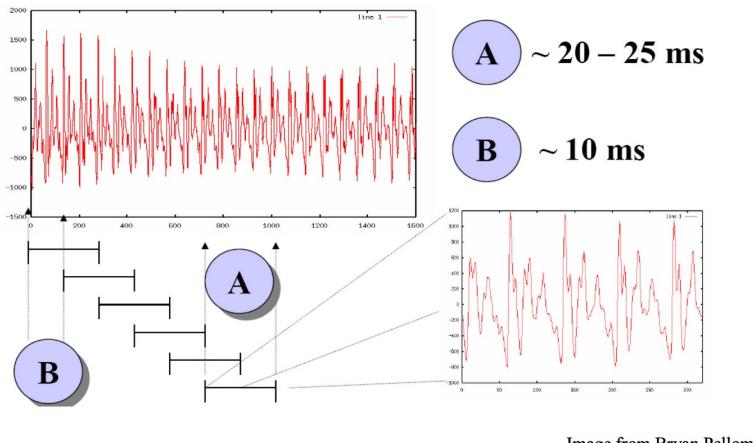


Image from Bryan Pellom

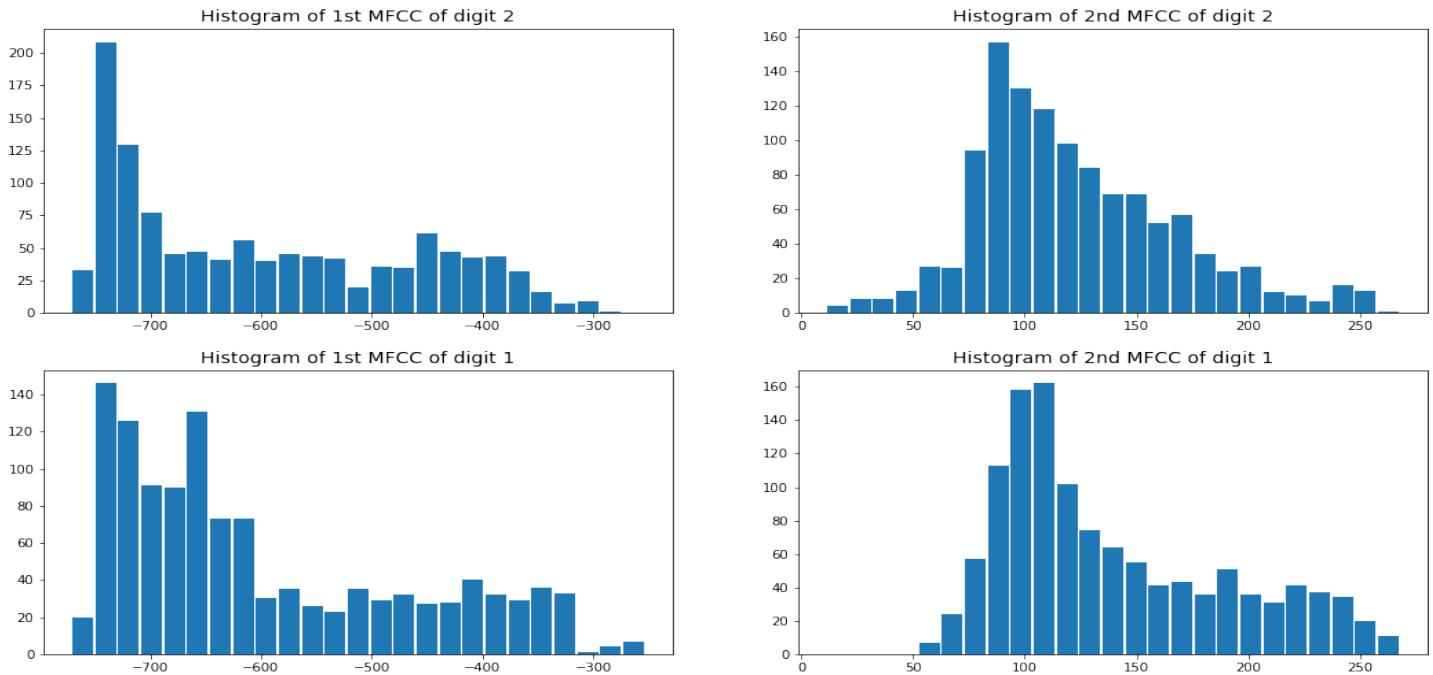
Σχήμα 16: Παραθύρωση ακουστικών σημάτων

Στη συνέχεια, με χρήση της συνάρτησης **librosa.feature.delta** υπολογίζουμε την πρώτη και δεύτερη (θέτουμε $order=2$) παράγωγο των χαρακτηριστικών που εξαγάγαμε προηγουμένως.

Για κάθε wav αρχείο προσθέτουμε τα MFCCs του και τα deltas, deltas-deltas στις αντίστοιχες λίστες **MFCCs**, **deltas** και **deltas_deltas**.

Βήμα 4: Ιστογράμματα 1ου και 2ου MFCC των ψηφίων n1 και n2

Θέλουμε τώρα να αναπαραστήσουμε τα ιστογράμματα του 1ου και του 2ου MFCC των ψηφίων $n1 = 2$ και $n2 = 1$ για όλες τις εκφωνήσεις. Για το σκοπό αυτό, με χρήση της συνάρτησης **concatenate** της βιβλιοθήκης numpy, δημιουργούμε τέσσερα διανύσματα στα οποία αποθηκεύουμε την 1η και 2η στήλη αντίστοιχα του πίνακα MFCC για τα wav των ψηφίων $n1$ και $n2$. Κάθε ένα δηλαδή από τα τέσσερα διανύσματα που δημιουργούμε, περιέχει όλες τις τιμές ενός συγκεκριμένου MFCC (1ου ή 2ου) και ενός δεδομένου ψηφίου ($n1$ ή $n2$). Στη συνέχεια, με χρήση της συνάρτησης **hist** της matplotlib δημιουργούμε τα ζητούμενα ιστογράμματα. Για λόγους σύγκρισης, απεικονίζουμε τα ιστογράμματα κάθε MFCC στο ίδιο εύρος (και για τα 2 ψηφία), δηλαδή ορίζουμε κοινό αριθμό από bins. Αυτά φαίνονται στην ακόλουθη εικόνα:



Σχήμα 17: Ιστογράμματα 1ου και 2ου MFCC των ψηφίων $n1 = 2$ και $n2 = 1$

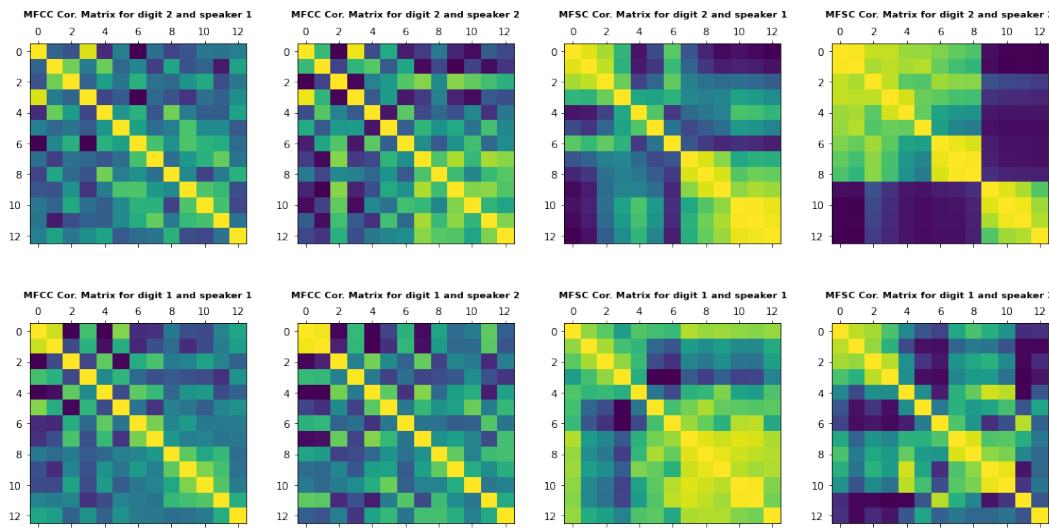
Παρατηρούμε και για τα 2 ψηφία μια παρόμοια συμπεριφορά ως προς το 1o και το 2o MFCC. Συγκεκριμένα, είναι εμφανής η κοινή τάση των τιμών του 1ou MFCC να συσσωρεύονται κάτω από το -700 και των τιμών του 2ou να κυμαίνονται γύρω από το 100.

Για τον προσδιορισμό της απόκλισης ανάμεσα στο πρώτο και το δεύτερο MFCC χρησιμοποιούμε την συνάρτηση **mutual_info_score** του scikit-learn. Πρακτικά πρόκειται για μια προσέγγιση του αλγορίθμου για Kullback-Leibler divergence. Για την απόκλιση ανάμεσα στο 1o MFCC των δύο ψηφίων λαμβάνουμε τιμή **2.831**. Η αντίστοιχη τιμή ανάμεσα στο 2o MFCC είναι ίση με **2.72**. Βλέπουμε επομένως, ότι υπάρχει απόκλιση μεταξύ των ιστογραμμάτων των διαφορετικών ψηφίων, η οποία μπορεί να αξιοποιηθεί σε εφαρμογές κατηγοριοποίησης, όπως spoken digit recognition.

Εξάγουμε τώρα για 2 εκφωνήσεις των n1 και n2, από 2 διαφορετικούς ομιλητές, τα **Mel Filterbank Spectral Coefficients (MFSCs)**, δηλαδή τα χαρακτηριστικά που εξάγονται αφού εφαρμοστεί η συστοιχία φίλτρων της ακίμακας Mel πάνω στο φάσμα του σήματος φωνής αλλά χωρίς να εφαρμοστεί στο τέλος ο μετασχηματισμός DCT (εξάγουμε και πάλι χαρακτηριστικά διάστασης 13). Η επιλογή των εκφωνήσεων γίνεται τυχαία με χρήση της συνάρτησης **random.choice** της numpy. Για την εξαγωγή των MFSCs από κάθε αρχείο ήχου χρησιμοποιούμε το **librosa**, αντίστοιχα με το Βήμα 3. Εδώ απλώς χρησιμοποιούμε την συνάρτηση **librosa.feature.melspectrogram**. Οι παράμετροι που ρυθμίζουμε είναι οι εξής:

1. **sr**: Είναι η συχνότητα δειγματοληψίας η οποία και εδώ είναι ίση με $F_s=16\text{kHz}$.
2. **n_mfcc**: Είναι ο αριθμός των features που θέλουμε να εξάγουμε (13).
3. **n_fft**: Είναι το μήκος του παραθύρου και ισούται με τον ρυθμό δειγματοληψίας επί την διάρκειά του σε seconds.
4. **hop_length**: Είναι το βήμα με το οποίο μετακινούνται τα κυλιόμενα παράθυρα κατά την παραθύρωση του σήματος. Τίθεται ίσο με 10 ms.

