Εθνικό Μετσόβιο Πολυτεχνείο

N.T.U.A.

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

E.C.E.

Εργαστήριο 2: Αναγνώριση φωνής με το Kaldi Toolkit

Ελευθερία-Άννα Βαλή, Μεταπτυχιακή Φοιτήτρια ΕΔΕΜΜ

ΑΜ : 03400122 , eleannavali@ntua.gr

Κωνσταντίνος Πουλινάκης, Μεταπτυχιακός Φοιτητής ΕΔΕΜΜ

ΑΜ : 03400153 , poulinakis.kon@gmail.com

Η εργασία κατατέθηκε για το μάθημα:

Επεξεργασία Φωνής και Φυσικής Γλώσσας

May 17, 2022

Εισαγωγή

Στο παρόν εργαστήριο θέλουμε να δημιουργήσουμε ένα αυτόματο σύστημα αναγνώρισης φωνής χρησιμοποιώντας το KALDI TOOLKIT. Πιο συγκεκριμένα, στοχεύουμε στην αναγνώριση φωνημάτων (Phone recognition) από ηχογραφήσεις του USC-TIMIT dataset. Το dataset αποτελείται από .wav αρχεία ήχου. Οι ηχογραφήσεις αποτελούνται από 4 ομιλητές (2 άντρες, 2 γυναίκες) που εκφωνούν 460 προτάσεις ο καθένας. Συνοδευτικά στα αρχεία ήχου, έχουμε και το αρχείο transcript.txt όπου είναι γραμμένοι οι υπότιτλοι για κάθε πρόταση σε φυσική γλώσσα.

Βασικές έννοιες θεωρίας

Προπαρασκευή

Κατά τη διαδικασία της προπαρασκευής, εγκαταστήσαμε το εργαλείο Kaldi και στήσαμε το περιβάλλον κατάλληλα ώστε να μπορούμε στη συνέχεια να εξάγουμε τα χαρακτηριστικά MFCC και να δημιουργήσουμε γλωσσικά και φωνητικά μοντέλα.

Τα βήματα που ακολουθήσαμε είναι αυτά ακριβώς που περιγράφονται στην εκφώνηση της άσκησης. Η υλοποίηση γίνεται σε περιβάλλον Linux χρησιμοποιώντας bash shell scripts και python scripts.

4.1 Προετοιμασία Γλωσσικού Μοντέλου

Για να δημιουργήσουμε το γλωσσικό μοντέλο, είναι απαραίτητο πρώτα να δημιουργήσουμε το λεξικό της γλώσσας. Συγκεκριμένα δημιουργούμε αρχεία που περιέχουν τα 40 φωνήματα που χρησιμοποιούνται στις προτάσεις μας, συμπεριλαμβανομένων των συμβόλων αρχής και τέλους πρότασης <s>, </s>

αλλά και του Out of vocabulary συμβόλου <oov>. Δημιουργούμε επίσης ένα αρχείο που περιέχει το σύμβολο της σιωπής <sil>.

Έχοντας δημιουργήσει τα απαραίτητα αρχεία, δημιουργούμε και κάνουμε compile 2 γλωσσικά μοντέλα, ένα μοντέλο unigram και ένα bigram. Τα μοντέλα αυτά κάνουν προβλέψεις σε επίπεδο φυσικής γλώσσας. Συγκεκριμένα προβλέπουν το επόμενο φώνημα μίας πρότασης μαθαίνοντας τον πίνακα μετάβασης μέσω της συχνοτικής εμφάνισης μονογραμμάτων και διγραμμάτων.

Ερώτημα 1

Ένα γλωσσικό μοντέλο είναι ενα στατιστικό μοντέλο το οποίο αντιστοιχίζει πιθανότητες σε λέξεις και προτάσεις. Υπάρχουν δύο προσεγγίσεις για να αξιολογήσουμε ένα τέτοιο μοντέλο. Extrinsic και Intrinsic. Με μία extrinsic αξιολόγηση, βάζουμε το μοντέλο να εκτελέσει το task για το οποίο έχει σχεδιαστεί (πχ. translation). Αυτό θα μας δείξει την πραγματική ικανότητα του μοντέλου πάνω στο task. Η δεύτερη προσέγγιση είναι η Intrinsic όπου το μοντέλο αξιολογείται με βάση κάποια μετρική σε ένα μικρό test set. Το perplexity αποτελεί μία τέτοια μετρική.

Ένα γλωσσικό μοντέλο υπολογίζει την πιθανότητα εμφάνισης μίας λέξης ή πρότασης μέσα σε ένα κείμενο.

$$P(Sentence) = P(w_1, w_2, w_3, \cdots, w_N)$$

Η παραπάνω πιθανότητα υπολογίζεται διαφορετικά για ένα unigram, ένα bigram ή n-gram. Όλα όμως τα μοντέλα οφείλουν να αποδίδουν υψηλές πιθανότητες σε λέξεις ή προτάσεις που είναι συνηθισμένες ή μοιάζουν λογικές σε σχέση με τα συμφραζόμενα και χαμηλή πιθανότητα σε προτάσεις ή λέξεις που είναι λάθος ή δεν βγάζουν νόημα σε σχέση με το κείμενο. Έτσι λοιπόν ορίζουμε το μετρικό perplexity ως

$$PP(W) = \sqrt[N]{\frac{1}{P(w_1, w_2, \cdots, w_N}}$$

Το Perplexity υπολογίζεται πάνω στο test set. Ένα μοντέλο με καλή απόδοση θα πρέπει να αποδώσει υψηλή πιθανότητα στις προτάσεις του test set, καθώς πρόκειται για σωστές και λογικές προτάσεις που δεν θα έπρεπε να "μπερδεύουν" το μοντέλο μας ("The model shouldn't be perplexed by those sentences"). Συνεπώς ένα αποδοτικό μοντέλο θα πρέπει να δίνει χαμηλό perplexity πάνω στο tes

