Πτυχιακή Εργασία

Ανίχνευση της νόσου Parkinson αναλύοντας το σήμα της ομιλίας

Γκάγκος Πολυδεύκης
Επιβλέπων: κ. Κωνσταντίνος Κοτρόπουλος

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΑΚΑΔΗΜΑΪΚΟ ΕΤΟΣ 2016-2017

Ευχαριστίες

Η εκπόνηση της πτυχιακή εργασίας έγινε με την παρουσία, υποστήριξη και συμπαράσταση ενός συνόλου ανθρώπων τους οποίους θα ήθελα να ευχαριστήσω. Πρώτα απο όλους ευχαριστώ τον επιβλέποντα Δρ. Κωνσταντίνο Κοτρόπουλο για την καθοδήγηση, την υπομονή και την υποστήριξη που μου παρείχε καθόλη τη διάρκεια εκπόνησης της παρούσας πτυχιακής εργασίας, καθώς και όλους τους καθηγητές της κατεύθυνσης ψηφιακών μέσων για τη γνώση που μου μετέφεραν κατά τη διάρκεια των σπουδών μου. Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου για την ανιδιοτελή και ανυπολόγιστη υποστήριξη που μου παρείχε όλα αυτά τα χρόνια.

Περίληψη

Η παρούσα πτυχιακή εργασία ασχολείται με την μελέτη και υλοποίηση συνελικτικών/αναδρομικών νευρωνικών δικτύων και τη χρήση τους στην ταξινόμηση και την επεξεργασία ήχου. Σκοπός της εργασίας είναι με την παροχή της απαραίτητης ηχητικής πληροφορίας η μελέτη και εξαγωγή συμπερασμάτων μέσω των νευρωνικών δικτύων τόσο σε θεωρητικό όσο και σε πρακτικό επίπεδο, ξεκινώντας με το πρώτο κεφάλαιο όπου σχετίζεται το Parkinson και η ομιλία, συνεχίζοντας με το δεύτερο κεφάλαιο όπου γίνεται μια σύντομη ανασκόπηση της θεωρίας επεξεργασίας ήχου. Στο τρίτο κεφάλαιο παρουσιάζεται η θεωρία της μηχανικής μάθησης και αναλύονται οι κατηγορίες στις οποίες αυτή χωρίζεται, οι τεχνικές παλινδρόμησης καθώς και μια σύντομη αναφορά στην ταξινόμηση ήχων. Στο τέταρτο κεφάλαιο της εργασίας, παρουσιάζεται η θεωρία που διέπει τα συνελικτικά νευρωνικά δίκτυα. Στο πέμπτο κεφάλαιο παρουσιάζονται τα αναδρομικά δίκτυα και πιο συγκεκριμένα τα δίκτυα μακράς βραχυχρόνιας μνήμης (LSTM). Στο έκτο κεφάλαιο περιγράφονται αναλυτικά η βάση δεδομένων και τα μέτρα αξιολόγησης των μοντέλων. Στο έβδομο κεφάλαιο παρουσιάζεται η υλοποίηση της, τα αποτελέσματα και συμπεράσματα εκτέλεσης αλγορίθμων επιβλεπόμενης μάθησης με χρήση βαθιάς εκμάθησης για την βάση δεδομένων mPower.

Abstract

The present dissertation engages in the study and development of convolutional/recurrent neural networks and their use in the classification and editing for sound. The aim of this study is, using the necessary sound information, the production of conclusions aided by neural networks both in a theoretical and a practical level. The first chapter concerns itself with Parkinson's disease and speech, while the second chapter presents a quick review of the theory of sound editing. The third chapter presents some key concepts on the theory of machine learning and analyses the categories that comprise it, back-propagation techniques, and sound classification. The fourth chapter deals with the theory regarding convolusional neural networks. The fifth chapter presents recurring networks and more specifically the long short-term memory networks (LSTM). The sixth chapter details the database as well as the model evaluation metrics. The seventh chapter presents the implementation results and conclusions obtained from the execution of supervised learning algorithms using both shallow and deep learning techniques for the database.

Κατάλογος Σχημάτων

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Η νόσος Parkinson

Η νόσος Parkinson [1], γνωστή και ως ιδιοπαθής ή πρωτοπαθής παρκινσονισμός ή τρομώδης παράλυση, είναι μια εκφυλιστική διαταραχή του κεντρικού νευρικού συστήματος. Χαρακτηρίζεται κατά βάση από κινητικά φαινόμενα, και συγκεκριμένα το χαρακτηριστικό τρέμουλο, την έλλειψη αυθόρμητης κινητικότητας και τη βραδύτητα όταν γίνονται εκούσιες κινήσεις, τη δυσκαμψία, δηλαδή την αντίσταση που αισθάνεται ο γιατρός όταν κινεί παθητικά τα μέλη του αρρώστου, κάτι που ο ίδιος ο άρρωστος αισθάνεται ως «μάγκωμα», και τις διαταραχές στάσης και βάδισης. Είναι χαρακτηριστική η κυρτή στάση του κορμιού και η τάση των ασθενών να περπατούν με μικρά βήματα. Η ιατρική προσέγγιση της διάγνωσης της νόσου βασίζεται σε αυτήν την κλινική σημειολογία.

1.2 Parkinson και ομιλία

Παρά το γεγονός ότι σε ασθενείς με Parkinson συχνά χαρακτηριστικά στα ηχητικά σήματα ομιλίας είναι η μειωμένη ένταση του λόγου (hypophonia), η μειωμένη προσοδιακή χρειά του τόνου της φωνής (hypoprosodia), το αυξημένο άγχος, η βραχνή και ξέπνοη ποιότητα της φωνής (dysphonia) και η λανθασμένη άρθρωση. Διάφορα είδη διαταραχής της ομιλίας δίνουν κίνητρο στην ολιστική επεξεργασία πληροφοριών σχετικά με την παθολογική ομιλία κατά συνέπεια ένα εύρος ακουστικών παραγόντων ομιλίας. Τις τελευταίες δεκαετίες έχει υπάρξει ένα σποραδικό ενδιαφέρον για τον χαρακτηρισμό αυτών των συμπτωμάτων καθώς και την αξιολόγηση της σοβαρότητας των παραγλωσσολογικών σημάτων ομιλίας σε ασθενείς με Parkinson. Η αξιολόγηση της σοβαρότητας των παραγλωσσολογικών γνωρισμάτων της ομιλίας είναι απαραίτητη για τη συνεχή θεραπεία και παρακολούθηση των ασθενών. Παρά την τεράστια ζήτηση για αντικειμενική, ακριβή και αξιόπιστη αξιολόγηση στην κλινική πρακτική, η τελευταία μέθοδος αξιολόγησης εξακολουθεί να βασίζεται σε υποκειμενικές εκτιμήσεις των εμπειρογνωμόνων, οι οποίες είναι δαπανηρές και χρονοβόρες. Για το λόγο αυτόν, καταβλήθηκαν προσπάθειες για την ανάπτυξη ενός αυτόματου συστήματος αξιολόγησης με τη χρήση της ομιλίας των ασθενών. Η χρήση της ομιλίας έχει πολλά πλεονεκτήματα συγκριτικά με άλλες μεθοδολογίες, καθώς (1) οι περισσότεροι ασθενείς με Parkinson υποφέρουν από διαταραχές ομιλίας και (2) η απόκτηση

των δεδομένων είναι σχετικά εύκολη και βολική, γιατί μπορεί να γίνει απομακρυσμένα και σε τακτά χρονικά διαστήματα.

1.3 Στόχοι της διπλωματικής

Ο στόχος της διπλωματικής είναι να αναπτύξει έναν αλγόριθμο ταξινόμησης ηχητικών καταγραφών από άτομα που πάσχουν από την νόσο Parkinson (PD) και άτομα που είναι υγιείς (HC). Με την επίτευξη το στόχου αυτού θα μπορούσε να δωθεί μια πρώτη αντικειμενικά έγκυρη εκτίμηση για την διάγνωση της νόσου σε έναν ανθρώπου.

Στην παρούσα διπλωματική, αναπτύχθηκε μία εφαρμογή ταξινόμησης ηχητικών καταγραφών, που χρησιμοποιεί ως ταξινομητές βαθιά νευρωνικά δίκτυα. Συγκεκριμένα, υλοποιήθηκαν δύο διαφορετικά πειράματα ταξινόμησης: το ένα με τη χρήση συνελικτικών νευρωνικών δικτύων, και το άλλο με τη χρήση αναδρομικών νευρωνικών δικτύων. Οι ηχητικές καταγραφές που χρησιμοποιήθηκαν, πάρθηκαν απο την βάση δεδομένων mPower, για την οποία θα γίνει αναφορά ακολούθως. Τα δεδομένα και οι αρχιτεκτονικές των δικτύων που υλοποιήθηκαν θα αναλυθούν σε παρακάτω κεφάλαια με παραδείγματα.

Τα αποτελέσματα της εφαρμογής δείχνουν την ουσιαστική ικανότητα και την επαυξημένη επίδοση των νευρωνικών δικτύων σαν ταξινομητές. Και στα δύο πειράματα για τα αποτελέσματα χρησιμοποιήθηκε μεγάλη υπολογιστική δύναμη από κατάλληλους για τη δουλειά υπολογιστές. Οι εκτελέσεις συγκρίθηκαν με πραγματικά αποτελέσματα από το δημογραφικό ερωτηματολόγιο της βάσης mPower.

1.4 Δομή της διπλωματικής

Ακολουθεί η οργάνωση της διπλωματικής σε κεφάλαια:

- Στο δεύτερο κεφάλαιο δίνεται μια μικρή εισαγωγή και κάποια γενικά χαρακτηριστικά πάνω στα επεξεργασία ηχητικών σημάτων και στην εξαγωγή χαρακτηριστικών.
- Στο τρίτο κεφάλαιο παρουσιάζεται η θεωρία της μηχανικής μάθησης και αναλύονται οι κατηγορίες στις οποίες αυτή χωρίζεται, οι τεχνικές παλινδρόμησης καθώς και μια σύντομη αναφορά στην ταξινόμηση ήχων.
- Στο τέταρτο κεφάλαιο της εργασίας, παρουσιάζεται η θεωρία που διέπει τα συνελικτικά νευρωνικα δίκτυα, οι μονάδες που τα αποτελούν και η συνδεσμολογία τους.
- Στο πέμπτο κεφάλαιο παρουσιάζονται τα αναδρομικά νευρωνικά δίκτυα και πιο συγκεκριμένα τα δίκτυα μακράς βραχυχρόνιας μνήμης Long Short-Term Memory (LSTM).
- Στο έκτο κεφάλαιο περιγράφονται αναλυτικά η βάση δεδομένων καθώς και τα μέτρα αξιολόγησης των μοντέλων.

- Στο έβδομο κεφάλαιο παρουσιάζεται η υλοποίηση, τα αποτελέσματα και τα συμπεράσματα εκτέλεσης αλγορίθμων επιβλεπόμενης μάθησης με χρήση ρηχούς αλλά και βαθιάς εκμάθησης για την βάση δεδομένων mPower.

Κεφάλαιο 2

Εισαγωγή στην επεξεργασία ήχου

2.1 Βασικές Έννοιες από την Ψηφιακή Επεξεργασία Σήματος

2.1.1 Κανονικοποιημένος Χρόνος και Συχνότητα

Τα σήματα διακριτού χρόνου συμβολίζονται με $s(n)$ και ορίζονται ως εξής :

$$s(n) = s_a(nT) = s_a(t)|_{t=nT} \quad n =, ..., -1, 0, 1, ... \tag{2.1}$$

Όπου ο ακέραιος n είναι ο δείκτης των ληφθέντων δειγμάτων, το $s(n)$ το δείγμα της αναλογικής κυματομορφής που δειγματοληπτείται με περίοδο T. Η γνώση της περιόδου T είναι απαραίτητη για την απόκτηση της πλήρους χρονικής πληροφορίας. Η θεώρηση του κανονικοποιημένου χρόνου μας επιτάσσει να δεχτούμε ότι ο πραγματικός χρόνος κλιμακώνεται κατά τον παράγοντα T πριν τη συλλογή δειγμάτων:

$$\acute{t} = \frac{t}{T}. \tag{2.2}$$

Η κανονικοποίηση συνεπάγεται ότι:

1. η περίοδος δειγματοληψιας είναι μοναδιαία
2. η συχνότητα δειγματοληψίας είναι μοναδιαία
3. τα παραπάνω μεγέθη είναι αδιάστατα.

Κατά συνέπεια η κυκλική συχνότητα δειγματοληψίας είναι 2π (αδιάστατη) και η ανώτερη συχνότητα τους σήματος είναι πάντοτε 0.5 norm-Hz ή πnorm-rps. Η μετατροπή από τις πραγματικές αναλογικές συχνότητες F σε Hz και Ω σε rad/sec στις κανονικοποιημένες αντίστοιχές τους δίνεται από τις παρακάτω σχέσεις:

$$f = FT \tag{2.3}$$

$$\omega = \Omega T. \tag{2.4}$$

Παράδειγμα:Εστω ένα ημίτονο συνεχούς χρόνου κυκλικής συχνότητας Ω :

$$x_a(t) = A\sin(\Omega t + \varphi) = A\sin(\Omega T\frac{t}{T} + \varphi) \Leftrightarrow$$

$$x_a(\acute{t}) = A\sin(\omega \acute{t} + \varphi). \tag{2.5}$$

Παίρνουμε τα ίδια δείγματα αν υποβάλλουμε σε δειγματοληψία το σήμα $x_a(t)$ σε χρονικές στιγμές t=nT και το σήμα $x_a(\acute{t})$ σε χρονικές στιγμές $\acute{t} = n$.

2.1.2 Βασικές ακολουθίες

Μοναδιαίο δείγμα

$$\delta(n) = \begin{cases} 1, & n = 0 \\ 0, & \text{αλλού} \end{cases} \tag{2.6}$$

Μοναδιαίο βήμα

$$u(n) = \begin{cases} 1, & n \geq 0 \\ 0, & \text{αλλού} \end{cases} \tag{2.7}$$

2.1.3 Σήματα ενέργειας και ισχύος

Ενέργεια σήματος διακριτού χρόνου (ΔΧ)

$$E_x = \sum_{n=-\infty}^{n=+\infty} |x(n)|^2 \tag{2.8}$$

Ένα σήμα είναι σήμα **ενέργειας** αν και μόνο αν το E είναι αριθμός πεπερασμένος και θετικός $(0 < E_x < +\infty)$
Η **Ισχύς** ενός σήματος ΔΧ ορίζεται ως

$$P_x = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} |x(n)|^2 \tag{2.9}$$

Ένα σήμα ισχύος έχει πεπερασμένη μη μηδενική ισχύ αν και μόνο αν $0 < P_x < \infty$.
Ένα σήμα δεν μπορεί να είναι ταυτόχρονα και σήμα ισχύος και σήμα ενέργειας, επειδή αν $E_x < \infty$, τότε $P_x = 0$. Επίσης , ένα σήμα μπορεί να μην είναι ούτε ενέργειας ούτε ισχύος αν $P_x = \infty$ ή $E_x = 0$. **Παραδείγματα σημάτων ενέργειας.**

1. Μεταβατικά σήματα (συνήθως μιγαδικά εκθετικά που αποσβένονται με το χρόνο)

$$x_1(n) = \alpha^n u(n), \qquad |\alpha| < 1 \tag{2.10}$$

$$x_2(n) = \alpha^{|n|}\cos(n\omega_0 + \psi), \qquad |\alpha| < 1. \tag{2.11}$$

2. Ακολουθίες χρονοπερατές

$$x_3(n) = e^{\beta n}[u(n+3) - u(n-246)], \qquad |\beta| < \infty. \tag{2.12}$$

Παραδείγματα σημάτων ισχύος.

1. Σταθερά σήματα

$$x_4(n) = \alpha, \qquad -\infty < \alpha < \infty \tag{2.13}$$

2. Περιοδικά σήματα

$$x_5(n) = a \sin(n\omega_0 + \psi), \qquad -\infty < \alpha < \infty \tag{2.14}$$

$$x_6(n) = [x_3(n)]_{mod\,512} = \sum_{i=-\infty}^{+\infty} x_3(n + i512). \tag{2.15}$$

2.1.4 Συνέλιξη

Η συνέλιξη αποτελεί μια πράξη πολύ σημαντική γιατί σχετίζεται με την ανάλυση συστημάτων, αλλά και με το γεγονός ότι η συνέλιξη μετατρέπεται σε γινόμενο όταν αλλάζουμε χώρους(από τον χρόνο στην συχνότητα και αντίστροφα) [2]. Ορίζεται ως πράξη ανάμεσα σε δύο σήματα:

$$y(n)=x(n) * h(n) \tag{2.16}$$

Στην εξίσωση (2.16) περιγράφεται η έξοδος ένος ΓΧΑ συστήματος ως η συνέλιξη της εισόδου $x(n)$ με την κρουστική απόκριση $h(n)$.

1. **Κυκλική Συνέλιξη**

   Η κυκλική συνέλιξη ορίζεται βάσει της κυκλικής μετατόπισης. Για να υπολογίσουμε την κυκλική συνέλιξη δύο ακολουθιών, θα πρέπει να έχουν τον ίδιο αριθμό δειγμάτων. Σε αντίθετη περίπτωση θα πρέπει να εφαρμόσουμε τη μέθοδο zero-padding, δηλαδή να προσθέσουμε μηδενικά στο τέλος της ακολουθίας με τα λιγότερα δείγματα.

2. **Γραμμική Συνέλιξη**

   Στηριζόμενοι στη κυκλική συνέλιξη, μπορούμε να υπολογίσουμε και τη γραμμική, αρκεί να παραγεμίσουμε με ηδενικά και τις δύο ακολουθίες ώστε να έχουν μήκος ίσο με εκείνο της γραμμικής συνέλιξής τους.
   Παράδειγμα: Έχουμε τις ακολουθίες $x(n) = [1\ 2\ 2\ 1]$ και $y(n) = [1\ -1\ -1\ 1]$ και εκτελούμε κυκλική συνέλιξη με N=4 + 4 - 1 =7, δηλαδή εφαρμόζουμε τη μέθοδο zero-padding. Το αποτέλεσμα θα είναι μια ακολουθία $w(n)$ =[1 1 -1 -2 -1 1 1].

2.1.5 Μετασχηματισμοί

Discrete Time Fourier Transform (DTFT) - Παρατηρούμε ότι $\acute{t} = n$, οπότε ο μετασχηματισμός Fourier διακριτού χρόνου ορίζεται απο το ζεύγος εξισώσεων:

$$X(\omega) = \sum_{n=-\infty}^{+\infty} x(n)\, e^{-j\omega n} \tag{2.17}$$

$$x(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(\omega)\, e^{j\omega n}\, d\omega \tag{2.18}$$

Όταν η ακολουθία $x(n)$ αναπαριστά δείγματα του αναλογικού σήματος $x_a(t)$, τότε ο DTFT είναι μια περιοδική επανάληψη με περίοδο 2π και ενδεχομένως επικαλυπτόμενη του μετασχηματισμού Fourrier συνεχούς χρόνου, $X_a(\omega)$, δηλαδή:

$$X(\omega) = \sum_{i=-\infty}^{+\infty} X_\alpha(\omega - 2\pi i). \tag{2.19}$$

Η εξίσωση 2.19 στην γενική περίπτωση που δεν χρησιμοποιείται κανονικοποιημένα χρόνος για ένα σήμα $X_a(t)$ μπορεί να γραφτεί ισοδύναμα ως:

$$X(\Omega) = \frac{1}{T} \sum_{i=-\infty}^{+\infty} X_\alpha(\Omega - \frac{2\pi}{T} i) \tag{2.20}$$

Αξίζει να τονιστεί ότι ο μετασχηματισμός DTFT υπάρχει, εφόσον η ακολουθία $x(n)$ είναι απολύτως αθροίσιμη, δηλαδή (ικανή συνθήκη):

$$\sum_{n=-\infty}^{+\infty} |x(n)| < \infty. \tag{2.21}$$

Όπως προαναφέρθηκε, μία απολύτως αθροίσιμη ακολουθία είναι **σήμα ενέργειας** αφού:

$$E_x = \sum_{n=-\infty}^{+\infty} |x(n)|^2 \leq \Big[\sum_{n=-\infty}^{+\infty} |x(n)| \Big]^2. \tag{2.22}$$

Αλλά ένα σήμα ενέργειας **δεν είναι πάντα απολύτως αθροίσιμο**. Τα σήματα ενέργειας που δεν είναι απολύτως αθροίσιμα, έχουν μετασχηματισμό Fourier διακριτού χρόνου για τον οποίο όμως η σειρά

$$\sum_{n=-\infty}^{+\infty} x(n) e^{-j\omega n}. \tag{2.23}$$

συγκλίνει υπό πιο ασθενή έννοια ως προς το μέσο τετράγωνο. Σε αυτό το συμπέρασμα καταλήγουμε αν θεωρήσουμε την (1.16) ως επέκταση σε σειρά Fourier της περιοδικής συνάρτησης $X(w)$ με συντελεστές Fourier $x(n)$:

$$X(\omega) = \sum_{n=-\infty}^{+\infty} x(n) e^{-j\omega n}. \tag{2.24}$$

Αν η ενέργεια σε μία μόνη περίοδο της συνάρτησης είναι πεπερασμένη, τότε η σειρά συγκλίνει ως προς το μέσο τετράγωνο. Χρησιμοποιώντας τη σχέση του Parseval προκύπτει:

$$\int_{-\pi}^{\pi} |X(\omega)|^2\, d\omega = 2\pi \sum_{n=-\infty}^{+\infty} |x(n)|^2 = 2\pi\, E_x < \infty \tag{2.25}$$

επομένως ο DTFT συγκλίνει ως προς το μέσο τετράγωνο. Πρακτικά, αυτό σημαίνει ότι το άθροισμα στον ορισμό του DTFT θα συγκλίνει στο X(ω) σε όλα τα σημεία συνέχειας της $x(t)$ και στα σημεία ασυνέχειας θα συγκλίνει στο μέσο όρο των τιμών εκατέρωθεν της ασυνέχειας. Ο DTFT είναι πολύ χρήσιμος σε θεωρητική βάση, αλλά δεν είναι υπολογίσιμος σε έναν ψηφιακό υπολογιστή λόγω του συνεχούς ορίσματος. Η λύση του προβλήματος καλείται Διακριτός Μετασχηματισμός Fourier (Discrete Fourier Transform, DFT), ο οποίος ορίζεται σε ακολουθίες πεπερασμένου μήκους:

$$x(n), \qquad n = 0, 1, 2, ..., N-1. \tag{2.26}$$

Ο DFT ορίζεται από τη σχέση:

$$X(k) = \begin{cases} \sum_{n=0}^{N-1} x(n)\ e^{-j(\frac{2\pi}{N})kn}, & k = 0, 1, .., N-1 \\ 0, & \text{αλλού.} \end{cases} \tag{2.27}$$

Και ο αντίστροφος DFT (inverse DFT) ορίζεται ως εξής :

$$x(n) = \begin{cases} \frac{1}{N} \sum_{k=0}^{N-1} X(k) e^{j(\frac{2\pi}{N})kn}, & n = 0, 1, .., N-1 \\ 0, & \text{αλλού} \end{cases} \tag{2.28}$$

Αυτό που θα πρέπει να μείνει είναι ότι ο DFT αναπαριστά ακριβώς τα δείγματα του DTFT μιας πεπερασμένης ακολουθίας $x(n)$ σε N ισαπέχουσες συχνότητες $\omega_k = \frac{2\pi}{N}k$ για $k \in [0, N-1]$.

Discrete Fourier Series (DFS) - Η διακριτή σειρά Fourier (DFS) συνδέεται στενά με τον DFT υπολογιστικώς, αλλά έχει εντελώς διαφορετική φιλοσοφία. Η DFS αναπαριστά μια περιοδική ακολουθία με περίοδο N χρησιμοποιώντας το σύνολο διανυσμάτων βάσης $e^{j\frac{2\pi}{N}kn}$, $k = 0, 1, 2, ..., N-1$ που αναπαριστούν τις N αρμονικές συχνότητες που μπορεί να ενυπάρχουν στο σήμα. Για μία περιοδική ακολουθία$y(n)$, η επέκταση είναι

$$y(n) = \sum_{k=0}^{N-1} C(k)\ e^{j(\frac{2\pi}{N})kn} \tag{2.29}$$

οπου οι συντελεστές της επέκτασης δίνονται από την :

$$C(k) = \frac{1}{N} \sum_{n=0}^{N-1} y(n) e^{-jk(\frac{2\pi}{N})n} \tag{2.30}$$

που είναι ο **DFT** του περιοδικού σήματος. Ο DTFT συντίθεται από τις αρμονικές που υπολογίζονται από τις συνιστώσες του DFS δίνεται από την σχέση:

$$Y(\omega) = 2\pi \sum_{k=-\infty}^{+\infty} C(k)\delta_\alpha(\omega - k\frac{2\pi}{N}). \tag{2.31}$$

DTFT περιοδικών σημάτων Αν και ο DTFT δεν υπάρχει για ένα περιοδικό σήμα, μπορούμε να τον υπολογίσουμε με τη βοήθεια του ορίου

$$\bar{Y}(\omega) = \lim_{N\to\infty} \frac{1}{2N+1} \sum_{n=-N}^{N} y(n) e^{-j\omega n} \tag{2.32}$$

το οποίο με περαιτέρω σκέψη προκύπτει ότι είναι ισοδύναμο με τον υπολογισμό του αθροίσματος σε διάστημα μιας περιόδου.

Fast Fourier Transform (FFT) - Τόσο ο DFT όσο και αντίστροφος DFT μπορούν να θεωρηθούν απλά ως υπολογιστικές μέθοδοι οι οποίες δέχονται μιγαδικούς αριθμούς σε ένα πεδίο και δημιουργούν μιγαδικούς αριθμούς σε ένα άλλο. Η υπολογιστική μέθοδος του FFT ρίχνει την πολυπλοκότητα ένος τέτοιου αλγορίθμου σε $O(N \log N)$ αντί $O(N^2)$ που είχε ο DFT. Είναι ουσιαστικά ένα σύνολο αλγορίθμων που χρησιμοποιούνται για το γρήγορο υπολογισμό του DFT.

2.2 Διαισθητικά Χαρακτηριστικά του Ήχου

Τα χαρακτηριστικά του ήχου και με τα οποία μπορούμε να περιγράψουμε τα ακουστικά σήματα, χωρίζονται σε δύο κατηγορίες, τα φυσικά χαρακτηριστικά, τα οποία είναι εκείνα τα οποία περιγράφουν με μαθηματικό τρόπο τα ακουστικά σήματα, όπως είναι η συχνότητα, τα φασματικά χαρακτηριστικά και το πλάτος, καθώς και τα διαισθητικά χαρακτηριστικά, τα οποία είναι βασισμένα στο πώς αντιλαμβάνεται τους ήχους το ανθρώπινο αυτί και είναι πολύ δύσκολο να μετρηθούν με συμβατικές μεθόδους ή όργανα, καθότι βασίζονται κυρίως στην ανθρώπινη αντίληψη του ήχου. Για τη μέτρηση των διαισθητικών χαρακτηριστικών, συνήθως χρησιμοποιούνται τεστ από ομάδες ακροατών, με αντίστοιχες γνώσεις, που προσδιορίζουν κατά προσέγγιση αυτά τα χαρακτηριστικά.

Ακουστότητα: Η ακουστότητα, ή αλλιώς ένταση είναι ένα χαρακτηριστικό που ορίζει πόσο δυνατός είναι ένας ήχος. Η ακουστότητα είναι ένα υποκειμενικό μέγεθος και για την μέτρηση του απαιτείται να ορίσουμε ένα σύστημα αναφοράς , το οποίο ορίστηκε από τους Flecher and Munson το 1933, και είναι η συχνότητα 1000 Hz. Παράλληλα, απέδειξαν την διαφοροποίηση μεταξύ συχνότητας και έντασης, δηλαδή, ότι εάν δύο ήχοι έχουν την ίδια ένταση αλλά έχουν διαφορετική συχνότητα, το ανθρώπινο αυτί θα ξεχωρίσει αυτόν με την υψηλότερη συχνότητα. Ως μονάδα μέτρησης της στάθμης της ακουστότητας (loudness level - LL) ορίσθηκε από τον Barkhausen το 1926, το Phon.

Τονικό ύψος: Το τονικό ύψος είναι ένα ακόμη διαισθητικό χαρακτηριστικό των ηχητικών σημάτων και είναι αυτό που διαχωρίζει τους ήχους σε χαμηλούς και υψηλούς. Το τονικό ύψος εξαρτάται, κατά κύριο λόγο, από την συχνότητα (f), αλλά και από την στάθμη της ηχητικής πίεσης, και έχει ως μονάδα μέτρησης το Mel. Η σχέση μεταξύ της συχνότητας f(Hz) και της κλίμακας Mel καθορίστηκε από το πείραμα που διεξήγαν οι Stevens, Volkmann and Newman. Στην συνέχεια παρουσιάζεται η λογαριθμική αντιστοιχία μεταξύ της συχνότητας Hz και της διαισθανόμενης κλίμακας Mel, όπως προέκυψε από το πείραμα:

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{10}) \tag{2.33}$$

Χροιά: Η χροιά είναι ένα χαρακτηριστικό που κάνει δύο ήχους που έχουν την ίδια ένταση (intensity) και την ίδια ακουστότητα (loudness) να διαφέρουν. Η διαφορετική κατανομή της

ηχητικής ενέργειας είναι αυτό που προκαλεί την διαφορετικότητα στην χροιά. Για να περιγραφεί η χροιά χρησιμοποιούνται φασματικά διαστήματα μικρής διάρκειας, περίπου 10s - 40s (D. Howard, 2009).

2.3 Προεπεξεργασία

2.3.1 Παραμετροποίηση Σήματος Ομιλίας

Τα ψηφιοποιημένα δεδομένα έχουν υψηλό πλεονασμό σε πληροφορία. Η **συμπίεση** του όγκου των δεδομένων εξαρτάται από την εξαγωγή κατάλληλων **παραμέτρων**, που περιέχουν την **αναγκαία πληροφορία**, για τη **συγκεκριμένη χρήση**. Η απόρριψη προβληματικών ή χωρίς πληροφορία χαρακτηριστικών είναι ένα μεγάλο βοήθημα για την μείωση του όγκου των δεδομένων. Τέτοιες περιπτώσεις μπορεί να είναι η έλλειψη ομιλίας (σιγή), η υψηλή στάθμη θορύβου, ο κορεσμός του ψηφιοποιητή καθώς και η χαμηλή στάθμη έντασης ομιλίας. Απώτερος στόχος της βαθμίδας παραμετροποίησης είναι ο προσδιορισμός προτύπων για ευκολότερη ανάλυση όπως παραμετρικά διανύσματα, τα οποία είναι καλώς διαχωρίσιμα στις κατηγορίες τα φωνήματα [3]. Οι παράμετροι πρέπει να έχουν τα χαρακτηριστικά:

- Να υπολογίζονται εύκολα (Υπολογιστικός χρόνος).
- Να εμφανίζονται συχνά και φυσικά στην ομιλία.
- Να μη μεταβάλλονται στον χρόνο.
- Να είναι εύρωστα δηλαδή να μην υπάρχει σε αυτό επίδραση περιβάλλοντος.