Τέλος, αναπαριστούμε γραφικά τη συσχέτιση τόσο των MFSCs όσο και των MFCCs για την κάθε εκφώνηση. Για το σκοπό αυτό μετατρέπουμε όλα τα χαρακτηριστικά από numpy arrays σε pandas Dataframes με χρήση της συνάρτησης **pd.DataFrame**. Αυτό μας επιτρέπει να εξάγουμε πίνακες συσχέτισης (Correlation Matrixes) με χρήση της συνάρτησης **pandas.DataFrame.corr**. Οι πίνακες αυτοί απεικονίζεται στην ακόλουθη εικόνα:



Σχήμα 18: Πίνακες Συσχέτισης των MFSCs και των MFCCs για την κάθε εκφώνηση

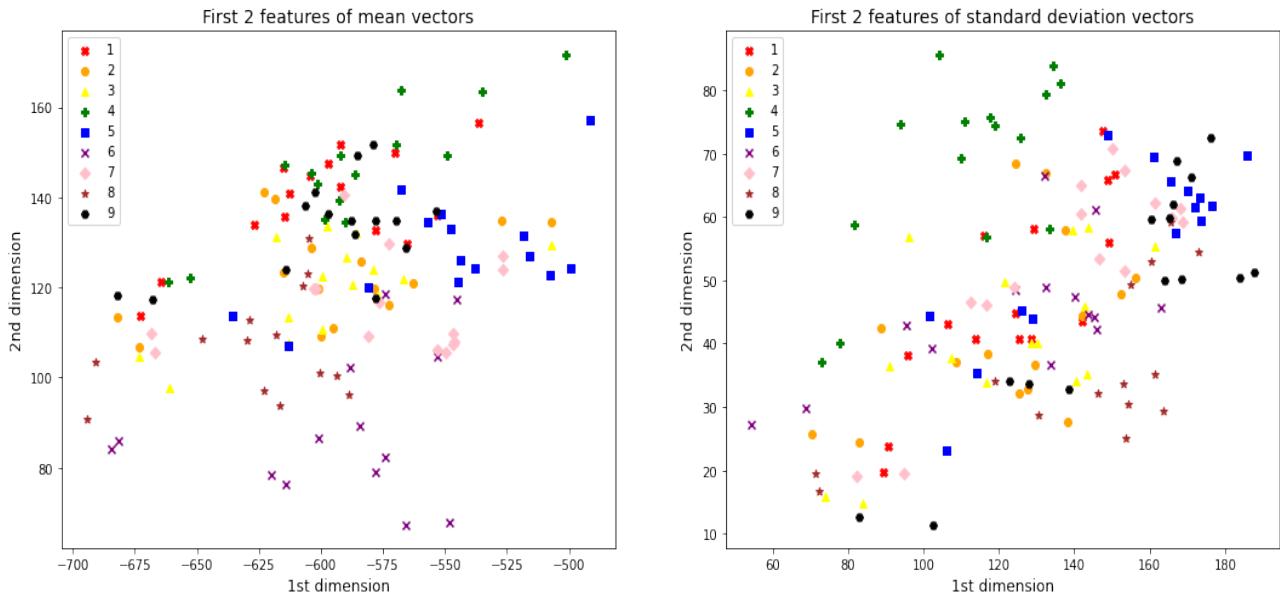
Παρατηρούμε πως **τα MFCCs εμφανίζουν σημαντικά μικρότερη συσχέτιση** από ότι τα MFSCs. Αυτό οφείλεται στην εφαρμογή του μετασχηματισμού DCT, ο οποίος αποσυσχετίζει τα χαρακτηριστικά. Για τον λόγο αυτό, στην ανάλυση και αναγνώριση φωνής χρησιμοποιούνται ευρύτερα τα MFCCs από τα MFSCs. Τα δεδομένα αυτά αναπαριστούν περισσότερη πληροφορία και είναι πιο εύκολα μοντελοποιήσιμα.

Βήμα 5: Εξαγωγή ενός μοναδικού διανύσματος για κάθε εκφώνηση

Μια πρώτη προσέγγιση για την αναγνώριση των ψηφίων είναι η εξαγωγή ενός μοναδικού διανύσματος χαρακτηριστικών για κάθε εκφώνηση.

Για τον σκοπό αυτό ενώνουμε για κάθε αρχείο, μέσω της **np.concatenate**, τα αντίστοιχα MFCCs, deltas και deltas-deltas του Βήματος 3, σε έναν ενιαίο πίνακα **wav_features** διαστάσεων (frames, 39), ο οποίος γίνεται κάθε φορά append στην λίστα **features**. Για την μετατροπή του πίνακα σε διάνυσμα λαμβάνουμε αρχικά με χρήση των συναρτήσεων **np.mean** και **np.std** τα διανύσματα των μέσων τιμών (**wav_means**) και των τυπικών αποκλίσεων (**wav_stds**), διαστάσεων 39 το καθένα. Το τελικό διάνυσμα χαρακτηριστικών για μία εκφώνηση είναι διαστάσεων 78 και προκύπτει από την συνένωση των δύο προηγούμενων διανυσμάτων (πρώτα το **wav_means** και μετά το **wav_stds**). Στην συνέχεια, το αποθηκεύουμε στην αντίστοιχη γραμμή του πίνακα **means_stds** τον οποίο έχουμε αρχικοποιήσει ως έναν δισδιάστατο πίνακα μεγέθους (133, 78), όπου 133 είναι ο συνολικός αριθμός των ηχογραφήσεων.

Αναπαριστούμε τώρα με scatter plot τις 2 πρώτες διαστάσεις των διανυσμάτων των μέσων τιμών και τυπικών αποκλίσεων χρησιμοποιώντας διαφορετικό χρώμα και σύμβολο για κάθε ψηφίο. Από τον τρόπο κατασκευής του πίνακα **means_stds**, οι 2 πρώτες συνιστώσες του διανύσματος των μέσων τιμών για ένα συγκεκριμένο .wav αρχείο βρίσκονται στις στήλες 0 και 1. Για το διάνυσμα των τυπικών αποκλίσεων οι αντίστοιχες στήλες είναι οι 39 και 40 (αφού η αρίθμηση ξεκινάει από το μηδέν).



Σχήμα 19: Αναπαράσταση των δύο πρώτων διαστάσεων των διανυσμάτων μέσων τιμών και τυπικών αποκλίσεων

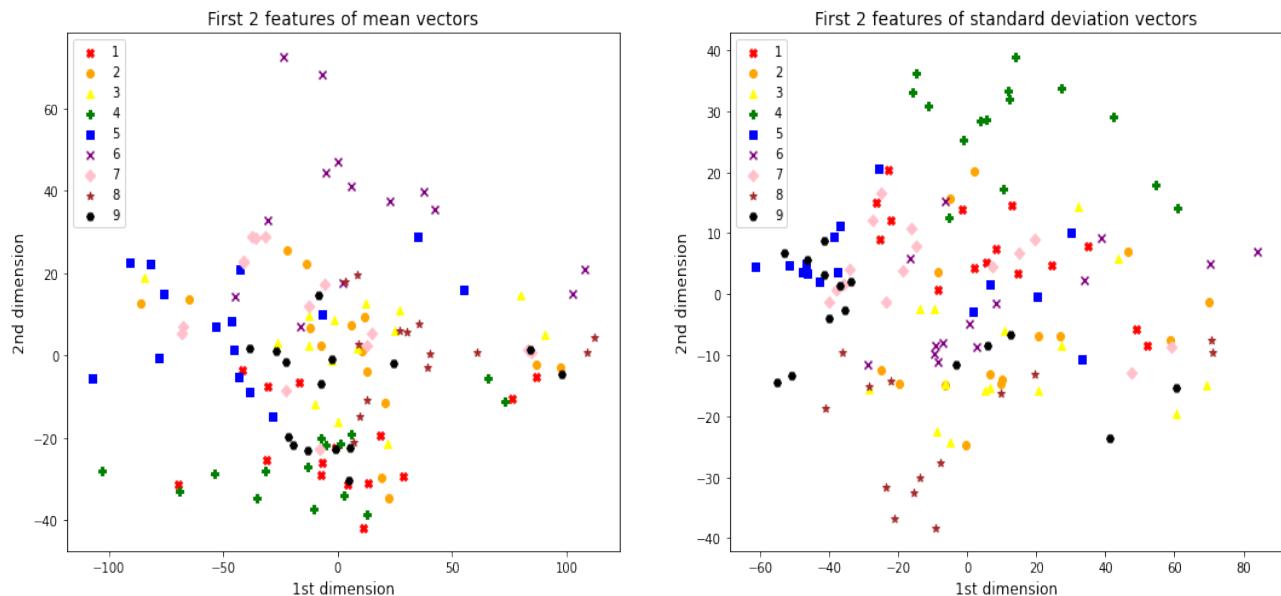
Από τα παραπάνω διαγράμματα δεν μπορούμε να εξάγουμε κάποια ποιοτικά συμπεράσματα καθώς δεν διακρίνουμε κάποια αυστηρή ομαδοποίηση των δύο χαρακτηριστικών ανάλογα με το ψηφίο. Αντιθέτως, υπάρχει σημαντική επικάλυψη μεταξύ των 10 κατηγοριών.

Βήμα 6: Μείωση διαστατικότητας διανυσμάτων με εφαρμογή PCA

Μια καλή τακτική για απεικόνιση πολυδιάστατων διανυσμάτων είναι η μείωση των διαστάσεών τους με **Principal Component Analysis (PCA)**.

