ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΑΝΑΓΝΩΡΙΣΗ ΠΡΟΤΥΠΩΝ

(2021-2022)

2ο Εργαστηριακό Project

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

Ονοματεπώνυμο:

- Χρήστος Τσούφης

Αριθμός Μητρώου:

- 03117176

Στοιχεία Επικοινωνίας:

- el17176@mail.ntua.gr

Περιγραφή

Σκοπός 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, χρησιμοποιώντας τα κατάλληλα πακέτα Python, καθώς και η ανάλυση και απεικόνισή τους με σκοπό την κατανόηση και την εξαγωγή χρήσιμων πληροφοριών από αυτά. Τα εν λόγω χαρακτηριστικά είναι στην ουσία ένας αριθμός συντελεστών cepstrum που εξάγονται μετά από ανάλυση των σημάτων με μια ειδικά σχεδιασμένη συστοιχία φίλτρων (filterbank). Η συστοιχία αυτή είναι εμπνευσμένη από ψυχοακουστικές μελέτες. Πιο συγκεκριμένα, το σύστημα που αναπτύσσεται αφορά την αναγνώριση μεμονωμένων ψηφίων (isolated digits) στα Αγγλικά. Τα δεδομένα που χρησιμοποιούνται, περιέχουν εκφωνήσεις 9 ψηφίων από 15 διαφορετικούς ομιλητές σε ξεχωριστά .wav αρχεία. Συνολικά υπάρχουν 133 αρχεία, αφού 2 εκφωνήσεις θεωρήθηκαν προβληματικές και δεν έχουν συμπεριληφθεί. Τα ονόματα των αρχείων (π.χ. eight8.wav) υποδηλώνουν τόσο το ψηφίο που εκφωνείται (π.χ. eight), όσο και τον ομιλητή (οι ομιλητές είναι αριθμημένοι από 1-15). Οι εκφωνήσεις έχουν ηχογραφηθεί με συχνότητα δειγματοληψίας ίση με Fs = 16kHz και η διάρκειά τους διαφέρει.

Τεχνολογίες & Τρόπος Εκτέλεσης εφαρμογής

Η παρούσα εργασία υλοποιήθηκε σε ένα Python περιβάλλον και συγκεκριμένα με την χρήση των Notebooks του Google Colab τα οποία στην συνέχεια έγιναν export σε .py αρχεία.

Σημείωση: Τελικά χρησιμοποιήθηκαν οι βιβλιοθήκες και οι συγκεκριμένες εκδόσεις που αναφέρονται στο Github.

Βιβλιοθήκες:

- Διάβασμα αρχείων ήχου και εξαγωγή χαρακτηριστικών: librosa
- Αλγόριθμοι ταξινόμησης: scikit-learn
- Διαγράμματα: matplotlib, seaborn etc.
- Hidden Markov Models: pomegranate
- Νευρωνικά δίκτυα: pytorch

Δομή:

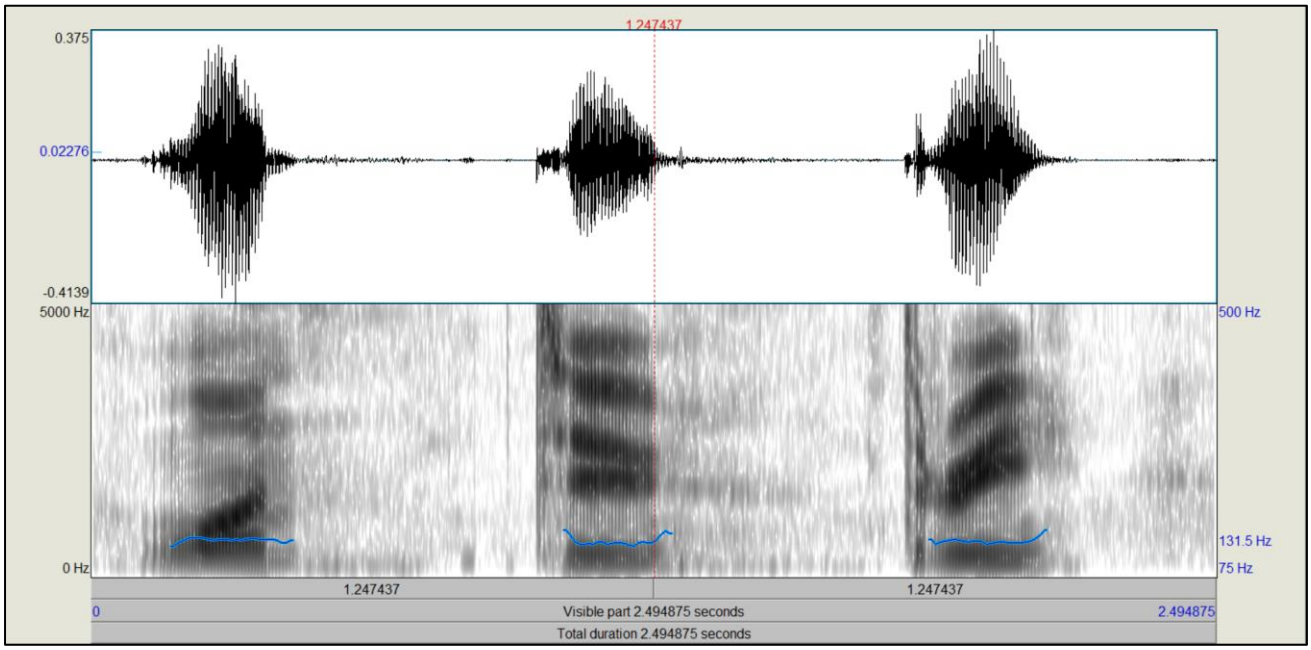
Αποτελείται από το αρχείο `patrec_lab2_main.py` και `patrec_lab2_prep.py` που περιέχει όλα τα βήματα και τις συναρτήσεις. Η εκτέλεση των αρχείων γίνεται μέσω του `Colab` αφού πρώτα τοποθετηθούν στον ίδιο φάκελο τα κατάλληλα αρχεία του φακέλου (`pr_lab2_2020-21_data`) (`recordings`) που θα χρησιμοποιηθούν ως input. Επίσης, πρέπει οι φάκελοι digits και recordings να βρίσκονται στο ίδιο directory.

Εκτέλεση

Βήμα 1

Ανάλυση αρχείων ήχου με το Praat (το οποίο πρέπει να εγκαταστήσετε από το παραπάνω link): Ανοίξτε τα αρχεία onetwothree1.wav και onetwothree8.wav με το πρόγραμμα Praat. Τα αρχεία αυτά περιέχουν την πρόταση "one two three" από τους ομιλητές 1 και 8, οι οποίοι είναι άντρας και γυναίκα αντίστοιχα. Παρατηρήστε τις κυματομορφές και τα spectrograms και έπειτα εξάγετε τη μέση τιμή του pitch στα φωνήεντα "α", "ου", "ι" για τα 3 ψηφία και για κάθε ομιλητή. Έπειτα, εξάγετε τα 3 πρώτα formants του κάθε φωνήεντος. Παρουσιάστε τα αποτελέσματα και γράψτε τις παρατηρήσεις σας.

Αρχικά, γίνεται η εγκατάσταση του προγράμματος Praat. Έπειτα, κάνοντας Open το αρχείο onetwothree1.wav και μετά View & Edit, οπτικοποιούνται οι κυματομορφές και προκύπτει η ακόλουθη εικόνα:

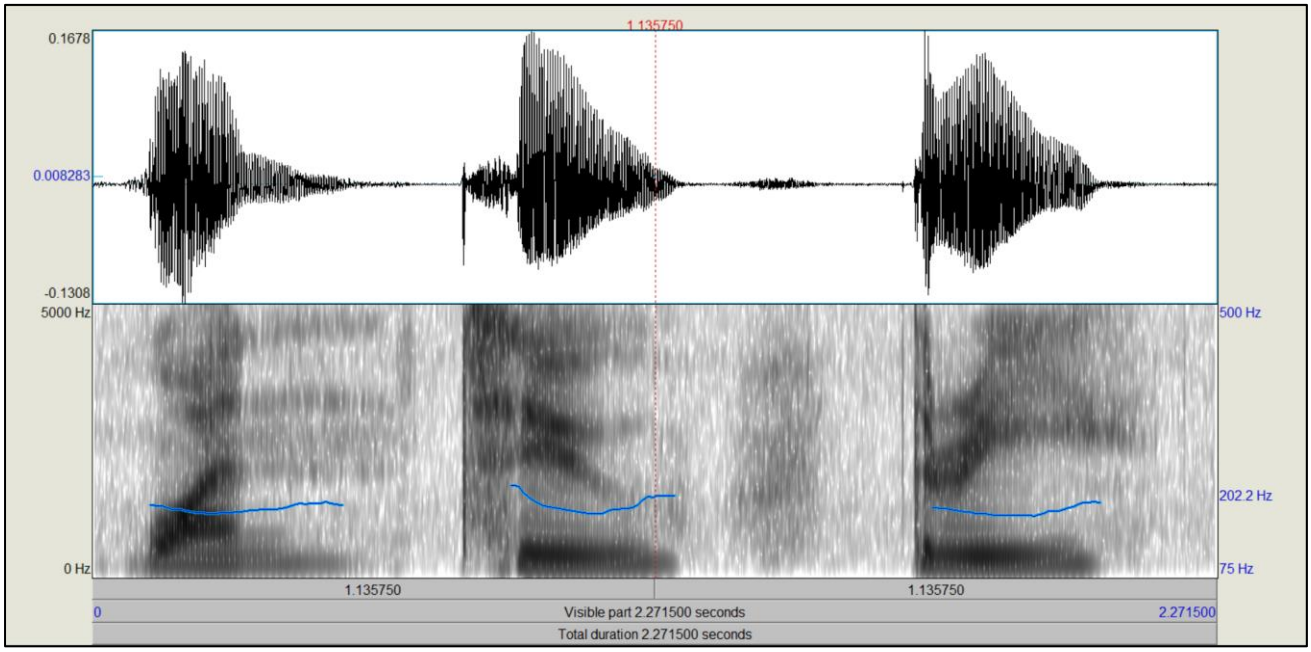


Το πρώτο αρχείο ήχου αντιστοιχεί σε άνδρα εκφωνητή. Κατά προσέγγιση, χωρίζονται οι περιοχές των φωνηέντων που ζητούνται και τελικά προκύπτουν οι ακόλουθες τιμές για το mean pitch.

Άνδρας Ομιλητής:

Αριθμός	**Φωνήεν**	**Pitch (Hz)**
One	"ου"	127.63
	"α"	135.52
Two	"ου"	128.41
Three	"ι"	130.30

Στην συνέχεια, κάνοντας Open το αρχείο onetwothree8.wav και μετά View & Edit, προκύπτει η ακόλουθη εικόνα:



Το δεύτερο αρχείο ήχου αντιστοιχεί σε γυναίκα εκφωνήτρια. Κατά προσέγγιση, χωρίζονται οι περιοχές των φωνηέντων που ζητούνται και τελικά προκύπτουν οι ακόλουθες τιμές για το mean pitch.

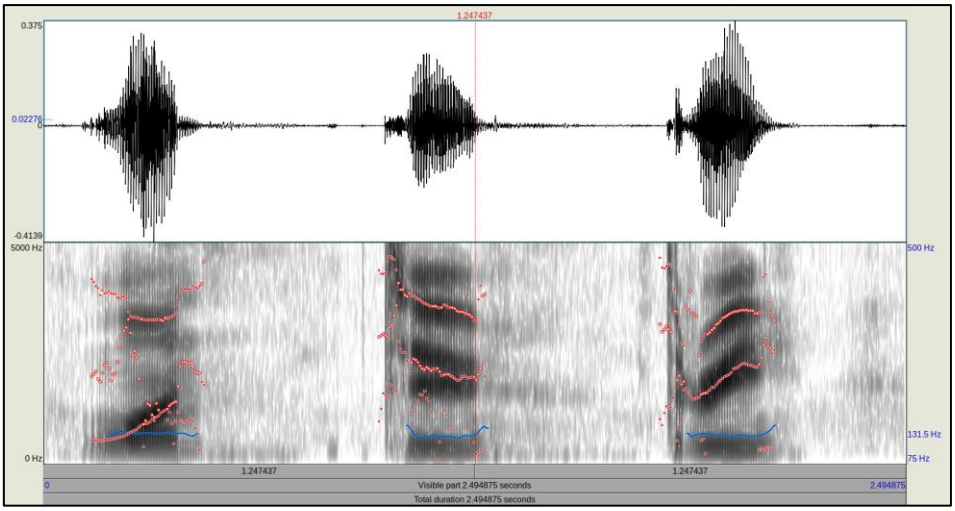
Γυναίκα ομιλήτρια:

Αριθμός	**Φωνήεν**	**Pitch (Hz)**
One	“ου”	187.45
	“α”	177.15
Two	“ου”	185.03
Three	“ι”	173.15

Ύστερα, εξάγεται η μέση τιμή του pitch στα φωνήεντα “α”, “ου”, “ι” για τα τρία ψηφία και για κάθε ομιλητή καθώς και τα 3 πρώτα formats του κάθε φωνήεντος. Δεδομένου ότι η εξαγωγή των χαρακτηριστικών γίνεται χειροκίνητα μέσω του Praat, τα αποτελέσματα κατά προσέγγιση φαίνονται παρακάτω. Η εξαγωγή γίνεται επιλέγοντας το διάστημα της κυματομορφής που βρίσκεται το φωνήεν και μετά από την καρτέλα “Pitch” επιλέγεται το “Get Pitch”. Μετά, κεντράρεται το επιθυμητό σημείο και από την καρτέλα “Format” επιλέγεται το “Get format”.

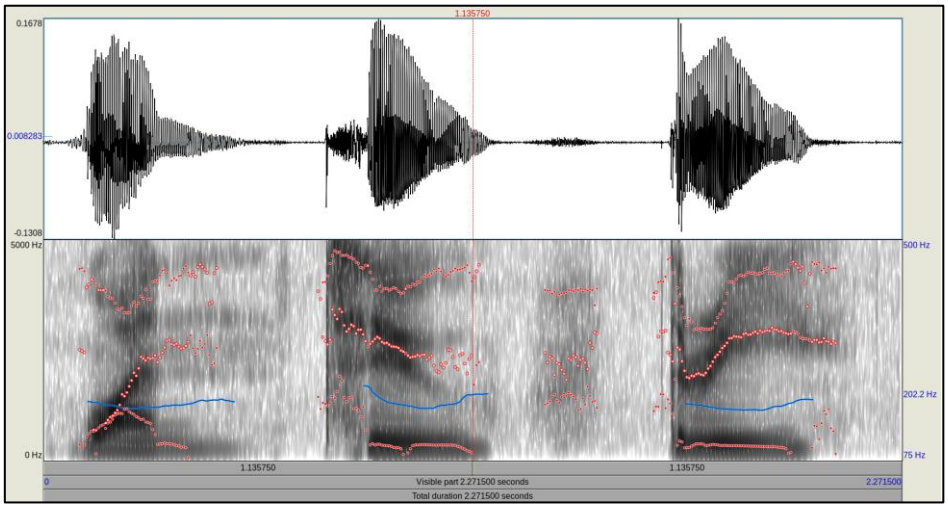
Έτσι, εξάγονται τα 3 πρώτα formats για τα φωνήεντα για τον Άνδρα Ομιλητή:

Αριθμός	**Φωνήεν**	**F_1 (Hz)**	**F_2 (Hz)**	**F_3 (Hz)**
One	“ου”	506.32	978.82	2358.59
	“α”	772.70	959.20	2363.03
Two	“ου”	323.37	1805.17	2412.06
Three	“ι”	376.04	1937.40	2375.71



Τέλος, εξάγονται τα 3 πρώτα formats για τα φωνήεντα για την Γυναίκα Ομιλήτρια:

Αριθμός	**Φωνήεν**	**F_1 (Hz)**	**F_2 (Hz)**	**F_3 (Hz)**
One	“ου”	507.61	834.68	2356.23
	“α”	832.74	1219.70	2771.04
Two	“ου”	356.30	1846.94	2620.60
Three	“ι”	348.79	2232. 90	2603.08



Σχολιασμός:

Παρατηρείται ότι τα formats είναι περίπου ίδια και για τους δύο εκφωνητές διότι εξαρτώνται από το εκάστοτε φωνήεν. Συνεπώς, τα formats είναι χρήσιμα για τον διαχωρισμό των φωνηέντων.

Επιπλέον, η γυναίκα φαίνεται να έχει μεγαλύτερο pitch στην φωνή της σε σχέση με τον άντρα όπως προκύπτει και από τους παραπάνω πίνακες. Μάλιστα, το pitch φαίνεται να διαφέρει και δεν μεταβάλλεται ουσιαστικά όταν αλλάζει το φωνήεν. Επομένως, το pitch είναι ένα χαρακτηριστικό που θα μπορούσε να χρησιμεύσει για να ξεχωρίσει κανείς το φύλο του ομιλητή.

Βήμα 2

Φτιάξτε μία συνάρτηση (data parser) που να διαβάζει όλα τα αρχεία ήχου που δίνονται μέσα στο φάκελο digits/ και να επιστρέφει 3 λίστες Python, που να περιέχουν: Το wav που διαβάστηκε με librosa, τον αντίστοιχο ομιλητή και το ψηφίο.

Στο σημείο αυτό υλοποιήθηκε μια custom συνάρτηση parser() η οποία ουσιαστικά διαβάζει όλα τα αρχεία από τον φάκελο /digits/ που δόθηκε και επιστρέφει τρεις λίστες Python. Η πρώτη λίστα επιστρέφει τα αρχεία .wav τα οποία φορτώνονται με την βοήθεια της librosa.core.load, η δεύτερη λίστα επιστρέφει τους εκφωνητές και η τρίτη λίστα επιστρέφει τα ψηφία.

Βήμα 3

Εξάγετε με το librosa τα Mel-Frequency Cepstral Coefficients (MFCCs) για κάθε αρχείο ήχου. Εξάγετε 13 χαρακτηριστικά ανά αρχείο. Χρησιμοποιήστε μήκος παραθύρου 25 ms και βήμα 10 ms. Επίσης, υπολογίστε και την πρώτη και δεύτερη τοπική παράγωγο των χαρακτηριστικών, τις λεγόμενες deltas και delta-deltas (hint: υπάρχει έτοιμη υλοποίηση στο librosa).

