ΗΜΜΥ ΕΜΠ

Αναγνώριση Προτύπων (2022-2023)

2η Εργαστηριακή Άσκηση

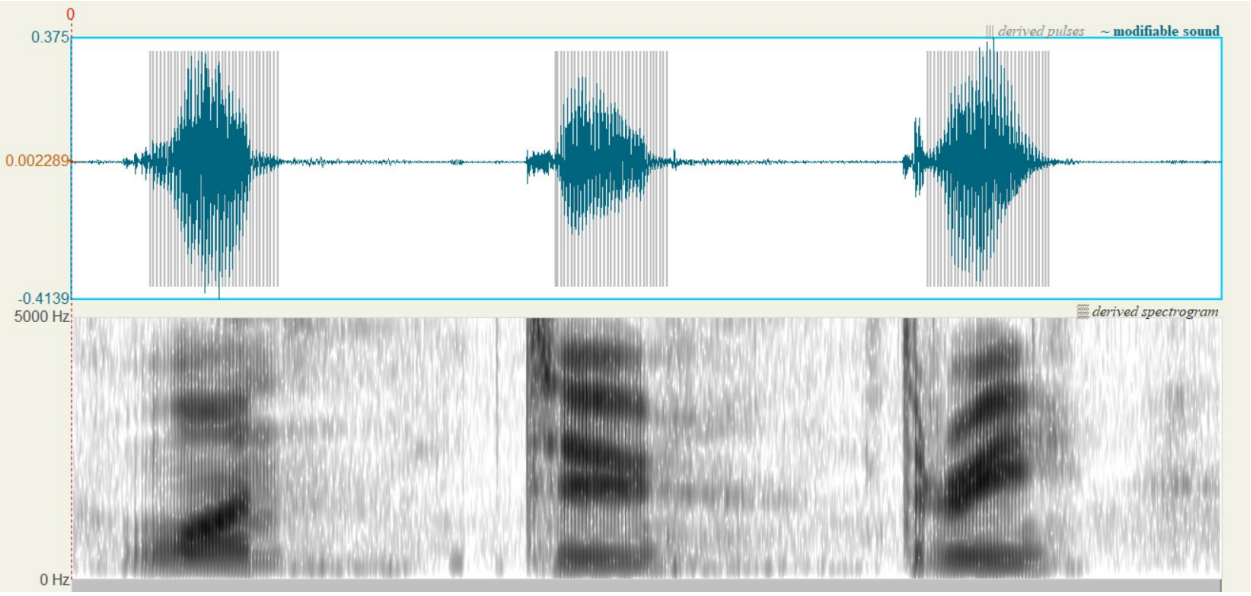
Ηλιόπουλος Γεώργιος:	03118815	giliopoulos301@gmail.com
Σερλής Εμμανουήλ Αναστάσιος:	03118125	manosserlis@gmail.com

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

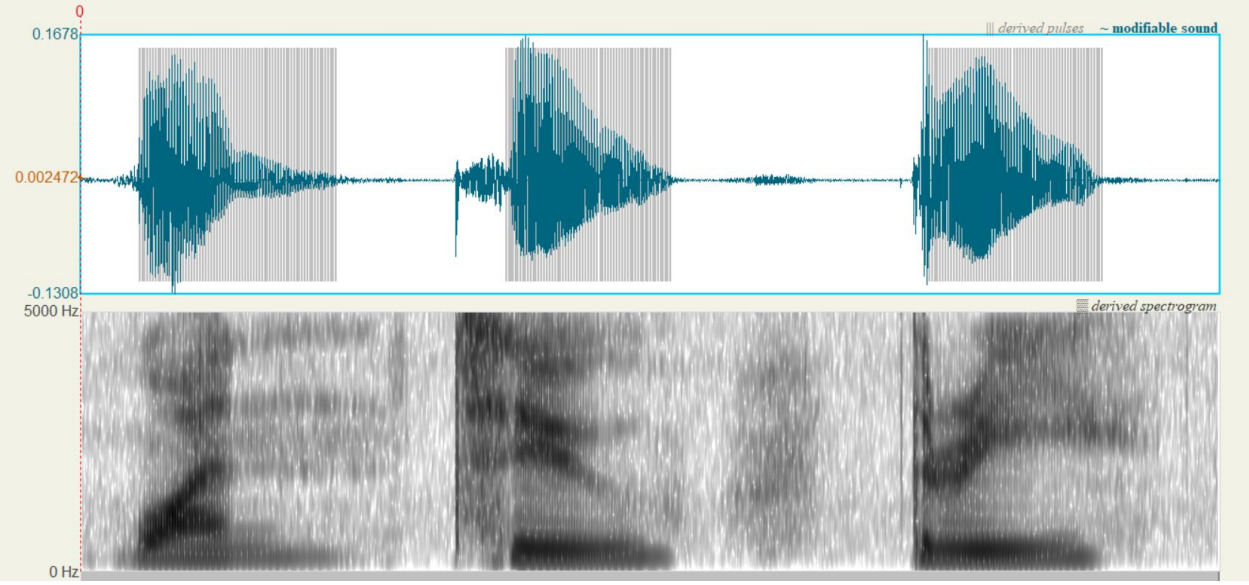
Σκοπός είναι η υλοποίηση ενός συστήματος επεξεργασίας και αναγνώρισης φωνής, με εφαρμογή σε αναγνώριση μεμονωμένων λέξεων. Στο στάδιο της προπαρασκευής θα γίνει εξαγωγή κατάλληλων ακουστικών χαρακτηριστικών από φωνητικά δεδομένα, χρησιμοποιώντας τα κατάλληλα πακέτα python, καθώς και ανάλυση και απεικόνισή τους με σκοπό την κατανόηση και την εξαγωγή χρήσιμων πληροφοριών από αυτά.

Βήμα 1: Ανάλυση φωνής μέσω του λογισμικού Praat

Αρχικά, φορτώσαμε στο λογισμικό Praat τα αρχεία ήχου onetwothree1.wav και onetwothree8.wav, με τις παραγόμενες κυματομορφές και τα αντίστοιχα spectrograms να παρατίθενται στις εικόνες 1 και 2.