Μειώνουμε σε 2 τις διαστάσεις των διανυσμάτων του προηγούμενου βήματος με PCA και δημιουργούμε εκ νέου το scatter plot. Χρησιμοποιούμε την έτοιμη υλοποίηση της `sklearn` ύστοτος το `n_components` ίσο με 2. Τα μετασχηματισμένα διανύσματα, `means_reduced` και `stds_reduced`, προκύπτουν εφαρμόζοντας τη μέθοδο `fit_transform` στα αρχικά `means` και `stds`. Για ένα δεδομένο ψηφίο η πρώτη και η δεύτερη διάσταση των διανυσμάτων των μέσων τιμών και των τυπικών αποκλίσεων είναι η πρώτη και η δεύτερη στήλη των `means_reduced` και `stds_reduced` αντίστοιχα.

Επαναλαμβάνουμε τώρα την απεικόνιση στο δισδιάστατο χώρο όπως στο Βήμα 5.

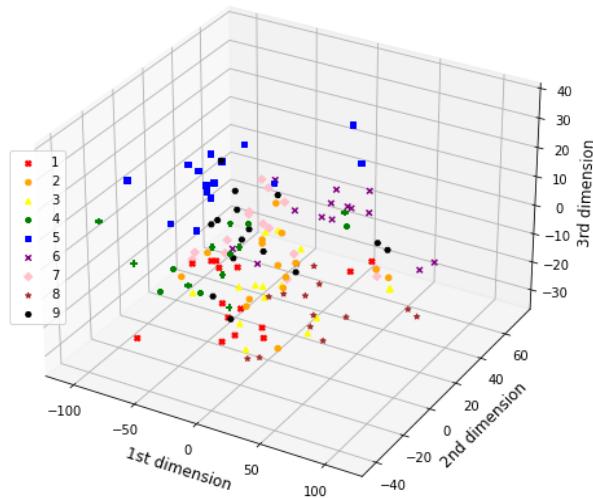


Σχήμα 20: Αναπαράσταση των δύο διαστάσεων των μετασχηματισμένων με PCA διανυσμάτων

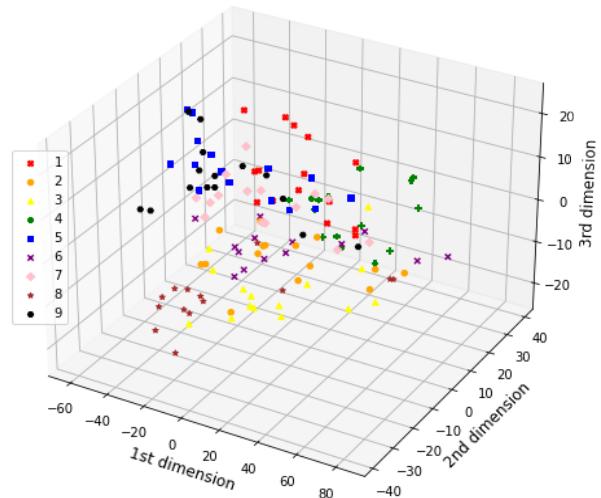
Παρατηρούμε ότι τα αποτελέσματα που λαμβάνουμε δεν είναι πάλι ποιοτικά. Μπορούμε να διωχρίνουμε ίσως κάποιες κατανομές, όπως π.χ. το ψηφίο 4 ή 5, ωστόσο δεν φαίνεται να πετύχαμε κάποια αισθητή βελτίωση από την μείωση της διαστατικότητας με PCA.

Δοκιμάζουμε τώρα να προσθέσουμε μία ακόμα διάσταση ύστοτας την παράμετρο `n_components` ίση με 3. Τα νέα διαγράμματα φαίνονται στις ακόλουθες εικόνες:

First 3 features of mean vectors



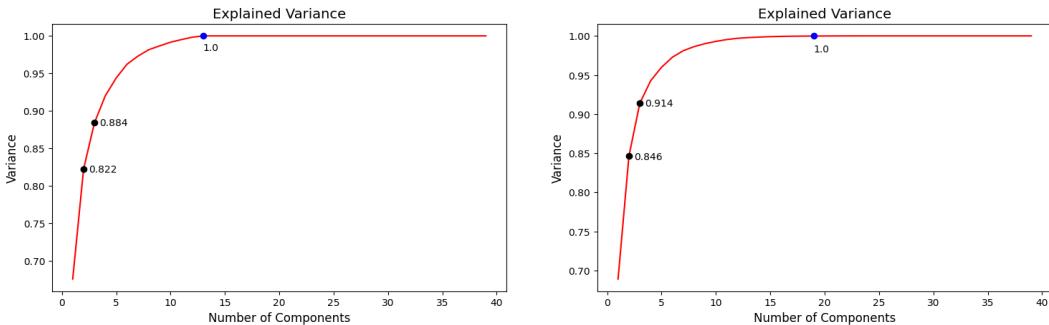
First 3 features of standard deviation vectors

**Σχήμα 21:** Τρισδιάστατη απεικόνιση των τριών διαστάσεων των μετασχηματισμένων με PCA διανυσμάτων

Σε σχέση με τις δύο διαστάσεις μπορούμε τώρα να διαχρίνουμε περισσότερες κατανομές, όπως π.χ. του ψηφίου 8 ή 9. Το γεγονός αυτό είναι αναμενόμενο, καθώς με βάση την “κατάρα της διαστατικότητας” (curse of dimensionality), όσο αυξάνεται ο αριθμός των διαστάσεων τόσο τα δεδομένα γίνονται πιο ευκόλως διαχωρίσιμα και τόσες περισσότερες ομάδες από στοιχεία της ίδιας κατανομής αρχίζουν να εμφανίζονται.

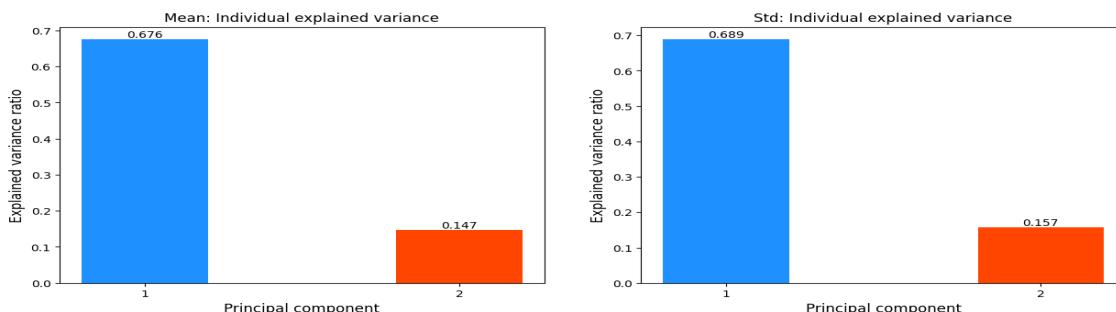
Ένας σημαντικός παράγοντας που πρέπει να λαμβάνουμε υπόψη κατά την εφαρμογή της μεθόδου PCA είναι η ικανότητα εκτίμησης του αριθμού των συνιστωσών (components) που απαιτούνται για την περιγραφή των δεδομένων. Αυτό μπορεί να προσδιοριστεί εξετάζοντας το explained variance ratio, το οποίο αποτελεί μία μετρική για την αξιολόγηση της χρησιμότητας ενός principal component. Το explained variance ratio υπάρχει σαν παράμετρος (`.explained_variance_ratio_`) στην υλοποίηση της PCA της `sklearn` και εκφράζει το ποσοστό της αρχικής διασποράς που αποδίδεται από κάθε μία από τις επιλεγμένες συνιστώσες. Ιδιαίτερα, θα προσθέταμε components μέχρις ότου το αυθροιστικό (cumulative) explained variance ratio να φτάσει περίπου 0.8 ή 80% για να αποφύγουμε το overfitting.

Για τον σκοπό αυτό σχεδιάζουμε το cumulative explained variance ratio ως συνάρτηση του αριθμού των συνιστωσών. Στο ίδιο διάγραμμα επισημειώνουμε τις τιμές που προκύπτουν για components ίσα με 2 και 3 καθώς επίσης και την πρώτη συνιστώσα για την οποία η συνεισφορά της έχει σαν αποτέλεσμα το cumulative explained variance ratio να γίνει ίσο με 1.

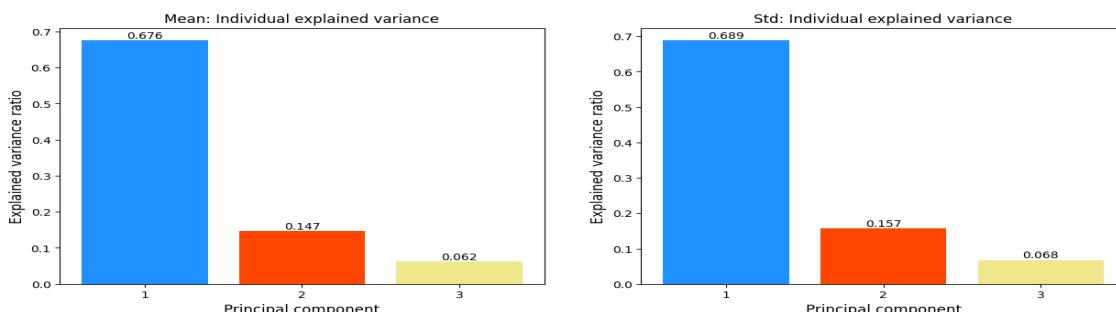


Σχήμα 22: Απεικόνιση του cumulative explained variance συναρτήσει του αριθμού των συνιστώσων

Από τα παραπάνω διαγράμματα παρατηρούμε πως στην περίπτωση εφαρμογής της PCA στο διάνυσμα των μέσων τιμών, το αριθμητικό ποσοστό της αρχικής διασποράς που διατηρείται με 2 συνιστώσες είναι ίσο με 82.2% ενώ με 3 είναι 88.4%. Για το διάνυσμα των τυπικών αποκλίσεων τα αντίστοιχα ποσοστά είναι ίσα με 84.6% και 91.4%. Παρατηρούμε επομένως, πως και στις δύο περιπτώσεις, η PCA καταφέρνει να μειώσει τις διαστάσεις, διατηρώντας πάνω από το 80% της αρχικής διασποράς των δεδομένων. Ωστόσο, φαίνεται πως η μέθοδος αυτή εξακολουθεί να μην είναι ιδιαίτερα αποτελεσματική στο πρόβλημα της αναγνώρισης φωνής. Τέλος, για την απεικόνιση των αντίστοιχων ποσοστών μεμονωμένα για κάθε συνιστώσα (και όχι αριθμητικά όπως πριν) ορίζουμε την συνάρτηση `pca` - η οποία δέχεται για ορίσματα τα διανύσματα των μέσων τιμών και των τυπικών αποκλίσεων καθώς επίσης και τα `n_components`. Σχεδιάζει δύο bar plots (ένα για κάθε διάνυσμα) με τις συνιστώσες (από 1 μέχρι και `n_components`) και το αντίστοιχο ποσοστό της αρχικής διασποράς που διατηρεί κάθε μία από αυτές. Προφανώς, τα ποσοστά αυτά μειώνονται καθώς αυξάνεται ο αριθμός των συνιστώσων.