Με την χρήση των συναρτήσεων της βιβλιοθήκης librosa εξάγονται οι MFCCs για κάθε αρχείο wav που διαβάστηκε στο προηγούμενο βήμα φτιάχνοντας μια λίστα που περιέχει τα MFCCs. Αυτό επιτυγχάνεται παίρνοντας κάθε στοιχείο wav από την προηγούμενη λίστα στο Βήμα 2 και με την βοήθεια της librosa.feature.mfcc δημιουργούνται τα framesμε τις παραμέτρους:

- με 13 χαρακτηριστικά οπότε n_mfcc = 13
- με window 25ms οπότε n_fft = int (sr * window_length)
- με step 10ms οπότε hop_length = int (sr * hop_length)
- με sr η συχνότητα δειγματοληψίας (Sampling rate) του αρχείου wav.

Επιπλέον, γίνεται ο υπολογισμός της πρώτης και της δεύτερης τοπικής παραγώγου των χαρακτηριστικών μέσω συναρτήσεων της librosa και συγκεκριμένα, της librosa.feature.delta, φτιάχνοντας δύο λίστες που περιέχουν τα deltas και delta-deltas αντίστοιχα.

Βήμα 4

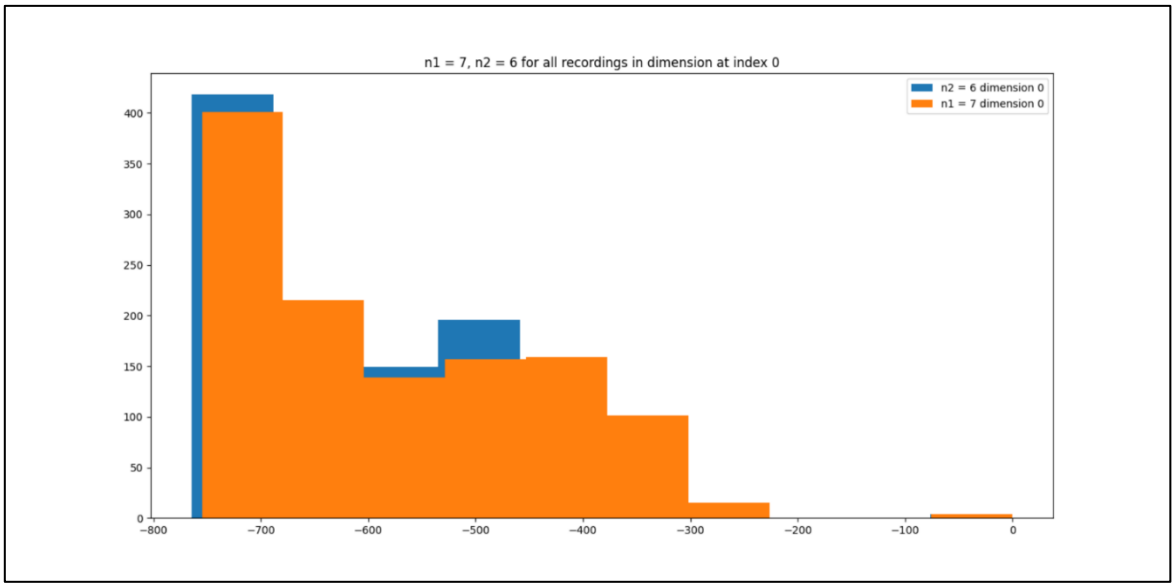
Αναπαραστήστε τα ιστογράμματα του 1ου και του 2ου MFCC των ψηφίων n1 και n2 για όλες τους τις εκφωνήσεις. Πόση απόκλιση υπάρχει? Εξάγετε για 2 εκφωνήσεις των n1 και n2 από 2 διαφορετικούς ομιλητές τα Mel Filterbank Spectral Coefficients (MFSCs), δηλαδή τα χαρακτηριστικά που εξάγονται αφού εφαρμοστεί η συστοιχία φίλτρων της κλίμακας Mel πάνω στο φάσμα του σήματος φωνής αλλά χωρίς να εφαρμοστεί στο τέλος ο μετασχηματισμός DCT (εξάγετε και πάλι χαρακτηριστικά διάστασης 13). Αναπαραστήστε γραφικά τη συσχέτιση των MFSCs για την κάθε εκφώνηση. Σε ξεχωριστά διαγράμματα πραγματοποιήστε το ίδιο για τα MFCCs. Τι παρατηρείτε? Γιατί χρησιμοποιούμε τα MFCCs αντί των MFSCs?

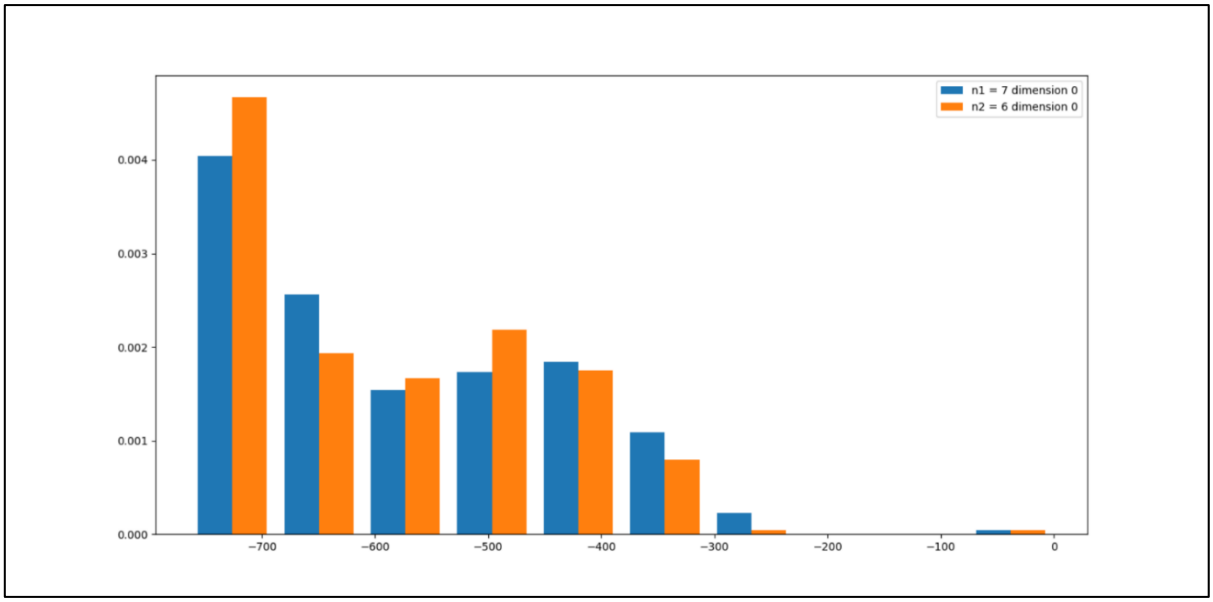
ΣΗΜΕΙΩΣΗ: Τα ψηφία n1 και n2 που αναφέρονται παραπάνω, είναι το προτελευταίο και το τελευταίο ψηφίο του Α.Μ. σας αντίστοιχα. Αν συμμετέχουν δύο συνεργάτες διαλέξτε το τελευταίο ψηφίο κάθε ΑΜ. Αν κάποιο είναι το 0 ή είναι ίδια τα ψηφία, τότε ορίστε το ως το προηγούμενο ή επόμενο του άλλου ψηφίου, το οποίο δεν είναι μηδενικό.

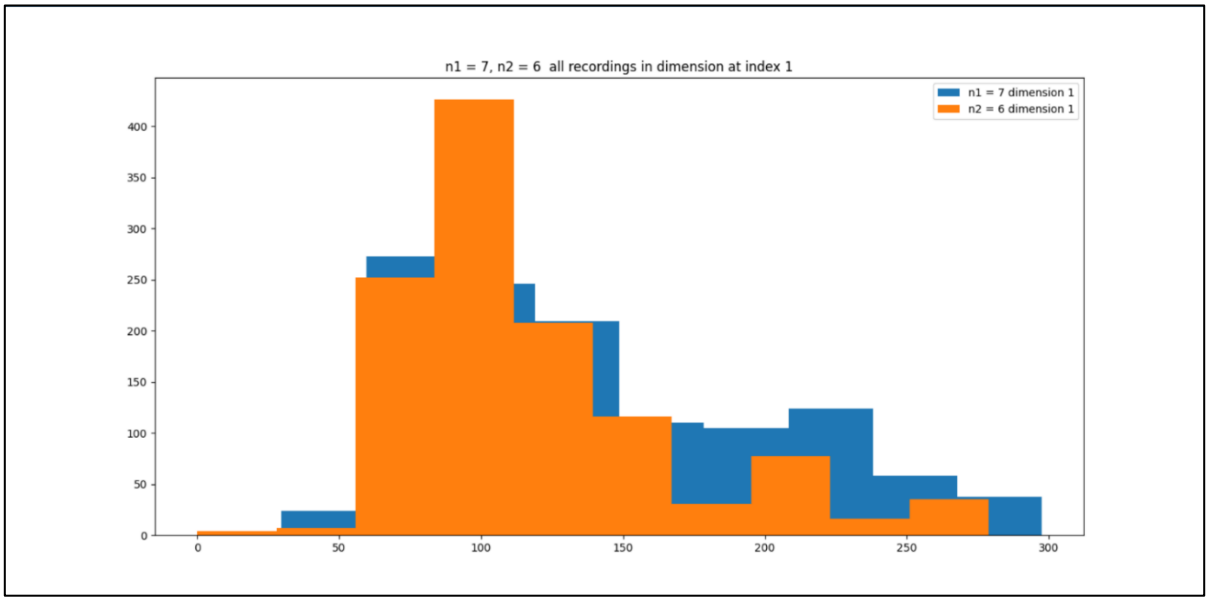
Προκύπτει ότι οι αριθμοί n1 και n2 είναι αντίστοιχα οι 7 και 6.

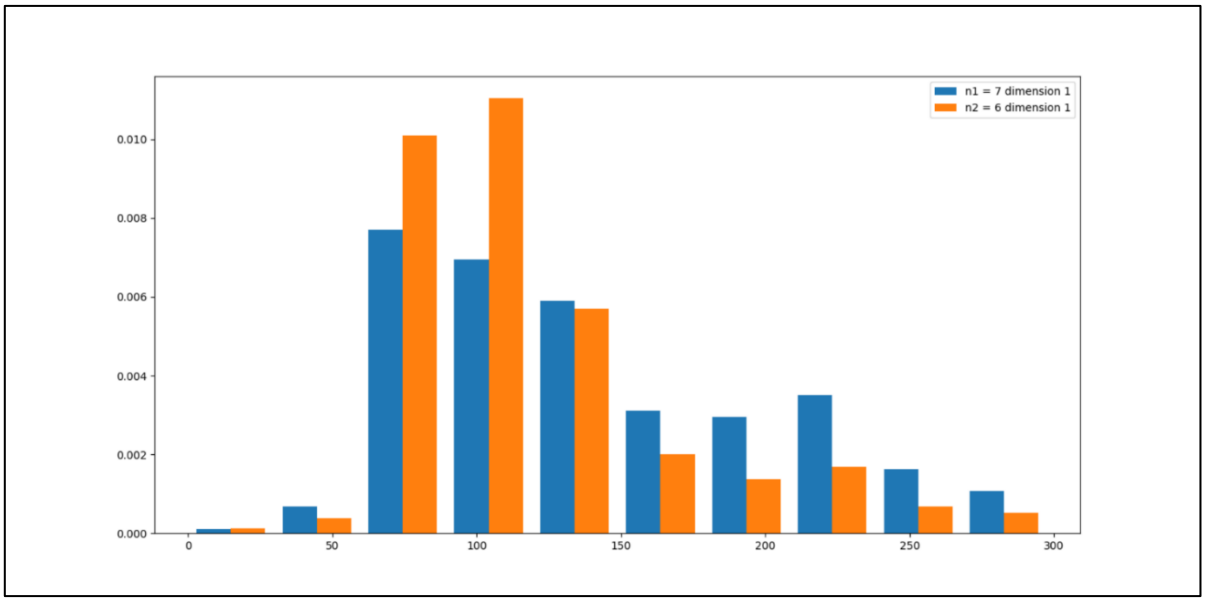
Παρακάτω φαίνεται η αναπαράσταση των ιστογραμμάτων του 1ου και του 2ου MFCC των ψηφίων n1 και n2. Στην προπαρασκευή δημιουργήθηκαν δύο ιστογράμματα, ένα για κάθε dimension των ψηφίων. Για κάθε γράφημα δημιουργήθηκαν 2 λίστες, μία για κάθε ψηφίο, στις οποίες είναι συγκεντρωμένες όλες οι ηχογραφήσεις του ψηφίου διατηρώντας την επιθυμητή διάσταση μέσω των συναρτήσεων make_hist και prep_hist. Τέλος, με plotting οι δύο λίστες εμφανίζονται στο ίδιο γράφημα.

Τα ιστογράμματα που προέκυψαν κατά την εκτέλεση της προπαρασκευής διαφέρουν από εκείνα της τελικής εκτέλεσης αλλά συμπεριλαμβάνονται για πληρότητα και είναι τα εξής:

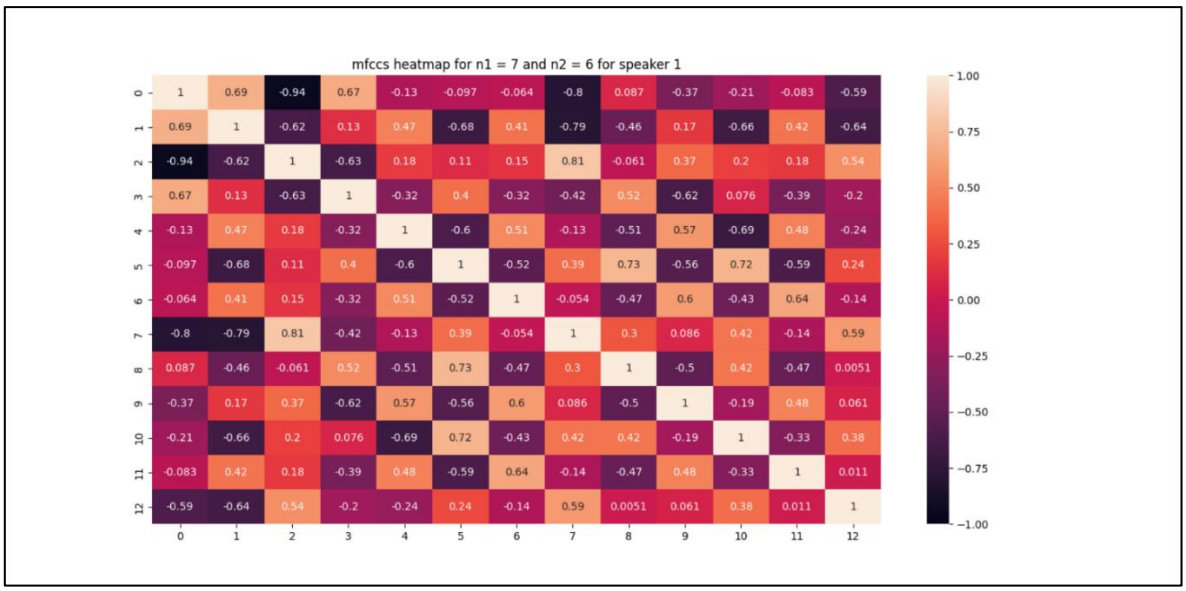


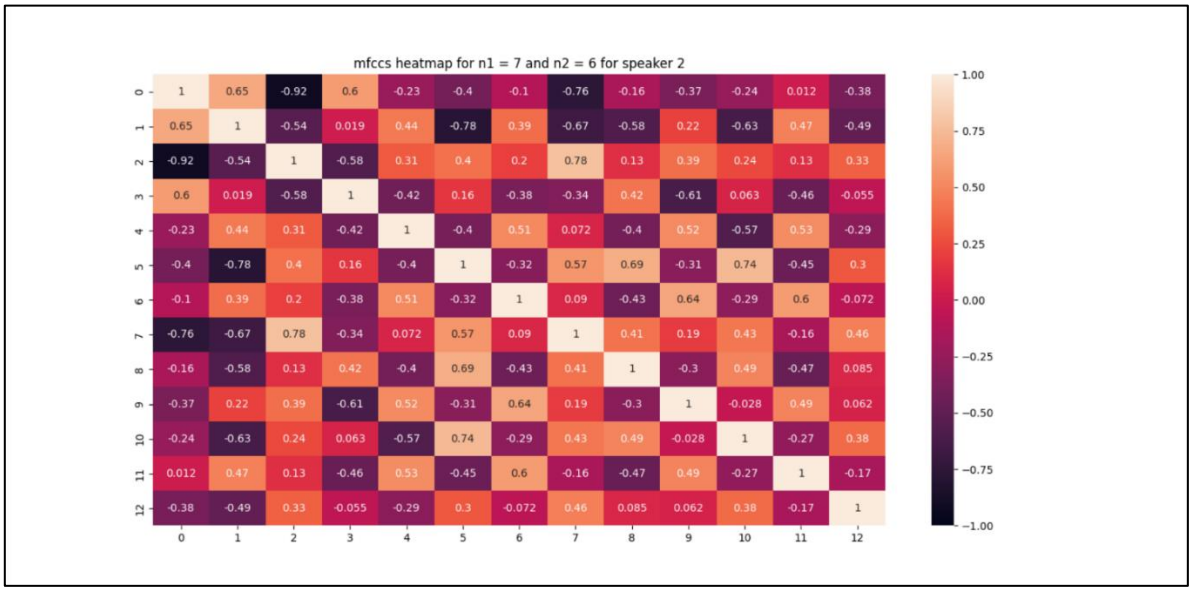

n1 = 7 dimension 0
n2 = 6 dimension 0
0.004
0.003
0.002
0.001
0.000
−700
−600
−500
−400
−300
−200
−100
0


n1 = 7, n2 = 6 all recordings in dimension at index 1
n1 = 7 dimension 1
n2 = 6 dimension 1
400
350
300
250
200
150
100
50
0
0
50
100
150
200
250
300