Εικόνα 1: Κυματομορφή και spectrogram για το αρχείο ήχου onetwothree1.wav



Εικόνα 2: Κυματομορφή και spectrogram για το αρχείο ήχου onetwothree2.wav

Στην συνέχεια, για κάθε αρχείο ήχου, εξάγαμε τις μέσες τιμές των pitches (μέσω της εντολής "Get Pitch") καθώς και τα 3 πρώτα formants (μέσω των εντολών ''Get first/second/third formant") για καθένα από τα φωνήεντα «α», «ου» και «ι». Συγκεντρωτικά αποτελέσματα παρατίθενται στους κάτωθι πίνακες

Αρχείο	Φωνήεν	Pitch(Hz)
onetwothree1.wav	'α'	135.2
onetwothree1.wav	'ου'	129.9
onetwothree1.wav	'ι'	131.08
onetwothree8.wav	'α'	180.06
onetwothree8.wav	'ου'	190.09
onetwothree8.wav	'ι'	179.58

Πίνακας 1: Μέσες τιμές των pitches για κάθε αρχείο ήχου

Αρχείο	Φωνήεν	Formant1(Hz)	Formant2(Hz)	Formant3(Hz)
onetwothree1.wav	'α'	706.52	1102.28	2368
onetwothree1.wav	'ου'	433.85	1834.82	2491.39
onetwothree1.wav	'ι'	394.74	1845.37	3232.95
onetwothree8.wav	'α'	713.17	1407.42	2827.28
onetwothree8.wav	'ου'	343.56	1754.9	2702.77
onetwothree8.wav	'ι'	377.9	2113.15	2750.97

Πίνακας 2: 3 πρώτα formants για κάθε αρχείο ήχου

Παρατηρούμε ότι οι 2 ομιλητές παρουσιάζουν διαφορετικές τιμές mean pitch για τα ίδια φωνήεντα, με την γυναίκα ομιλήτρια (onetwothree8.wav) να έχει τις υψηλότερες τιμές σε σχέση με τον άντρα (onetwothree1.wav). Απεναντίας, οι τιμές των formants μεταξύ των 2

ομιλητών δεν έχουν κάποιο σταθερό μοτίβο διαφοροποίησης. Συνεπώς, τα mean pitch values μοιάζουν να είναι ένα ικανό feature για την διάκριση του φύλου του ομιλητή.

Στην συνέχεια, παρατηρούμε ότι-για τον ίδιο ομιλητή- καθένα εκ των formants 1,2 και 3 λαμβάνουν αισθητά διαφορετικές τιμές ανά φωνήεν, γεγονός που τα καθιστά κατάλληλα για την διάκριση φωνηέντων.

Βήμα 2: Data Parser

Σε αυτό το βήμα φτιάξαμε μία συνάρτηση data parser που διαβάζει το σύνολο των δεδομένων μας και επιστρέφει το wav, τον ομιλητή και το ψηφίο. Αφού τρέξουμε την συνάρτηση διαπιστώνουμε πως έχουμε 133 wav αρχεία ήχου για 9 ψηφία $(1-9)$ από 15 διαφορετικούς εκφωνητές.

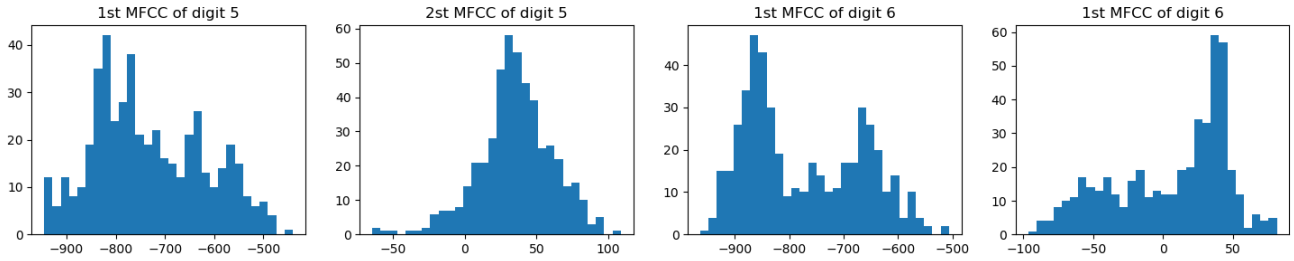
Βήμα 3: Mel-Frequency Cepstral Coefficients (MFCCs)

Για την εξαγωγή των mfccs χρησιμοποιήθηκαν παράθυρα $25ms$ και βήμα $10ms$. Επίσης υπολογίστηκαν η πρώτη και η δεύτερη τοπική παράγωγος των χαρακτηριστικών deltas και delta-deltas. Για κάθε wav αρχείο δημιουργήθηκε ένα διάνυσμα 13×30 που υποδηλώνει τα 13 χαρακτηριστικά για τα 30 παράθυρα. Τα διανύσματα αυτά αποθηκεύτηκαν σε μία λίστα μήκους 133, όσες δηλαδή και οι εκφωνήσεις.

Βήμα 4: Ιστογράμματα MFCCs και Mel Filterbank Spectral Coefficients (MFSCs)

Επειδή ο αριθμός μητρώου και των δύο μας τελειώνει σε 5 πήραμε ως $n1$ και $n2$ τα ψηφία 5 και 6 αντίστοιχα.

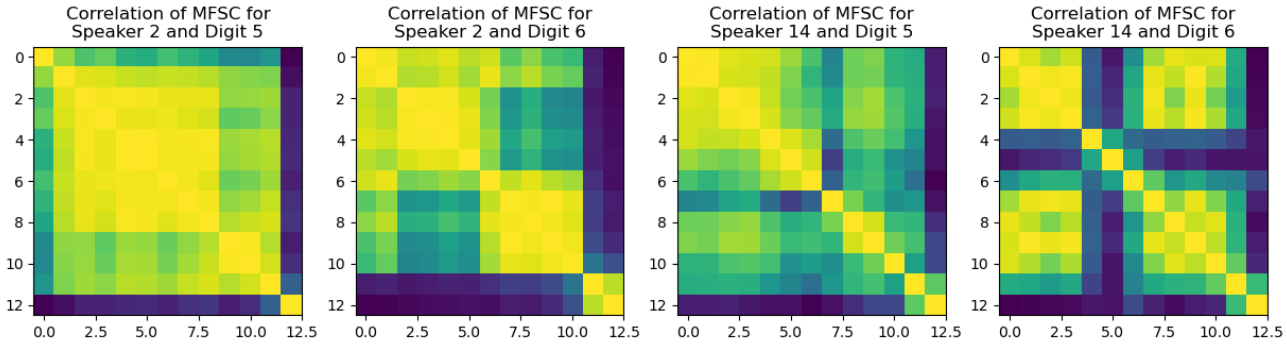
Στην εικόνα 3 απεικονίζουμε τα ιστογράμματα των πρώτων 2 mfccs των ψηφίων 5 και 6 από όλες τις εκφωνήσεις. Παρατηρούμε πως αν και ως φωνητικό άκουσμα οι λέξεις «five» και «eight» διαφέρουν αρκετά, τα ιστογράμματα των πρώτων 2 mfcc χαρακτηριστικών διαφέρουν ελάχιστα. Αντιλαμβανόμαστε λοιπόν πως για ένα μοντέλο φωνητικής αναγνώρισης ψηφίων απαιτούνται και τα υπόλοιπα χαρακτηριστικά που υπολογίσαμε στο βήμα 3.



