|  |
| --- |
| Χατζηθεοδώρου Ιάσων 03117089 Κουνούδης Δημήτρης 03117169 |
| Αναγνώριση Προτύπων |
| 2η Εργαστηριακή Άσκηση |

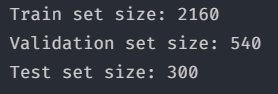
|  |
| --- |
|  |

## Εργαστηριακό Μέρος

Σε αυτό το μέρος θα χρησιμοποιηθεί ένα μεγαλύτερο dataset (Free Spoken Digit Dataset) και θα ακολουθήσουμε παρόμοι διαδικασία έτσι ώστε να εξάγουμε τα MFCC τα οποία θα χρησιμοποιηθούν για την εκπαίδευση των μοντέλων και την αναγνώριση των ψηφίων.

Βήμα 9

Σε αυτό το βήμα κατεβάσαμε το dataset και το εισάγαμε χρησιμοποιώντας την συνάρτηση parser. Χωρίσαμε επιπλέον τα train δεδομένα σε train και validation προσέχοντας να επιλέξουμε stratified split, ώστε να είναι ισορροπημένα τα δύο σύνολα που προέκυψαν. Επίσης, χρησιμοποιώντας την joblib.dump αποθηκεύσαμε τα χωρισμένα δεδομένα στο δίσκο ώστε να μην χρειάζεται να επαναλαμβάνεται η παραπάνω διαδικασία. Τα μεγέθη των splits φαίνονται παρακάτω



Επίσης, παρατηρήσαμε ότι τα δείγματα είχαν διαφορετικό πλήθος γραμμών το κάθε ένα, αλλά σταθερό πλήθος στηλών ίσο με 6.

Βήμα 10

Για την υλοποίηση του HMM μοντέλου χρησιμοποιήσαμε την βιβλιοθήκη pomegranate και πιο συγκεκριμένα το GeneralMixtureModel, MultivariateGaussianDistribution και ΗiddenMarkovModel. Ο πίνακας μετάβασης έχει μηδενικά σε όλες τις θέσεις εκτός από τις διαδοχικές, δηλαδή της μορφής στις οποίες δίναμε ίσες πιθανότητες μετάβασης, 0.5. Το μοντέλο που υλοποιήθηκε ήταν της μορφής left-right, οπότε ο αρχικός πίνακας έδινε πιθανότητα 1 στην πρώτη κατάσταση ενώ ο τελικός πίνακας έδινε πιθανότητα 1 στην τελευταία κατάσταση. Τέλος, επιλέξαμε να θέσουμε max\_iterations=5 κατά την εκπαίδευση, καθώς αυτή η επιλογή μείωνε τον χρόνο εκτέλεσης και δεν επηρέαζε σημαντικά την απόδοση.

Βήμα 11

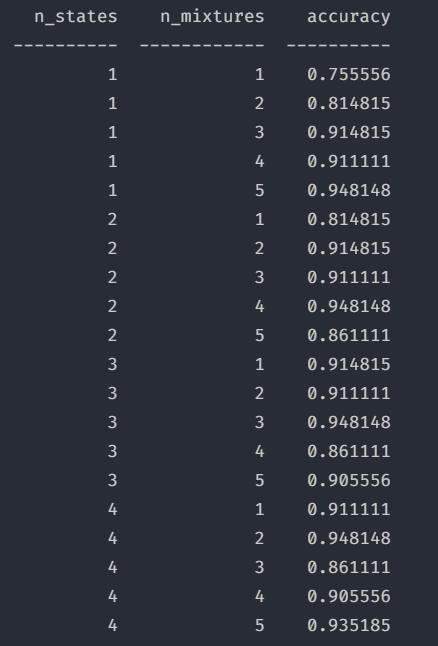
Για να μπορέσουμε να εκπαιδεύσουμε το μοντέλο, πρώτα φτιάξαμε μια λίστα από arrays, που περιείχε τα δείγματα από κάθε ψηφίο. Έπειτα υλοποιήσαμε την train\_models, η οποία εκπαιδεύει το HMM με τις τιμές των υπερπαραμέτρων που της δίνονται.

Βήμα 12

Σε αυτό το βήμα εκπαιδεύσαμε τα παραπάνω μοντέλα και τα αξιολογήσαμε χρησιμοποιώντας το validation dataset. Ο λόγος που ακολουθούμε αυτή τη μεθοδολογία είναι πως το μοντέλο μας δεν πρέπει να έχει πρόσβαση στα δεδομένα test μέχρι να τελειώσει η εκπαίδευσή του και η βελτιστοποίηση των υπερπαραμέτρων του. Σε αντίθετη περίπτωση κινδυνεύουμε να έχουμε χρησιμοποιήσει πληροφορίες που υπό κανονικές συνθήκες δεν θα είχαμε διαθέσιμες και άρα να πάρουμε καλύτερα αποτελέσματα που δεν γενικεύονται. Για την αξιολόγηση χρησιμοποιήθηκε ο αλγόριθμος Viterbi. Οι υπερπαράμετροι που δοκιμάστηκαν ήταν οι παρακάτω