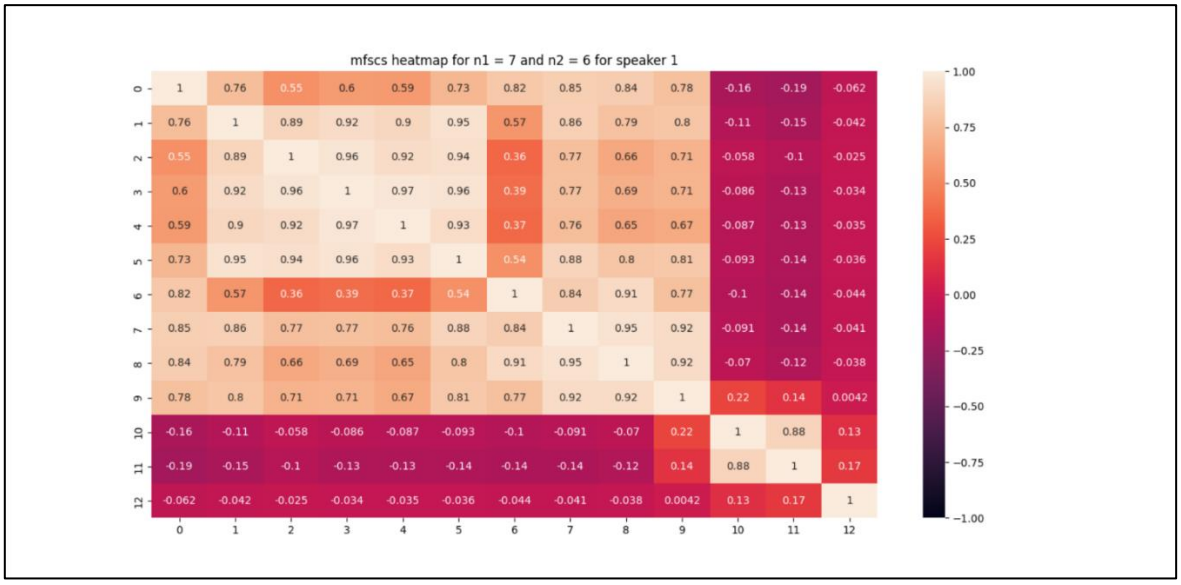

n1 = 7 dimension 1
n2 = 6 dimension 1
0.010
0.008
0.006
0.004
0.002
0.000
0
50
100
150
200
250
300

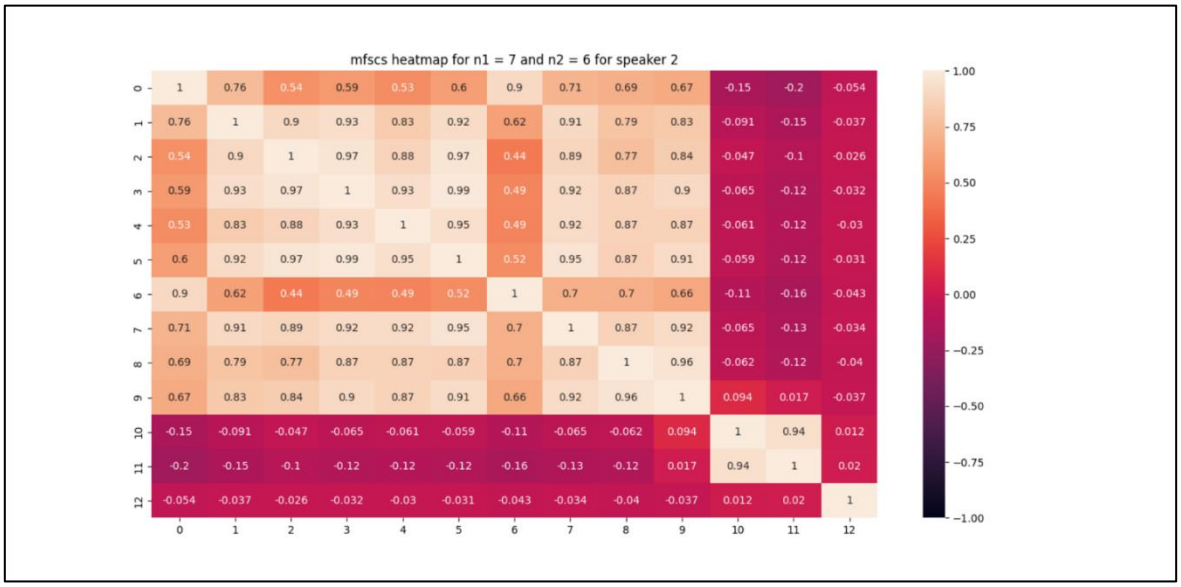
Επιπλέον, οι MFCCs που προέκυψαν και αυτοί κατά την προπαρασκευή είναι οι εξής:





Επίσης, οι MFSCs που προέκυψαν και αυτοί κατά την προπαρασκευή είναι οι εξής:

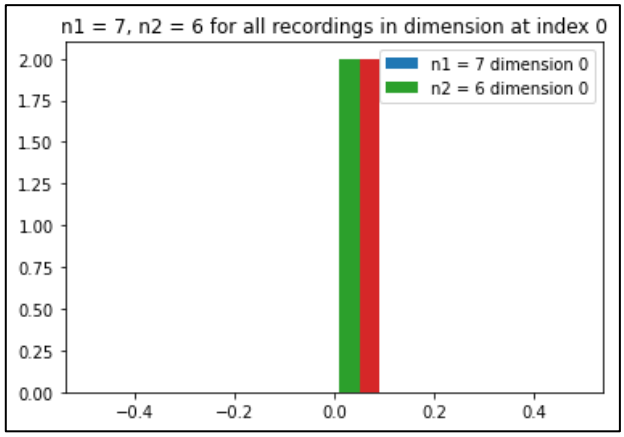


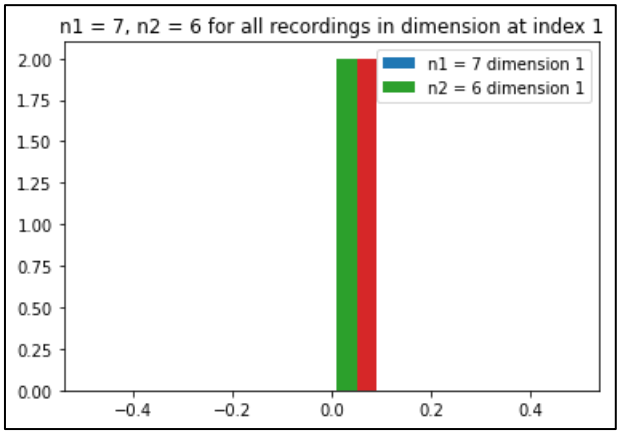


Σχολιασμός:

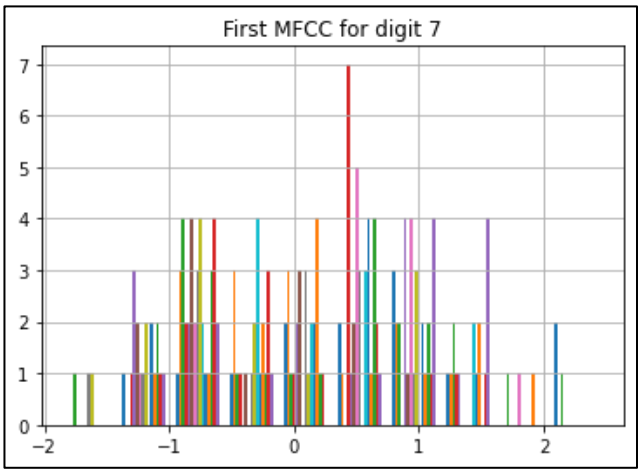
Παρατηρείται ότι για συγκεκριμένες τιμές των MFCCs υπάρχει μεγάλη απόκλιση στις συχνότητες. Επιπλέον, στα χαρακτηριστικά των MFSCs φαίνεται πως έχουν μεγαλύτερη συσχέτιση μεταξύ τους ενώ στα χαρακτηριστικά των MFCCs φαίνεται πως έχουν μεγαλύτερη ανεξαρτησία. Συνεπώς, τα MFCCs θα είναι πιο χρήσιμα για ταξινόμηση ηχητικών κυμάτων εν προκειμένω. Αυτή η ιδιότητα θα είναι ιδιαίτερα χρήσιμη σε αρκετούς ταξινομητές οι οποίοι βασίζονται στην ανεξαρτησία μεταξύ των χαρακτηριστικών.

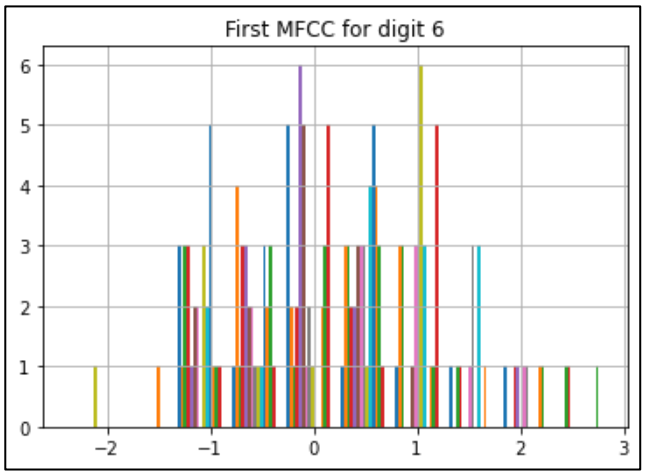
Ωστόσο, κατά την τελική εκτέλεση, στον ίδιο κώδικα προέκυψαν οι εξής γραφικές παραστάσεις:

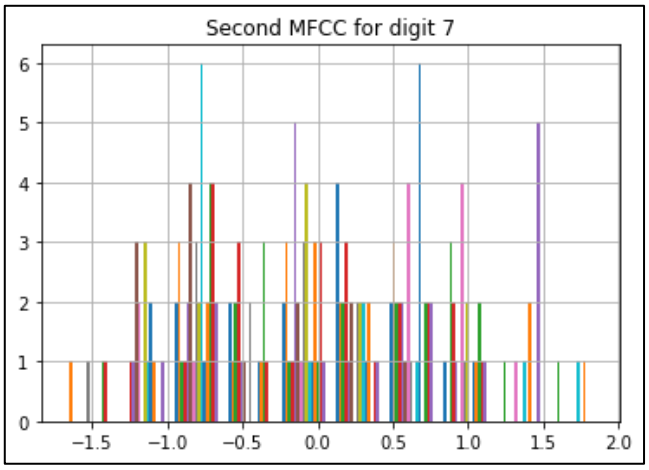


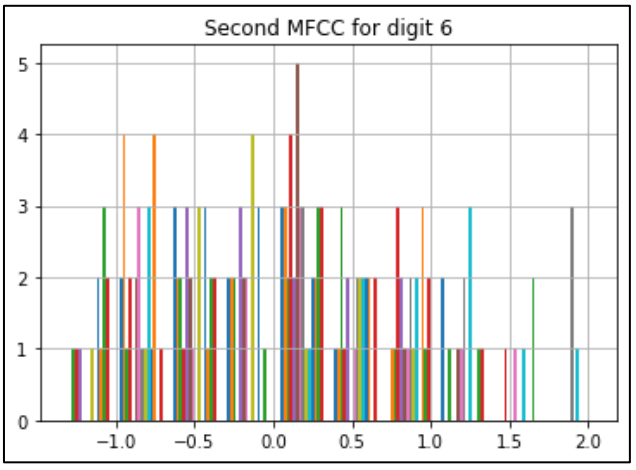


Περισσότερη σημασία όμως έχουν τα ιστογράμματα του 1ου και 2ου MFCC για όλες τις εκφωνήσεις μέσω των συναρτήσεων make_hist και prep_hist.

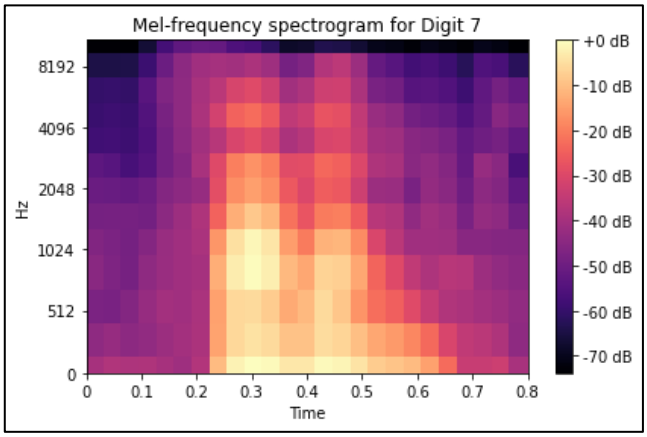


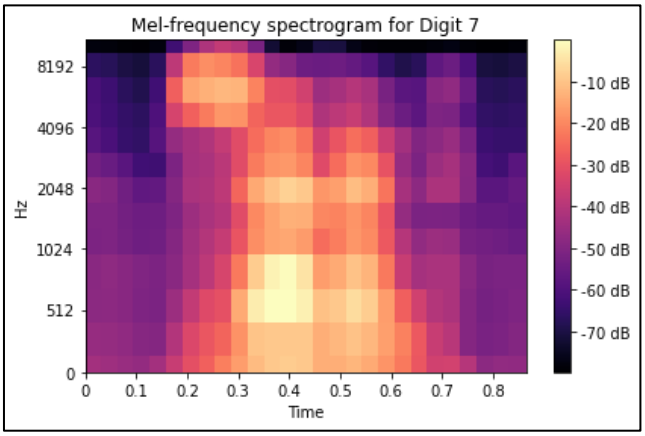


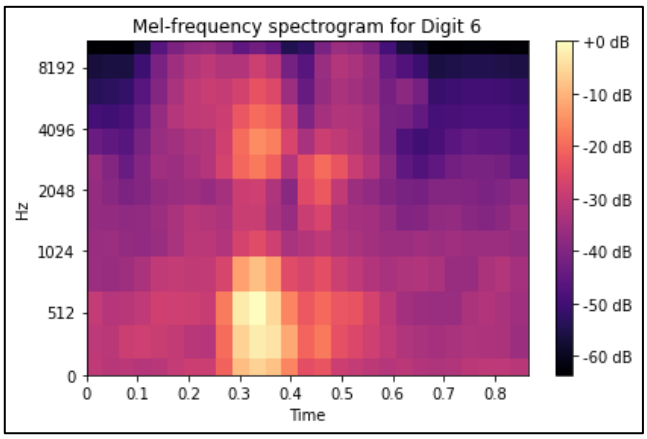


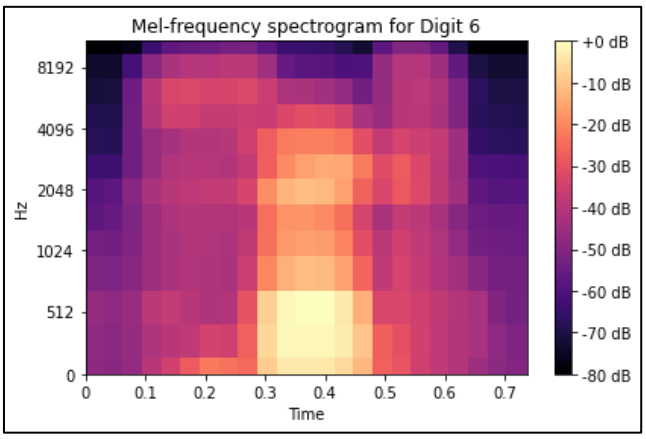


Στην συνέχεια, επιλέγονται οι 2 πρώτες εκφωνήσεις των ψηφίων, εφαρμόζοντας την συνάρτηση melspectrogram εξάγονται τα MFSCs. Με άλλα λόγια, έχει εφαρμοστεί η συστοιχία φίλτρων της κλίμακας Mel πάνω στο φάσμα του σήματος φωνής χωρίς όμως Μ/Σ DCT στο τέλος ώστε να εξαχθούν τα επιθυμητά χαρακτηριστικά.

