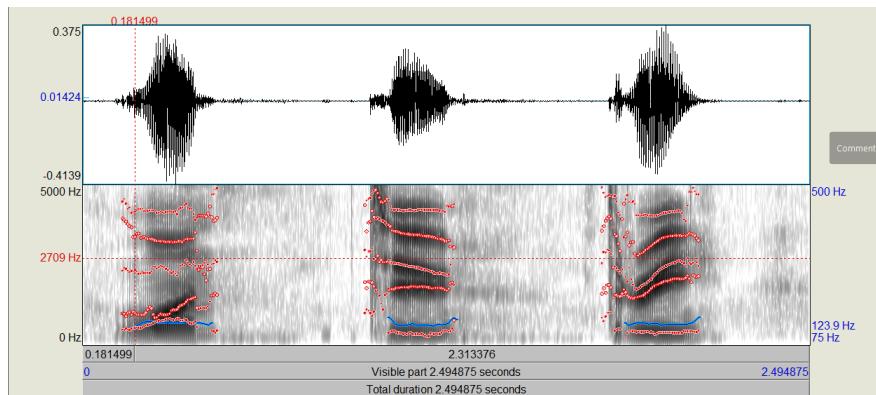


Προπαρασκευή Εργαστηρίου

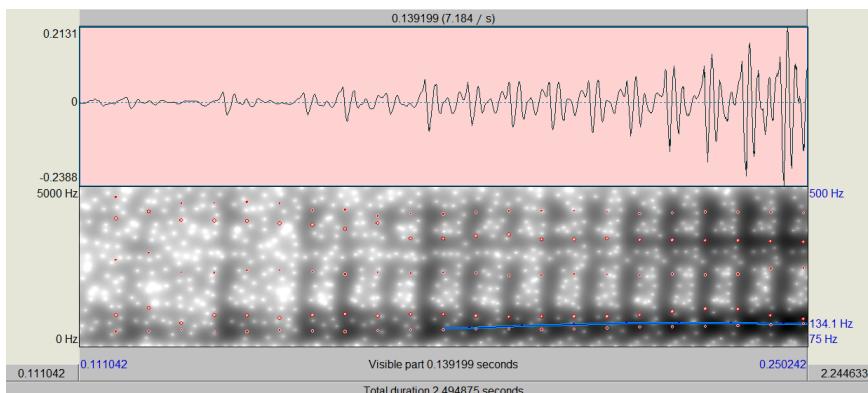
Στην προπαρασκευή του εργαστηρίου, θα ασχοληθούμε με την υλοποίηση ενός συστήματος επεξεργασίας και αναγνώρισης φωνής, με εφαρμογή σε αναγνώριση μεμονωμένων λέξεων. Μας δίνονται κάποια φωνητικά δεδομένα σε αρχεία, πάνω στα οποία θα εφαρμόσουμε τα μοντέλα μας, ώστε να εξάγουμε χαρακτηριστικά που θα αντιπροσωπεύουν τα ψηφία 1 έως 9.

Βήμα 1

Αρχικά, για το πρώτο αρχείο με ομιλιτή τον χρήστη 1, βλέπουμε στο Praat την κυματομορφή του αρχείου καθώς και το spectrogramm.

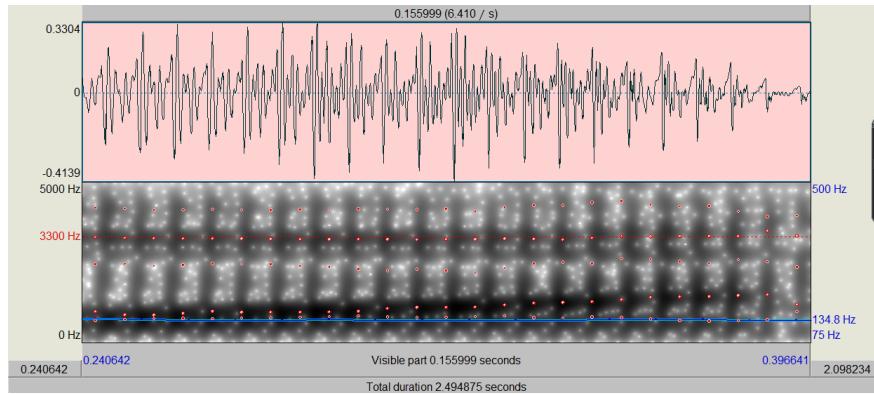


Για να απομονώσουμε τα φωνήματα που μας ζητούνται επάνω στο συνολικό αρχείο, πειραματιστήκαμε ακούγοντας πιθανά υποδιαστήματα και εν τέλει επιλέξαμε ως αντιπροσωπευτικά των φωνημάτων που μας ζητούνται εκείνα με τα παρακάτω διαγράμματα και σπεκτογράμματα:

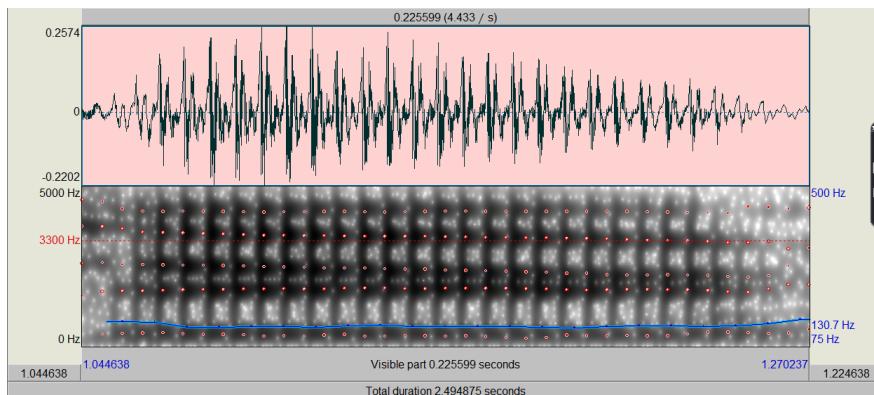


Σχήμα 1: Φώνημα /ou/ στην λέξη one

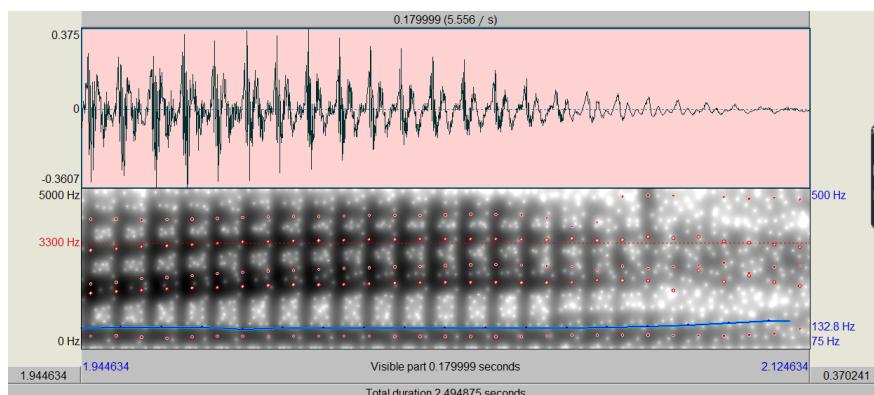
Η μέση τιμή του pitch είναι η μπλέ τιμή που εμφανίζεται δεξιά από τα διαγράμματα έτσι για το φωνήν "ou" στη λέξη one αυτή η τιμή είναι 134.1 και στη λέξη two είναι 130.7. Επίσης η μέση τιμή του pitch για το φωνήν "a" στην λέξη one είναι 134.8 και τέλος



