**ΚΕΦΑΛΑΊΟ 4 – ΑΝΑΠΑΡΑΣΤΑΣΗ ΔΕΔΟΜΈΝΩΝ**

**4.1 ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ**

Για την επεξεργασία ακουλουθιακών δεδομένων (όπως κείμενο ή μουσική) υπάρχουν τουλάχιστον τρείς τύποι και στάδια όπου η αναπαράσταση των δεδομένων πρέπει εξεταστεί:

* *Είσοδος κατά την διαδικασία Εκπαίδευσης (Training Input):* Η αναπαράσταση των δεδομένων που χρησιμοποιείται κατά την διαδικασία εκπαίδευσης του δικτύου.
* *Είσοδος για την πρόβλεψη (Test Input):* Η είσοδος στο δίκτυο για την παραγωγή δεομένων. Για παράδειγμα σε ένα σύστημα που παράγει την συνέχεια μιας μελωδίας ως αρχική είσοδος για την παραγωγή μπορεί να είναι η αρχική μελωδία ή ένα κομμάτι αυτής.
* *Παραγόμενη ακολουθία:* Ο στόχος παραγωγής*.*

Οι παραπάνω επιλογές καθορίζουν την μορφή και την αρχιτεκτονική του δικτύου. Για παράδειγμα για την παραγωγή μελωδιών στο [14] τόσο η είσοδος εκπαίδευσης όσο και η είσοδος για την πρόβλεψη καθώς και η παραγόμενη ακολουθία είναι μελωδίες η οποίες περιορίζονται σε μέγεθος από έναν αριθμό νοτών (μέγεθος ακολουθίας). Αντίθετα σε άλλα συστήματα με διαφορετικό στόχο, όπως στο [15], που παράγονται συνοδευτικές συγχορδίες σε μελωδίες, η είσοδος εκπαίδευσης καθώς και η είσοδος για την σύνθεση είναι μελωδίες ενώ η παραγόμενη έξοδος είναι ένα σύνολο συγχορδιών.

Σε όλα τα μοντέλα που κατασκευάστηκαν στα πλαίσια της εργασίας αυτής χρησιμοποιήθηκε η τεχνική που αναφέρεται στο [14], δηλαδή οι αναπαραστάσεις σε όλα τα στάδια είναι μονοφωνικές μελωδίες δηλαδή απλές ακολουθίες νοτών.

**4.2 ΑΝΑΠΑΡΆΣΤΑΣΗ ΔΕΔΟΜΕΝΩΝ**

Ανάλογα την αρχιτεκτονική του δικτύου που χρησιμοποιείται για την σύνθεση, η κωδικοποίηση των παραπάνω δεδομένων μπορεί να διαφέρει. Αυτό οφείλεται στο γεγονός ότι μια μελωδία έχει διάφορους τρόπους αναπαράστασης όπως σε παρτιτούρα (με μορφή εικόνας), σε ένα συνεχές σήμα (audio signal), σε συμβολική μορφή κ.α. Γενικότερα οι περισσότερες βαθιές αρχιτεκτονικές που ασχολούνται με την μουσική (είτε αναφορά την σύνθεση είτε αλλά προβλήματα) χρησιμοποιούν συμβολικές αναπαραστάσεις. Η πλέον ευρέως χρησιμοποιούμενη συμβολική κωδικοποίηση είναι η μορφή Midi η οποία περιγράφεται παρακάτω.

**4.2.1 – Κωδικοποίηση MIDI**

Για την εκπαίδευση των δικτύων θα χρησιμοποιήσουμε dataset από διάφορα μουσικά όργανα αλλά και συνδυασμούς τους και θα αξιολογήσουμε τα διάφορα αποτελέσματα. Τα δεδομένα αυτά είναι σε μορφή Midi (Musical Instrument Digital Interface) και ουσιαστικά κάθε κομμάτι αποτελείται από έναν συγκεκριμένο αριθμό διακριτών μηνυμάτων, οπού το κάθε ένα περιέχει πληροφορίες σχετικά με τις νότες που παίζονται κάθε στιγμή.

**4.1 Συλλογή Δεδομένων**

**4.1.1 Κωδικοποίηση Midi**

Η κωδικοποίηση Midi είναι ένα πρωτόκολλό που επιτρέπει την εύκολη επικοινωνία μεταξύ υπολογιστών και ηλεκτρονικών μουσικών οργάνων όπως πιάνο, keyboards, synthesizer και άλλων.

Ένα Midi αρχείο αποτελείται από διακριτά μηνύματα (Midi Events) τα όποια μπορεί να έχουν διάφορους προορισμούς. Κάθε ένα Midi αρχείο μπορεί να επικοινωνεί συγχρόνως με 16 ανεξάρτητα κανάλια (channels ή streams). Αυτή ιδιότητα του πρωτοκόλλου, δηλαδή η ταυτόχρονη διαχείριση 16 ανεξάρτητων συνομιλίων, επιτρέπει το ταίριασμα των σημαντικών πληροφοριών μεταξύ προορισμών. Σε κάθε ένα κανάλι στέλνεται κάθε φορά ένα Midi Event το οποίο μπορεί να έχει διαφορετικό τύπο και προορισμό ανάλογα με την πληροφορία που φέρει.

Ένα Midi Εvent αποτελείται από ένα 8-bit status byte το οποίο δηλώνει τον τύπο του μηνύματος και ακολουθείται από άλλο ένα ή δυο byte δεδομένων (data bytes) τα οποία περιέχουν τις πληροφορίες σχετικά με το event αυτό. Υπάρχουν διάφοροί τύποι τέτοιων μηνυμάτων αλλά σε ένα γενικότερο πλαίσιο αφαίρεσης χωρίζονται σε Μηνύματα Καναλιού (Channel Messages) και Μηνύματα Συστήματος (System Messages). Τα πρώτα είναι μηνύματα τα οποία απευθύνονται σε συγκεκριμένα κανάλια και ο αριθμός του καναλιού αυτού περιέχεται στο status byte. Αντίθετα τα Μηνύματα Συστήματος δεν αναφέρονται σε κάποιο κανάλι και συνεπώς δεν υπάρχει ο αριθμός καναλιού μέσα σε στο status byte.

Με την σειρά τους τα Channel Messages χωρίζονται σε Channel Voice Messages και Mode Messages. Τα Channel Voice Messages περιέχουν τις μουσικές πληροφορίες των κομματιών και ουσιαστικά αποτελούν το μεγαλύτερο μέρος των μηνυμάτων σε ένα Midi αρχείο. Σε αυτή την κατηγορία μηνυμάτων ανήκουν τα παρακάτω events:

* *Note On*
* *Note off*
* *Program Change*
* *Aftertouch*
* *Pitch Bend*
* *Control Change*
* *Bank Select*
* *RPN / NRPN*

Αντίθετα τα Channel Mode Messages επηρεάζουν τον τρόπο με τον οποίον ένα όργανο λήψης θα ανταποκρίνεται στα Voice Messages. Αυτά με την σειρά τους χωρίζονται σε System Exclusive Messages για την μεταφορά μηνυμάτων σε κάθε ένα κατασκευαστή οργάνων μοναδικά και τα Real Time Messages τα οποία είναι υπεύθυνα για τον έλεγχο των Midi συσκευών.

Από τα παραπάνω είδη μηνυμάτων τα πραγματικά χρήσιμα για την αποκωδικοποίηση της μουσικής πληροφορίας αλλά και την ανακατασκευή αυτής είναι τα δύο πρώτα που ορίζουν το ποια νότα παίζεται κάθε φορά και με πόση δύναμη (ένταση) αυτή πατήθηκε. Η μορφή των μηνυμάτων αυτών παρουσιάζεται παρακάτω:

* *Note on*: Η ενεργοποίηση μιας νότας είναι ένα μήνυμα τύπου Note on όπου στο status byte του περιέχεται ο αριθμός του καναλιού του οργάνου, με πεδίο τιμών [0,15] και ακολουθείται από δυο data bytes που εκφράζουν την νότα που πατήθηκε (pitch), με πεδίο τιμών [0, 127] και την επιτάχυνση (velocity) αυτής, με πεδίο τιμών [0, 127]. Ως επιτάχυνση ορίζεται η δύναμη με την οποία πατιέται ένα πλήκτρο (μια νότα). Για παράδειγμα το μήνυμα <Note on, 0, 60, 50> σημαίνει την ενεργοποίηση μια νότας στο κανάλι 1, με pitch 60 (middle C) με επιτάχυνση 50.
* *Note off:* Όταν αφήνεται μια νότα στέλνεται ένα Note off μήνυμα το οποίο έχει επίσης την ίδια μορφή με ένα Note on μήνυμα. Σε αυτήν όμως την περίπτωση τα data bytes περιέχουν πληροφορία σχετικά με το ποια νότα αφήνεται (pitch) και με πόση δύναμη (velocity) αφήνεται αυτή. Στα περισσότερα πραγματικά συστήματα το velocity των note off νοτών αγνοείται. Για παράδειγμα το μήνυμα <Note off, 0, 60, 20> σημαίνει ότι στο κανάλι 1, σταματάει να παίζει η νότα 60 και αυτή αφήνεται με επιτάχυνση 20.

Κάθε συμβάν Midi ενσωματώνεται σε ένα γεγονός που περιέχει μια τιμή χρόνου delta η οποία περιλαμβάνει πληροφορίες χρονισμού για το event αυτό. Η τιμή delta αντιπροσωπεύει την χρονική στιγμή του event και μπορεί να αναπαριστά είτε τον αριθμό των χτύπων είτε τον πραγματικό χρόνο (σε δευτερόλεπτα) από την αρχή του κομματιού. Αξίζει να σημειωθεί ότι οι 2 παραπάνω αναπαραστάσεις είναι ισοδύναμες και μπορούμε γνωρίζοντας μερικά χαρακτηριστικά του κομματιού να τις εναλλάσσουμε. Συνεπώς ο χρόνος για τον οποίον μια νότα είναι ενεργή υπολογίζεται άμεσα ως η διαφορά της χρονικής στιγμής που τελειώνει μια νότα (Note off event) μείον την στιγμή που η αντίστοιχη νότα ενεργοποιείται (Note On event). Συνεπώς η τιμή του χρόνου είναι μια συνεχής μη αρνητική μεταβλητή και συνεπώς δεν μπορεί να μοντελοποιηθεί όπως το Pitch και το Velocity (δεν αποτελεί ένα πρόβλημα ταξινόμησης).