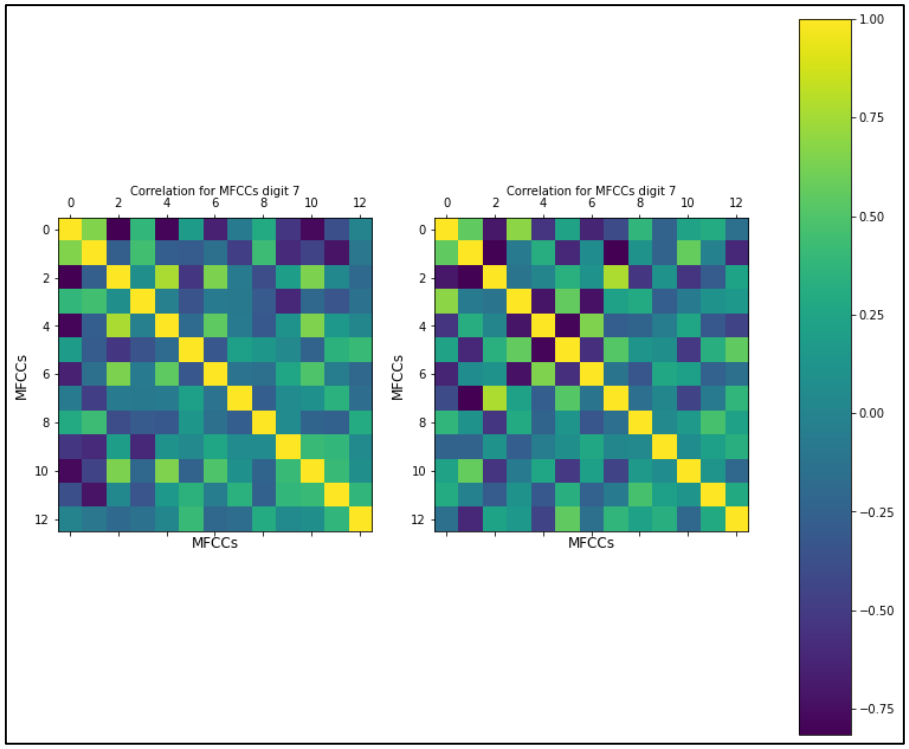


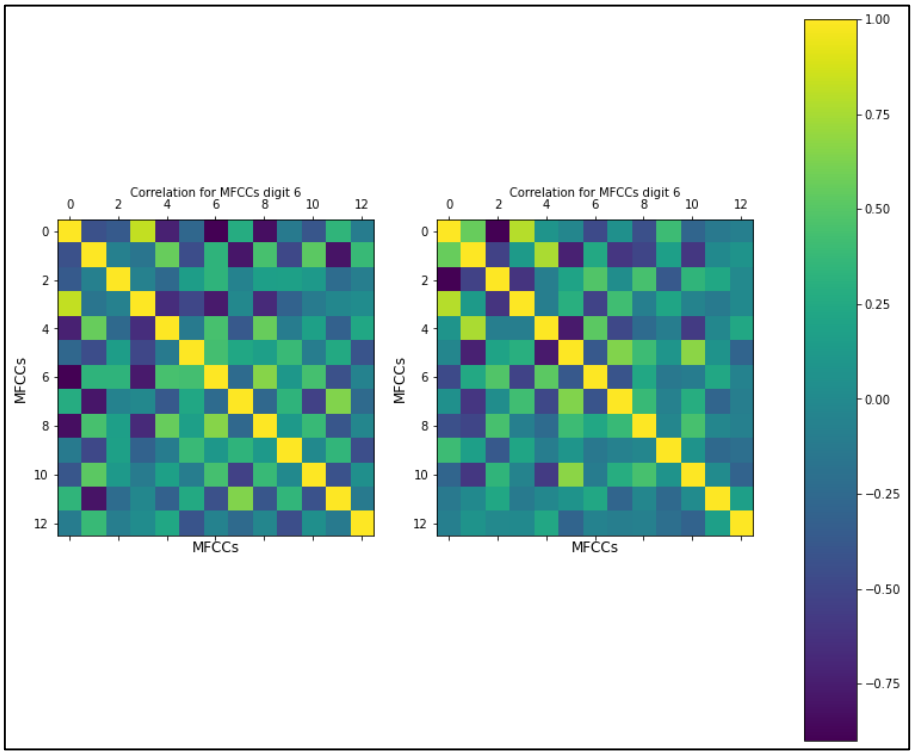






Τέλος, φαίνεται ο υπολογισμός και η απεικόνιση της Correlation (συσχέτισης) για MFCCs.

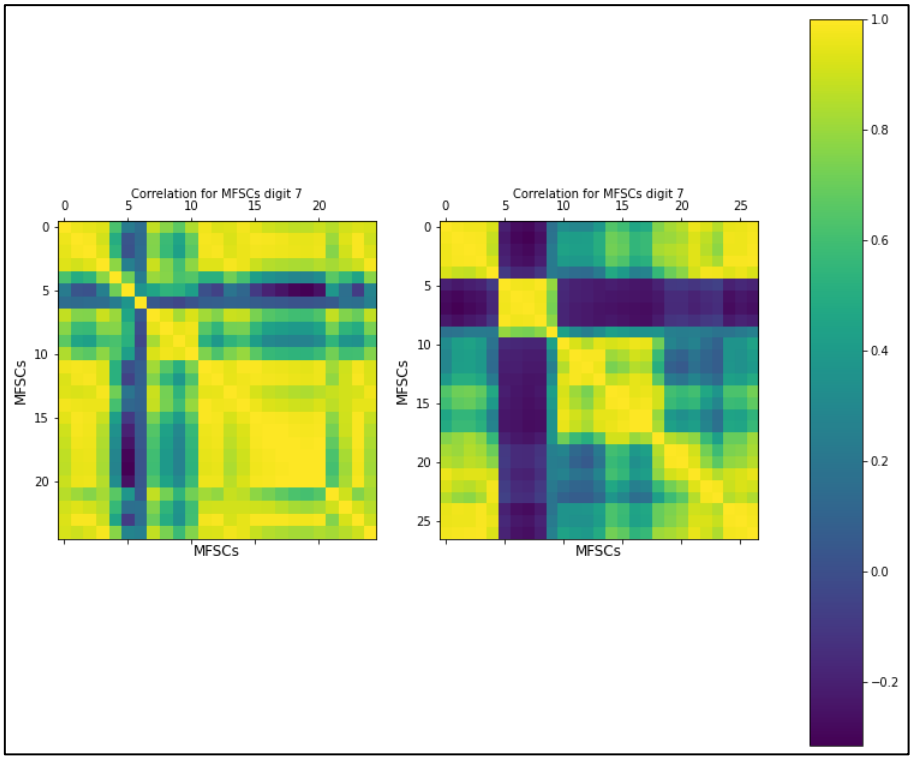


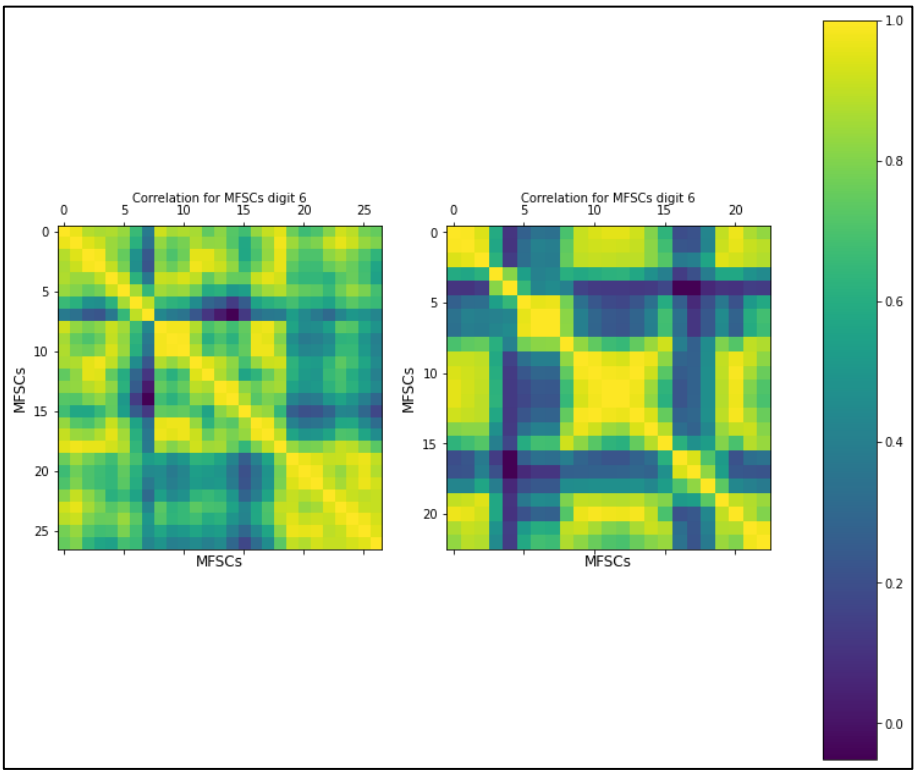


Σχολιασμός:

Παρατηρείται με άλλα λόγια, ότι οι MFCCs φαίνεται να είναι σχεδόν ασυσχέτιστοι. Όπως φαίνεται και στα σχήματα, στην διαγώνιο τα στοιχεία έχουν τιμή 1 που σημαίνει ότι κάθε συντελεστής είναι γραμμικά εξαρτώμενος από τον εαυτό του. Επειδή στον υπόλοιπο πίνακα υπάρχουν χαμηλές έως μηδενικές τιμές συμπεραίνει κανείς ότι γενικά οι συντελεστές MFCC είναι ανεξάρτητοι μεταξύ τους.

Καθώς επίσης, και ο υπολογισμός και η απεικόνιση της Correlation (συσχέτισης) για MFSCs





Σχολιασμός:

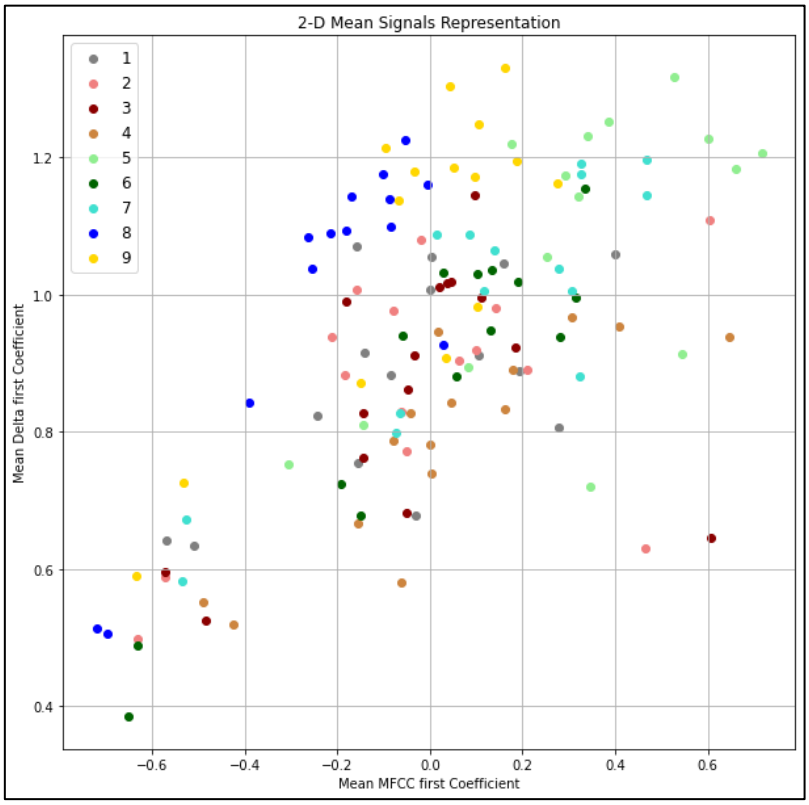
Παρατηρείται με άλλα λόγια, ότι οι MFSCs φαίνεται να είναι αρκετά correlated. Αυτό επιβεβαιώνεται διότι αυτό που χωρίζει τους MFSC από τους MFCC είναι ο μετασχηματισμός DCT που γίνεται ώστε να εξασφαλισθεί η ανεξαρτησία. Για αυτό το λόγο οι MFCCs είναι καλύτεροι ως features στο classification task.

Βήμα 5

Μια πρώτη προσέγγιση για την αναγνώριση των ψηφίων είναι η εξαγωγή ενός μοναδικού διανύσματος χαρακτηριστικών για κάθε εκφώνηση. Ενώστε τα mfccs – deltas – delta-deltas και έπειτα για κάθε εκφώνηση δημιουργείστε ένα διάνυσμα παίρνοντας τη μέση τιμή και την τυπική απόκλιση κάθε χαρακτηριστικού για όλα τα παράθυρα της εκφώνησης. Αναπαραστήστε με scatter plot τις 2 πρώτες διαστάσεις των διανυσμάτων αυτών, χρησιμοποιώντας διαφορετικό χρώμα και σύμβολο για κάθε ψηφίο. Σχολιάστε το διάγραμμα.

Σε αυτό το βήμα γίνεται μια πρώτη προσέγγιση για την αναγνώριση ψηφίων και για αυτό δημιουργείται το διάνυσμα χαρακτηριστικών για κάθε εκφώνηση. Επιπλέον, γίνεται υπολογισμός της mean value και της τυπικής απόκλισης για τα MFCCs, τα deltas και τα delta-deltas σε όλα τα windows της κάθε εκφώνησης και από την ένωση προκύπτει το τελικό διάνυσμα χαρακτηριστικών.

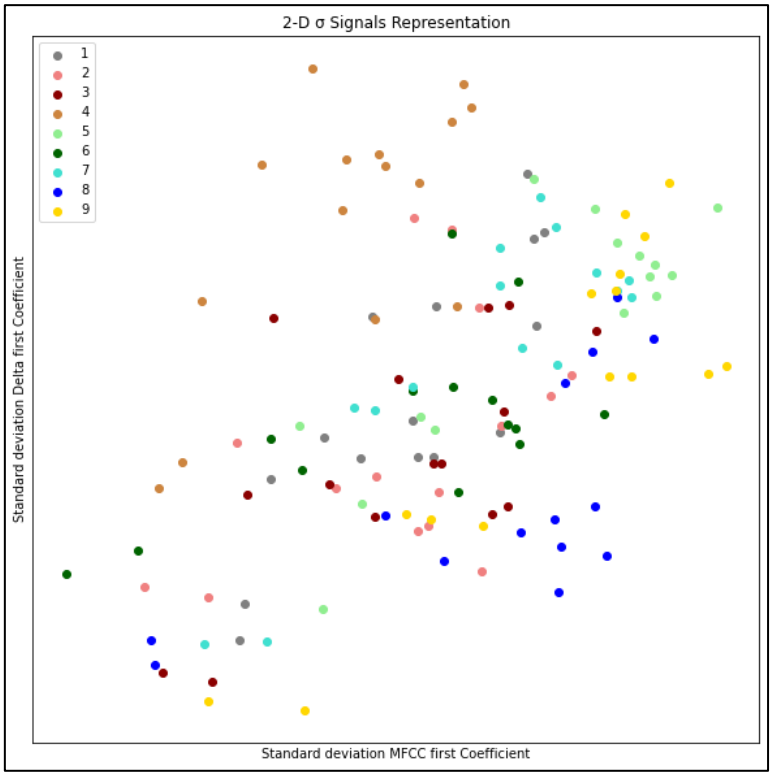
Στο παρακάτω διάγραμμα φαίνεται η αναπαράσταση των πρώτων 2 διαστάσεων των διανυσμάτων (mean of mfccs, mean of deltas), έχοντας επιλέξει διαφορετικό χρώμα για κάθε ψηφίο.



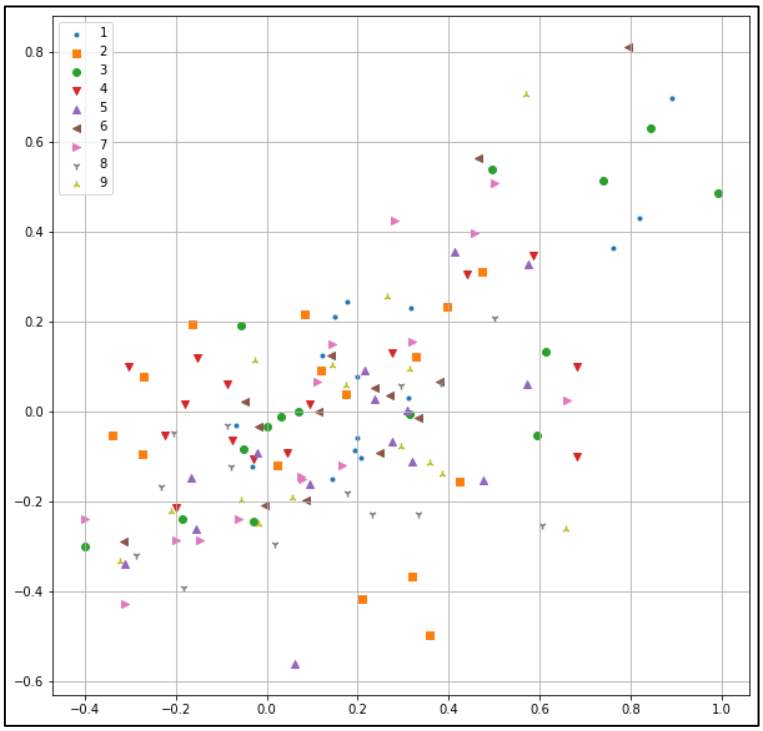
Σχολιασμός:

Παρατηρείται ότι από τις δύο πρώτες διαστάσεις των διανυσμάτων δεν προκύπτουν αρκετά συμπεράσματα, όπως φαίνεται από το γράφημα, οπότε δεν αποτελούν καλό κριτήριο για την κατηγοριοποίηση των εκφωνήσεων. Παρατηρείται ωστόσο ότι δημιουργούνται κάποια clusters των κλάσεων. Έτσι, τα 2 αυτά χαρακτηριστικά θα έδιναν πολύ καλύτερα αποτελέσματα στο classification, αφού έχουν μικρότερη συσχέτιση μεταξύ τους. Με άλλα λόγια, υπάρχουν κάποιες περιοχές όπου υπερισχύουν κάποια ψηφία αλλά δεν είναι εύκολος ο προσδιορισμός.

Επιπλέον, φαίνεται και η τυπική απόκλιση:



Τέλος, φαίνεται και πάλι η μέση τιμή για μια διαφορετική υλοποίηση η οποία είχε γίνει στην προπαρασκευή.



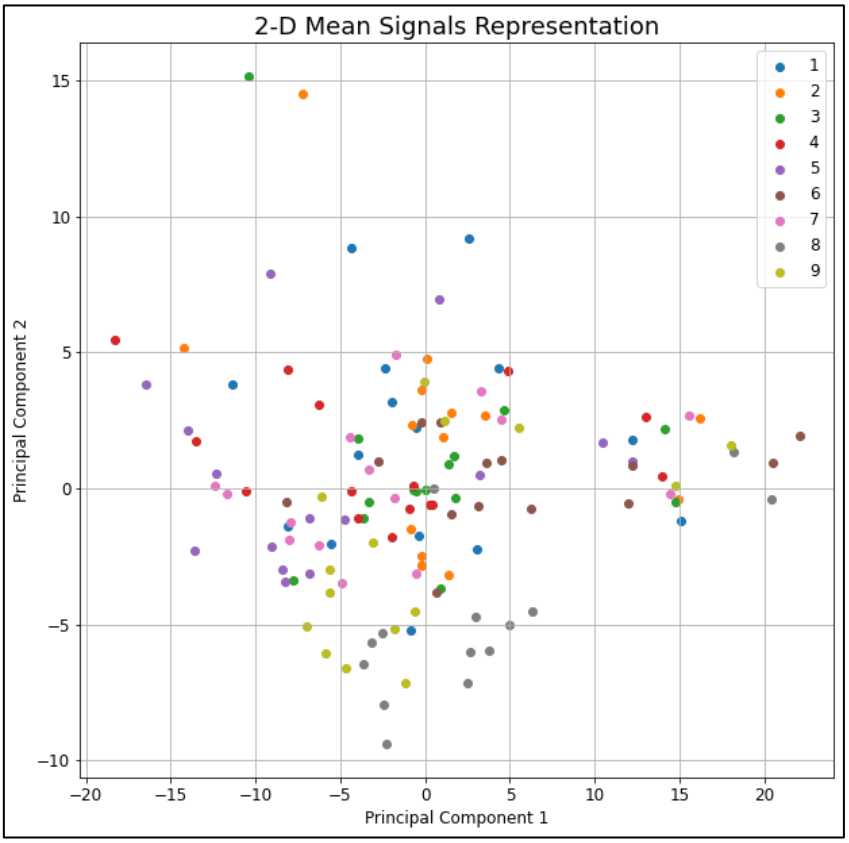
Βήμα 6

Μια καλή τακτική για απεικόνιση πολυδιάστατων διανυσμάτων είναι η μείωση των διαστάσεών τους με Principal Component Analysis (PCA). Μειώστε σε 2 τις διαστάσεις των διανυσμάτων του προηγούμενου βήματος με PCA και δημιουργείστε εκ νέου το scatter plot. Σχολιάστε και επαναλάβετε τη διαδικασία για 3 διαστάσεις και τρισδιάστατο scatter plot. Τι ποσοστό της αρχικής διασποράς διατηρούν οι συνιστώσεις που προέκυψαν? Τι πληροφορία δίνουν αυτά τα νούμερα για τα principal components? Είναι επιτυχημένη η μείωση διαστάσεων?