Σχήμα 2: Φώνημα /a/ στην λέξη one



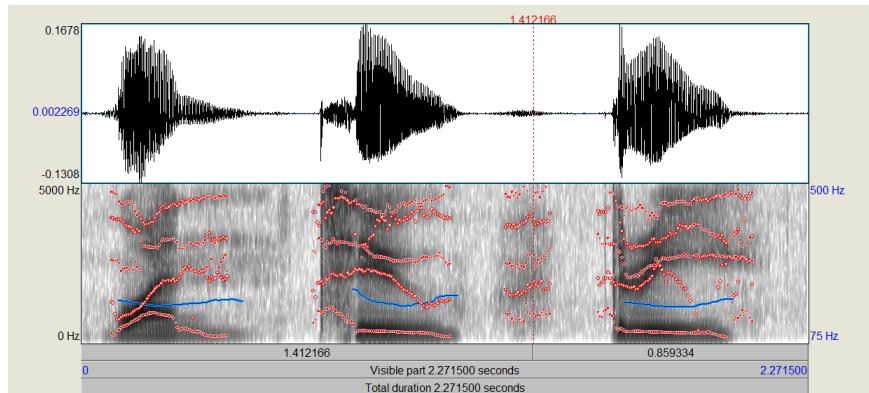
Σχήμα 3: Φώνημα /ou/ στην λέξη two



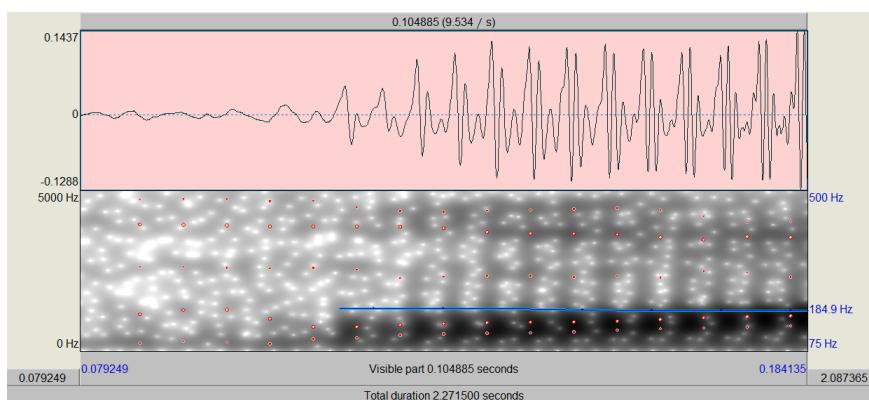
Σχήμα 4: Φώνημα /i/ στην λέξη three

για το φωνήεν "i" στην λέξη three είναι 132.8. Τα formants είναι αυτά που παρουσιάζονται με κόκκινα σημαδάκια επάνω στα σπεκτογράμματα και δηλώνουν κάποια peaks της φωνής λόγω συντονισμού. Δηλώνουν μέγιστα που εμφανίζονται στο φάσμα της ομιλίας.

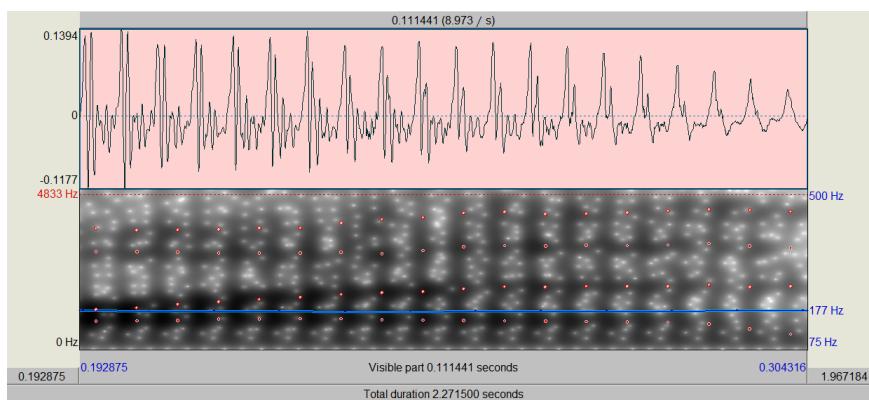
Παρακάτω ακολουθούν τα αντίστοιχα διαγράμματα για τον δεύτερο ομιλιτή.



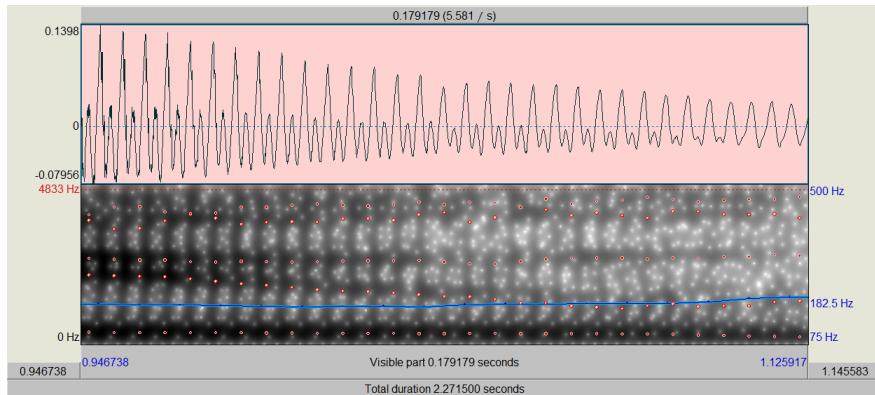
Σχήμα 5: Συνολική κυματομορφή και σπεκτόγραμμα δεύτερου ομιλιτή



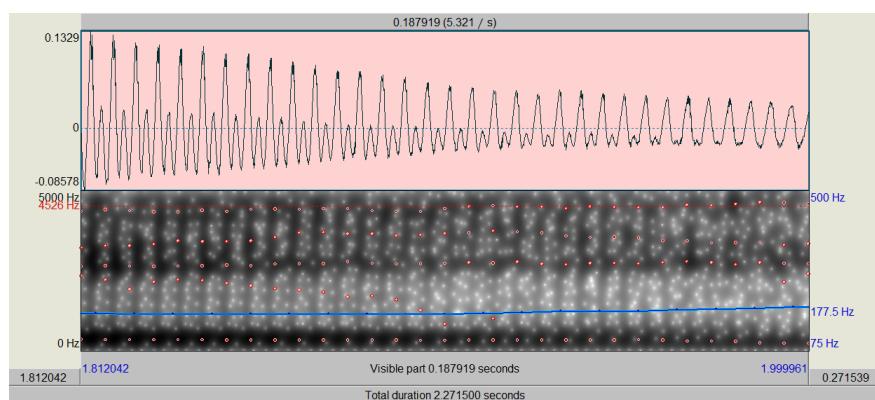
Σχήμα 6: Φώνημα /ou/ στην λέξη one, δεύτερος ομιλιτής



Σχήμα 7: Φώνημα /a/ στην λέξη one, δεύτερος ομιλιτής



Σχήμα 8: Φώνημα /ou/ στην λέξη two, δεύτερος ομιλιτής



Σχήμα 9: Φώνημα /i/ στην λέξη three, δεύτερος ομιλιτής

Τα τρία πρώτα formants είναι αυτά που εμφανίζονται πρώτα χρονικά στα spectrograms (αριστερά). Αυτό που παρατηρούμε, είναι ότι στον δεύτερο ομιλητή, που έχει ψιλότερη φωνή, τα formants έχουν μεγαλύτερη συχνότητα εμφάνισης. Επίσης, παρατηρούμε ότι η μέση τιμή του pitch στον δεύτερο ομιλητή είναι μεγαλύτερη, που συμφωνεί με τις παραπάνω παρατηρήσεις.

Βήμα 2

Στο αρχείο lib.py είναι υλοποιημένη η συνάρτηση data_parser, η οποία διαβάζει τα αρχεία ήχου (στο φάκελο digits) και επιστρέφει τα σήματα ήχου, τον ομιλητή και το ψηφίο που εκφωνείται.