Σχήμα 23: Bar plots απεικόνισης explained variance ratio για κάθε μία από τις 2 συνιστώσες



Σχήμα 24: Bar plots απεικόνισης explained variance ratio για κάθε μία από τις 3 συνιστώσες

Βήμα 7: Ταξινόμηση με χρήση Naive Bayes, kNN και SVM

Χωρίζουμε τα δεδομένα σε train-test με αναλογία 70% - 30%. Αυτό γίνεται με χρήση της συνάρτησης `train_test_split` της sklearn, δίνοντας για όρισμα τον πίνακα `means_stds` διαστάσεων (133,78), όπου 133 είναι ο αριθμός των δειγμάτων/ηχογραφήσεων και 78 είναι η διάσταση των χαρακτηριστικών. Προτού προχωρήσουμε στην ταξινόμηση, κανονικοποιούμε τα δεδομένα με τον **Standard Scaler**. Αυτός αφαιρεί από τα δεδομένα τη μέση τιμή όλων και διαιρεί με την διασπορά τους. Στη συνέχεια, χρησιμοποιούμε τους ακόλουθους ταξινομητές από την πρώτη εργαστηριακή άσκηση:

- Custom Naive Bayes
- Sklearn Naive Bayes
- KNeighbors
- Linear SVM
- Polynomial SVM
- Rbf SVM

Παρουσιάζουμε συγκεντρωτικά τα ποσοστά επιτυχίας για τους παραπάνω ταξινομητές στον ακόλουθο πίνακα.

Classifier	Accuracy
Custom Naive Bayes	75.0 %
Sklearn Naive Bayes	75.0 %
KNeighbors (n=1)	80.0 %
KNeighbors (n=3)	67.5 %
KNeighbors (n=5)	60 %
Linear SVM	92.5 %
Polynomial SVM	52.5 %
Rbf SVM	80.0 %

Παρατηρούμε αρχικά ότι το ποσοστό επιτυχίας του custom Naive Bayes συμπίπτει με εκείνο της sklearn, γεγονός που πιστοποιεί την ορθότητα της υλοποίησής μας. Αναφορικά με τον ταξινομητή Knn, το μεγαλύτερο ποσοστό προκύπτει για αριθμό γειτόνων ίσο με 1. Σημαντική μείωση του accuracy παρατηρείται για n_neighbors ίσο με 3 και 5. Τέλος, μεταξύ των SVM ταξινομητών ο Linear παρουσιάζει την καλύτερη επίδοση, πετυχαίνοντας μάλιστα το μεγαλύτερο ποσοστό από όλους τους ταξινομητές που εξετάσαμε (**92.5%**). Ωστόσο, όπως φαίνεται και από τον παραπάνω πίνακα τα ποσοστά (με εξαίρεση αυτό του Linear SVM Classifier) παραμένουν στην πλειοψηφία των περιπτώσεων αρκετά χαμηλά. Αυτό ενδεχομένως να οφείλεται στον μικρό αριθμό των χαρακτηριστικών που λαμβάνουμε υπόψην για κάθε δείγμα.

BONUS

Για να εξετάσουμε την επίδραση των χαρακτηριστικών προσθέτουμε για κάθε δείγμα στο υπάρχον διάνυσμα χαρακτηριστικών τα **zero-crossing rates** με τα αντίστοιχα deltas και deltas_deltas. Για τον υπολογισμό των πρώτων χρησιμοποιούμε την συνάρτηση **librosa.feature.zero_crossing_rate**, θέτοντας κατάλληλα τις παραμέτρους **frame_length** και **hop_length**. Τα zero-crossing-rates τα οποία επιστρέφονται από την librosa είναι πίνακες της μορφής (1,t), όπου t ο (μεταβλητός) αριθμός των frames για κάθε .wav αρχείο. Προκειμένου κάθε διάνυσμα να είναι ίδιου μεγέθους, από τις t στήλες των πινάκων επιλέγουμε κάθε φορά τις πρώτες 13. Το νέο διάνυσμα για κάθε αρχείο προκύπτει από την συνένωση των zero_crossing_rates με τα deltas και deltas_deltas και αποθηκεύεται ως ξεχωριστή γραμμή στον πίνακα **zcr_features** διαστάσεων (133,39). Ο τελικός πίνακας **augmented_features** προκύπτει από το concatenation των επιμέρους means_stds του βήματος 5 και του zcr_features.

Στον ακόλουθο πίνακα κατ' αναλογία με πριν παρουσιάζουμε συγχεντρωτικά τα ποσοστά επιτυχίας για τους ταξινομητές που μελετάμε.

Classifier	Accuracy
Custom Naive Bayes	67.5 %
Sklearn Naive Bayes	67.5 %
KNeighbors (n=1)	77.5 %
KNeighbors (n=3)	57.5 %
KNeighbors (n=5)	45.0 %
Linear SVM	85.0 %
Polynomial SVM	35.0 %
Rbf SVM	75.0 %

Παρατηρούμε πως με την προσθήκη των επιπλέον χαρακτηριστικών δεν βελτιώθηκε η επίδοση κανενός ταξινομητή. Αντιθέτως, διαπιστώνουμε μία άλλοτε μικρή και άλλοτε μεγάλη πτώση των ποσοστών σε σχέση με αυτά που είχαμε προηγουμένως. Το γεγονός αυτό ενδεχομένως να οφείλεται στο ότι τα συγκεκριμένα ηχητικά χαρακτηριστικά που επιλέξαμε να προσθέσουμε στα διανύσματα δεν προσφέρουν τελικώς κάποια χρήσιμη πληροφορία η οποία θα βοηθούσε στην καλύτερη αναγνώριση των ψηφίων. Τέλος, η στάσιμη και χαμηλή επίδοση των ταξινομητών μπορεί να οφείλεται και στο μικρό μέγεθος του dataset μας καθώς αποτελείται από μόνο 133 διαφορετικά αρχεία / δείγματα.

Βήμα 8: Πρόβλεψη συνημιτόνου με δεδομένη την ακολουθία ημιτόνου

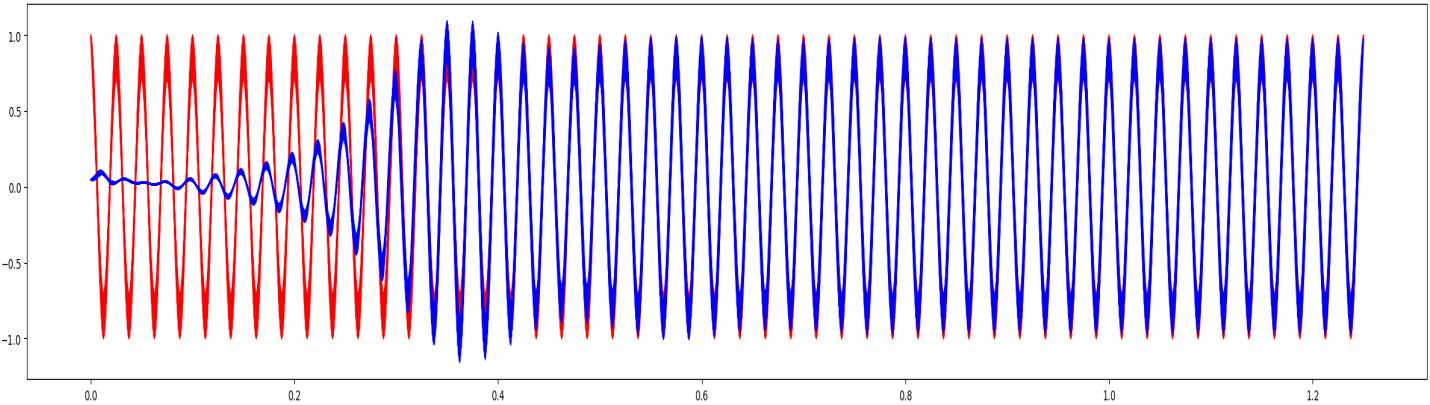
Στο βήμα αυτό δημιουργούμε ακολουθίες 10 σημείων ενός ημιτόνου και ενός συνημιτόνου με συχνότητα $f = 40 \text{ Hz}$. Σκοπός είναι η πρόβλεψη του συνημιτόνου με δεδομένη την ακολουθία του ημιτόνου. Επιλέγουμε να κατασκευάσουμε συνολικά 200 ακολουθίες ημιτόνου, για τις οποίες ψευδορούμε σταθερή απόσταση $\frac{T/4}{10} = 6.25 \cdot 10^{-4}$ ανάμεσα στα διαδοχικά σημεία.

Ορίζουμε λοιπόν ένα Αναδρομικό Νευρωνικό Δίκτυο (Recurrent Neural Network–RNN), το οποίο δέχεται ως input τις ακολουθίες του ημιτόνου και ως ετικέτες για τις εισόδους (labels) τις αντίστοιχες (δηλαδή για το ίδιο χρονικό διάστημα) ακολουθίες του συνημιτόνου.

Βασιζόμενοι σε αυτή την υλοποίηση και κάνοντας διάφορες τροποποιήσεις, ορίζουμε και εκπαιδεύουμε το μοντέλο για τις ακόλουθες τιμές παραμέτρων:

- `input_size=1`
- `hidden_size=32`
- `num_layers=3`
- `batch_first=True`

Πειραματίζομαστε με χρήση απλού RNN και στη συνέχεια με μονάδες LSTM και GRU, με την τελευταία να μας δίνει τελικά την καλύτερη ποιοτική προσέγγιση, η οποία και φαίνεται στην ακόλουθη εικόνα:



Σχήμα 25: Πρόβλεψη ακολουθίας συνημιτόνου με δεδομένη την ακολουθία ημιτόνου με χρήση GRU

Παρατηρούμε πως το μοντέλο σταδιακά προσομοιώνει την **συνάρτηση μοναδιαίου συνημιτόνου** (η οποία απεικονίζεται με **χόκχινο χρώμα**) και διατηρεί μια πολύ καλή **πρόβλεψη** (η οποία απεικονίζεται με **μπλε χρώμα**) μετά από ένα συγκεκριμένο αριθμό εποχών.