Έχοντας εκ των προτέρων διαχωρίσει το dataset σε train, test, val, εκπαιδεύσαμε τα μοντέλα μας πάνω στο train set. Στη συνέχεια αξιολογήσαμε τα δύο μοντέλα πάνω στο val και test set μέσω του perplexity. Τα αποτελέσματα που δόθηκαν από το Kaldi παρουσιάζονται παρακάτω.

Model	PP	PPWP	OOV
ug_test	54.56	19.75	2.79%
ug_dev	56.46	21.16	2.91%
bg_test	27.65	10.01	2.79%
bg_dev	28.54	10.70	2.91%

Table 1: Perplexity and out of vocabuary ratio for unigram and bigram models.

4.3 Εξαγωγή Ακουστικών Χαρακτηριστικών

Mel-Frequency Cepstral Coefficients (MFCCs)

Ένα σήμα ομιλίας, πέρα από τις πληροφορίες σχετικά με το τι έχει εκφωνηθεί, περιέχει επίσης και διάφορες άλλες παράπλευρες πληροφορίες που ποικίλουν ανάλογα με τις συνθήκες εγγραφής και σχετίζονται κυρίως με το θόρυβο στο background, την παραμόρφωση του σήματος που προκαλέι το μικρόφωνο και την πιθανή αντήχηση. Επιπλέον, παράμετροι που εξαρτώνται από τον ομιλητή, όπως η δομή της φωνητικής οδού, η προφορά, η διάθεση και το στυλ ομιλίας επηρεάζουν σημαντικά το σήμα. Συνεπώς, ένα Automatic Speech Recognition (ASR) σύστημα απαιτεί μια αξιόπιστη διαδικασία εξαγωγής χαρα-

κτηριστικών που να απομονώνει και να καταγράφει μόνο την πληροφορία σχετικά με το κείμενο που εκφωνήθηκε.
Τα MFCC έχουν αποδειχθεί ότι είναι μια αποτελεσματική διαδικασία εξαγωγής χαρακτηριστικών για το ASR. Στη συνέχεια παρουσιάζονται εν συντομία τα βασικά βήματα της διαδικασίας υπολογισμού και εξαγωγής των MFCC χαρακτηριστικών.

- Preemphasis: Με την εφαρμογή preemphasis φίλτρου πάνω στο αρχικό σήμα εξισορροπείται το φάσμα συχνοτήτων, καθώς αυξάνεται το μέγεθος της ενέργειας των υψηλότερων συχνοτήτων. Επιπλέον, η τεχνική αυτή συχνά βελτιώνει τον λόγο του σήματος προς το θόρυβο (Signal-to-Noise Ratio - SNR). Ένα preemphasis φίλτρο μπορεί να εφαρμοστεί σε ένα ακουστικό σήμα $x(t)$, χρησιμοποιώντας πρώτης τάξης φίλτρα ως εξής:

$$y(t) = x(t) - ax(t-1) \tag{1}$$

  Οι τυπικές τιμές του a είναι 0.95 και 0.97.

- Framing: Σε αυτό το στάδιο χωρίζουμε το σήμα σε short-time frames. Είναι γνωστό ότι οι συχνότητες σε ένα σήμα αλλάζουν με την πάροδο του χρόνου, επομένως στις περισσότερες περιπτώσεις δεν έχει νόημα να κάνουμε τον μετασχηματισμό Fourier σε ολόκληρο το σήμα, καθώς θα χάναμε με την πάροδο του χρόνου τα περιγράμματα συχνότητας του σήματος. Για να αποφευχθεί αυτό, μπορούμε να υποθέσουμε ότι οι συχνότητες σε ένα σήμα είναι ακίνητες για πολύ σύντομο χρονικό διάστημα. Επομένως μπορούμε πλέον να εφαρμόσουμε έναν μετασχηματισμό Fourier σε κάθε short-time frame. Τα τυπικά μεγέθη των frames είναι από 20ms μέχρι 40ms, με πιο δημοφιλές το 25ms, και μια συνήθης τιμή για το stride είναι τα 10ms.

- Windowing: Στη συνέχεια, εφαρμόζουμε μια window συνάρτηση προκειμένου να εξομαλύνουμε την υπόθεση του FFT ότι τα δεδομένα είναι άπειρα και να μειώσουμε το spectral leakage. Μια τέτοια συνάρτηση είναι το Hamming window που έχει την ακόλουθη μορφή:

$$w[n] = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right) \tag{2}$$

- Discrete Fourier Transform (DFT): Μετατρέπουμε το σήμα από το πεδίο του χρόνου στο πεδίο της συχνότητας εφαρμόζοντας τον μετασχηματισμό DFT και υπολογίζουμε το τετράγωνο του μέτρο του (power spectrum) [3]. Για τα ηχητικά σήματα, η ανάλυση στον τομέα συχνότητας είναι ευκολότερη από ό,τι στον τομέα του χρόνου.

$$P = \frac{|FFT(x_i)|^2}{N} \tag{3}$$

- Log Mel filterbank (LMFB): Σε αυτό το στάδιο, εφαρμόζουμε ένα τριγωνικό filterbank σε Mel-scale προκειμένου να λάβουμε ένα single coefficient για κάθε φίλτρο και τελικά να πάρουμε το λογάριθμο αυτού του συντελεστή. Η προσπαθεί να προσομοιάσει τη μη γραμμική αντίληψη του ήχου από το ανθρώπινο αυτί, με το να διαχωρίζει καλύτερα τις χαμηλότερες συχνότητες από ότι τις υψηλότερες.