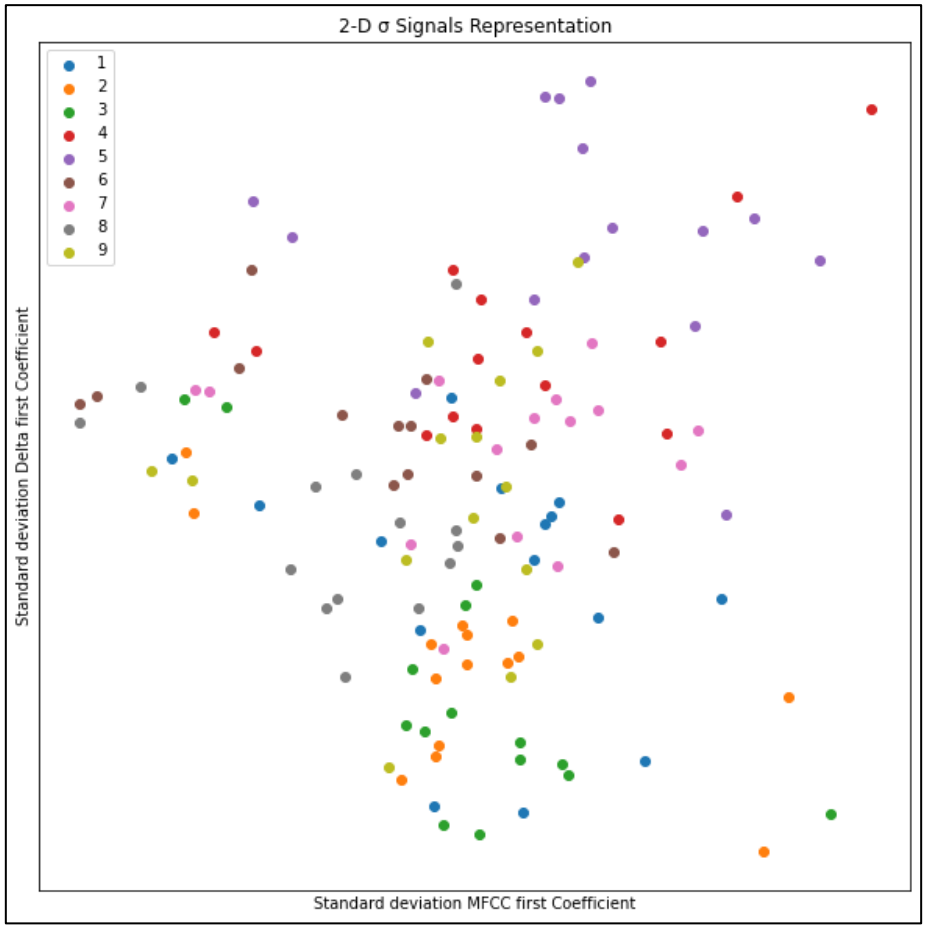
Σε αυτό το βήμα, με PCA εφαρμόζεται μείωση στις διαστάσεις των διανυσμάτων χαρακτηριστικών σε 2 διαστάσεις οπότε η πληροφορία για τις υπόλοιπες διαγράφεται.

Η νέα αναπαράσταση των διανυσμάτων στον νέο χώρο 2 διαστάσεων θα είναι η ακόλουθη.

Για την μέση τιμή:



Για την τυπική απόκλιση:



Σχολιασμός:

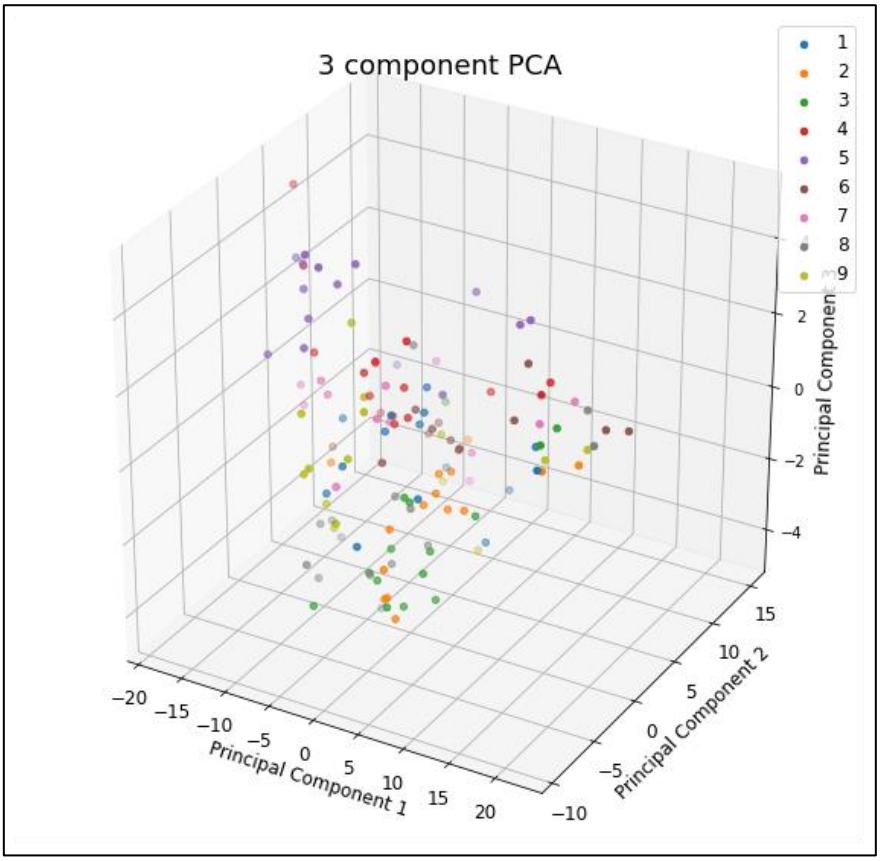
Παρατηρείται πλέον ότι έχει μειωθεί αισθητά η διασπορά τους. Ωστόσο, τα ψηφία είναι και πάλι μη διαχωρίσιμα αλλά διακρίνονται κάποιες περιοχές όπου υπάρχει μεγαλύτερη πλειοψηφία συγκεκριμένων ψηφίων με καλύτερη ευκρίνεια συγκριτικά με το προηγούμενο γράφημα.

Όμως, δεν γίνεται σαφής η αρχική πληροφορία, αλλά σίγουρα οι θέσεις των ψηφίων έχουν αλλάξει ανάλογα με τις κύριες συνιστώσες του PCA, οι οποίες διατηρούν κάποιο ποσοστό της αρχικής διασποράς. Για αυτό το λόγο εφαρμόζεται το ίδιο αλλά για 3 διαστάσεις οπότε προκύπτει το ακόλουθο.

Σημειώνεται ότι οι κύριες συνιστώσες αντιπροσωπεύουν τους κύριους άξονες των δεδομένων και το μήκος των συνιστωσών είναι μια ένδειξη του πόσο σημαντικός είναι αυτός ο άξονας στην περιγραφή της κατανομής των δεδομένων. Αναλυτικότερα, είναι ένα μέτρο της διακύμανσης των δεδομένων όταν προβάλλονται πάνω σε αυτόν τον άξονα. Η προβολή κάθε σημείου δεδομένων στους κύριους άξονες αντικατοπτρίζει τις κύριες συνιστώσες του PCA.

Έπειτα, στις 3 διαστάσεις, οι απεικονίσεις θα είναι οι ακόλουθες:

Για την μέση τιμή ενδεικτικά:



Σχολιασμός:

Κατά την προπαρασκευή, προέκυψε λοιπόν ότι:

Τύπος	**Component 1**	**Component 2**	**Component 3**	**Sum**
PCA-2 mean	62.03353117305860	11.77767265354162		73.8112038266002
PCA-3 mean	62.03353117305860	11.77767265354162	9.015917566828332	82.8271213934285

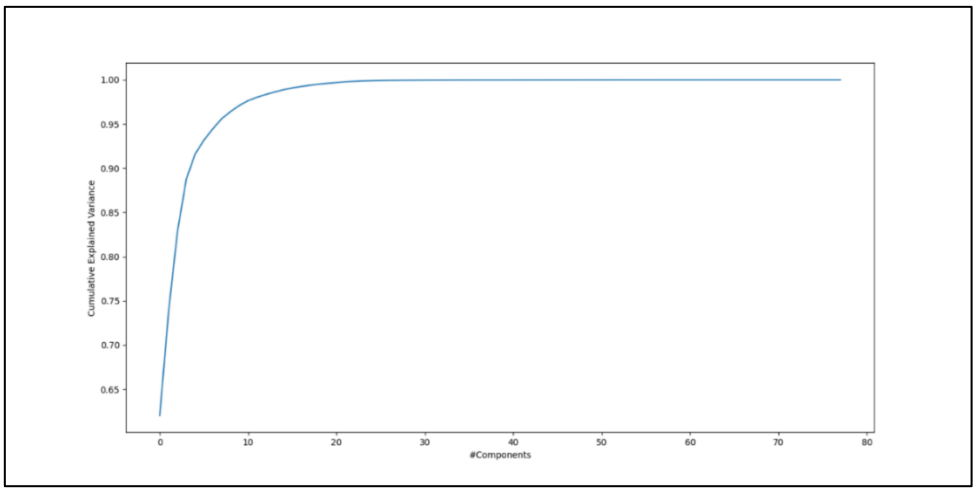
Και κατά την τελική εκτέλεση προέκυψε ότι:

Τύπος	**Component 1**	**Component 2**	**Component 3**	**Sum**
PCA-2 mean	0.7113319	0.16116666		~ 85 %
PCA-2 σ	0.85571554	0.07376451		
PCA-3 mean	0.7113319	0.16116666	0.05227362	~ 90%

Δηλαδή, το ποσοστό της αρχικής διασποράς που διατηρείται είναι περίπου στον PCA 2 ήταν 74% και στον PCA 3 ήταν 83% κατά την προπαρασκευή.

Παρατηρεί λοιπόν κανείς ότι η τρίτη διάσταση που προστέθηκε συνέβαλε κατά 9% στην προπαρασκευή και κατά 5% κατά την τελική εκτέλεση, σε σχέση με την αρχική διασπορά. Έτσι, φαίνεται πως είναι μικρό το ποσοστό σε σχέση με την υπολογιστική πολυπλοκότητα που προσθέτει η 3η διάσταση στο PCA.

Τέλος, σημειώνεται ότι κατά την προπαρασκευή φάνηκε ότι η μείωση διαστάσεων ήταν σχετικά επιτυχημένη ειδικά στις 3 διαστάσεις. Όμως, θα ήταν πλήρως επιτυχημένη αν επιλέγονταν περισσότερες συνιστώσες (περίπου 10) όπως φαίνεται και παρακάτω.



Βήμα 7

Χωρίστε τα δεδομένα σε train-test με αναλογία 70%-30%. Ταξινομήστε με χρήση του Bayesian ταξινομητή της πρώτης εργαστηριακής άσκησης, καθώς και του Naive Bayes του scikit-learn. Χρησιμοποιήστε επίσης, άλλους 3 ταξινομητές της επιλογής σας. Αναφέρετε το ποσοστό επιτυχίας στο test set και συγκρίνετε τα αποτελέσματα. Σημείωση: Τα δεδομένα πριν την ταξινόμηση πρέπει να κανονικοποιηθούν. (Bonus: Θα αυξηθεί το ποσοστό επιτυχίας αν προσθέσω επιπλέον ηχητικά χαρακτηριστικά στο διάνυσμά μου, όπως π.χ. zero-crossing rate? Χρησιμοποιήστε ελεύθερα τέτοια επιπλέον χαρακτηριστικά και εάν αυξηθεί το ποσοστό επιτυχίας αναφέρετε τη σχετική αύξηση, διαφορετικά αναφέρετε τους λόγους που απέτυχε η προσπάθειά σας).

Σε αυτό το στάδιο έγινε μια προσπάθεια ταξινόμησης των δεδομένων χρησιμοποιώντας ταξινομητές από το προηγούμενο εργαστήριο. Δοκιμάστηκαν στην τελική εκτέλεση οι GNB, kNN, MLP, Linear SVM αλλά έχει συμπεριληφθεί και μια ακόμα υλοποίηση από την προπαρασκευή που περιέχει και επιπλέον RFC. Αρχικά, τα δεδομένα χωρίστηκαν σε 70% - 30% σε train set & test set και τα αποτελέσματα ήταν:

Accuracy of Gaussian Naive Bayes: 0.5

Accuracy of k-Nearest Neighbors, k = 1: 0.42500000000000004

Accuracy of 2-Layer Perceptron: 0.675

Accuracy of Linear SVM: 0.6

Και από την προπαρασκευή προέκυψαν τα εξής:

Classification Results for simple features vectors

Accuracy of sklearn NB: 0.5

Accuracy of custom NB: 0.5

Score of 2-Layer Perceptron (MLP) is: 0.5

Score of Random Forest Classifier is: 0.5

Score of SVM classifier: 0.125

Score of KNN (n=2) classifier: 0.575

Accuracy of Gaussian Naive Bayes: 0.5

Accuracy of k-Nearest Neighbors, k = 1: 0.55

Accuracy of 2-Layer Perceptron: 0.825

Accuracy of Linear SVM: 0.725

Και για το (bonus) ερώτημα:

Classification Results with Extended (Zero Crossing Rate) Feature vectors

Accuracy of sklearn NB: 0.525

Accuracy of custom NB: 0.525

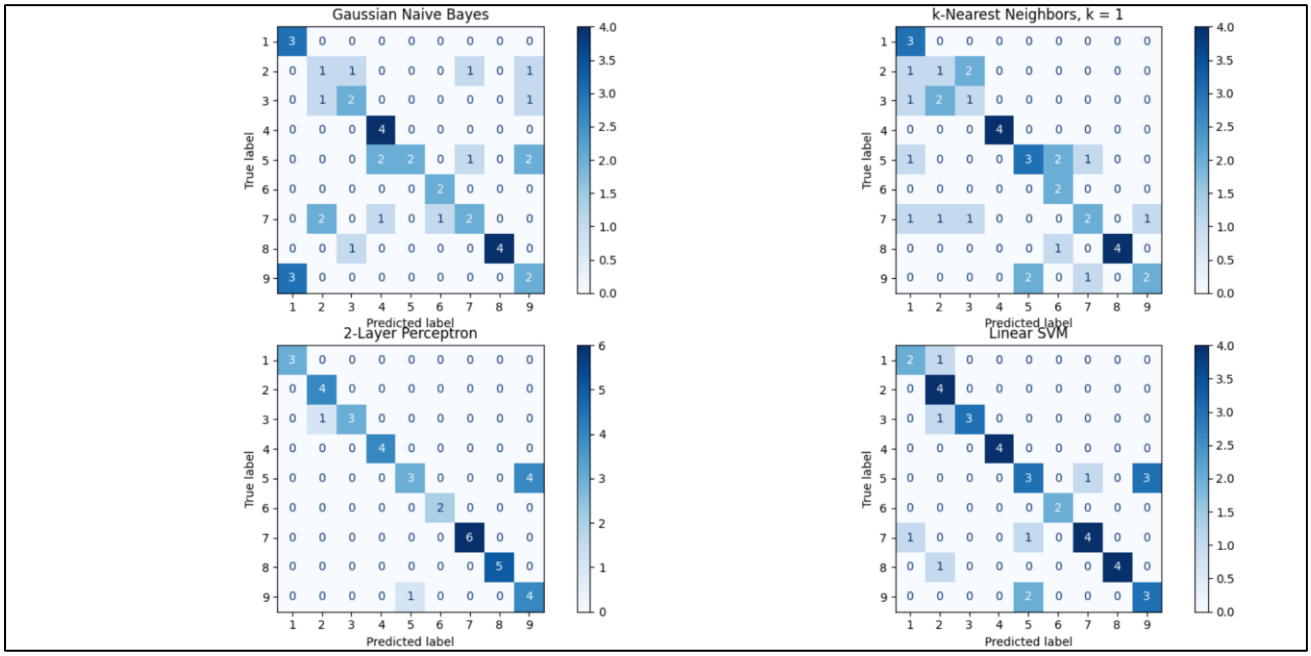
Score of 2-Layer Perceptron (MLP) is: 0.525

Score of Random Forest Classifier is: 0.525

Score of SVM classifier: 0.125

Score of KNN (n=2) classifier: 0.575

Τέλος, προέκυψαν οι ακόλουθες απεικονίσει για μερικούς ταξινομητές:



Σχολιασμός:

Παρατηρείται ότι ο custom GNB, ο GNB της Sklearn και ο RFC έχουν το ίδιο ποσοστό επιτυχίας ταξινόμησης. Ο καλύτερος ταξινομητής φαίνεται να είναι ο KNN (k = 2) ενώ, ο χειρότερος είναι ο SVM με το χαμηλότερο ποσοστό επιτυχίας ταξινόμησης.