Γενικά τα μοντέλα LSTM και GRU είναι αρκετά πιο διαδεδομένα από ότι τα απλά Recurrent Neural Networks. Αυτό οφείλεται στο γεγονός ότι τα απλά RNNs εμφανίζουν το πρόβλημα της πρόσφατης μνήμης (**short-term memory**). Αυτό σημαίνει πως παρουσιάζουν αδυναμία

μεταφοράς πληροφορίας για αρκετά μεγάλες ακολουθίες εισόδου. Ωστόσο, σε πολλές εφαρμογές, όπως στην επεξεργασία φυσικής γλώσσας (NLP), είναι συχνά επιθυμητό να γίνεται capture ένα αρκετά ευρύ context γύρω από το σημείο που εξετάζουμε. Για παράδειγμα, αν επεξεργαζόμαστε μια παράγραφο ενός κειμένου με στόχο να κάνουμε προβλέψεις λέξεων, τότε τα RNNs ενδέχεται να μη συγχρατήσουν τμήμα χρήσιμης πληροφορίας που βρίσκεται στην αρχή της παραγράφου.

Κατά τη διάρκεια του back-propagation, τα Αναδρομικά Νευρωνικά Δίκτυα, πάσχουν από το πρόβλημα του **vanishing gradient**. Οι παράγωγοι σχετίζονται άμεσα με την ανανέωση των βαρών του δικτύου. Το πρόβλημα του vanishing gradient εμφανίζεται όταν οι τιμές των παραγώγων μικραίνουν υπερβολικά καθώς διαδίδονται στο χρόνο, με αποτέλεσμα τα βάρη του μοντέλου πρακτικά να μη μεταβάλλονται (ελάχιστη μεταβολή).

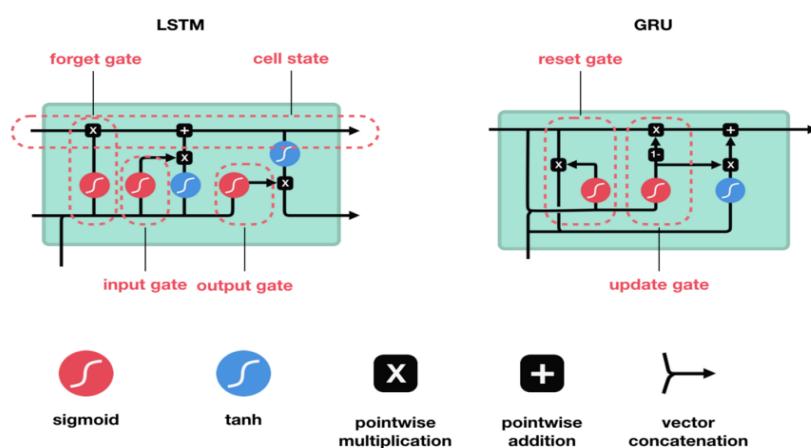
$$\text{new weight} = \text{weight} - \text{learning rate} * \text{gradient}$$

2.0999	= 2.1	-	0.001
Not much of a difference			update value

Σχήμα 26: Παράδειγμα προβλήματος vanishing gradient κατά το οποίο τα βάρη μεταβάλλονται ελάχιστα

Έτσι λοιπόν, στα απλά RNNs, τα επίπεδα που λαμβάνουν πολύ μικρές τιμές παραγώγων παύουν να εκπαιδεύονται. Αυτά είναι συνήθως τα προγενέστερα layers. Επειδή λοιπόν ορισμένα επίπεδα δεν μαθαίνουν κατά το training, τα δίκτυα ζεχνούν πληροφορία που βρίσκεται στο “μακρινό παρελθόν”, για αυτό και χαρακτηρίζονται, όπως αναφέραμε από μικρή μνήμη (short-term memory).

Τα **LSTMs** και τα **GRUs** δημιουργήθηκαν με στόχο την επίλυση του συγκεκριμένου προβλήματος. Διαιθέτουν εσωτερικούς μηχανισμούς που ονομάζονται πύλες (gates), οι οποίες επιτρέπουν την ρύθμιση της ροής πληροφορίας.



Σχήμα 27: Εσωτερική δομή μοντέλων LSTM και GRU

Οι πύλες αυτές μπορούν να μάθουν ποια δεδομένα σε μια ακολουθία είναι σημαντικό να διατηρηθούν και ποια όχι. Με τον τρόπο αυτό, είναι εφικτό να μεταφερθεί σχετική πληροφορία κατά μήκος της μεγάλης αλυσίδας των ακολουθιών για την παραγωγή προβλέψεων. Σχεδόν όλα τα σύγχρονα αποτελέσματα που βασίζονται σε Αναδρομικά Νευρωνικά Δίκτυα επιτυγχάνονται με χρήση των δύο αυτών μοντέλων, τα οποία εμφανίζουν πολύ μικρές διαφορές. Όπως φαίνεται και στο Σχήμα 17, το GRU συγχωνεύει το cell state και το hidden state για την μετάδοση της πληροφορίας. Επίσης, διαθέτει μόνο 2 πύλες (σε αντίθεση με το LSTM που έχει 3), την reset gate και την update gate. Η τελευταία προκύπτει με συνδυασμό των πυλών forget gate και input gate του κλασικού LSTM.

Βήμα 9: Χωρισμός train δεδομένων σε training και validation set

Στο βήμα αυτό χρησιμοποιούμε αρχικά την δοθείσα συνάρτηση **parser**, η οποία δέχεται ως όρισμα το directory του dataset. Διαβάζει τα wav αρχεία και υπολογίζει τα MFCCs σε κάθε παράθυρο των ακουστικών σημάτων. Επιστρέφει τις λίστες **X_train**, **X_test**, **y_train** και **y_test**. Κάθε στοιχείο των δύο πρώτων είναι ένας πίνακας μεταβλητού αριθμού γραμμών και 6 στηλών με τα MFCCs σε κάθε πλαίσιο (frame) των δειγμάτων του train και test set αντίστοιχα. Αφότου ορίσουμε τις παραπάνω λίστες, τις μετατρέπουμε σε numpy arrays με χρήση της εντολής **np.array**. Στη συνέχεια, χωρίζουμε τα train δεδομένα σε training και validation set με ποσοστό 80% - 20% και με τέτοιο τρόπο ώστε να διατηρηθεί ίδιος ο αριθμός των διαφορετικών ψηφίων σε κάθε set (stratified split). Για τον σκοπό αυτό, αξιοποιούμε την συνάρτηση **train_test_split** δίνοντας στην παράμετρό της **stratify** τον πίνακα **y_train** με τα labels των δεδομένων εκπαίδευσης.

Βήμα 10: Αναγνώριση ψηφίων με GMM-HMM

Εξετάζουμε τώρα την εφαρμογή των **Gaussian Mixture Models - Hidden Markov Models** ή αλλιώς **GMM - HMM** στο πρόβλημα της αναγνώρισης ψηφίων. Για το βήμα αυτό καθώς και για τα επόμενα χρησιμοποιούμε τη βιβλιοθήκη **pomegranate** της Python.

Αρχικοποιούμε ένα GMM-HMM μοντέλο **για κάθε ψηφί**. Το μοντέλο είναι της μορφής left-right. Συγκεκριμένα, αν $A = \{a_{ij}\}$ είναι ο πίνακας μεταβάσεων του μοντέλου, τότε $a_{ij} = 0$ για $j < i$, ενώ οι αρχικές πιθανότητες των καταστάσεων είναι:

$$\pi_i = \begin{cases} 0 & i \neq 1 \\ 1 & i = 1 \end{cases}$$

Επιπλέον επιτρέπονται μεταβάσεις μόνο μεταξύ διαδοχικών καταστάσεων, δηλαδή υπάρχει ο περιορισμός ότι $a_{ij} = 0$ για $j > i + 1$.

Για την υλοποίηση όλων των παραπάνω ορίζουμε τη συνάρτηση **hmm** η οποία δέχεται για ορίσματα τα δεδομένα από τον πίνακα **X_train** που αντιστοιχούν σε ένα εκάστοτε ψηφίο, τον αριθμό των καταστάσεων **n_states** και τον αριθμό των γκαουσιανών **n_mixtures** για το GMM. Ορίζει τους πίνακες **starts** και **ends** οι οποίοι δηλώνουν την πιθανότητα μία κατάσταση να είναι αρχική ή τελική. Η πιθανότητα μιας κατάστασης να είναι αρχική είναι 1 για την πρώτη και 0 για οποιαδήποτε άλλη. Αντίστοιχα, η πιθανότητα μιας κατάστασης να είναι τελική είναι 1 για την τελευταία και 0 για τις υπόλοιπες. Επιπλέον, ορίζει τον πίνακα μετάβασης **trans_mat**, ο οποίος είναι μηδενικός εκτός της κύριας και της ακριβώς από πάνω διαγωνίου, αφού ισχύουν οι περιορισμοί ότι το μοντέλο είναι left-right και ότι $a_{ij} = 0$ για $j > i + 1$. Όλα τα στοιχεία του, με εξαίρεση αυτό που βρίσκεται στην τελευταία γραμμή και στήλη, είναι ίσα με $1/2$ υποδηλώνοντας ότι η πιθανότητα να παραμείνουμε στο state είναι ίση με την πιθανότητα να μεταβούμε στο επόμενο. Τέλος, αρχικοποιεί το GMM-HMM μοντέλο και επιστρέφει το εκπαιδευμένο.