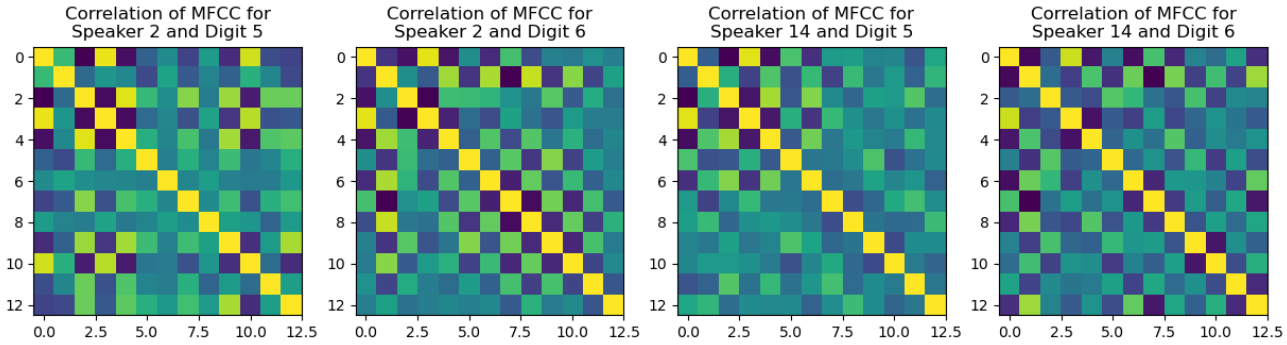
Εικόνα 3: Ιστογράμματα 1ου και 2ου mfcc χαρακτηριστικού των ψηφίων 5 και 6

Στη συνέχεια μελετήσαμε 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. Στην εικόνα 4 παρουσιάζουμε την συσχέτιση των mfsc δύο

διαφορετικών εκφωνήσεων από το ψηφίο 5 και 6 (4 εκφωνήσεις σύνολο). Για σκοπούς σύγκρισης στην εικόνα 5 βλέπουμε την συσχέτιση των mfcc των ίδιων εκφωνήσεων.



Εικόνα 4: Συσχέτιση MFSC των ψηφίων 5 και 6 από τους εκφωνητές 2 και 14



Εικόνα 5: Συσχέτιση MFCC των ψηφίων 5 και 6 από τους εκφωνητές 2 και 14

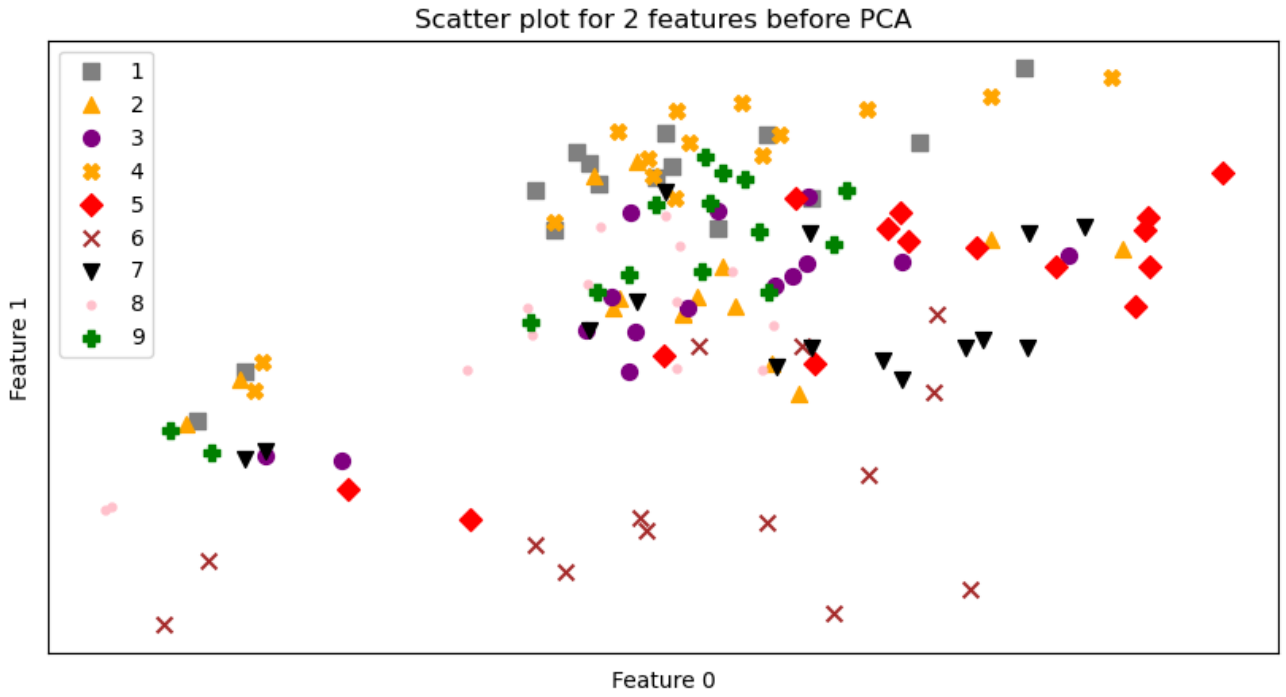
Συγκρίνοντας τις εικόνες 4 και 5 διαπιστώνουμε πως υπάρχει αρκετά υψηλή συσχέτιση στα MFSC χαρακτηριστικά αφού παρατηρούνται υψηλές τιμές σε αρκετές γειτονιές pixel του πίνακα συσχέτισης και όχι μόνο στη διαγώνιο. Αντίθετα, για την περίπτωση των MFCC χαρακτηριστικών, υπάρχουν υψηλές τιμές κυρίως στη διαγώνιο, γεγονός που υποδεικνύει χαμηλή συσχέτιση. Για να έχουμε υψηλότερη διακριτική ικανότητα επιλέγουμε τα χαρακτηριστικά με τη χαμηλότερη συσχέτιση, δηλαδή τα MFCC χαρακτηριστικά αφού αυτά είναι που θα μας δώσουν περισσότερη πληροφορία ανά ψηφίο.

Βήμα 5: Εξαγωγή μοναδικού διανύσματος χαρακτηριστικών για κάθε εκφώνηση

Για την αναγνώριση των ψηφίων απαιτείται να εξαχθεί ένα διάνυσμα χαρακτηριστικών για κάθε εκφώνηση. Αυτό το διάνυσμα περιέχει τα χαρακτηριστικά mfcc, τα deltas και τα delta-deltas. Συγκεκριμένα έγινε υπολογισμός της μέση τιμής και της διασποράς όλων των παραθύρων κάθε εκφωνήσεις. Έτσι καταλήγουμε σε ένα διάνυσμα της μορφής:

$$features = \left[\mu_{mfcc}|\mu_{delta}|\mu_{delta-delta}|\sigma_{mfcc}|\sigma_{delta}|\sigma_{delta-delta}\right]_{1\times 78}$$

για κάθε εκφώνηση, συνολικά δηλαδή 133 διανύσματα. Στην εικόνα 6 φαίνεται ένα scatter plot με τις πρώτες δύο διαστάσεις του διανύσματος features.