(Bonus): Με το Zero Crossing Rate φαίνεται πως οι περισσότεροι ταξινομητές βελτιώνονται εκτός από τον SVM και τον KNN που παραμένουν ίδιοι. Για τον SVM ίσως να οφείλεται το σύνολο δεδομένων με το οποίο τροφοδοτήθηκε καθώς ήταν μεγάλο και imbalanced. Επίσης, η μεγαλύτερη διαστατικότητα των feature vectors φαίνεται πως ευνόησε τον Naive Baye, που χρησιμοποιεί ευκλείδιες αποστάσεις, να τα κάνει disentangle.

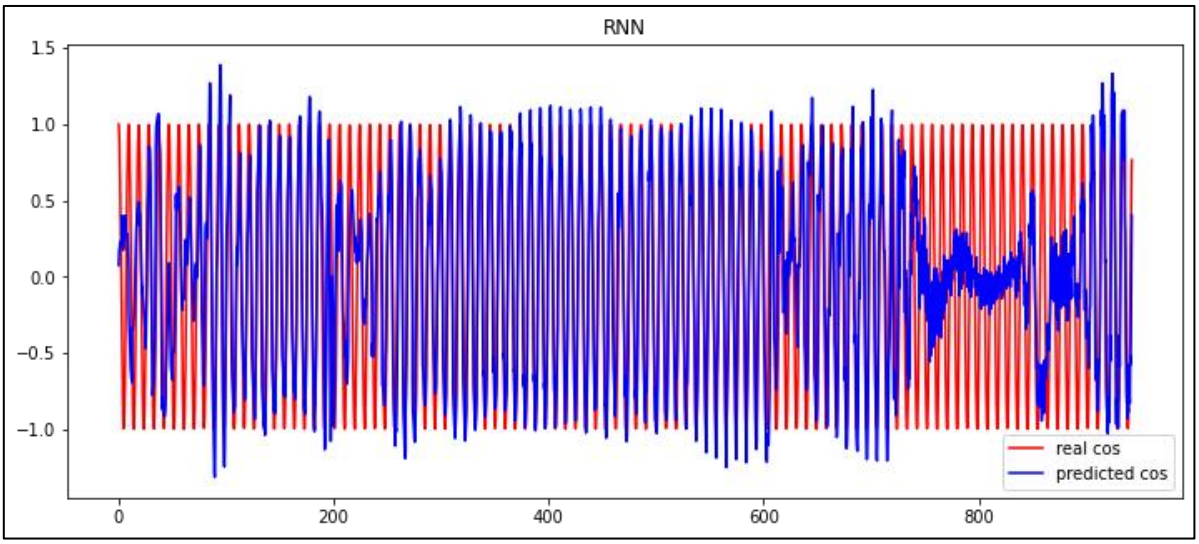
Εν κατακλείδι, κρίνεται απαραίτητη η χρήση μεγαλύτερου και πιο αξιόπιστου πλήθους δεδομένων για την εκπαίδευση για την βέλτιστη απόδοση των ταξινομητών. Ανεξαρτήτως αποτελέσματος, το χαρακτηριστικό που τα παραπάνω συστήματα δεν εκμεταλλεύονται είναι η χρονική αλληλουχία των σημάτων.

Βήμα 8

Εξοικείωση με το PyTorch: Δημιουργείστε ακολουθίες 10 σημείων ενός ημιτόνου και ενός συνημίτονου με συχνότητα f = 40 Hz. Σκοπός είναι η πρόβλεψη του συνημίτονου με δεδομένη την ακολουθία του ημιτόνου. Επιλέξτε σταθερή και μικρή απόσταση ανάμεσα στα διαδοχικά σημεία. Εκπαιδεύστε ένα Αναδρομικό Νευρωνικό Δίκτυο (Recurrent Neural Network – RNN), το οποίο θα δέχεται ως είσοδο τις ακολουθίες του ημιτόνου και θα πρέπει να προβλέπει τις αντίστοιχες ακολουθίες συνημίτονου. Αντί για χρήση του απλού RNN μπορούν να χρησιμοποιηθούν και οι μονάδες LSTM και GRU (δώστε το λόγο που τις χρησιμοποιήσατε και γιατί είναι τόσο διαδεδομένες).

1ος Τρόπος Επίλυσης:

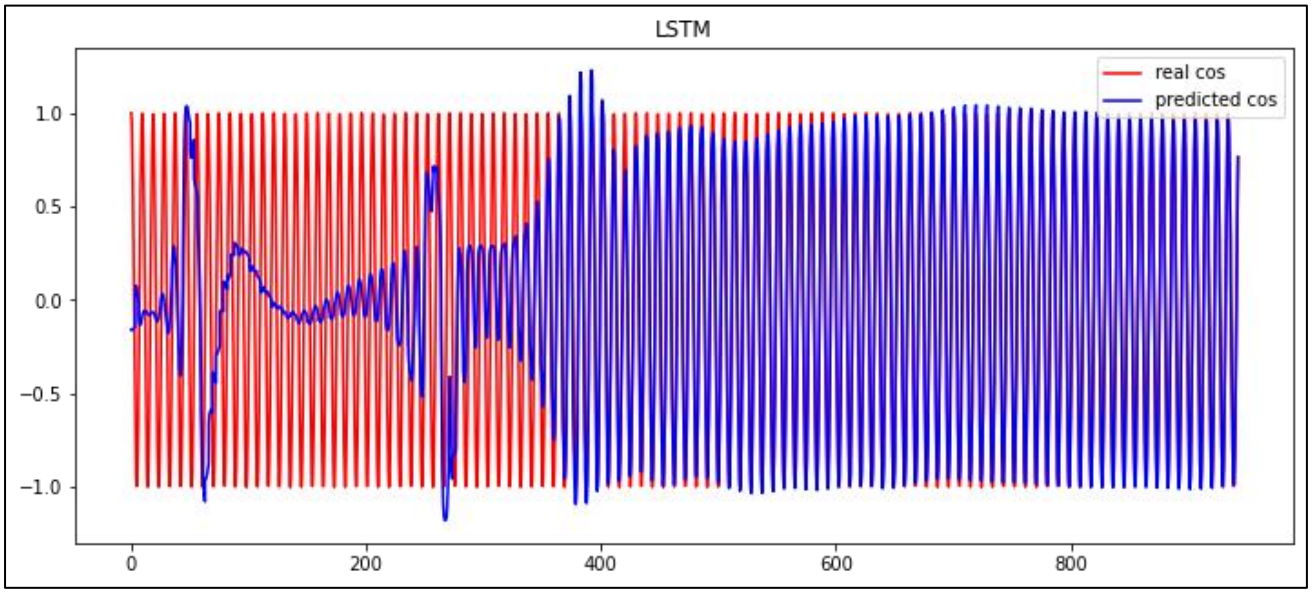
Η εκπαίδευση για την επίλυση προβλημάτων που απαιτούν εκμάθηση μακροχρόνιων χρονικών εξαρτήσεων των τυπικών RNN δεν είναι εύκολη. Πιο συγκεκριμένα, το gradient της loss function μειώνεται εκθετικά με τον χρόνο.



Σχολιασμός:

Παρατηρείται ότι το RNN ομαλοποιείται στο διάστημα (300, 600).

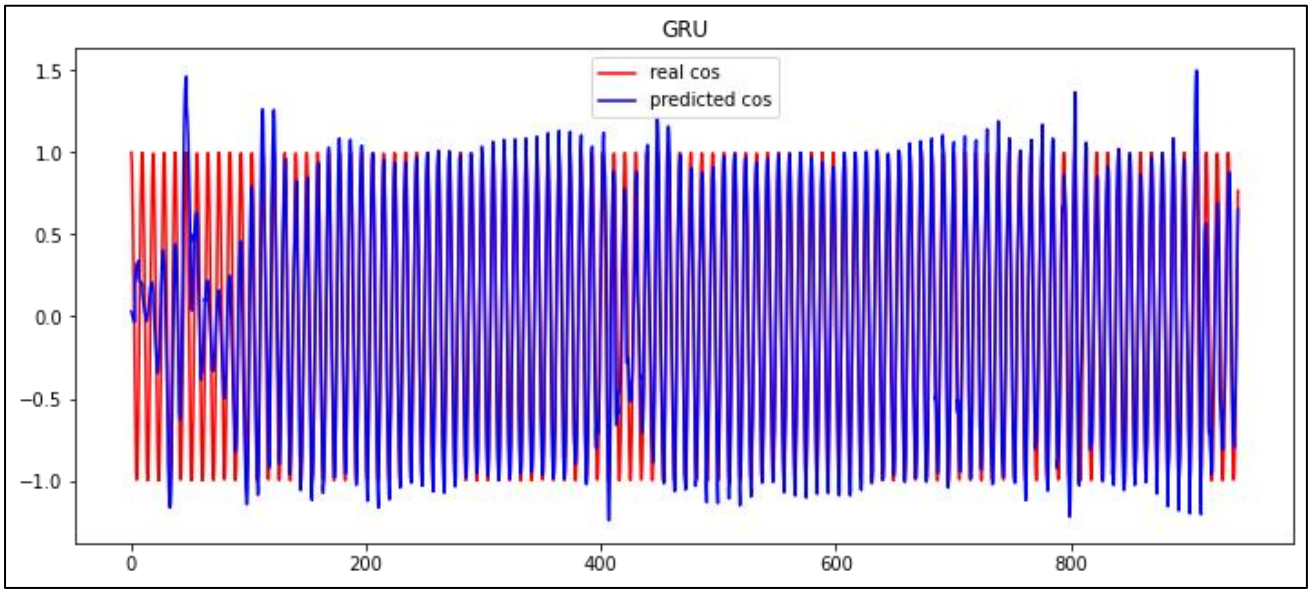
Έπειτα, τα LSTM χρησιμοποιούν ένα σύνολο πυλών για να ελέγχουν πότε οι πληροφορίες εισέρχονται στη μνήμη, πότε εξέρχονται και πότε ξεχνιούνται. Χρησιμοποιεί επιπλέον ειδικές μονάδες από τα RNN, όπως το memory cell, οπότε διατηρεί πληροφορίες για μεγάλα χρονικά διαστήματα.



Σχολιασμός:

Παρατηρείται ότι το LSTM αποδίδει καλύτερα από όλα

Τέλος, τα GRU μοιάζουν με τα LSTM, αλλά σε μια απλοποιημένη δομή. Και αυτά χρησιμοποιούν ένα σύνολο πυλών για τον έλεγχο της ροής πληροφοριών, αλλά λιγότερο σε πλήθος απ' ότι τα LSTM γιατί δεν χρησιμοποιούν ξεχωριστά memory cells.

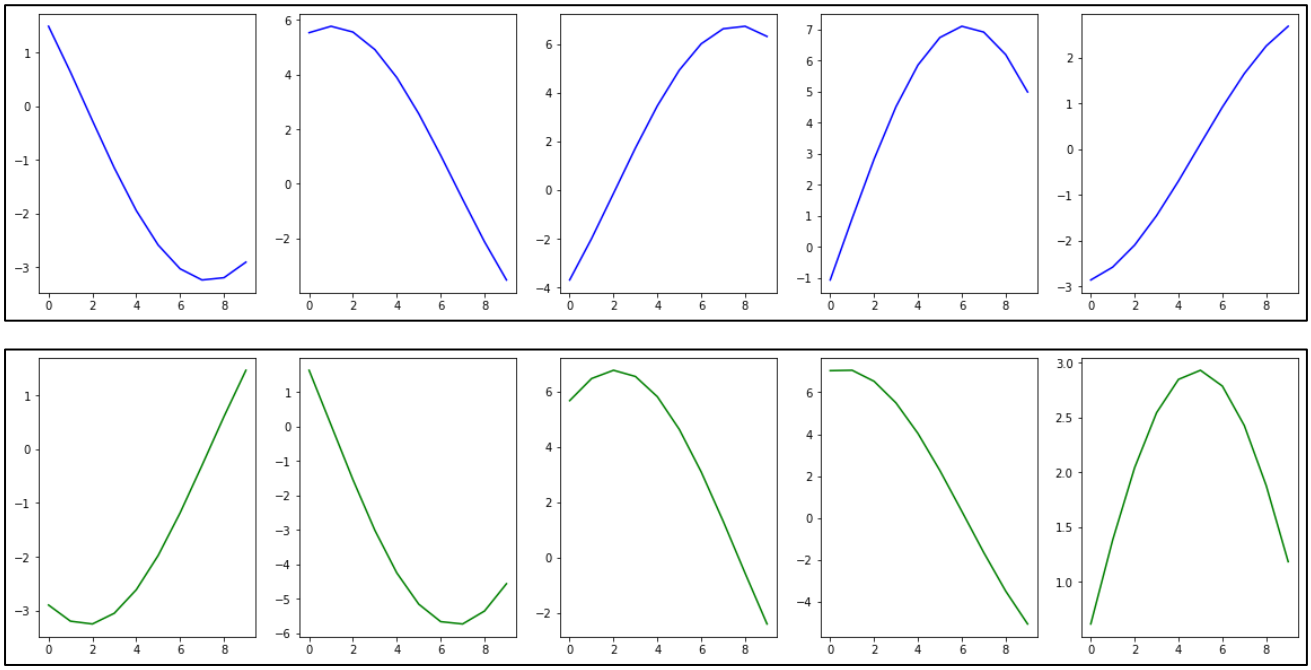


Σχολιασμός:

Παρατηρείται ότι το GRU σε κάποιες περιόδους (400, 800) έχει μεγάλες διακυμάνσεις κι αυτό λόγο της απλότητας που έχει σε σχέση με το LSTM και υπάρχει απώλεια πληροφορίας για αυτό το λόγο.

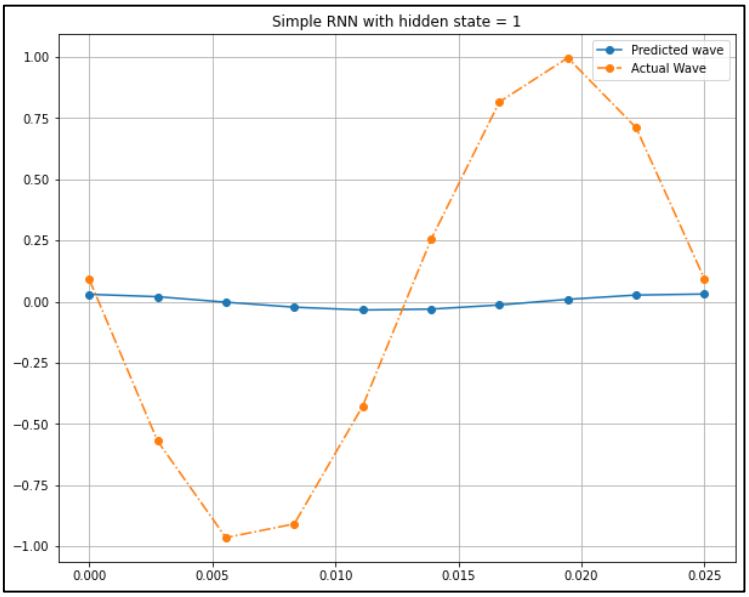
2ος Τρόπος Επίλυσης:

Σε αυτό τον τρόπο επίλυσης, αρχικά παρατίθενται μερικά ζεύγη cos και sin.

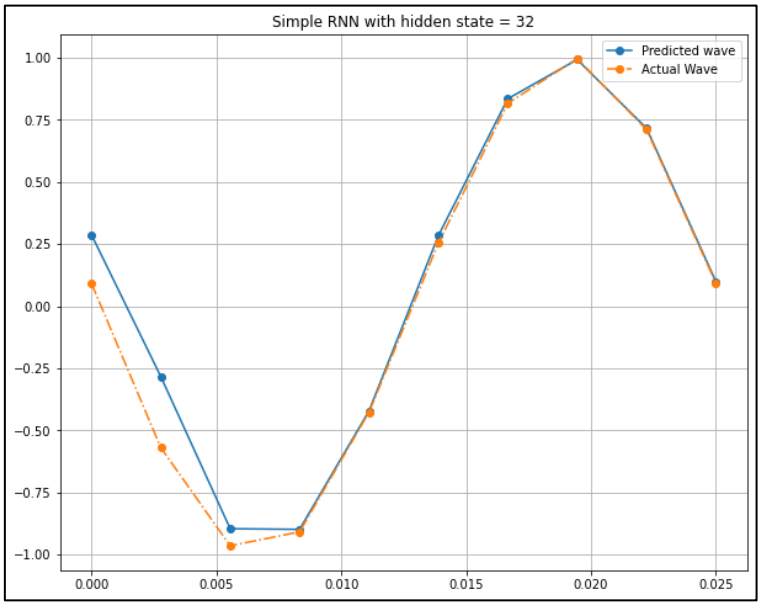


Σε αυτόν τον τρόπο επίλυσης, δημιουργήθηκαν ακολουθίες 10 σημείων από ημίτονα και συνημίτονα και έγινε πρόβλεψη της ακολουθίας του συνημίτονου χρησιμοποιώντας αυτήν του ημιτόνου. Για την δημιουργία ενός δείγματος, επιλέχθηκε μια φάση ομοιόμορφα στο [0, 2π) και για τον χρόνο t, χρησιμοποιήθηκαν 10 σημεία από 0 έως T με step T/10 ενώ το input vector ήταν το sin(ωt + φ) και το target vector ήταν το cos(ωt + φ), όπου f = 40Hz.