2.3.2 Δειγματοληψία (Sampling)

Ανά χρονικό διάστημα T, παίρνουμε ένα δείγμα. Χρειαζόμαστε τουλάχιστον δύο δείγματα ανά περίοδο σύμφωνα με το θεώρημα του Nyquist. Άρα η συχνότητα δειγματοληψίας εκλέγεται τουλάχιστον διπλάσια της μέγιστης συχνότητας του σήματος. Η ανθρώπινη ομιλία όπως καταγράφεται από μικρόφωνα ευρείας ζώνης (wideband) ειναι κυμαίνεται περίπου στα 8.000 Hz. Σαν αποτέλεσμα χρειαζόμαστε συχνότητα δειγματοληψίας τουλάχιστον $f_s = 16.000$ Hz.

2.3.3 Κβάντιση (Quantization)

Ένα συνεχές σήμα, όπως η φωνή, έχει συνεχές πεδίο τιμών πλάτους και συνεπώς τα δείγματά του έχουν συνεχές πεδίο τιμών πλάτους. Με άλλα λόγια μέσα στο πεπερασμένο πεδίο τιμών του σήματος βρίσκουμε έναν άπειρο αριθμό σταθμών πλάτους. Στην πραγματικότητα όμως δεν είναι απαραίτητο να αποθηκεύουμε τα ακριβή πλάτη των δειγμάτων. Το αυτί, σαν τελικός δέκτης, μπορεί να ανιχνεύσει πεπερασμένες διαφορές έντασης. Αυτό σημαίνει ότι το αρχικό συνεχές σήμα μπορεί να προσεγγιστεί από ένα σήμα το οποίο κατασκευάζεται από διακριτά πλάτη, επιλεγμένα από ένα διαθέσιμο σύνολο με βάση την ελαχιστοποίηση του σφάλματος. Προφανώς εάν καθορίσουμε διακριτές στάθμες πλάτους με αρκετά μικρή απόσταση μεταξύ τους, μπορούμε να κάνουμε το προσεγγιζόμενο σήμα να μη ξεχωρίζει πρακτικά από το αρχικό συνεχές σήμα.

Η αντιστοίχιση των αναλογικών τιμών σε πεπερασμένες ψηφιακές τιμές ονομάζεται κβάντιση.

2.3.4 Βραχυχρόνια ανάλυση

Η αρχή της βραχυχρόνιας ανάλυσης στηρίζεται στη διαπίστωση ότι τα χαρακτηριστικά της ανθρώπινης ομιλίας μεταβάλλονται σχετικά αργά στο χρόνο. Ο ρυθμός μεταβολής της κυματομορφής του σήματος ομιλίας είναι πολύ μεγαλύτερος του ρυθμού άρθρωσης ενός ομιλητή. Η κλασική επεξεργασία σήματος εφαρμόζεται σε στάσιμα σήματα. Η φωνή είναι ένα μη στάσιμο σήμα και επομένως δεν μπορεί να εφαρμοστούν οι τυπικές μέθοδοι ανάλυσης. Η φωνή έχει **τοπικά στατικές** ιδιότητες και έτσι για την ανάλυση της χρησιμοποιούνται **βραχυχρόνιες επεξεργασίες**. Μικρά τεμάχια ομιλίας απομονώνονται και υφίστανται επεξεργασία σαν να περιέχουν ήχο με στάσιμες ιδιότητες. Αυτά τα μικρά τμήματα ομιλίας (N δειγμάτων) θα αναφέρονται ως διαστήματα ομιλίας (frames).

Διάστημα (frame) μίας κυματομορφής φωνής $x(m)$ γύρω από το σημείο n ορίζεται ως το γινόμενο του σήματος της φωνής επί κάποιο σήμα-**παράθυρο** $w(m)$ μετατοπισμένο στην γειτονιά του σημείου n.

$$x_n(m) = x(m)\ (n - m). \tag{2.34}$$

2.3.5 Παράθυρα

Το παράθυρο έχει μη μηδενικές τιμές σε μία περιοχή κοντά στο μηδέν και έτσι το πλαίσιο $x_n(m)$ έχει μη μηδενικές τιμές μόνο σε μία περιοχή κοντά στο σημείο n.

Τετραγωνικό Παράθυρο

$$w_R(n) = \begin{cases} 1, & 0 \le n \le N-1 \\ 0, & \text{αλλού} \end{cases} \tag{2.35}$$

Ο μετασχηματισμός Fourier του τετραγωνικού παραθύρου είναι:

$$W_R(e^{j\omega}) = \sum_{n=0}^{N-1} e^{-j\omega n} = \frac{1 - e^{-j\omega N}}{1 - e^{-j\omega}} = \frac{e^{-j\omega N/2}(e^{j\omega N/2} - e^{-j\omega N/2})}{e^{-j\omega/2}(e^{j\omega/2} - e^{-j\omega/2})} = e^{-\frac{j\omega(N-1)}{2}} \frac{\sin(\omega N/2)}{\sin(\omega/2)} \tag{2.36}$$

Ο μετασχηματισμός Fourier του τετραγωνικού παραθύρου μηδενίζεται στις συχνότητες $2l\pi/N$, και έτσι ο κεντρικός λοβός του τετραγωνικού παραθύρου έχει εύρος $\frac{2\pi}{N}$.

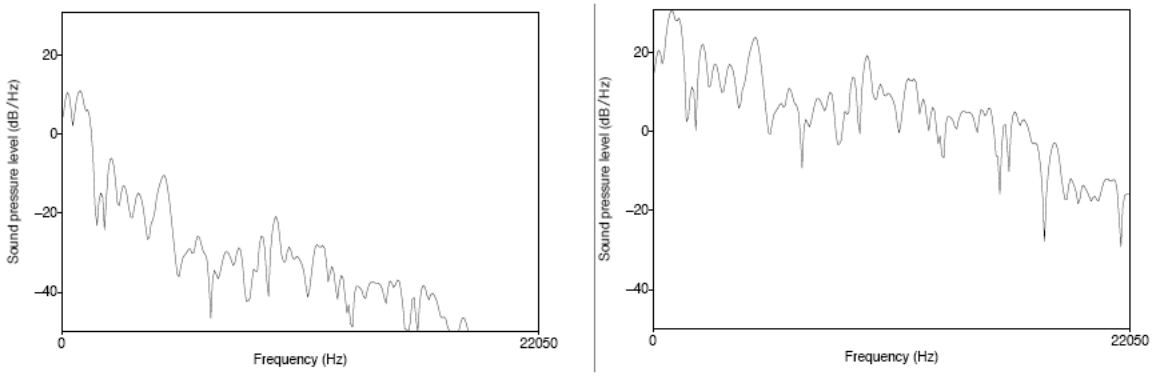
Παράθυρο Hamming

$$w_R(n) = \begin{cases} 0.54 - 0.46cos(2\pi\frac{n}{N-1}), & 0 \le n \le N-1 \\ 0, & \text{αλλού} \end{cases} \tag{2.37}$$

Το παράθυρο Hamming έχει ευρύτερο κεντρικό λοβό από τον τετραγωνικό, αλλά έχει μεγαλύτερη απόσβεση στους πλευρικούς λοβούς με εύρος λοβου $\frac{4\pi}{N}$.

2.3.6 Προ-Έμφαση (Pre-Emphasis)

Για τημ ενίσχυση της ενέργειας στις υψηλές συχνότητες χρησιμοποιείτε το φάσμα που στα ηχηρά διαστήματα έχει περισσότερη ενέργεια στις χαμηλής συχνότητες από ότι στις υψηλές. Ονομάζεται φασματική κλίση (spectral tilt), και αποδίδεται στην παρουσία του γλωττιδικού παλμού (glottal pulse).



Σχήμα 2.1: Το σήμα πριν και μετά από προ-Έμφαση στο φωνήεν [aa],
Πηγή : **https://bit.ly/2LBrYyl**

2.3.7 Βραχύχρονος Μετασχηματισμός Fourier (Short-time Fourier Transform - STFT)

Ορίζεται ως ο μετασχηματισμός Fourier Διακριτού Χρόνου του διαστήματος περί το δείγμα n:

$$X_n(e^{jw}) = \sum_{m=-\infty}^{\infty} w(n-m)x(m)e^{-jwm} = \sum_{m=-\infty}^{\infty} x_n(m)e^{-jwm}. \tag{2.38}$$

Ο Μετασχηματισμός Fourier Βραχέως Χρόνου είναι συνάρτηση δύο μεταβλητών:

1. Του χρόνου (n - διακριτή μεταβλητή)

2. Της κανονικοποιημένης συχνότητας (ω - συνεχής μεταβλητή)

Με άλλα λόγια, ο STFT μας δίνει το μετασχηματισμό Fourier διακριτού χρόνου (DTFT για ένα παράθυρο του αρχικού σήματος στην γειτονιά του δείγματος n.

Ερμηνεία του STFT στο Πεδίο της Συχνότητας

Αν το παράθυρο $w(m)$ έχει μη μηδενικές τιμές για $m = 0, ..., L-1$, τότε το πλαίσιο $x_n(m)$

έχει μη μηδενικές τιμές στην περιοχή

$$x_n(m) = \begin{cases} \neq 0 & , n - \mathrm{L} + 1 \leq m \leq n \\ = 0 & , \text{αλλού} \end{cases} \tag{2.39}$$

Έτσι, ο STFT μπορεί να οριστεί για το παράθυρο αυτής της μορφής ως:

$$X_n(e^{i\omega}) = \mathrm{DTFT}[x_n(m)] = \sum_{m=n-L+1}^{n} x_n(m)e^{-j\omega m} \tag{2.40}$$

Ο STFT είναι ο μετασχηματισμός Fourier του γινομένου $w(n-m)x(m)$ για κάποια τιμή του n:

$$X_n(e^{j\omega}) = DTFT[x_n(m)] = DTFT[w(n-m)x(m)] \tag{2.41}$$

όπου θεωρήσαμε μεταβλητή του χρόνου την m. Έστω οι μετασχηματισμοί Fourier του αρχικού σήματος και του παραθύρου:

$$X_n(e^{i\omega}) = \sum_{m=-\infty}^{\infty} x(m)e^{j\omega} \tag{2.42}$$

$$W(e^{i\omega}) = \sum_{m=-\infty}^{\infty} w(m)e^{j\omega} \tag{2.43}$$

με αντίστροφο μετασχηματισμό:

$$w(m) = \frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\omega})\ e^{j\omega m}\ d\omega \tag{2.44}$$

Ο STFT του $x(m)$ μπορεί να γραφτεί:

$$X_n(e^{j\omega}) = \sum_{m=-\infty}^{\infty} x(m)[\frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\theta})e^{j\theta(n-m)}d\theta]e^{-j\omega m} \tag{2.45}$$

ή ισοδύναμα:

$$X_n(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{pi} W(e^{j\theta})e^{j\theta n}[\sum_{m=-\infty}^{\infty} x(m)e^{-j(\theta+\omega)m}]d\theta \tag{2.46}$$

δηλαδή:

$$X_n(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} W(e^{j\theta})\ X(e^{j(\theta+\omega)})\ d\theta\ e^{j\theta n} \tag{2.47}$$

Επομένως:

- Ο STFT αντιστοιχεί στην **συνέλιξη** των μετασχηματισμών Fourier του παραθύρου και του αρχικού σήματος.
- **Μεγάλες τιμές** του μήκους του παραθύρου L θα δώσουν καλή ανάλυση στην συχνότητα. Όσο μεγαλώνει η τιμή του L, τόσο ο μετασχηματισμός Fourier $W(e^{j\omega})$ πλησιάζει την συνάρτηση δέλτα.
- Στην πράξη, η προηγούμενη παρατήρηση σημαίνει ότι το παράθυρο θα πρέπει να είναι μεγάλο σε σχέση με τη θεμελιώδη συχνότητα (pitch) της φωνής.
- **Μικρές τιμές** του L θα δώσουν χειρότερη ανάλυση στην συχνότητα, αλλά θα παρακολουθούν χρονικές μεταβολες καλύτερα.

2.3.8 Φασματογράφημα (Spectrogram)

Το σήμα της φωνής είναι εν γένει ένα **μη στάσιμο** σήμα. Μπορεί να θεωρηθεί μόνο **τοπικά στάσιμο** (local stationary) κατά τη διάρκεια ενός συγκεκριμένου βασικού ήχου. Για το σκοπό αυτό, μια χρήσιμη μορφή αναπαράστασης του σήματος της φωνής είναι το **φασματόγραμμα**, το οποίο παριστάνει γραφικά την μεταβολή του φασματικού περιεχομένου του σήματος της φωνής ως προς τον χρόνο. Το φασματόγραμμα είναι μια δισδιάστατη γραφική αναπαράσταση του σήματος της φωνής, όπου ο χρόνος απεικονίζεται στον άξονα των τετμημένων η συχνότητα στον άξονα των τεταγμένων και η ένταση της εικόνας σε κάθε σημείο να εξαρτάται από την ενέργεια που περιέχει το σήμα στη συγκεκριμένη συχνότητα και χρονική στιγμή.

- Υπολογίζεται χρησιμοποιώντας τον STFT ενός διαστήματος φωνής ανά μικρά χρονικά διαστήματα και σχεδιάζοντας το μέτρο.
- Σύμφωνα με την αρχή της Αβεβαιότητας (Uncertainty Principle), δεν μπορούμε να έχουμε ακριβή ανάλυση και στο χρόνο και στη συχνότητα, υπάρχει δηλαδή μια αντιστάθμιση.
- **Φασματόγραμμα Ευρείας Ζώνης** (Wideland Spectrogram): Δίνει έμφαση στην ανάλυση στο χρόνο, έτσι ώστε να παρακολουθεί γρήγορες μεταβολές του φάσματος. Υπολογίζεται χρησιμοποιώντας μικρά εύρη παραθύρων (περίπου 5-20 msec).
- **Φασματόγραμμα Στενής Ζώνης** (Narrowband Spectrogram): Δίνει έμφαση στην ανάλυση στο συχνότητα, έτσι ώστε να αναπαριστά το φάσμα με αρκετή λεπτομέρεια. Υπολογίζεται χρησιμοποιώντας παράθυρα μεγαλύτερης διάρκειας (της τάξης των 50 msec).

2.3.9 Ερμηνεία του STFT Χρησιμοποιώντας Φίλτρα

Ο STFT ενός σήματος μπορεί να γραφεί:

$$\begin{aligned} X_n(e^{j\omega}) = \sum_{m=-\infty}^{\infty} w(n-m)x(m)e^{-j\omega m} = \\ e^{-j\omega n} \sum_{m=-\infty}^{\infty} w(n-m)e^{j\omega(n-m)}x(m) = \\ e^{-j\omega n}\Big\{\big\{w(n)e^{j\omega n}\big\} * x(n)\Big\} \end{aligned} \tag{2.48}$$

Υπολογίζεται, δηλαδή από τη συνέλιξη του $x(n)$ με ένα ζωνοδιαβατό φίλτρο με κρουστική απόκριση $w(n)e^{j\omega n}$

- Το φίλτρο $w(n)$ είναι κατωδιαβατό. Άρα το φίλτρο με απόκριση $w(n)e^{j\omega_0 n}$ που έχει μετασχηματισμό Fourier

$$DTFT\big[w(n)e^{j\omega_0 n}\big] = W(e^{j(\omega-\omega_0)}) \tag{2.49}$$

  είναι ένα μιγαδικό ζωνοδιαβατό φίλτρο γύρω από την συχνότητα ω_0.

- Η ερμηνεία αυτή υπαγορεύει τον υπολογισμό του STFT **χρησιμοποιώντας σειρές από τέτοια φίλτρα που ονομάζονται τράπεζες φίλτρων**.

2.3.10 Τράπεζες φίλτρων

Οι τράπεζες φίλτρων που χρησιμοποιούνται για τον υπολογισμό του STFT χαρακτηρίζεται από τις κεντρικές συχνότητες ω_0 των φίλτρων, αλλά και από το εύρος ζώνης που έχει το ζωνοπερατό φίλτρο σε κάθε συχνότητα.

Ομοιόμορφες τράπεζες φίλτρων

Οι κεντρικές συχνότητες στην περίπτωση των ομοιόμρφων (**uniform**)τραπεζών φίλτρων είναι ομοιόμορφα κατανεμημένες στην κλίμακα της συχνότητας:

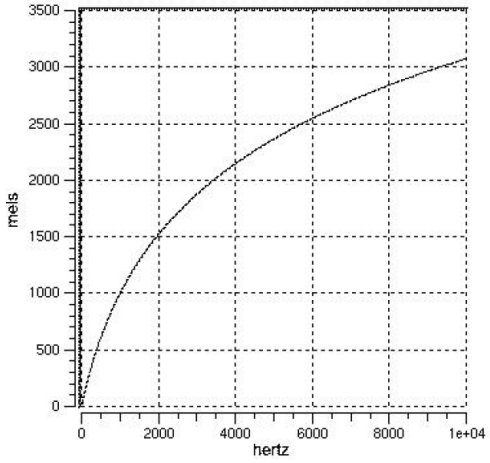
$$f_i = \frac{f_S}{N} i, \qquad 1 \leq i \leq Q \tag{2.50}$$

όπου:

- f_S είναι η συχνότητα δειγματοληψίας
- Ο αριθμός των φίλτρων Q ικανοποιεί $Q \leq \frac{N}{2}$
- Το εύρος ζώνης κάθε φίλτρου ικανοποιεί $b_i \leq \frac{f_S}{N}$

2.4 Κλίμακα και Συντελεστές Mel (Mel-frequency Cepstral Coefficients - MFCC)

2.4.1 Κλίμακα Mel



Σχήμα 2.2: Κλίμακα Mel, Πηγή : **https://bit.ly/2l3bx2J**

Η αντίληψη της συχνότητας των τόνων απο τον άνθρωπο **δεν ακολουθεί γραμμική κλίμακα**. Το ανθρώπινο αυτί είναι ένας μη γραμμικός δέκτης: ενισχύει κάποια χαρακτηριστικά και υποβιβάζει άλλα. Το κρίσιμο εύρος ζώνης γύρω από μια κεντρική συχνότητα είναι το εύρος στο οποίο όλες οι συχνότητες ηχούν το ίδιο. Η κλίμακα Mel ορίζεται ως η υποκειμενική εκτίμηση της συχνότητας ενός τόνου συγκεκριμένης συχνότητας. Έχει υπολογιστεί με

ψυχοακουστικές μελέτες. Ορίζεται ως η απεικόνιση των πραγματικών συχνοτήτων σε συχνότητες εκφρασμένες σε Mel. Η αντιστοιχία μεταξύ πραγματικής κλίμακας συχνοτήτων $f(Hz)$ και αντιλαμβανόμενης κλίμακας συχνοτήτων $Mel(f)$ είναι:

$$Mel(f) = 2595 \log_{10}(1 + \frac{f}{10}) \tag{2.51}$$

2.4.2 Συντελεστές χάσματος στην κλίμακα Mel

Οι συντελεστές χάσματος στην κλίμακα Mel (Mel-frequency cepstral coefficients - MFCC), χρησιμοποιούνται ευρέως στην αναγνώριση της ομιλίας. Έχουν αποδειχτεί αρκετά αποτελεσματικά στις εφαρμογές εξόρυξης μουσικής πληροφορίας. Επίσης τα MFCC μας δίνουν γενικότερα πληροφορίες για το φασματικό περιεχόμενο και την φασματική περιβάλλουσα ενός διαστήματος ομιλίας. Υπολογίζονται από το διακριτό μετασχηματισμό συνημίτονου (DCT) του λογαριθμικού φάσματος, μετά από μια μη γραμμική στρέβλωση συχνοτήτων, στη κλίμακα Mel .Πιο συγκεκριμένα, η διαδικασία περιλαμβάνει την κατάτμηση του σήματος σε διαστήματα των 20 - 30 msec τα οποία είναι αλληλεπικαλυπτόμενα με επικάλυψη 50% - 75% και εφαρμόζεται σε αυτά το παράθυρο Hamming, ώστε να εξομαλυνθούν τα άκρα του ακουστικού σήματος. Στην συνέχεια χρησιμοποιείται ο διακριτός μετασχηματισμός Fourier (Discrete Fourier Transform - DFT) , ώστε να υπολογιστεί το φάσμα ισχύος του κάθε τμήματος, και στο οποίο φάσμα απεικονίζεται η κάθε συχνότητα Hz στην κλίμακα Mel, χρησιμοποιώντας μια τράπεζα φίλτρων (Filterbanks). Επειδή όμως το ανθρώπινο σύστημα ακοής αντιλαμβάνεται λογαριθμικά την ένταση ενός ηχητικού σήματος, υπολογίζεται ο λογάριθμος των ενεργειών στην έξοδο όλων των φίλτρων σε κάθε μία από τις συχνότητες Mel και η διαδικασία ολοκληρώνεται, εφαρμόζοντας ανάστροφο DCT.
Με την χρήση των MFCC καταλήγουμε σε ένα πολύ απλοποιημένο μοντέλο ακουστικής επεξεργασίας, καθώς έιναι και εύκολο και γρήγορα υλοποιήσιμο. Ο υπολογισμός των MFCC συντελεστών ακολουθεί τα παρακάτω βήματα:

1. Το σήμα κατατμείται χρησιμοποιώντας παράθυρα (Hamming ή Hanning) μήκους 10-20 ms τα οποία μετατοπίζονται κατά 5-10 ms.

2. Υπολογίζεται φάσμα για κάθε διάστημα χρησιμοποιώντας τον μετασχηματισμό Fourier.

3. Το διάστημα στην συνέχεια φιλτράρεται από τράπεζα φίλτρων της κλίμακας Mel για να αποκτήσει αντιστοιχία με τους συντελεστές Mel.

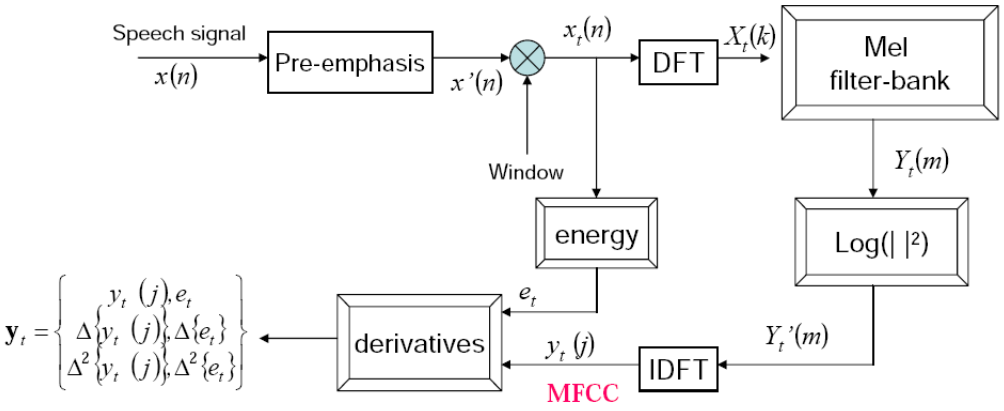
4. Στην συνέχεια υπολογίζονται οι λογάριθμοι των συντελεστών Mel.

5. Ο διακριτός μετασχηματισμός συνημίτονου χρησιμοποιείται για την μετατροπή τους στον χώρο της ισχνότητας. 6

6. Οι μη-απαραίτητοι (υψηλής ισχνότητας) MFCC συντελεστές απορρίπτονται.

20 MFCC συντελεστές χρησιμοποιούνται συνήθως σε συστήματα αναγνώρισης συναισθημάτων, αλλά ακόμα και 10-12 είναι επαρκείς. Το πιο σημαντικό μειονέκτημα της χρήσης των συντελεστών MFCC είναι η ευαισθησία στον θόρυβο, λόγω της εξάρτησης τους από την περιβάλλουσα του φάσματος.

Το σήμα της ομιλίας δεν είναι στάσιμο και θέλουμε να προσθέσουμε μεταβολές αυτές των χαρακτηριστικών. Τα χαρακτηριστικά αυτά ονομάζονται Delta και Delta-Delta που είναι στην ουσία η πρώτη και δεύτερη παράγωγος των συντελεστών Mel και μπορούν να μεταφραστούν σε χαρακτηριστικά επιτάχυνσης (acceleration features). Για την τελική εξαγωγή των χαρακτηριστικών του σήματος της ομιλίας χρειαζόμαστε και την παράμετρο της ενέργειας που περιέχει χρήσιμη πληροφορία καθως οι συντελεστές Mel δεν εμπεριέχουν ενέργεια. Καταλήγοντας τελικά στο παρακάτω σύστημα εξαγωγής χαρακτηριστικών της ομιλίας που και χρησιμοποιήθηκε στην παρούσα πτυχιακή εργασία [4].



Σχήμα 2.3: Σύστημα εξαγωγής χαρακτιριστικών, Πηγή : **https://bit.ly/2HyHA3g**

Κεφάλαιο 3

Εισαγωγή στα Νευρωνικά Δίκτυα

3.1 Τεχνητή Νοημοσύνη

Ο όρος τεχνητή νοημοσύνη (ΤΝ) ή στα Αγγλικά Artificial Intelligence (AI) σχετίζεται με τον κλάδο της πληροφορικής που μελετά τη σχεδίαση υπολογιστικών συστημάτων τα οποία μιμούνται την ανθρώπινη συμπεριφορά υποδηλώνοντας την παρουσία στοιχειώδους ευφυΐας, καθώς και την υλοποίησή τους [5]. Σχετίζεται επίσης με την εκμάθηση και προσαρμογή στο περιβάλλον, η εξαγωγή συμπερασμάτων και γενικότερα η επίλυση απλών ή και πολύπλοκων προβλημάτων. Ο Τζον Μακάρθι, μια από τις σημαντικότερες φυσιογνωμίες στον τομέα της θεωρητικής πληροφορικής ορίζει τον συγκεκριμένο κλάδο ως «Τη επιστήμη και μεθοδολογία της δημιουργίας νοήμονων μηχανών».

Μεγάλος αριθμός επιστημών συναντάται και συνεισφέρει στην ΤΝ, όπως για παράδειγμα η επιστήμη της πληροφορικής, της ψυχολογίας και της φιλοσοφίας, η νευρολογία, η επιστήμη των μηχανών, ακόμη και η επιστήμη της γλωσσολογίας προκειμένου να γίνει εφικτή η σύνθεση ευφυούς συμπεριφοράς, η εκμάθηση και προσαρμογή στο εκάστοτε περιβάλλον μηχανών και υπολογιστών συγκεκριμένου συνήθως σκοπού. Η ΤΝ διαχωρίζεται σε δυο κομμάτια, τη συμβολική και τη στατιστική νοημοσύνη. Η πρώτη συνίσταται στην προσπάθεια εξομοίωσης της ανθρώπινης συμπεριφοράς με χρήση ειδικών αλγορίθμων που χρησιμοποιούν ένα σύνολο συμβόλων και λογικών κανόνων υψηλού επιπέδου, ενώ η δεύτερη στόχο έχει την προσέγγιση ή ακόμα και την αναπαραγωγή της ανθρώπινης ευφυΐας μέσα από αριθμητικά μοντέλα, τα οποία επαγωγικά συνθέτουν συμπεριφορές που υποδηλώνουν ευφυΐα, προσεγγίζοντας πραγματικές βιολογικές συμπεριφορές όπως η διαδικασία της εξέλιξης και η λειτουργία του ανθρώπινου εγκεφάλου. Αξιοποιεί την αναγνώριση προτύπων/μηχανική μάθηση για να επιλύσει προβλήματα ομαδοποίησης και ταξινόμησης.

Ανάλογα με τον επιθυμητό στόχο η ΤΝ μπορεί να χωριστεί σε ένα ευρύτερο σύνολο τομέων. Παραδείγματος χάριν στη μηχανική εκμάθηση, την επίλυση προβλημάτων, τα συστήματα γνώσης, υπολογιστική όραση, σύνθεση και αναγνώριση φυσικής γλώσσας και ρομποτική, οι οποίες μπορούν να θεωρηθούν ανεξάρτητες συνιστώσες - πεδία της σύγχρονης ΤΝ.

Ο κινηματογραφικά έργα επιστημονικής φαντασίας αλλά και η λογοτεχνία από τις αρχές της δεκαετίας 1960 έχουν ασχοληθεί με την ΤΝ περιγράφοντας μια ιδεατή πραγματικότητα όπου ρομπότ και υπολογιστικά συστήματα συνυπάρχουν με τους ανθρώπους και τους εξυπηρετούν στην καθημερινότητα. Ωστόσο, η λανθασμένη εντύπωση που έχει προκληθεί στο ευρύ κοινό περί κατασκευής μηχανικών ανδροειδών, αυτοσυνείδητων υπολογιστικών συστημάτων με σκοπό ακόμη και την αντικατάσταση του ανθρώπου δεν εκλείπουν, έχοντας επηρεάσει ακόμα και τους πρωτοπόρους επιστήμονες της ΤΝ. Στην πραγματικότητα, οι ερευνητές της ΤΝ έχουν ως σκοπό την ανάπτηξη λογισμικού και μηχανών ικανών να επιλύουν πραγματικά προβλήματα διαφόρων τύπων (ασθενής ΤΝ), ενώ αρκετοί αποβλέπουν στην προσομοίωση της πραγματικής ευφυΐας, τη λεγόμενη ισχυρή ΤΝ. Στην εποχή μας, η ΤΝ αποτελεί ένα από τα ταχύτερα εξελισσόμενα πεδία της επιστήμης, ενώ με τη χρήση εργαλείων τα εφαρμοσμένα μαθηματικά και επιστήμες μηχανικών έχει ξεφύγει από τα πλαίσια της θεωρητικής πληροφορικής.

3.2 Μηχανική μάθηση - Γενικά

Η μηχανική μάθηση (machine learning), αποτελεί ένα από τα σημαντικότερα κομμάτια της ΤΝ. Αφορά την ανάπτυξη αλγορίθμων κατάλληλων που θα δώσουν τη δυνατότητα της ''εκμάθησης'' π.χ. μιας συνάρτησης που υλοποιεί ένα ταξινομητή ή ένα μοντέλο για εκτίμηση μιας τιμής εξόδου. Το λογισμικό που χρησιμοποιείται από τους υπολογιστές γίνεται πλέον ευέλικτο και προσαρμόσιμο με βάση την ανάλυση των δεδομένων που λαμβάνουν, αντί της κλασικής πλέον προσαρμογής τους με βάση την διαίσθηση του μηχανικού που προγραμματίζει κάποιο σύστημα. Όλη η ουσία της μηχανικής μάθησης συνοψίζεται στη χρήση αλγορίθμων, ικανών να αναγνωρίζουν πρότυπα στα δεδομένα προκειμένου να λάβουν αποφάσεις, βασιζόμενων στη στατιστική, τη θεωρία των πιθανοτήτων διαλογής και τη βελτιστοποίηση. Χάρις στην μηχανική μάθηση, απολαμβάνουμε υπηρεσίες, όπως φίλτρα ανεπιθύμητης αλληλογραφίας, αναγνώριση κειμένου και φωνής, αξιόπιστες μηχανές αναζήτησης στο διαδίκτυο, και ελπίζουμε σύντομα σε αυτό-οδηγούμενα μέσα μεταφοράς. Ανάλογα με το επιθυμητό αποτέλεσμα, οι αλγόριθμοι ΤΝ χωρίζονται στις εξής κατηγορίες:

- Επιτηρούμενη μάθηση ή μάθηση υπο επίβλεψη (supervised learning), όπου ο αλγόριθμος κατασκευάζει μια συνάρτηση που απεικονίζει επισημειωμένες εισόδους (labeled examples) σε γνωστές - επιθυμητές εξόδους (σύνολο εκπαίδευσης), κάνοντας προβλέψεις και διορθώνοντας τις προβλέψεις σε περίπτωση λάθους, με απώτερο στόχο τη γενίκευση της συνάρτησης αυτής για εισόδους με άγνωστη έξοδο (σύνολο ελέγχου).