Βήματα 11-12: Εκπαίδευση 10 μοντέλων με χρήση Expectation Maximization

Στο βήμα αυτό μεταβάλλουμε τις παραμέτρους **n_states** και **n_mixtures** του μοντέλου με στόχο να βρούμε τον καλύτερο συνδυασμό με το μεγαλύτερο accuracy. Συγκεκριμένα, εξετάζουμε αριθμό καταστάσεων HMM από 1 έως 4 και αριθμό Γκαουσιανών κατανομών από 1 έως 5. Για κάθε συνδυασμό αρχικοποιούμε μία κενή λίστα **hmm_models** στην οποία προσθέτουμε τα 10 εκπαιδευμένα, με χρήση του αλγορίθμου **Expectation Maximization**, μοντέλα όπως αυτά επιστρέφονται από την συνάρτηση **hmm** του προηγούμενου βήματος. Για την επιλογή των βέλτιστων τιμών των υπερπαραμέτρων ορίζουμε την συνάρτηση **calculate_accuracy** η οποία δέχεται για ορίσματα την λίστα **hmm_models** και τα δεδομένα (**data**, **labels**) από το validation set. Στο σημείο αυτό δεν έχουμε πρόσβαση στα test δεδομένα και επομένως θα ήταν λάθος να προσαρμόσουμε τις παραμέτρους του μοντέλου μας ώστε να γίνεται βέλτιστο σε αυτά. Για κάθε δείγμα από το σύνολο των validation αρχικοποιείται ένα διάνυσμα **logp** 10 διαστάσεων. Το στοιχείο στην i-οστή θέση του διανύσματος αυτού αντιστοιχεί στην log-probability που επιστρέφει ο αλγόριθμος Viterbi. Το μοντέλο εκείνο το οποίο δίνει τη μέγιστη πιθανοφάνεια είναι και το αποτέλεσμα της αναγνώρισης. Αν αυτό συμπίπτει με το πραγματικό label του υπό εξέταση δεδομένου τότε αυξάνεται κατά ένα ο αριθμός των σωστών προβλέψεων. Η συνάρτηση επιστρέφει το **accuracy** (λόγος των σωστών εκτιμήσεων προς το μέγεθος του validation/test set) καθώς επίσης και μία λίστα **predictions** με τις προβλέψεις του μοντέλου για όλα τα δεδομένα του συνόλου. Τα validation accuracies αποθηκεύονται σε ένα λεξικό, **validation_scores**, το οποίο έχει για κλειδιά τις τιμές των παραμέτρων σε μορφή tuple (**n_states**, **n_mixtures**) και για τιμές το αντίστοιχο accuracy. Οι βέλτιστες υπερπαράμετροι είναι εκείνες για τις οποίες το validation accuracy γίνεται μέγιστο.

Στην συνέχεια, για τις τιμές αυτές εκπαιδεύουμε τα 10 μοντέλα μας και βρίσκουμε την ακρίβεια στα validation και test δεδομένα. Προφανώς, ενδέχεται το ποσοστό για το test set να είναι

μικρότερο από το μέγιστο που λάβαμε προηγουμένως. Αυτό εξηγείται καθώς οι υπερπαράμετροι βελτιστοποιούνται μέσω διασταυρούμενης επικύρωσης (Grid Search) στα validation data και όχι στα test. Πράγματι, τα ποσοστά που λαμβάνουμε για τις βέλτιστες τιμές των n_states (n_states = 4) και n_mixtures (n_mixtures = 5) στο validation και test set είναι αντίστοιχα 98.7% και 98.67%. Επιπλέον παρατηρούμε πως για τις ίδιες τιμές των υπερπαραμέτρων το ποσοστό 98.7% (στα validation data) δεν συμπίπτει ακριβώς με το 99.07% που πετύχαμε κατά την βελτιστοποίηση. Το γεγονός αυτό οφείλεται στην τυχαιότητα που υπεισέρχεται κατά την εκπαίδευση των μοντέλων (πχ. λόγω των Γκαουσιανών κατανομών) οδηγώντας κάθε φορά σε ελαφρώς διαφορετικά αποτελέσματα.

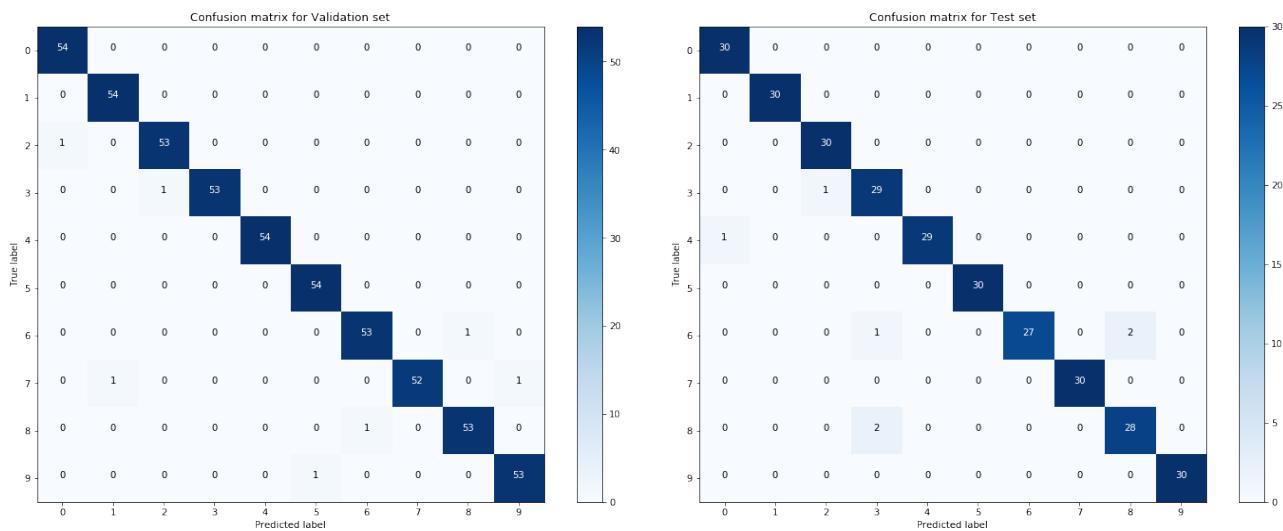
Στον ακόλουθο πίνακα παρουσιάζουμε συγκεντρωτικά τα ποσοστά ακρίβειας που λάβαμε κατά την διαδικασία βελτιστοποίησης των ταξινομητών για όλους τους συνδυασμούς των παραμέτρων n_states και n_mixtures.

Number of states	Number of mixtures	Validation accuracy
1	1	80.0 %
1	2	86.85 %
1	3	90.19 %
1	4	89.63 %
1	5	96.48 %
2	1	85.74 %
2	2	91.3 %
2	3	92.22 %
2	4	95.56 %
2	5	97.04 %
3	1	90.93 %
3	2	93.52 %
3	3	96.48 %
3	4	98.33 %
3	5	98.52 %
4	1	95.0 %
4	2	97.41 %
4	3	97.78 %
4	4	97.96 %
4	5	99.07 %

Ο λόγος για τον οποίο πραγματοποιούμε όλη αυτή τη διαδικασία που περιγράψαμε για την βελτιστοποίηση των υπερπαραμέτρων είναι για να είμαστε αμερόληπτοι στην αξιολόγηση αφήνοντας τελείως έξω το test set. Σε διαφορετική περίπτωση, αν χρησιμοποιούσαμε και τα δεδομένα του συνόλου ελέγχου θα καταλήγαμε σε ένα μοντέλο και ένα σύνολο παραμέτρων το οποίο θα λειτουργούσε καλύτερα για **όλα** τα δεδομένα και συνεπώς θα ήταν περισσότερο επιρρεπές σε **overfitting**. Επιπλέον, δεν θα είχαμε εκτίμηση για το πόσο καλά θα αποδώσει το μοντέλο σε νέα δεδομένα καθώς θα είχε ήδη ‘δει’ και χρησιμοποιήσει το test set για το tuning των υπερπαραμέτρων.

Βήμα 13: Σχηματισμός Confusion Matrixes για validation και test set

Παρουσιάζουμε τώρα τους Confusion Matrixes οι οποίοι περιέχουν τα αποτελέσματα της αναγνώρισης του προηγούμενου Βήματος για το validation και test set. Πιο συγκεκριμένα, οι πίνακες σύγχυσης περιέχουν στις γραμμές τους τα προς ταξινόμηση ψηφία, ενώ στις στήλες τους τις κλάσεις στις οποίες αυτά ταξινομήθηκαν. Για την εύρεση των 10×10 πινάκων χρησιμοποιούμε την συνάρτηση **confusion_matrix** της sklearn, ενώ για τον σχεδιασμό τους την έτοιμη **plot_confusion_matrix** που δίνεται από [εδώ](#).



Σχήμα 28: Πίνακες σύγχυσης για το validation και test set

Από το παραπάνω Σχήμα επιβεβαιώνουμε την άριστη επίδοση του μοντέλου μας καθώς και οι δύο πίνακες έχουν σχεδόν όλες τους τις τιμές συγκεντρωμένες στην κύρια διαγώνιο τους.

Βήμα 14: Δημιουργία LSTM και ρύθμιση των υπερπαραμέτρων δικτύου

Στο βήμα αυτό εκπαιδεύουμε ένα Αναδρομικό Νευρωνικό Δίκτυο πάνω στο training set, χρησιμοποιώντας το validation set για τη ρύθμιση των υπερπαραμέτρων.

Ερωτήματα 1,2: Δημιουργία απλού LSTM

Για τη δημιουργία ενός απλού LSTM συμπληρώνουμε κατάλληλα το βοηθητικό κώδικα του αρχείου **lstm.py** που μας δίνεται.

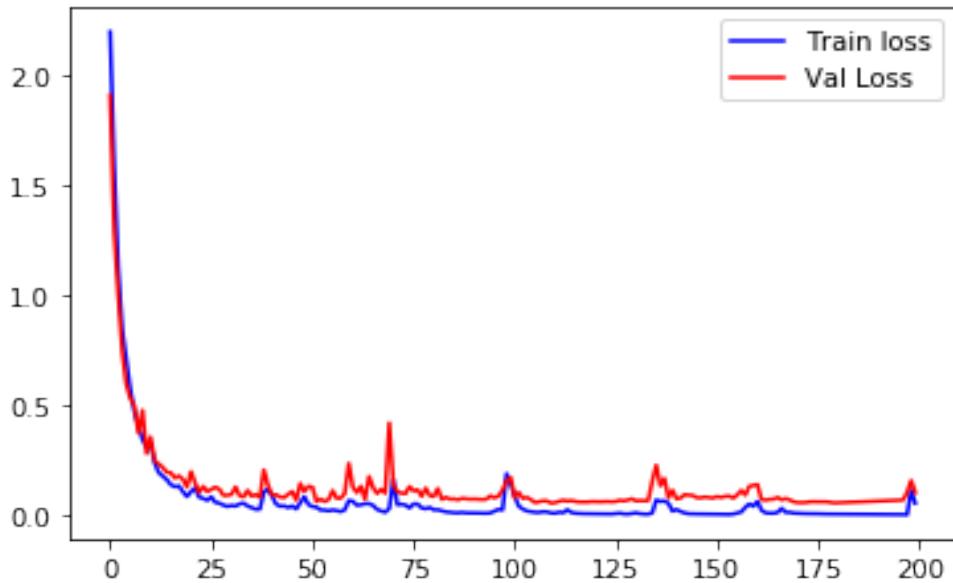
Επιλέγουμε τις εξής παραμέτρους για το δίκτυο:

- `input_dim = 6`
- `hidden_dim = 32`
- `layer_dim = 2`
- `output_dim = 10`

Ερωτήματα 3,4: Εκπαίδευση απλού LSTM

Για το training του μοντέλου χρησιμοποιούμε **Cross Entropy loss function**, **Adam optimizer**, **num_epochs = 200** και **batch_size = 32**.