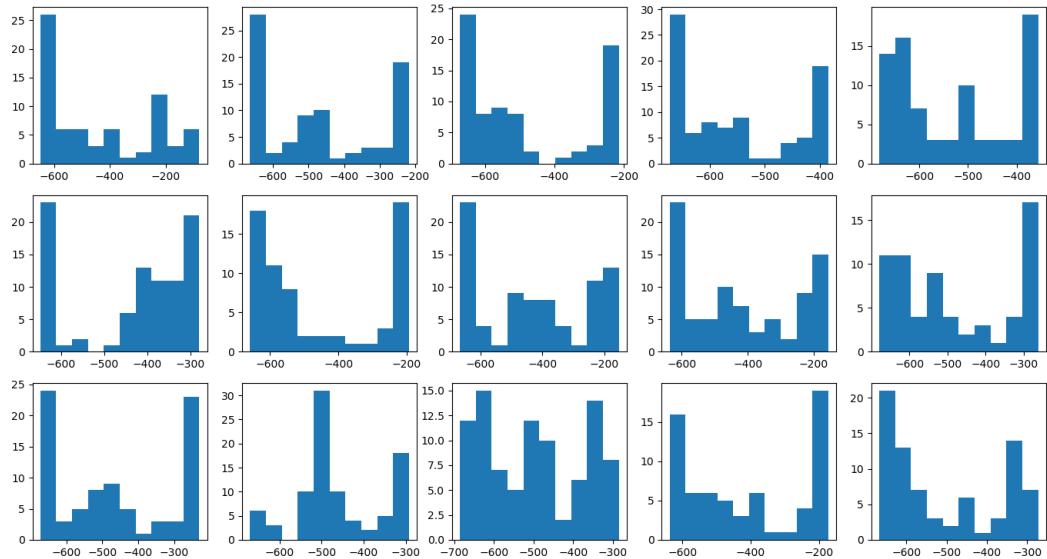
Βήμα 3

Για την εξαγωγή των Mel-Frequency Cepstral Coefficients (MFCC), ορίσαμε παράθυρο μήκους 25 ms και βήμα παραθύρου 10 ms, τα οποία μεταφράζονται σε αριθμό δειγμάτων, πολλαπλασιάζοντας με τη συχνότητα δειγματοληψίας των σημάτων ήχου. Εκτός από τους cepstral συντελεστές, εξήχθησαν και οι παράγωγοι αυτών, deltas και delta-deltas.

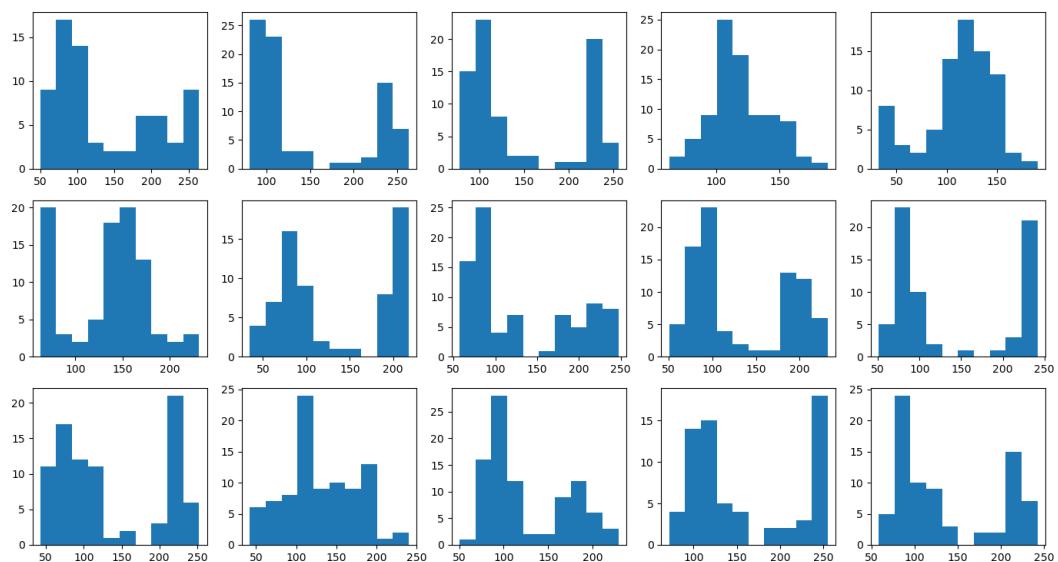
Βήμα 4

Παρακάτω φαίνονται γραφικά τα ιστογράμματα για τα ψηφία 5 και 4, από τους συντελεστές 0 και 1, για κάθε ομιλητή:

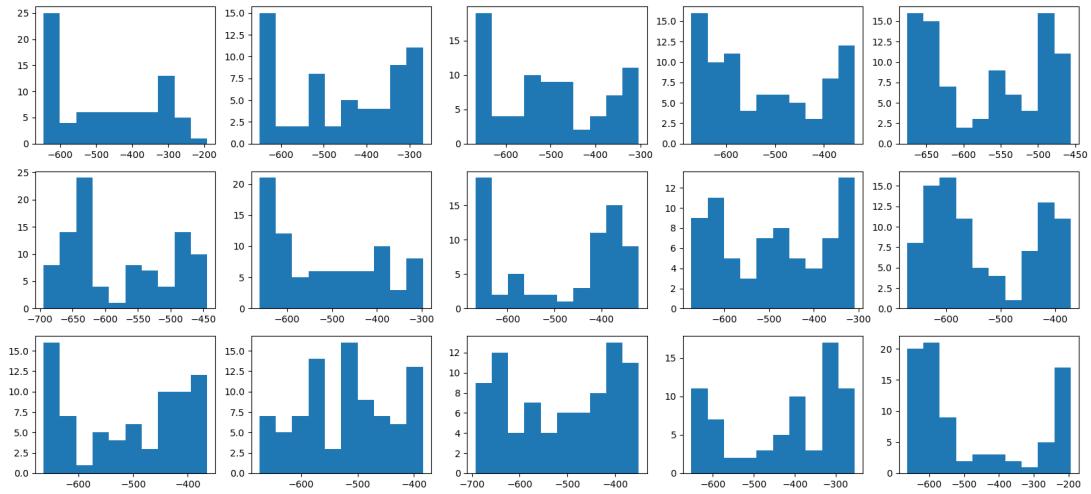
histograms of number five from every speaker coefficient 0



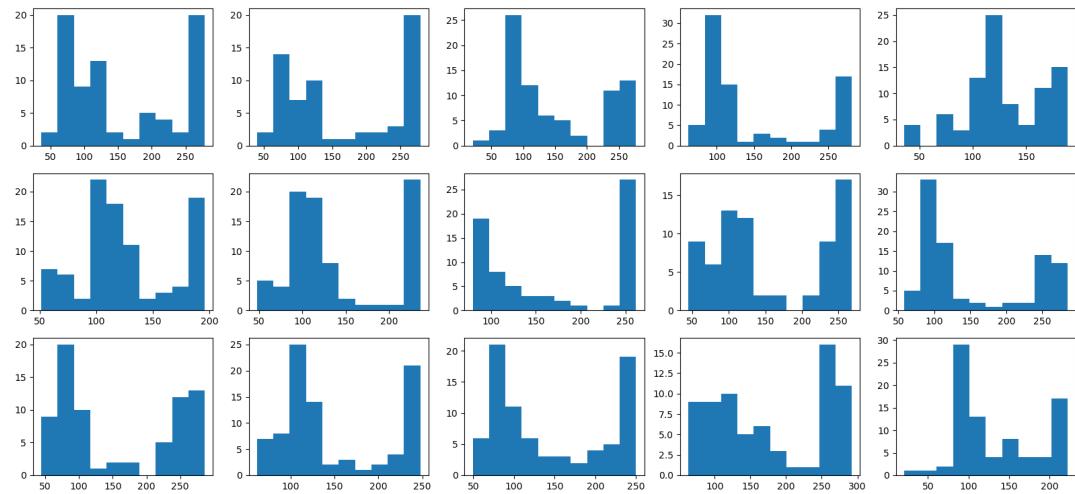
histograms of number five from every speaker coefficient 1