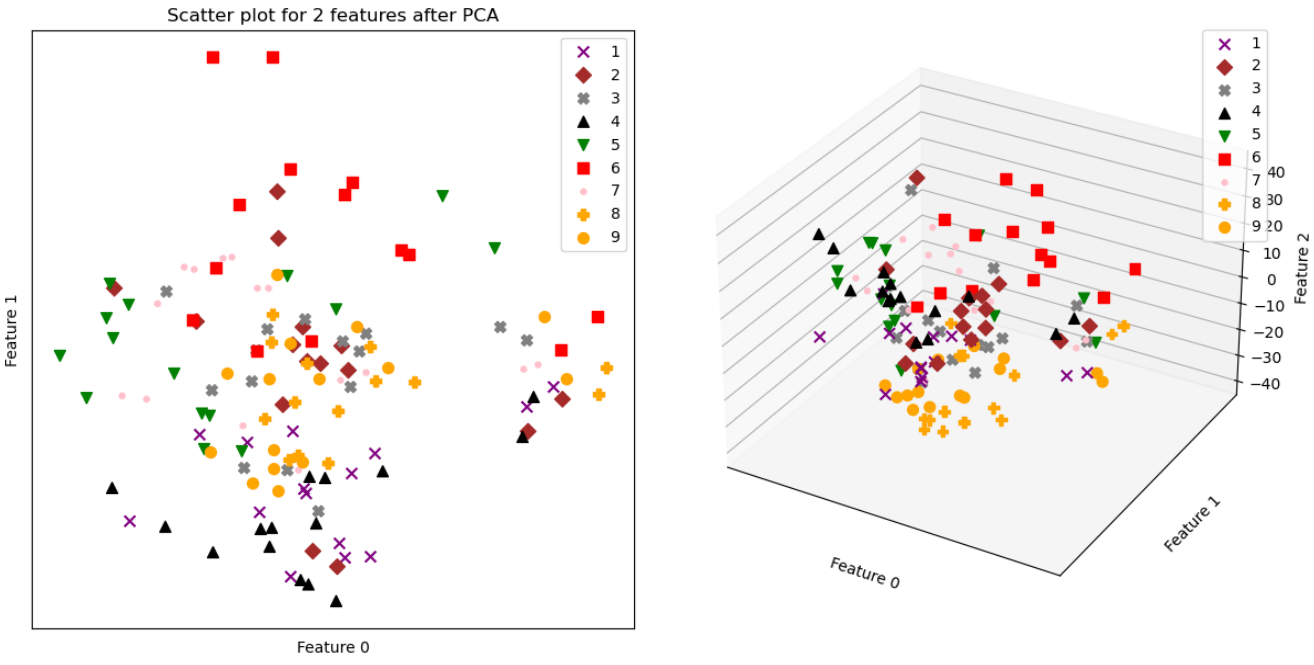


Εικόνα 6: Scatter plot για δύο χαρακτηριστικά πριν το PCA

Παρατηρούμε πως τα δείγματά μας δεν ομαδοποιούνται ιδιαιτέρως λαμβάνοντας υπόψιν μόνο τα πρώτα δύο χαρακτηριστικά και έτσι δεν μπορούμε να ορίσουμε «περιοχές» για το κάθε ψηφίο. Αυτό είναι λογικό αφού η επιλογή των δύο πρώτων χαρακτηριστικών από κάθε δείγμα έγινε αυθαίρετα.

Βήμα 6: Principal Component Analysis (PCA)

Στη συνέχεια για τη μείωση των διαστάσεων των χαρακτηριστικών εφαρμόζουμε PCA και μειώνουμε σε 2 και 3 διαστάσεις. Στην εικόνα 7 μπορούμε να δούμε ένα scatter plot για 2 και για 3 διαστάσεις.



(α) (β)

Εικόνα 7: (α) Scatter plot μετά από μείωση σε 2 διαστάσεις με PCA, (β) Scatter Plot μετά από μείωση σε 3 διαστάσεις με PCA

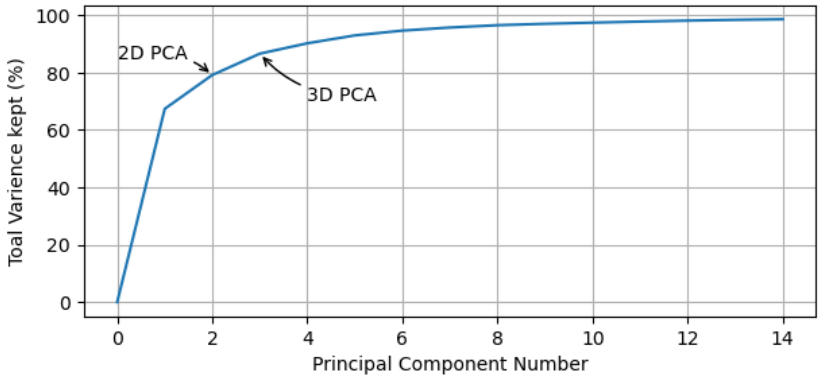
Παρατηρούμε πως με μείωση τόσο σε 2 διαστάσεις όσο και σε 3 δεν έχουμε ευκρινώς διαχωρίσιμες κλάσεις, αφού τα δείγματα μας είναι αρκετά μπλεγμένα μεταξύ τους.

Χρησιμοποιώντας .explained_variance_ratio_ μπορούμε να δούμε το ποσοστό της διασποράς των χαρακτηριστικών που κρατά η μείωση PCA. Τα στατιστικά φαίνονται στον πίνακα 3.

Διαστάσεις PCA	1η συνιστώσα	2η συνιστώσα	3η συνιστώσα	Συνολικά
2	58.79%	11.86%	-	79.18%
3	58.79%	11.86%	10.84%	86.63%

Πίνακας 3: Ποσοστό διατήρησης αρχικής διασποράς μετά την εφαρμογή PCA

Παρατηρούμε πως η μείωση σε 2 ή 3 διαστάσεις δεν είναι ικανοποιητική αφού ένα μεγάλο μέρος της αρχικής διασποράς χάνεται. Στην εικόνα 8 παρατηρούμε πως για PCA σε περίπου 7 διαστάσεις και πάνω διατηρείται ικανοποιητικό ποσοστό της διασποράς των χαρακτηριστικών.



Εικόνα 8: Σχέση αριθμού διαστάσεων PCA και κρατούμενου ποσοστού διασποράς χαρακτηριστικών

Βήμα 7: Ταξινόμηση

Προκειμένου να γίνει η ταξινόμηση των δεδομένων αρχικά τα χωρίζουμε σε train και test set με αναλογία $70\% - 30\%$ και τα κανονικοποιούμε κατά τις υποδείξεις της εκφώνησης. Για την ταξινόμηση χρησιμοποιήθηκαν 4 ταξινομητές. Τα αποτελέσματα της ταξινόμησης συμπυκνώνονται στον πίνακα 4.