- Μη επιτηρούμενη μάθηση ή μάθηση χωρίς επίβλεψη (unsupervised learning), όπου ο αλγόριθμος κατασκευάζει ένα μοντέλο για κάποιο σύνολο εισόδων χωρίς να γνωρίζει τις επιθυμητές εξόδους (unlabeled examples) για το σύνολο εκπαίδευσης ανακαλύπτοντας δομές, όπως για παράδειγμα για εξαγωγή γενικών κανόνων.

- Ημι-επιτηρούμενη μάθηση (semi-supervised learning), όπου τα δεδομένα εκπαίδευσης είναι μια μίξη γνωστών και αγνώστων δειγμάτων (mixture of labeled and unlabeled examples) σε ένα πρόβλημα π.χ. πρόβλεψης, αλλά το μοντέλο πρέπει να μάθει δομές για

να οργανώσει τα δεδομένα και να κάνει προβλέψεις, όταν δεν υπάρχει πληροφορία κλάσης, όπως συμβαίνει για το μεγαλύτερο όγκο των δεδομένων.

Η ανάλυση των δυο πρώτων αλγορίθμων θα γίνει στη συνέχεια, ενώ ο τρίτος αναφέρεται για λόγους πληρότητας.

3.3 Νευρωνικά Δίκτυα - Γενικά

Τα Νευρωνικά Δίκτυα (ΝΔ), χωρίζονται σε Βιολογικά Νευρωνικά Δίκτυα (ΒΝΔ) και σε Τεχνητά Νευρωνικά Δίκτυα (ΤΝΔ). Τα ΒΝΔ αποτελούν μέρος του κεντρικού νευρικού συστήματος βιολογικών συστημάτων, π.χ. του ανθρώπου. Συνίστανται από βιολογικό ιστό, χημικές ουσίες και ηλεκτρικά σήματα. Αντιθέτως τα ΤΝΔ που προσπαθούν να μιμηθούν τα ΒΝΔ χρησιμοποιώντας ένα σύνολο ηλεκτρονικών και μηχανικών συστημάτων συνοδευόμενα από ευφυείς αλγορίθμους.

3.3.1 Βιολογικά Νευρωνικά Δίκτυα

Το ανθρώπινο σώμα αποτελείται από ένα τεράστιο αριθμό κυττάρων, περίπου. Κάθε κύτταρο αποτελεί μια στοιχειώδη μονάδα μεταφοράς και επεξεργασίας ηλεκτρικών σημάτων και χημικών διαδικασιών και ονομάζεται νευρώνας. Εκτιμήσεις δείχνουν ότι ένας ανθρώπινος εγκέφαλος περιλαμβάνει περίπου νευρώνες με διάφορα σχήματα και μεγέθη περίπου $100\mu m$.
Οι νευρώνες είναι παρόμοιοι με άλλα κύτταρα γιατί:

1. Περιβάλλονται από κυτταρική μεμβράνη,
2. Έχουν πυρήνα που περιλαμβάνει γονίδια,
3. Περιέχουν κυτταρόπλασμα, μιτοχόνδρια και άλλα οργανίδια,
4. Πραγματοποιούν βασικές κυτταρικές διεργασίες όπως πρωτεϊνική σύνθεση και παραγωγή ενέργειας.

Ωστόσο διαφέρουν από άλλα κύτταρα γιατί:

1. Έχουν ειδικευμένα μέρη που ονομάζονται δενδρίτεσ(dendrites) και άξονες ή (axons). Οι πρώτοι μεταφέρουν ηλεκτρικά σήματα στον νευρώνα και οι δεύτεροι από τον νευρώνα προς κάποιο άλλο μέρος του εγκεφάλου,
2. Οι νευρώνες επικοινωνούν μεταξύ τους μέσω ηλεκτροχημικών διεργασιών,
3. Περιλαμβάνουν ορισμένες εξειδικευμένες δομές, όπως για παράδειγμα τις συνάψεις (synapses) και χημικά όπως για παράδειγμα τους νευροδιαβιβαστές (neurotransmitters).

Οι συνάψεις ενώνουν τους βιολογικούς νευρώνες και δίνουν τη δυνατότητα ανταλλαγής του ηλεκτρικού δυναμικού μεταξύ των δενδριτών. Οι ανταλλαγές αυτές λαμβάνουν χώρα σε μεγάλη πυκνότητα με πολύ αργό ρυθμό περίπου 100Hz. Ο μέσος άνθρωπος έχει περίπου 10^{14} συνάψεις και επομένως συμβαίνουν 10^{16} interactions/sec. Οι δυνατότητες αυτές βρίσκονται πέρα από τις

αντίστοιχες κάθε συστήματος που μπορεί να κατασκευαστεί ή και να σχεδιαστεί. Οι σημερινοί υπολογιστές προσεγγίζουν τις 10^9 interactions/sec. Ένας τρόπος κατηγοριοποίησής τους, είναι ο αριθμός των επεκτάσεων (extensions) από το σώμα του νευρώνα και η κατεύθυνσή τους.

3.3.2 Τεχνητά Νευρωνικά Δίκτυα

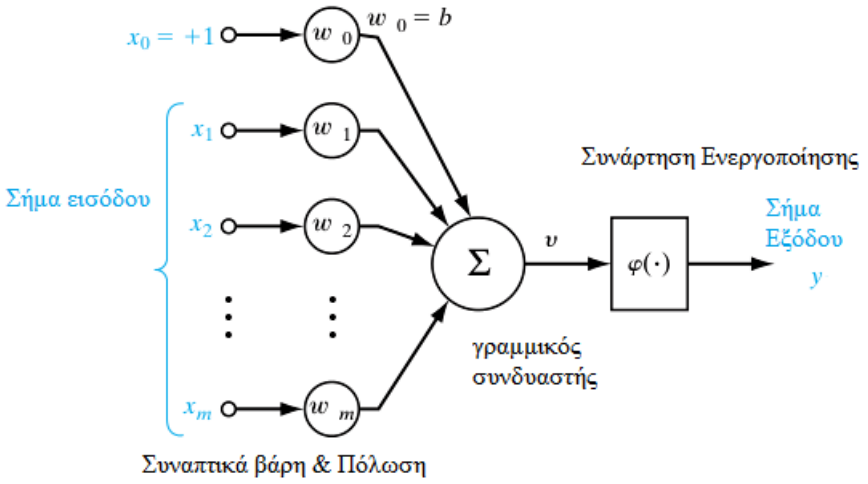
Τα ΤΝΔ αποτελούν αναπόσπαστο κομμάτι του πεδίου της μηχανικής μάθησης. Η έρευνα σχετικά με τα ΝΔ είναι κατά ένα μεγάλο ποσοστό εμπνευσμένη από τη λειτουργία και την δομή του ανθρώπινου εγκεφάλου. Τα τελευταία χρόνια από την δεκαετία του 1980, τα νευρωνικά δίκτυα βρίσκουν πολλές εφαρμογές και εφαρμόζονται σε πολλούς διαφορετικούς επιστημονικούς κλάδους, όπως για παράδειγμα στην ιατρική, στη φυσική, στη ρομποτική, στα χρηματοοικονομικά κ.α. Η επιτυχία τους βασίζεται στην ικανότητά τους να προσαρμόζουν αυτόματα τις παραμέτρους τους καθώς και στην ικανότητα τους να μοντελοποιούν πολύπλοκες και μη γραμμικές σχέσεις μεταξύ των δεδομένων. Η τελευταία ιδιότητα επιτρέπει στα ΝΔ να δώσουν αποδοτικές λύσεις σε διάφορα προβλήματα αναγνώρισης προτύπων και ταξινόμησης.

Ένα ΝΔ εκτελεί μία διαδικασία αναγνώρισης προτύπων αφού πρώτα περάσει από μία διαδικασία εκπαίδευσης κατά την οποία παρουσιάζεται στο δίκτυο επαναληπτικά ένα σύνολο προτύπων εισόδου μαζί με την κλάση στην οποία ανήκει το καθένα, δηλαδή την επιθυμητή έξοδο. Στη συνέχεια, αφού ολοκληρωθεί η διαδικασία της εκπαίδευσης, το δίκτυο είναι σε θέση να αναγνωρίσει την κλάση στην οποία ανήκουν άγνωστα πρότυπα βάση της πληροφορίας που έχει εξαχθεί κατά την διάρκεια της εκπαίδευσης. Η συγκεκριμένη ιδιότητα είναι η πιο σημαντική ιδιότητα των νευρωνικών δικτύων και είναι γνωστή ως **μάθηση**. Ένα ΝΔ είναι σε θέση να αναπαραστήσει ένα σύνολο προτύπων σ΄ έναν πολυδιάστατο χώρο αποφάσεων, ο οποίος διαιρείται σε περιοχές ανάλογα με τις κλάσεις του προβλήματος.

3.4 Το μοντέλο του τεχνητού νευρώνα

Το 1943 οι McCulloch και Pitts περιέγραψαν ένα απλό μοντέλο ενός τεχνητού νευρώνα. Το 1958 ο Rosenblatt πρότεινε το Perceptron [6], ως το πρώτο μοντέλο μάθησης αποτελούμενο από έναν νευρώνα. Η κατάσταση ενός νευρώνα περιγράφεται από ένα δυαδικό αριθμό y όπου όταν $y = 0$ ο νευρώνας είναι αδρανής (δεν ενεργοποιείται), ενώ αντίθετα όταν $y = 1$ ο νευρώνας ενεργοποιείται. Η βασική δομή ενός τεχνητού νευρώνα φαίνεται παρακάτω στο Σχήμα 3.1.

Ένας νευρώνας αποτελείται από m εισόδους $[x_1, x_2, \ldots, x_m]^T$, m συναπτικά βάρη $[w_1, w_2, \ldots, w_m]^T$ καθώς και μία πόλωση b (bias). Ο νευρώνας αφού πολλαπλασιάσει τις τιμές των εισόδων με τις τιμές των αντίστοιχων βαρών υπολογίζει το σταθμισμένο άθροισμα των εισόδων $\sum_i w_i x_i$ και εφαρμόζει σε αυτό μία μη γραμμική συνάρτηση η οποία λέγεται συνάρτηση ενεργοποίησης. Οι λειτουργία της άθροισης των σημάτων εισόδου, σταθμισμένα από τα αντίστοιχα συναπτικά βάρη συνιστούν έναν γραμμικό συνδυαστή (linear combiner). Η συνάρτηση ενεργοποίησης περιορίζει το επιτρεπτό εύρος πλάτους του σήματος εξόδου σε κάποια περιορισμένη τιμή. Τυπικά, το κανονικοποιημένο εύρος τιμών πλάτους της εξόδου ενός νευρώνα γράφεται ως μοναδιαίο κλειστό διάστημα με τη μορφή $[0, 1]$ ή $[-1, 1]$. Μπορούμε να περιγράψουμε το



Σχήμα 3.1: Το μοντέλο ενός τεχνητού νευρώνα, Πηγή : **https://bit.ly/2MexmIL**

νευρώνα του σχήματος με τις παρακάτω εξισώσεις:

$$v = \sum_{i=1}^{m} w_i x_i + b \tag{3.1}$$

$$y = \phi(v) \tag{3.2}$$

όπου v συμβολίζουμε την έξοδο του γραμμικού συνδιαστή, $\phi(\cdot)$ είναι η συνάρτηση ενεργοποίησης και y είναι το σήμα εξόδου του νευρώνα. Συνήθως, η πόλωση b εκφράζεται μέσω ενός συναπτικού βάρους $w_0 = b$ το οποίο αναφέρεται σε μία είσοδο $x_0 = 1$.

Η εξίσωση $v = \sum_{i=1}^{m} w_i x_i + b = 0$ αντιστοιχεί σε ένα υπερεπίπεδο στο χώρο $\mathbb{R}^m$. Τα σημεία που αντιστοιχούν σε θετικές τιμές $v > 0$ βρίσκονται από τη μία πλευρά του υπερεπιπέδου ενώ τα σημεία για τα οποία ισχύει $v < 0$ βρίσκονται από την άλλη πλευρά του υπερ-επιπέδου. Συνεπώς, το υπερεπίπεδο $v = 0$ διαμερίζει τον χώρο $\mathbb{R}^m$ σε δύο μέρη, όπου το ένα αντιστοιχεί σε $y = 0$ και το άλλο αντιστοιχεί σε $y = 1$. Στην περίπτωση όπου οι κλάσεις του προβλήματος είναι **γραμμικά διαχωρίσιμες** το υπερεπίπεδο διαχωρίζει πλήρως τις κλάσεις και ο νευρώνας λειτουργεί ως **γραμμικός ταξινομητής**.

3.5 Συναρτήσεις Ενεργοποίησης

Η συνάρτηση ενεργοποίησης $\phi(\cdot)$ ορίζει την έξοδο ενός νευρώνα βάσει της τιμής ενεργοποίησης v. Στην συνέχεια, περιγράφονται οι πιο κοινές επιλογές συναρτήσεων ενεργοποίησης.

1. Συνάρτηση Κατωφλίου (Treshold) Η συνάρτηση κατωφλίου είναι η πιο απλή μορφή συνάρτησης ενεργοποίησης και ορίζεται ως:

$$\phi(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v \leq 0 \end{cases} \tag{3.3}$$

Η συνάρτηση 3.3 αναφέρεται συχνά και ως συνάρτηση Heaviside και v η τιμή ενεργοποίησης που δίνεται στην 3.1.

2. Σιγμοειδής συνάρτηση (Sigmoid Function) ή λογιστική συνάρτηση (Logistic Function). Η σιγμοειδής συνάρτηση, είναι η πλέον κοινή μορφή συνάρτησης ενεργοποίησης που χρησιμοποιείται για την κατασκευή νευρωνικών δικτύων και ορίζεται ως:

$$\phi(v) = \frac{1}{1 + e^{-av}}. \tag{3.4}$$

Η σιγμοειδής συνάρτηση μπορεί να λάβει τιμές από ένα συνεχές πεδίο τιμών, από 0 έως 1. Για πολύ μεγάλες αρνητικές τιμές εισόδου η έξοδος τείνει 0 και για πολύ μεγάλες θετικές τιμές τείνει στο 1. Η συγκεκριμένη συνάρτηση μοντελοποιεί ικανοποιητικά τη λειτουργία ενεργοποίησης ενός νευρώνα. Η κατάσταση μη-ενεργοποίησης κωδικοποιείται με το 0, ενώ η ενεργοποίηση κωδικοποιείται με 1. Το γεγονός πως η σιγμοειδής συνάρτηση είναι διαφορίσιμη είναι πολύ σημαντικό στη θεωρία των νευρωνικών δικτύων. Μία ιδιαίτερα ανεπιθύμητη ιδιότητα της συγκεκριμένης συνάρτησης είναι πως η κλίση της για πολύ μικρές ή πολύ μεγάλες τιμές εισόδου είναι σχεδόν μηδενική. Κατά την διαδικασία μάθησης με οπισθοδιάδοση (back propagation) η τοπική κλίση του νευρώνα θα είναι σχεδόν μηδενική με αποτέλεσμα το σφάλμα να μην προωθείται σωστά προς τα προηγούμενα στρώματα και να σταματήσει η διαδικασία της μάθησης.

3. Υπερβολική εφαπτομένη (Hyberbolic Tangent).

Η συνάρτηση υπερβολικής εφαπτομένης ορίζεται ως:

$$\phi(v) = \tanh(v). \tag{3.5}$$

Η συνάρτηση υπερβολικής εφαπτομένης δίνει έξοδο η οποία βρίσκεται στο διάστημα $[-1, 1]$. Το γεγονός πως οι τιμές εξόδου είναι κεντραρισμένες γύρω από το μηδέν κάνει την συνάρτηση υπερβολικής εφαπτομένης προτιμότερη από την σιγμοειδή συνάρτηση.

4. Ανορθωμένη γραμμική ή συνάρτηση ράμπας (Rectified Linear Unit - ReLu).

Η συνάρτηση Ράμπας έχει την εξής μορφή:

$$\phi(v) = \max(0, v). \tag{3.6}$$

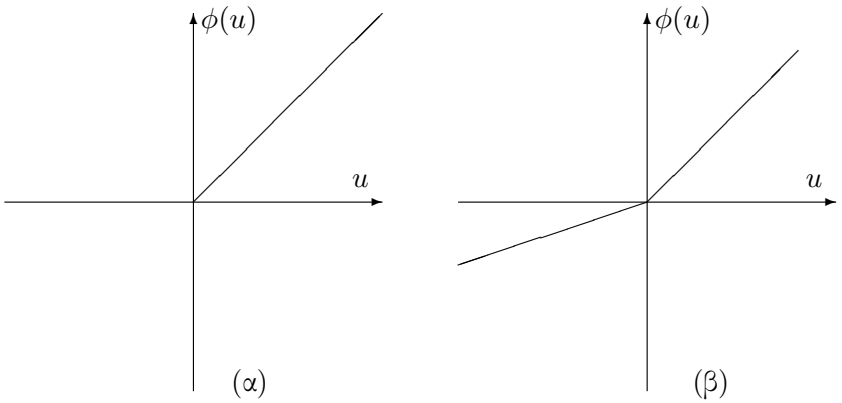
Η συγκεκριμένη συνάρτηση αποτελεί μία κοινή επιλογή για την εκπαίδευση συνελικτικών νευρωνικών δικτύων. Η έξοδος της είναι μη γραμμική και κορεσμένη γεγονός που επιταχύνει την εκπαίδευση με στοχαστική κατάβαση κατά το διάνυσμα κλίσεως της συνάρτησης

κόστους (stohastic gradient descent). Ωστόσο, ένα σημαντικό μειονέκτημα της συγκεκριμένης συνάρτησης είναι πως ορισμένες φορές μπορεί να οδηγήσει τους νευρώνες σε κάποιες τιμές βαρών, οι οποίες τους αποτρέπουν από το να ενεργοποιηθούν. Έτσι, αυτοί οι νευρώνες νεκρώνουν δηλαδή σταματάνε να ενεργοποιούνται και να μαθαίνουν.

5. Παραμετροποιημένη συνάρτηση ράμπας (Parametric Rectified Linear Unit - PReLU)

$$\phi(u) = \begin{cases} u, & u > 0 \\ ua, & \text{αλλού} \end{cases} \tag{3.7}$$

Η συγκεκριμένη συνάρτηση χρησιμοποιήθηκε επίσης για την εκπαίδευση συνελικτικών νευρωνικών δικτύων [7] και προσπαθεί να επιδιορθώσει το πρόβλημα των νεκρών νευρώνων πολλαπλασιάζοντας την έξοδο με μία μικρή τιμή a στην περίπτωση που η είσοδος είναι αρνητική. Στην περίπτωση που $a = 0$ η συνάρτηση μετατρέπεται στην συνάρτηση ReLU ενώ εαν το a πάρει μία μικρή και σταθερή τιμή (πχ $a = 0.01$) τότε η συνάρτηση ονομάζεται Leaky ReLU. Το Σχήμα 3.2 δείχνει τις συναρτήσεις ReLU και PReLU. Περισσότερα για την συγκεκριμένη συνάρτηση και τη λειτουργία της στα συνελικτικά νευρωνικά δίκτυα αναφέρονται στο Κεφάλαιο 5.

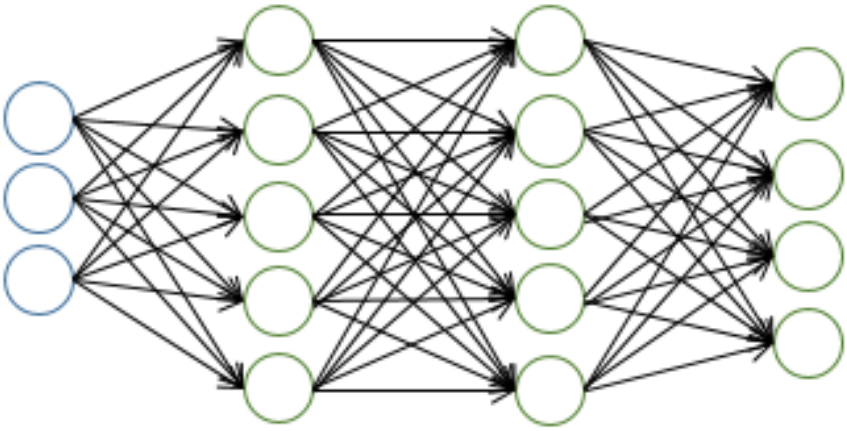


Σχήμα 3.2: (α) η συνάρτηση ReLU και (β) η συνάρτηση PReLU στα δεξιά

3.6 Αρχιτεκτονικές Νευρωνικών Δικτύων

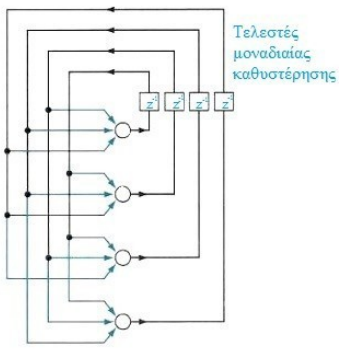
Συνδυάζοντας πολλούς νευρώνες μαζί κατασκευάζεται ένα νευρωνικό δίκτυο. Κάθε νευρώνας αντιπροσωπεύεται από ένα σύνολο γραμμικών συναπτικών συνδέσεων, μια εξωτερικά εφαρμοζόμενη πόλωση και μια πιθανώς μη γραμμική συνάρτηση ενεργοποίησης. Η πόλωση αντιπροσωπεύεται από μια συναπτική σύνδεση συνδεδεμένη σε μια είσοδο με σταθερή τιμή +1. Το σταθμισμένο άθροισμα των σημάτων εισόδου ορίζει το τοπικό πεδίο του νευρώνα. Ένα παράδειγμα μίας τέτοιας αρχιτεκτονικής φαίνεται στο Σχήμα 3.3. Το συγκεκριμένο δίκτυο είναι τύπου **πρόσθιας τροφοδότησης** (feedforward) και ονομάζεται Perceptron πολλών επιπέδων (Multilayer Perceptron). Τα ενδιάμεσα επίπεδα μεταξύ των επιπέδων εισόδου και εξόδου ονομάζονται **κρυμμένα επίπεδα** ενώ οι νευρώνες που τα απαρτίζουν ονομάζονται **κρυμμένοι νευρώνες**. Ο όρος κρυφός αναφέρεται στο γεγονός πως αυτό το μέρος το δικτύου δεν

είναι άμεσα ορατό από τα επίπεδα εξόδου και εισόδου. Τα σήματα εξόδου ενός επιπέδου χρησιμοποιούνται ως σήματα εισόδου για το επόμενο επίπεδο. Το νευρωνικό δίκτυο του Σχήματος 3.3 ονομάζεται **πλήρως συνδεδεμένο** υπό την έννοια ότι κάθε κόμβος σε κάθε επίπεδο συνδέεται με κάθε άλλο κόμβο του επόμενου (προς τα εμπρός) επιπέδου. Στην περίπτωση που λείπουν ορισμένες συνδέσεις το δίκτυο λέμε πως είναι **μερικώς συνδεδεμένο**.



Σχήμα 3.3: Ένα πλήρες συνδεδεμένο νευρωνικό δίκτυο πρόσθιας τροφοδότησης,
Πηγή : **https://bit.ly/2JMnK6i**

Μία δεύτερη κατηγορία αρχιτεκτονικής νευρωνικών δίκτύων αποτελούν τα **αναδρομικά νευρωνικά δίκτυα** (Recurrent Neural Networks - RNN). Ένα αναδρομικό νευρωνικό δίκτυο παρουσιάζεται στο Σχήμα 3.4 και διαφέρει από ένα δίκτυο πρόσθιας τροφοδότησης στο ότι περιλαμβάνει τουλάχιστον έναν **βρόχο ανάδρασης**. Το συγκεκριμένο δίκτυο αποτελείται από ένα επίπεδο νευρώνων στο οποίο κάθε νευρώνας τροφοδοτεί το σήμα εξόδου του στις εισόδους όλων των άλλων νευρώνων. Η ύπαρξη βρόχων ανάδρασης παίζει σημαντικό ρόλο κατά την διαδικασία της μάθησης. Συνήθως, οι βρόγχοι ανάδρασης απαιτούν την χρήση στοιχείων μοναδιαίας χρονικής καθυστέρησης τα οποία συμβολίζουμε με z^{-1} [8].



Σχήμα 3.4: Αναδρομικό νευρωνικό δίκτυο, Πηγή : **https://bit.ly/2MexmIL**

3.7 Μάθηση και Γενίκευση

3.7.1 Διαδικασίες Μάθησης

Ένα νευρωνικό δίκτυο μπορεί να εκπαιδευτεί πάνω σε ένα σύνολο παραδειγμάτων προκειμένου να μπορέσει να αναπαραστήσει τη γνώση για το περιβάλλον του. Διακρίνουμε δύο βασικές λειτουργίες ενός νευρωνικού δικτύου: την **ανάκληση** και την **εκπαίδευση**. Ανάκληση (recall) ονομάζεται η διαδικασία υπολογισμού ενός διανύσματος εξόδου για συγκεκριμένες τιμές εισόδου και βαρών. Εκπαίδευση (training) ονομάζεται η διαδικασία της τροποποίησης των τιμών των βαρών του δικτύου με σκοπό δοθέντος ενός διανύσματος εισόδου να παραχθεί το επιθυμητό διάνυσμα εξόδου. Η ικανότητα ενός νευρωνικού δικτύου να εκτιμά με επιτυχία την κλάση ενός άγνωστου προτύπου ονομάζεται **γενίκευση**. Μπορούμε να κατηγοριοποιήσουμε τις διαδικασίες μάθησης σε δύο βασικές κατηγορίες:

1. Μάθηση με επίβλεψη (supervised learning). Η γνώση του περιβάλλοντος αντιπροσωπεύεται από ένα σύνολο παραδειγμάτων εισόδου-εξόδου. Κατά την μάθηση με επίβλεψη παρέχεται στο νευρωνικό δίκτυο μια επιθυμητή απόκριση για ένα συγκεκριμένο διάνυσμα εκπαίδευσης [9]. Οι παράμετροι του δικτύου προσαρμόζονται από την επιρροή του διανύσματος εκπαίδευσης αλλά και του σήματος σφάλματος. Το **σήμα σφάλματος** ορίζεται ως η διαφορά μεταξύ της επιθυμητής απόκρισης και της πραγματικής απόκρισης του δικτύου. Τα συναπτικά βάρη του δικτύου αντιπροσωπεύουν τη λειτουργία μιας μακροπρόθεσμης μνήμης καθώς κωδικοποιούν και αποθηκεύουν την γνώση του περιβάλλοντος.

2. Μάθηση χωρίς επίβλεψη (unsupervised learning). Κατά τη διαδικασία της μάθησης χωρίς επίβλεψη δεν υπάρχουν χαρακτηριστικά παραδείγματα της λειτουργίας που πρέπει να μάθει ένα δίκτυο. Αντιθέτως, οι ελεύθερες παράμετροι του δικτύου βελτιστοποιούνται σε σχέση με κάποιο μέτρο που βαθμολογεί την ποιότητα της αναπαράστασης των δεδομένων. Το νευρωνικό δίκτυο είναι σε θέση από μόνο του να σχηματίσει εσωτερικές αναπαραστάσεις για την κωδικοποίηση των χαρακτηριστικών της εισόδου και να δημιουργήσει μόνο του τις αντίστοιχες κλάσεις.

Τα παραδείγματα εκπαίδευσης μπορούν να είναι είτε **χαρακτηρισμένα** (labeled) είτε **μη χαρακτηρισμένα** (unlabeled). Στα χαρακτηρισμένα παραδείγματα, κάθε παράδειγμα που αναπαριστά ένα σήμα εισόδου συσχετίζεται με μια αντίστοιχη **επιθυμητή απόκριση** (έξοδο). Για παράδειγμα, στο πρόβλημα της ανίχνευσης ομιλίας τα χαρακτηρισμένα δεδομένα θα έχουν την ετικέτα 1 για την περίπτωση που το δείγμα ομιλίας ανήκει σε έναν άνθρωπο με Parkinson και 0 για την περίπτωση που το δείγμα ομιλίας ανήκει σε έναν άνθρωπο που δεν έχει Parkinson. Ένα σύνολο ζευγών εισόδου-εξόδου, όπου κάθε ζεύγος περιλαμβάνει το αντίστοιχο σήμα εισόδου και την αντίστοιχη επιθυμητή απόκριση αποκαλείται **σύνολο δεδομένων εκπαίδευσης**. Οι όμοιες είσοδοι που προέρχονται από όμοιες κλάσεις θα πρέπει συνήθως να παράγουν όμοιες αναπαραστάσεις μέσα στο δίκτυο και συνεπώς να ταξινομούνται ως ανήκουσες στην ίδια κλάση.

3.7.2 Γραμμικοί Ταξινομητές (Linear Classifiers)

Το πρόβλημα της ταξινόμησης και της ανίχνευσης π.χ αντικειμένων σε εικόνες αποτελεί ένα πρόβλημα επιβλεπόμενης μάθησης. Θα συμβολίσουμε ως $D = \{\mathbf{x}^{(i)}, \mathbf{d}^{(i)}\}$ το σύνολο εκπαίδευσης. Κάθε στοιχείο του συνόλου εκπαίδευσης είναι ένα ζεύγος ενός διανύσματος εισόδου $\mathbf{x}^{(i)} \in \mathbb{R}^m$ και ενός διανύσματος στόχου $\mathbf{d}^{(i)} \in \mathbb{R}^n$. Το πρόβλημα της ανίχνευσης ομιλίας του Parkinson αποτελείται από δύο κλάσεις (έχει Parkinson, δεν έχει Parkinson) επομένως $d^{(i)} \in \{0, 1\}$. Εάν είχαμε n κλάσεις τότε θα ίσχυε ότι $d \in \{1, \ldots, n\}$ και θα είχε τη μορφή $[0, \ldots, 1, \ldots, 0]^T$ όπου το 1 βρίσκεται στη θέση που αντιστοιχεί στην αντίστοιχη κλάση του $\mathbf{x}^{(i)}$.