- Discrete Cosine Transform (DCT): Τέλος, προκειμένου να αποσυσχετίσουμε τους συντελεστές που παράγαμε στο προηγούμενο βήμα εφαρμόζουμε DCT σε κάθες frame των LMFB συντελεστών. Ο λόγος για την επιβολή αυτής της αποσυσχέτισης είναι καθαρά τεχνικός, καθώς σε ορισμένους αλγόριθμους μηχανικής μάθησης είναι απαραίτητη. Επιπλέον η τεχνική αυτή, βοηθάει και στη μείωση της διαστατικότητας των τελικών χαρακτηριστικών.

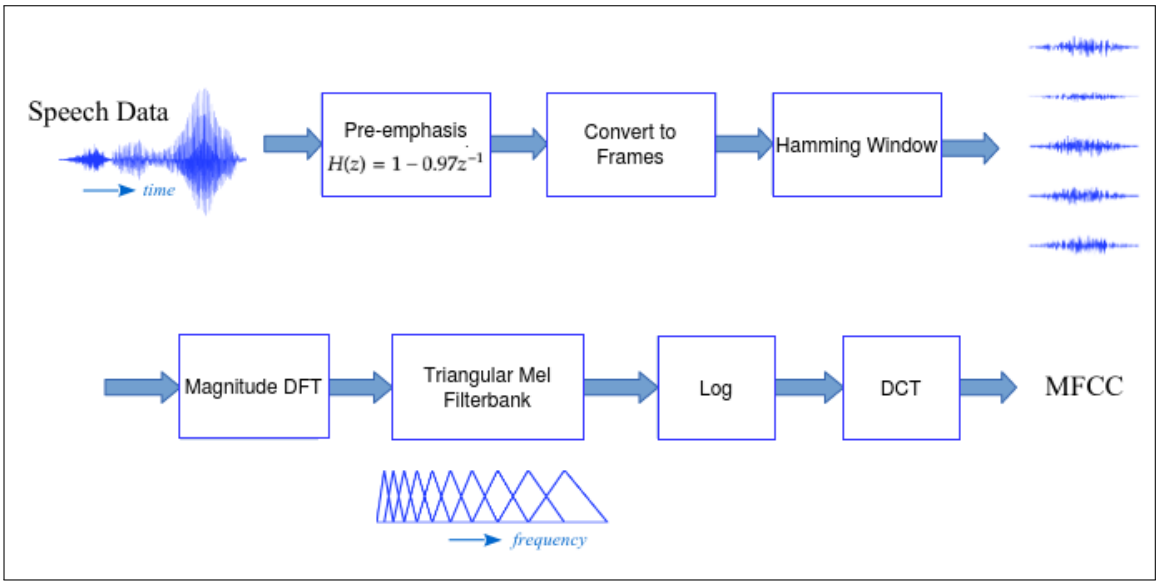

Figure 1: Οπτικοποίηση της διαδικασία εξαγωγής MFCCs χαρακτηριστικών.

Ερώτημα 2

Η Cepstral Mean and Variance Normalization (CMVN) είναι μια υπολογιστικά αποτελεσματική τεχνική κανονικοποίησης για αναγνώριση ομιλίας. Η απόδοση του CMVN είναι γνωστό ότι μειώνεται για σύντομες σε διάρκεια εκφωνήσεις. Αυτό οφείλεται στην έλλειψη επαρκών δεδομένων για την εκτίμηση των παραμέτρων και στο σφάλμα διαχωρισμού των πληροφοριών, καθώς όλες οι εκφωνήσεις αναγκάζονται να πάρουν μηδενική μέση τιμή και μοναδιαία διακύμανση.

Το CMVN ελαχιστοποιεί την παραμόρφωση που προκείπτει λόγω θορύβου, μετατρέποντας γραμμικά τα cepstral coefficients ώστε να έχουν τα ίδια segmental statistics (zero mean, unit variance). Οι παράμετροι μετασχηματισμού μπορούν να εκτιμηθούν είτε από προηγούμενες εκφωνήσεις, είτε από την τρέχουσα, είτε τμηματικά ή αναδρομικά.
Στην περίπτωση του τμηματικού (segmental) CMVN τα MFCC χαρακτηριστικά κανονικοποιούνται σύμφωνα με τον εξής μετασχηματισμό:

$$\hat{x}_t(i) = \frac{x_t(i) \ - \ \mu_t(i)}{\sigma_t(i)} \tag{4}$$

όπου $\hat{x}_t(i)$ είναι το i-οστό component του αρχικού διανύσματος χαρακτηριστικών τη χρονική στιγμή t, $\mu_t(i)$ η μέση τιμή του και $\sigma_t(i)$ η τυπική του απόκλιση.