Παρόλα αυτά με βάση μουσικούς κανόνες ο χρόνος μπορεί να μεταλλαχθεί και να μετασχηματιστεί από μια συνεχής μεταβλητή σε μια διακριτή. Έτσι για παράδειγμα στο dataset με το πιάνο υπάρχουν 87 διαφορετικές τιμές Pitch, 103 διαφορετικές τιμές Velocity ενώ μόλις 54 διαφορετικές τιμές χρόνου ……………………………………………………………………………………………………………………………………………………………………………………………….

Αξίζει να σημειωθεί ότι ο παραπάνω μετασχηματισμός έφερε και απώλεια κάποιας μουσικής πληροφορίας. Για παράδειγμα σύνθετες νότες (δηλαδή δύο ή περισσότερες νότες με το ίδιο pitch ενωμένες όπου η διάρκειά της είναι το άθροισμα βασικών χρονικών μονάδων) αναπαριστόνται ως η ακολουθία νοτών με το ίδιο Pitch με μικρότερες διάρκειες όπου το άθροισμα των διαρκειών τους ισούται με την διάρκεια της αρχικής νότας. Αυτή η επιλογή δεν είναι η μοναδική που θα μπορούσε να γίνει. Για παράδειγμα θα μπορούσε να αυξήσουμε τον διαφορετικό αριθμό των χρονικών διαρκειών και οι νότες αυτού του είδους να μείνουν ως έχουν. Η επιλογή αυτή στο σύνολο δεδομένων του πιάνο αυξάνει τον συνολικό αριθμό των διαφορετικών διαρκειών από 54 σε 498. Η αύξηση δεν είναι δραματικά μεγάλή αλλά τελικά στα μοντέλα επιλέχθηκε η πρώτη τεχνική όπου ο χρόνος έχει μικρότερο πεδίο τιμών (περισσότερα για την επιλογή αυτή εξηγούνται παρακάτω).

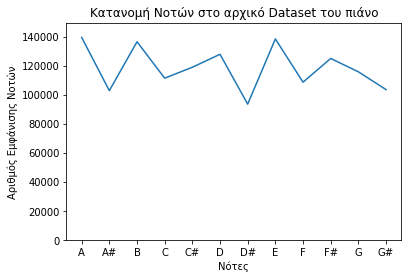
**4.2 ΑΝΆΛΥΣΗ ΔΕΔΟΜΈΝΩΝ**

Στα πλαίσια της διπλωματικής αυτής κατασκευάστηκαν την διαφορά μοντέλα και εκπαιδευτήκαν με dataset από διαφορετικά μουσικά όργανα. Η χρήση διαφορετικών μουσικών οργάνων για την εκπαίδευση πιστεύεται ότι θα αλλάξει κατά πολύ τα αποτελέσματα και την ποιότητα των παραγόμενων αποτελεσμάτων. Σε έναν παραλλήλισμό με την παραγωγή κειμένου κάτι αντίστοιχό θα ήταν αν άλλαξε το ύφος των κειμένων που χρησιμοποιείται για την εκπαίδευση (κωμικό, δραματικό, περιπέτεια). Συγκεκριμένα χρησιμοποιήθηκαν δεδομένα από πιάνο και από κιθάρα καθώς και παραλλαγές τους αλλά και συνδυασμοί αυτών.

**4.2.1 ΔΕΔΟΜΈΝΑ ΑΠΌ ΠΙΆΝΟ**

Σε αυτό το dataset υπάρχουν 623 κομμάτια από κλασσικού πιάνου τα οποία δημιουργούν μια ακολουθία 1.425.671 νοτών. Κάθε κομμάτι μεταχειρίζεται ως ακολουθία νοτών και λόγω του ότι τα κομμάτια είναι πολύ λίγα σε σχέση με των ρυθμό εμφάνισης των νοτών δεν χρησιμοποιήθηκαν κάποιες ειδικές τιμές για την εναλλαγή των κομματιών, μιας και λόγω του bias αυτές θα δυσκολεύαν την εκπαίδευση. Για να γίνει καλύτερα κατανοητό στο σύνολο αυτό η μέση τιμή εμφάνισης κάθε pitch είναι 13.172,87 σημαντικά μεγαλύτερη του 623 (που είναι οι εναλλαγές). Επίσης σκοπός των μοντέλων δεν ήταν η παραγωγή μελωδιών συγκεκριμένου μήκους (όπως σε ένα σύστημα μετάφρασης όπου για κάθε είσοδο πρέπει να παραχθεί συγκεκριμένος αριθμός λέξεων) αλλά σκοπός ήταν να δοκιμαστούν τα συστήματα στην παραγωγή μελωδιών αυθαίρετου μήκους και ιδιαίτερα να εξεταστεί η συμπεριφορά τους σε μεγάλες ακολουθίες εισόδου και εξόδου. Αυτός ήταν και ο λόγος που στο κόψιμό των κομμάτων σε ακολουθίες δεν χρησιμοποιήθηκαν κατάλληλες τιμές για την έναρξη και για την λήξη των ακολουθιών. Συνεπώς σε κάθε σύνολο υπάρχουν απλά νότες στην σειρά, χωρίς να παραβάλλεται κάτι αναμεσά τους και πρακτικά μπορούν όλες να ξεκινήσουν να παίζουν σαν να ήταν ένα ενιαίο κομμάτι. Δοκιμάστηκαν και άλλες τεχνικές (ειδικές τιμές για αρχή κομματιού, για αρχή και λήξη ακολουθιών κ.α.) αλλά αυτή επέδειξε τα καλύτερα αποτελέσματα.

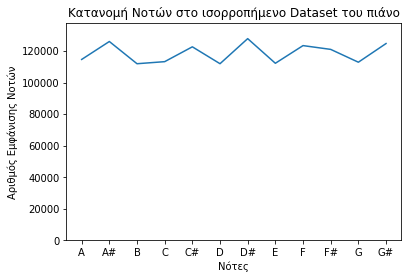
Σε αυτό τα dataset υπάρχουν 87 διαφορετικές τιμές Pitch, 103 διαφορετικές τιμές Velocity ενώ μόλις 54 διαφορετικές τιμές χρόνου. Στο παρακάτω διάγραμμα παρουσιάζεται ο αριθμός εμφάνισης κάθε νότας στο σύνολο δεδομένων του πιάνο.



Από το παραπάνω διάγραμμα φαίνεται ότι δεν υπάρχει κάποια ομοιομορφία στην κατανομή των νοτών στα δεδομένα εκπαίδευσης. Η πιο συνηθισμένη νότα στο παραπάνω σύνολο είναι η Α όπου εμφανίζεται 141.016 ενώ αυτή που εμφανίζεται τις λιγότερες φορές είναι η D# και εμφανίζεται 92.642 (διαφορά 48.374 φορές). Αυτό αποτελεί σημαντικό πρόβλημα για την εκπαίδευση των μοντέλων, όπως αναφέραμε και παραπάνω λόγω του μεγάλου bias που υπάρχει στις πιο συχνές νότες. Πρακτικά από τα αποτελέσματα φάνηκε ότι όταν ως είσοδος για τη παραγωγή δίνονται λιγότερα συνηθισμένες νότες τα μοντέλα μπορούσαν να συνεχίσουν την μελωδία δυσκολότερα από ότι άλλες φορές και κατέληγαν πολύ σύντομα να ανακυκλώνουν την έξοδό τους (παραπάνω στην αξιολόγηση των μοντέλων). Το πρόβλημα αυτό ήταν πιο μεγάλο και πιο αισθητό στα μοντέλα που είχαν εκπαιδευτεί με τα δεδομένα κιθάρας.

Για να λυθεί το παραπάνω πρόβλημα ……………………………………………..

Στο ισορροπημένο dataset υπάρχουν 93 διαφορετικές τιμές pitch ενώ το Velocity και ο χρόνος παραμένει ο ίδιος

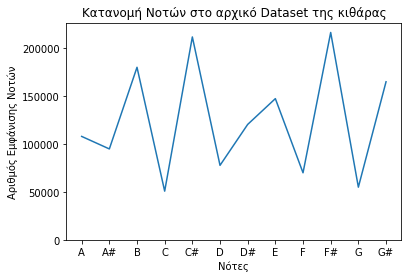


Όπως φαίνεται και από το παραπάνω διάγραμμα τα δεδομένα αυτά είναι πιο ισορροπημένα καθώς η πιο συχνή νότα είναι η A η οποία εμφανίζεται 124.849 ενώ η πιο ασυνήθιστη είναι η Β όπου εμφανίζεται 114.941 φορές (η διαφορά μειώθηκε σε 9.908), **ενώ η διασπορά του συνόλου εμφανίσεων κάθε νότας υποτριπλασιάστηκε**.

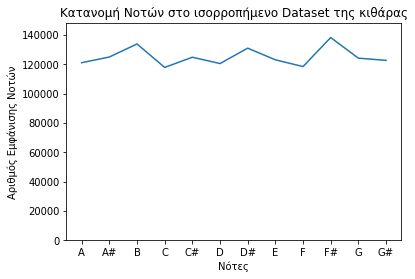
Τελικά κρατήθηκαν και τα 2 παραπάνω σύνολα δεδομένων και εκπαιδευτήκαν ορισμένα μοντέλα και με αυτά για να μπορούν να συγκριθούν και να αξιολογηθούν τα αποτελέσματά τους καθώς και η επίδραση του bias στην σύνθεση.

**4.2.2 ΔΕΔΟΜΕΝΑ ΑΠΟ ΚΙΘΆΡΑ**

Σε αυτό το σύνολο δεδομένων υπάρχουν 711 κομμάτια τα όποια δημιουργούν μια ακολουθία επίσης 1.499.904 νοτών (για να υπάρχει μια ομοιομορφία στην εκπαίδευση των μοντέλων). Σε αυτό υπάρχουν 68 διαφορετικές τιμές Pitch, 124 διαφορετικές τιμές Velocity και 178 διαφορετικές τιμές χρόνου. Όπως και με το σύνολο δεδομένων του πιάνου έτσι και σε αυτό δεν χρησιμοποιήθηκαν κάποιες ειδικές τιμές νοτών για την έναρξη ή την λήξη κομματιών ή ακολουθιών. Στο παρακάτω διάγραμμα παρουσιάζεται ο αριθμός εμφάνισης κάθε νότας στο σύνολο δεδομένων της κιθάρας.