Το σκεπτικό της κατασκευής μοντέλων χρησιμοποιείται σε κάθε επιστημονικό κλάδο που ασχολείται με τη στατιστική ανάλυση δεδομένων. Ζητούμενο μπορεί να θεωρηθεί ως η εύρεση των σχέσεων μεταξύ ενός συνόλου τυχαίων μεταβλητών. Η εξαρτώμενη μεταβλητή στην προκειμένη περίπτωση θα είναι η επιθυμητή απόκριση d. Το πρόβλημα της ταξινόμησης έχει πολλά κοινά με την προσέγγιση συναρτήσεων. Θεωρούμε πως υπάρχει μια αντιστοίχιση εισόδου εξόδου $d = f(x)$ και προσπαθούμε να προσεγγίσουμε την άγνωστη διανυσματική συνάρτηση $f(\cdot)$. Στο πρόβλημα της ταξινόμησης, η διανυσματική συνάρτηση θα πρέπει να δέχεται ως είσοδο ένα επεξεργασμένο διάνυσμα ομιλίας και να έχει ως έξοδο μία τιμή που αντιπροσωπεύει την κλάση της εισόδου. Αναζητούμε δηλαδή n συναρτήσεις της μορφής $f : \mathbb{R}^m \to \mathbb{R}$ οι οποίες αντιστοιχούν τις τιμές των δυανισμάτων στην κατηγορία την οποία ανήκει το σήμα μας. Η πιο απλή περιγραφή θα ήταν ένα γραμμικό μοντέλο της μορφής:

$$f(\mathbf{x};\mathbf{w}) = \sum_{j=0}^{m} w^{(j)} x^{(j)} = \mathbf{w}^T \mathbf{x} \tag{3.8}$$

όπου $\mathbf{x} = [1, x_1, \ldots, x_m]^T$ και $\mathbf{w} = [b, w_0, w_1, \ldots, w_m]^T$ τα επαυξημένα διανύσματα εισόδου και βαρών. Ορίζοντας n τέτοιες συναρτήσεις κατασκευάζουμε n γραμμικά μοντέλα:

$$f_i(\mathbf{x}; w_i) = w_i^T \mathbf{x} \tag{3.9}$$

Η διανυσματική συνάρτηση $\mathbf{f} = [f_1, \ldots, f_n] : \mathbb{R}^m \to \mathbb{R}^n$ που περιγράφει το μοντέλο θα έχει τη μορφή:

$$f(x; W) = Wx \tag{3.10}$$

όπου

$$W = \begin{bmatrix} w_1^T \\ \vdots \\ w_k^T \end{bmatrix} \tag{3.11}$$

δηλαδή ο πίνακας W είναι ο συγκεντρωτικός πίνακας των βαρών μεγέθους $n \times m$. Όπως περιγράφεται στο Κεφάλαιο 4 τα συνελικτικά νευρωνικά δίκτυα τα οποία μας ενδιαφέρουν στη συγκεκριμένη εργασία έχουν ως βάση τους τέτοιους γραμμικούς ταξινομητές.

3.7.3 Συναρτήσεις Κόστους

Κατά την διαδικασία της μάθησης με επίβλεψη είναι πολύ σημαντικό να χαρακτηριστεί η απόδοση του δικτύου. Η αξιολόγηση του διανύσματος των συναπτικών βαρών $\mathbf{w}$ κρίνεται μέσω μίας **συνάρτησης κόστους** $J(\mathbf{w})$ η οποία ονομάζεται και **συνάρτηση εμπειρικού ρίσκου** (empirical risk). Η συνάρτηση κόστους εκτιμά την ακρίβεια (accuracy) και εκφράζει την απόκλιση της πρόβλεψης από την επιθυμητή έξοδο. Η διαδικασία της μάθησης πραγματοποιείται μέσω της ελαχιστοποίησης μίας συνάρτησης κόστους:

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} L(f(\mathbf{x}^{(i)};\mathbf{w}), d^{(i)}) + \lambda R(\mathbf{w}) \tag{3.12}$$

όπου το N αφορά το συνολικό αριθμό των δειγμάτων εκπαίδευσης, η $L(\cdot)$ είναι η συνάρτηση απώλειας (Loss Function). Η παραπάνω έκφραση περιέχει τον όρο της $R(\mathbf{w})$ ο οποίος κλιμακώνεται με μία σταθερά λ. Η σταθερά αυτή είναι γνωστή ως **παράμετρος εξομάλυνσης** (regularization term) και ελέγχει την επιρροή του όρου εξομάλυνσης. Ένα κλασικό παράδειγμα συνάρτησης σφάλματος αποτελεί το **Μέσο Τετραγωνικό Σφάλμα** (Mean Square Error):

$$J_{mse}(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}[d^{(i)} - f(\mathbf{x}^{(i)};\mathbf{w})]^2 + \lambda R(\mathbf{w}) \tag{3.13}$$

Ένα άλλο κριτήριο κόστους που χρησιμοποιείται πολύ συχνά είναι το κόστος τύπου softmax. Η συγκεκριμένη συνάρτηση απεικονίζει ένα διάνυσμα εισόδου $\mathbf{x}$ σε ένα διάνυσμα με τιμές οι οποίες βρίσκονται στο διάστημα $[0, 1]$. Δοθέντος ενός διανύσματος $\mathbf{x}^{(i)}$ θα θέλαμε να υπολογίσουμε την πιθανότητα $P(\mathbf{y}^{(i)} = \mathbf{d}^{(j)}|\ \mathbf{x}^{(i)})$ δηλαδή τη δεσμευμένη πιθανότητα το διάνυσμα εξόδου $\mathbf{y}$ να ισούται με το διάνυσμα στόχων $\mathbf{d}$ δεδομένου διανύσματος εισόδου $\mathbf{x}^{(i)}$. Η συνάρτηση που θα προσεγγίσει το δίκτυο θα πρέπει να έχει έξοδο ένα διάνυσμα n διαστάσεων όσες και οι κλάσεις του προβλήματος του οποίου οι τιμές έχουν άθροισμα 1 και έχει τη μορφή:

$$f(\mathbf{x}^{(i)};\mathbf{w}_) = \begin{bmatrix} P(\mathbf{y}^{(i)} = 1|\ \mathbf{x}^{(i)};\mathbf{w}_1) \\ P(\mathbf{y}^{(i)} = 2|\ \mathbf{x}^{(i)};\mathbf{w}_2) \\ \vdots \\ P(\mathbf{y}^{(i)} = n|\ \mathbf{x}^{(i)};\mathbf{w}_n) \end{bmatrix} = \frac{1}{\sum_{j=1}^{n} e^{\mathbf{w}_j^T \mathbf{x}_{(i)}}} \begin{bmatrix} e^{\mathbf{w}_1^T\ \mathbf{x}_{(i)}} \\ e^{\mathbf{w}_2^T\ \mathbf{x}_{(i)}} \\ \vdots \\ e^{\mathbf{w}_n^T\ \mathbf{x}_{(i)}} \end{bmatrix} \tag{3.14}$$

όπου $\mathbf{w}_1, \ldots, \mathbf{w}_n$ οι εσωτερικές παράμετροι του μοντέλου. Ο παρανομαστής του κλάσματος κανονικοποιεί την συνάρτηση $f(\cdot)$ μετατρέποντας την σε πιθανότητα διακριτής τυχαίας μεταβλητής έτσι ώστε το άθροισμα όλων των στοιχείων να είναι 1. Η πιθανότητα δηλαδή η έξοδος να ανήκει στην j-οστή κλάση δεδομένου ενός διανύσματος $\mathbf{x}$ είναι:

$$L(f(x^{(i)};w_j)) = P(\mathbf{y}^{(i)} = j|\ \mathbf{x}^{(i)};\mathbf{w}_j) = \frac{e^{\mathbf{w}_j^T \mathbf{x}_{(i)}}}{\sum_{j=1}^{n} e^{\mathbf{w}_j^T \mathbf{x}_{(i)}}} \tag{3.15}$$

Το συνολικό κόστος για όλα τα παραδείγματα N μπορεί να εκφρασθεί ως:

$$J(\mathbf{w}) = \frac{-1}{N}\sum_{i=1}^{N} \log \frac{e^{\mathbf{w}_j^T \mathbf{x}_{(i)}}}{\sum_{j=1}^{n} e^{\mathbf{w}_j^T \mathbf{x}_{(i)}}} + \lambda R(\mathbf{w}) \tag{3.16}$$

Η ταξινόμηση με αυτήν την συνάρτηση κόστους λέγεται **πολυωνυμική λογιστική παλινδρόμηση** (Multinomial Logistic Regression). Στην περίπτωση που έχουμε δύο κλάσεις ονομάζεται λογιστική παλινδρόμηση (Logistic Regression).

3.8 Βελτιστοποίηση με τη μέθοδο της κατάβασης του διανύσματος κλήσης της συνάρτησης κόστους (Gradient descent)

Υποθέτουμε ότι μας δίνεται μία συνάρτηση κόστους $J(\mathbf{w})$ η οποία είναι συνεχώς διαφορίσιμη συνάρτηση κάποιου άγνωστου διανύσματος βαρών $\mathbf{w} = [w_1, \ldots, w_m]^T$. Η διαδικασία της μάθησης του νευρωνικού δικτύου μπορεί να θεωρηθεί ως μία διαδικασία ελαχιστοποίησης της συνάρτησης κόστους ως προς το διάνυσμα βαρών $\mathbf{w}$. Μία πολύ δημοφιλής μέθοδος βελτιστοποίησης συναρτήσεων m μεταβλητών είναι η μέθοδος κατάβασης του διανύσματος κλίσης της συνάρτησης κόστους γνωστη και ως μέθοδος *κατάβασης δυναμικού*. Η αναγκαία συνθήκη για το βέλτιστο $\mathbf{w}^*$ είναι:

$$\nabla J(\mathbf{w}^*) = \mathbf{0} \tag{3.17}$$

όπου:

- ∇ είναι ο *τελεστής κλίσης* $\nabla = [\frac{\partial}{\partial w_1}, \ldots, \frac{\partial}{\partial w_m}]^T$,
- $\nabla J(\mathbf{w})$ είναι το *διάνυσμα κλίσης* της συνάρτησης κόστους $\nabla J(\mathbf{w}) = [\frac{\partial J}{\partial w_1}, \ldots, \frac{\partial J}{\partial w_m}]^T$
- $\mathbf{w}^* = [w_1^*, \ldots, w_m^*]$ το *τοπικό ελάχιστο* στο $\Re^m$.

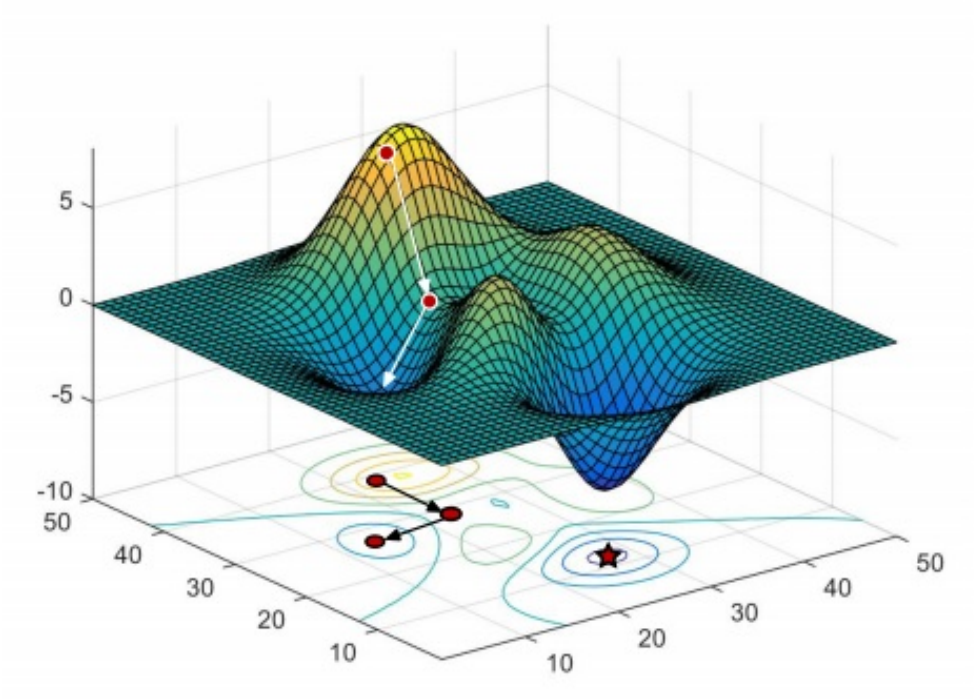
Ο αλγόριθμος κατάβασης δυναμικού ξεκινάει από ένα τυχαίο σημείο $w(0)$ και προχωράει με διαδοχικές επισκέψεις σε άλλα σημεία $w(1), w(2), \ldots$ τέτοια ώστε η συνάρτηση κόστους $J(\mathbf{w})$ να μειώνεται σε κάθε επανάληψη όπως υποδεικνύει η σχέση:

$$J(\mathbf{w}(k+1)) < J(\mathbf{w}(k)) \tag{3.18}$$

όπου $\mathbf{w}(k)$ είναι η τιμή του διανύσματος βαρών κατά το επαναληπτικό βήμα k. Οι διαδοχικές προσαρμογές στο διάνυσμα βαρών $\mathbf{w}$ θα πρέπει να είναι προς την κατεύθυνση της πλέον απότομης κατάβασης, δηλαδή σε κατεύθυνση αντίθετη προς το διάνυσμα κλίσης $\nabla J(\mathbf{w})$. Κατά τη μετάβαση από το επαναληπτικό βήμα k στο επαναληπτικό βήμα $k+1$ ο κανόνας κατάβασης δυναμικού για το βάρος i είναι:

$$w_i(k+1) = w_i(k) - \beta \frac{\partial J}{\partial w_i}\Big|_{w_i(k)} \tag{3.19}$$

Η παράμετρος β ονομάζεται *ρυθμός μάθησης* και έχει μεγάλη επίδραση στην σύγκλιση του αλγορίθμου. Όταν το β είναι αρκετά μικρό η τροχιά του διανύσματος $\mathbf{w}$ ακολουθεί μία ομαλή διαδρομή στο $\Re^m$. Αντίθετα, όταν το β λάβει μία μεγάλη τιμή η τροχιά του διανύσματος $\mathbf{w}$ ακολουθεί μία διαδρομή σχήματος zigzag ή στην χειρότερη περίπτωση ο αλγόριθμος αποκλίνει από την λύση. Η σωστή τιμή για το β είναι δύσκολο να προσδιοριστεί και εξαρτάται κάθε φορά από τη φύση και το είδος της συνάρτησης προς βελτιστοποίηση. Συνήθως, χρησιμοποιούνται τιμές μικρότερες έως πολύ μικρότερες της μονάδας οι οποίες εκτιμώνται μέσω μεθόδων δοκιμής και σφάλματος.



Σχήμα 3.5: Η μέθοδος κατάβασης για μία συνάρτηση δύο διαστάσεων,
Πηγή : **https://bit.ly/2Jvaovd**

Μία παραλλαγή της μεθόδου κατάβασης δυναμικού αποτελεί η *Στοχαστική Κατάβαση Δυναμικού* (Stohastic Gradient Descent) η οποία χρησιμοποιεί ένα υποσύνολο του συνόλου εκπαίδευσης προκειμένου να υπολογίσει την κλίση της συνάρτησης κόστους $\nabla J(\mathbf{w})$. Ο αριθμός των δειγμάτων εκπαίδευσης που χρησιμοποιούνται σε μία επανάληψη του αλγορίθμου ονομάζεται *μέγεθος παρτίδας* (batch size). Ο στοχαστικός χαρακτήρας της έχει το επιθυμητό αποτέλεσμα ότι μειώνει την πιθανότητα η διαδικασία μάθησης να παγιδευτεί σ΄ ένα τοπικό ελάχιστο.

3.9 Αλγόριθμος Back Propagation (BK)

Ο αλγόριθμος Back Propagation αναπτύχθηκε στα μέσα της δεκαετίας του 80 και αποτέλεσε ορόσημο στην ιστορία και την εξέλιξη των νευρωνικών δικτύων. Ο αλγόριθμος Back Propagation περιλαμβάνει δύο διαφορετικές φάσεις:

1. Στην πρώτη φάση το σήμα εισόδου διαδίδεται στο δίκτυο προς τα εμπρός μέχρι να φτάσει στο τελευταίο επίπεδο εξόδου. Τα συναπτικά βάρη του δικτύου είναι σταθερά.

2. Η δεύτερη φάση του αλγορίθμου εξελίσσεται προς τα πίσω, από το τελευταίο επίπεδο εξόδου προς το επίπεδο εισόδου. Ένα σήμα σφάλματος παράγεται ως η διαφορά της πραγματικής εξόδου του δικτύου με μία επιθυμητή έξοδο και διαδίδεται επίπεδο προς επίπεδο με κατεύθυνση προς τα πίσω. Τα συναπτικά βάρη του δικτύου προσαρμόζονται και μεταβάλλονται.

Θα θεωρήσουμε ένα νευρωνικό δίκτυο Perceptron πολλών επιπέδων με L στρώματα, $n = N(0)$

εισόδους και $m = N(L)$ εξόδους. Για μία σειρά από P διανύσματα εισόδου επιθυμούμε οι έξοδοι να επιτύχουν τιμές που δίνονται από αντίστοιχα P διανύσματα στόχων. Συνεπώς συμβολίζουμε:

- $\mathbf{x}^{(p)} = [x_1^{(p)}, \ldots, x_n^{(p)}]$ το p-οστό διάνυσμα εισόδου
- $\mathbf{y}^{(p)} = [y_1^{(p)}, \ldots, y_m^{(p)}]$ το p-οστό διάνυσμα εξόδου
- $\mathbf{d}^{(p)} = [d_1^{(p)}, \ldots, d_m^{(p)}]$ το p-οστό διάνυσμα στόχων

Το σύνολο όλων των διανυσμάτων $\{\mathbf{x}^{(1)}, \mathbf{d}^{(1)}\}\{\mathbf{x}^{(2)}, \mathbf{d}^{(2)}\}, \ldots, \{\mathbf{x}^{(P)}, \mathbf{d}^{(P)}\}$ αποτελεί το σύνολο των δεδομένων εκπαίδευσης. Ιδανικά, θα θέλαμε όλα τα διανύσματα εξόδων να ταυτιστούν με τα διανύσματα των στόχων δηλαδή $\mathbf{y}^{(t)} = \mathbf{d}^{(t)}$ για $t = 1 \ldots P$. Συνήθως, αρκούμαστε στη βέλτιστη προσέγγιση αυτής της κατάστασης βελτιστοποιώντας ένα κριτήριο κόστους J. Στην περίπτωση που χρησιμοποιήσουμε ως κριτήριο κόστους το μέσο τετραγωνικό σφάλμα θα ισχύει:

$$J = \frac{1}{P}\sum_{p=1}^{P}\|\mathbf{d}^{(p)} - \mathbf{y}^{(p)}\|^2 = \frac{1}{P}\sum_{p=1}^{P}\sum_{i=1}^{n}[d_i^{(p)} - y_i^{(p)}]^2 \tag{3.20}$$

Προκειμένου να ελαχιστοποιηθεί το J θα πρέπει να μεταβάλλουμε τις τιμές των συναπτικών βαρών w_{ij}. Ο συμβολισμός w_{ij} αναφέρεται στο i-οστό βάρος του νευρώνα j. Το διάνυσμα των εισόδων $\mathbf{x}^{(p)}$ καθώς και το διάνυσμα των στόχων $\mathbf{d}^{(p)}$ παραμένουν σταθερά. Θα συμβολίσουμε επίσης:

- $w_{ij}(l,k)$ το συναπτικό βάρος του νευρώνα j ο οποίος ανήκει στο επίπεδο $l-1$ κατά την χρονική στιγμή k.

Σύμφωνα με την μέθοδο κατάβασης η μεταβολή της παραμέτρου w_{ij} ως προς το χρόνο t είναι:

$$\frac{dw_{ij}}{dt} = -\frac{\partial J}{\partial w_{ij}} \tag{3.21}$$

Μία διακριτή έκδοση της 3.17 μπορεί να εκφρασθεί ως:

$$w_{ij}(l,k+1) - w_{ij}(l,k) = -\beta\frac{\partial J}{\partial w_{ij}(l,k)} \tag{3.22}$$

όπου το $w_{ij}(l,k+1) - w_{ij}(l,k)$ αναφέρεται στη διαφορά της τιμής του συναπτικού βάρους i του νευρώνα j του επιπέδου $l-1$ κατά τις χρονικές στιγμές k και $k+1$. Η έξοδος του νευρώνα i του στρώματος l μπορεί να εκφρασθεί από τις εξισώσεις:

$$a_i^{(k)}(l) = \phi(v_i^{(k)}(l)) \tag{3.23}$$

$$v_i^{(k)}(l) = \sum_{\xi=1}^{N(l-1)} a_\xi^{(k)} w_{i\xi}(l,k)(l-1) + w_{i0}(l,k) \tag{3.24}$$

όπου το $v_i^{(k)}$ ονομάζεται **τιμή ενεργοποίησης** του νευρώνα i κατά την χρονική στιγμή k. Η δικτυακή διέγερση ισούται με το άθροισμα των διεγέρσεων των νευρώνων του προηγούμενου στρώματος τη χρονική στιγμή k, $a_\xi^{(k)}(l-1)$ συνδυασμένων με τα συναπτικά βάρη $w_{i\xi}(l,k)$.

Θα εκφράσουμε την παράγωγο της συνάρτησης κόστους ως προς τη δικτυακή διέγερση του νευρώνα i:

$$\delta_i^k(l) = -\frac{\partial J}{\partial v_i^{(k)}(l)} \tag{3.25}$$

Η παράμετρος δ_i αφορά το σφάλμα του νευρώνα i. Μπορούμε να γράψουμε πως:

$$\frac{\partial J}{\partial w_{ij}(l,k)} = \frac{\partial J}{\partial v_i^{(k)}(l)} \frac{\partial v_i^{(k)}(l)}{\partial w_{ij}(l,k)} = -\delta_i^{(k)}(l) \frac{\partial v_i^{(k)}(l)}{\partial w_{ij}(l,k)} \tag{3.26}$$

Προκύπτει ως ο τελευταίος όρος της παραπάνω σχέσης, δηλαδή η παράγωγος $\frac{\partial v_i^{(k)}(l)}{\partial w_{ij}(l,k)}$ μπορεί να υπολογισθεί ως εξής:

- $\frac{\partial v_i^{(k)}(l)}{\partial w_{ij}(l,k)} = a_j^{(k)}(l-1)$ για $j \neq 0$
- $\frac{\partial v_i^{(k)}(l)}{\partial w_{i0}(l,k)} = 1$ για $j = 0$

Τελικά με βάση την σχέση 3.22 η παράγωγος του κόστους ως προς το συναπτικό βάρος $w_{ij}(l,k)$ είναι:

$$\frac{\partial J}{\partial w_{ij}(l,k)} = \begin{cases} -\delta_i^{(k)}(l) a_j^{(k)}(l-1), & j = 1, \ldots, N(l-1) \\ -\delta_i^{(k)}(l), & j = 0 \end{cases} \tag{3.27}$$

Έστω:

$$\phi(v) = \begin{cases} 1 & , v \geq 0 \\ 0 & , v < 0 \end{cases} \tag{3.28}$$

Η προηγούμενη σχέση μπορεί να απλοποιηθεί αρκετά εάν ορίσουμε $a_0^{(k)}(l) = 1$ για όλα τα στρώματα $l = 1, 2, \ldots, L$:

$$\frac{\partial J}{\partial w_{ij}(l,k)} = -\delta_i^{(k)} a_j^{(k)}(l-1), \qquad \text{για } j = 0, 1, 2, \ldots, N(l-1) \tag{3.29}$$

Ο υπολογισμός των $\delta_i^{(k)}$ θα πρέπει να γίνει ξεκινώντας από το στρώμα εξόδου προς τα πίσω. Διακρίνουμε τις εξής περιπτώσεις:

- Ένας νευρώνας ανήκει στο στρώμα εξόδου L. Η παράγωγος του κόστους ως προς τη διέγερση $v_i^{(k)}(L)$ είναι:

$$\delta_i^{(k)}(L) = -\frac{\partial J}{\partial a_i^{(k)}(L)} \frac{\partial a_i^{(k)}(L)}{\partial v_i^{(k)}(L)} = -\frac{\partial J}{\partial a_i^{(k)}(L)} \frac{\partial \phi(v_i^{(k)}(L))}{\partial v_i^{(k)}(L)} \tag{3.30}$$

  Τελικά καταλήγουμε πως $\delta_i^{(k)}(L) = (d_i^{(k)} - y_i^{(k)})\phi'(v_i^{(k)}(L))$, δηλαδή το σφάλμα $\delta_i^{(k)}(L)$ είναι η διαφορά της τιμής του στόχου $d_i^{(k)}$ από την έξοδο του νευρώνα $y_i^{(k)}$ επί την παράγωγο ϕ' της συνάρτησης ενεργοποίησης της τιμής ενεργοποίησης $v_i^{(k)}(L)$.

- Ένας νευρώνας ανήκει σε οποιοδήποτε στρώμα $l = 1, 2, \ldots, L-1$. Σε κάθε άλλη περίπτωση δηλαδή το σφάλμα $\delta_i^{(k)}(L)$ ισούται με:

$$\delta_i^{(k)}(L) = -\frac{\partial J}{\partial v_i^{(k)}(l)} = -\sum_{m=1}^{N(l+1)} \frac{\partial J}{\partial v_m^{(k)}(l+1)} \frac{\partial v_m^{(k)}(l+1)}{\partial a_i^{(k)}(l)} \frac{\partial a_i^{(k)}(l)}{\partial v_i^{(k)}(l)} \tag{3.31}$$

Τελικά καταλήγουμε πως $\delta_i^{(k)}(L) = \sum_{m=1}^{N(l+1)} \delta_m^{(k)}(l+1) w_{mi}^{(k)}(l+1) \phi'(v_i^{(k)}(l))$. Συνεπώς, για τον υπολογισμό του σφάλματος $\delta_i(l)$ του επιπέδου l χρησιμοποιούνται όλα τα σφάλματα του επιπέδου $l+1$. Η τελευταία ιδιότητα περιγράφει την προώθηση των σφαλμάτων προς τα πίσω δηλαδή προς την κατεύθυνση του πρώτου επιπέδου. Συνοψίζοντας, ο αλγόριθμος back propagation αναλύεται στη παρακάτω μορφή:

Algorithm 1: Back-Propagation

Είσοδος: P ζεύγη διανυσμάτων εισόδων στόχων $\{\mathbf{x}^{(p)}, \mathbf{d}^{(p)}\}$

Έξοδος: Τα εκπαιδευμένα βάρη $w_{ij}(l)$

Αρχικοποίηση των βαρών $w_{ij}(l)$ σε τυχαίες τιμές για κάθε στώμα l. Το βάρος $w_{i0}(l)$ αντιστοιχεί στο κατώφλι του νευρώνα i του στρώματος l

$n \leftarrow 1$

Επανάλαβε

Για κάθε *πρότυπο* $p = 1 \ldots P$ **κάνε**

Ανάκληση:

- Υπολογισμός των εξόδων $a_i(L)$ για τα στρώματα $L = 1 \ldots L$

Υπολογισμός δ:

- Υπολογισμός των σφαλμάτων $\delta_i(L) = \phi'(v_i(L)(d_i - y_i)$ του στρώματος L
- Υπολογισμός των σφαλμάτων $\delta_i(l) = \phi'(v_i(L) \sum_{m=1}^{N(l+1)} w_{m1}(l+1)\delta_m(l+1)$ του για τα στρώματα $l = 1 \ldots L$

Ενημέρωση Βαρών:

- Ενημέρωση $w_{ij}(l, n+1) = w_{ij}(l, n) + \beta\delta_i(l)a_j(l-1)$ για $j = 0, \ldots, N \ldots (l)$ και $l = 1, \ldots, L$

$n = n + 1$

Μέχρι *Μέχρι το συνολικό σφάλμα J να είναι μικρότερο από κάποιο κατώφλι ε·*

Κεφάλαιο 4

Συνελικτικά Νευρωνικά Δίκτυα

Τα συνελικτικά νευρωνικά δίκτυα (ΣΝΔ) είναι αρχιτεκτονικές τεχνητών νευρωνικών δικτύων πρόσθιας τροφοδότησης (feed-forward networks) τα οποία έχουν ευρεία εφαρμογή σε πολλά προβλήματα ταξινόμησης π.χ. αναγνώρισης εικόνας καθώς και επεξεργασία φυσικής γλώσσας. Ο αρχικός σχεδιασμός τους βασίστηκε πάνω σε κάποιες πληροφορίες που προέκυψαν από μελέτες πάνω στο οπτικό σύστημα των ζωντανών οργανισμών. Από το 1962 η εργασία των Hubel και Wiesel, [10], έδειξε πως τα κύτταρα του οπτικού φλοιού του εγκεφάλου των ζώων είναι οργανωμένα σε σύνθετες ιεραρχικές δομές και είναι ευαίσθητα σε μικρές υπό-περιοχές του οπτικού πεδίου οι οποίες ονομάζονται (receptive fields). Κατά αυτόν τον τρόπο τα οπτικά κύτταρα λειτουργούν σαν χωρικά φίλτρα πάνω στο οπτικό σήμα και εκμεταλλεύονται την ισχυρή τοπική συσχέτιση των περιοχών που υπάρχουν στις φυσικές εικόνες. Τα ΣΝΔ εφαρμόζουν αντίστοιχα φίλτρα πάνω στην είσοδο χρησιμοποιώντας συνελίξεις οι οποίες εκτελούνται σε πολλά διαφορετικά επίπεδα. Οι περιοχές σάρωσης, δηλαδή τα δεκτικά πεδία των φίλτρων είναι αλληλεπικαλυπτόμενα προκειμένου να εξαχθούν ομαλές αναπαραστάσεις της εικόνας εισόδου και να αξιοποιηθεί το γεγονός πως στις φυσικές εικόνες υπάρχουν πανομοιότυπα χαρακτηριστικά (όπως για παράδειγμα ακμές) σε διάφορα σημεία στο χώρο.

4.1 Βασικές αρχές λειτουργίας των Συνελικτικών Νευρωνικών Δικτύων

Ένα ΣΝΔ είναι ένα Perceptron πολλών επιπέδων σχεδιασμένο έτσι ώστε να μπορεί να αναγνωρίζει σχήματα δύο διαστάσεων με υψηλό βαθμό αναισθησίας στην μετατόπιση, την κλιμάκωση, την στρέβλωση και άλλες μορφές παραμόρφωσης. Σύμφωνα με τους LeCun και Bengio [11] η συγκεκριμένη εργασία διδάσκεται με επιβλεπόμενο τρόπο μέσω, ενός δικτύου του οποίου η δομή περιλαμβάνει τις ακόλουθες μορφές περιορισμών:

1. Εξαγωγή Χαρακτηριστικών. Κάθε νευρώνας ενός συνελικτικού νευρωνικού δικτύου λαμβάνει είσοδο από ένα τοπικό **δεκτικό πεδίο** του προηγούμενου επιπέδου με αποτέλεσμα να εξάγει τοπικά χαρακτηριστικά. Ιδιαίτερη σημασία έχει το γεγονός πως εφόσον εξαχθεί ένα χαρακτηριστικό από τα δεδομένα, η ακριβής του θέση δεν έχει πλέον τόσο μεγάλη σημασία, εφόσον διατηρείται (προσεγγιστικά) η σχετική του θέση ως προς άλλα χαρα-

κτηριστικά.