Ερώτημα 3

Ο αριθμός των frames που παράγονται για μία ηχογράφηση εξαρτάται από την διάρκεια του σήματος ήχου, το μήκος του frame window που χρησιμοποιείται και το stride, δηλαδή πόσο μετακινείται το κέντρο του window κάθε φορά. Μαθηματικά, ο αριθμός των frames που εξάγονται από κάθε ηχογράφηση είναι

$$N = int(\frac{T-f}{s})$$

, όπου T η διάρκεια του σήματος, f το μήκος του frame και s το stride. Για την περίπτωση μας, θα θεωρήσουμε $f = 25ms$ και $s = 10ms$.

Έτσι, τελικά για τις πρώτες 5 ηχογραφήσεις του train set παίρνουμε 317, 371, 399, 328 και 464 frames αντιστοίχως.

Τα MFCC αποθηκεύτηκαν από το kaldi σε format .ark, που χρησιμοποιείται από το kaldi για αποθήκευση πινάκων και διανυσμάτων. Χρησιμοποιώντας την εντολή του kaldi copy-feats <mffc.ark> mfcc_feat.txt μπορέσαμε να οπτικοποιήσουμε τα χαρακτηριστικά MFCC για όλο το training set. Όπως αναμέναμε, κάθε πρόταση περιέχει 13 στήλες, η οποία αντιστοιχεί στα 13 πρώτα MFCC features, τα υπόλοιπα MFCC δεν τα διατηρούμε καθώς αναπαριστούν έντονες και γρήγορες αλλαγές στα filter banks τα οποία δεν είναι χρήσιμα και ευεργετικά για ένα σύστημα ASR. Τέλος, κάθε πρόταση έχει τόσες γραμμές όσες και τα frames της, που υπολογίζονται με τον τύπο παραπάνω.

4.4 Εκπαίδευση ακουστικών Μοντέλων και αποκωδικοποίηση προτάσεων.

Με την εντολή word count (wc -w) του bash μετράμε πόσα φωνήματα έχουμε στο test και στο dev αρχείο transcript. Η εντολή μας δείχνει ότι το test αρχείο περιέχει 12795 φωνήματα ενώ το dev 4930 φωνήματα, συμπεριλαμβανομένων των φωνημάτων sil στην αρχή και το τέλος της πρότασης.

$$PER_{bg\_dev} = 100\frac{insertions + substitutions + deletions}{phonemes} = 100\frac{115 + 1150 + 947}{4782} = 46.257\%$$

$$PER_{bg\_test} = 100\frac{insertions + substitutions + deletions}{phonemes} = 100\frac{181 + 2861 + 2555}{12428} = 45.035\%$$

$$PER_{ug\_dev} = 100\frac{insertions + substitutions + deletions}{phonemes} = 100\frac{78 + 1066 + 1370}{4782} = 52.572\%$$

$$PER_{ug\_test} = 100\frac{insertions + substitutions + deletions}{phonemes} = 100\frac{113 + 2577 + 3722}{12428} = 51.593\%$$

Οι παραπάνω υπολογισμοί επαληθεύονται και με την εντολή score.sh του kaldi. Η εντολή score.sh έχει 2 υπερπαραμέτρους, οι οποίες παίζουν σημαντικό ρόλο στην εξαγωγή του PER.

Η πρώτη παράμετρος είναι το **wip (word insertion penalty**). Η παράμετρος αυτή ορίζει το κόστος που έχει η εισαγωγή μίας λέξης από τον Transducer. Η παράμετρος αυτή χρησιμοποιείται ώστε να αποτρέψουμε το μοντέλο από το να προτιμάει πολύ μεγάλες ή πολύ μικρές προτάσεις. Στα αποτελέσματα παρακάτω, παρατηρούμε ότι σε όλες πέραν μία των περιπτώσεων τα καλύτερα μοντέλα έχουν παράμετρο $wip = 0.0$.

Η δεύτερη παράμετρος είναι το lmwt. Το lmwt ορίζει την βαρύτητα που έχει το γλωσσικό μοντέλο έναντι του ακουστικού κατά την διαδικασία του decoding. Με άλλα λόγια, αν υπάρχει ασυμφωνία μεταξύ των δύο, ένα υψηλό lmwt θα τείνει να δίνει περισσότερη σημαντικότητα στην απόφαση του γλωσσικού μοντέλου έναντι του ακουστικού.

Παρακάτω παρουσιάζουμε το PER πάνω στο test και το val set, για την καλύτερη τιμή των παραμέτρων wip,lmwt για κάθε μοντέλο που σχεδιάσαμε.

Model	PER	lmwt	wip
Mono_ug_test	51.59%	7	0.0
Mono_ug_dev	52.57%	7	0.0
Mono_bg_test	45.04%	7	0.0
Mono_bg_dev	46.26%	7	0.0

Table 2: Phonem error ratio για τα μονοφωνικά μοντέλα.

Model	PER	lmwt	wip
Tri_ug_test	39.17%	7	0.0
Tri_ug_dev	40.34%	8	0.0
Tri_bg_test	35.10%	7	0.0
Tri_bg_dev	36.60%	7	1.0

Table 3: Phonem error ratio για τα τριφωνικά μοντέλα.

Ερώτημα 4

Ένα Automatic Speech Recognition system έχει ως σκοπό να δημιουργήσει ένα στατιστικό μοντέλο που να μπορεί να προβλέψει μία πρότασης W από μία ακολουθία χαρακτηριστικών X.

$$P(W^*) = argmax(P(W|X)) = argmax(P(X|W)P(X)$$

Ένα ASR σύστημα απαιτεί τον συνδυασμό δύο μοντέλων, ενός γλωσσικού που θα μπορεί να υπολογίσει την πιθανότητα $P(W)$ και ενός φωνητικού/ακουστικού (generative model) που θα υπολογίζει τη $P(X|W)$. Μία συχνή υλοποίηση που χρησιμοποιείται είναι το GMM-HMM.