Στο dataset αυτό όπως φαίνεται και από το παραπάνω διάγραμμα το πρόβλημα της ανομοιομορφίας των δεδομένων είναι πολύ μεγαλύτερο από ότι σε αυτό του πιάνο. Εδώ η πιο χρησιμοποιούμενη νότα είναι η F# η οποία εμφανίζεται 217.460 φορές ενώ η ελάχιστα συνηθισμένη νότα είναι η C η οποία εμφανίζεται 50.188 (διαφορά 167.272 φορές). Στο ισορροπημένο dataset η πιο συχνή νότα είναι η F# η οποία εμφανίζεται 137.179 φορές ενώ η ελάχιστα συνηθισμένη είναι η C η οποία εμφανίζεται 127.813 φορές (η διαφορά μειώθηκε σε 9366), **ενώ η διασπορά του συνόλου εμφανίσεων υποδεκαπλασιάστηκε**. Στο παρακάτω διάγραμμα παρουσιάζεται ο αριθμός εμφανίσεων των νοτών στο ισορροπημένο dataset της κιθάρας.



Όπως φαίνεται και από το παραπάνω διάγραμμα το σύνολο αυτών των δεδομένων είναι πλέον πολύ πιο ισορροπημένο και συνεπώς θα αποφεύγονται διάφορα προβλήματα κατά την εκπαίδευση. Όπως και με το dataset του πιάνου τα μοντέλα εκπαιδευτήκαν και με το αρχικό αλλά και με το ισορροπημένο dataset για να μπορεί να γίνει αξιολόγηση των αποτελεσμάτων της ανομοιομορφίας αυτής στην σύνθεση των κομματιών.

**4.2.3 ΚΟΙΝΑ ΔΕΔΟΜΈΝΑ**

Σε αυτό το σύνολο δεδομένων υπάρχουν 1.334 κομμάτια (623 συν 711) τα όποια αποτελούν συνδυασμό των δυο παραπάνω (αρχικών) συνόλων.

Όπως έχει αναφερθεί και προηγουμένως κάθε κομμάτι αποτελεί μια ακολουθία από νότες και συνεπώς δεν θα ήταν πρακτικό απλά να αναμειχθούν τα δύο παραπάνω σύνολα μεταξύ τους. Αντί αυτού αναμείχθηκαν τα τραγούδια μεταξύ τους και στην συνέχεια έγινε η μετατροπή των τραγουδιών σε μια ενιαία ακολουθία νοτών (που ουσιαστικά είναι το σύνολο δεδομένων). Με αυτόν τον τρόπο το δίκτυο διαβάζει τυχαία κάθε φορά κομμάτια από διαφορετικά όργανα. Η ακολουθία αυτού του συνόλου αποτελείται από 2.922.496 νότες. Αυτές περιέχουν 87 διαφορετικές τιμές Pitch και 204 διαφορετικές τιμές αξιών (διακριτές τιμές χρόνου). Όπως και στα παραπάνω σύνολα δεδομένων δεν χρησιμοποιήθηκαν κάποιες ειδικές τιμές για την έναρξη ή την λήξη τραγουδιών ή ακολουθιών για τους ίδιους λόγους που δεν χρησιμοποιήθηκαν και στα ξεχωριστά σύνολα δεδομένων.

**4.3 ΠΡΟΕΠΕΞΕΡΓΑΣΙΑ ΔΕΔΟΜΕΝΩΝ**

Όπως έχει αναφερθεί και προηγουμένως από κάθε Midi αρχείο εξάγεται μια ακολουθία νοτών η οποίες αν παιχθούν ακολουθιακά η μια πίσω από την άλλη θα ακουστεί το αρχικό κομμάτι (αγνοούνται άλλες παράμετροι όπως κανάλια κ.α. γιατί οι μελωδίες των δεδομένων είναι μονοφωνικές). Κάθε μια από τις νότες αυτές απαρτίζεται από 3 διαφορετικές μεταβλητές το pitch της νότας, το Velocity καθώς και τον χρόνο. Παρόλα αυτά τα μοντέλα εκπαιδευτήκαν στο να διαβάζουν και να προβλέπουν μόνο τις 2 από αυτές το pitch και τον χρόνο, μιας και μόνο με αυτές τις παραμέτρους δίνεται μια αρκετά καλή εικόνα της μουσικής πληροφορίας και ένας άνθρωπός (απλός χρήστης ή μουσικός) μπορεί να αξιολογήσει αν ένα κομμάτι με αυτές μονό τις τιμές είναι ενδιαφέρον ή όχι.

Συγκεκριμένα εξετάστηκαν 2 διαφορετικές προσεγγίσεις στην κωδικοποίηση αυτών των παραμέτρων. Η πρώτη ήταν το Pitch και ο χρόνος να κωδικοποιηθούν ξεχωριστά και έπειτα να γίνει η συνένωση των κωδικοποιήσεων τους. Για παράδειγμα στο αρχικό dataset του πιάνου αν μια νότα είχε Pitch 62 και η αξια της ήταν ¼ ( 7η διαφορετική τιμή) τότε η είσοδος στο δίκτυο θα ήταν η παρακάτω:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Τύπος | Pitch | | | | | | | | Αξία | | | | | |
| Θέση | 1 | 2 | … | 61 | 62 | 63 | … | 87 | 1 | … | 7 | 8 | … | 54 |
| Είσοδος | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Δηλαδή η είσοδος αποτελείται από 87 + 54 = 141 διαφορετικές τιμές (για το αρχικό σύνολο δεδομένων του πιάνου) με 139 τιμές 0 και δύο 1 (μια για το Pitch και μια για την αξία).

Η κωδικοποίηση αυτή δεν επιλέχθηκε τελικά λόγω του μεγάλου bias στις τιμές των μεταβλητών. Συγκεκριμένα στο αρχικό σύνολο του πιάνου το **57.38%** των νοτών έχουν αξία 0. Αντίστοιχο πρόβλημα υπάρχει και στο pitch, όπου η νότα με pitch 62 εμφανίζεται με συχνότητα **52.37%**. Αυτό σημαίνει ότι πάνω από τις μισές νότες έχουν το ίδιο pitch και την ίδια αξία. Αξίζει να σημειωθεί ότι τα ποσοστά αυτά προέρχονται **623 διαφορετικά** κομμάτια. Το πρόβλημα αυτό είναι πολύ εντονότερο αν τα μελετηθούν τα ποσοστά των νοτών που είναι κοινά σε κάθε κομμάτι ή σε τμήματα των κομματιών, π.χ. σε κάθε 20 ή 30 νότες. Εκεί φαίνεται ότι αν για παράδειγμα σπάσουμε την αρχική ακολουθία (των 1.500.000 νοτών) σε κομμάτια μήκους **30 νοτών** τότε τα κομμάτια όπου έχουν πάνω από το **80%** του pitch τους ίδια είναι πάνω από **7000** ενώ υπάρχουν **3125** κομμάτια όπου **και οι 30 νότες έχουν ακριβώς το ίδιο pitch**. Στο πεδίο του χρόνου επειδή οι διαφορετικές τιμές είναι λιγότερες το πρόβλημα είναι πιο αισθητό. Συγκεκριμένα με το ίδιο μήκος ακολουθιών τα κομμάτια των οποίων οι νότες έχουν πάνω από το **70%** των αξιών τους ίδια είναι **227.497 (15.91%)** ενώ πάνω από **90%** είναι **11.597** και τελικά σε **9145** τέτοια κομμάτια και οι **30 νότες έχουν ακριβώς την ίδια αξία**. Όπως είναι λογικό όσο μειώνεται το μήκος των ακολουθιών που μελετώνται τόσο το παραπάνω πρόβλημα μεγαλώνει. Παρακάτω παρουσιάζεται ένας συνοπτικός πίνακας με τις παραπάνω πληροφορίες ανά σύνολο δεδομένων και μήκος ακολουθίας.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Pitch | | | | Αξία | | | |
|  | 20%< | >70% | >80% | =100% | 20%< | >70% | >80% | =100% |
| Πιάνο | 12.584 | 10.741 | 6.243 | 2.863 | 4 | 228.732 | 49.392 | 11.488 |
| Κιθάρα | 5.144 | 32.460 | 17.149 | 9.878 | 3.089 | 175.930 | 40.314 | 610 |

Seq\_length = 20

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Pitch | | | | Αξία | | | |
|  | 20%< | >70% | >80% | =100% | 20%< | >70% | >80% | =100% |
| Πιάνο | 59.443 | 9.971 | 5.372 | 2.040 | 0 | 227.497 | 48.479 | 9.145 |
| Κιθάρα | 37.338 | 25.337 | 12.064 | 5.890 | 4.897 | 173.229 | 39.082 | 460 |

Seq\_length = 30

Οι παραπάνω πληροφορίες είναι καίριας σημασίας διότι ένα νευρωνικό δίκτυο δεν εξετάζει κάθε φορά ολόκληρο το σύνολο δεδομένων ούτε ολόκληρα τραγούδια αλλά κομμάτια αυτών μήκους 20 ή 30 νοτών (μεγαλύτερές ακολουθίες αυξάνουν το βάθος του δικτύου και συνεπώς αυξάνουν κατά πολύ τον χρόνο αλλά και την ποιότητα εκπαίδευσης βλ. παρακάτω). Συνεπώς η πραγματική εικόνα της εισόδου που βλέπουν τα δίκτυα είναι αυτή που παρουσιάστηκε στους παραπάνω πίνακες στην οποία τα δεδομένα κατανέμονται πολύ ανομοιόμορφα.

Αυτή η ασσυμετρία όπως έχει αναφερθεί και παραπάνω δυσκολεύει κατά πολύ την εκμάθηση. Οι πρώτες απόπειρες εκπαίδευσης των μοντέλων με αυτή την κωδικοποίηση της εισόδου δεν έδωσαν καθόλου καλά αποτελέσματα μιας και η μάθηση γινόταν παρά πολύ γρήγορά αλλά κατά την πρόβλεψη, όπως αναμενόταν, δεν έφεραν κανένα ουσιαστικό- ενδιαφέρον αποτέλεσμα (ανακύκλωναν συνεχώς της έξοδό τους).