Classifier	Normalization	Score
Custom Naive Bayes	No	62.5%
	Yes	57.5%
sklearn Naive Bayes	No	62.5%
	Yes	57.5%
kNeighbors ($k = 3$)	No	52.5%

	Yes	52.5%
Logistic Regression	No	72.5%
	Yes	17.5%
SVM with linear kernel	No	**75%**
	Yes	17.5%

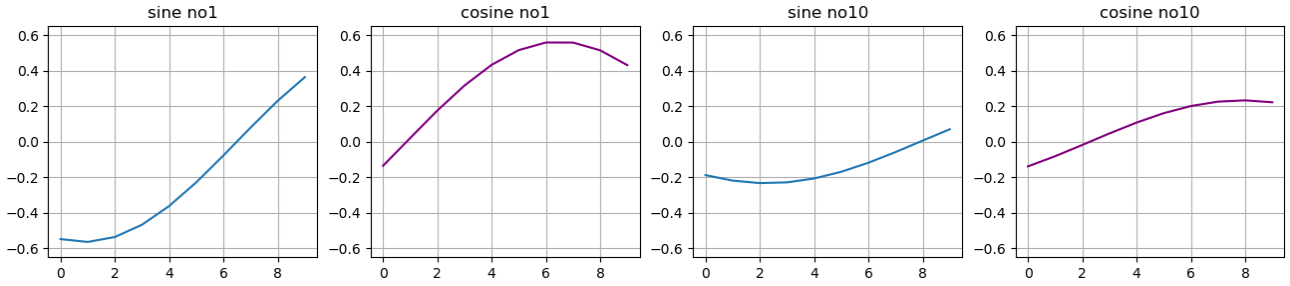
Πίνακας 4: Ποσοστά επιτυχίας ταξινομητών

Παρατηρήσεις:

- Για όλους τους ταξινομητές παρατηρούμε καλύτερη επίδοση για μη κανονικοποιημένα δεδομένα. Αυτό είναι λογικό αφού μετά την κανονικοποίηση των δεδομένων έχουμε χάσει ένα μικρό κομμάτι πληροφορίας χρήσιμο για την ταξινόμηση.
- Την καλύτερη επίδοση πετυχαίνει ο SVM χωρίς κανονικοποιημένα δεδομένα ενώ την χειρότερη ο SVM και ο Logistic Regression με κανονικοποιημένα δεδομένα.
- Για διαφορετικά split οι επίδοσης μεταβάλλονται, αφού υπάρχει τυχαιότητα κατά τον διαχωρισμό.
- Τα μειωμένα ποσοστά επιτυχίας σχετίζονται άμεσα με το γεγονός ότι τα άνωθι μοντέλα δεν λαμβάνουν υπόψιν την χρονική εξάρτηση των ακουστικών σημάτων, πρόβλημα το οποίο θα αντιμετωπισθεί με την χρήση RNNs παρακάτω.

Βήμα 8: Εξοικείωση με PyTorch σε σήματα ημιτόνων/συνημίτονων

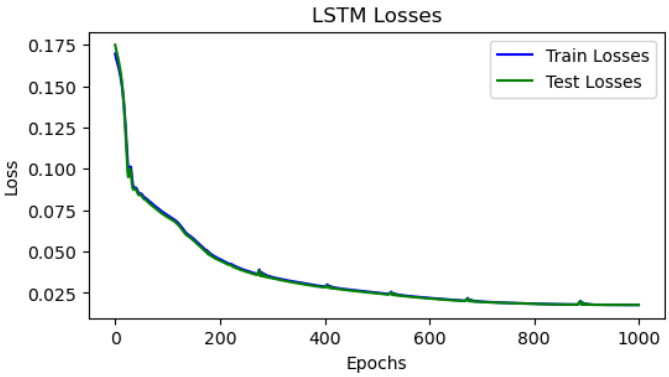
Αρχικά δημιουργούμε τα σήματα που θα χρησιμοποιήσουμε. Συγκεκριμένα δημιουργούνται σήματα ημιτόνων και συνημίτονων συχνότητας $f = 40Hz$ με τυχαίο πλάτος και αρχική φάση (κάθε ημίτονο δημιουργούμε το αντίστοιχο συνημίτονο). Στην εικόνα 8 φαίνονται ως παράδειγμα κάποια δείγματα.



Εικόνα 9: Δείγματα που δημιουργήθηκαν για το βήμα 8

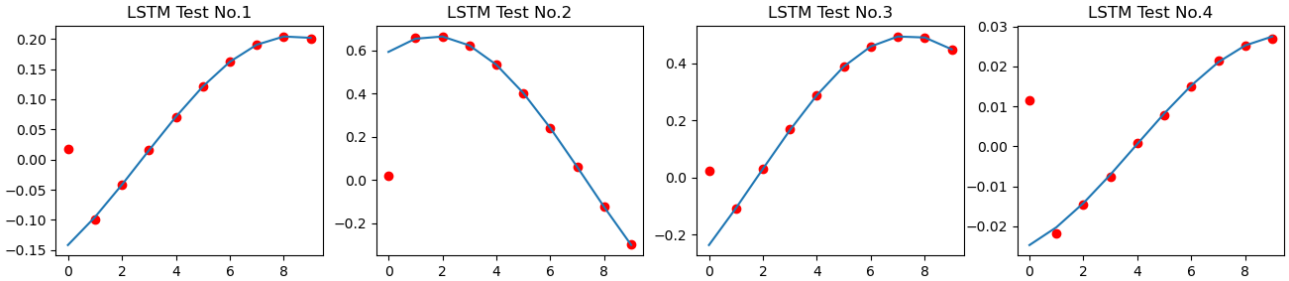
Σκοπός μας είναι να δημιουργηθεί ένα Αναδρομικό Νευρωνικό Δίκτυο (Recurrent Neural Network – RNN) το οποίο θα δέχεται ως είσοδο τις ακολουθίες του ημιτόνου και θα πρέπει να προβλέπει τις αντίστοιχες ακολουθίες συνημίτονου.

Για το σκοπό αυτό χρησιμοποιούμε ένα LSTM (Long Short-Term Memory) μοντέλο με 100 κρυφά επίπεδα και συνάρτηση κόστους το μέσο τετραγωνικό σφάλμα. Ως βελτιστοποιητής χρησιμοποιήθηκε ο Adam optimizer με ρυθμό εκμάθησης 10^{-3}. Το μοντέλο εκπαιδεύτηκε σε 1000 εποχές και τα δεδομένα χωρίστηκαν σε train και test set σε ποσοστά $70\% - 30\%$ αντίστοιχα. Στην εικόνα 10 φαίνεται πως εξελίσσεται το κριτήριο κόστους.