Η πιθανότητα P(W) μοντελοποιείται εύκολα χρησιμοποιώντας ένα γλωσσικό μοντέλο όπως ένα unigram ή bigram.

Στην αναγνώριση φωνής είναι εύκολο να βρεθεί η πολυδιάστατη κατανομή $P(X|W)$ χρησιμοποιώντας ένα μοντέλο μίξης γκαουσιανών, Gaussian Mixture Model. Χρησιμοποιώντας δεδομένα εκπαίδευσης, δηλαδή αρχεία ήχου με υπότιτλους φωνημάτων (transcripts), μπορούμε να εκπαιδεύσουμε το GMM ώστε να βρει από ποια πολυδιάστατη γκαουσιανή κατανομή μπορεί να περιγραφεί καλύτερα ένα φώνημα. Οι κατανομές εκφράζουν τις τιμές των ακουστικών χαρακτηριστικών, είτε αυτά είναι Filter Banks είτε MFCCs. Παρόλα αυτά ένα GMM εξετάζει κάθε frame ήχου αυτόνομα, χωρίς να έχει την ικανότητα να συσχετίσει το παρόν frame με αυτά που προηγήθηκαν (δηλαδή τα συμφραζόμενα

φωνήματα). Μπορούμε να μοντελοποιήσουμε την πιθανότητα μετάβασης από ένα φώνημα σε κάποιο άλλο σε σχέση με την παρατήρηση που έχουμε με ένα κρυφό μαρκοβιανό μοντέλο HMM. Σε αυτό το HMM οι κρυφές καταστάσεις Q είναι το σύνολο των φωνημάτων $Q = \{a, ah, .., sh, zh, ...\}$, ενώ οι φανερές καταστάσεις (observations) είναι τα ακουστικά χαρακτηριστικά. Η έξοδος του GMM, όταν δωθεί μία φανερή κατάσταση ως είσοδος αποτελεί την emission probability $b_i(O_j)$ του μαρκοβιανού μοντέλου, που πρακτικά μας δείχνει πόσο πιθανό είναι είναι να εμφανίζονται οι συγκεκριμένες τιμές χαρακτηριστικών αν βρισκόμαστε στην συγκεκριμένη κρυφή κατάσταση, δηλαδή φώνημα.

Μετά την εκπαίδευση του το GMM εκτελεί clustering των MFCC features, όπου κάθε cluster αντιστοιχεί σε ένα φώνημα. Το HMM δίνει context knowledge, έτσι συνδυάζοντας τα 2 μοντέλα μπορούμε να βελτιώσουμε την απόδοση σε σχέση με ένα απλό GMM.

Ερώτημα 5

Σε ένα σύστημα ASR, σκοπός είναι να αναγνωριστεί η φυσική γλώσσα που "κρύβεται" πίσω από ένα σήμα ήχου. Ο κανόνας του Bayes για την a-posteriori πιθανότητα, στην περίπτωση του speech recognition γράφεται ως:

$$P(utterance|audio) = \frac{p(audio|utterance)P(utterance)}{p(audio)}$$

Η πιθανότητα $P(utterance)$ δίνεται από ένα γλωσσικό μοντέλο, όπως για παράδειγμα ένα n-gram model. Η κατανομή $p(audio|utterance)$ είναι μία στατιστική κατανομή που παράγεται μέσω της εκπαίδευσης ενός φωνητικού μοντέλου. Ο όρος $p(audio)$ είναι όρος κανονικοποίησης και μπορεί να αγνοηθεί.

Στην δική μας περίπτωση, θέλουμε να αναγνωρίσουμε το φώνημα που εκφωνείται σε μία στιγμή μέσω των χαρακτηριστικών MFCCs του σήματος ήχου. Έτσι ο τύπος του Bayes μπορει να γραφεί πιο περιγραφικά ως

$$P(phonem|MFCC) = \frac{p(MFCC|phonem)P(phonem)}{p(MFCC)}$$

Ένα ASR αποτελείται από ένα γλωσσικό και ένα ακουστικό μοντέλο. Μέσω του γλωσσικού μας μοντέλου, για την περίπτωση μας ένα bigram fst, υπολογίζουμε την πιθανότητα

$$P(phonem_t) = P(phonem_t|phonem_{t-1})P(phonem_{t-1})$$

Όπου οι παραπάνω πιθανότητες υπολογίζονται εύκολα μέσω υπολογισμού της συχνότητα εμφάνισης των φωνημάτων, δηλαδή

$$P(phonem_t|phonem_{t-1}) = \frac{\#(phonem_t|phonem_{t-1})}{\#(phonem_t)}$$

$$P(phonem_t) = \frac{\#(phonem_t)}{\#(phonems)}$$

Μέσω του φωνητικού μας μοντέλου, δηλαδή ενός GMM, προσπαθούμε να αντιστοιχίσουμε πολυδιάστατες κατανομές, δηλαδή τις τιμές των MFCC, σε φωνήματα.

Ερώτημα 6

Ο γράφος HCLG αποτελείται από 4 αντικείμενα

1. G : ένας acceptor που κωδικοποιεί την γραμματική ή αλλιώς το γλωσσικό μοντέλο. Τα σύμβολα εισόδου και εξλοδου του είναι τα ίδια.
2. L : το λεξικό, η έξοδος του είναι οι λέξεις ενώ οι εισόδοι του είναι τα φωνήματα.
3. C : περιγράφει την εξάρτηση από τα συμφραζόμενα (contect dependency). Οι έξοδοί του είναι φωνήματα ενώ οι εισόδοι του είναι τα context-dependent φωνήματα, δηλαδή ακολουθία από N φωνήματα.