histograms of number four from every speaker coefficient 0



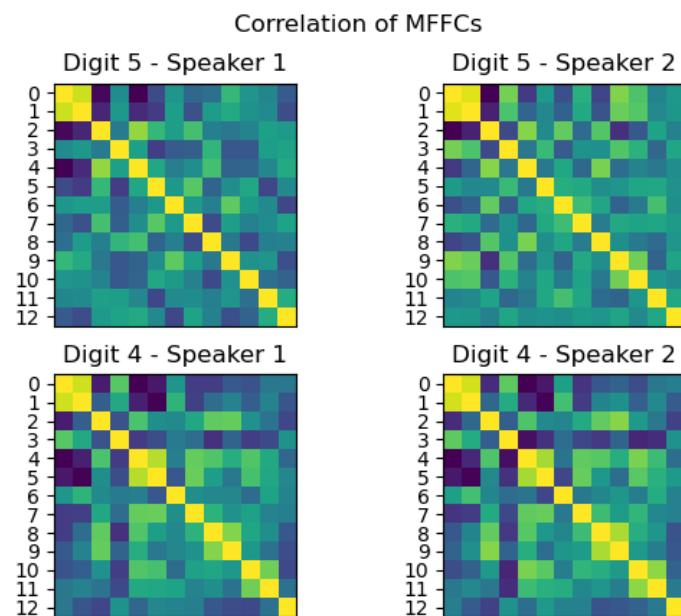
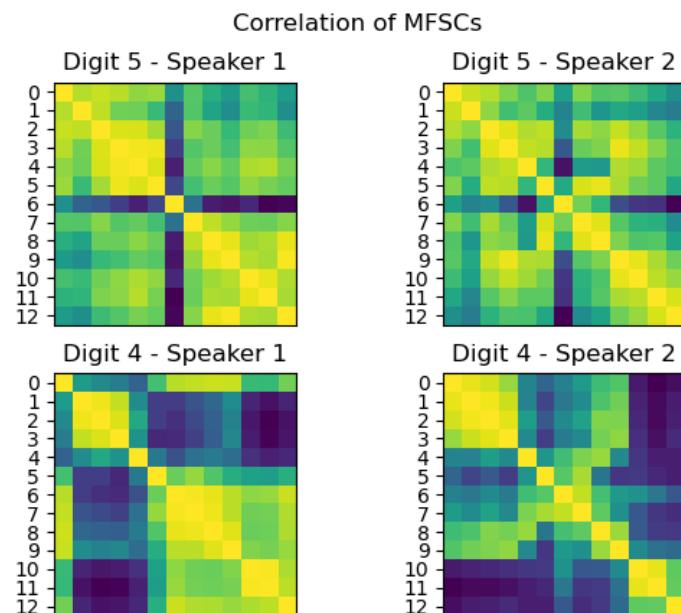
histograms of number four from every speaker coefficient 1



Τα παραπάνω ιστογράμματα αναπαριστούν το πλήθος εμφάνισης κάθε τιμής που παίρνει το διάνυσμα χαρακτηριστικών των συντελεστών. Τα ιστογράμματα παρουσιάζουν ποικιλομορφία για τα διαφορετικά bins, γεγονός που δείχνει ότι περιέχουν σημαντική πληροφορία. Επίσης, παρατηρούμε ότι για το ίδιο ψηφίο, υπάρχει ένα κοινό μοτίβο για κάθε συντελεστή mfcc στις διαφορετικές εκφωνήσεις του.

Στη συνέχεια, πειραματίζόμαστε με χρήση των συντελεστών Mel Frequency Spectral Coefficients (MFSC), που ουσιαστικά είναι οι συντελεστές που έχουν προκύψει όπως τους MFCC, χωρίς να έχει εφαρμοστεί ο τελικός μετασχηματισμός DCT επάνω τους. Γενικά, ο DCT συμβάλλει στην αποσυγχέτιση των coefficients.

Παρακάτω αναπαρίστανται γραφικά οι συσχετίσεις των MFSC και MFCC για τα ψηφία 4 και 5 και εκφωνήσεις από δύο ομιλητές:

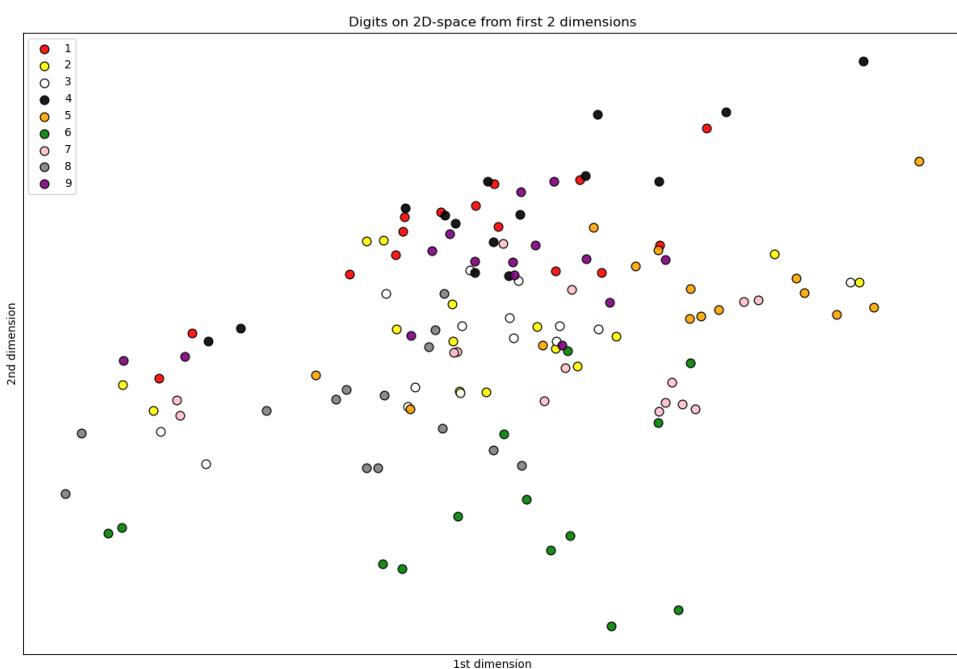


Στα παραπάνω πλέγματα, οι τιμές χυμαίνονται από το σκούρο μωβ (χαμηλές), μέχρι το κίτρινο (υψηλές) και δείχνουν τη συσχέτιση δύο συντελεστών, ανάλογα με τη γραμμή και τη στήλη. Αναμενόμενο είναι λοιπόν, η διαγώνιος να έχει τη μέγιστη συσχέτιση, αφού αφορά στον ίδιο συντελεστή.

Συγκρίνοντας τις δύο μεθόδους, παρατηρούμε ότι πράγματι οι MFCCs, οι οποίοι έχουν προκύψει από DCT, παρουσιάζουν μικρότερη συσχέτιση (πιο σκούρα χρώματα), σε σύγκριση με τους MFSCs (πιο ανοιχτά χρώματα, προς το κίτρινο). Αυτός είναι και ο λόγος, για τον οποίο προτιμάμε τη χρήση των MFCCs.