Το **validation accuracy** που προκύπτει για το δίκτυο μας ανέρχεται στο **96.67%**, ποσοστό εξαιρετικά υψηλό, δεδομένου ότι το δίκτυο είναι unidirectional ενώ δεν έγινε και χρήση κάποιας εκ των τεχνικών αντιμετώπισης του overfitting που θα δούμε στη συνέχεια. Τα διαγράμματα για το **train** και **validation loss** φαίνονται στην ακόλουθη εικόνα:

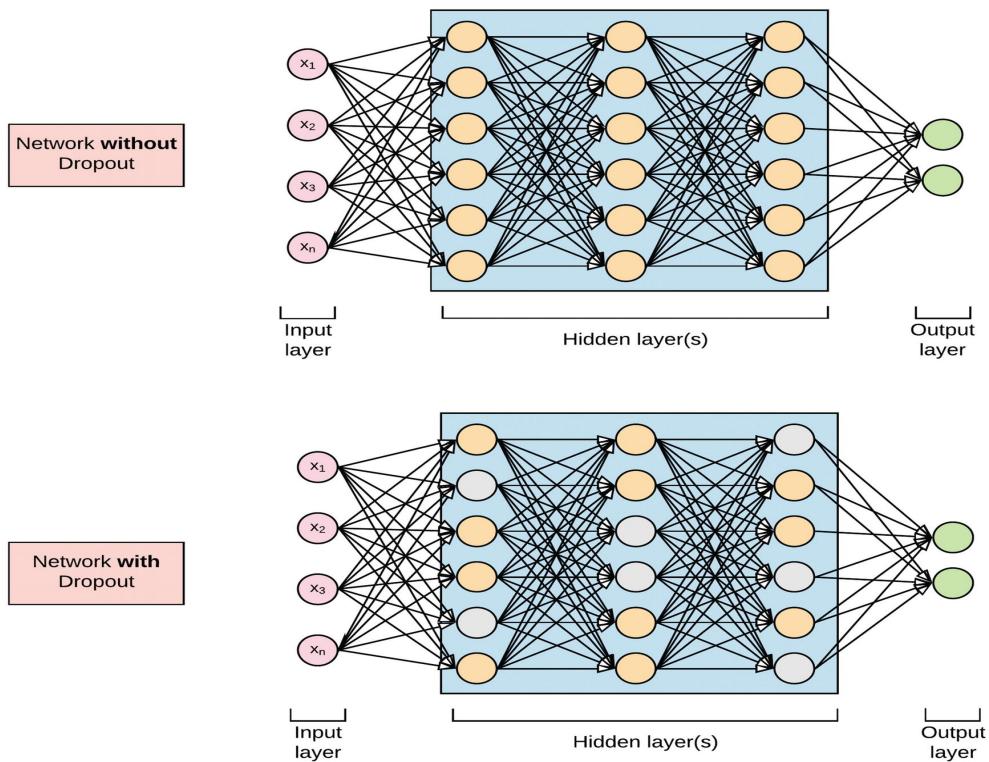


Σχήμα 29: Διαγράμματα train και validation loss απλού LSTM

Ερώτημα 5: Προσθήκη Dropout και L2 Regularization

Στο ερώτημα αυτό εισάγουμε στο μοντέλο μας **Dropout** και **L2 Regularization**.

- **Dropout:** Μια τεχνική για τη μείωση του overfitting είναι η εισαγωγή του dropout στο δίκτυο. Είναι μια μορφή κανονικοποίησης που αναγκάζει τα βάρη στο δίκτυο να λαμβάνουν μόνο μικρές τιμές, γεγονός που καθιστά την κατανομή των τιμών βάρους κανονική και το δίκτυο μπορεί να μειώσει την υπερεκπαίδευση σε μικρά δείγματα εκπαίδευσης. Όταν εφαρμόσουμε το dropout σε ένα επίπεδο, 'πετάμε' τυχαία (θέτοντας τα μηδέν) ένα πλήθος μονάδων εξόδου από το επίπεδο που το εφαρμόζουμε, κατά τη διάρκεια της διαδικασίας εκπαίδευσης. Το Dropout παίρνει έναν κλασματικό αριθμό ως τιμή εισόδου του, όπως 0.1, 0.2, 0.4, κλπ. Αυτό ισοδυναμεί με κατάργηση 10%, 20% ή 40% των μονάδων εξόδου τυχαία από το εφαρμοζόμενο επίπεδο. Για το μοντέλο μας επιλέγουμε dropout ίσο με **0.2**.



Σχήμα 30: Παράδειγμα δικτύου με και χωρίς Dropout

- **L2 Regularization:** Η τεχνική αυτή προσθέτει στη loss function ένα penalty term που ισούται με το τετράγωνο της L2 νόρμας του διανύσματος βαρών και αποσκοπεί στην αντιμετώπιση του overfitting.

$$Loss = Error(y, \hat{y}) + \lambda \sum_{i=1}^N w_i^2$$

Σχήμα 31: Προσθήκη στην loss function του όρου που σημειώνεται με κίτρινο χρώμα

Το μέγεθος της κανονικοποίησης ρυθμίζεται από την παράμετρο λ . Αν αυτή είναι μηδενική, τότε δεν υφίσταται καθόλου regularization και η loss function αποτελείται μόνο από το σφάλμα μεταξύ της εξόδου y και της πρόβλεψης \hat{y} . Για πολύ μεγάλες τιμές του λ , προστίθεται μεγάλο επιπλέον βάρος και αυτό οδηγεί το μοντέλο σε underfitting. Φαίνεται λοιπόν πως η κατάλληλη επιλογή του λ είναι ιδιαίτερα σημαντική. Στο μοντέλο μας εισάγουμε την τεχνική L2 Regularization μέσω του ορίσματος **weight_decay** του optimizer, θέτοντας την τιμή αυτή ίση με **10⁻⁴**.

Εφαρμόζουμε λοιπόν τις δύο παραπάνω μεθόδους στο μοντέλο μας και το εκπαιδεύουμε ξανά. Στην περίπτωση αυτή, λαμβάνουμε τελικό ποσοστό για το **validation accuracy** ίσο με **98.15%**.

Ερώτημα 6: Υλοποίηση Early Stopping και Checkpoints

Στο ερώτημα αυτό υλοποιούμε τη διαδικασία του **Early Stopping**. Το Early Stopping αποτελεί μια τεχνική αντιμετώπισης του overfitting, κατά την οποία η εκπαίδευση του δικτύου διακόπτεται πρόωρα αν δεν εμφανίζεται βελτίωση ως προς κάποια μετρική απόδοσης που παρακολουθούμε (συνήθως αυτή είναι το validation loss). Για το μοντέλο μας, υλοποιούμε τη διαδικασία του Early Stopping, κάνοντας monitor το loss που προκύπτει από τα validation data σε κάθε εποχή. Θεωρούμε μια παράμετρο ανοχής **patience**, η οποία καθορίζει το πόσο ανεκτικοί είμαστε ως προς την επιδείνωση του loss. Συγκεκριμένα, αν αυτό δεν βελτιωθεί μετά από patience εποχές, τότε διακόπτουμε την εκπαίδευση του δικτύου. Να σημειωθεί ότι κάθε φορά που λαμβάνουμε ένα νέο ολικό ελάχιστο για τη συνάρτηση του validation loss αποθηκεύουμε το εκάστοτε μοντέλο σε pickle με χρήση της εντολής **torch.save**, ώστε να μπορούμε να το ανακαλέσουμε στη συνέχεια, με χρήση της **torch.load**.

Εφαρμόζουμε την τεχνική του Early Stopping θέτοντας **patience = 10**. Τα αποτελέσματα του accuracy για το ίδιο μοντέλο χωρίς και με χρήση του πρόωρου τερματισμού φαίνονται στον ακόλουθο πίνακα.

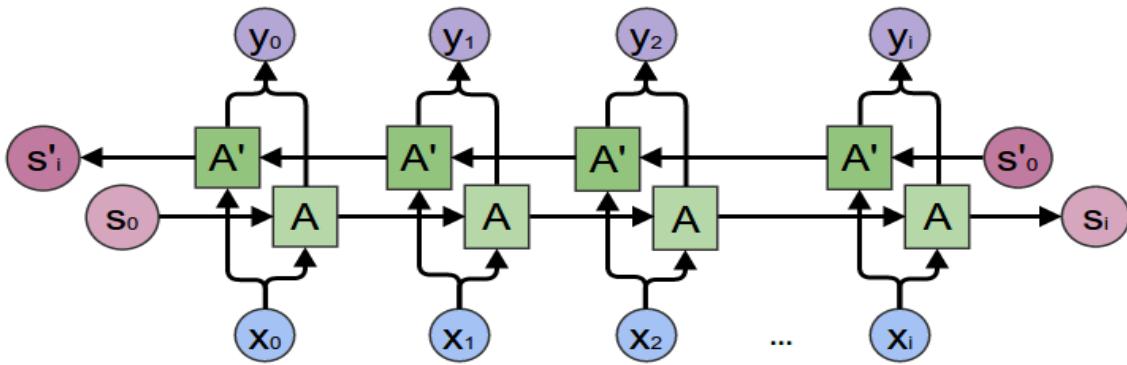
Early Stopping	Accuracy on validation set	Accuracy on test set
✗	95.6 %	93.3 %
✓	98.7 %	97.0 %

Παρατηρούμε πως το accuracy στο validation set βελτιώθηκε πάνω από 3 % με χρήση του Early Stopping. Αντίστοιχα, και η ακρίβεια στα test δεδομένα αυξήθηκε κατά 3.7 %.