4. H : περιγράφει το HMM μοντέλο. Οι έξοδοι του είναι τα context dependent φωνήματα που αναφέραμε παραπάνω ενώ οι εισόδοι του είναι κωδικοί μετάβασης που κωδικοποιούν πιθανοτικές κατανομές

Η fst που παράγεται υπόκειται σε minimization και determinization. Επίσης, θέλουμε η fst να δίνει στοχαστικές εξόδους. Μαθηματικά ο γράφος μπορεί να περιγραφεί με την έκφραση

$$HCLG = asl(min(rds(det(H'omin(det(Comin(det(LoG))))))))$$

4.5 Bonus: Μοντέλο DNN-HMM με PyTorch

1 & 2

Χρησιμοποιώντας την εντολή align_si.sh εκτελούμε allignment των train,dev,test συνόλων. Στη συνέχεια χρησιμοποιώντας τις εντολές που δίνονται στο run_dnn.sh υπολογίζουμε τα cmvn στατιστικά.

3

Χρησιμοποιώντας την κλάση TorchSpeechDataset που μας δόθηκε, φορτώνουμε τα ακουστικά χαρακτηριστικά που εξήχθησαν μέσω του kaldi στην Python. Παρατηρούμε ότι το αντικείμενο της κλάσης TorchSpeechDataset που δημιουργήσαμε περιέχει 195 χαρακτηριστικά για κάθε ένα από τα συνολικά 598450 frames ήχου, που αντιστοιχούν στις 1315 προτάσεις που εκφωνούνται στο training set. Αντίστοιχα, έχουμε 598450 ετικέτες, μία για κάθε frame. Να τονιστεί ότι παρότι οι ετικέτες αντιστοιχούν σε φωνήματα, ο συνολικός αριθμός ξεχωριστών ετικετών είναι 920, που σημαίνει ότι σε κάθε (πραγματικό) φώνημα αντιστοιχούν 23 καταστάσεις. Οι παραπάνω καταστάσεις υπάρχουν ώστε να μπορούμε να εκφράσουμε ένα φώνημα με όλες του τις διαφορετικές εκφάνσεις.

4. Εκπαίδευση του DNN

Αρχικά σχεδιάζουμε ένα βαθύ νευρωνικό feedforward network. Το νευρωνικό μας αποτελείται από 4 κρυφά επίπεδα και έναν ταξινομητή στην κορυφή.

Κάθε κρυφό επίπεδο αποτελείται από ένα dense fully connected layer, ακολουθούμενο από ένα Batch Normalization layer, μία ReLU activation function και τέλος ένα Dropout layer για περιορισμό του overfitting. Τα 4 κρυφά επίπεδα αποτελούνται από 256, 256, 128, 64 νευρώνες αντίστοιχα.

Στην κορυφή του νευρωνικού, ως ταξινομητή, χρησιμοποιούμε ένα fully connected layer με 920 νευρώνες όσες και οι διαφορετικές κλάσεις (κρυφές καταστάσεις του HMM). Ως activation function χρησιμοποιείται η Softmax ώστε να μας επιστρέψει ως έξοδο πιθανότητες.

Για την εκπαίδευση του νευρωνικού δικτύου χρησιμοποιούμε τον optimizer Adam και ως συνάρτηση κόστους την CrossEntropyLoss.

$$L_{CE}(x,y) = -\sum_{c=1}^{C} log(x)y$$

```python
class TorchDNN(nn.Module):
    """Create a DNN to extract posteriors that can be used for HMM decoding
    Parameters:
        input_dim (int): Input features dimension
        output_dim (int): Number of classes-phonems
        num_layers (int): Number of hidden layers
        batch_norm (bool): Whether to use BatchNorm1d after each hidden layer
        hidden_dim (int): Number of neurons in each hidden layer
        dropout_p (float): Dropout probability for regularization
    """
    def __init__(
        self, input_dim, output_dim, num_layers=4, batch_norm=True, hidden_dim=[256,256,128,64], dropout_p=0.2
    ):
        super(TorchDNN, self).__init__()
        self.input_dim = input_dim
        self.output_dim = output_dim

        # self.fc_block_1=self.__block__(input_dim,hidden_dim[-1],batch_norm,dropout_p)

        self.fc_block_1=self.__block__(input_dim,hidden_dim[0],batch_norm,dropout_p)
        self.fc_block_2=self.__block__(hidden_dim[0],hidden_dim[1],batch_norm,dropout_p)
        self.fc_block_3=self.__block__(hidden_dim[1],hidden_dim[2],batch_norm,dropout_p)
        self.fc_block_4=self.__block__(hidden_dim[2],hidden_dim[3],batch_norm,dropout_p)
        self.classifier = nn.Sequential(
                nn.Linear(hidden_dim[-1], output_dim),
                nn.Softmax()
            )
```

Figure 2: Αρχικοποίηση νευρωνικού δικτύου.

```
def forward(self, x):
    '''
    Forward-pass
    '''
    out= self.fc_block_1(x)
    out= self.fc_block_2(out)
    out= self.fc_block_3(out)
    out= self.fc_block_4(out)

    return self.classifier(out)

def __block__(self,in_features,out_features,batch_norm,drop=0.3):
    if batch_norm:
        return nn.Sequential(
            nn.Linear(in_features, out_features),
            nn.BatchNorm1d(out_features),
            nn.ReLU(),
            nn.Dropout(p=drop)
        )
    else:
        return nn.Sequential(
            nn.Linear(in_features, out_features),
            nn.ReLU(),
            nn.Dropout(p=drop)
        )
```

Figure 3: Feed forward συνάρτηση DNN.