Για να αποφευχθεί το παραπάνω πρόβλημα χρησιμοποιήθηκε διαφορετική κωδικοποίηση της εισόδου. Σε αυτήν το pitch και η αξία κάθε νότας συνενώνονται πριν την τελική κωδικοποίηση σχηματίζοντας μια νέα δομή όπου κάθε τέτοια απαρτίζεται από τις δυο αυτές παραμέτρους. Για παράδειγμα η νότα με Pitch 62 και αξία ¼ είναι πλέον η νότα 956 όπου κωδικοποιείται ως εξής:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Θέση | 0 | 1 | 2 | … | 955 | 956 | 957 | … | 2048 |
| Είσοδος | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Δηλαδή σε αυτήν την περίπτωση με την θέση της νότας εννοείται και το pitch αλλά και η αξία της. Η αλλαγή αυτή έδειξε μείωση της ανομοιομορφίας της εισόδου και το δίκτυο πλέον «βλέπει» μια πιο ομοιόμορφη κατανομή. Αυτό οφείλεται στο γεγονός ότι η θέση της νότας αλλάζει κάθε φορά είτε αν αλλάξει το Pitch είτε αν αλλάξει η αξία της. Παρακάτω παρουσιάζονται οι ανάλογοι πίνακες που παρουσιάστηκαν και προηγουμένως, με το ποσοστό των κοινών τιμών στα κομμάτια των εισόδου για μήκος ακολουθιών 20 και 30 και για τα αρχικά σύνολα δεδομένων των δυο διαφορετικών οργάνων.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 20%< | >70% | >80% | =100% |
| Πιάνο | 23.449 | 8.669 | 3.585 | 22 |
| Κιθάρα | 27.707 | 13.556 | 3.608 | 79 |

Seq\_length = 20

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 20%< | >70% | >80% | =100% |
| Πιάνο | 80.269 | 8.744 | 3.360 | 0 |
| Κιθάρα | 84.828 | 12.181 | 3.100 | 40 |

Seq\_length = 30

Η κωδικοποίηση αυτή έδειξε πολύ καλύτερα αποτελέσματα σε σχέση με την προηγούμενη και η τελική εκπαίδευση των μοντέλων έγινε με τις εισόδους όπως περιγράφηκαν παραπάνω. Ο συνολικός αριθμός των πιθανών τιμών είσοδού δεν αυξήθηκε σημαντικά. Πλέον στο αρχικό σύνολο δεδομένων του πιάνο υπάρχουν 2060 διαφορετικές τιμές, στις κιθάρας 2440 ενώ στο αποκεινού dataset (το οποίο είναι σχεδόν διπλάσιο σε μέγεθος) υπάρχουν 3.326. Στα ισορροπημένα σύνολα νοτών η εικόνα δεν διαφέρει σημαντικά όπου για το πιάνο έχουμε 2182 διαφορετικές τιμές ενώ για την κιθάρα 2764.

**Κεφάλαιο 5 – ΕΠΙΛΟΓΉ ΥΠΕΡΠΑΡΑΜΕΤΡΩΝ**

Για την εκπαίδευση των μοντέλων τα δεδομένα κάθε ενός dataset χωριστήκαν το κάθε ένα σε 3 διαφορετικά κομμάτια, ως είθισται:

* Το σύνολο εκπαίδευσης (train set) το οποίο χρησιμοποιείται για την εκπαίδευση του εκάστοτε μοντέλου. Αποτελεί το 60- με 70% του συνόλου των δεδομένων.
* Το σύνολο ανάπτυξης (development set) το οποίο περιέχει δεδομένα από την ίδια κατανομή με το σύνολο εκπαίδευσης και χρησιμοποιείται για την αξιολόγηση του μοντέλου καθώς και για την διασπορά των δεδομένων.
* Σύνολο δοκιμής (test set), το οποίο περιέχει δεδομένα από διαφορετική κατανομή από το σύνολο εκπαίδευσης και χρησιμοποιείται για την τελική αξιολόγηση των μοντέλων καθώς και της πόλωσης των δεδομένων.

Ο χωρισμός των δεδομένων σε train και dev sets γίνεται τυχαία πριν από κάθε εκτέλεση. Από την άλλη το σύνολο δοκιμής (test set) είναι το ίδιο σε όλες τις εκτελέσεις των αλγορίθμων εκτέλεσης των μοντέλων. Η επιλογή αυτή έγινε για να μπορέσει να υπάρχει μια κοινή βάση σε όλα τα μοντέλα και να μπορέσουν να συγκριθούν τα αποτελέσματά τους.

Επίσης σημαντικό ρόλο στην ταχύτητα αλλά και την ποιότητα εκπαίδευσης διαδραματίζουν και οι επιλογές των υπερπαραμέτρων των μοντέλων. Παρακάτω παρουσιάζονται μερικές από αυτές αλλά και τα κριτήρια με τα οποία έγιναν οι επιλογές τους.

**5.1 Συναρτήσεις Ενεργοποίησης (Activation Functions)**

Ως συνάρτηση ενεργοποίησης ορίζεται μια μη-γραμμική συνάρτηση η οποία παραβάλλεται μεταξύ επιπέδων και ουσιαστικά τα διαχωρίζει μεταξύ τους. Για παράδειγμα σε ένα απλό Feed Forward δίκτυο με εκατό Hidden Layer (δηλαδή βάθους 100) αν δεν παραβάλλεται κάποια μη γραμμική συνάρτηση μεταξύ των επιπέδων τότε αυτό είναι ακριβώς ισοδύναμο με ένα δίκτυο ενός Hidden Layer με ανάλογο αριθμό βαρών (χάνεται η έννοια του βάθους) μιας και μπορούμε να γράψουμε την έξοδο ως συνδυασμό γραμμικών εισόδων.

Υπάρχουν διάφορες επιλογές μεταξύ των συναρτήσεων και οι πιο διαδεδομένες παρουσιάζονται παρακάτω:

* Sigmoid Activation (σ): Η συνάρτηση αυτή έχει αρκετά καλές ιδιότητες όπως ότι είναι μη- γραμμική, διαφορίσιμη καθώς και το πεδίο τιμών τις είναι (0,1) και μπορούν να εκφραστούν τα αποτελέσματα της ως πιθανότητες. Παρόλα αυτά έχει το μειονέκτημα ότι η μέγιστή τιμή της παραγώγου της sigmoid είναι πολύ μικρή, μόλις 0.25. Αυτό δημιουργεί πρόβλημα κατά την εκπαίδευση βαθιών αρχιτεκτονικών, μιας και στον back propagation περνιέται μόνο ένα μικρό μέρος του σφάλματος στα πίσω επίπεδα, με αποτέλεσμα το δίκτυο τα πρώτα επίπεδά των δικτύων να εκπαιδεύονται πολύ πιο αργά (Vanishing Gradient) .
* Tanh: Η συνάρτηση αυτή είναι επίσης μια μη- γραμμική και διαφορίσιμή συνάρτηση . Σε αντίθεση με την sigmoid όμως δεν το πεδίο τιμών της είναι στο (-1, 1), το οποίο δεν είναι υποσύνολό του πεδίου τιμών των πιθανοτήτων. Παρόλα αυτά ή μέγιστή τιμή της παραγώγου της είναι 1 με αποτέλεσμα να μην παρουσιάζονται τα προβλήματα της πρώτης.
* ReLU (Rectified Linear Unit): Το πεδίο τιμών της ReLU είναι το (0, +∞) και επίσης είναι μη- διαφορίσιμή στο 0 (γενικά υπάρχει λύση για το πρόβλημα αυτό). Το μεγάλο θετικό αυτής της οικογένειας συναρτήσεων (ReLU, PReLU, RPeLu) είναι ότι η τιμή της παραγώγου για τιμές > 0 είναι πάντα 1, δηλαδή κάθε φορά περνιέται το μέγιστο δυνατό σφάλμα στα προηγούμενα επίπεδα κατά την εκτέλεση του back propagation.
* Softmax: Η συνάρτησή αυτή χρησιμοποιείται σαν activation function στο τελευταίο επίπεδο των δικτύων που λύνουν προβλήματα ταξινόμησης (Logistic Regression). Η εξίσωση αυτή παρουσιάζεται παρακάτω:

Όπου Ν είναι το πλήθος των πιθανών προβλέψεων των εξόδων, δηλαδή αν έχουμε 4 πιθανές εξόδους (συνεπώς κατανομή 4 παραμέτρων) τότε το Ν=4.

Δηλαδή η έξοδος της softmax εκφράζει την πιθανότητα κάθε μιας από τις εξόδους ενός επιπέδου να είναι η πραγματική. Οπότε το άθροισμα όλων των εξόδων ενός επιπέδου ισούται με 1 και συνεπώς η έξοδος της softmax είναι μια κατανομή Ν μεταβλητών.

Για τους παραπάνω λόγους ως activation function στο τελευταίο επίπεδο χρησιμοποιείται η softmax, ενώ σε άλλα επίπεδα όπως στο εσωτερικό του LSTM χρησιμοποιείται η sigmoid και η tanh. Τέλος ανάμεσα των πλήρως συνδεδεμένων επιπέδων χρησιμοποιείται η ReLU. Η παραπάνω επιλογές είναι και οι πιο συνήθης στην βιβλιογραφία.

**5.2 Συναρτήσεις Κόστους**

Η συνάρτηση κόστους στο πλαίσιο των νευρωνικών δικτύων είναι η συνάρτηση που επιθυμούμε να ελαχιστοποιήσουμε και ουσιαστικά εκφράζουν την απόσταση των προβλεπόμενων τιμών από τις πραγματικές.