Εικόνα 10: Train και Test losses για το LSTM μοντέλο

Στην εικόνα 11 φαίνονται ως παράδειγμα οι προβλέψεις του μοντέλου για 4 δείγματα και συγκρίνονται με τις αντίστοιχες ground truth κυματομορφές:



Συμπερασματικά, παρατηρώντας τις εικόνες 10 και 11, βλέπουμε πως το μοντέλο έχει πολύ καλή απόδοση αφού μπορεί να προβλέψει με τεράστια ακρίβεια τα σημεία των συνημίτονων. Φυσικά χάνει το πρώτο σημείο γιατί ακόμα το μοντέλο δεν έχει αρκετές πληροφορίες και τα αναδρομικά μοντέλα αποθηκεύουν στη μνήμη τους πληροφορία για την προηγούμενη χρονική ακολουθία.

Πλέον δεν χρησιμοποιούνται τα vanilla RNN αλλά οι παραλλαγές του, όπως το LSTM που χρησιμοποιήσαμε παραπάνω. Το LSTM έχει τη δυνατότητα να εντοπίζει απομακρυσμένες εξαρτήσεις στα σήματα κάτι που είναι αρκετά χρήσιμο για χρονικά σήματα. Το πρόβλημα των vanilla RNNs είναι κυρίως το φαινόμενο vanishing gradient, δηλαδή η εκθετική μείωση του gradient της συνάρτησης κόστους στο χρόνο και έτσι χάνονται οι μακρινές εξαρτήσεις. Επίσης τα LSTM μοντέλα χρησιμοποιούν εκτός των κανονικών units και ειδικά memory cells για τη διατήρηση πληροφορίας για μεγάλα χρονικά διαστήματα.

Βήμα 9: Εισαγωγή Free Spoken Digit Dataset (FSDD)

Στα επόμενα βήματα θα χρησιμοποιηθεί ένα μεγαλύτερο set δεδομένων, το Free Spoken Digit Dataset (FSDD).

Χρησιμοποιώντας τις συναρτήσεις που μας δίνονται, κάναμε εισαγωγή των δεδομένων, εξάγαμε MFCCs και τα κανονικοποιήσαμε με τη χρήση ενός Standard Scaler βάσει της μέσης τιμής και της τυπικής απόκλισης. Συγκεκριμένα έχουμε 6000 δείγματα και από αυτά έγινε εξαγωγή 6 MFFCs ανά frame. Τα δεδομένα χωρίστηκαν σε training και validation set με αναλογία $80\% - 20\%$.

Βήμα 10: Αναγνώριση ψηφίων με Gaussian Mixture Models (GMMs) και Hidden Markov Models (HMMs)

Για αυτό το βήμα χρησιμοποιήθηκε η βιβλιοθήκη pomegranate της python.

Για κάθε ψηφίο φτιάχνουμε ένα GMM-HMM μοντέλο. 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 $a_{ij} = 0$ για $j > i + 1$. Η πιθανότητα για κάθε μετάβαση είναι 0.5.

Αρχικά χωρίσαμε τα δεδομένα μας ανά ψηφίο ώστε να φτιαχτούν τα 10 μοντέλα (1 για κάθε ψηφίο) και στη συνέχεια υλοποιήσαμε μία συνάρτηση που δέχεται ως όρισμα τα δεδομένα, τον αριθμό των HMM καταστάσεων και τον αριθμό των Γκαουσιανών κατανομών και επιστρέφει το μοντέλο μας.

Βήμα 11: Εκπαίδευση μοντέλων

Σε αυτό το βήμα εκπαιδεύουμε ως παράδειγμα ένα μοντέλο με 2 HMM καταστάσεις, 2 Γκαουσιανές κατανομές και max_iterations = 30 (στο βήμα 12 θα πειραματιστούμε με περισσότερους συνδυασμούς) χρησιμοποιώντας όλα τα δεδομένα που έχουμε διαθέσιμα για κάθε ψηφίο.

Βήμα 12: Testing Μοντέλων

Για να κάνουμε testing των μοντέλων εξάγουμε τις προβλέψεις του μοντέλου για το validation set και τις συγκρίνουμε με τις αληθείς τιμές ώστε να δούμε ποιος συνδυασμός παραμέτρων είναι ο βέλτιστος.

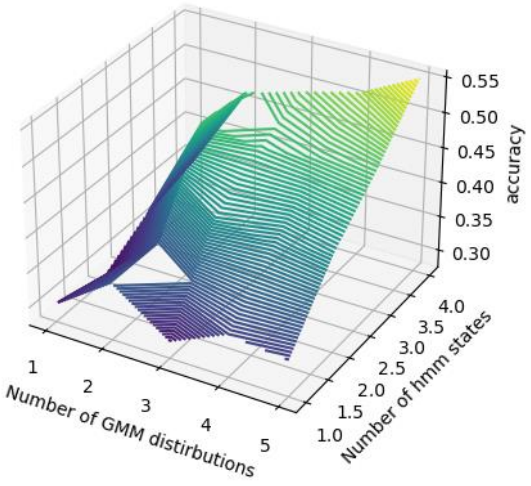
Για την εξαγωγή των προβλέψεων υπολογίζεται για κάθε εκφώνηση ο λογάριθμος της πιθανοφάνειας για όλα τα μοντέλα και το ψηφίο ταξινομείται στο μοντέλο που δίνει τη μέγιστη πιθανοφάνεια.

Πειραματιστήκαμε με 1-4 HMM καταστάσεις, 1-5 Γκαουσιανές κατανομές και διάφορες τιμές για maximum iterations. Καταλήγουμε πως το καλύτερο αποτέλεσμα το έχουμε για όσο μεγαλύτερο αριθμό maximum iterations. Τα ποσοστά ακρίβειας για τους υπόλοιπους συνδυασμούς με 30 maximum iterations φαίνονται στον πίνακα 5.

Αριθμός HMM καταστάσεων	Αριθμός Γκαουσιανών κατανομών	Ακρίβεια
1	1	0.3009
	2	0.3565
	3	0.2986

	4	0.3356
	5	0.3264
2	1	0.2801
	2	0.3565
	3	0.4167
	4	0.4028
	5	0.4167
3	1	0.3218
	2	0.4769
	3	0.4606
	4	0.4699
	5	0.4792
4	1	0.3426
	2	0.4676
	3	0.4907
	4	0.5116
	5	0.5532