2. Αντιστοίχιση των χαρακτηριστικών. Κάθε επίπεδο ενός συνελικτικού νευρωνικού δικτύου αποτελείται από πολλούς **χάρτες χαρακτηριστικών** (feature maps). Σε κάθε χάρτη χαρακτηριστικών, όλοι οι νευρώνες μοιράζονται το ίδιο σύνολο συναπτικών βαρών. Η συγκεκριμένη ιδιότητα προσφέρει δύο πολύ σημαντικά πλεονεκτήματα:

   - αμεταβλητότητα στη μετατόπιση (shift invariance)
   - μείωση του αριθμού των ελεύθερων παραμέτρων

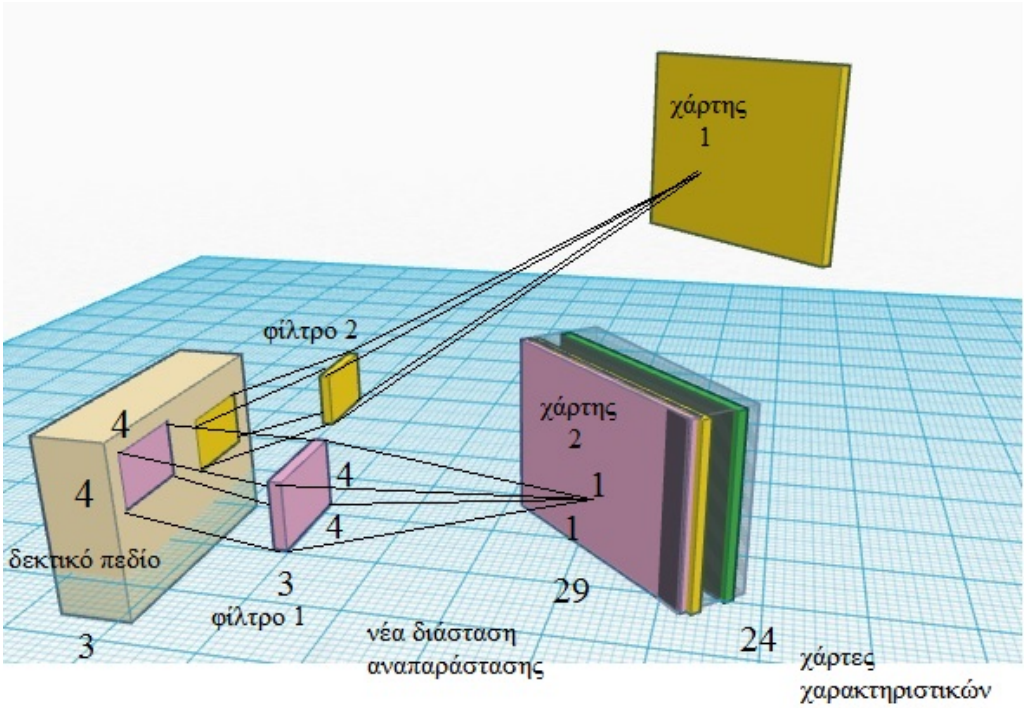
4.2 Τύποι Επιπέδων ενός ΣΝΔ

Ένα ΣΝΔ αποτελείται κάποιους βασικούς τύπους επιπέδων/στρωμάτων τα οποία μπορούν να κατηγοριοποιηθούν ανάλογα με τη λειτουργία που εκτελούν. Τα κυριότερα από αυτά είναι το συνελικτικό επίπεδο (convolutional layer), το επίπεδο δειγματοληψίας (pooling layer) και το πλήρως συνδεδεμένο επίπεδο (fully connected layer).

4.2.1 Συνελικτικό Επίπεδο (Convlutional layer)

Ένα ΣΝΔ υλοποιεί την πράξη της συνέλιξης στα δεδομένα που δέχεται ως είσοδο στο συνελικτικό επίπεδο. Οι παράμετροι αυτού του επιπέδου συνιστούν ένα σύνολο από εκπαιδευόμενα φίλτρα. Κάθε φίλτρο υλοποιείται μέσω των συναπτικών βαρών ενός νευρώνα και είναι συνδεδεμένο στο πλήρες βάθος του όγκου εισόδου. Κάθε φίλτρο μετακινείται (συνελίσσεται) κατά μήκος του πλάτους και του ύψους του όγκου εισόδου με αποτέλεσμα την παραγωγή ενός δισδιάστατου χάρτη ενεργοποίησης αυτού του φίλτρου ο οποίος ονομάζεται **χάρτης χαρακτηριστικού**. Η διαδικασία της συνέλιξης μπορεί να ερμηνευθεί ως ο υπολογισμός του εσωτερικού γινομένου μεταξύ των τιμών του φίλτρου και των τιμών του όγκου εισόδου. Ο όγκος εξόδου ενός επιπέδου αποτελείται από το σύνολο όλων των χαρτών χαρακτηριστικών τοποθετημένους κατά τη διάσταση του βάθους του όγκου δεδομένων.

Μπορούμε να θεωρήσουμε είτε πως κάθε φίλτρο υλοποιείται από τα συναπτικά βάρη ενός νευρώνα ο οποίος μετακινείται κατά μήκος του πλάτους και του ύψους του όγκου εισόδου είτε πως όλοι οι νευρώνες που παράγουν έναν χάρτη χαρακτηριστικού μοιράζονται το ίδιο σύνολο συναπτικών βαρών. Το παράδειγμα του Σχήματος 4.1 απεικονίζει τη λειτουργία της συνέλιξης σ' έναν όγκο εισόδου $32 \times 3 \times 3$ ο οποίος αναπαριστά μία έγχρωμη τρικαναλική εικόνα. Το δεικτικό πεδίο του νευρώνα που υλοποιεί το συγκεκριμένο φίλτρο είναι 4×4 ή αλλιώς η λειτουργία της συνέλιξης γίνεται με έναν **συνελικτικό πυρήνα** (convolution kernel) μεγέθους 4×4. Συνεπώς, ο νευρώνας αυτός θα περιλαμβάνει $4 \times 4 \times 3 = 48$ συναπτικά βάρη. Η σύνδεση είναι χωρικά τοπική 4×4, αλλά εκτείνεται στο πλήρες βάθος (3). Εάν εφαρμόσουμε 24 διαφορετικά φίλτρα πάνω στην αρχική εικόνα το συνελικτικό επίπεδο θα παράγει μία αναπαράσταση μεγέθους $29 \times 29 \times 24$ η οποία θα αποτελείται από 24 χάρτες χαρακτηριστικών.



Σχήμα 4.1: Η λειτουργία της πράξης της συνέλιξης στα ΣΝΔ

Υπάρχουν τρεις διαφορετικές υπερπαράμετροι οι οποίες θα πρέπει να καθοριστούν σε ένα συνελικτικό επίπεδο:

- K: Ο αριθμός των φίλτρων που εφαρμόζουμε σε κάθε επίπεδο. Καθορίζει τον αριθμό των νευρώνων οι οποίοι θα βλέπουν την ίδια περιοχή του όγκου εισόδου. Όλοι αυτοί οι νευρώνες θα μάθουν να ενεργοποιούνται υπό την παρουσία διαφορετικών χαρακτηριστικών.

- S: Το βήμα μετακίνησης κατά την διαδικασία της συνέλιξης. Καθορίζει ανά πόσα στοιχεία του όγκου εισόδου θα μετακινηθεί το φίλτρο. Μεγάλες τιμές αυτής της παραμέτρου θα οδηγήσουν σε λιγότερο αλληλεπικαλυπτόμενα δεκτικά πεδία.

- P: Το μέγεθος της επέκτασης με μηδενικά. Πολλές φορές επιλέγουμε να συμπληρώσουμε τον όγκο εισόδου ενός συνελικτικού επιπέδου με μηδενικά προκειμένου να έχουμε καλύτερο έλεγχο του μεγέθους του όγκου εξόδου.

  Μπορούμε να υπολογίσουμε τις διαστάσεις του όγκου εξόδου ως συνάρτηση της διάστασης του όγκου εισόδου (W), του δεκτικού πεδίου (F) του νευρώνα, του βήματος μετακίνησης (S) και του μεγέθους της επέκτασης με μηδενικά (P). Ένα συνελικτικό επίπεδο:

  - Δέχεται έναν όγκο εισόδου $W_1 \times H_1 \times D_1$
  - Απαιτεί τη ρύθμιση 4 υπερπαραμέτρων:
    1. Αριθμός των φίλτρων ή αριθμός νευρώνων που βλέπουν στην ίδια περιοχή K
    2. Δεκτικό πεδίο των νευρώνων F

3. Βήμα μετακίνησης S
4. Μέγεθος της επέκτασης με μηδενικά P

- Παράγει έναν όγκο εξόδου μεγέθους $W_2 \times H_2 \times D_2$ όπου

$$W_2 = \frac{W_1 - F + 2P}{S} + 1$$

$$H_2 = \frac{H_1 - F + 2P}{S} + 1$$

$$D_2 = K$$

- Δημιουργεί $(F \cdot F \cdot D)$ βάρη για κάθε φίλτρο, συνολικά δηλαδή $(F \cdot F \cdot D) \cdot K$ βάρη για όλο το επίπεδο

Εάν ο το μέγεθος της διάστασης εξόδου δεν είναι ακέραιος, τότε οι νευρώνες του επιπέδου δεν μπορούν να διαταχθούν συμμετρικά κατά μήκος του όγκου εισόδου.

4.2.2 Επίπεδο Υποδειγματοληψίας (Pooling Layer)

Το επίπεδο υποδειγματοληψίας συνήθως παρεμβάλλεται ανάμεσα σε δύο συνελικτικά επίπεδα. Η χρήση του συνήθως εξυπηρετεί την ανάγκη μείωσης των χωρικών διαστάσεων μίας αναπαράστασης καθώς και τη μείωση του συνολικού αριθμού των ελεύθερων παραμέτρων του δικτύου. Η μείωση των ελεύθερων παραμέτρων μειώνει την πιθανότητες για υπερ-εκπαίδευση. Το επίπεδο υποδειγματοληψίας δεν περιλαμβάνει κάποιους νευρώνες με τη συνήθη έννοια, αλλά εκτελεί μία προκαθορισμένη λειτουργία.

Το συγκεκριμένο επίπεδο εφαρμόζει τη λειτουργία του μεμονωμένα σε κάθε χάρτη χαρακτηριστικών χωρίς να μειώνει το βάθος του όγκου δεδομένων. Οι δύο κύριοι τύποι υποδειγματοληψίας που χρησιμοποιούνται είναι η **υποδειγματοληψία μεγίστου** (max pooling) και η **υποδειγματοληψία μέσου όρου** (average pooling) . Όπως και στην πράξη της συνέλιξης, θα πρέπει να ορισθεί η χωρική έκταση (F) της λειτουργίας που θα εκτελεσθεί. Μία λειτουργία μέγιστης υποδειγματοληψίας με χωρική έκταση $F \times F$ θα εφαρμοσθεί σε μία περιοχή μεγέθους $F \times F$ και θα περιλαμβάνει στην έξοδο μόνο τη μέγιστη τιμή αυτής της περιοχής. Η υποδειγματοληψία μέσου όρου θα περιλαμβάνει το μέσο όρο των τιμών της περιοχής $F \times F$. Ένα επίπεδο υποδειγματοληψίας:

- Δέχεται έναν όγκο εισόδου $W_1 \times H_1 \times D_1$
- Απαιτεί τη ρύθμιση 2 υπερπαραμέτρων:
    1. Χωρική έκταση της υποδειγματοληψίας F
    2. Βήμα μετακίνησης S

- Παράγει έναν όγκο εξόδου μεγέθους $W_2 \times H_2 \times D_2$ όπου

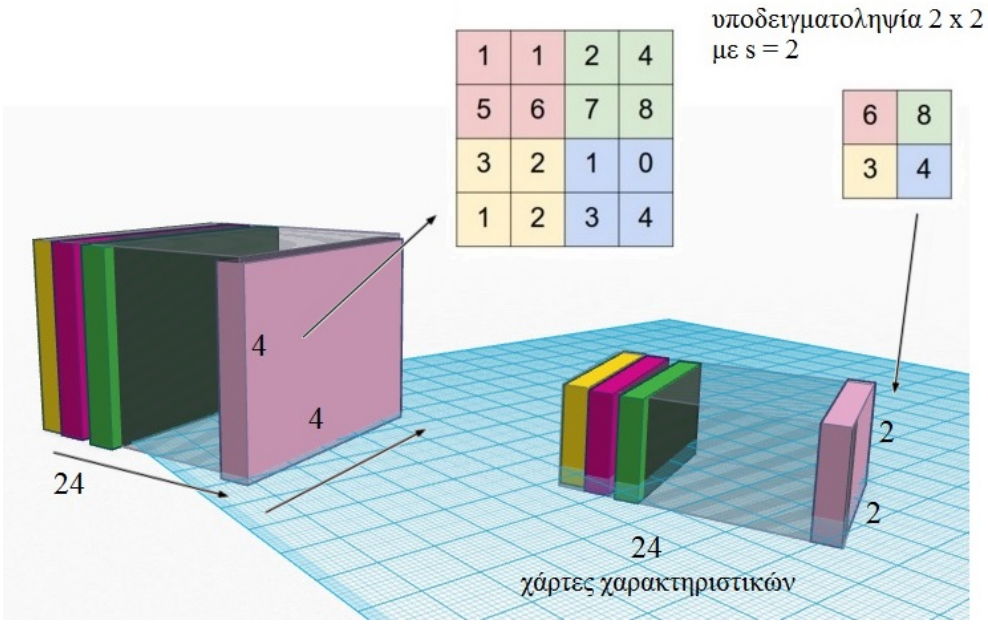
$$W_2 = \frac{W_1 - F}{S} + 1$$

$$H_2 = \frac{H_1 - F}{S} + 1$$

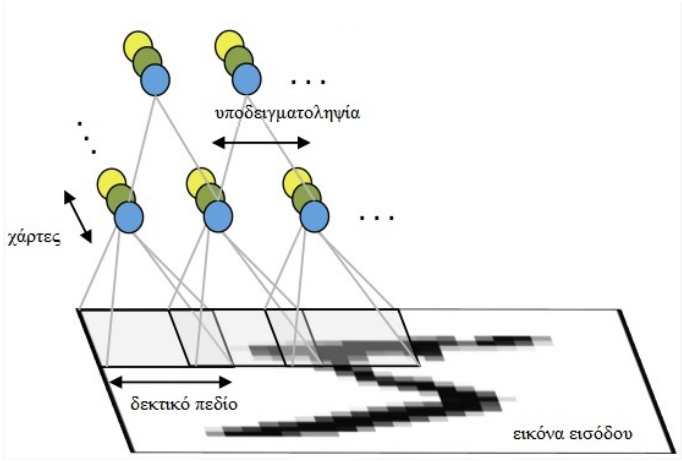
$$D_2 = K$$

- Εκτελεί μία προκαθορισμένη λειτουργία ανάλογα με το είδος της υποδειγματοληψίας.

Το Σχήμα 4.2 απεικονίζει τη λειτουργία μίας διαδικασίας υποδειγματοληψίας μίας αναπαράστασης μεγέθους $4 \times 4 \times 24$. Η υποδειγματοληψία μεγίστου 2×2 δημιουργεί έναν όγκο δεδομένων μεγέθους $2 \times 2 \times 24$. Το Σχήμα 4.3 απεικονίζει τη λειτουργία ενός επιπέδου συνέλιξης και ενός επιπέδου υποδειγματοληψίας. Οι νευρώνες που απεικονίζονται με ίδιο χρώμα έχουν τα ίδια συναπτικά βάρη, ενώ οι νευρώνες που απεικονίζονται με διαφορετικό χρώμα αφορούν διαφορετικούς χάρτες χαρακτηριστικών.



Σχήμα 4.2: Η λειτουργία ενός επιπέδου υποδειγματοληψίας μεγίστου,
Πηγή : **https://stanford.io/2ahO7ka**



Σχήμα 4.3: Το συνελικτικό επίπεδο και το επίπεδο υποδειγματληψίας,
Πηγή : **https://stanford.io/2ahO7ka**

4.2.3 Πλήρως Συνδεδεμένο Επίπεδο (Fully Connected Layer)

Σε ένα πλήρως συνδεδεμένο επίπεδο οι νευρώνες είναι συνδεδεμένοι με όλους τους νευρώνες του προηγούμενου επιπέδου. Η πρώτη διαφορά μεταξύ ενός συνελικτικού επιπέδου και ενός πλήρως συνδεδεμένου επιπέδου είναι πως οι νευρώνες στο συνελικτικό επίπεδο συνδέονται μόνο σε μία τοπική περιοχή της εισόδου. Μία δεύτερη διαφοροποίηση αφορά το γεγονός πως οι νευρώνες ενός χάρτη χαρακτηριστικών μοιράζονται το ίδιο σύνολο συναπτικών βαρών. Ωστόσο, και στα δύο επίπεδα η λειτουργία των νευρώνων είναι παρόμοια καθώς εκτελούν την πράξη υπολογισμού ενός εσωτερικού γινομένου, γεγονός που επιτρέπει τη μετατροπή ενός συνελικτικού επιπέδου σε ένα πλήρες συνδεδεμένο επίπεδο.

Ένα πλήρως συνδεδεμένο επίπεδο μπορεί να θεωρηθεί ως ένα συνελικτικό επίπεδο στο οποίο οι νευρώνες έχουν δεκτικό πεδίο με μέγεθος ίσο με το μέγεθος της αναπαράστασης που εισάγεται στο επίπεδο. Με άλλα λόγια, θα πρέπει να θέσουμε την χωρική έκταση του φίλτρου της συνέλιξης ακριβώς ίδιο μέγεθος με το μέγεθος του όγκου εισόδου. Για παράδειγμα, ένα πλήρες συνδεδεμένο επίπεδο το οποίο βλέπει μία περιοχή μεγέθους $7 \times 7 \times 32$ με $K = 64$ είναι ισοδύναμο με ένα συνελικτικό επίπεδο με παραμέτρους $F = 7, P = 0, S = 1$ και $K = 64$. Η έξοδος του επιπέδου συνέλιξης θα είναι ένας όγκος διαστάσεων $1 \times 1 \times 64$, παράγοντας ένα αποτέλεσμα ίδιο με αυτό που θα παρήγαγε ένα πλήρες συνδεδεμένο επίπεδο.

Ένα ΣΝΔ το οποίο αποτελείται μόνο από επίπεδα συνέλιξης ονομάζεται **πλήρως συνελικτικό** (fully convolutional). Ένα βαθύ νευρωνικό δικτύο (deep neural network) μπορεί να θεωρηθεί ως μία διαδικασία προσέγγισης μίας μη γραμμικής συνάρτησης. Αντίστοιχα, η λειτουργία ενός ΣΝΔ το οποίο είναι πλήρως συνελικτικό μπορεί να ερμηνευθεί ως μία διαδικασία εφαρμογής ενός μη γραμμικού φίλτρου το οποίο ονομάζεται βαθύ φίλτρο (deep filter). Ένα πλήρως συνελικτικό νευρωνικό δίκτυο μπορεί να δεχθεί ως είσοδο εικόνες οποιουδήποτε μεγέθους και να δημιουργήσει στην έξοδό του έναν χάρτη θερμότητας (heatmap). Κάθε θέση του χάρτη θερμότητας περιγράφει την πιθανότητα ύπαρξης μίας συγκεκριμένης κλάσης στην

αντίστοιχη θέση της εικόνας εισόδου.

4.2.4 Επίπεδο αποβολής (Dropout Layer)

Η τεχνική της απόσυρσης, [12], δημιουργήθηκε για να αντιμετωπιστεί αποτελεσματικά το πρόβλημα της **υπερ-εκπαίδευσης**. Ο όρος υπερ-εκπαίδευση περιγράφει την υπερπροσαρμογή του δικτύου στα δεδομένα εκπαίδευσης. Σκοπός της διαδικασίας της απόσυρσης είναι να εισάγει μία λειτουργία σύμφωνα με την οποία κατά τη διάρκεια της μάθησης, ένα τυχαίο πλήθος νευρώνων διατηρούνται ενεργοί ενώ οι υπόλοιποι απενεργοποιούνται διαδίδοντας μηδενικά στην έξοδο. Η τυχαιότητα περιγράφεται μέσω μίας τυχαίας μεταβλητής η οποία ακολουθεί κατανομή Bernoulli. Το αποτέλεσμα της απόσυρσης είναι πως το εκπαιδευόμενο νευρωνικό δίκτυο συμπεριφέρεται όπως πολλά νευρωνικά δίκτυα μαζί. Συγκεκριμένα εάν συμβολίσουμε με p την πιθανότητα διατήρησης μίας σύνδεσης στην περίπτωση που αποκόβονται k συνδέσεις το νευρωνικό δίκτυο συμπεριφέρεται ως μια συλλογή $(\frac{1}{p})^k$ νευρωνικών δικτύων.

Κεφάλαιο 5

Αναδρομικά Νευρωνικά Δίκτυα

Τα αναδρομικά νευρωνικά δίκτυα είναι αυτά που η δομή τους μοιάζει περισσότερο με τη λειτουργία του ανθρώπινου εγκεφάλου, όπου οι διεργασίες της αντίληψης, της λήψης αποφάσεων και της εκμάθησης είναι αμιγώς μη γραμμικές. Το κύριο χαρακτηριστικό τους που τα διαφοροποιεί από τα δίκτυα πρόσθιας τροφοδότησης είναι ότι περιέχουν τουλάχιστον μια ανάδραση ανάμεσα στους κόμβους του ίδιου επιπέδου ή κόμβους διαφορετικών επιπέδων, όπως αναφέρθηκε στην προηγούμενη ενότητα. Το πρώτο είναι αυτό της αντίληψης του χρόνου, δηλαδή σε εφαρμογές με δυναμικά συστήματα μπορούν να χρησιμοποιηθούν για να προβλέψουν την έξοδο της στιγμής $t+1$ έχοντας δεδομένα της στιγμής t ή και προηγούμενων. Το επόμενο χαρακτηριστικό είναι η εισαγωγή της μνήμης στα νευρωνικά δίκτυα. Τέτοιου είδους δίκτυα μπορούν να χρησιμοποιηθούν εγγενώς σε προβλήματα στα οποία οι είσοδοι δεν είναι ανεξάρτητες μεταξύ τους, αλλά υπάρχει κάποια συσχέτιση με προηγούμενα δείγματα.

Τα αναδρομικά νευρωνικά δίκτυα είναι δυναμικά συστήματα που μπορούν να μοντελοποιήσουν και να αναπαραστήσουν χρονικά μεταβαλλόμενα μοντέλα. Το γεγονός ότι χρησιμοποιούνται για να προσομοιώσουν συστήματα με τη χρήση κοινών διαφορικών εξισώσεων, τα κάνει κατάλληλα για ψηφιακή υλοποίηση και έχουν επιπλέον ισχυρές υπολογιστικές δυνατότητες όταν χρησιμοποιούνται σε αυτά τα μοντέλα. Τέτοιου είδους μοντέλα είναι και όλα τα βιολογικά νευρωνικά δίκτυα. Μπορούν, επίσης, να χρησιμοποιηθούν αποδοτικά σε θέματα που περιλαμβάνουν την αναγνώριση και αντίστροφη ταυτοποίηση συστημάτων, το φιλτράρισμα και την πρόβλεψη πληροφοριών, την κατάταξη δεδομένων σε κλάσεις, την μοντελοποίηση στοχαστικών ακολουθιών, την συμπίεση δεδομένων κ.α. [13]. Ένας από τους τομείς που χρησιμοποιούνται αποδιτικά είναι και η **αναγνώριση ομιλίας**.

5.1 Βασικές κατηγορίες των Αναδρομικών Νευρωνικών Δικτύων (RNN)

Το βασικό δομικό στοιχείο ενός αναδρομικού νευρωνικού δικτύου είναι ο τρόπος σύνδεσης των νευρώνων [14], [15]. Τα αναδρομικά νευρωνικά δίκτυα κατηγοριοποιούνται σε πλήρως αναδρομικά δίκτυα, στα οποία επιτρέπεται η σύνδεση όλων των νευρώνων μεταξύ τους και στα τοπικά αναδρομικά δίκτυα. Στα πλήρως αναδρομικά δίκτυα δεν υπάρχει διάκριση στους κόμ-

βους εισόδου και κάθε κόμβος μπορεί να αποτελέσει είσοδο για οποιονδήποτε άλλο κόμβο, συμπεριλαμβανομένου και του εαυτού του. Τα δεύτερα περιέχουν, συνήθως, μόνο αναδρομικές συνδέσεις μεταξύ των νευρώνων που βρίσκονται ίδιο επίπεδο και η διαδικασία της διάδοσης προς τα εμπρός κατά την εκμάθηση μοιάζει πολύ με αυτή των δικτύων πρόσθιας τροφοδότησης. Συγκριτικά με τα τοπικά αναδρομικά δίκτυα, τα πλήρως αναδρομικά πάσχουν από θέματα ευστάθειας κατά την εκπαίδευση και απαιτούν τη χρήση περίπλοκων και χρονοβόρων αλγορίθμων εκμάθησης. Τα τοπικά αναδρομικά δίκτυα έχουν, αντιθέτως, πιο απλή δομή που τα κάνει πιο αποδοτικά κατά τη διαδικασία εκμάθησης και παρέχουν τη δυνατότητα για έλεγχο της ευστάθειας στους εσωτερικούς τους κόμβους. Μια ακόμα διάκριση των RNN που χρησιμοποιούνται σε εφαρμογές διακριτού χρόνου είναι σε αναδρομικά δίκτυα με χρονοκαθυστέρηση και σύγχρονα αναδρομικά δίκτυα. Τα δίκτυα που λειτουργούν με χρονοκαθυστέρηση εκπαιδεύονται με στόχο τη μείωση του σφάλματος πρόβλεψης, ενώ τα σύγχρονα δίκτυα δε στοχεύουν στην ιδιότητα του να έχουν μνήμη ή καλύτερη πρόβλεψη όσο εκπαιδεύεται το δίκτυο, αλλά κάνουν χρήση των αναδράσεων με στόχο να προσφέρουν καλύτερη δυνατότητα προσέγγισης συναρτήσεων σύμφωνα με τις αρχές της θεωρίας του Turing και της πολυπλοκότητας. Σε αυτό τον τομέα έχει αποδειχθεί ότι έχουν πολύ ισχυρές ικανότητες και έχει φανεί πειραματικά ότι μπορούν να ''μάθουν'' οποιαδήποτε συνάρτηση πηγάζει από κάποιο MLP (Multi Layer Perceptron), χωρίς όμως να ισχύει το αντίστροφο.

5.2 Αλγόριθμοι και τεχνικές εκμάθησης στα Αναδρομικά Νευρωνικά Δίκτυα

Τις τελευταίες δεκαετίες, που έχει γνωρίσει μεγάλη άνθιση ο τομέας των νευρωνικών δικτύων, έχουν εφαρμοστεί πολλές μέθοδοι που χρησιμοποιούνται στη διαδικασία της εκπαίδευσης τους. Αναμφίβολα, ο πιο δημοφιλής κανόνας εκμάθησης στα νευρωνικά δίκτυα είναι αυτός του Back Propagation (αλγόριθμος οπισθοδιάδοσης) όπου έχουμε προαναφερθεί στην Ενότητα (3.9) και έχει κατά συνέπεια επεκταθεί και στα αναδρομικά νευρωνικά δίκτυα με την ονομασία Back Propagation through time (BPTT) [16]. Αποτελεί μια γενίκευση του αλγορίθμου Ελαχίστων Μέσων Τετραγώνων (Least Mean Squares - LMS) και ονομάζεται, επίσης, γενικευμένος κανόνας δέλτα (generalized delta rule). Χρησιμοποιεί μια τεχνική αναζήτησης αντίρροπα προς το διάνυσμα κλίσης (gradient) με στόχο να ελαχιστοποιήσει μια συνάρτηση κόστους που έχει οριστεί για την αξιολόγηση της απόκλισης μεταξύ του επιθυμητού σήματος και της εξόδου του νευρωνικού δικτύου. Η συνάρτηση κόστους που αξιολογείται είναι συνήθως το Μέσο Τετραγωνικό Σφάλμα (Mean Squared Error - MSE).

5.2.1 Back Propagation

Ο αλγόριθμος του Back Propagation κάνει επαναληπτικά την εφαρμογή σχέσεων για κάθε εποχή εκπαίδευσης μέχρι να επιτύχει μείωση του σφάλματος σε τιμή μικρότερη από κάποιο προκαθορισμένο κατώφλι (threshold) ή μέχρι η μεταβολή του σφάλματος να πέσει κάτω από ένα άλλο προκαθορισμένο κατώφλι ή μέχρι να ολοκληρωθεί ο αριθμός των εποχών που έχει τεθεί στην αρχή της εκπαίδευσης. Συνήθως απαιτείται κάποιος αρκετά μεγάλος αριθμός εποχών

(training epochs) για να εκπληρωθεί κάποιος από τους παραπάνω περιορισμούς. Η προσέγγιση μέσω της μεθόδου gradient descent και του αλγορίθμου του Back Propagation έχει το μειονέκτημα της αργής σύγκλισης, διότι τυπικά επιλέγεται κάποια μικρή σταθερά εκμάθησης για να αποφευχθεί η αποσταθεροποίηση του δικτύου. Υπάρχουν, όμως, τρόποι για την επίσπευση της σύγκλισης, όπως η χρήση δευτεροβάθμιων τεχνικών gradient descent (3.8) που εκμεταλλεύονται την καμπυλότητα της κλίσης του σφάλματος, αλλά έχουν υψηλότερη πολυπλοκότητα. Ένα ακόμη αρνητικό που έχουν οι τεχνικές gradient descent είναι ότι γίνεται αναζήτηση μόνο του τοπικού ελαχίστου του σφάλματος. Το πρόβλημα αυτό μπορεί να αντιμετωπιστεί με διάφορους τρόπους, όπως η προσθήκη θορύβου κατά την εκμάθηση του δικτύου, η επανάληψη όλης της διαδικασίας με διαφορετική αρχικοποίηση των βαρών ή με τη χρήση υπαρχόντων πληροφοριών των δεδομένων εισόδου- εξόδου που μπορούν να φανούν χρήσιμες κατά την εκπαίδευση.