Σε όλα τα μοντέλα ως συνάρτηση κόστους χρησιμοποιήθηκε η κατηγορηματική διασταυρούμενη εντροπία (categorical cross entropy). Η συνάρτηση αυτή όπως αναφέρθηκε και προηγουμένως υπολογίζει την διαφορά – απόσταση μεταξύ της πραγματικής και της προβλεπόμενης εξόδου. Πρακτικά έστω ότι η πραγματική έξοδος έχει την κατανομή ενώ η έξοδος έπειτα από κάποια πρόβλεψη έχει την κατανομή Το αποτέλεσμα της categorical cross entropy είναι:

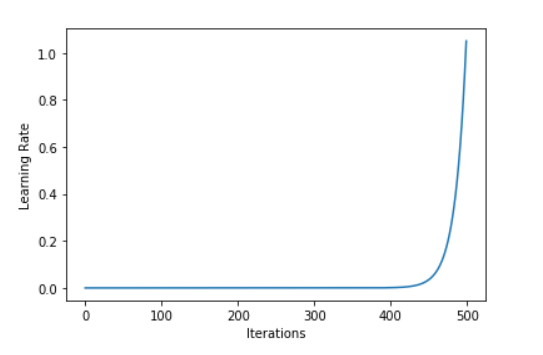
Όπου Ν είναι το πλήθος των πιθανών προβλέψεων των εξόδων. Συνεπώς κατά την εκπαίδευση ενός δικτύου με αυτήν την συνάρτηση κόστους προσπαθούμε να κάνουμε την προβλεπόμενη κατανομή κάθε στιγμή να ταυτίζεται με την πραγματική. Σημαντικό για την λειτουργία των παραπάνω είναι η προβλεπόμενη έξοδος να είναι πραγματικά μια κατανομή δηλαδή κάθε έξοδος να αποτελείτε από Ν αριθμός με πεδίο ορισμού το [0, 1] όπου θα αθροίζουν στο 1. Συνεπώς από τα παραπάνω η μοναδική συνάρτηση ενεργοποίησης που πλήρη της προϋποθέσεις αυτές είναι η softmax (και η sigmoid έχει κατάλληλο πεδίο τιμών αλλά τα αποτελέσματα της δεν αποτελούν κατανομή μιας και δεν αθροίζουν στο 1). Η σύνθεση αυτή χρησιμοποιείται σχεδόν πάντα στην βιβλιογραφία σε προβλήματα ταξινόμησης.

**5.3 Ρυθμός Μάθησης (Learning Rate)**

Επίσης σημαντική παράμετρος ενός δικτύου είναι ο ρυθμός μάθησης ο οποίος καθορίζει πόσο μεγάλα θα είναι οι αλλαγές που θα γίνονται στα βάρη του δικτύου σε κάθε βήμα εκπαίδευσης. Ακόμα και σε προσαρμοστικούς βελτιστοποιήτες (όπως είναι ο Adam και Rmsprop) η αρχικοποίηση του ρυθμού μάθησης παίζει μεγάλο ρόλο στην ταχύτητά εκπαίδευσης.

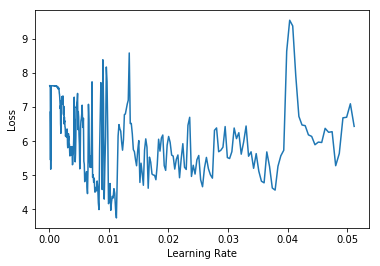
Αν ο ρυθμός μάθησης είναι πολύ μικρός τότε η διαδικασία της εκπαίδευσής είναι μεν πιο αξιόπιστη, αλλά πολύ πιο αργή μιας και τα βήματα μέχρι να οδηγηθεί, η συνάρτηση κόστους, σε κάποιο ελάχιστο είναι πολύ μικρά. Αντίθετα αν ο ρυθμός μάθησης είναι πολύ μεγάλος τότε μπορεί η διαδικασία να μην συγκλίνει. Αυτό μπορεί να συμβεί αν οι αλλαγές στις παραμέτρους είναι τόσο μεγάλες που να υπερβαίνουν το ελάχιστο με αποτέλεσμα να χειροτερεύει συνεχώς το σφάλμα. Συνεπώς πρέπει να υπάρξει μια κατάλληλη ρύθμιση του ρυθμού εκπαίδευσης κάπου ανάμεσα ώστε ο αλγόριθμός να συγκλίνει σε κάποιον λογικό χρόνο. Η συνήθης πρακτική για την εύρεση της κατάλληλης τιμής αυτής της παραμέτρου είναι η δοκιμή μεταξύ διαφόρων τιμών και η χειροκίνητη αλλαγή τους με βάση τους παραπάνω κανόνες.

Στα δίκτυα που κατασκευάστηκαν στα πλαίσια της διπλωματικής αυτής ο ρυθμός εκπαίδευσης επιλέχθηκε με βάση έναν αυτόματο τρόπο[1]. Αρχικά η εκπαίδευση ξεκινά με έναν πολύ μικρό ρυθμό εκπαίδευσης ο οποίος αυξάνεται εκθετικά μετά από κάθε mini-batch όπως φαίνεται στο παρακάτω διάγραμμα.



Αρχικά μιας και ο ρυθμός μάθησης είναι πολύ μικρός το σφάλμα σε κάθε midi-batch θα παραμένει σταθερό (μαθαίνει πολύ αργά το δίκτυο). Στην συνέχεια καθώς ο ρυθμός αυξάνεται το σφάλμα θα αρχίσει να μειώνεται όλο και πιο γρήγορα μέχρι το σημείο όπου ο ρυθμός θα μεγαλώσει τόσο που το σφάλμα απλά θα ταλαντώνεται και θα αυξάνεται συνεχώς. Η ιδανική συνεπώς τιμή του ρυθμού μάθησης είναι αυτή για την οποία το σφάλμα μειώνεται με τον μεγαλύτερο δυνατό ρυθμό. Αυτή η τιμή θα κάνει το δίκτυο να μαθαίνει όσο τον δυνατόν γρηγορότερα χωρίς ταυτόχρονα να το περιορίζει.

Ενδεικτικά παρακάτω παρουσιάζεται η καμπύλη του σφάλματος σε σχέση με τον ρυθμό μάθησης για την αρχιτεκτονική του Autoncoder με lstm\_cells = 512, seq\_length\_in = 30, seq\_length\_out = 30, batch\_size = 64.



Από την παραπάνω γραφική παράσταση φαίνεται ότι για αυτό το μοντέλο ο ιδανικός ρυθμός μάθησης είναι περίπου στο 0.001. Με αντίστοιχο τρόπο υπολογίστηκαν και οι ρυθμοί μάθησης για τα υπόλοιπα μοντέλα που εκπαιδευτήκαν.

**5.4 Βελτιστοποιήτες (Optimizers)**

Η επιλογή του κατάλληλου βελτιστοποιήτη είναι μια πολύ σημαντική υπερπαράμετρος η οποίο επηρεάζει σημαντικά τον χρόνο εκπαίδευσης των δικτύων. Η επιλογή αυτού εξαρτάται περισσότερο από το είδος και την δομή του δικτύου και λιγότερο από τα δεδομένα εκπαίδευσης. Για τον λόγο αυτόν δεν έχουν χρησιμοποιηθεί οι ίδιοι βελτιστοποιήτες για τις 2 διαφορετικές αρχιτεκτονικές που έχουν χρησιμοποιηθεί (Feed Forward LSTM και Seq2Seq Encoder- Decoder). Η πρώτη έχει εκπαιδευτεί με βάση τον αλγόριθμό rmsprop ενώ η δεύτερη χρησιμοποιώντας τον Adam. Αυτή η επιλογή είναι και η πιο συνήθης επιλογή στην βιβλιογραφία, μιας και για τα συγκεκριμένα δίκτυα έχουν επιδείξει ταχύτατα αποτελέσματα εκπαίδευσης.

Γενικότερα η πιο κλασσική και η πιο ευρέως δοκιμασμένη τεχνική εκπαίδευσης νευρωτικών δικτύων είναι η μέθοδος κατάβασης δυναμικού (Stochastic Gradient Decent ή SGD). Όλοι οι αλγόριθμοί εκπαίδευσης έχουν κοινή αρχή λειτουργιάς με αυτή του SGD και για αυτό αναφέρεται συνοπτικά ο τρόπος λειτουργίας του.

Ο SGD στηρίζεται στην ανανέωση όλων των παραμέτρων ανάλογα με την τιμή των παραγώγων πρώτης τάξης της συνάρτησης κόστους ως προς τις παραμέτρους του δικτύου την δεδομένη χρονική στιγμή. Η αλλαγή των τιμών αυτή γίνεται προς την κατεύθυνση της αρνητικής κλίσης με σκοπό ουσιαστικά την ελαχιστοποίηση της συνάρτησης κόστους. Παρόλα αυτά το γεγονός ότι η μέθοδος αυτή είναι πρώτης τάξης σημαίνει ότι λαμβάνεται υπόψιν η κλίση της συνάρτησης κόστους αλλά όχι η καμπυλότητά της με αποτέλεσμα να μην προσαρμόζεται αυτόματα ο ρυθμός μάθησης αλλά να παραμένει ορισμένος από την αρχή, ανεξάρτητα την τρέχουσα κατάσταση.

Και οι δύο μέθοδοι που έχουν χρησιμοποιηθεί είναι δεύτερης τάξης και για αυτό ανήκουν στην κατηγορία των προσαρμοστικών αλγορίθμων μάθησης.

*RMSPROP*: Ο αλγόριθμός αυτός όπως αναφέραμε και προηγούμενος είναι ένας προσαρμοστικός αλγόριθμος μάθησης και συνεπώς δεν χρειάζεται προσαρμογή ο ρυθμός μάθησης μιας και ρυθμίζεται αυτόματα. Επίσης κατασκευάζει ένα ξεχωριστό ρυθμό μάθησης για κάθε μια παράμετρο του δικτύου ξεχωριστά. Οι εξισώσεις ανανέωσης των βαρών για κάθε μια παράμετρο παρουσιάζονται παρακάτω:

όπου:

* : η αρχική τιμή του ρυθμού μάθησης
* : Ο εκθετικός μέσος όρος των τετραγώνων της παραγώγου
* : Η παράγωγος την χρονική στιγμή t σε σχέση με την παράμετρο του δικτύου.
* : υπερπαράμετρος του αλγορίθμου (προτεινόμενη τιμή 0.9)

[ http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture\_slides\_lec6.pdf]

Γενικά η αυτή η μέθοδος συνιστάτε για αναδρομικά δίκτυα (RNN) όπως το πρώτο μοντέλο που χρησιμοποιήθηκε[https://keras.io/optimizers].

*ADAM*: η μέθοδος εκπαίδευσης Adam είναι άλλη μια προσαρμοστική μέθοδος που υπολογίζει και αυτή έναν διαφορετικό ρυθμό μάθησης για κάθε μια παράμετρο του δικτύου. Οι εξισώσεις ανανανέωσης κάθε μιας παραμέτρου παρουσιάζονται παρακάτω:

*Όπου:*

* : η αρχική τιμή του ρυθμού μάθησης
* : η παράγωγος την χρονική στιγμή ως προς την παράμετρο
* : υπερπαραμέτροι αλγορίθμου, όπως γ στον RMSPROP
* : ο εκθετικός μέσος όρος της παραγώγου
* : ο εκθετικός μέσος όρος των τετραγώνων των παραγώγων

Ο αλγόριθμος αυτός έχει χρησιμοποιηθεί σε πολλές διαφορετικές αρχιτεκτονικές με εξαιρετικά αποτελέσματα [https://arxiv.org/pdf/1412.6980.pdf].