Ερώτημα 7: Εκπαίδευση Bidirectional LSTM

Τα **Bidirectional** Αναδρομικά Νευρωνικά Δίκτυα (RNNs) αποτελούν απλώς το συνδυασμό δύο ανεξάρτητων unidirectional RNNs. Η δομή αυτή επιτρέπει στα δίκτυα να αντλούν κάθε χρονική στιγμή πληροφορίες τόσο από το παρελθόν όσο και από το μέλλον μιας ακολουθίας (backward and forward information). Αυτό συμβαίνει κανώς ένα bidirectional δίκτυο διαβάζει την είσοδο με δύο τρόπους:

1. Από το παρελθόν στο μέλλον
2. Από το μέλλον στο παρελθόν



Σχήμα 32: Σχεδιάγραμμα ενός Bidirectional LSTM

Αυτό λοιπόν που διαφοροποιεί την προσέγγιση αυτή από τα μονοκατευθυντικά δίκτυα (unidirectional) είναι το γεγονός ότι το LSTM που διαβάζει την ακολουθία από το τέλος προς στην αρχή, διατηρεί πληροφορία από το μέλλον και έτσι με τον συνδυασμό των δύο hidden states, το bidirectional δίκτυο είναι ικανό οποιαδήποτε στιγμή να αξιοποιεί πληροφορίες και από τις δύο κατευθύνσεις. Έτσι, το μοντέλο αποκτά μια καλύτερη εποπτεία του context και αποδίδει καλύτερα.

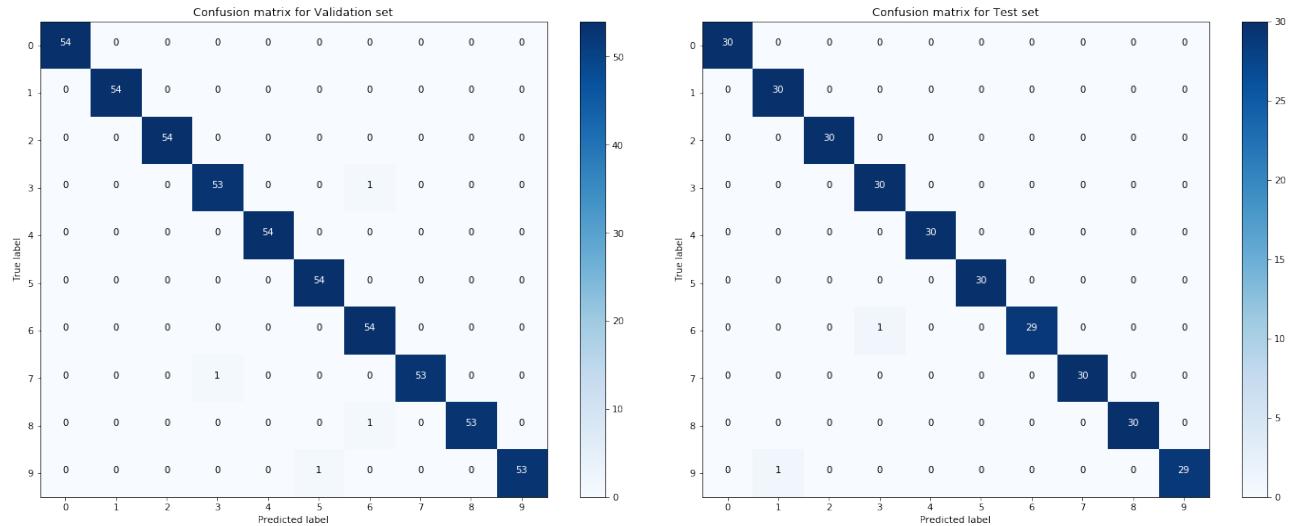
Θέτουμε λοιπόν στο μοντέλο μας την παράμετρο **bidirectional = True** και το εκπαιδεύουμε ξανά. Στην περίπτωση αυτή, λαμβάνουμε τελικό ποσοστό για το **validation** αλλά και για το **test accuracy** ίσο με **99.3%**.

Συγκεντρώνουμε τα αποτελέσματα για όλα τα μοντέλα που εκπαιδεύσαμε στον ακόλουθο πίνακα:

Dropout	L2 Regularization	Early Stopping	Bidirectional	Validation Accuracy
0	0	✗	✗	96.67 %
0.2	10^{-4}	✗	✗	98.15 %
0.2	10^{-4}	✓	✗	98.7 %
0.2	10^{-4}	✓	✓	99.3 %

Παρατηρούμε πως το καλύτερο μοντέλο μας, με μικρή διαφορά, είναι το **Bidirectional LSTM**, το οποίο πετυχαίνει σχεδόν άψογη κατηγοριοποίηση. Για το συγκεκριμένο, πρόβλημα δεν είναι ιδιαίτερα αισθητή η βελτίωση του accuracy με χρήση bidirectional δικτύου, καθώς ακόμα και το απλό LSTM αποδίδει σχεδόν τέλεια. Αυτό είναι φυσιολογικό εφόσον έχουμε ένα εύκολο πρόβλημα κατηγοριοποίησης, στο οποίο **τα δεδομένα δεν περιέχουν θόρυβο**.

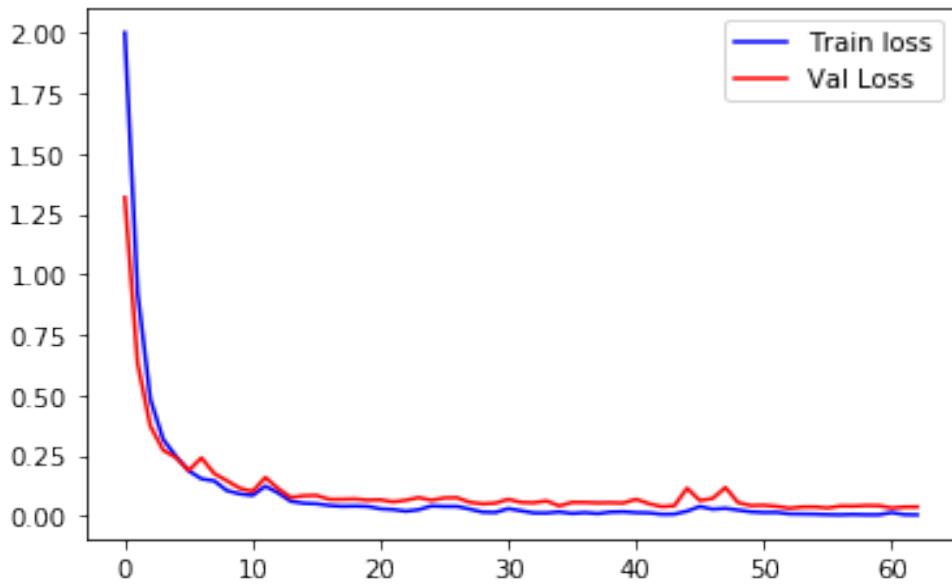
Τα **Confusion Matrixes** του Bidirectional LSTM για το validation και test set φαίνονται στην ακόλουθη εικόνα:



Σχήμα 33: Πίνακες σύγχυσης του βέλτιστου μοντέλου για το validation και test set

Από το παραπάνω Σχήμα επιβεβαιώνουμε την άριστη επίδοση του μοντέλου μας καθώς και οι δύο πίνακες έχουν σχεδόν όλες τις τιμές συγκεντρωμένες στην κύρια διαγώνιο τους.

Τα διαγράμματα για το **train** και **validation loss** φαίνονται στην ακόλουθη εικόνα:



Σχήμα 34: Διαγράμματα train και validation loss βέλτιστου μοντέλου

Από τα παραπάνω δύο διαγράμματα είναι ορατό πως δεν υπάρχει το πρόβλημα του overfitting ενώ παράλληλα επιτυγχάνουμε σχεδόν μηδενικές τιμές για τα δύο losses.

Ερώτημα 8 (BONUS): Χρήση pack_padded_sequence στην εκπαίδευση

Τέλος, υλοποιούμε το **pack_padded_sequence** στην εκπαίδευση του μοντέλου. Για το σκοπό αυτό, ορίζουμε τη συνάρτηση **pack_pad**, η οποία αρχικά ταξινομεί όλα τα training data σε φθίνουσα σειρά μήκους (length). Στη συνέχεια, η συνάρτηση δημιουργεί το ταξινομημένο dataset, αξιοποιώντας την κλάση **FrameLevelDataset** που έχουμε ορίσει, ενώ καλεί και την **Dataloader** για την δημιουργία του αντίστοιχου sorted dataloader. Να σημειωθεί ότι στην παρούσα φάση τα batches περιέχουν δεδομένα ταξινομημένα ως προς το αρχικό τους μήκος, ωστόσο έχουν όλα υποστεί zero padding για να αποκτήσουν μέγεθος 153 (όσο δηλαδή και το length της μεγαλύτερης ηχογράφησης). Για κάθε batch η συνάρτηση εντοπίζει το αρχικό μήκος της μεγαλύτερης ηχογράφησης που περιέχεται στο batch και έπειτα διαγράφει τα περιττά μηδενικά από όλα τα δεδομένα που έχουν μικρότερο αρχικό length από αυτό το max. Αυτό επιτυγχάνεται κρατώντας, με κατάλληλο slicing, μόνο τις max πρώτες γραμμές κάθε αρχείου. Έτσι, δημιουργούμε batches διαφορετικών μεγεθών, τα οποία ωστόσο διατηρούν όλη την αρχική πληροφορία. Η επιλογή ή όχι της εφαρμογής της τεχνικής pack_padded_sequence γίνεται μέσω μιας boolean παραμέτρου.

Εκπαιδεύουμε λοιπόν πάλι το προηγούμενο δίκτυο μας, αλλά τώρα μόνο για 50 εποχές και χωρίς Early Stopping, μια φορά χωρίς και μια φορά με pack_padded_sequence, έτσι ώστε να συγκρίνουμε τους χρόνους εκπαίδευσης. Οι χρόνοι συνοψίζονται στο παρακάτω πίνακα:

pack_padded_sequence	fit time
✗	757.4 sec
✓	185.4 sec

Όπως φαίνεται από τον πίνακα, ο χρόνος που απαιτείται για την εκπαίδευση του νευρωνικού με τη χρήση **pack_padded_sequence** είναι σημαντικά μικρότερος (περίπου υποτετραπλάσιος). Αυτό είναι αναμενόμενο καθώς παρόλο που η μεγαλύτερη ηχογράφηση έχει μήκος 153, η πλειοψηφία των ηχογραφήσεων έχει length μικρότερο του 80, με αποτέλεσμα πολλά recordings να αυξάνουν αχρείαστα κατά πολύ το μέγεθός τους.