Αρχικά, έγινε προσέγγιση ενός RNN με hidden size ίσο με 1 και, όπως ήταν αναμενόμενο, διαφέρει αρκετά όπως φαίνεται και στο predicted signal παρακάτω:



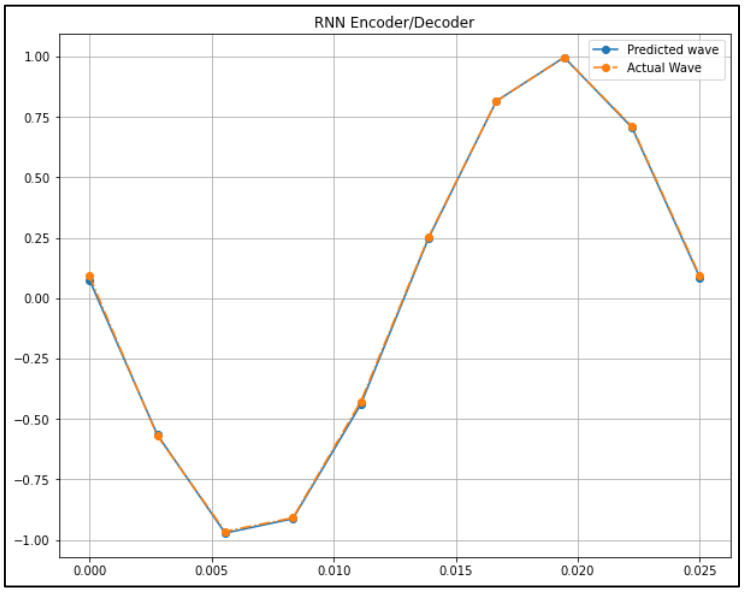
Έπειτα, έγινε μια προσπάθεια να αυξηθούν τα hidden size σε 32 ώστε να προκύψουν καλύτερα αποτελέσματα. Συνεπώς, το output του Νευρωνικού κάθε στιγμή είναι ένα διάνυσμα με 32 διαστάσεις. Σημειώνεται ότι στην έξοδο του RNN προστέθηκε και ένα fully connected layer ώστε να είναι scalar.



Σχολιασμός:

Εδώ φαίνεται πως το RNN κάνει καλύτερη πρόβλεψη από πριν αφού τα σημεία στα οποία υπάρχει error είναι συστηματικά τα πρώτα, το οποίο είναι αναμενόμενο αφού βλέποντας μόνο το πρώτο σημείο της ακολουθίας εισόδου δεν είναι εύκολο να καταλάβει την φάση του ημιτόνου.

Τέλος, για την βελτιστοποίηση της αρχιτεκτονικής έγινε και μια τρίτη προσέγγιση σύμφωνα με την οποία αφαιρείται το σφάλμα στα πρώτα σημεία των ακολουθιών. Έτσι, το RNN παίρνει όλη την ακολουθία και μετά αρχίζει να παράγει έξοδο. Αρχικά η είσοδος περνάει από encoder RNN κρατώντας μόνο το τελικό hidden state και μετά δίνεται το hidden state σε ένα decoder RNN και μετά από 10 βήματα προκύπτει το output.



Τα παρακάτω βήματα δεν αποτελούν μέρος της προπαρασκευής

Για το κυρίως μέρος της άσκησης θα χρησιμοποιηθεί ένα μεγαλύτερο σετ δεδομένων, το Free Spoken Digit Dataset (FSDD), το οποίο μπορείτε να κατεβάσετε από εδώ: https://github.com/Jakobovski/free-spoken-digit-dataset. Στο βοηθητικό υλικό που θα σας δοθεί, υπάρχει υλοποιημένη η διαδικασία διαβάσματος των νέων δεδομένων, εξαγωγής των MFCCs, κανονικοποίησης και χωρισμού σε train και test set (parser.py). Η συνάρτηση αυτή θα πρέπει να πάρει ως όρισμα τη διεύθυνση του φακέλου recordings.

Βήμα 9

Χωρίστε τα train δεδομένα σε training και validation set με ποσοστό 80%-20%. Προσέξτε να διαχωρίσετε με τέτοιο τρόπο τα δεδομένα ώστε να διατηρηθεί ίδιος ο αριθμός των διαφορετικών ψηφίων σε κάθε set (stratified split).

Σε αυτό το βήμα χωρίζονται τα train δεδομένα σε train και validation με αναλογία 80-20 χρησιμοποιώντας το Free Spoken Digit Dataset και τον κώδικα parser.py. Θα πρέπει τα δεδομένα να διαχωριστούν έτσι ώστε να διατηρηθεί ίδιος ο αριθμός των διαφορετικών ψηφίων σε κάθε set.

Βήμα 10

Αναγνώριση ψηφίων με GMM-HMM (Gaussian Mixture Models – Hidden Markov Models).

Για το ερώτημα αυτό θα χρησιμοποιήσετε τη βιβλιοθήκη pomegranate της Python (εγκατάσταση: pip install pomegranate). Αρχικοποιήστε ένα GMM-HMM μοντέλο για κάθε ψηφίο. Το μοντέλο θα πρέπει να 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: $\pi_i = \{0$ για $i \neq 1$ και 1 για $i=1\}$.

Επιπλέον επιτρέπονται μεταβάσεις μόνο μεταξύ διαδοχικών καταστάσεων, δηλαδή υπάρχει ο περιορισμός $a_{ij} = 0$ για $j > i+1$. Ένα διάνυσμα ακουστικών χαρακτηριστικών, όπως αυτό εξάγεται από την επεξεργασία ενός πλαισίου φωνής, αποτελεί μια πιθανή παρατήρηση σε κάποια κατάσταση. Λόγω του ότι είναι επιτρεπτές συνεχείς μεταβολές τέτοιων παρατηρήσεων, η πιθανότητα τους μοντελοποιείται με ένα μίγμα Γκαουσιανών κατανομών (GMM). (Δείτε το βοηθητικό κώδικα (hmm.py) ως πρότυπο για την υλοποίηση ενός τέτοιου μοντέλου).

Σε αυτό το στάδιο αρχικοποιείται ένα GMM-HMM μοντέλο για κάθε ψηφίο χρησιμοποιώντας τον βοηθητικό κώδικα hmm.py. Για κάθε ένα εκ των 10 κλάσεων (μια για κάθε ψηφίο), αρχικοποιείται ένα μοντέλο GMM-HMM της μορφής left-right. Ειδικότερα, ενώνονται τα δεδομένα των εκφωνήσεων, ορίζεται το γεγονός αν θα είναι GMM το μοντέλο (μέσω της βιβλιοθήκης pomegranate). Έπειτα ορίζεται ο πίνακας μεταβάσεων σύμφωνα με τις προδιαγραφές της εκφώνησης. Επιλέγεται με πιθανότητα p η κάθε κατάσταση να μεταβαίνει στον εαυτό της και 1-p να μεταβαίνει στην επόμενη, ενώ αν δεν υπάρχει επόμενη κατάσταση ορίζεται η πιθανότητα ίση με 1 να μεταβεί στον εαυτό της. Οι αρχικές πιθανότητες είναι οι $\pi_i = \{0$ για $i \neq 1, 1$ για $i = 1\}$ και οι τελικές πιθανότητες είναι $\pi_f = \{0$ για $i \neq 1, 1$ για $i = N\}$ όπου N ο αριθμός των καταστάσεων του HMM.

Βήμα 11

Στη φάση αυτή εκπαιδεύονται τα 10 μοντέλα με χρήση του αλγορίθμου Expectation Maximization. Ο αλγόριθμος εφαρμόζεται για καθορισμένο πλήθος επαναλήψεων N_{iter} ή έως να υπάρξει σύγκλιση. Η σύγκλιση ελέγχεται μέσω της μεταβολής του αλγορίθμου της πιθανοφάνειας (Log Likelihood, πιθανότητα των δεδομένων με γνωστό μοντέλο). Για την εκπαίδευση κάθε μοντέλου (που αντιστοιχεί σε κάποιο ψηφίο) χρησιμοποιείστε όλα τα διαθέσιμα δεδομένα για το ψηφίο αυτό. Χρησιμοποιήστε από 1 έως 4 καταστάσεις HMM και από 1 έως 5 Γκαουσιανές κατανομές.

Εδώ, ακολουθείται η εξής διαδικασία.

- Αρχικά, εκπαιδεύονται δέκα GMM-HMM μοντέλα (ένα για κάθε ψηφίο) με τον αλγόριθμο Expectation Maximization (EM).
- Γίνονται δοκιμές για αριθμό καταστάσεων για το HMM από μια έως τέσσερις και αριθμό Gaussian για το GMM από μια έως πέντε.
- Ύστερα, υπολογίζεται το accuracy των μοντέλων με τις παραπάνω παραμέτρους και επιλέγεται το ζεύγος παραμέτρων που δίνει την μεγαλύτερη ακρίβεια.
- Επειδή, χωρίς τα κατάλληλα πακέτα της Python το accuracy είναι μικρότερο από 60%, η εκτέλεση έγινε με τα δοθέντα.
- Τέλος, σημειώνεται ότι η διαδικασία εύρεσης των βέλτιστων παραμέτρων γίνεται μόνο πάνω στο validation set προκειμένου το μοντέλο να γίνει όσο πιο αντικειμενικό γίνεται και να αποφευχθεί το overfitting που θα προέκυπτε αν γινόταν αυτή τη διαδικασία στο test set.

Βήμα 12

Αναγνώριση μεμονωμένων ψηφίων – Testing. Ολοκληρώνοντας τη διαδικασία της εκπαίδευσης, έχετε καταλήξει στις εκτιμήσεις των παραμέτρων των 10 μοντέλων (δηλαδή ένα μοντέλο για κάθε ψηφίο). Στη συνέχεια υπολογίζεται ο λογάριθμος της πιθανοφάνειας (log likelihood) για κάθε εκφώνηση η οποία ανήκει στο σύνολο των δεδομένων για αναγνώριση. Το μοντέλο το οποίο δίνει τη μέγιστη πιθανοφάνεια είναι και το αποτέλεσμα της αναγνώρισης για τη συγκεκριμένη εκφώνηση. Τέλος, για κάθε μοντέλο (ψηφίο) υπολογίζεται το πλήθος των αποτελεσμάτων όπως αυτά κατανέμονται στις διαφορετικές κατηγορίες ψηφίων.

Θα πρέπει να πραγματοποιήσετε αυτή τη διαδικασία αρχικά μόνο στο validation set, μεταβάλλοντας τις παραμέτρους εκπαίδευσης του μοντέλου ώστε να καταλήξετε στο καλύτερο δυνατό αποτέλεσμα. Έπειτα, με τις παραμέτρους του καλύτερου μοντέλου θα αναγνωρίσετε τα ψηφία του test set. Γιατί πραγματοποιούμε αυτή τη διαδικασία? Σε τι μας βοηθάει και εάν δεν την κάνουμε τι κίνδυνος υπάρχει?

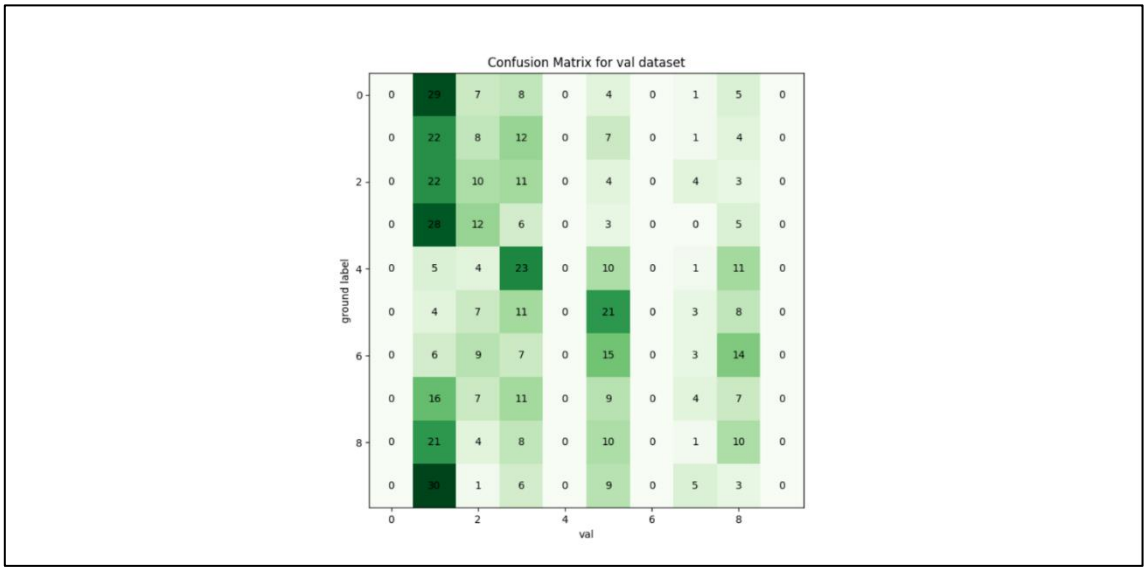
Εδώ, χρησιμοποιείται το validation set για hyperparameter tuning διότι η αξιολόγηση σε δεδομένα εκτός των όσων χρησιμοποιήθηκαν για την εκπαίδευση των παραμέτρων του μοντέλου είναι πιο αντιπροσωπευτική των δυνατοτήτων του. Συγκεκριμένα, τα μοντέλα εκπαιδεύονται έτσι ώστε να μπορούν να αποδίδουν καλά και σε δεδομένα που ανήκουν σε κατανομές εκτός του training set. Στο βήμα αυτό, το prediction για κάθε ψηφίο γίνεται βρίσκοντας ποιο από τα 10 μοντέλα δίνει την μεγαλύτερη Log Likelihood η οποία υπολογίζεται μέσω του αλγόριθμου Viterbi. Με loop πάνω σε όλα τα δεδομένα στο test set θα προκύψουν τελικά τα predictions.

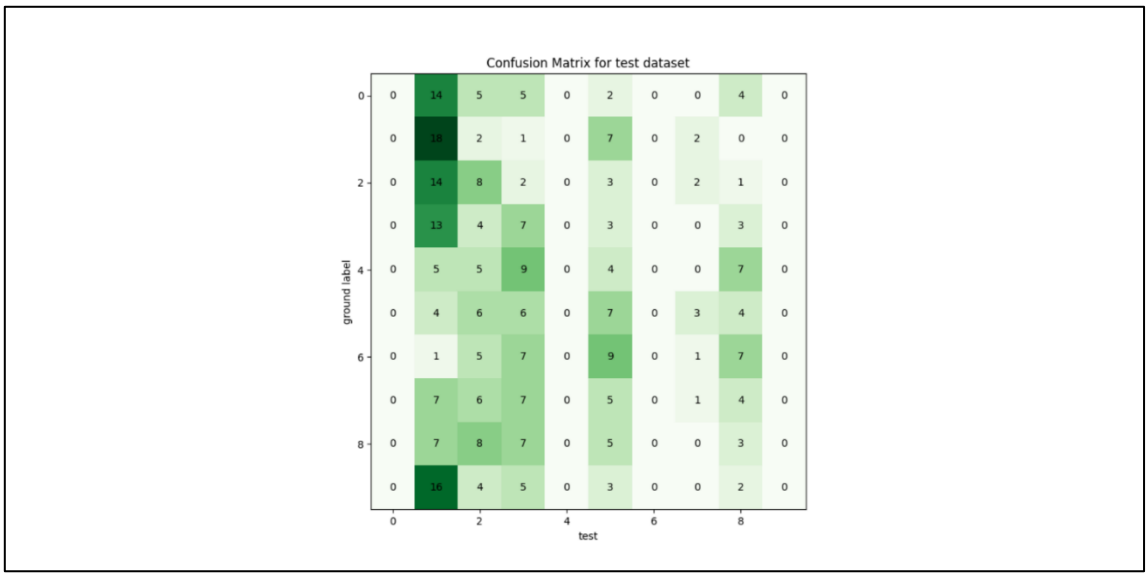
Βήμα 13

Confusion Matrix: Σχηματίστε 2 πίνακες οι οποίοι θα περιέχουν τα αποτελέσματα της διαδικασίας του βήματος 12 για το validation και το test set. Πιο συγκεκριμένα, ο Confusion Matrix πίνακας είναι της μορφής 10×10, και στις γραμμές του περιέχει τα προς ταξινόμηση ψηφία, ενώ στις στήλες του τις κλάσεις στις οποίες αυτά ταξινομήθηκαν.

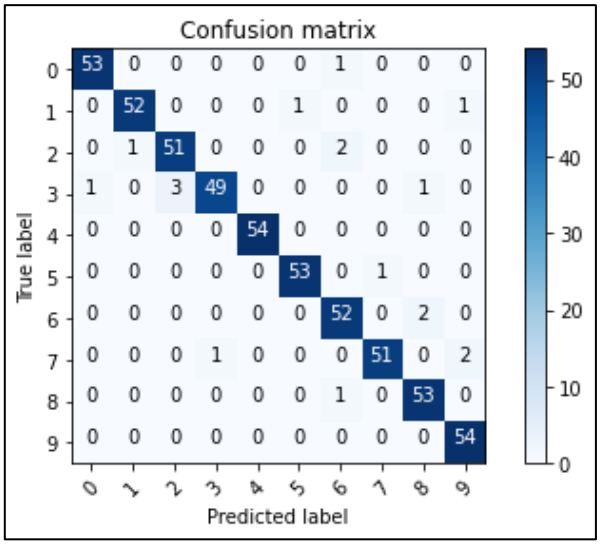
Accuracy: Επίσης, υπολογίστε ένα ολικό ποσοστό αναγνώρισης ως το ποσοστό των σωστά κατηγοριοποιημένων εκφωνήσεων.