Βήμα 5

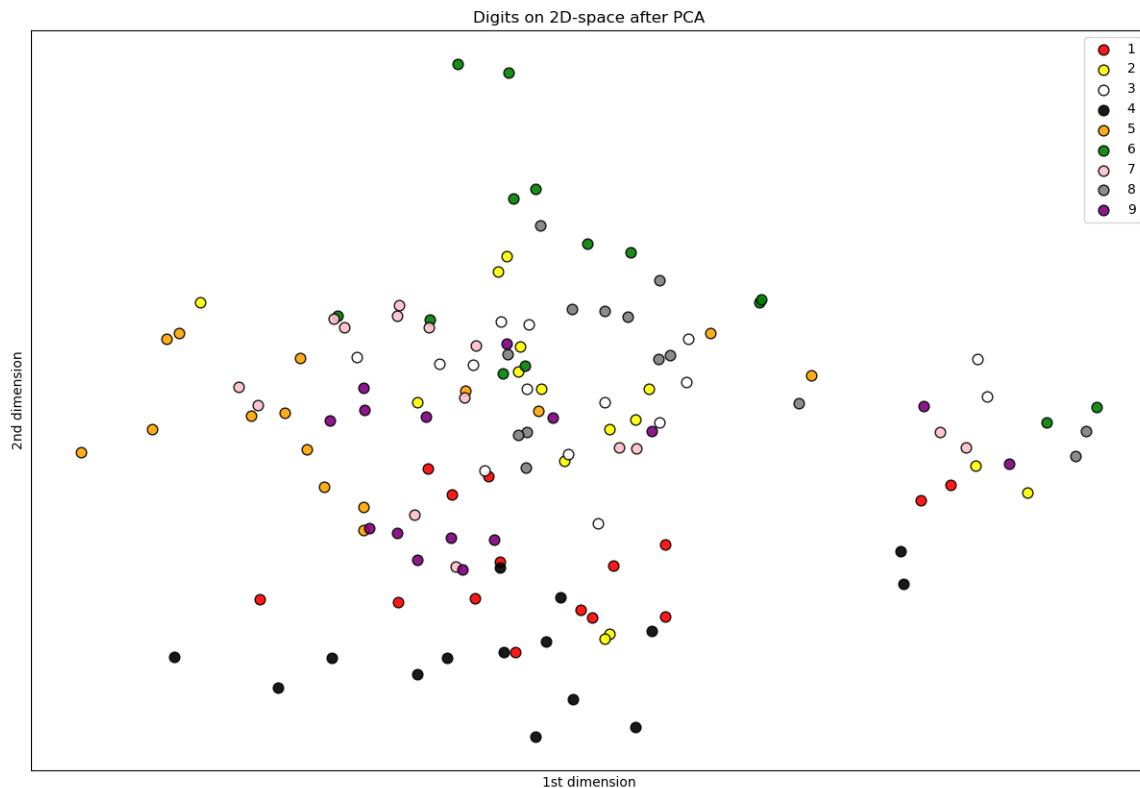
Στη συνέχεια, δημιουργούμε ένα διάνυσμα χαρακτηριστικών για κάθε ψηφίο. Αυτό το κάνουμε, συνενώνοντας τις μέσες τιμές και τις τυπικές αποκλίσεις των συντελεστών MFCC, των deltas και των delta-deltas, οπότε προκύπτει για κάθε δείγμα ένα διάνυσμα μεγέθους $2 \times 3 \times 13 = 78$. Παρακάτω γίνεται μια απλή απεικόνιση των δειγμάτων στο επίπεδο, χρησιμοποιώντας τις 2 πρώτες διαστάσεις των διανυσμάτων αυτών:



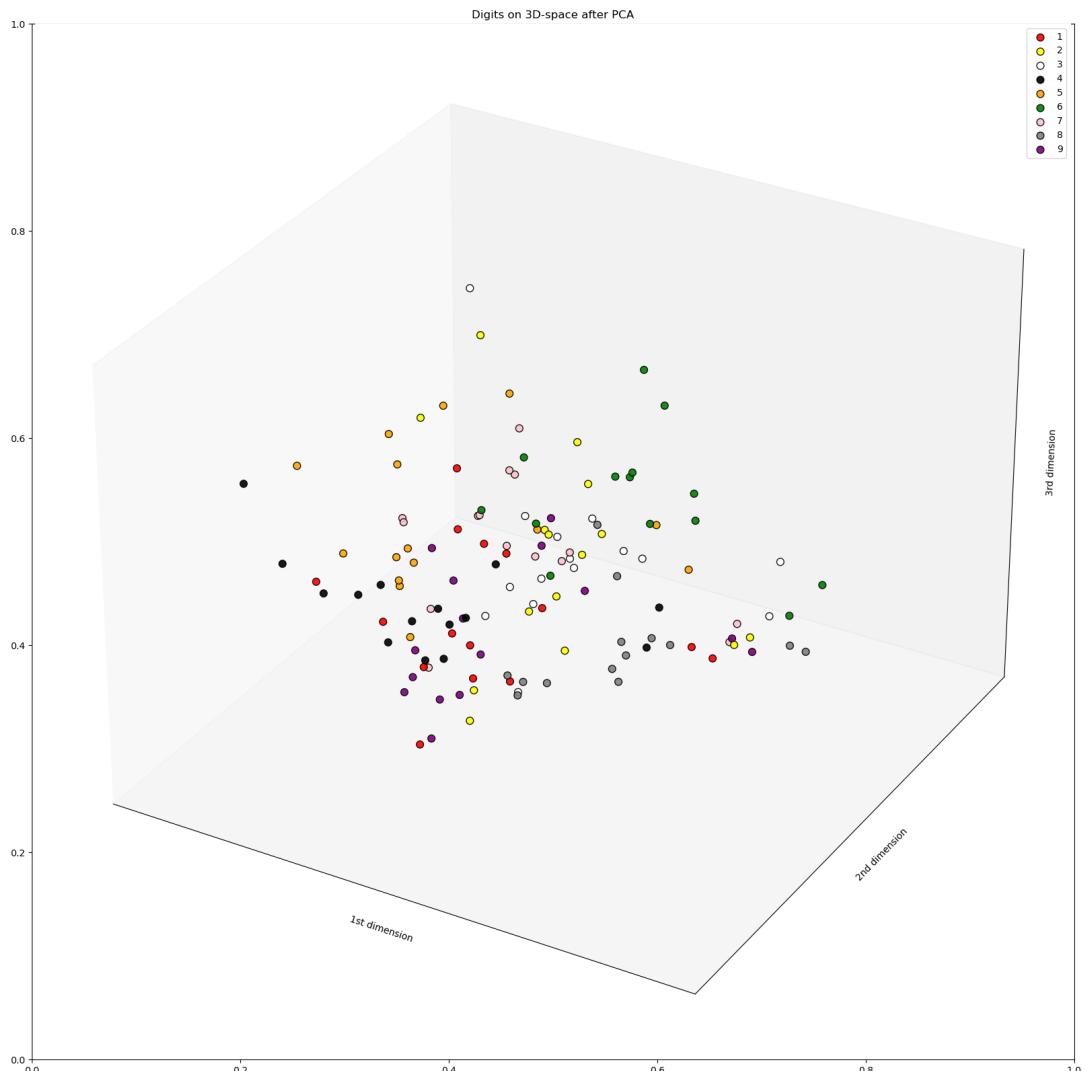
Παρατηρούμε ότι με αυτή την αναπαράσταση των δειγμάτων, , διαφορετικές εκφωνήσεις του ίδιου ψηφίου κατανέμονται πολύ διάσπαρτα, ενώ παράλληλα επικαλύπτονται σε σημαντικό βαθμό με εκφωνήσεις άλλων ψηφίων. Συνεπώς, δεν είναι ένα καλό σύνολο δεδομένων για να εκπαιδευτεί ένας ταξινομητής.

Βήμα 6

Χρησιμοποιώντας Ανάλυση Πρωτεύουσων Συνιστώσων (PCA), αποσκοπούμε σε μια συμπίεση της διαστατικότητας των δειγμάτων, διατηρώντας όσο περισσότερη πληροφορία μπορούμε. Παρακάτω εφαρμόζεται στα δεδομένα με 2 συνιστώσες:



Μετά την ανάλυση σε 2 πρωτεύουσες συνιστώσες, η προβολή των δειγμάτων στο επίπεδο αναμένουμε να είναι πολύ καλύτερη, το οποίο και αντικατοπτρίζεται στο παραπάνω γράφημα. Οι εκφωνήσεις του ίδιου ψηφίου είναι κοντινότερα μεταξύ τους και παρουσιάζουν μικρότερη επικάλυψη με άλλων ψηφίων. Συνεχίζουμε με εφαρμογή του PCA στα δεδομένα με 3 συνιστώσες:



Διατηρώντας, τώρα, τις 3 κύριες συνιστώσες των διανυσμάτων χαρακτηριστικών των δειγμάτων μας, παρατηρούμε ακόμα καλύτερο διαχωρισμό των κλάσεων των ψηφίων στο χώρο.

Προβάλλοντας στις 2 διαστάσεις, διατηρούμε αθροιστικά περίπου το 70% της πληροφορίας ($58.49\% + 12.01\%$ από κάθε συνιστώσα), ενώ στις 3, περίπου το 80% ($58.49\% + 12.01\% + 10.94\%$). Αυτό σημαίνει ότι το χάνουμε το 20-30% της πληροφορίας μας, το οποίο δεν είναι επιψυμητό και θα μειώσει αρκετά την επίδοση των ταξινομητών, που θα εκπαιδευτούν πάνω στα προβεβλημένα δείγματα. Άρα, χρίνουμε την μείωση των διαστάσεων, όχι ιδιαίτερα ικανοποιητική.