Η υλοποίηση της συνάρτησης train του DNN που κατασκευάσαμε παραπάνω παρουσιάζεται στο Figure 5.Η συνάρτηση αυτή εκπαιδεύει το μοντέλο μας σε batches ενώ συγχρόνως υπολογίζει το accuracy και το μέσο σφάλμα ανά εποχή. Στο τέλος της εκπαίδευσης για καλύτερη εποποτεία των μετρικών που χρησιμοποιήθηκαν κατά τη διάρκεια του training δημιουργούνται δύο διαγράμματα: (α) το διάγραμμα του σφάλματος εκπαίδευσης σε αντιπαραβολή με το σφάλμα του validation set και (β) το διάγραμμα του accuracy εκπαίδευσης σε αντιπαραβολή με το αντίστοιχο του validation set.

```python
def train(model, criterion, optimizer, train_loader, dev_loader, epochs=50, patience=3):
    """Train model using Early Stopping and save the checkpoint for
    the best validation loss
    """

    plot_loss_tr=[]
    plot_loss_dev=[]
    plot_ac_tr=[]
    plot_ac_dev=[]
    for epoch in range(epochs):
        print("EPOCH:",epoch)
        run_loss=0
        dev_loss=0
        cor_dev=0
        cor_tr=0
        model.train()
        for j,data in enumerate(train_loader):
            X_batch,y_batch = data
            optimizer.zero_grad()
            out = model(X_batch)
            loss = criterion(out,y_batch)
            loss.backward()
            optimizer.step()

            run_loss += loss.detach().item()
            cor_tr+=(torch.argmax(out,dim=1)==y_batch).float().sum()

        plot_loss_tr.append(run_loss/(i+1))
        plot_ac_tr.append(100*cor_tr/(i*128))
```

```python
        model.eval()
        for i,data in enumerate(dev_loader):
            X_batch,y_batch = data
            out = model(X_batch)
            loss = criterion(out,y_batch)

            dev_loss += loss.detach().item()
            cor_dev+=(torch.argmax(out,dim=1).item()==y_batch).float().sum()

        plot_loss_dev.append(dev_loss/(i+1))
        plot_ac_dev.append(100*cor_dev/(i*128))


        print("For epoch ",epoch," train accuracy=",100*cor_tr.item()/(j*128)," and dev accuracy=",
        100*cor_dev.item()/(i*128))

        if epoch % 5 == 0:
            print("Epoch: %d, train loss: %1.5f, dev loss: %1.5f", epoch, plot_loss_tr[epoch],
            plot_loss_dev[epoch])

    fig , ax = plt.subplots(figsize = (10,10))

    plt.plot(plot_loss_dev, label='Val_loss') #actual plot
    plt.plot(plot_loss_tr, label='Train_loss') #predicted plot
    plt.title('Training and validation loss curves')
    plt.legend()
    plt.show()
```

Figure 4: Training συνάρτηση για το DNN.

Ενδεικτικά παρουσιάζονται στη συνέχεια τα διαγράμματα accuracy και loss για τις 3 πρώτες εποχές. Παρατηρούμε ότι το σφάλμα πέφτει όσο αυξάνονται οι εποχές, ενώ το accuracy λειτουργεί αντίστροφα. Η παρατήρηση αυτή αποτελεί εύλογη ένδειξη ότι το μοντέλο μας λειτουργεί σωστά.