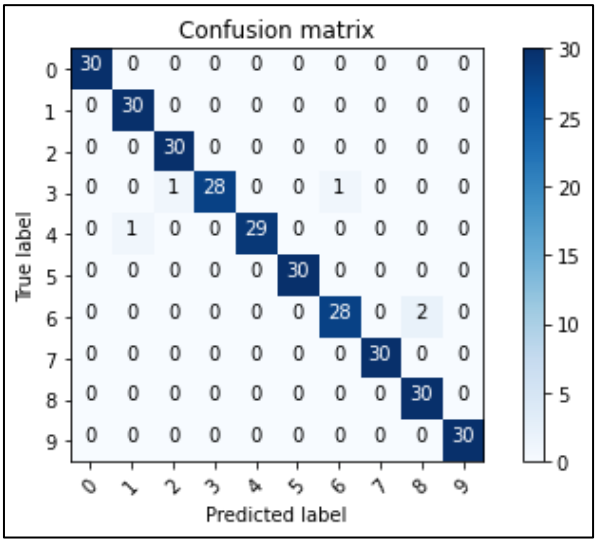
Σε αυτό το προτελευταίο βήμα, στην προπαρασκευή υπολογίστηκαν τα predictions για το validation και το test set μέσω του βοηθητικού κώδικα plot_confusion_matrix.py και της συνάρτησης confusion_matrix του scikit-learn. Τελικά βγήκαν οι ακόλουθοι πίνακες:





Κατά την τελική εκτέλεση προέκυψαν οι εξής:





	precision	**recall**	**f1-score**	**support**
0	1.00	1.00	1.00	30
1	0.97	1.00	0.98	30
2	0.97	1.00	0.98	30
3	1.00	0.93	0.97	30
4	1.00	0.97	0.98	30
5	1.00	1.00	1.00	30
6	0.97	0.93	0.95	30
7	1.00	1.00	1.00	30
8	0.94	1.00	0.97	30
9	1.00	1.00	1.00	30
accuracy			0.98	300
macro avg	0.98	0.98	0.98	300
weighted avg	0.98	0.98	0.98	300

Βήμα 14

Εκπαιδεύστε ένα Αναδρομικό Νευρωνικό Δίκτυο πάνω στο training set, χρησιμοποιώντας το validation set για τη ρύθμιση των υπερπαραμέτρων.

1. *Χρησιμοποιήστε το βοηθητικό κώδικα (lstm.py) για την υλοποίηση του δικτύου.*

2. *Αρχικά υλοποιήστε ένα απλό LSTM δίκτυο.*

3. *Εκπαιδεύστε το δίκτυο, τυπώνοντας μόνο το training loss σε κάθε εποχή.*

4. *Εκπαιδεύστε το δίκτυο, τυπώνοντας το training loss και σε κάθε εποχή κάντε αποτίμηση του μοντέλου στο validation set και τυπώστε το validation loss.*

5. *Προσθέστε στο μοντέλο σας Dropout και L2 Regularization. Εξηγήστε τι κάνει το καθένα και σε τι βοηθάνε κατά την εκπαίδευση.*

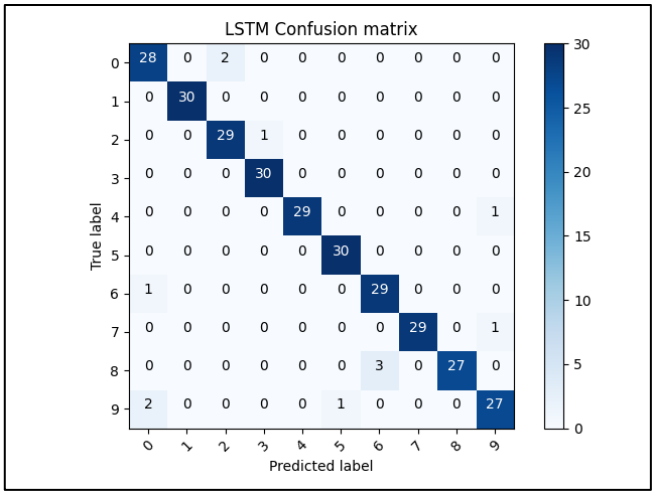
6. *Υλοποιήστε Early Stopping και Checkpoints (δηλαδή αποθήκευση του καλύτερου μοντέλου σε pickle). Εξηγήστε γιατί είναι σωστό να χρησιμοποιούμε Early Stopping κατά την εκπαίδευση.*

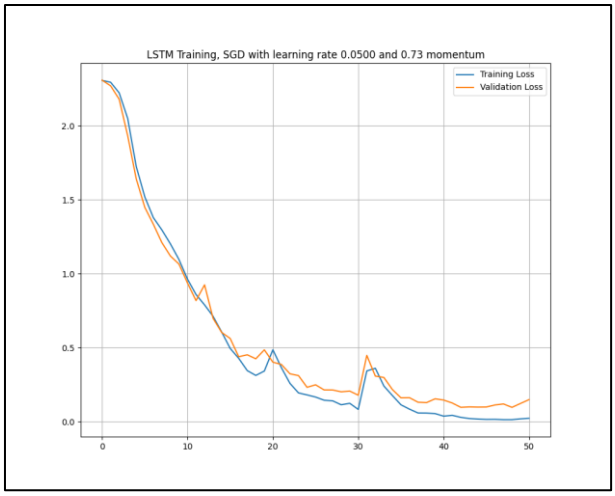
7. *Εκπαιδεύστε ένα Bidirectional LSTM. Τι αλλάζει εσωτερικά στο δίκτυο και τι μας προσδίδει αυτή η αλλαγή?*

8. *(Bonus) Χρησιμοποιήστε το pack_padded_sequence στην εκπαίδευση του μοντέλου. Θα χρειαστεί να ταξινομήσετε τις ακολουθίες που δίνετε στη συνάρτηση αυτή του PyTorch κατά φθίνουσα σειρά μήκους (μην ξεχάσετε να κρατήσετε και τα αντίστοιχα labels όταν γίνει η ταξινόμηση). Συγκρίνετε την ταχύτητα εκπαίδευσης του δικτύου με πριν.*

Παρουσιάστε το ποσοστό επιτυχίας του καλύτερου μοντέλου στο validation και στο test set, καθώς και το confusion matrix. Επίσης, δώστε το διάγραμμα του σφάλματος στο training set για κάθε εποχή εκπαίδευσης και στο ίδιο διάγραμμα με διαφορετικό χρώμα επίσης σχεδιάστε το σφάλμα στο validation set για κάθε εποχή.

Τέλος, χρησιμοποιώντας τον βοηθητικό κώδικα υλοποιήθηκε ένα LSTM με ένα fully connected layer στην έξοδό του. Έτσι, η εκπαίδευση του νευρωνικού έγινε πάνω στο training set για 50 epochs χρησιμοποιώντας Stochastic Gradient descent. Το καλύτερο μοντέλο πετυχαίνει 97% στο test set.





Με άλλα λόγια, τα Βήματα 9 έως και 14 μπορούν να συνοψιστούν ως εξής:

Το καλύτερο μοντέλο GMM-HMM που βρέθηκε με βάση το tuning είναι για 50 επαναλήψεις με 1 κατάσταση HMM και 5 Γκαουσιανές κατανομές. Το tuning πρέπει να πραγματοποιείται στο validation set και όχι εξ ολοκλήρου στο training διότι έτσι αποφεύγεται το overfit στα train data.

Το test set χρησιμοποιείται παρ' όλο που έχει γίνει μια καλή αξιολόγηση πάνω στο validation set λόγω των επαναληπτικών αξιολογήσεων με αλλαγές παραμέτρων στο μοντέλο πάνω στο validation set. Συνεπώς, η επιλογή των κατάλληλων παραμέτρων έγινε με βάση το validation set οπότε είναι λογικό να έχει γίνει προσαρμογή στο validation set.

Η έγκυρη αξιολόγηση του μοντέλου προκύπτει μόνο εάν εφαρμοστεί σε δεδομένα τα οποία δεν έχει επεξεργαστεί ποτέ.

Το accuracy score στην προπαρασκευή ήταν 91.6% στο validation set και το accuracy score ήταν 91.3% στο test set.

Το accuracy score και για validation και για test set ήταν 91.5%.

Επομένως, από τους πίνακες παραπάνω προκύπτει ότι η ταξινόμηση των ηχητικών ψηφίων ήταν πολύ καλή (με μια πολύ μικρή σύγχυση μεταξύ του 8 – 6, 2 – 9 και 3 – 7).

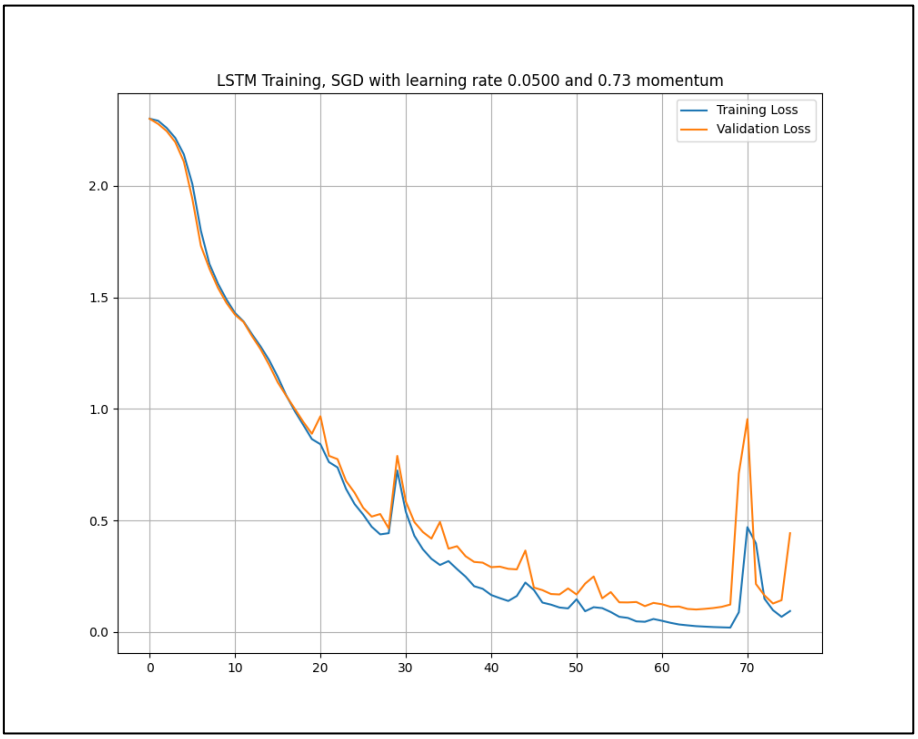
Dropout: είναι ένας διαφορετικός τρόπος για να προσπαθήσουμε να αποφύγουμε το overfitting. Ουσιαστικά αγνοούνται νευρώνες του δικτύου με βάση μια πιθανότητας p. Δηλαδή σε κάθε φάση του training μεμονωμένοι νευρώνες ενός δικτύου είτε διατηρούνται σε αυτό με πιθανότητα p είτε αποκόπτονται από αυτό με πιθανότητα 1-p. Αρκετές φορές, διαδοχικά layers μπορεί να καταλήξουν να αλληλοσυνεργάζονται με περίπλοκο τρόπο, διορθώνοντας λάθη προηγούμενων επιπέδων βασισμένα στις τιμές που εμφανόζονται στα samples του training set, αυτό όμως πιθανότατα δεν θα κάνει generalization

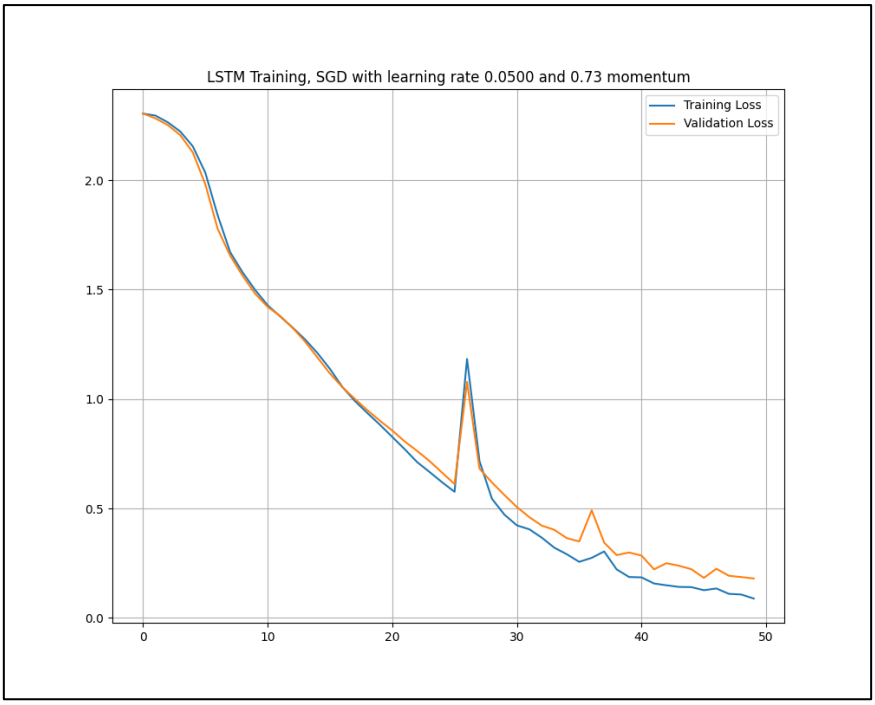
L2 Regularization: στην loss function προστέθηκε ένας όρο ποινής στα βάρη του μοντέλου ανάλογος του τετραγώνου του εύρους των βαρών ώστε σε κάθε επανάληψη οι τιμές των βαρών να μειώνονται αναλογικά με την προηγούμενη τιμή που είχαν και έτσι τελικά αποφεύγεται το overfitting και παράλληλα απλοποιείται το μοντέλο διατηρώντας τα βάρη που δίνουν καλύτερα αποτελέσματα. Ωστόσο, αν ο βαθμός ομαλοποίησης λ είναι πολύ μεγάλος τότε μπορεί να συμβεί underfit. Το L2 Regularization κάνει penalize τα μεγάλα βάρη και ωθεί το νευρωνικό στο να πετύχει χαμηλό loss στο training set αλλά με μικρά βάρη. Η τεχνική αυτή χρησιμοποιείται για την αποφυγή του overfitting στο training set καθώς στην πράξη πολλές φορές παρατηρούνται περιπτώσεις όπου οι ταξινομητές έχουν καταλήξει σε πολύ μεγάλες τιμές παραμέτρων πετυχαίνοντας πολύ μικρό loss στο training set αλλά δεν μπορούν να κάνουν generalization.

Early stopping: κατά την εκπαίδευση μπορεί να βλέπουμε το loss του training set να μειώνεται, ωστόσο αυτό μπορεί να σημαίνει ότι η αρχιτεκτονική μας κάνει overfit στο training set και στην πραγματικότητα χάνει την ικανότητα της να γενικεύσει σε κάθε εποχή υπολογίζεται το μέσο loss του validation set και όταν θα είναι το μικρότερο μέχρι εκείνη την εποχή τότε αποθηκεύεται το μοντέλο. Μετά, με ένα patience εάν δεν παρατηρηθεί βελτίωση στο validation set για 5 εποχές τότε σταματά η διαδικασία της εκπαίδευσης του μοντέλου έχοντας αποθηκεύσει το καλύτερο μοντέλο. Στην εποχή 45 και μετά, το training loss συνεχίζει να μειώνεται ενώ το validation loss αρχίζει να αυξάνεται. Για τον λόγο αυτό, είναι μια καλή πρακτική να σταματάμε νωρίς την εκπαίδευση για να αποφύγουμε το overfit στο training set.

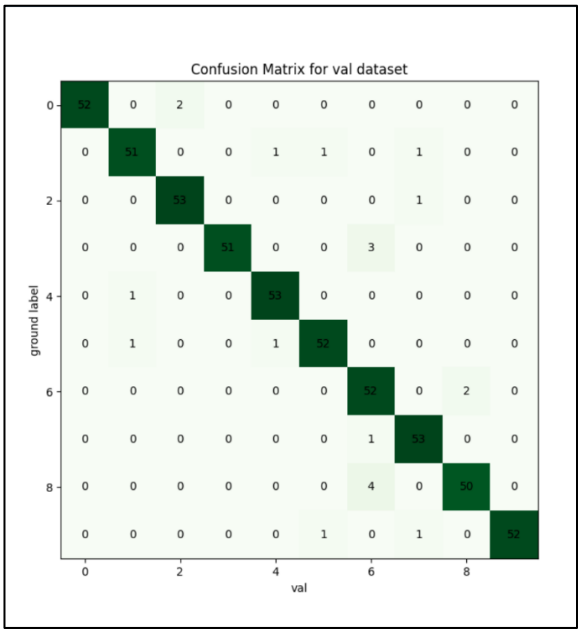
Bidirectional LSTM: διαχειρίζεται τις εισόδους είτε προς τα εμπρός (από παρελθοντική πληροφορία σε μελλοντική) είτε προς την αντίστροφη κατεύθυνση (από μελλοντική σε παρελθοντική πληροφορία). Πρακτικά αυτό σημαίνει ότι το hidden state του bidirectional LSTM θα είναι διπλάσιο. Συνεπώς, το bidirectional LSTM περιέχει δύο δίκτυα γεγονός που το καθυστερεί αλλά είναι και πιο αποτελεσματικό αφού οι προβλέψεις του βασίζονται τόσο από το παρελθόν όσο και από το μέλλον.

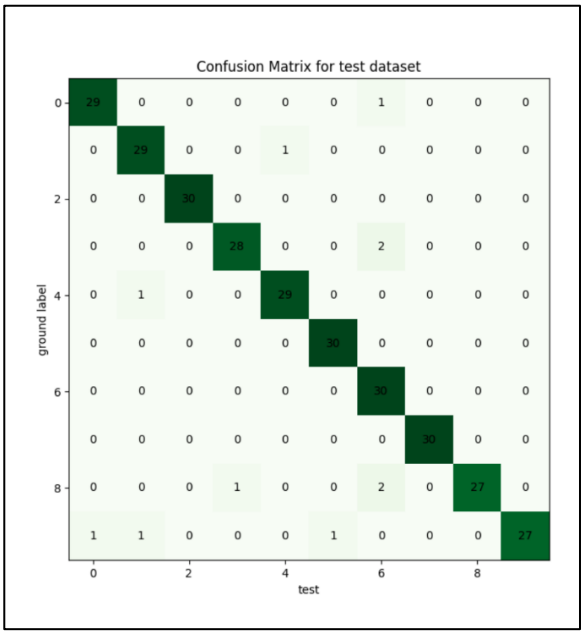
Εν τέλει, το καλύτερο μοντέλο με διαφορά είναι αυτό στο οποίο χρησιμοποιήθηκε bidirectional LSTM με learning rate 0.05 και momentum 0.9. Επιπροσθέτως, με early stopping σε συγκεκριμένες εποχές, το accuracy score στο validation set είναι 97.2% και στο test set 97.3%.





Και οι confusion matrices, όπως προέκυψαν από την προπαρασκευή, θα είναι:





Βιβλιογραφία - Αναφορές

Σημειώσεις / διαφάνειες μαθήματος & παλαιότερο υλικό