Βήμα 7

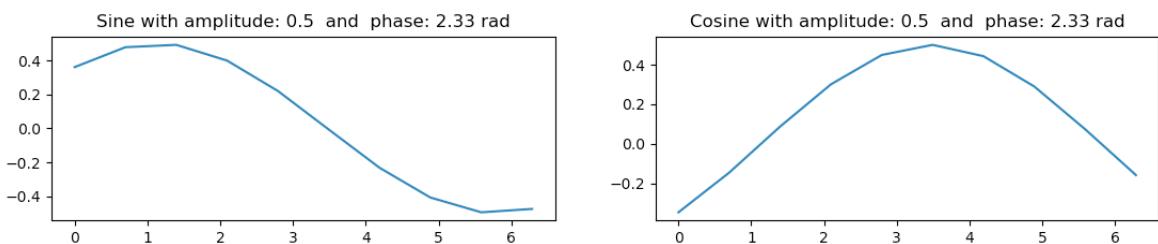
Θα προχωρήσουμε στον πειραματισμό με διάφορα μοντέλα για κατηγοριοποίηση τυχαίων δειγμάτων από σήματα φωνής που διαθέτουμε. Στον πίνακα που ακολουθεί, συνοψίζουμε την ακρίβεια ταξινόμησης του κάθε μοντέλου, καθώς και τις παραμέτρους που χρησιμοποιήθηκαν και αν τα δεδομένα κανονικοποιήθηκαν πριν το training:

| Model | Normalized Data | Accuracy (%) |
|-----------------------------|-----------------|--------------|
| Custom Gaussian Naive Bayes | Yes | 60 |
| Scikit Gaussian Naive Bayes | No | 62.5 |
| Scikit Gaussian Naive Bayes | Yes | 57.5 |
| K($=1$) Nearest Neighbors | No | 80 |
| K($=1$) Nearest Neighbors | Yes | 80 |
| K($=2$) Nearest Neighbors | No | 65 |
| K($=2$) Nearest Neighbors | Yes | 62.5 |
| SVM with linear kernel | No | 77.5 |
| SVM with linear kernel | Yes | 12.5 |
| SVM with rbf kernel | No | 12.5 |
| SVM with poly kernel | No | 12.5 |
| MLP (50,25) with ReLU | No | 67.5 |
| MLP (50,25) with ReLU | Yes | 75 |

Παρατηρούμε, πως τα καλύτερα αποτελέσματα τα δίνει ο αλγόριθμος KNN-1 με ακρίβεια 80%, ενώ εξίσου υψηλές δίνει ο MLP σε κανονικοποιημένα δεδομένα και ο SVM με linear kernel σε μη-κανονικοποιημένα δεδομένα. Στα κανονικοποιημένα, παρατηρούμε για τον συγκεκριμένο πολύ κακή ακρίβεια με διάφορα kernels, ενώ για τους άλλους ταξινόμητές, βλέπουμε παρόμοιες επιδόσεις είτε γίνει είτε δε γίνει το scaling. Η καλύτερη επιλογή θα ήταν ο SVM ή ο MLP, και ίσως όχι ο KNN-1, αφού μπορεί ένα outlier να καθορίσει πλήρως την ταξινόμηση κάποιων δειγμάτων και να μην έχουμε πάντα τόσο υψηλή ακρίβεια.

Βήμα 8

Στη συνέχεια, προχωράμε με τη δημιουργία ενός συνόλου δεδομένων, όπου τα δεδομένα εκπαίδευσης θα είναι ακολουθίες 10 σημείων από ημίτονα και η έξοδος-στόχος, ακολουθίες, πάλι 10 σημείων, από συνημίτονα. Δημιουργήσαμε 20 ζευγάρια τέτοιων ακολουθιών με τυχαίο πλάτος στο διάστημα $(0, 1)$ και τυχαία αρχική φάση στο $(0, 2\pi)$, ένα παράδειγμα από τα οποία φαίνεται παρακάτω:



Σκοπός μας είναι η δημιουργία ενός Αναδρομικού Νευρωνικού Δικτύου (RNN), το οποίο θα δέχεται ως είσοδο τα ημίτονα και θα προβλέψει τα συνημίτονα.

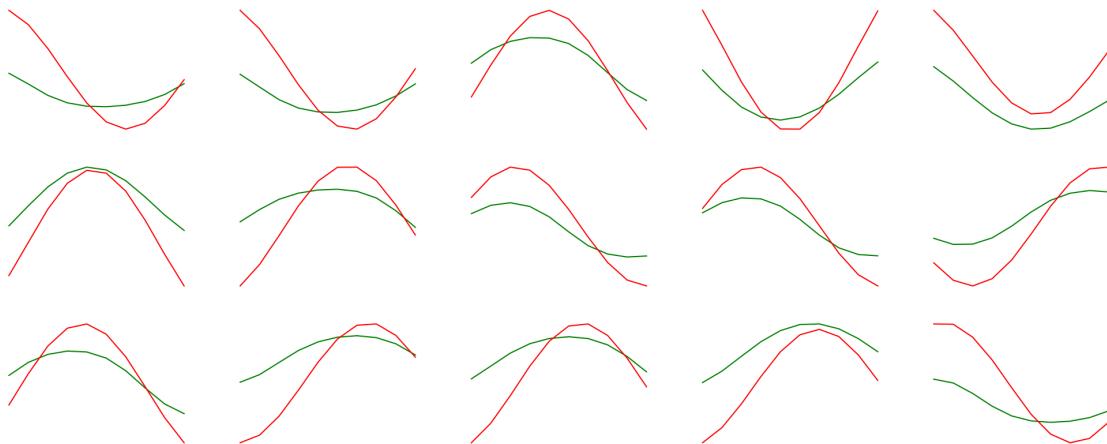
Επιλέξαμε μια μονάδα LSTM, που είναι μια πιο ειδική μορφή RNN, για ένα πλήθος πλεονεκτημάτων που έχει ως δομή. Ο κύριος λόγος είναι, επειδή μπορούν να “χαταλάβουν” εξαρτήσεις μεταξύ κάποιων sequences (πχ. λέξεων), οι οποίες μπαίνουν ως είσοδο στο δίκτυο με μεγάλη χρονική διαφορά, κάτι στο οποίο τα Vanilla RNNs αδυνατούν. Αυτή η ιδιότητά τους είναι “έμφυτη”, γιατί σχεδιάστηκαν για αυτό ακριβώς το σκοπό.

Για την εκπαίδευση του μοντέλου μας, χρησιμοποιήσαμε τις εξής υπερπαραμέτρους:

- learning rate: 0.001
- epochs: 4500
- optimizer: Adam

και MSE για τη συνάρτηση σφάλματος, η τιμή της οποίας έπεφτε περίπου μέχρι 0.08 κατά το training, αλλά και το testing.

Παραχάτω φαίνονται κάποιες προβλέψεις (πράσινο) του μοντέλου πάνω στο αντίστοιχο ground truth (κόκκινο):



Παρατηρούμε ότι το μοντέλο προσπαθεί να πλησιάσει το στόχο, σχηματίζοντας στις περισσότερες περιπτώσεις σωστά το σχήμα της κυματομορφής, αλλά τα αποτελέσματα δεν είναι ικανοποιητικά και απέχουν από την ακριβή μορφή.

Κυρίως Μέρος Εργαστηρίου

Στο κυρίως μέρος του εργαστηρίου ωστε χρησιμοποιήσουμε το Free Spoken Digit Dataset, που περιέχει περισσότερα δεδομένα ανάγνωσης των ψηφίων 0 έως 9.

Βήμα 9

Αρχικά, χωρίσαμε τα δεδομένα σε train και test, με βάση τη συνάρτηση parser που μας δόθηκε και στη συνέχεια χωρίσαμε το train set σε train και validation με αναλογία 80/20%, ώστε να είναι stratified.