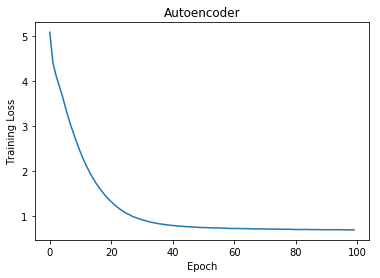
Γενικά στα μοντέλα χρησιμοποιήθηκαν και άλλοι βελτιστοποιήτες αλλά οι συγκεκριμένη έδειξαν τα καλύτερα αποτελέσματα.

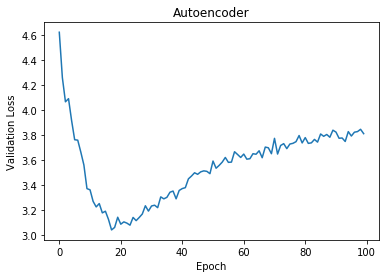
**5.5 Συστηματοποιήση – Regularization**

Η συνεχής αύξηση της υπολογιστικής δύναμης δίνει την δυνατότητα εκπαίδευσης πιο σύνθετών μοντέλων, με μεγαλύτερο αριθμό παραμέτρων. Αυτό έχει σαν αποτέλεσμα για την επίλυση προβλημάτων να χρησιμοποιούνται πιο μεγάλα μοντέλα από αυτά που είναι απαραίτητα. Παρόλα αυτά οι πιο σύνθετες και πολύπλοκες αρχιτεκτονικές μπορούν να δημιουργούν φαινόμενα υπερεκαπαίδευσης (overfittng).

Το πρόβλημα αυτό ουσιαστικά πηγάζει από το γεγονός ότι το μοντέλο εκπαιδεύτηκε τόσο πολύ που πλέον έχει απομνημονεύσει τα δεδομένα εκπαίδευσής και έχει πολύ μικρό σφάλμα για αυτά. Πρακτικά όμως το δίκτυο δεν έχει μάθει να λύνει το πρόβλημα (δεν έχει γενικεύσει σωστά) και συνεπώς το σφάλμα, όταν αυτό δοκιμάζεται σε άλλα δεδομένα (όπου αυτός είναι και ο ρόλος του), αυξάνεται συνεχώς.

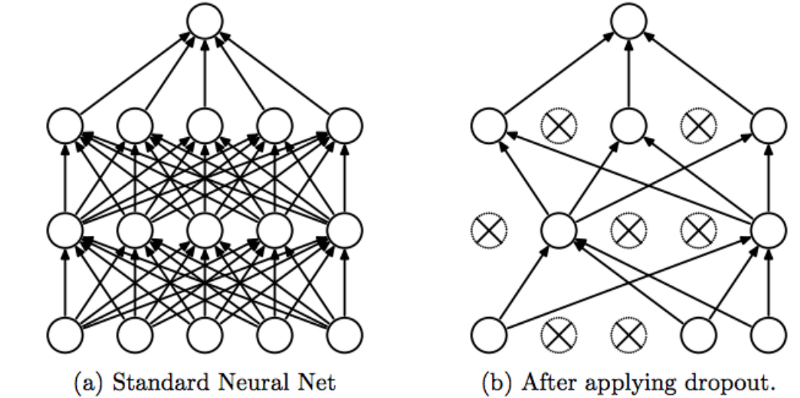
Παρακάτω φαίνεται η γραφική παράσταση του του σφάλματος κατά την διάρκεια εκπαίδευσης του Autoencoder με lstm\_cells = 512, seq\_length\_in = 20, seq\_length\_out = 20.





Εκ πρώτης όψεις φαίνεται ότι το σφάλμα του δικτύου συνεχώς μειώνεται άλλα στην πραγματικότητα το validation loss από ένα σημείο και έπειτα αυξάνεται συνεχώς. Αυτή είναι μια κλασσική εικόνα ενός δικτύου το οποίο έχει υπερεκπαιδευτεί.

Ο τρόπος που αντιμετωπίζεται το overfitting είναι η συστηματοποιήση- regularization. Υπάρχουν πολλές τεχνικές συστηματοποιήσης αλλά στα πλαίσια της διπλωματικής χρησιμοποιήθηκε η Dropout. Η τεχνική αυτή αυτή είναι μια τεχνική η οποία βοηθά στην μείωση της εξάρτησης μεταξύ των νευρώνων[4]. Κατά την διαδικασία εκπαίδευσης (training phase) για κάθε νευρώνα κάθε επιπέδου με Dropout και σε κάθε επανάληψη αγνοούνται , δηλαδή μηδενίζονται οι έξοδοι, ορισμένων κόμβων (ποσοστού p), ενώ ταυτόχρονα κλιμακώνονται οι έξοδοι των υπολοίπων. Πρακτικά αν στην έξοδο ενός επιπέδου εφαρμοστεί Dropout με ποσοστό 0.5 τότε σε κάθε βήμα εκπαίδευσης θα επιλέγονται τυχαία και θα μηδενίζονται οι έξοδοι των μισών κόμβων ενώ τον άλλων μισών θα διπλασιάζονται. Η κλιμάκωση των εξόδων γίνεται για να διατηρείτε η κατανομή των εξόδων σε κάθε βήμα. Με αυτόν τον τρόπο ουσιαστικά δεν εκπαιδεύεται μόνο ένα δίκτυο αλλά πολλά δίκτυα μαζί μιας και σε κάθε βήμα εκπαίδευσης αλλάζουν οι εσωτερικές συνδέσεις και συνεπώς η διαρρύθμισή του. Αυτό επίσης μειώνει όπως είναι λογικό και την εξάρτηση των τιμών που έχει ο κάθε νευρώνας από τους υπολοίπους στο ίδιο επίπεδο. Παρακάτω παρουσιάζεται μια εικόνα ενός Feed Forward νευρωνικού δικτύου σε ένα βήμα εκπαίδευσης όπου ανάμεσα από κάθε επίπεδο εφαρμόζεται η παραπάνω τεχνική.



Srivastava, Nitish, et al. ”Dropout: a simple way to prevent neural networks from

overfitting”, JMLR 2014

Τέλος κατά την πρόβλεψη χρησιμοποιούμε όλες τις εξόδους των νευρώνων απλά της μειώνουμε κάθε φορά κατά τον παράγοντα p (δηλαδή στο παραπάνω παράδειγμα τις μειώνουμε κατά 2).

**5.6 Αρχικοποίηση Παραμέτρων**

Πριν από την εκπαίδευση κάθε μοντέλου πρέπει να γίνει η αρχικοποίηση των παραμέτρων του. Η αρχικοποίηση αυτή επηρεάζει την απόδοση και την ταχύτητα σύγκλισης του δικτύου. Υπάρχουν διάφορες τεχνικές για την αρχικοποίηση των παραμέτρων ανάλογα τον τύπο του δικτύου, τις activation function κάθε επιπέδου των αριθμό των παραμέτρων κ.α.

Γενικότερα οι παράμετροι του δικτύου αρχικοποιούνται με τυχαίες μεταβλητές της κανονικής κατανομής (με mean = 0, stdv = 1). Η αρχικοποιήση αυτή όμως μπορεί να δυσκολέψει την εκμάθηση λόγω των προβλημάτων Exploding ή Vanishing Gradient. Για παράδειγμα έστω ένα επίπεδο με Ν νευρώνες εισόδου που συνδέονται σε έναν κόμβο εξόδου. Για απλότητα όλες οι είσοδοι είναι 1 ενώ όλα τα βάρη έχουν αρχικοποιηθεί με την κανονική κατανομή με mean = 0 και stdv= 1. ( Τότε το αποτέλεσμα της άθροισης στον κόμβο εξόδου θα είναι το ίδιο με το άθροισμα των τιμών της κανονικής κατανομής (όλες οι είσοδοι ίσες με 1), δηλαδή θα ακολουθεί και αυτή την κανονική κατανομή με μέση τιμή mean = 0 και τυπική απόκλιση = . Αυτό σημαίνει ότι αν N = 512 τότε το άθροισμα στον νευρώνα εξόδου θα ακολουθεί την κανονική κατανομή με διακύμανση 22.62. Αυτή η μεγάλη τιμή της διακύμανσης δημιουργεί πρόβλημα μιας και αν ως activation function έχει χρησιμοποιηθεί η sigmoid στον νευρώνα εξόδου, τότε τις περισσότερες φορές το αποτέλεσμα του δικτύου θα είναι 0 ή 1 (λόγω του κορεσμού της συνάρτησης ενεργοποίησης).

Για να μην δημιουργείται το παραπάνω πρόβλημα το αρχικό βάρος κάθε παραμέτρου ενός δικτύου επιλέγεται όπως προηγουμένως τυχαία από τη κανονική κατανομή και έπειτα διαιρείται με , όπου k είναι κάποιο heuristic, ενώ sizein, sizeout είναι το μέγεθος εισόδου και εξόδου αντίστοιχα. Η τεχνική αυτή ονομάζεται αρχικοποιήση Xavier[5,6,7] (ή glorot) και αυτή χρησιμοποιήθηκε σε όλα τα τελικά μοντέλα που εκπαιδευτήκαν. Το heuristic k επιλέχθηκε ίσο με 6 η όποια είναι και η προτεινόμενη τιμή [8], για αναδρομικά μοντέλα, όπως αυτά που κατασκευάστηκαν.

**ΚΕΦΆΛΑΙΟ 6 – ΚΟΜΜΆΤΙΑ ΔΙΚΤΎΟΥ**

**6.1 Πλήρες Συνδεδεμένο Επίπεδο (Fully Connected Layer)**

Πλήρες συνδεδεμένο (Fully Connected Layers) ονομάζεται ένα επίπεδο του οποίου όλοι οι νευρώνες συνδέονται όλους τους νευρώνες του προηγούμενου επιπέδου. Η εξίσωση που το περιγράφει είναι:

Όπου W, b είναι εκπαιδεύσιμες παράμετροι του δικτύου. Ο συνολικός αριθμός παραμέτρων που χρησιμοποιούνται είναι το άθροισμα των μεγεθών των πινάκων W, b, δηλαδή:

Όπου nin το μέγεθος εισόδου και nout το σύνολο των νευρώνων του επιπέδου, δηλαδή το μέγεθος της εξόδου.