Πίνακας 5: Ακρίβειες για διάφορα GMM-HMM μοντέλα



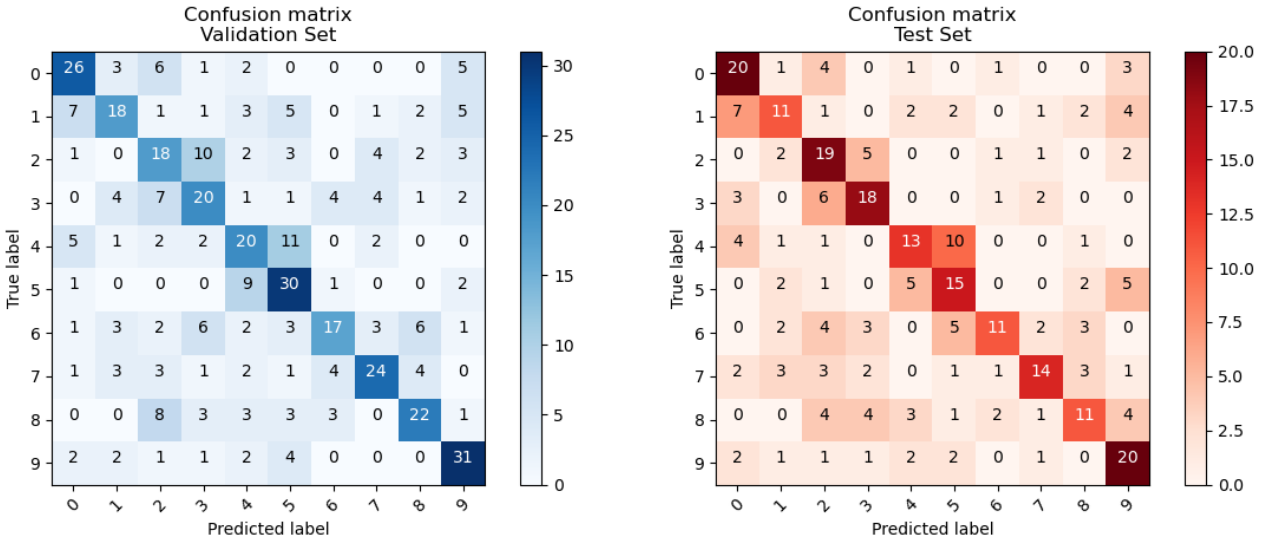
Εικόνα 11: Ακρίβειες για διάφορα GMM-HMM μοντέλα σε 3D plot

Παρατηρούμε λοιπόν πως ο καλύτερος συνδυασμός είναι 4 HMM καταστάσεις και 5 γκαουσιανές κατανομές που πετυχαίνει ακρίβεια 0.5532 για το test set. Βάσει της εικόνας 11 μπορούμε να δούμε πως εν γένει πετυχαίνουμε μεγαλύτερη ακρίβεια για μοντέλα με μεγαλύτερο αριθμό HMM καταστάσεων και μεγαλύτερο αριθμό γκαουσιανών κατανομών.

Η παραπάνω τακτική, ο καθορισμός δηλαδή των παραμέτρων βάσει του validation set και όχι βάσει του train set είναι χρήσιμη καθώς έτσι αποφεύγουμε το over-fitting. Πιο συγκεκριμένα, οι παράμετροι του μοντέλου καθορίζονται βάσει δεδομένων διαφορετικών του train set που είναι άγνωστα προς αυτό ώστε να μην προσαρμοστεί πλήρως στα γνωστά του δεδομένα και μπορεί να προβλέψει μόνο αυτά. Ένα τέτοιο μοντέλο θα ήταν άχρηστο αφού δε θα μπορούσε να ανταπεξέλθει σε νέα δεδομένα.

Βήμα 13: Confusion Matrix

Στην εικόνα 12 φαίνονται οι confusion matrices για το μοντέλο με 3 HMM καταστάσεις και 4 γκαουσιανές κατανομές για το validation και το test set.



$accuracy_{dev} = 0.523$ $accuracy_{test} = 0.507$

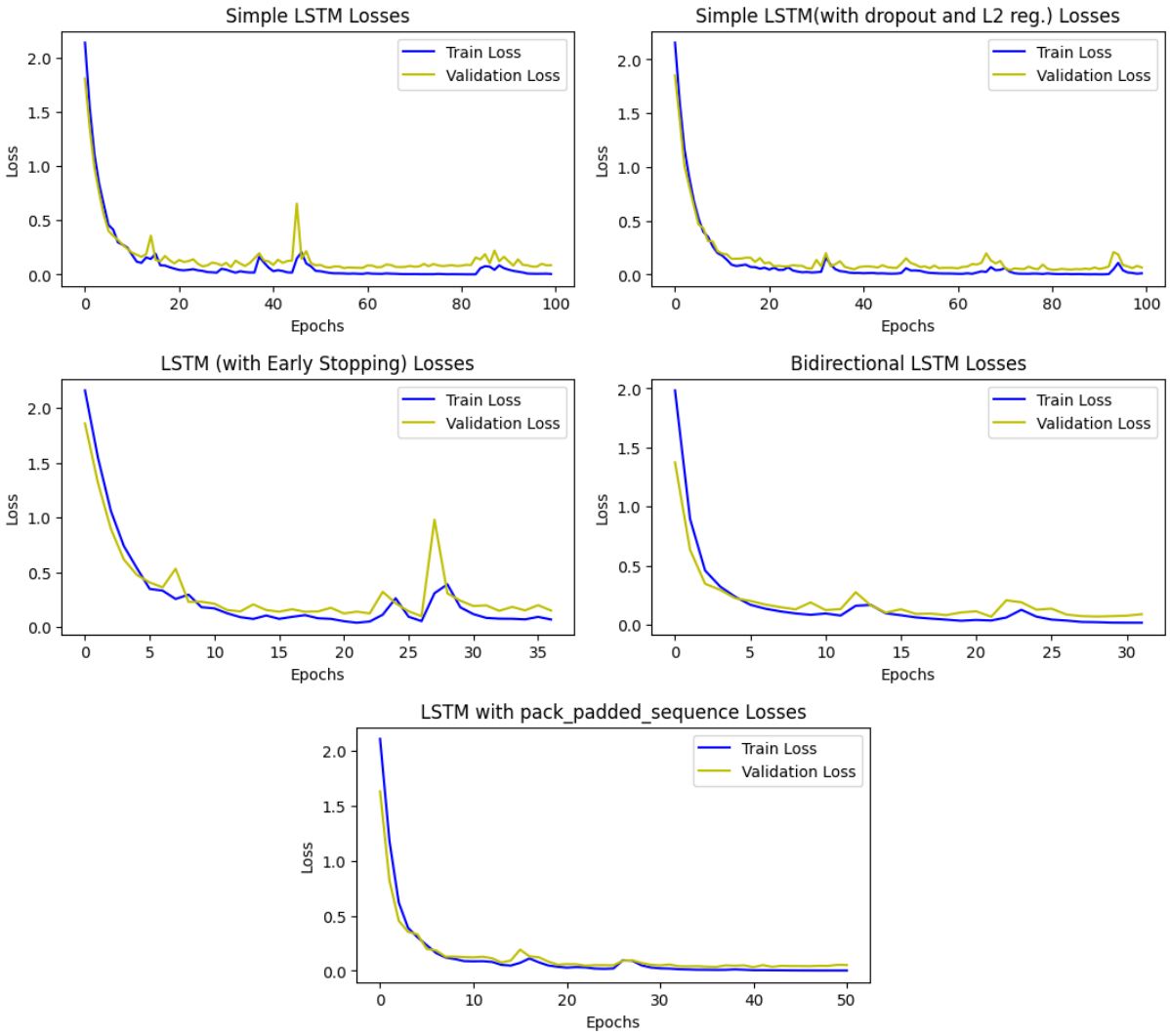
Εικόνα 12: Confusion Matrices για Validation (αριστερά) και Test (δεξιά) set