Βήμα 10, 11

Θα αναπτύξουμε Κρυφά Μαρκοβιανά Μοντέλα (HMM) με Μοντέλα Μείγματος Γκαουσιανών (GMM), με σκοπό την αναγνώριση των ψηφίων. Συγκεκριμένα, για κάθε δείγμα φωνής, αφού εξαχθούν οι συντελεστές MFCC, ωστε να χρησιμοποιήσουμε ένα αριθμό καταστάσεων για το κάθε HMM, και ένα αριθμό GMM για την κάθε κατάσταση.

Ορίσαμε, αρχικά, τον πίνακα μεταβάσεων και τις αρχικές και τελικές πιθανότητες των καταστάσεων, όπως αναφέρονται στην εκφώνηση. Επίσης, χωρίσαμε τα δεδομένα κάθε ψηφίου, που προκύπτουν από την εξαγωγή των MFCC, ώστε να εκπαιδεύσουμε 10 διαφορετικά μοντέλα GMM-HMM, ένα για κάθε ψηφίο.

Μετά τη δημιουργία των GMM για κάθε state και τον ορισμό των HMM, γίνεται χρήση του αλγορίθμου Expectation Maximization με μέγιστο αριθμό επαναλήψεων 5, για να γίνει το training για κάθε ψηφίο.

Βήμα 12

Η τελική πρόβλεψη γίνεται διαλέγοντας, εκείνη την πρόβλεψη του ενός από τα δέκα HMM του μοντέλου μας, η οποία δίνει το μέγιστο λογάριθμο της πιθανοφάνειας (log likelihood).

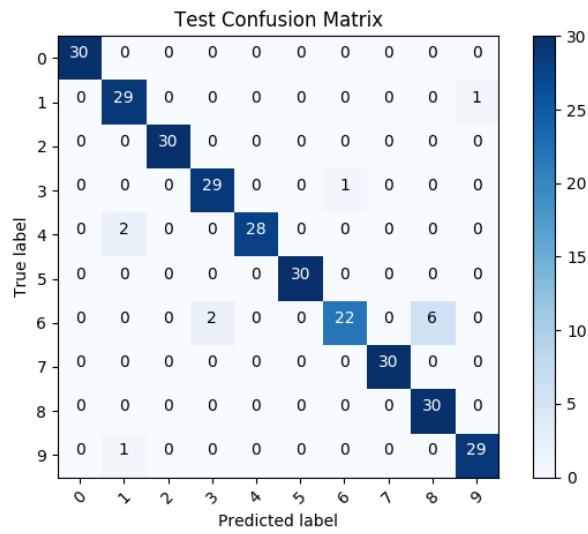
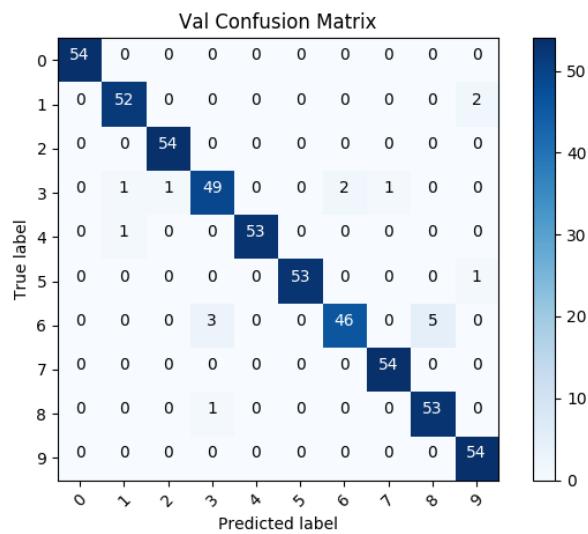
Πειραματιστήκαμε με τις τιμές 1 και 2 για τις καταστάσεις των HMM (με μεγαλύτερο αριθμό είχαμε σφάλματα λόγω ιδιομορφίας της βιβλιοθήκης pomegranate) και με τις τιμές 2, 3 και 4 για τα GMM. Κάναμε δηλαδή όλη την παραπάνω διαδικασία, για κάθε συνδυασμό των παραμέτρων, πάνω στο validation set. Αυτό, το κάνουμε, γιατί αν βελτιστοποιήσουμε τις παραμέτρους μας στο test set, δηλαδή ψάχουμε το μοντέλο το οποίο δίνει την καλύτερη ακρίβεια προβλέψεων, τότε το μοντέλο μας ωστε να είναι biased. Στην πραγματικότητα, το test set δεν το έχουμε στη διάθεσή μας, οπότε έτσι πρέπει να φερθούμε και εδώ, δηλαδή να το χρησιμοποιήσουμε μόνο για το evaluation και για να μας δώσει απλά ένα νούμερο αξιολόγησης.

Άρα, τελικά, διαλέγουμε το ένα από τα 6 μοντέλα, του οποίου τα 10 HMM έδωσαν τις καλύτερες ακρίβειες στο validation set. Παρακάτω φαίνεται ένας πίνακας με τις ακρίβειες των διαφορετικών μοντέλων που δοκιμάστηκαν:

| Accuracy (%) | HMM hidden states | 1 | 2 |
|---------------|-------------------|--------------|-------|
| GMM gaussians | | | |
| 2 | | 96.11 | 69.44 |
| 3 | | 66.96 | 95.56 |
| 5 | | 29.63 | 87.22 |

Βήμα 13

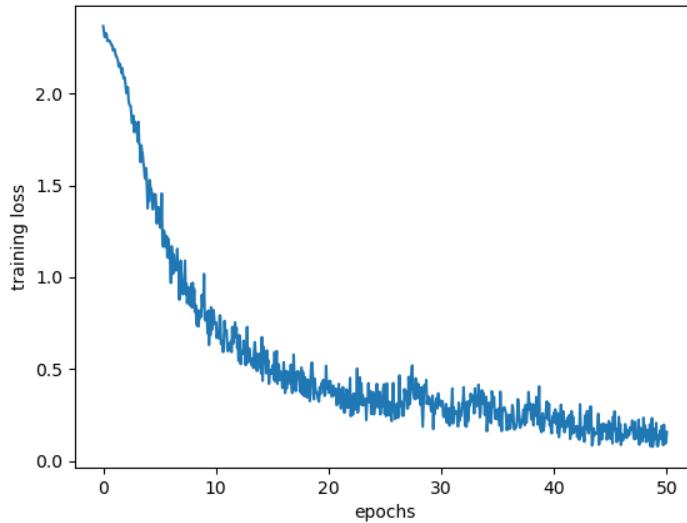
Το μοντέλο με τη μεγαλύτερη ακρίβεια στο validation set αξιολογείται, στη συνέχεια, στο test set. Συγκεκριμένα το μοντέλο με παραμέτρους 1, 2 για HMM hidden states και GMM gaussians, αντίστοιχα, είχε ακρίβεια 95.67% στο test set. Παρακάτω φαίνονται οι Confusion Matrices για την απόδοση του στο validation και στο test set:



Παρατηρούμε ότι πράγματι τα σύνολα δεδομένων μας είναι stratified και οι προβλέψεις είναι πολύ ακριβείς.