**6.2 Embedding**

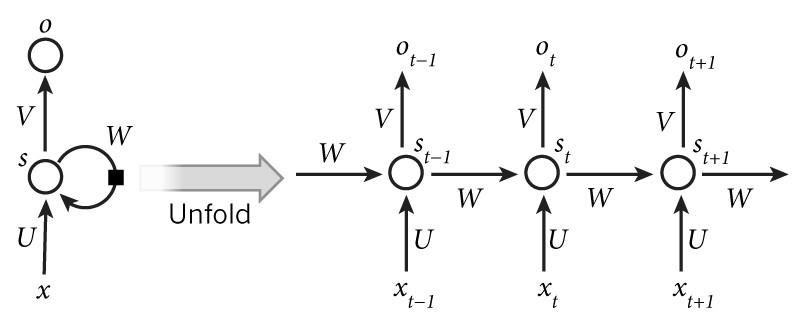
Γενικότερα η είσοδος κατηγορηματικών τιμών σε ένα νευρωνικό δίκτυο είναι ένα διάνυσμα με μηδενικά και ένα μοναδικό 1 στην θέση της αντίστοιχης τιμής (one hot encoded). Δηλαδή αν έχουμε 4 πιθανές τιμές εισόδου το διάνυσμα εισόδου για την είσοδο με τιμή 3 είναι το [0, 0, 1, 0]. Η αναπαράσταση αυτή είναι πολύ ακριβή μιας και για εισόδους με πολλές πιθανές τιμές (> 2000 όπως στην περίπτωση που εξετάζει η διπλωματική) το διάνυσμα εισόδου θα έχει μόνο ένα 1 και 2000 μηδενικά. Για την μείωση της διαστατικότητας εισόδου χρησιμοποιείται η τεχνική Embedding. Ένα Embedding Layer είναι ουσιαστικά ένα πίνακας που για κάθε τιμή εισόδου παράγει ένα διάνυσμα δεκαδικών αριθμών με συγκεκριμένου μεγέθους. Συνεπώς αν οι πιθανές είσοδοι είναι *n\_inp* και το επιθυμητό μέγεθος του διανύσματος εισόδου είναι *emb\_size* τότε ο Embedding matrix θα είναι ένας πίνακας μεγέθους *(emb\_size, n\_inp)* όπου για κάθε μια τιμή εισόδου θα παράγει ένα διάνυσμα *(1, n\_inp)* που βρίσκεται στην αντίστοιχη γραμμή. Συνεπώς και ο αριθμός των παραμέτρων του επιπέδου αυτού είναι:

Οι τιμές του πίνακα αυτού εκπαιδεύονται όπως οι παράμετροι ενός κανονικού δικτύου (back propagation).

Η χρήση της τεχνικής αυτής στην περίπτωση την σύνθεσης μουσικής έγινε για την μείωση της διαστατικότητας της εισόδου ώστε να είναι πιο διαχειρισιμη από το υπόλοιπο δίκτυο. Ο πίνακας αυτός δίνει και μια γραφική απεικόνιση των δεδομένων. Για έναν πίνακα που έχει εκπαιδευτεί με κείμενο συχνά φαίνεται ότι το αποτέλεσμα λέξεων με κοινό νόημα να έχουν πολύ μικρότερη απόσταση από λέξεις χωρίς σχέση μεταξύ τους. Έτσι αν απεικονιστούν σε δυο άξονες αυτές οι τιμές θα σχηματιστούν διάφορες γειτονίες με λέξεις.

**6.3 Αρχιτεκτονική Απλού Αναδρομικού Δικτύου (RNN-LSTM)**

Τα αναδρομικά δίκτυα χρησιμοποιούνται για την επίλυση προβλημάτων όπου υπάρχει χρονική εξάρτηση μεταξύ των δεδομένων. Για παράδειγμα ένα πρόβλημα στο οποίο υπάρχουν τέτοιου είδους εξαρτήσεις είναι στην επεξεργασία κείμενου ή μουσικής. Τα έξοδος κάθε στιγμή εξαρτάται σε μεγάλο βαθμό από τα δεδομένα που έχουν περάσει μέχρι τώρα (προηγούμενες λέξεις ή νότες) και το δίκτυο πρέπει με κάποιον τρόπο να τα ‘θυμάται’. Σχηματικά ένα αναδρομικό νευρωνικό δίκτυο παρουσιάζεται παρακάτω[10] όπου φαίνεται η είσοδος η έξοδος και το βασικό κελί:



Από το πρώτο σχήμα φαίνεται επίσης γιατί το δίκτυο ονομάζεται αναδρομικό. Ουσιαστικά σε κάθε βήμα η έξοδος υπολογίζεται όπως ακριβώς οι προηγούμενες αποθηκεύοντάς την συγχρόνως την είσοδο στην ‘μνήμή’. Στο δεξί σχήμα φαίνεται η λειτουργία του δικτύου για κάθε βήμα μιας ακολουθίας (ξεδιπλωμένο ουσιαστικά στον χρόνο). Ως st ορίζουμε την εσωτερική κατάσταση του δικτύου (hidden state). Πρακτικά αυτή είναι η μνήμη του δικτύου και υπολογίζεται από την είσοδο xt και την μέχρι τώρα πληροφορία που έχει αποθηκευτεί ως εξής: , όπου είναι κάποια activation function ενώ η έξοδος του δικτύου δίνεται από την σχέση όπου g συνήθως η softmax. Οι πράξεις αυτές ορίζουν την λειτουργία και την αρχιτεκτονική του βασικού κελιού (cell) ενός αναδρομικού δικτύου. Οι W, U, V είναι εκπαιδευσίμες παράμετροι του δικτύου και καθορίζουν την ροή και την αποθήκευση πληροφορίας σε αυτό. Αξίζει να σημειωθεί ότι για κάθε βήμα μιας ακολουθίας οι παράμετροι W,V,U είναι οι ίδιες (μοιραζόμενες παράμετροι). Θεωρητικά ένα αναδρομικό δίκτυο μπορεί να αποθηκεύσει και να ανταποκριθεί σε αυθαίρετα μεγάλες ακολουθίες εισόδου. Στην πραγματικότητα όμως κάτι τέτοιο δεν ισχύει και πρακτικά το δίκτυο ‘θυμάται’ πληροφορία μόνο μερικών προγενέστερων βημάτων[10] . Συνεπώς κατά την επεξεργασία μιας παραγράφου ή ενός μουσικού κομματιού το παραπάνω δίκτυο θα θυμάται μόνο τις τελευταίες λέξεις ή νότες αντίστοιχα.

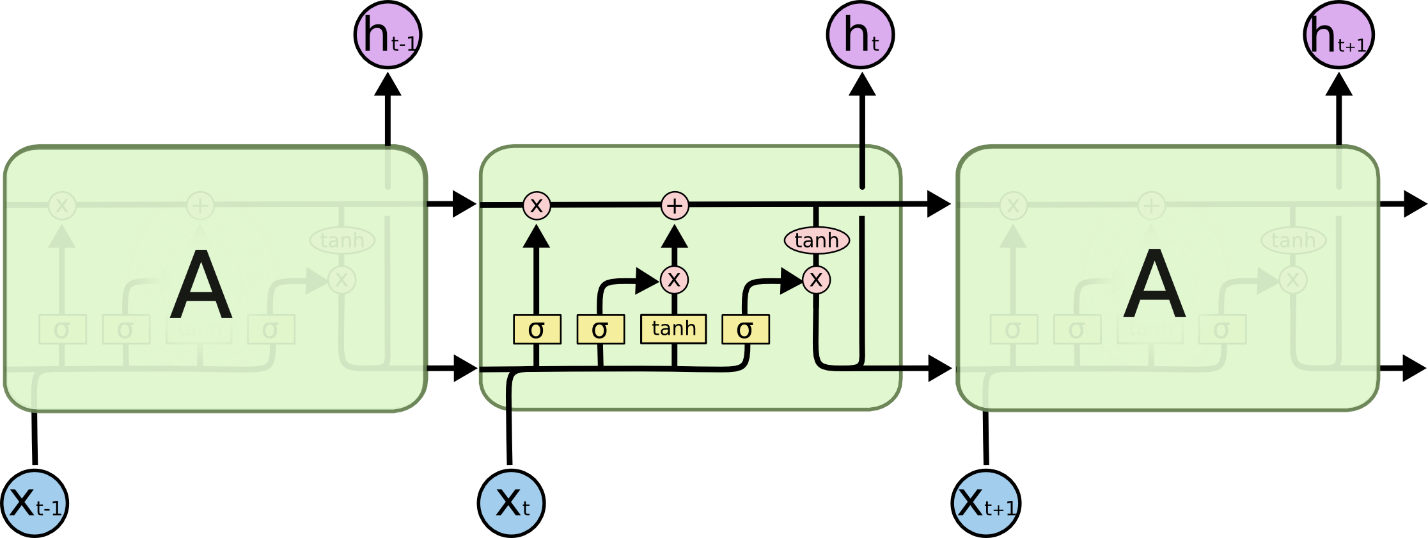
Για την εκπαίδευση αναδρομικών δικτύων χρησιμοποιείται ο αλγόριθμος back propagation through time (BPTT) όπου ουσιαστικά η διόρθωση των παραμέτρων γίνεται στο ξεδιπλωμένο δίκτυο και οι παράγωγοι υπολογίζονται όπως στο παρακάτω διάγραμμα[8].



Συνεπώς αν η ακολουθία εισόδου είναι μεγάλη (δηλαδή το ξεδιπλωμένο δίκτυο είναι πολύ βαθύ) τότε θα εμφανίζονται πάλι προβλήματα Vanishing και Exploding Gradient, δηλαδή το σφάλμα που μεταβιβάζεται προς τα προγενέστερα βήματα θα τείνει στο 0 ή στο ∞. Όπως αναφέραμε και προηγούμενος αυτό δυσκολεύει ή ακόμα καθιστά αδύνατο στο δίκτυο να εκαπιδευτεί[12].

Με τον καιρό έχουν προταθεί διάφορα μοντέλα για την επίλυση των παραπάνω προβλημάτων. Οι διαφορές των αρχιτεκτονικών αυτών βρίσκονται στην εσωτερική οργάνωση και λειτουργία των πράξεων που γίνονται σε κάθε βήμα μιας ακολουθίας (δηλαδή της αρχιτεκτονικής κάθε κελιού). Η πλέον διαδεδομένη αρχιτεκτονική κελίου για αναδρομικά νευρωνικά δίκτυα είναι το Σύστημα Μακράς Βραχυπρόθεσμης Μνήμης – LSTM [13].