5.2.2 Back Propagation Through Time (BPTT)

Ο αλγόριθμος του Back Propagation που χρησιμοποιείται στα feedforward δίκτυα δεν μπορεί να μεταφερθεί ως έχει στα αναδρομικά, για το λόγο ότι προϋποθέτει την ύπαρξη αποκλειστικά ακυκλικών συνδέσεων μεταξύ των κόμβων του δικτύου ούτως ώστε να γίνει η προς τα πίσω διάδοση του σφάλματος. Η λύση δίνεται με την εισαγωγή του backpropagation through time αλγορίθμου που ‘‘εδιπλώνει’’ τις συνδέσεις μεταξύ των κόμβων σε ξεχωριστά χρονικά βήματα, δημιουργώντας πανομοιότυπα αντίγραφα και ανακατευθύνει τις συνδέσεις μεταξύ αυτών ώστε να προκύψει ένα feedforward δίκτυο [16].

Η διαδικασία της εκμάθησης ξεκινάει από το πρώτο επίπεδο και προχωράει σταδιακά στα επόμενα επίπεδα της στοίβας που έχει δημιουργηθεί από το ξεδίπλωμα των επιπέδων στο χρόνο. Σε κάθε αντίγραφο των επιπέδων τη χρονική στιγμή n διαβάζεται η είσοδος $u(n)$, υπολογίζεται το $x(n)$ των ενδιάμεσων επιπέδων με βάση τα $u(n), x(n-1)$ και $y(n-1)$ (όταν το τελευταίο δεν είναι 0) και τέλος υπολογίζεται η έξοδος $y(n)$. Με συνάρτηση την συναρτηση σφάλματος που ελαχιστοποιείται την:

$$E = \sum_{n=1}^{T} \Big| d(n) - y(n) \Big|^2 = \sum_{n=1}^{T} E(n) \tag{5.1}$$

με τη διαφορά ότι η έννοια του t έχει μετατραπεί από τον αύξοντα αριθμό του δείγματος εκπαίδευσης σε χρονική στιγμή.

Ο αλγόριθμος που ακολουθεί το BPTT έχει ως εξής:

Είσοδος: η χρονοσειρά των δεδομένων εκπαίδευσης και τα βάρη w_{ij} που αντιστοιχούν στην συγκεκριμένη χρονική στιγμή.

Έξοδος: τα νέα βάρη των συνδέσεων.

Υπολογιστικά βήματα του αλγορίθμου:

1. Forward pass, όπως περιγράφηκε παραπάνω μέχρι την έξοδο $y(n)$.

2. Υπολογισμός από το τέλος (T) προς την αρχή των επιπέδων (για $n = T, \ldots, 1$). Η ενεργοποίηση των κόμβων $x_i(n), y_j(n)$ ενός όρου της διάδοσης του σφάλματος $\delta_i(n)$ με $z_i(n), z_i(T)$ να είναι η μέγιστη τιμή που μπορούν να πάρουν οι συγκεκριμένοι κόμβοι, δίνεται από τους τύπους:

$$\delta_j(T) = (d_j(T) - y_j(T))\frac{\partial f(u)}{\partial u}\Big|_{u=z_j(T)} \tag{5.2}$$

για τους κόμβους εξόδου στο χρονικό επίπεδο T,

$$\delta_i(T) = \Big[\sum_{j=1}^{L} \delta_j(T) w_{ji}^{out}\Big]\frac{\partial f(u)}{\partial u}\Big|_{u=z_i(T)} \tag{5.3}$$

για τους εσωτερικούς κόμβους $x_i(T)$ στο χρονικό επίπεδο T,

$$\delta_j(T) = \Big[(d_j(T) - y_j(T)) + \sum_{i=1}^{N} \delta_i(n+1) w_{ij}^{back}\Big]\frac{\partial f(u)}{\partial u}\Big|_{u=z_j(n)} \tag{5.4}$$

για τους κόμβους εξόδου των προηγούμενων χρονικών επιπέδων και

$$\delta_i(n) = \Big[\sum_{i=1}^{N} \delta_j(n+1) w_{ij} + \sum_{j=1}^{L} \delta_j(n) w_{ji}^{out}\Big]\frac{\partial f(u)}{\partial u}\Big|_{u=z_i(n)} \tag{5.5}$$

για τους εσωτερικούς κόμβους των προηγούμενων χρονικών επιπέδων.

3. Εκ νέου υπολογισμός των βαρών σύμφωνα με τις σχέσεις:

$$\text{new } w_{ij} = w_{ij} + \gamma \sum_{n=1}^{T} \delta_i(n) x_j(n-1) \qquad \text{για } x_j(n-1) = 1 \text{ και } n = 1 \tag{5.6}$$

$$\text{new } w_{ij}^{in} = w_{ij}^{in} + \gamma \sum_{n=1}^{T} \delta_i(n) u_j(n) \tag{5.7}$$

$$\text{new } w_{ij}^{out} = w_{ij}^{out} + \gamma \begin{cases} \sum_{n=1}^{T} \delta_i(n) u_j(n), & \text{αν το } j \text{ είναι κόμβος της εισόδου} \\ \sum_{n=1}^{T} \delta_i(n) x_j(n-1), & \text{αν το } j \text{ είναι εσωτερικός κόμβος} \end{cases} \tag{5.8}$$

$$\text{new } w_{ij}^{back} + \gamma \sum_{n=1}^{T} \delta_i(n) y_j(n-1) \qquad \text{για } y_j(n-1) = 1 \text{ και } n = 1 \tag{5.9}$$

Το θέμα της αργής συγκλισης που αναφέρθηκε για αλγόριθμο οπισθοδιάδοσης στα πρόσθια δίκτυα [17] εξακολουθεί να υπάρχει και στο BPTT και η πολυπλοκότητα του αλγορίθμου που περιγράφηκε είναι $O(TN^2)$, με N τον αριθμό των εσωτερικών κόμβων. Συνήθως, χρειάζονται πολλές χιλιάδες εποχές για να ολοκληρωθεί η διαδικασία της εκπαίδευσης. Η συνεχόμενη εκτέλεση αυτών των εποχών έχει ως αποτέλεσμα τη δημιουργία ενός πολύπλοκου δυναμικού συστήματος που συχνά μπορεί να παρεκκλίνει της επιθυμητής συμπεριφοράς. Επομένως, είναι

πιθανό να δημιουργηθούν διακλαδώσεις όταν οι τιμές αρχικοποίησης των βαρών του δικτύου είναι αρκετά διαφορετικές από τη δυναμική του συστήματος που προσπαθούμε να μοντελοποιήσουμε. Αποτέλεσμα αυτών των διακλαδώσεων μπορεί να είναι η αλλοίωση των πληροφοριών της κλίσης και η εκτόξευση του σφάλματος σε μη αποδεκτές (πολύ υψηλές) τιμές, το οποίο στην περίπτωση του BPTT δεν εγγυάται τη σύγκλιση σε κάποια κοντινή περιοχή του ελαχίστου του. Τα προβλήματα αυτά δεν συναντώνται στα feedforward δίκτυα για το λόγο ότι μοντελοποιούν μόνο απλές συναρτήσεις και όχι δυναμικά συστήματα. Τέλος, δεν υπάρχει κάποια συγκεκριμένη τεχνική για να προσπεραστούν αυτά τα προβλήματα και συνήθως χρειάζονται αρκετά πειράματα και υπολογιστικός χρόνος για να επιτευχθεί ένα ικανοποιητικό αποτέλεσμα. Για τους λόγους αυτούς, η τεχνική του BPTT χρησιμοποιείται σχεδόν αποκλειστικά σε μικρά δίκτυα μεγέθους 3-20 κόμβων ανά επίπεδο και η χρήση μεγαλύτερων δικτύων αποδεικνύεται πολύ δαπανηρή από άποψη χρήσης υπολογιστικού εξοπλισμού και χρόνου. Ένα ακόμη μειονέκτημα που αφορά την ομαδική εκπαίδευση που γίνεται στο BPTT, αλλά και ο απλός αλγόριθμος οπισθοδιάδοσης στα πρόσθια δίκτυα είναι ότι η μεταβολή των βαρών γίνεται αποκλειστικά στο τέλος κάθε εποχής, μετά από ένα πλήρες πέρασμα των δεδομένων εκμάθησης. Το γεγονός αυτό καθιστά την τεχνική του αλγορίθμου οπισθοδιάδοσης μη κατάλληλη για εφαρμογές που τρέχουν σε πραγματικό χρόνο και απαιτούν τη συνεχή ενημέρωση των βαρών. Σε τέτοιου είδους εφαρμογές χρησιμοποιείται συνήθως ένας άλλος αλγόριθμος εκμάθησης, αυτός του **Real Time Recurrent Learning** (RTRL), που πραγματοποιεί την ενημέρωση των βαρών στο τέλος κάθε βήματος του αλγορίθμου.

5.2.3 Real Time Recurrent Learning (RTRL)

Σύμφωνα με αυτή την τεχνική, η επίδραση της αλλαγής των βαρών στη δυναμική του δικτύου μπορεί να φανεί από την παρακάτω συνάρτηση ενεργοποίησης [16]. Για ευκολία και καλύτερη κατανόηση χρησιμοποιείται η σήμανση i για την ενεργοποίηση όλων των κόμβων (ανεξάρτητα αν είναι κόμβοι εισόδου, εξόδου ή εσωτερικοί), με το i να παίρνει τιμές από 1 έως N για τους εσωτερικούς κόμβους, $N+1$ έως $N+L$ για τους εξωτερικούς και $N+L+1$ έως $N+L+K$ για τους κόμβους εισόδου. Τα βάρη μεταξύ των συνδέσεων αναφέρονται ως w_{kl}.

$$\frac{\partial v_i(n+1)}{\partial w_{kl}} = f'(z_i(n))\Big[\Big(\sum_{j=1}^{N+L} w_{ij}\frac{\partial v_i(n)}{\partial w_{kl}} + \delta_{ik}v_l(n)\Big)\Big] \tag{5.10}$$

όπου $i = 1, \ldots, N+L$,$k, l <= N+L+K$ και $z_i(n)$ η μέγιστη τιμή του κόμβου. Το δ_{ik} αναφέρεται στη συνάρτηση δέλτα του $Kronecker$ (ισούται με 1 όταν $i = k$ και είναι 0 αλλού) και ο όρος $\delta_{ik}v_l(n)$ στην επίδραση του βάρους w_{kl} στον κόμβο k του συστήματος. Το άθροισμα που βρίσκεται στις αγκύλες είναι η συγκεντρωτική επίδραση των βαρών του δικτύου στο σύστημα. Η συνάρτηση (5.10) αποτελεί ένα δυναμικό σύστημα διακριτού χρόνου, διάστασης $N+L$ με συντελεστές που μεταβάλλονται στο χρόνο. Η δυναμική μεταβλητή του συστήματος έχει τη μορφή:

$$\Big(\frac{\partial v_1}{\partial w_{kl}}, \ldots, \frac{\partial v_{N+L}}{\partial w_{kl}}\Big) \tag{5.11}$$

Για το λόγο ότι η αρχική κατάσταση του δικτύου είναι ανεξάρτητη από τα βάρη των συνδέσεών του, μπορούμε να αρχικοποιήσουμε το παραπάνω διάνυσμα με μηδενικές τιμές. Με αυτό τον τρόπο υπολογίζεται η (5.11) επαναλαμβάνοντας τη συνάρτηση (5.10) για διαδοχικά χρονικά βήματα. Για τη λύση αυτή, γίνεται υπολογισμός της κλίσης του σφάλματος που δίνεται από τη σχέση:

$$\frac{\partial E}{\partial w_{kl}} = 2\sum_{n=1}^{T}\sum_{i=N}^{N+L}\big(v_i(n) - d_i(n)\big)\frac{\partial v_i(n)}{\partial w_{kl}}. \tag{5.12}$$

Στην περίπτωση του αλγορίθμου ομαδικής εκμάθησης με κατάβαση του διανύσματος οξύτατης κλίσης θα έπρεπε να γίνει συσσώρευση της κλίσης του σφάλματος της σχέσης (5.12) για μία ολόκληρη εποχή της ακολουθίας εκμάθησης και έπειτα να γίνει η ανανέωση των βαρών με τη σχέση:

$$\text{new } w_{kl} = w_{kl} - \gamma\frac{\partial E}{\partial w_{kl}}, \tag{5.13}$$

όπου με γ συμβολίζεται και πάλι το learning rate. Μια εναλλακτική προσέγγιση που εισάγει το RTRL είναι η ανανέωση της κλίσης του σφάλματος μετά από την περάτωση κάθε βήματος σύμφωνα με τον τύπο:

$$w_{kl}(n+1) = w_{kl}(n) - \gamma\sum_{i=1}^{L}\big(v_i(n) - d_i(n)\big)\frac{\partial v_i(n)}{\partial w_{kl}}. \tag{5.14}$$

Η προσέγγιση αυτή καθιστά το RTRL κατάλληλο για εφαρμογές που απαιτούν τη σύγχρονη ενημέρωση των βαρών. Έχει, όμως, πολυπλοκότητα $O(N+L)^4$ για κάθε βήμα, λόγω του ότι πρέπει να επιλυθεί ένα σύστημα διάστασης $(N+L)$ για την τελική προσαρμογή κάθε βάρους. Για το λόγο αυτό, της υψηλής πολυπλοκότητας, το RTRL χρησιμοποιείται κυρίως σε θέματα που απαιτούν συνεχόμενη προσαρμογή των βαρών και μπορούν να υλοποιηθούν με σχετικά μικρά δίκτυα.

5.3 Δίκτυα Μακράς Βραχυχρόνιας Μνήμης (Long Short Term Memory - LSTM)

5.3.1 Εισαγωγή

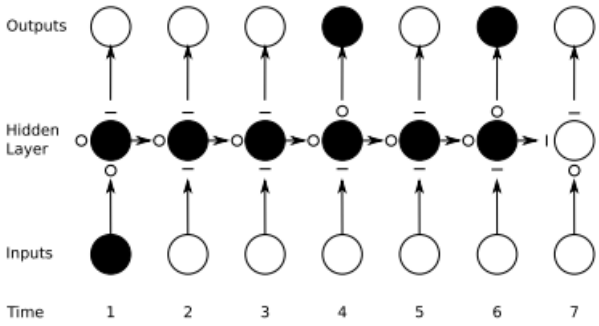
Ένα από τα χαρακτηριστικά των αναδρομικών νευρωνικών δικτύων, που τα έχει καταστήσει πολύ δημοφιλή στην κοινότητα της μηχανικής μάθησης, είναι ότι έχουν τη δυνατότητα να συνδυάζουν πληροφορίες από προηγούμενα δεδομένα εκμάθησης και να τις χρησιμοποιούν κατά την εκπαίδευση των τωρινών δειγμάτων [18]. Η αλήθεια, όμως, είναι πως στην πράξη υπάρχουν κάποιοι περιορισμοί. Παραδείγματος χάρη, άμα αναλογιστούμε ένα μοντέλο φυσικής γλώσσας και η πρόθεσή μας είναι να προβλέψουμε την επόμενη λέξη του κειμένου μιας ακολουθίας, όπως ‘‘τα σύννεφα βρίσκονται στον ουρανό’’, ένα απλό RNN θα μπορούσε με ευκολία να προβλέψει ότι η επόμενη λέξη πρέπει να είναι ο ‘‘ουρανός’’. Στην περίπτωση, όμως, που θα είχαμε μια μεγαλύτερη ακολουθία, όπως, ‘‘Γεννήθηκα και μεγάλωσα στην Ελλάδα... μιλάω άπταιστα ελληνικά’’, οι πρόσφατες πληροφορίες που έχει το δίκτυο καταδεικνύουν ότι, προφανώς, ακολουθεί το όνομα μιας γλώσσας. Προκειμένου, όμως, να περιορίσουμε τις διαθέσιμες επιλογές

χρειαζόμαστε την πληροφορία της ''Ελλάδας '', που βρίσκεται αρκετά πιο πίσω στο κείμενο. Υπάρχει μεγάλη πιθανότητα, δηλαδή, σε αυτό το μοντέλο της γλώσσας, η σχετική θέση των πληροφοριών που απαιτούνται για την εξαγωγή του σωστού αποτελέσματος να είναι αρκετά απομακρυσμένη. Όσο το κενό μεταξύ των αλληλεξαρτήσεων μεγαλώνει, τα RNN με τις κλασικές μεθόδους εκμάθησης όπως με την κατάβαση του διανύσματος οξύτατης κλίσης που περιγράφηκαν στην προηγούμενη ενότητα, καθίστανται ανίκανα να συνδυάσουν αυτές τις πληροφορίες, λόγω του ότι ο αριθμός των προηγούμενων βημάτων που χρησιμοποιούνται σαν μνήμη κατά την εκπαίδευση είναι πεπερασμένος. Στην πράξη αυτό σημαίνει ότι η εκπαίδευση σε τέτοιου είδους ακολουθίες καταναλώνει πάρα πολύ χρόνο και πόρους και πολλές φορές το σφάλμα εκτοξεύεται σε υψηλές τιμές, προκαλώντας μεγάλες μεταβολές των βαρών του συστήματος σε κάθε επανάληψη της διαδικασίας εκμάθησης.

Για να ξεπεραστεί αυτό το πρόβλημα των συσχετίσεων που βρίσκονται μακριά μεταξύ τους, έχουν δοκιμαστεί πολλές προσεγγίσεις, που ουσιαστικά αποτελούν μικρές ή μεγάλες μεταποιήσεις των αναδρομικών νευρωνικών δικτύων, όπως η εισαγωγή χρονικής καθυστέρησης, η εφαρμογή της διάδοσης του σφάλματος σε διακριτούς χρόνους ή η εισαγωγή κάποιων σταθερών στα δίκτυα. Η τεχνική, όμως, που έχει επικρατήσει και έχει να αναδείξει τα καλύτερα αποτελέσματα σε τέτοιου είδους αλλά και σε πολλές άλλες εφαρμογές, είναι αυτή των (LSTM). Εισήχθησαν πρώτη φορά το 1997 από τους Sepp Hochreiter και Jôrgen Schmidhuber. Η συγκράτηση πληροφοριών στη μνήμη για μεγάλα χρονικά διαστήματα είναι το βασικό εξ΄ ορισμού χαρακτηριστικό της λειτουργίας τους. Σχεδόν όλα τα καθιερωμένα αποτελέσματα των αναδρομικών νευρωνικών δικτύων έχουν επιτευχθεί με τη χρήση αυτών.

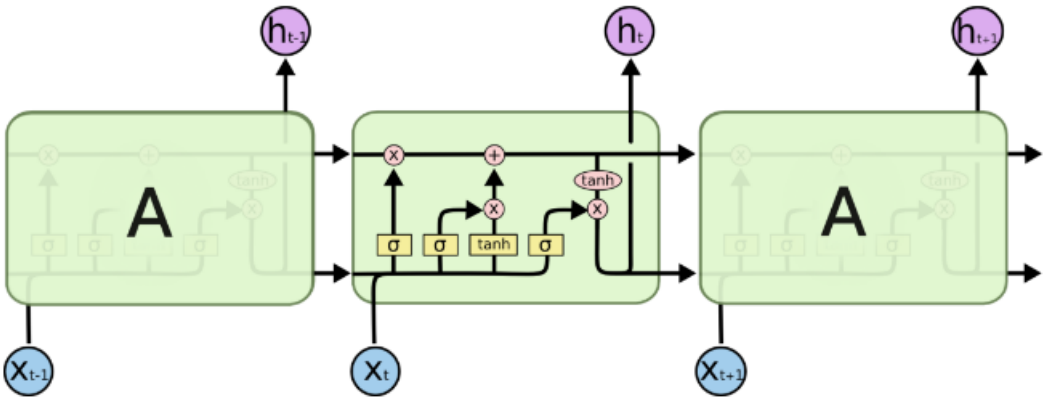
5.3.2 Αρχιτεκτονική και Λειτουργία των Δικτύων LSTM

Η αρχιτεκτονική των LSTM μοιάζει αρκετά με αυτή των υπόλοιπων RNN, με την έννοια ότι απαρτίζονται και αυτά από έναν αριθμό μονάδων συνδεδεμένων αλυσιδωτά μεταξύ τους σε κάθε επίπεδο. Η διαφορά βρίσκεται στο ότι οι μονάδες (modules ή blocks) που βρίσκονται στα εσωτερικά επίπεδα του δικτύου περιέχουν κάποια επιπλέον στοιχεία και ονομάζονται σε αυτή την περίπτωση μονάδες μνήμης (memory blocks). Πιο συγκεκριμένα, κάθε μονάδα των LSTM αποτελείται από ένα ή περισσότερα κελιά μνήμης που συνδέονται μεταξύ τους και τρία ακόμη στοιχεία, τις πύλες εισόδου, εξόδου και επιλεκτικής αμνησίας (forget gate), οι οποίες είναι αντιστοίχως υπεύθυνες για τις λειτουργίες εγγραφής, ανάγνωσης και επαναφοράς των κελιών. Η χρήση αυτών των πυλών διασφαλίζει την αποθήκευση και πρόσβαση στις πληροφορίες ακόμα και με την πάροδο μεγάλων χρονικών περιόδων ή πολλών βημάτων. Στο σχήμα 5.1 φαίνεται η διατήρηση της πληροφορίας του βήματος 1 με την πάροδο του χρόνου σε ένα LSTM με ένα κρυφό επίπεδο. Ο συμβολισμός ○ και - σημαίνει ότι η εκάστοτε πύλη είναι αντίστοιχα ανοιχτή ή κλειστή. Παρατηρούμε, λοιπόν, ότι η μονάδα μνήμης είναι σε θέση να συγκρατήσει την πληροφορία του πρώτου βήματος, εφόσον η πύλη εισόδου είναι κλειστή και αυτή της επιλεκτικής συγκράτησης ανοιχτή. Για απλούστευση του παραδείγματος, οι πύλες είναι είτε πλήρως ανοιχτές (1), είτε κλειστές (0).



Σχήμα 5.1: Διατήρηση της πληροφορίας στα δίκτυα LSTM,
Πηγή : **https://bit.ly/2HzAono**

Στο Σχήμα 5.2 φαίνονται τα δομικά στοιχεία των μονάδων μνήμης των LSTM, καθώς και οι συνδέσεις μεταξύ αυτών. Κάθε γραμμή περιέχει ένα διάνυσμα που μεταφέρεται από την έξοδο ενός block στις εισόδους των επόμενων. Οι γραμμές που ενώνονται καταδεικνύουν τις συγχωνεύσεις και αυτές που διακλαδώνονται περιέχουν αντίγραφα της ίδιας πληροφορίας. Οι ροζ κύκλοι αφορούν τις πράξεις μεταξύ των διανυσμάτων και τα κίτρινα πλαίσια είναι διακριτά επίπεδα αναδρομικών δικτύων που χρησιμοποιούνται στην εκπαίδευση των LSTM και περιλαμβάνουν κάποιες συναρτήσεις, όπως η σιγμοειδής και η υπερβολική εφαπτομένη.

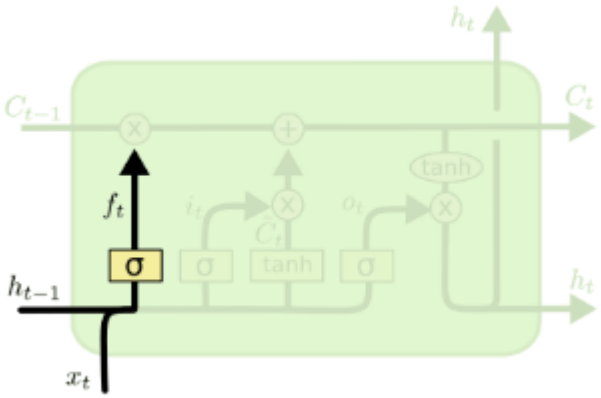


Σχήμα 5.2: Τα δομικά στοιχεία και οι συνδέσεις των μονάδων μνήμης στα LSTM,
Πηγή : **https://bit.ly/2JFrLgk**

Το βασικό στοιχείο των LSTM είναι η οριζόντια γραμμή που φαίνεται να διασχίζει το επάνω μέρος του διαγράμματος και απεικονίζει την κατάσταση των μονάδων της μνήμης τους. Μπορεί να τη συγκρίνει κανείς με έναν ιμάντα μεταφοράς που διασχίζει ολόκληρη την αλυσίδα των blocks και δεν έχει παρά μόνο λίγες γραμμικές αλληλεπιδράσεις με τα υπόλοιπα στοιχεία. Είναι πολύ εύκολο, δηλαδή, η περιεχόμενη πληροφορία να περάσει αναλλοίωτη. Η αλληλεπίδραση με τα υπόλοιπα στοιχεία γίνεται μέσω των πυλών που αναφέραμε προηγουμένως. Αυτές αποτελούνται από μια σιγμοειδή συνάρτηση, που παίρνει τιμές από 0 εώς 1 και μια πράξη πολλαπλασιασμού ή πρόσθεσης που αναλαμβάνει να προσθέσει την πληροφορία στις ήδη υπάρχουσες της μονάδας

μνήμης. Η τιμή της σιγμοειδούς συναρτήσεως καθορίζει το ποσοστό της πληροφορίας που θα περάσει για να προστεθεί στη μνήμη, με 0 να σημαίνει ότι δε θα επιτρέψει σε τίποτα να περάσει και με 1 η πληροφορία θα προσχωρήσει αυτούσια.

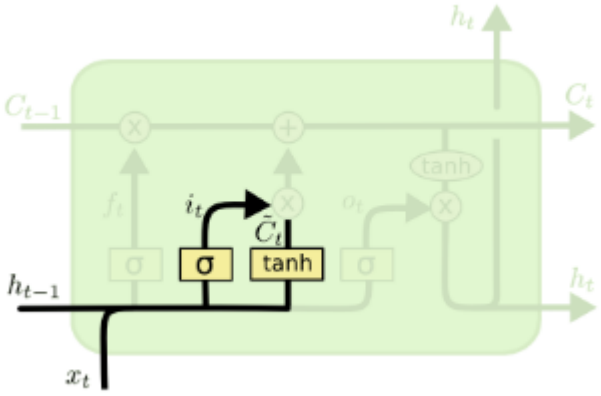
Το πρώτο βήμα στη λειτουργία του LSTM είναι να αποφασίσει ποιο μέρος της πληροφορίας θα αποδεσμεύσει από τη μνήμη. Υπεύθυνο για την απόφαση αυτή είναι το forget gate του συστήματος, το οποίο δέχεται την έξοδο του προηγούμενου επιπέδου h_{t-1} και την είσοδο x_t για να εξάγει μια τιμή από 0 εώς 1, μέσω της σιγμοειδούς. Η συνάρτηση που διέπει αυτή τη σχέση είναι η (5.15) και τα στοιχεία του δικτύου που είναι υπεύθυνα για αυτή τη λειτουργία φαίνονται στην Σχήμα (5.3). Στο παράδειγμα της πρόβλεψης της επόμενης λέξης σε μια ακολουθία ενός μοντέλου φυσικής γλώσσας, η διαδικασία αυτή θα μπορούσε να είναι υπεύθυνη για την επιλογή του γένους που θα χρησιμοποιηθεί, εφόσον θέλουμε να είναι σε θέση να ''ξεχάσει'' τα προηγούμενα δεδομένα όταν εμφανιστεί υποκείμενο διαφορετικού γένους.



Σχήμα 5.3: Πύλη αμνησίας στα LSTM,Πηγή : **https://bit.ly/2sUjk67**

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{5.15}$$

Το επόμενο βήμα είναι αυτό που θα πάρει την απόφαση για το ποια στοιχεία της νέας πληροφορίας θα συγκρατηθούν στη μνήμη του δικτύου. Η διαδικασία αυτή απαρτίζεται από δυο βήματα, με το πρώτο να περιλαμβάνει την πύλη εισόδου που ξεχωρίζει ποιες από τις υπάρχουσες πληροφορίες θα παραμείνουν στη μνήμη (5.16) και το δεύτερο να δημιουργεί ένα νέο διάνυσμα $\vec{C}_t$ με τις υποψήφιες τιμές που πρόκειται να προστεθούν σε αυτή (5.17).

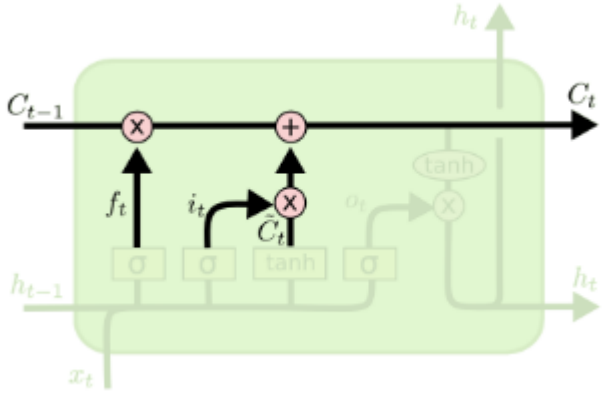


Σχήμα 5.4: Input Gate στα LSTM,Πηγή : **https://bit.ly/2sUjk67**.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{5.16}$$

$$\tilde{C}_t = \tanh\left(W_c[h_{t-1}, x_t] + b_c\right) \tag{5.17}$$

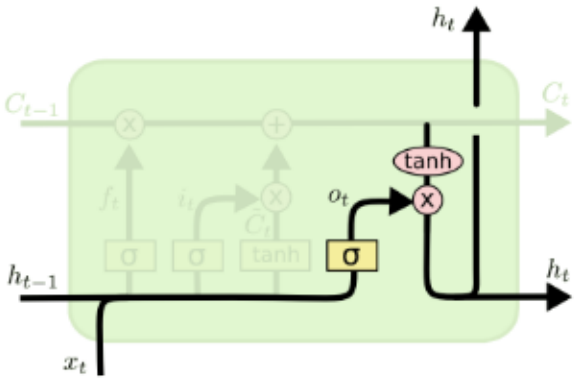
Στη συνέχεια γίνεται η ενημέρωση της κατάστασης της μνήμης από C_{t-1} σε C_t , πολλαπλασιάζοντας την παλιά κατάσταση C_{t-1} με τα δεδομένα της f_t που αποφασίστηκε να αποδεσμεύσουμε στο προηγούμενο βήμα και προσθέτοντας ένα ποσοστό των νέων υποψήφιων τιμών $i_t * \vec{C}_t$ (5.18). Σε αυτό το σημείο στην περίπτωση του παραδείγματος με το μοντέλο της φυσικής γλώσσας θα γινόταν η αφαίρεση των προηγούμενων πληροφοριών για το γένος του υποκειμένου και θα προσθέτονταν οι νέες



Σχήμα 5.5: Ενημέρωση των πληροφοριών της μνήμης στα LSTM,
Πηγή : **https://bit.ly/2sUjk67**

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{5.18}$$

Το τελευταίο βήμα αφορά την εξαγωγή του αποτελέσματος h_t , που θα γίνει είσοδος στο επόμενο επίπεδο και αποτελεί μια φιλτραρισμένη εκδοχή της κατάστασης της μνήμης. Αρχικά, γίνεται το πέρασμα της πληροφορίας εισόδου από μια σιγμοειδή συνάρτηση για να προσδιορισθεί ποιο κομμάτι αυτής θα προωθηθεί ως την έξοδο (5.19). Έπειτα εισέρχεται το περιεχόμενο της μνήμης μέσω της συνάρτησης της υπερβολικής εφαπτομένης (για να ωθήσει τις τιμές του διανύσματος στο διάστημα -1 εως 1) και πολλαπλασιάζεται με την έξοδο της σιγμοειδούς για να γίνει η προώθηση μόνο των κομματιών που έχουν προσδιορισθεί (5.20). Στο μοντέλο της φυσικής γλώσσας αυτή η διαδικασία θα μπορούσε να περιέχει τις πληροφορίες ενός ρήματος, όπως το αν βρίσκεται στον ενικό ή στον πληθυντικό, για να καθοριστεί η συνέχεια της πρότασης.