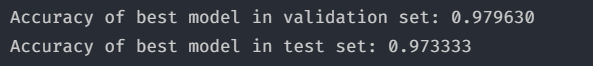


Τα αποτελέσματα που πήραμε για αυτές τις υπερπαραμέτρους ήταν τα εξής



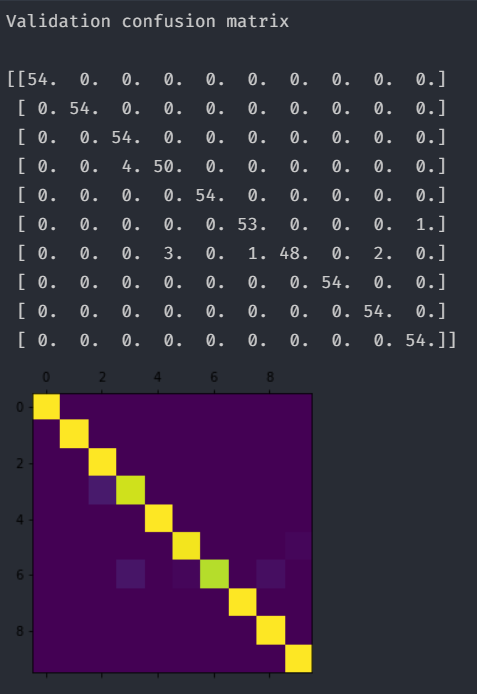
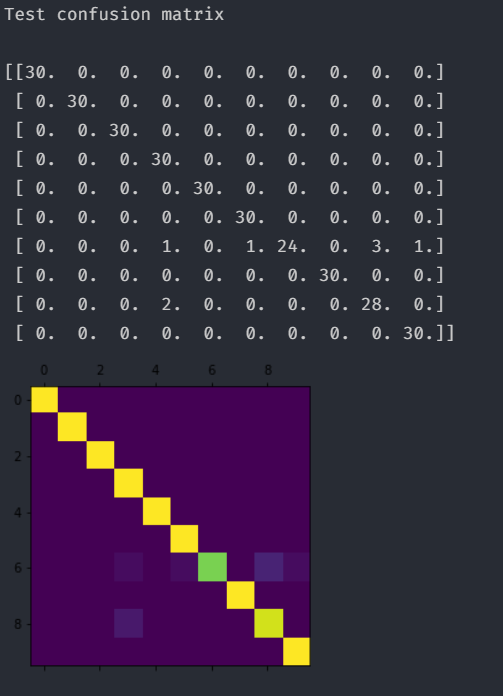
Όπως φαίνεται υπάρχουν 4 διαφορετικοί συνδυασμοί για τους οποίους πετυχαίνουμε το μέγιστο accuracy: (1, 5), (2, 4), (3,3), (4, 2)

Για την τελική αξιολόγηση του μοντέλου χρησιμοποιήθηκε το test dataset στο μοντέλο (3, 3) και πήραμε τα παρακάτω αποτελέσματα που δείχνουν αρκετά καλή ακρίβεια.



Βήμα 13

Για την παρουσίαση του confusion matrix τυπώσαμε και τον ίδιο τον πίνακα αλλά και τη γραφική παράσταση χρησιμοποιώντας την pyplot.matshow()



Βήμα 14

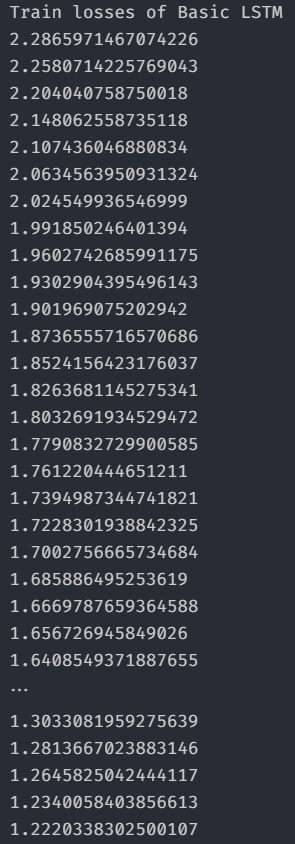
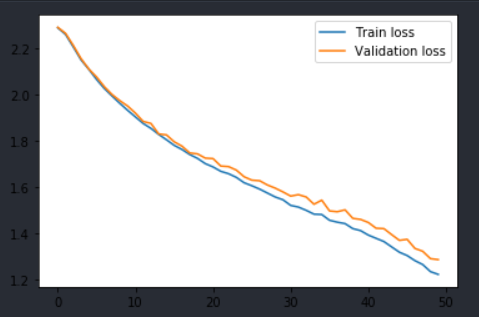
Στο συγκεκριμένο βήμα χρησιμοποιήθηκε ως μοντέλο το LSTM και διάφορες παραλλαγές του. Για την εκπαίδευση και την αξιολόγηση των μοντέλων υλοποιήθηκαν οι συναρτήσεις train\_epoch(), η οποία προχωράει κατά μία εποχή την εκπαίδευση του μοντέλου, η eval\_dataset() η οποία αξιολογεί το δεδομένο μοντέλο σε κάποιο dataset ως προς μια συνάρτηση κόστους, έτσι ώστε να επιστρέψει τις προβλέψεις του μοντέλου και η train\_lstm η οποία εκπαιδεύει πλήρως το μοντέλο και το επιστρέφει μαζί με τα losses ανά εποχή στα train και validation δεδομένα.