Βήμα 14: Ταξινόμηση με χρήση RNNs

Για τα ζητούμενα του εν λόγω ερωτήματος, οδηγηθήκαμε στην εκπαίδευση και ταξινόμηση 5 διαφορετικών RNNs τύπου LSTM, τα οποία αναλύονται παρακάτω:

1. **LSTM1:** Απλό LSTM δίκτυο
2. **LSTM2:** LSTM δίκτυο με dropout rate στο 20% και L2 Regularization (`weight_decay = 0.0001`). Η χρήση dropout rate οδηγεί στην απομάκρυνση ενός βάρους από το δίκτυο με πιθανότητα 20%, κάτι το οποίο γίνεται ανεξάρτητα σε κάθε εποχή προπόνησης. Η μέθοδος L2 regulization στοχεύει στην μείωση των βαρών με υψηλές τιμές, μέσω προσθήκης του αντίστοιχου τετραγωνικού όρου weight_penalty στο update των βαρών. Στόχος και των 2 τεχνικών είναι η μείωση του overfitting, μέσω επίδρασης τόσο στην αρχιτεκτονική του δικτύου όσο και στην τιμή του διανύσματος βαρών
3. **LSTM3:** LSTM δίκτυο με early stopping έπειτα από 10 διαδοχικές εποχές όπου το validation loss δεν μειώθηκε περαιτέρω. Έτσι, αποφεύγουμε την επίτευξη overfitting.
4. **LSTM4:** Bidirectional LSTM δίκτυο, το οποίο αποτελείται από 2RNNs, ένα forward που ξεκινά από την αρχή των δεδομένων και ένα backward που ξεκινά από το τέλος. Με αυτόν τον τρόπο, το τελικό output layer λαμβάνει πληροφορίες τόσο από past όσο και από future states.
5. **LSTM5:** LSTM δίκτυο που κάνει χρήση της συνάρτησης `pack_padded_sequence()`, αφού έχει προηγηθεί ταξινόμηση των ακολουθιών εισόδου κατά φθίνουσα σειρά μήκους.

Στον πίνακα 6 παρατίθενται αποτελέσματα που αφορούν τα loss values κατά το training των δικτύων, τα accuracy metrics καθώς και την διάρκεια προπόνησης των 5 LSTM NNs.



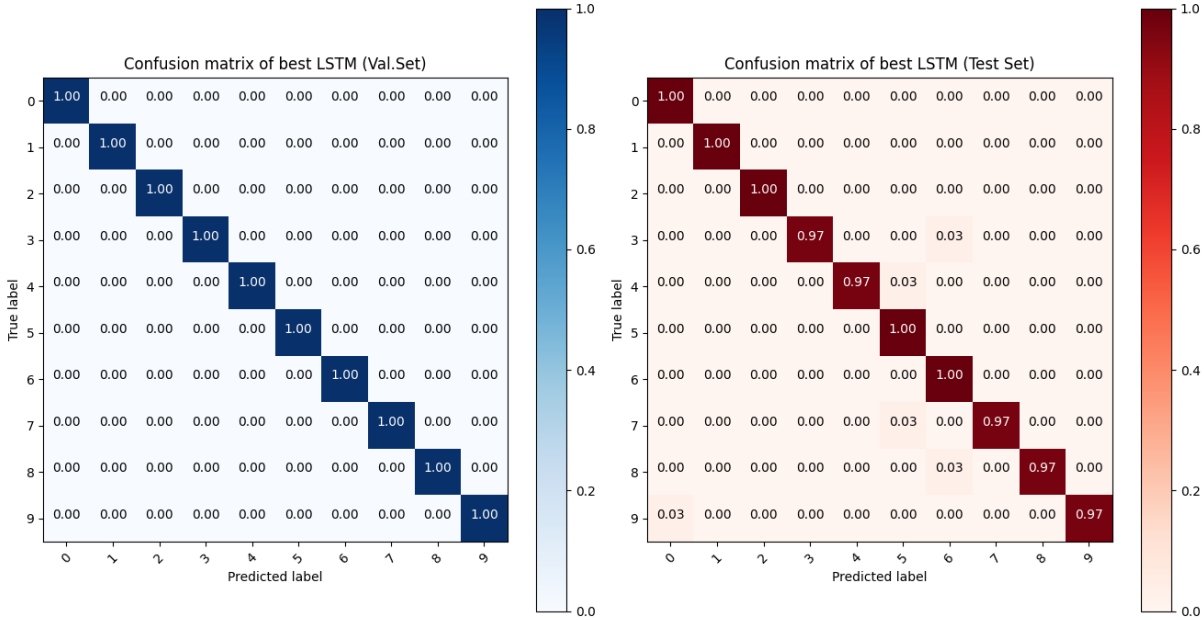
Εικόνα 13: Train και Test losses για τα μοντέλα

Network	Training Delay (mins)	Validation Accuracy	Test Accuracy	Mean Accuracy
LSTM1	15m39s	97.7%	**98.66%**	98.18%
LSTM2	16m37s	**98.51%**	98.33%	**98.42%**
LSTM3	**5m40s**	96.29%	96.33%	96.31%
LSTM4	10m47s	97.77%	97.33%	97.55%
LSTM5	6m23s	9.95%	10.66%	10.305%

Πίνακας 6: Σημαντικές μετρικές για το κάθε μοντέλο

Αρχικά, παρατηρούμε ότι τα loss values και για τα 5 δίκτυα δεν εμφανίζουν μεγάλες διαφορές, με την μόνη να έγκειται στην μείωση των σποραδικών peaks στο validation loss, που επιτυγχάνεται μέσω της χρήσης της συνάρτησης `pack_padded_sequence()`. Στην συνέχεια, βλέπουμε ότι πράγματι η χρήση τεχνικών μείωσης του overfitting οδηγεί σε βελτιωμένα αποτελέσματα (σε σχέση με το αρχικό δίκτυο LSTM1), με το δίκτυο LSTM2 να οδηγείται στο υψηλότερο mean accuracy score. Από την άλλη πλευρά, η ένταξη early stopping στο training του δικτύου μειώνει αισθητά τις εποχές προπόνησης και κατ' επέκταση την συνολική διάρκειά της (ειδικά στην περίπτωση του LSTM3), με κόστος την μικρή μείωση του accuracy κατά 1-2%. Τέλος, το LSTM5 παρά τα ικανοποιητικά loss scores, αδυνατεί να παρουσιάσει τα αναμενόμενα accuracy scores τόσο στο validation όσο και στο test set, οδηγούμενο σε ποσοστά ακρίβειας της τάξης του 10%.

Παρακάτω ακολουθούν τα confusion matrices για το πιο ακριβές δίκτυο (το LSTM2), τόσο πάνω στο validation set όσο και στο test set:



Πίνακας 7: Confusion Matrixes για Validation (αριστερά) και Test (δεξιά) set