Όπως δηλώνει και το όνομα το σύστημα LSTM έχει την δυνατότητα να ‘θυμάται’ μεγάλες ακολουθίες εισόδου, μιας και κατά την εκπαίδευση του δεν υποφέρει από τα προβλήματα Exploding και Vanishing Gradient. Η αρχιτεκτονική ενός κελιού LSTM παρουσιάζεται στο παρακάτω σχήμα:



Η λειτουργία ενός συστήματος μακράς βραχυπρόθεσμης μνήμης στηρίζεται σε 3 πύλες την πύλη διαγραφής (forget gate), την πύλη ανανέωσης (update gate) και την πύλη εξόδου (result gate). Οι εξισώσεις ενός LSTM κελίου μαζί με το μέγεθος των διανυσμάτων του παρουσιάζονται παρακάτω:

|  |  |  |
| --- | --- | --- |
| Όνομα | Εξίσωση | Μέγεθος |
| Είσοδος |  | p + lstm\_size |
| Forget Gate |  | lstm\_size |
| Update Gate |  | lstm\_size |
| Result Gate |  | lstm\_size |
| Input |  | lstm\_size |
| Νέα C |  | lstm\_size |
| Nεα H |  | lstm\_size |

Όπου:

* σ η sigmoid function
* n: μέγεθος κελιού και είναι ανάλογο με την μνήμη που έχει το δίκτυο
* p: το μέγεθος του διανύσματος εισόδου
* lstm\_size: το μέγεθος του κελιού

Ουσιαστικά η τιμή κάθε πύλης υπολογίζεται από ένα διαφορετικό νευρωνικό δίκτυο ενός επιπέδου. Ο προσδιορισμός πύλη πηγάζει από το γεγονός ότι αυτές έχουν τιμές από 0, 1 (λόγω της sigmoid) και πολλαπλασιάζονται ανά στοιχείο (elementwise) και συνεπώς ελέγχουν την ροή των δεδομένων από το ένα βήμα της ακολουθίας στο επόμενο. Η νέα κατάσταση Ct είναι αυτά που θα ξεχάσει το δίκτυο, δηλαδή το αποτέλεσμα της πύλης διαγραφής επί την προηγούμενη Ct-1, συν αυτά που θέλει να ανανεώσει το δίκτυο, δηλαδή το αποτέλεσμα της πύλης ανανέωσης επί την καινούργια είσοδο. Το αποτέλεσμα ht του κελίου θα είναι το γινόμενο ανά στοιχείο της πύλης αποτελέσματος επί την νέα κατάσταση Ct, προσθέτοντας μια μη- γραμμικότητα. Τέλος επειδή μπορεί η προβλεπόμενη έξοδος να μην έχει το ίδιο μέγεθος με το μέγεθος των διανυσμάτων εξόδου συνηθίζεται να προστίθεται ένα Fully Connected Layer στο τέλος του κελίου ώστε να γίνει ο μετασχηματισμός αυτός. Συνεπώς ο συνολικός αριθμός παραμέτρων ενός LSTM κελίου είναι:

Το 4 πηγάζει από το γεγονός ότι εσωτερικά βρίσκονται 4 Fully Connected Layers και το +1 από τον πίνακα βαρών των επιπέδων αυτών.

Έχουν προταθεί και άλλες βελτιστοποιήσεις του παραπάνω μοντέλου οι οποίες χρησιμοποιούν μικρότερο πλήθος παραμέτρων (όπως GRU), αλλά όλες στηρίζονται στην ίδια βασική λειτουργία.

Τέλος να σημειωθεί ότι αυτές οι βασικές μονάδες μπορούν να στοιβαχθούν η μια πάνω στην άλλη (stacked) δημιουργώντας ένα πιο βαθύ νευρωνικό δίκτυο.

**Κεφάλαιο 7 - Πειράματα**

Στα πλαίσια αυτής της διπλωματικής εργασίας έχουν ερευνώνται οι παρακάτω αρχιτεκτονικές για την αυτόματη σύνθεση μελωδιών:

* Αρχιτεκτονική απλού αναδρομικού δικτύου (RNN)
* Αρχιτεκτονική Κωδικοποιητή- Αποκωδικοποιητή (Encoder-Decoder ή Autoencoder)

Τα παραπάνω μοντέλα εκπαιδεύονται με datasets από διαφορετικά μουσικά όργανα:

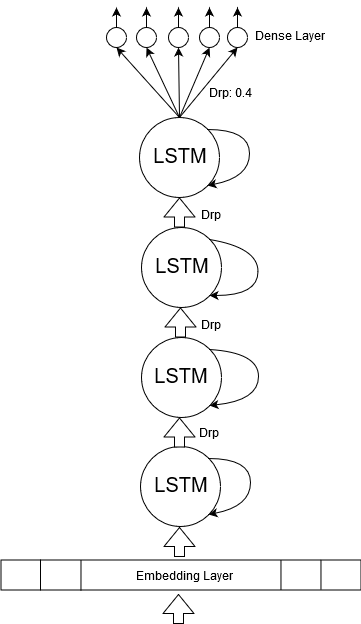
* Πιάνο
* Κιθάρα
* Πιάνο-Κιθάρα
* Επαυξημένο σύνολο δεδομένων

Καθως και με διαφορετικές τιμές υπερπαραμέτρων με τελικό σκοπό την αξιολόγηση των παραπάνω μοντέλων μοντέλων.

**7.1 Αρχιτεκτονική Αναδρομικού Δικτύου**

Σε πρώτη φάση χρησιμοποιήθηκε ένα απλό αναδρομικό δίκτυο Μακράς Βραχυπρόθεσμης Μνήμης.

Συγκεκριμένα η δομή του πρώτου μοντέλου που κατασκευάστηκε παρουσιάζεται στο παρακάτω σχήμα:



Κάθε τιμή μιας ακολουθίας εισόδου περνά από έναν Embedding Matrix όπου παράγεται ένα διάνυσμα το όποιοι είναι και η είσοδος stacked αναδρομικού δικτύου (4επιπέδων). Στην συνέχεια το αποτέλεσμα του LSTM περνά από ένα Fully Connected Layer με συνάρτηση ενεργοποίησης την softmax. Συνεπώς ως έξοδο του δικτύου σε κάθε βήμα έχουμε ένα διάνυσμα μεγέθους όσες και οι διαφορετικές τιμές της εξόδου και τιμών από 0 έως 1 που εκφράζουν την πιθανότητα κάθε μια νότα να είναι η επόμενη. Αξίζει να σημειώθει λοτι ανάμεσα από κάθε επίπεδο του παραπάνω δυκτίου υπάρχει ένα επίπεδο Dropout για να αποφευχθεί το Overfitting ¨όπως εξηγείται και παραπάνω.

Για την εκπαίδευση του μοντέλου δίνεται μια ακολουθία εισόδου (μεγέθους n) και εξετάζεται η έξοδος του δικτύου μόνο για την τελευταία νότα, δηλαδή αν προέβλεψε σωστά την νότα n + 1, ενώ όλες οι ενδιάμεσες έξοδοι του απορρίπτονται. Μαθηματικά η ποσότητα αυτή εκφράζεται από την παρακάτω σχέση:

Δηλαδή το δίκτυο εκπαιδεύεται να υπολογίζει την πιθανότητα της νότας n + 1 με δεδομένο της προηγούμενες n νότες.

Το δίκτυο αυτό εκπαιδεύτηκε με 2 διαφορετικές τιμές του μεγέθους του κελιού (lstm\_size), 256 και 512. Αξίζει να σημειωθεί ότι δοκιμάστηκαν και άλλες τιμές αλλά τα αποτελέσματα των παραπάνω τιμών είναι τα πλέον αντιπροσωπευτικά μιας και για τιμές μικρότερες του 256 το δίκτυο δεν μπορούσε να μάθει κάτι αξιόλογο ενώ για τιμές μεγαλύτερες του 512 η εκπαίδευση αργούσε πολύ και τα αποτελέσματα δεν διέφεραν αισθητά (σφάλμα του δικτύου διέφερε ελάχιστα).

Επίσης δοκιμάστηκαν και διαφορετικές τιμές της παραμέτρους n. Αξίζει να σημειωθεί ότι στην παραπάνω αρχιτεκτονική το μέγεθος της ακολουθίας εισόδου δεν αλλάζει τον αριθμό των παραμέτρων ή την δομή του μοντέλου. Το μόνο που αλλάζει είναι ο χρόνος εκπαίδευσης μιας και το ξεδιπλωμένο δίκτυο είναι πολύ πιο βαθύ με αποτέλεσμα να χρειάζεται περισσότερο χρόνο ο back propagation.

**7.1.1 Μέγεθος κελίου 256**

Στο μοντέλο οι υπεπαράμετροι του δικτύου αυτού εμφανίζονται οι παρακάτω:

* Embedding Size = 50
* Lstm\_size = 256
* Drp = 0.4
* Το Fully Connected Layer έχει μόνο ένα επίπεδο και η έξοδος του είναι όσες και οι διαφορετικές πιθανές τιμές εξόδου.

Το παραπάνω μοντέλο υπολογίζει ποια είναι η πιο πιθανή επόμενη νότα. Στην πραγματικότητα όμως για την σύνθεση μια μελωδίας αυτό που πρέπει να υπολογιστεί είναι η πιο πιθανή ακολουθία νοτών.

[1] <https://arxiv.org/pdf/1506.01186.pdf>

[2]<https://arxiv.org/pdf/1412.6980v8.pdf>

[3] <http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf>

[4] <http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf>

[5] <http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf>

[6] <https://keras.io/initializers/#glorot_uniform>

[7]<https://medium.com/usf-msds/deep-learning-best-practices-1-weight-initialization-14e5c0295b94>

[8] <https://danijar.com/tips-for-training-recurrent-neural-networks/>

[9] <https://www.researchgate.net/publication/13853244_Long_Short-term_Memory>

[10]<http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/>

[11]<http://www.wildml.com/2015/10/recurrent-neural-networks-tutorial-part-3-backpropagation-through-time-and-vanishing-gradients/>

[12]<http://proceedings.mlr.press/v28/pascanu13.pdf>

[13] <http://www.bioinf.jku.at/publications/older/2604.pdf>

[14] <https://arxiv.org/pdf/1604.08723.pdf> San to diko mou

[15] <https://arxiv.org/abs/1612.01010> Bach Chorales