Για τη μετατροπή των δεδομένων σε μορφή κατάλληλη για το PyTorch χρησιμοποιήθηκε η κλάση FrameLevelDataset η οποία κάνει zero pad στους πίνακες ώστε όλοι να έχουν ίδιο αριθμό γραμμών με την numpy.pad.

Το μοντέλο αποτελείται από δύο layers, το LSTM που παίρνει την είσοδο και ένα γραμμικό μετασχηματισμό για την έξοδο.

Παρακάτω αναλύουμε τα αποτελέσματα ανά μοντέλο.

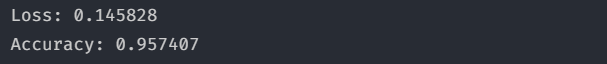
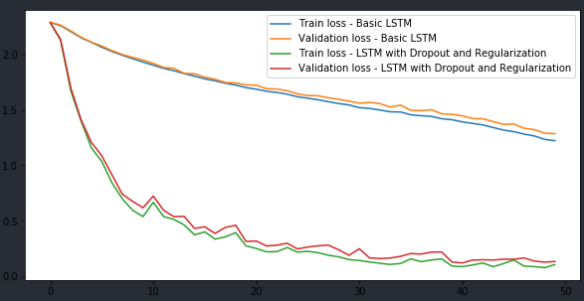
1. **Βασικό Μοντέλο**



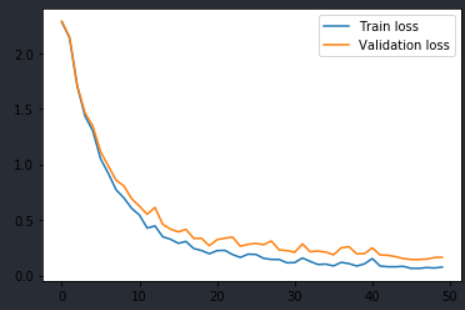


1. **Dropout – Regularization**

Σε αυτό το σημείο προσθέσαμε Dropout και L2 Regularization στο μοντέλο ώστε να αποτρέψουμε το overfitting. Πιο συγκεκριμένα, με το dropout αφαιρούμε μονάδες του νευρωνικού με τυχαίο τρόπο, ενώ με το regularization τιμωρούμε επιπλέον τα βάρη στην περίπτωση της λανθασμένης πρόβλην. Στην υλοποίησή μας ππιλέξαμε dropout = 0.2 και weight\_decay = 0.001.



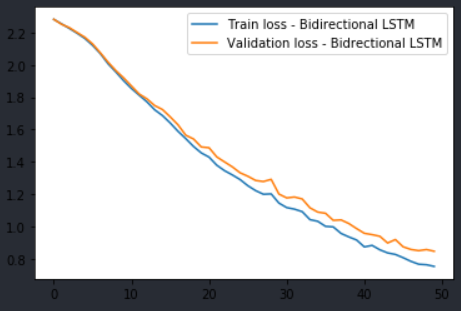
1. **Early Stopping**

Παρακάτω θα χρησιμοποιήσουμε την τεχνική Early Stopping , η οποία συμβάλλει στη μείωση του overfitting, σταματώντας την εκπαίδευση αν το validation loss δεν βελτιώνεται για πολύ συνεχόμενο χρόνο Επιλέξαμε να σταματάμε όταν 10 συνεχόμενες έξοδοι έχουν loss μικρότερο από το ελάχιστο, στο validation set.



1. **Bidirectional**

Παρακάτω δοκιμάζουμε να χρησιμοποιήσουμε το Bidirectional LSTM. Η διαφορά του με τα προηγούμενα είναι πως τρέχει τα δεδομένα από το παρελθόν στο μέλλον και από το μέλλον στο παρελθόν. Με αυτόν τον τρόπο και χρησιμοποιώντας τα hidden states το μοντέλο μπορεί κάθε στιγμή να έχει πρόσβαση σε πληροφορίες που προέρχοναι και από το παρελθόν και το μέλλον.



Όπως φαίνεται το καλύτερο μοντέλο ήταν αυτό που προέκυψε με Early Stopping. Παρακάτω φαίνονται με μεγαλύτερη λεπτομέρεια τα χαρακτηριστικά του και η αξιολόγησή του στα test δεδομένα.