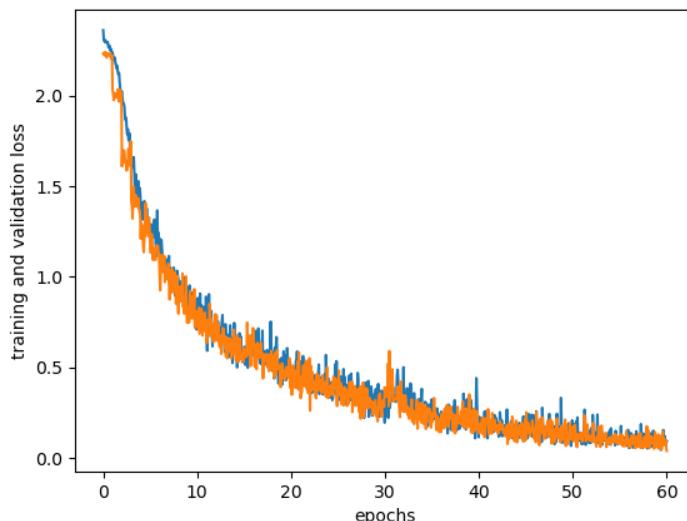
Βήμα 14

Αρχικά, εκπαιδεύσαμε το RNN δίκτυο μας με το training set και παρατηρήσαμε μόνο το training loss το οποίο υπολογίστηκε με την μέθοδο cross entropy. Παρακάτω φαίνεται ο ρυθμός μείωσης του loss για το training στις 50 εποχές:

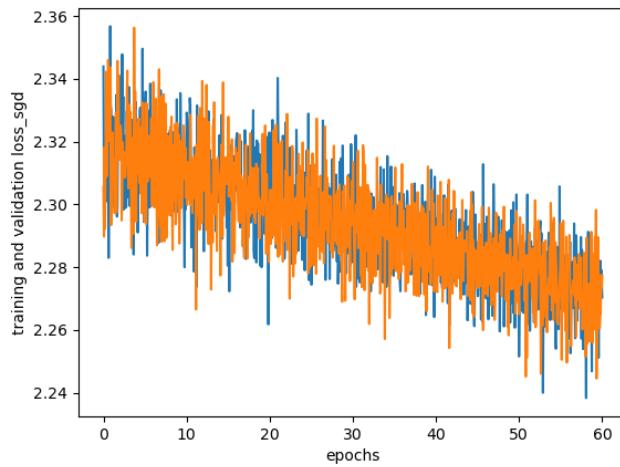


Σχήμα 10: Απλό training loss

Ήδη από το παραπάνω γράφημα φαίνεται ότι μειώνεται αρκετά ικανοποιητικά το training loss αν και παρατηρούνται αρκετές αστάθμειες στην καμπύλη, κάτι που οφείλεται και στον ADAM optimizer που χρησιμοποιούμε, ο οποίος έχει μεγάλο ρυθμό σύγκλισης. Παρακάτω εμφανίζεται και το loss επάνω στο validation set στην ίδια γραφική με το training loss πρώτα με ADAM optimizer και στην συνέχεια με τον SGD.

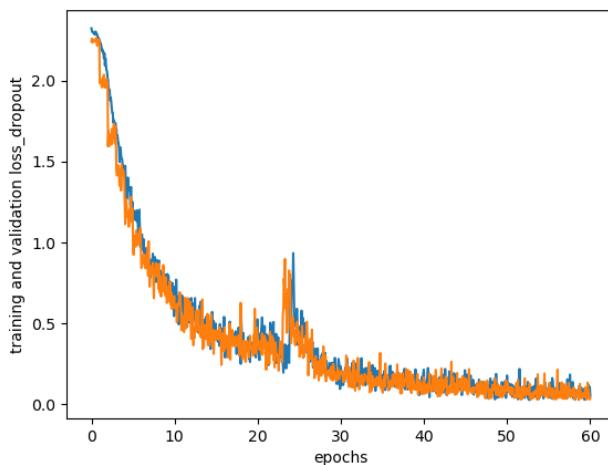


Σχήμα 11: Validation και training loss με ADAM



Σχήμα 12: Validation και training loss με SGD

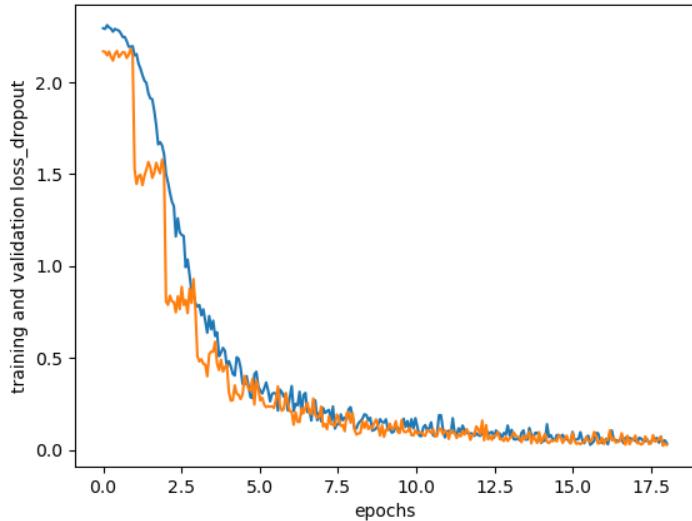
Αυτό που παρατηρούμε είναι ότι με τον optimizer adam έχουμε αρκετά καλύτερη σύγκλιση σε σχέση με τον SGD τόσο στο training όσο και στο validation loss. Τόσο το validation όσο και το training loss ήταν της τάξης του 0.05 μετά τις 60 εποχές εκπαίδευσης. Στο επόμενο βήμα, εκπαιδεύσαμε το νευρωνικό μας δίκτυο χρησιμποιώντας τις τεχνικές Dropout και L2 Regularization. Το Dropout σε κάθε κύκλο εκπαίδευσης θέτει ένα ποσοστό βαρών ίσο με 0 έτσι ώστε να μην κάνει κάθε βάρος training με όλα τα δεδομένα και έτσι το μοντέλο να αποφύγει πιθανό overfitting. Στον κώδικα, εφαρμόζεται δίνοντας την συγκεκριμένη παράμετρο την στιγμή δημιουργίας του μοντέλου και εμείς δώσαμε την τιμή 0.2. Επίσης, το L2 Regularization χρησιμποιείται με σκοπό να μην γίνεται υπερεκπαίδευση σε κάποιο κύκλο εκπαίδευσης και έτσι το μοντέλο να μπορεί τελικά να γενικεύει καλύτερα. Στον κώδικα αυτό το υλοποιήσαμε με την παράμετρο weight decay στον optimizer. Παρακάτω φαίνεται η γραφική των training και validation loss για το συγκεκριμένο μοντέλο.



Σχήμα 13: Validation και training loss με Dropout και L2 Regularization

Όπως και στις προηγούμενες γραφικές, έτσι και εδώ παρατηρείται ένας έντονος κυματισμός στην γραφική παράσταση του loss η οποία όμως όλο και μικραίνει και φαίνεται να συγκλίνει γύρω από την τιμή 0.03. Επίσης, το συγκεκριμένο δίκτυο το αξιολογήσαμε και με το test set, και είχε ποσοστό επιτυχίας 96 τοις εκατό κάτι που δείχνει ότι το συγκεκριμένο πρόβλημα είναι εύκολο και υπάρχει καλή προσαρμογή του δικτύου.

Επόμενο βήμα είναι να προσθέσουμε στο δίκτυο την τεχνική early stopping η οποία βοηθάει στο να μην εκπαιδεύουμε το δίκτυο για πολλές εποχές χωρίς να πετυχαίνουμε κάποια αξιοσημείωτη βελτίωση κάτι που έχει ως αποτέλεσμα σπατάλη πόρων και υπερεκπαίδευση του δικτύου. Επίσης εδώ υλοποιήσαμε το rnn bidirectional έτσι ώστε να γίνεται εκπαίδευση και προς τις δύο κατευθύνσεις. Αυτό που είναι καλύτερο σε αυτό το μοντέλο σε σχέση με το απλό forward μοντέλο είναι ότι για την εκπαίδευση μιας κατάστασης χρησιμοποιείται πληροφορία και από μελλοντικές καταστάσεις και όχι μόνο από προηγούμενες όπως γινόταν στο forward. Παρακάτω φαίνεται η γραφική για το training και το validation loss.



Σχήμα 14: Training και validation loss με bidirectional RNN και early stopping

Από την παραπάνω γραφική εύκολα παρατηρεί κανείς ότι το bidirectional μοντέλο καταφέρνει πολύ πιο γρήγορα να εκπαιδευτεί κάτι που είναι καλό καθώς μαζί με την τεχνική του early stopping γλυτώνουμε σημαντικό χρόνο εκπαίδευσης. Το παραπάνω μοντέλο δοκιμάστηκε και στο test set όπου και αυτό είχε ποσοστό επιτυχίας 96 τοις εκατό αλλά όπως τονίσαμε με σημαντικά μικρότερο χρόνο εκπαίδευσης.