Σχήμα 5.6: Ενημέρωση των πληροφοριών της μνήμης στα LSTM,
Πηγή : **https://bit.ly/2sUjk67**

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{5.19}$$

$$h_t = o_t * \tanh(C_t) \tag{5.20}$$

Όσον αφορά τον αλγόριθμο εκμάθησης στα LSTM, όπως και στα υπόλοιπα RNN, υπάρχουν πολλές διαθέσιμες επιλογές [15]. Η πιο συνηθισμένη τεχνική είναι ο κατά προσέγγιση υπολογισμός της κλίσης του σφάλματος μέσω συνδυασμού του RTRL και του BPTT [16]. Το BPTT χρησιμοποιείται μόνο στους υπολογισμούς που λαμβάνουν χώρα στο πρώτο βήμα της εκπαίδευσης και έπειτα το ρόλο της μνήμης αναλαμβάνουν αποκλειστικά τα memory blocks του LSTM. Το κύριο πλεονέκτημα αυτής της διαδικασίας είναι η σύγχρονη ενημέρωση των βαρών με τη μέθοδο του RTRL, που την καθιστά κατάλληλη για εφαρμογές συνεχούς χρόνου ή για την πρόβλεψη χρονοσειρών. Ωστόσο, είναι δυνατός και ο υπολογισμός της κλίσης χωρίς να χρειαστεί να αφαιρέσουμε το κομμάτι του BPTT που παρουσιάζει καλύτερη ακρίβεια και έχει το πλεονέκτημα της ευκολίας αποσφαλμάτωσης σε σχέση με την προηγούμενη τεχνική.

Κεφάλαιο 6

Βάση mPower και μέτρα αξιολόγησης

Σε αυτό το Κεφάλαιο περιγράφονται αναλυτικά η βάση δεδομένων mPower, η δημιουργία δεδομένων των συνόλων εκπαίδευσης και δοκιμές καθώς και τα μέτρα αξιολόγησης των ταξινομητών.

6.1 Περιγραφή της βάσης mPower

Το ερευνητικό έργο mPower [19] αξιολογεί νέες προσεγγίσεις για την παρακολούθηση των βασικών δεικτών της εξέλιξης και της διάγνωσης της νόσου Parkinson, συμπληρώνοντας τις παραδοσιακές μετρήσεις συμπεριφορικών συμπτωμάτων με νέες μετρήσεις που συλλέγονται από κινητές συσκευές που διαθέτουν αισθητήρες. Ως κλιμακούμενη, ανέξοδη και μη επεμβατική μέθοδος για τη συχνή μέτρηση και παρακολούθηση των συμπτωμάτων, η εφαρμογή Parkinson mPower εξέτασε μια μεγάλη, διαχρονική ομάδα εθελοντών με PD και διαφόρους ελέγχους. Σκοπός της μελέτης είναι η κατανόηση της συχνότητας και του βαθμού μεταβολής των συμπτωμάτων των ασθενών, των πηγών αυτών των παραλλαγών και των πιθανών διαμορφωτών αυτών των παραλλαγών. Οι γνώσεις που αποκτήθηκαν μέσω αυτών των δεδομένων μπορούν να βοηθήσουν στην ανάπτυξη εξατομικευμένων παρεμβάσεων για τον μετριασμό της επιδείνωσης της νόσου.

Διατήθενται δεδομένα από επτά ενότητες.

1. **Δημογραφικά στοιχεία:** Σε αυτή την έρευνα οι συμμετέχοντες απάντησαν σε ερωτήσεις σχετικά με γενικά δημογραφικά θέματα και ιστορικό υγείας.

2. **MDS-UPDRS:** Σε αυτήν την έρευνα, οι συμμετέχοντες απάντησαν σε επιλεγμένες ερωτήσεις από την διεθνή οργάνωση κλινικής αξιολόγησης ασθενών με Parkinson (Movement Disorder Society'As Unified Parkinson'As Disease Rating Scale - MDS-UPDRS).

3. **PDQ-8:** Σε αυτή την έρευνα οι συμμετέχοντες απάντησαν στο σύντομο έντυπο του ερωτηματολογίου για την ασθένεια του Parkinson.

4. **Μνήμη:** Σε αυτή την ενότητα οι συμμετέχοντες ολοκληρώνουν ένα σύντομο παιχνίδι οπτικοακουστικών μέσων που σχετίζεται με τη δοκιμή κτυπήματος στο τετράγωνο Corsi [20].

5. **Χτύπηματα:** Σε αυτήν την ενότητα οι συμμετέχοντες επανειλημμένα χτυπούν την οθόνη του τηλεφώνου τους.

6. **Φωνή:** Σε αυτή την ενότητα οι συμμετέχοντες καταγράφουν πρώτα το επίπεδο θορύβου περιβάλλοντος για 5 δευτερόλεπτα και αν είναι αποδεκτό, καταγράφουν τους εαυτούς τους λέγοντας το φώνημα [aa] για 10 δευτερόλεπτα.

7. **Περπάτημα:** Σε αυτήν την ενότητα οι συμμετέχοντες περπατούν μπρος και πίσω για 20-30 δευτερόλεπτα με το κινητό τους στην τσέπη τους. Στη συνέχεια τους ζητείται να σταματήσουν για άλλα 20-30 δευτερόλεπτα.

Χρησιμοποιήθηκαν τα δεδομένα μόνο από την Ενότητα της Φωνής. Στα υπόλοιπα δεδομένα δεν υπήρχε πρόσβαση. Τα δείγματα δεδομένων φωνής συγκροτούν ένα τυχαίο δείγμα των πραγματικών δεδομένων mPower. Διατίθεται σε μορφή csv η περιγραφή των δεδομένων, με τις στήλες `healthCode` και `recordId` να παράγονται τυχαία. Ακολουθεί η περιγραφή των στηλών του csv:

- **recordId:** Μοναδικός αναγνωριστικός κωδικός για κάθε εγγραφή.
- **healthCode:** Μοναδικός αναγνωριστικός κωδικός για κάθε συμμετέχοντα.
- **createdOn:** Χρονική σήμανση από το τηλέφωνο όταν ολοκληρώθηκε μια συγκεκριμένη εργασία.
- **appVersion:** Έκδοση της εφαρμογής που χρησιμοποιείται για μια συγκεκριμένη εργασία.
- **phoneInfo:** Ο τύπος τηλεφώνου που χρησιμοποίησε ο συμμετέχων αφού ολοκληρωθεί μια συγκεκριμένη εργασία.
- **audio_audio.m4a:** Αρχείο τύπου m4a για εγγραφή φωνής.
- **medTimepoint:** Χρονικό σημείο φαρμακευτικής αγωγής.

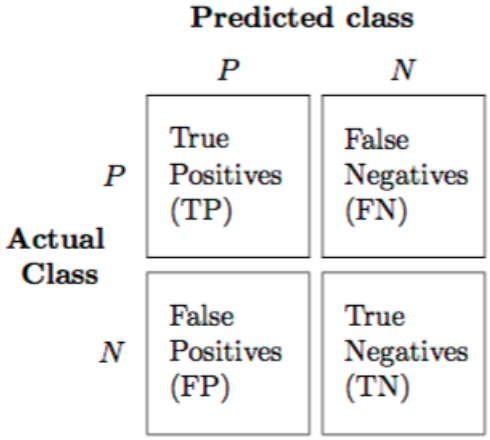
6.2 Μέτρα αξιολόγησης

Σε περιπτώσεις όπου οι κλάσεις δεν είναι ισομερώς κατανεμημένες ή όπου οι εσφαλμένες κατηγοριοποιήσεις διαφορετικών κλάσεων έχουν διαφορετικό κόστος, είναι σημαντική η εκτίμηση της ικανότητας πρόβλεψης του κατηγοριοποιητή για την κάθε κλάση. Για να εκτιμήσουμε τις ανά κλάση επιδόσεις ενός κατηγοριοποιητή, εισάγουμε την αναγκαία ορολογία. Για την περίπτωση μιας δυαδικής κλάσης όπως στην παρούσα πτυχιακή ισχύουν οι ακόλουθοι όροι:

- **Θετικές ταξινομήσεις (positive):** Είναι οι αποφάσεις του ταξινομητή που ανιχνεύουν οτι ο ομιλητής πάσχει από PD.

- **Αρνητικές ταξινομήσεις (negative):** Είναι οι αποφάσεις του ταξινομητή που ανιχνεύουν οτι ο ομιλητής δεν πάσχει από PD.
- **Αληθώς Θετικές Προβλέψεις (true positive - tp):** Είναι το πλήθος των επιτυχών προβλέψεων ότι ο ομιλητής έχει Parkinson και ο κατηγοριοποιητής ανιχνεύει ότι ο ομιλητής πάσχει από PD.
- **Αληθώς Αρνητικές Προβλέψεις (true negative - tn):** Είναι το πλήθος των επιτυχημένων προβλέψεων ότι ο ομιλητής δεν πάσχει από Parkinson και ο κατηγοριοποιητής ανιχνεύει ότι ο ομιλητής δεν πάσχει από PD.
- **Ψευδώς Θετικές Προβλέψεις (false positive - fp):** Είναι το πλήθος των αποτυχημένων προβλέψεων ότι ο ομιλητής μολονότι δεν πάσχει Parkinson ο κατηγοριοποιητής όμως τον ανιχνεύει ως ομιλητή που πάσχει από PD.
- **Ψευδώς Αρνητικές Προβλέψεις (false negative - fn):** Είναι το πλήθος των αποτυχημένων προβλέψεων ότι μολονότι ο ομιλητής έχει Parkinson , ο κατηγοριοποιητής αποτυγχάνει να τον ανιχνεύσει ως τέτοιο.

6.2.1 Πίνακας συγχύσεων - Confusion Matrix



Σχήμα 6.1: Πίνακας συγχύσεων 2 κλάσεων, Πηγή : **https://bit.ly/2MrdhPu**

Ένας τρόπος παρουσίασης των επιδόσεων ανά κλάση ενός κατηγοριοποιητή είναι με τη χρήση του πίνακα συγχύσεων (confusion matrix) [21]. Ο πίνακας συγχύσεων είναι ένας δισδιάστατος πίνακας, όπου οι στήλες αντιστοιχούν στις προβλέψεις του ταξινομητή και οι γραμμές στις πραγματικές τιμές κλάσης σύμφωνα με την βάσιμη αλήθεια. Στα κελιά του πίνακα αναγράφονται οι αληθώς θετικές, οι αληθώς αρνητικές, οι ψευδώς θετικές και οι ψευδώς αρνητικές προβλέψεις. Στο Σχήμα 6.1 απεικονίζεται ένας πίνακας συγχύσεων.
Ορισμένα πρόσθετα μέτρα για τις επιδόσεις ενός κατηγοριοποιητή είναι τα ακόλουθα:

- Ευαισθησία (*sensitivity*)

$$sensitivity = \frac{tp}{tp + fn} \tag{6.1}$$

- Εξειδίκευση ($specifity$)

$$specifity = \frac{tn}{tn + fp} \tag{6.2}$$

- Ακρίβεια ($precision$)

$$precision = \frac{tp}{tp + fp} \tag{6.3}$$

- Ανάκληση ($recall$)

$$recall = \frac{tp}{tp + fn} \tag{6.4}$$

Σύμφωνα με τα παραπάνω η πιθανότητα ορθής ταξινόμησης (accuracy) ορίζεται ως το ποσοστό των ορθών θετικών προβλέψεων επί το ποσοστό των θετικών παρατηρήσεων συν το ποσοστό των ορθών αρνητικών προβλέψεων επί το ποσοστό των αρνητικών παρατηρήσεων ή ισοδύναμα ως το πλήθος των ορθών προβλέψεων προς το πλήθος των παρατηρήσεων.

$$accuracy = sensitivity \cdot \frac{pos}{pos + negat} + specifity \cdot \frac{neg}{pos + negat} = \frac{tp + tn}{pos + negat} \tag{6.5}$$

$$\text{όπου,} \qquad pos = tp + fn, \qquad \text{και} \qquad negat = tn + fp$$

6.2.2 Βαθμολογία F1 (F-score)

Στη στατιστική ανάλυση της δυαδικής ταξινόμησης, η βαθμολογία F1 (επίσης βαθμολογία F ή μέτρο F) αποτελεί μέτρο της ακρίβειας της δοκιμής [22]. Λαμβάνει υπόψη τόσο την ακρίβεια ($precision$) όσο και την ανάκληση ($recall$) της δοκιμής για τον υπολογισμό της βαθμολογίας. Η βαθμολογία F1 είναι ο αρμονικός μέσος όρος της ακρίβειας και της ανάκλησης, όπου η βαθμολογία F1 φτάνει την καλύτερη τιμή της στο 1 (τέλεια ακρίβεια και ανάκληση) και χειρότερη στο 0.

$$F_1 = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{6.6}$$

6.2.3 Χαρακτηριστική Καμπύλη Λειτουργίας Δέκτη ROC (Receiver Operating Characteristics)

Ένα ισχυρό μέτρο για την εκτίμηση της ανά κλάση ακρίβειας είναι οι λεγόμενες χαρακτηριστικές καμπύλες λειτουργίας δέκτη ή αλλιώς καμπύλες ROC [23]. Οι καμπύλες ROC σχεδιάζονται σε ένα δισδιάστατο επίπεδο χώρο. Ο οριζόντιος άξονας εκφράζει το μέγεθος $1 - specifity$ το οποίο ονομάζεται False Positive Rate ή False Acceptance Rate.

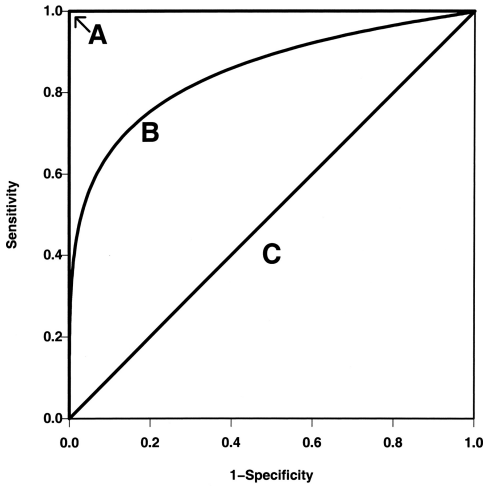
$$False\_Positive\_Rate = 1 - specifity = \frac{fp}{negat} \tag{6.7}$$

Ο κατακόρυφος άξονας εκφράζει το μέγεθος $sensitivity$ το οποίο ονομάζεται True Positive Rate.

$$True\_Positive\_Rate = sensitivity = \frac{tp}{pos} \tag{6.8}$$

Ουσιαστικά, ο οριζόντιος άξονας εκφράζει το ποσοστό των υγιών ατόμων, οι οποίοι ανιχνεύθηκαν ως ασθενής από την επεξεργασία του σήματος της ομιλίας και πάσχουν όντως απο PD

και ο κατακόρυφος άξονας εκφράζει το ποσοστό των ασθενών, οι οποίοι ανιχνεύθηκαν σωστά ως τέτοιοι απο την επεξεργασία του σήματος της ομιλίας. Το Σχήμα 6.2 απεικονίζει τον δισδιάστατο χώρο καμπυλών ROC . Κάθε σημειο του χώρου αυτού εκφράζει ένα ισοζύγιο ανάμεσα στο ποσοστό ορθών θετικών προβλέψεων και εσφαλμένων θετικών προβλέψεων. Το σημείο 0,0 είναι ένας κατηγοριοποιητής, που δεν προβλέπει ποτέ θετική παρατήρηση. Το σημείο (1,1) είναι ένας κατηγοριοποιητής, που προβλέπει πάντα θετική παρατήρηση. Η διαγώνια γραμμή C , από το σημείο (0,0) στο σημείο (1,1) είναι ένας κατηγοριοποιητής που προβλέπει τυχαία την κλάση. Οι κατηγοριοποιητές που βρίσκονται κάτω από τη διαγώνια γραμμή είναι χειρότεροι από την τυχαία πρόβλεψη. Οι κατηγοριοποιητές που βρίσκονται πάνω από τη διαγώνια γραμμή είναι καλύτεροι από την τυχαία πρόβλεψη (π.χ. B). Το σημείο A(0,1) είναι ο άριστος κατηγοριοποιητής, ο οποίος προβλέπει σωστά όλες τις θετικές και αρνητικές παρατηρήσεις. Γενικώς, όσο πιο μετατοπισμένο είναι προς τα επάνω και προς τα αριστερά ένα σημείο, τόσο καλύτερη θεωρείται η επίδοση.



Σχήμα 6.2: Δισδιάστατο χώρο καμπυλών ROC , Πηγή : **https://bit.ly/2JS87xK**

Η επίδοση των κατηγοριοποιητών στον χώρο ROC συμβολίζεται με μία καμπύλη. Για να συγκρίνουμε κατηγοριοποιητές χρειαζόμαστε ένα μέτρο σύγκρισης. Τέτοιο μέτρο σύγκρισης είναι το Εμβαδόν Κάτω από την Καμπύλη ROC (Area Under Curve - (AUC)). Η AUC εκφράζει το ποσοστό του χώρου που βρίσκεται κάτω από την καμπύλη, και παίρνει τιμές από 0 έως 1. Η διαγώνια γραμμή τυχαίας πρόβλεψης έχει AUC = 0.5. Συνεπώς, κάθε κατηγοριοποιητής καλύτερος της τυχαίας πρόβλεψης έχει AUC $>$ 0.5. Όσο μεγαλύτερη περιοχή AUC έχει ένας κατηγοριοποιητής τόσο καλύτερος είναι.

Κεφάλαιο 7

Υλοποίηση και Αποτελέσματα

Στην παρούσα πτυχιακή εργασία δημιουργήθηκε ένα ολοκληρωμένο σύστημα ανίχνευσης της νόσου Parkinson μέσω ενός σήματος ομιλίας. Κατά τη διάρκεια της υλοποίησης, πραγματοποιήθηκαν πολλά πειράματα εκπαίδευσης διαφορετικών αρχιτεκτονικών τα οποία εκπαιδεύτηκαν σε διάφορες παραλλαγές της αρχικής βάσης δεδομένων. Στα πειράματα αυτά χρησιμοποιούνται μέθοδοι και παράγοντες που παίζουν ρόλο στην εκμάθηση του νευρωνικού δικτύου, οι οποίοι έχουν περιγραφεί στα προηγούμενα κεφάλαια και είναι βελτιστοποιημένοι ως προς την καλύτερη απόδοση και συμπεριφορά αυτού. Για το σκοπό αυτό, δηλαδή για τον καταλληλότερο προσδιορισμό των μεταβλητών που έχουν ενεργό ρόλο στην εκπαίδευση, έχουν προηγηθεί αρκετές προηγούμενες προσπάθειες εύρεσης των καταλληλότερων τιμών.

7.1 Υλοποίηση

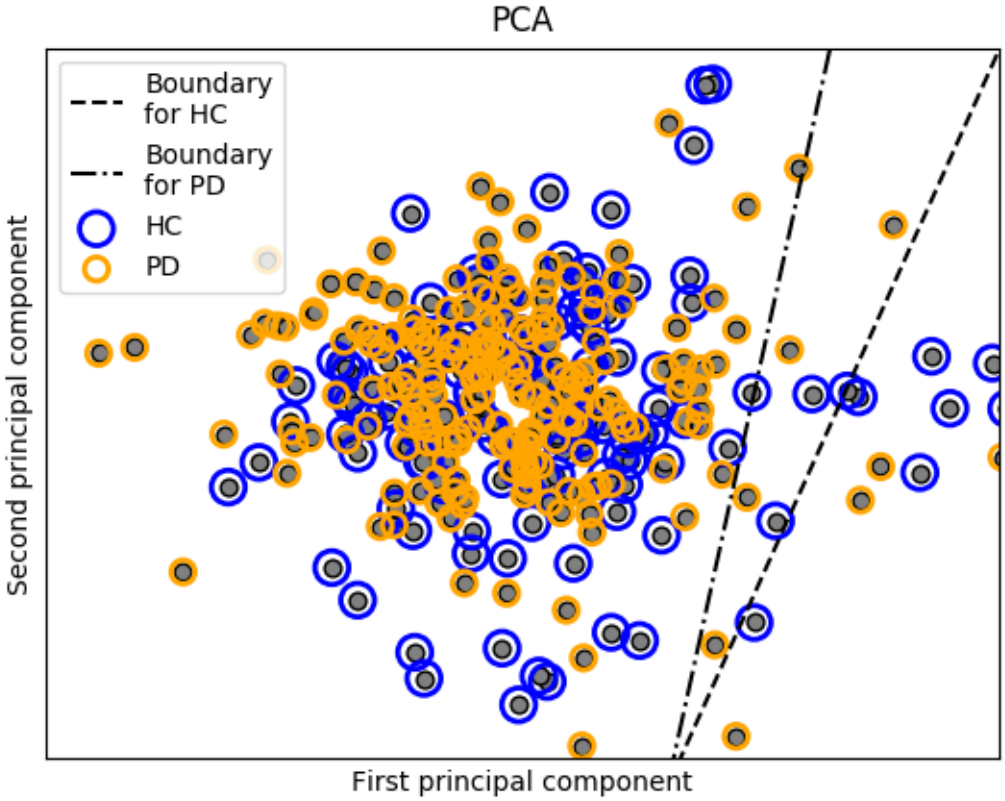
7.1.1 Το εργαλείο εκμάθησης Keras

Η παρούσα πτυχιακή υλοποιήθηκε στο περιβάλον του Keras. Το Kerasείναι μια βιβλιοθήκη νευρωνικών δικτύων ανοιχτού κώδικα γραμμένη στη Python. Είναι σε θέση να τρέχει πάνω από τη TensorFlow. Σχεδιασμένο για γρήγορο πειραματισμό με βαθιά νευρωνικά δίκτυα, εστιάζει στο να είναι φιλικό προς το χρήστη, αρθρωτό και επεκτάσιμο. Αναπτύχθηκε ως μέρος της ερευνητικής προσπάθειας του project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) και ο κύριος συγγραφέας και συντηρητής του είναι ο François Chollet, μηχανικός της Google. Περιέχει πολλές εφαρμογές των δομικών στοιχείων που χρησιμοποιούνται συνήθως σε νευρωνικά δίκτυα, όπως στρώματα, στόχοι, λειτουργίες ενεργοποίησης, βελτιστοποιητές και πλήθος εργαλείων που διευκολύνουν την εργασία με δεδομένα εικόνας και κειμένου. Ο κώδικας του είναι αναρτημένος στο GitHub και υπάρχουν πολλά φόρουμ υποστίριξης του. Αναπτύχθηκε με έμφαση στο γρήγορο πειραματισμό. Επιτρέπει την εύκολη και γρήγορη δημιουργία πρωτοτύπων χάρη στη φιλικότητα προς το χρήστη. Υποστηρίζει τόσο συνελικτικά δίκτυα όσο και αναδρομικά δίκτυα, καθώς και συνδυασμούς των δύο. Προσφέρει εύκολη επεκτασιμότητα καθώς οι νέες ενότητες είναι απλές να προστεθούν (ως νέες τάξεις και λειτουργίες) και οι υπάρχουσες μονάδες παρέχουν άφθονα παραδείγματα. Επιτρέπει τέλος, την πλήρη εκφραστικότητα, καθιστώντας το κατάλληλο για προηγμένη έρευνα.

7.1.2 Δημιουργία της βάσης δεδομένων

Η βάση περιλαμβάνει 5826 ηχητικά δεδομένα του φωνημάτος [aa] με πολλές επαναλήψεις των ίδιων ατόμων. Υπήρξε ένα πρόβλημα καθώς πολλά αρχεία περιήχαν θόρυβο από εξωτερικούς παράγοντες (π.χ. ήχοι ζώων και ανθρώπων). Τα ηχητικά δεδομένα ήταν κωδικοποιημένα με AAC (m4a) κωδικοποίηση και χρειάστηκε να μετατραπούν σε WAV για την ευκολότερη επεξεργασία τους. Η AAC και WAV μορφή χρησιμοποιεί κωδικοποίηση η οποία συμπιέζει το ψηφιακό σήμα. Για την δημιουργία της τελικής βάσης που χρησιμοποιήθηκε, έπρεπε να μειωθεί η συχνότητα δειγματοληψίας του κάθε ηχητικού αρχείου με downsampling έτσι ώστε να μειωθεί παράλληλα και ο όγκος των δεδομένων.Η αρχική συχνότητα δειγματοληψίας ήταν 44100 Hz και μετά το downsampling έγινε 16000 Hz. Έπειτα, οργανώνονται το σύνολο αρχεία σε 2 κατηγορίες: PD (Parkinson Diseased) και HC (Healthy Components). Γίνεται η εξαγωγή χαρακτηριστικών MFCC (20, 430), όπου 20 το πλήθος των χαρκατηριστικών και 430 το πλήθος των διαστημάτων που εξάγονται απο μία ηχητική καταγραφή. Στη συνέχεια γίνεται η εξαγωγή χαρακτηριστικών Delta και Delta-Delta όπως περιγράφηκε στην Ενότητα 2.4.2. Τα χαρακτηριστικά αποθηκεύονται σε μεμονωμένους πίνακες Numpy δύο διαστάσεων μεγέθους (60, 430) για ευκολότερη επεξεργασία και χρήση στον κώδικα.

Το σύνολο των δεδομένων που χρησιμοποιήθηκε ήταν 836 πίνακες Numpy, ο κάθε πίνακας συγκεντρώνει τα διανύσματα χαρακτηριστικών που εξήχθησαν από μία ηχητική καταγραφή που εφεξής αναφέρεται στο δείγμα. Τα 278 δείγματα είναι διαγνωσμένα με Parkinson και 556 να προέρχονται από υγιείς ομιλητές. Επαναλήψεις του ιδίου ατόμου δεν υπάρχουν. Από το σύνολό των δειγμάτων, το 80% χρησιμοποιήθηκε ως σύνολο εκπαίδευσης και το υπόλοιπο 20% χρησιμοποιήθηκε ως σύνολο δοκιμής. Τέλος έγινε μια αναπαράσταση των πινάκων Numpy με την τεχνική PCA [24] για να ρίξουμε τις διαστάσεις των διανυσμάτων στο δισδιάστατο χώρο. Παρατηρήθηκε ότι δεν υπήρχε μεγάλη διαχωρισιμότητα πράγμα που καθιστά το πρόβλημα προς επίλυση πολύ πιο περίπλοκο.



Σχήμα 7.1: PCA των 2 κλάσεων

7.2 Υλοποιημένα Δίκτυα

Η βασική ιδέα των πειραμάτων περιλαμβάνει την υλοποίηση δικτύων για την κλασικοποίηση (Classification) και την πρόβλεψη (Regression) νέων εγγραφών. Υλοποιήθηκαν τα εξείς δίκτυα:

1. Ένα βαθύ συνελικτικό νευρωνικό δίκτυο (Deep Convolutional Neural Network - deepCNN)
2. Ένα συνελικτικό νευρωνικό δίκτυο ενωμένο με ενα αναδρομικό νευρωνικό δίκτυο (Convolutional Recurent Neural Network - CRNN)
3. Ένα βαθύ αναδρομικό νευρωνικό δίκτυο χρησιμοποιώντας Δίκτυα Μακράς Βραχυχρόνιας Μνήμης (Deep Long Short Term Memory - deepLSTM)
4. Ένα υπολειμματικό νευρωνικό δίκτυο Residual neural network - resNET

7.2.1 Deep Convolutional Neural Network

Δεδομένης μιας ακολουθίας ακουστικών χαρακτηριστικών $\mathbf{X} \epsilon \Re^{c \cdot m \cdot n}$ με τον αριθμό των καναλιών $c = 1$, το πλάτος του κατά μήκος του άξονα συχνοτήτων ίσο με $m = 60$ και το μήκος

του κατά μήκος του άξονα του πλαισίου ίσο με το $n = 430$, το επίπεδο συνέλιξης συνελύεται με $k = 60$ φίλτρα $\left\{\mathbf{W}_i\right\}_k$ με το αποτέλεσμα να υπολογίζεται:

$$\mathbf{H}_i = \mathbf{W}_i * X + b_i, \quad i = 1, \ldots, k. \tag{7.1}$$

Όπου * το σύμβολο της συνέλιξης και b_i το βήμα.

Το δίκτυο αποτελείται απο 5 συνελικτικά μπλοκ και 2 πλήρως συνδεδεμένα επίπεδα στο τέλος, χρησιμοποιεί σαν συνάρτηση σφάλματος την binary crossentropy και σαν βελτιστοποιητή τον Adam με βαθμό εκμάθησης 0.004. Κάθε μπλοκ αποτελείται από ένα επίπεδο συνέλιξης με 60 φίλτρα (όσο και το πλήθος των features), με μία συνάρτηση ενεργοποίησης και ένα επίπεδο κανονικοποίησης. Στο πρώτο μπλοκ εισάγεται στο επίπεδο συνέλιξης το διάνυσμα εισόδου που περιγράφηκε από την σχέση (7.1), στην συνέχεια υπάρχει το επίπεδο κανονικοποίησης και η συνάρτηση ενεργοποίησης relu. Τα επόμενα 4 συνελικτικα μπλοκ είναι ίδια και αποτελούνται από το επίπεδο συνέλιξης, το επίπεδο κανονικοποίησης, την συνάρτηση ενεργοποίησης ELU (Exponential Linear Unit), ένα (2x2) επίπεδο υποδειγματοληψίας, και ένα επίπεδο απόσυρσης με πιθανότητα απόσυρσης 0.2. Τα 2 τελευταία πλήρως συνδεδεμένα επίπεδα αποτελούνται από: 128 νευρώνες με συνάρτηση ενεργοποίησης relu το προτελευταίο και από 2 νευρώνες με συνάρτηση ενεργοποίησης την softmax το τελευταίο.

7.2.2 Convolutional Recurent Neural Network

Το δίκτυο αποτελείται απο 1 συνελικτικό μπλοκ, 4 αναδρομικά επίπεδα και 2 πλήρως συνδεδεμένα επίπεδα με αυτήν την σειρά. Χρησιμοποιεί σαν συνάρτηση σφάλματος την binary crossentropy και σαν βελτιστοποιητή τον Adam με βαθμό εκμάθησης 0.004. Το πρώτο μπλοκ αποτελείται από ένα επίπεδο συνέλιξης με 60 φίλτρα (όσο και το πλήθος των features), με συνάρτηση ενεργοποίησης relu και ένα επίπεδο κανονικοποίησης. Στο μπλοκ αυτό εισάγεται στο επίπεδο συνέλιξης το διάνυσμα εισόδου που περιγράφηκε από την σχέση (7.1). Τα επόμενα 4 αναδρομικά επίπεδα LSTM απαρτίζονται απο 50 μονάδες μνήμης το καθένα και με εσωτερική συνάρτηση ενεργοποίησης την υπερβολική εφαπτομένη tanh. Τέλος τα 2 τελευταία πλήρως συνδεδεμένα επίπεδα αποτελούνται από: 128 νευρώνες με συνάρτηση ενεργοποίησης relu το προτελευταίο και από 2 νευρώνες με συνάρτηση ενεργοποίησης την softmax το τελευταίο μαζί με ένα επίπεδο απόσυρσης ανάμεσα τους με ποσοστό απόσυρσης 30% .