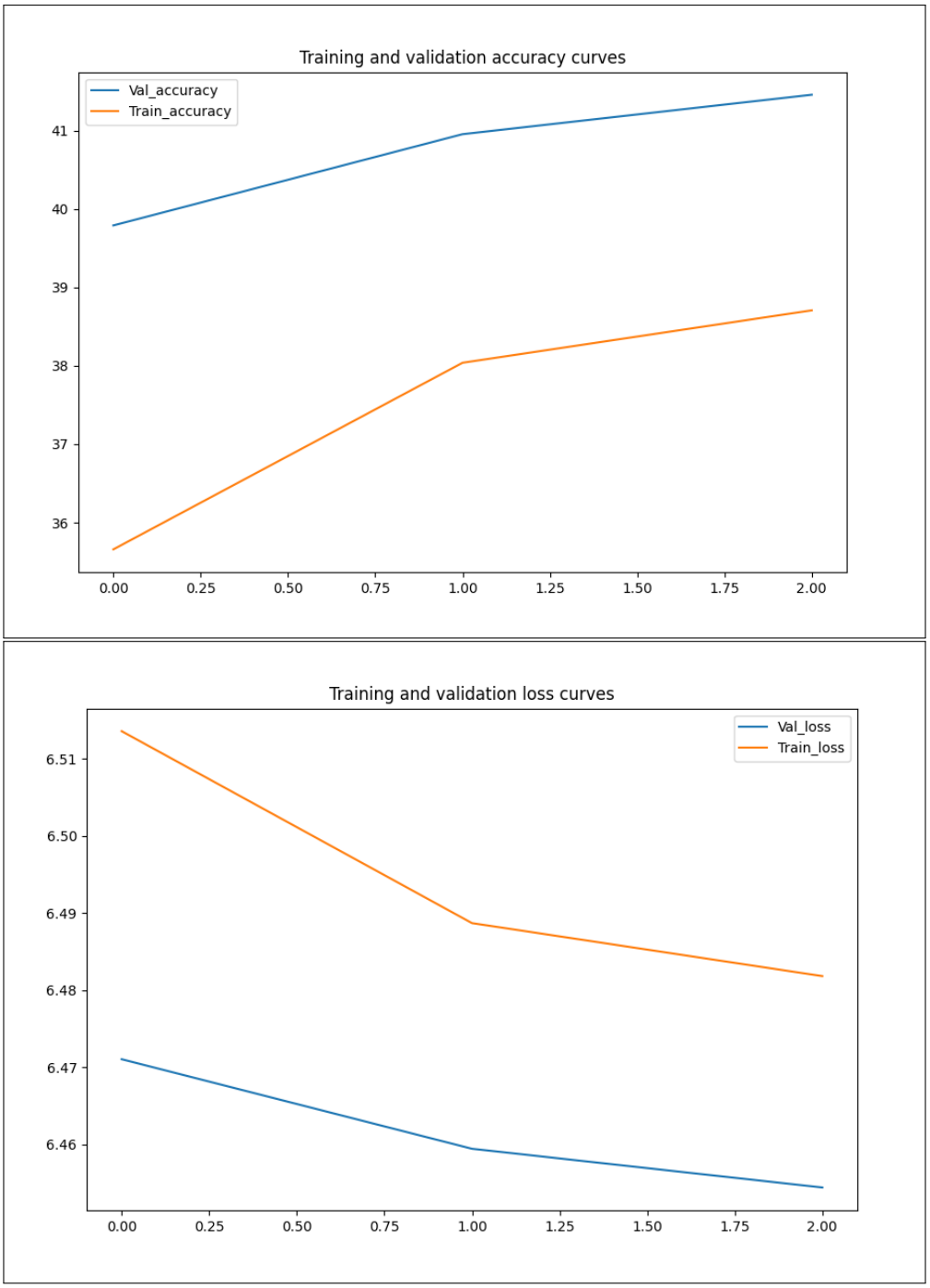

Figure 5: Διαγράμματα μετρικών για τις 3 πρώτες εποχές πάνω στο training και validation set.

Ερώτημα 7

Ενα DNN-HMM διαφέρει σε σχέση με ένα GMM-HMM ως προς το μοντέλο που εξάγει την πιθανότητα $P(X|W)$. Το DNN είναι αυτό που προβλέπει την

πιθανότητα να είμαστε σε μία κρυφή κατάσταση δεδομένων των χαρακτηριστικών MFCC που δίνονται ως είσοδοι ενώ το HMM πάλι είναι αυτό που μοντελοποιεί τις μεταβάσεις από το ένα φώνημα στο άλλο, δίνοντας Context Knowledge στο Feedforward Network. Κατά μία έννοια, ο συνδυασμός του feedforward DNN με ένα HMM οδηγεί σε λειτουργικότητα που προσομοιάζει ένα RNN, δίνει δηλαδή κατά κάποιον τρόπο μνήμη στο DNN.

Η προσθήκη του DNN έναντι του GMM μπορεί να φανεί ωφέλιμη διότι το DNN δεν χρησιμοποιεί απλά τις τιμές των MFCC αλλά παράγει κάποια representations από αυτά τα χαρακτηριστικά τα οποία μπορούν να οδηγήσουν τον τελικό ταξινομητή σε καλύτερες προβλέψεις. Ο λόγος που συμβαίνει αυτό είναι διότι ένα μη γραμμικά διαχωρίσιμο πρόβλημα μπορεί να γίνει γραμμικά διαχωρίσιμο στον χώρο που έχει μεταφέρει το DNN τα αρχικά χαρακτηριστικά.

Όμως, όπως φαίνεται και από το training loop παραπάνω, το νευρωνικό δίκτυο εκπαιδεύτηλε με supervised learning, δηλαδή απαιτούνται ετικέτες για κάθε data point. Για κάθε frame έχουμε 195 χαρακτηριστικά και την αντίστοιχη ''σωστή'' ετικέτα φωνήματος. Αρχικά όμως, οι μόνες ετικέτες που είχαμε ήταν τα transcipt των αρχείων ήχου. Για να παράξουμε τις ετικέτες για κάθε frame χρησιμοποιήσαμε το αρχικό GMM-HMM και την διαδικασία του alignment. Συνεπώς, δεν θα μπορούσαμε να εκπαιδεύσουμε απευθείας ένα νευρωνικό με αυτόν τον τρόπο καθώς μας έλειπαν οι ετικέτες για κάθε frame.

Ερώτημα 8

Κάθε επίπεδο ενός νευρωνικού δικτύου έχει εισόδους που χαρακτηρίζεται από μία κατανομή. Η κατανομή αυτή αλλάζει κατά την διαδικασία της εκπαίδευσης, επηρεάζεται από την αρχικοποίηση που δίνεται στα βάρη και τα biases αλλά και την τυχαιότητα των δεδομένων εκπαίδευσης. Η επίδραση αυτής της τυχαιότητας στην κατανομή εισόδου ενός επιπέδου ονομάζεται internal covariate shift.

Έχει φανεί ότι η χρήση της Batch Norm βοηθάει στο πρόβλημα του gradient/exploding gradients.