7.2.3 Deep Long Short Term Memory

Δεδομένης μιας ακολουθίας ακουστικών χαρακτηριστικών $\mathbf{X}\ \epsilon\ \Re^{m \cdot n}$, με το πλάτος του κατά μήκος του άξονα συχνοτήτων ίσο με $m = 60$ και το μήκος του κατά μήκος του άξονα του πλαισίου ίσο με το $n = 430$, το αναδρομικό δίκτυο υπολογίζει το κρυμμένο διάνυσμα συχνοτήτων $\mathbf{h} = (h_1, \ldots, h_T)$ και την ακολουθία του φορέα εξόδου $\mathbf{y} = (y_1, ..., y_T)$ με την ερμηνεία των ακόλουθων εξισώσεων από $t = 1$ έως T:

$$h_t = H(W_{xh}\, x_t + W_{hh}\, h_{t-1}) + b_h, \tag{7.2}$$

$$y_t = W_{hy}\, h_t + b_y, \tag{7.3}$$

όπου W ο πίνακας βαρών (π.χ. $W_{x\,h}$ ο πίνακας βαρών της κρυμμένης εισόδου), b το βήμα, και H η συνάρτηση του κρυμμένου επιπέδου.

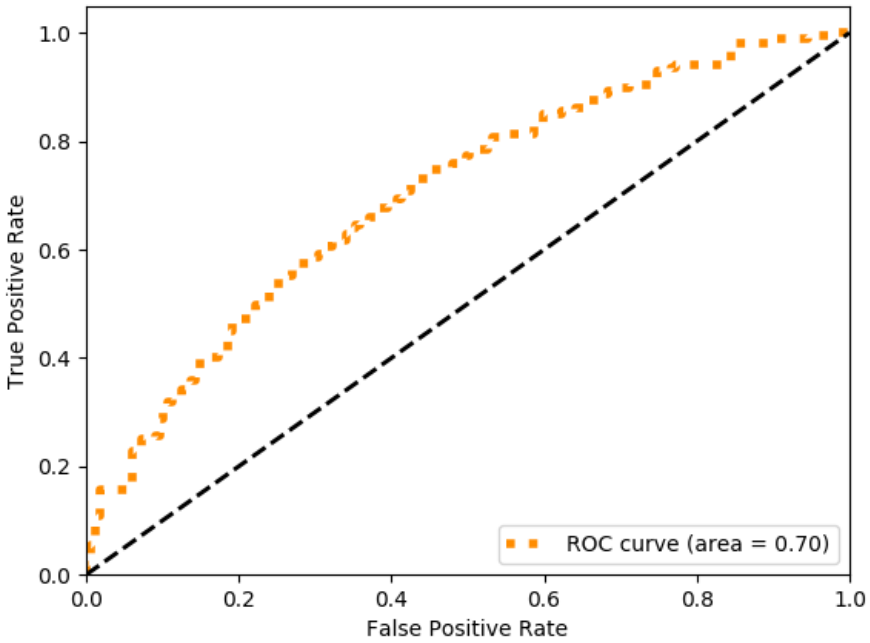
Το δίκτυο αποτελείται από 4 αναδρομικά μπλοκ και 1 πλήρως συνδεδεμένο επίπεδο 2 εξόδων. Το πρώτο μπλοκ δέχεται την ακολουθία εισόδου $\mathbf{X}$. Κάθε αναδρομικό μπλοκ αποτελείται από ένα LSTM επίπεδο 60 μονάδες μνήμης συνδεδεμένες αλυσιδωτά μεταξύ τους και με εσωτερική συνάρτηση ενεργοποίησης την υπερβολική εφαπτομένη tanh, ένα επίπεδο κανονικοποίησης, και ένα επίπεδο απόσυρσης με ποσοστό απόσυρσης 20%. Το τελευταίο επίπεδο αποτελείται από 2 νευρώνες εξόδου με συνάρτηση ενεργοποίσης την softmax. Το δίκτυο χρησιμοποιεί σαν συνάρτηση σφάλματος την binary crossentropy και σαν βελτιστοποιητή τον Adam με βαθμό εκμάθησης 0.004.

7.3 Αποτελέσματα

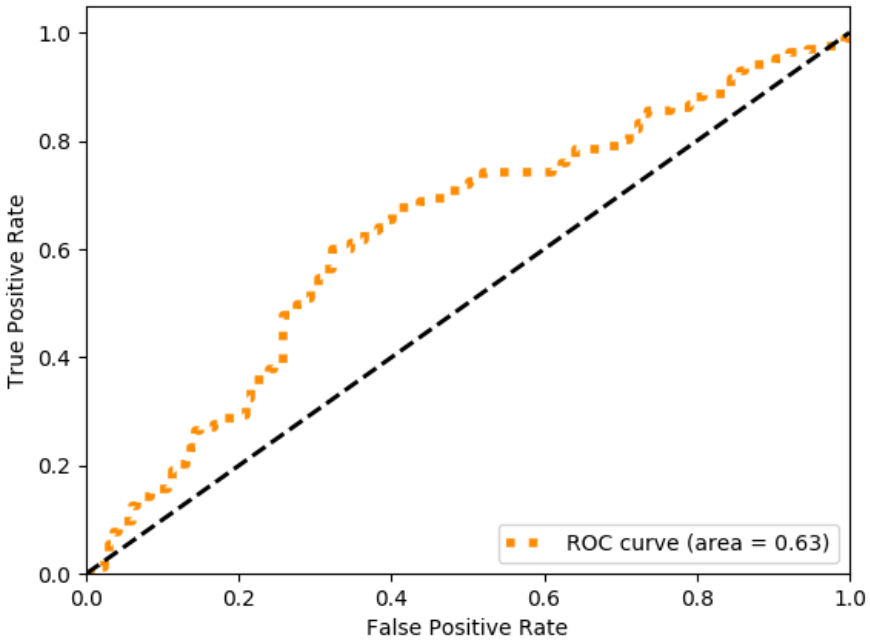
Σε αυτήν την Ενότητα παρουσιάζονται τα αποτελέσματα των εκτελέσεων των νευρωνικών δικτύων που περιγράφηκαν. Η εκπαίδευση έγινε σε 15 εποχές καθώς ο χρόνος που χρειαζόταν για την κάθε εποχή ήταν πολύ μεγάλος λόγο του όγκου των δεδομένων.

Evaluation				
Model	deepCNN	deepLSTM	CRNN	resNET
Fscore	0.645	0.656	0.701	0.713
AUC	0.632	0.614	0.699	0.731

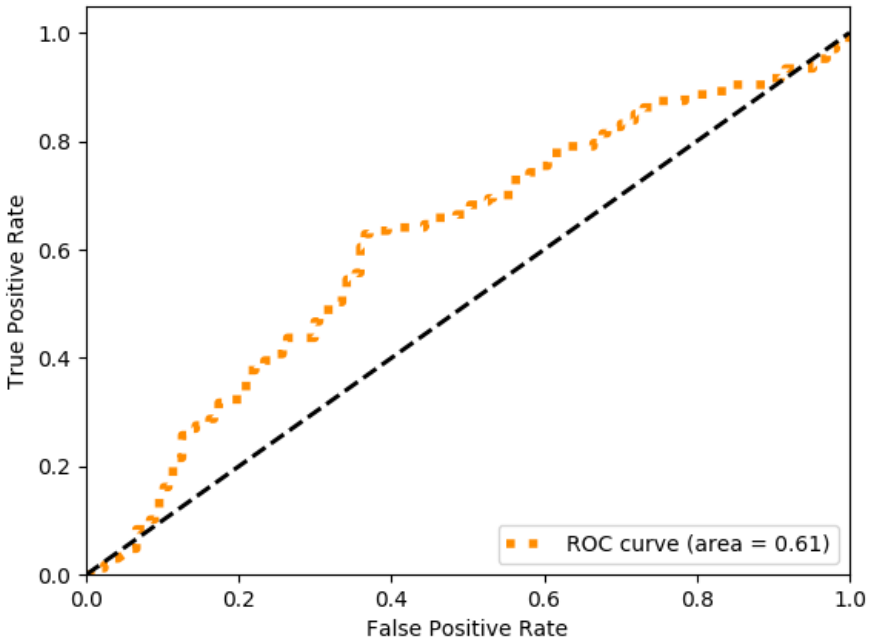
Ακολουθούν οι καμπύλες ROC για το CRNN (Σχήμα 7.2), για το deepCNN (Σχήμα 7.3), για το deepLSTM (Σχήμα 7.4) και για το resNET (Σχήμα 7.5):



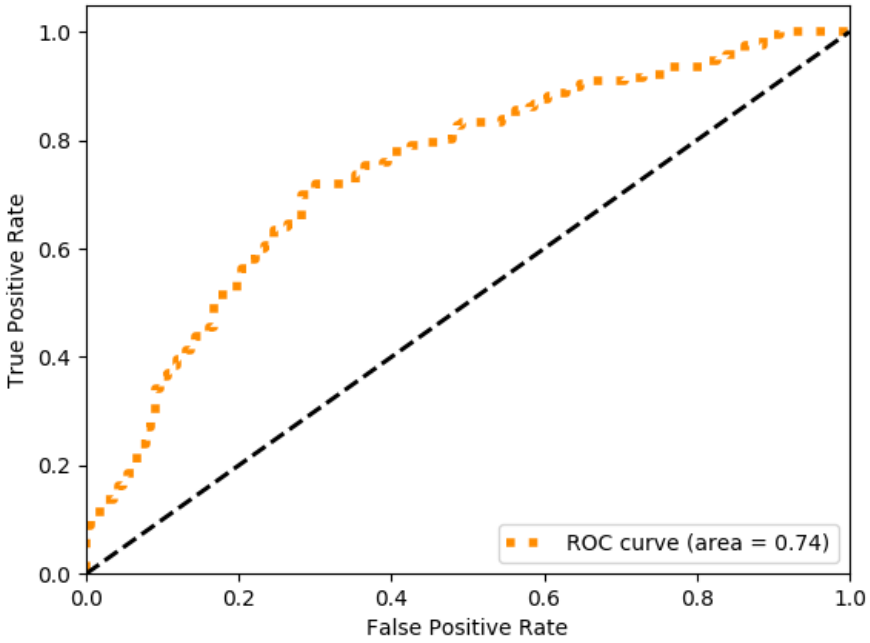
Σχήμα 7.2: Καμπύλη ROC του δικτύου CRNN.



Σχήμα 7.3: Καμπύλη ROC του δικτύου deepCNN.

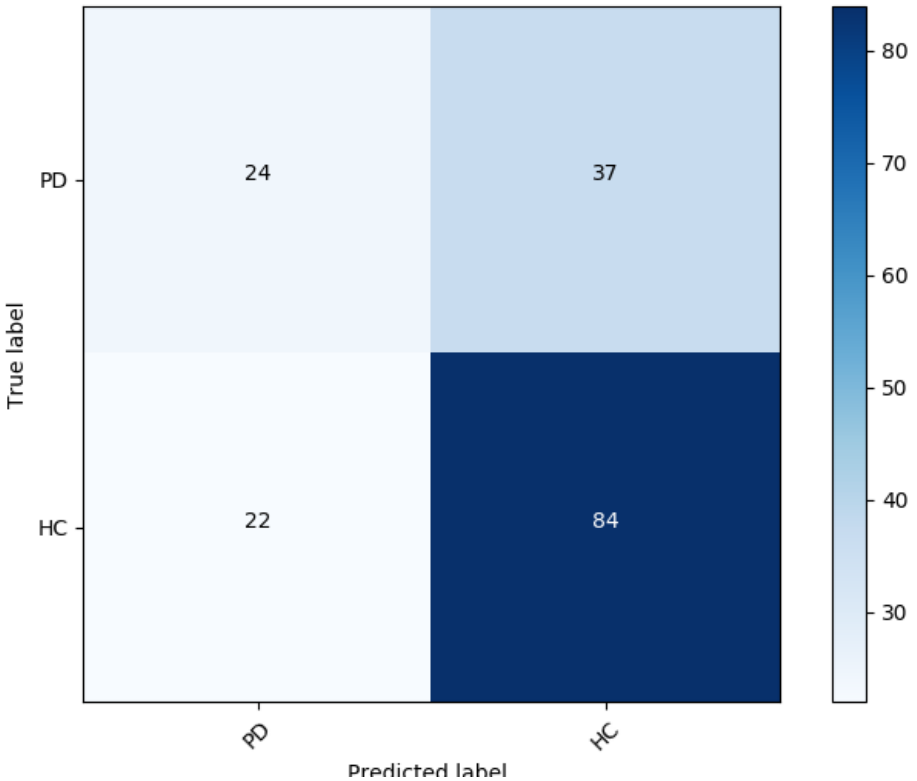


Σχήμα 7.4: Καμπύλη ROC του δικτύου deepLSTM.

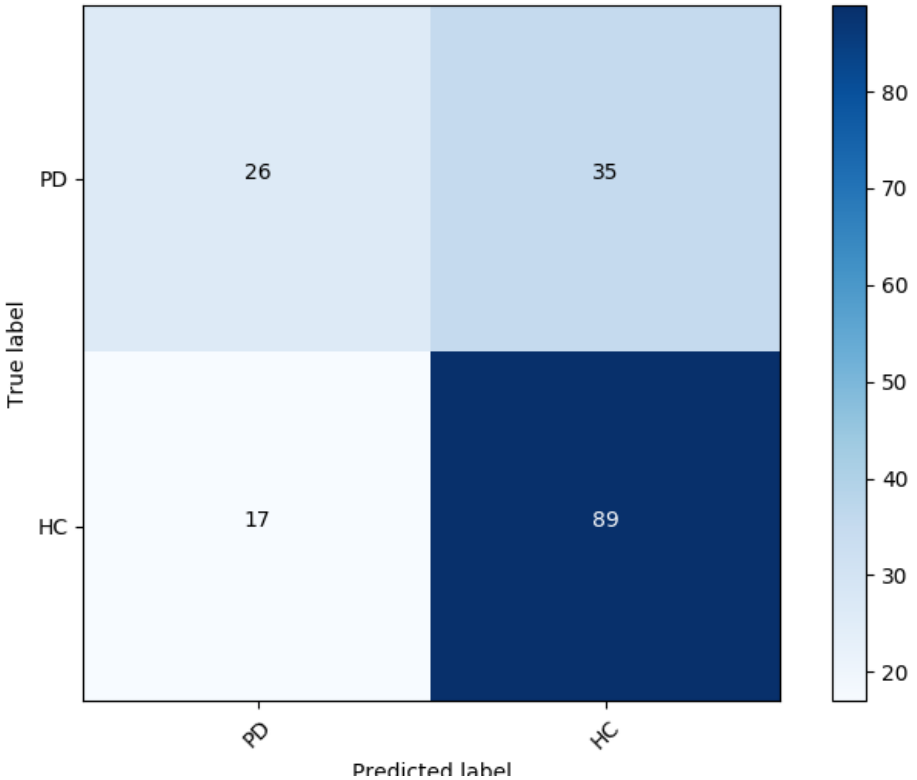


Σχήμα 7.5: Καμπύλη ROC του δικτύου resNET.

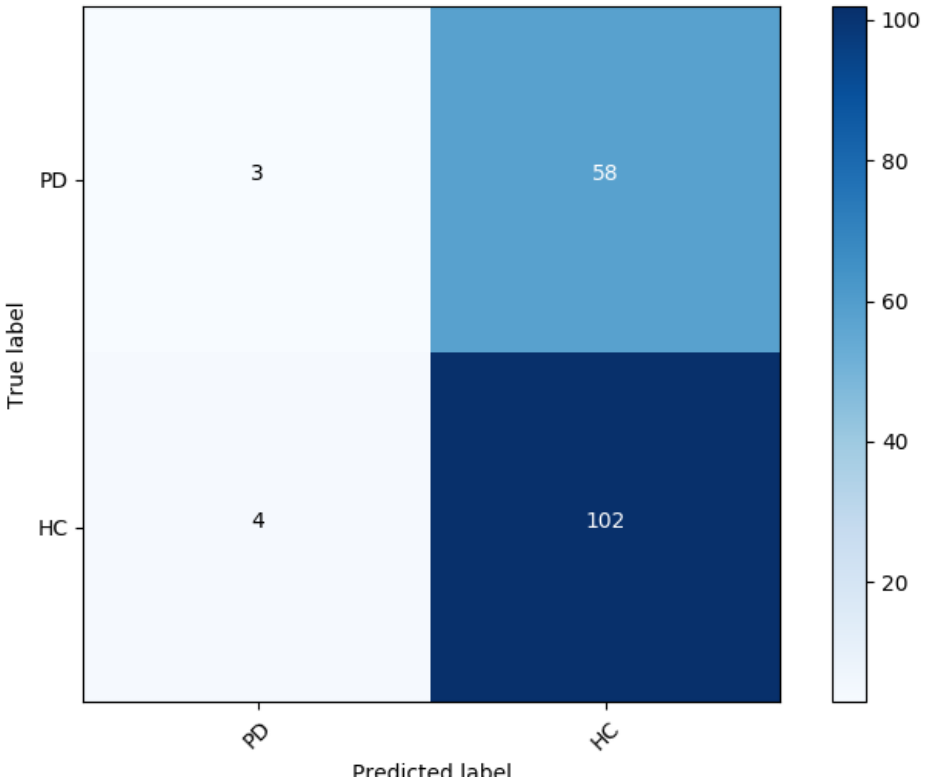
Ακόλουθούν οι πίνακες συγχύσεως για το CRNN (Σχήμα 7.6), για το deepCNN (Σχήμα 7.7), για το deepLSTM (Σχήμα 7.8) και για το resNET (Σχήμα 7.9):



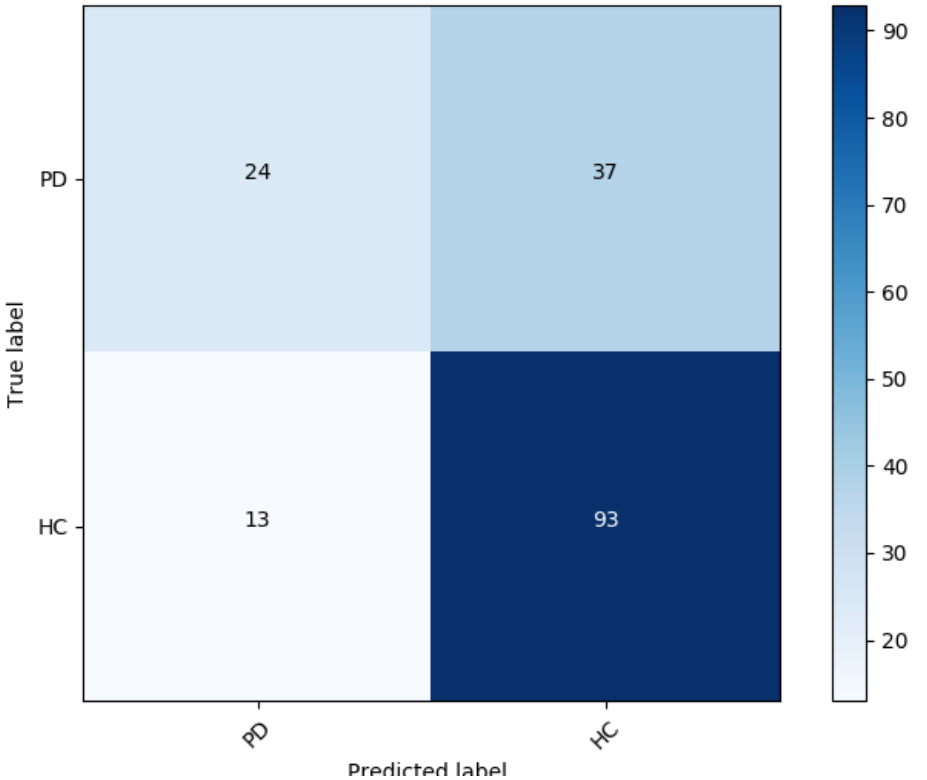
Σχήμα 7.6: Confusion Matrix των 2 κλάσεων για το CRNN.



Σχήμα 7.7: Confusion Matrix των 2 κλάσεων για το deepCNN.

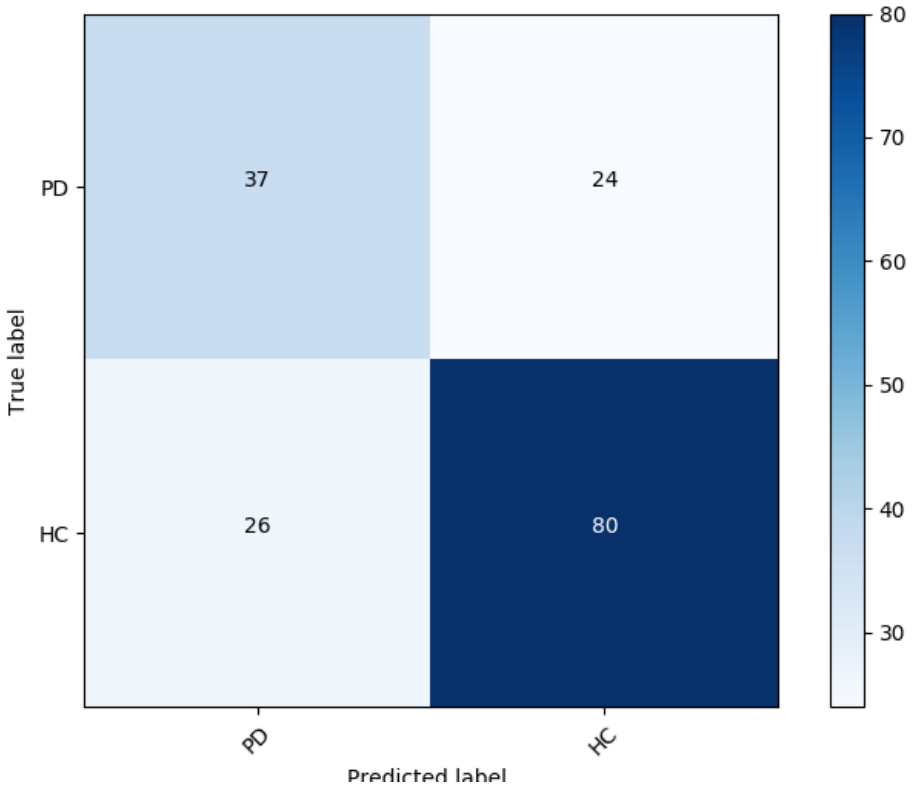


Σχήμα 7.8: Confusion Matrix των 2 κλάσεων για το deepLSTM.



Σχήμα 7.9: Confusion Matrix των 2 κλάσεων για το resNET.

Χρησιμοποιώντας τον κανόνα της πλειοψηφία (Majority Voting) για τις προβλέψεις των δικτύων: deepLSTM , resNET και CRNN παράγουμε τον τελικό πίνακα συγχύσεως Σχήμα 7.10 όπου είναι εμφανές ότι έχουμε βελτιωμένες προβλέψεις.



Σχήμα 7.10: Confusion Matrix των 2 κλάσεων μετά από Majority Voting των δικτύων: deepLSTM , resNET και CRNN.

7.3.1 Συμπεράσματα - Μελλοντική δουλειά

Το πρόβλημα για το οποίο έγινε προσπάθεια να επιλυθεί ήταν πολύ περίπλοκο καθώς προσπαθούμε να διαχωρήσουμε σε 2 ακουστικών καταγραφών απο ασθενείς με PD και υγιείς, οι οποίες αλληλοεπικαλύπτονται. Όπως είχε προαναφερθεί δεν είναι γραμμικά διαχωρίσιμο στις 2 διαστάσεις. Παρόλο που τα δίκτυα εκπαιδεύτηκαν με εξαντλητικό τρόπο δεν καταφέρνουν να ρυθμίσουν τα βάρη τους ώστε να ταξινομήσουν σωστά τις παρατηρήσεις με ικανοποιητικά αποτελέμσατα. Παρατηρήθηκε πολλές φορές υπερεκπαίδευση ως προς την κλάση HC (πράγμα λογικό καθώς αποτελούν τα 2/3 της βάσης) και σπάνια ως προς την κλάση PD. Καθώς τα αποτελέσματα δεν είναι πολύ ενθαρυντικά με αυτή την προσσέγγιση των δεδομένων καταλήγουμε στο συμπέρασμα ότι χρειάζονται αλλαγές στην προεπεξεργασία των δεδομένων, στον τρόπο εξαγωγής των ακουστικών χαρακτηριστικών και την υλοποίηση των δικτύων.

Στο σημείο αυτό προτείνονται ιδέες και τρόποι με τους οποίους μπορεί να υπάρξει συνέχεια στην υλοποίηση της συγκεκριμένης πτυχιακής εργασίας, έτσι ώστε να βελτιωθούν τα αποτελέσματα αυτής ή να προστεθούν επιπλέον δυνατότητες και εφαρμογές. Πολύ χρήσιμο θα ήταν η μεταποίηση του κώδικα ώστε να τρέχει παράλληλα σε όλους τους διαθέσιμους πυρήνες του

επεξεργαστή, ή ακόμα καλύτερα της κάρτας γραφικών, για να μπορεί το σύστημα να εφαρμοστεί σε πολύ μεγαλύτερα προβλήματα, χωρίς τον περιορισμό της αργής εκτέλεσης σε μόνο ένα πυρήνα του επεξεργαστή.

Για το πρώτο κομμάτι που αφορά την προεπεξεργασία των δεδομένων της βάσης mPower, ο κώδικας μπορεί να επεκταθεί ούτως ώστε να δέχεται περισσότερα χαρακτηριστικά και στοιχεία για τους ομιλητές (meta δεδομένα) από τη μέχρι τώρα προσσέγγιση. Τέτοια meta δεδομένα είναι τα ερωτηματολόγια που κλήθηκαν να απαντήσουν οι ομιλητές στην δημογραφική ενότητα του mPower, η ηλικία-φύλο των ομιλητών, αν καπνίζουν κλπ. Επιπλέον με την μείωση του θορύβου εξωτερικών παραγόντων τα οποία παρατηρήθηκαν σε αρκετά δεδομένα, θα αυξηθεί η ακρίβεια στις περιοχές ταξινόμησης. Τέλος, το σύνολο δεδομένων, θα πρέπει να αθυξηθεί στα πλαίσια μεγαλύτερου χρονικού διαστήματος από το τωρινό των 10 second του φωνήματος [aa].

Το δεύτερο μέρος αφορά την υλοποίηση της ορθής ταξινόμησης των ακουστικών χαρακτηριστικών από τα νευρωνικά δίκτυα. Με την χρήση περισσότερων δεδομένων απο ομιλητές με Parkinson θα υπήρχε περισσότερη πληροφορία και μία μεγαλύτερη βάση δεδομένων προς εκπαίδευση. Με την καλύτερη προσαρμογή των υπερπαραμέτρων (hyperparameters) των δικτύων θα μπορούσε να γίνει καλύτερη σύγκληση των ταξινομητών και ειδικότερα στο deepCNN δίκτυο όπου παρουσίασε την χαμηλότερη επίδοση. Τέλος, μελλοντική υλοποίηση θα μπορούσε να είναι η δημιουργία τυχαιοποιημένων δέντρων για ταξινόμηση σύμφωνα με το άρθρο του Quartieri [25], όπου προτείνετε η συγκεκριμένη μέθοδο και υλοποιείται πάνω στη βάση mPower επιτυγχάνοντας παραπλήσιες επιδόσεις.

Βιβλιογραφία

[1] K. Ho Aileen, R. Iansek, J. L. Bradshaw. Motor Instability in Parkinsonian Speech Intensity. *Neuropsychiatry, neuropsychology, and behavioral neurology*, 14:109–116, 2001.

[2] S. Damelin, W. Miller. *The Mathematics of Signal Processing*. Cambridge University Press, 2011.

[3] R. Rabiner, R. W. Schafer. *Digital Processing of Speech Signals*. Broken Hill Publishers Ltd, 1978.

[4] A. Graves, J. Schmidhuber. Framewise Phoneme Classification with Bidirectional LSTM networks. *2005 Joint Conference Neural Networks*, 14:2047–2052, 2005.

[5] I. Vlahavas, P. Kefalas, N. Bassiliades, F. Kokkoras, I. Sakellariou. *Artificial Intelligence - 3rd Edition*. University of Macedonia Press, 2011.

[6] F. Rosenblatt. *The Perceptron, a Perceiving and Recognizing Automaton Project Para*. Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957.

[7] H. Kaiming, Z. Xiangyu, R. Shaoqing, S. Jian . Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[8] S. Haykin. *Neural Networks and Learning Machines (3rd Edition)*. Prentice Hall, 2008.

[9] J. R. Stuart, P. Norvig. *Artificial Intelligence: A Modern Approach - 3rd Edition*. Prentice Hall Press, 2010.

[10] D. H. Hubel, T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160:106–154, 1962.

[11] Y. LeCun, Y. Bengio. *Convolutional Networks for Images, Speech and Time Series*. MIT Press, 1995.

[12] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15:1929–1958, 2014.

[13] S. Hochreiter, J. Schmidhuber. Long Short Term Memory. *Neural computation*, 9:1735–1780, 1997.

[14] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, J. Schmidhuber. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28:2222–2232, 2017.

[15] B. Hammer. Learning with Recurrent Neural Networks. *Springer-Verlag London Limited*, 2000.

[16] P.J. Werbos. Backpropagation through time: what it does and how to do it. 78:1550–1560, 1990.

[17] X. Glorot, Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. 8:249–256, 2010.

[18] H. Sak, F. Beaufays. Long Short-Term Memory recurrent neural network architectures for large scale acoustic modeling. 7:338–342, 2014.

[19] Sage Bionetworks. mPower Public Researcher Portal. 2016.

[20] P. M. Corsi. Human memory and the medial temporal region of the brain (Ph.D.), 1972.

[21] S. V. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62:77–89, 1997.

[22] D. D. Lewis, W. A. Gale. A sequential algorithm for training text classifiers. *In proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 1:3–12, 1994.

[23] F. Hsieh, B. W. Turnbull. Nonparametric and semiparametric estimation of the receiver operating characteristic curve. *The Annals of Statistics*, 24:25–40, 1996.

[24] K. Pearson. On Lines and Planes of Closest Fit to Systems of Points in Space. *Journal of Science*, 2:559–572, 1901.

[25] G. Ciccarelli, T. F. Quatieri, S. S. Ghosh. Neurophysiological Vocal Source Modeling for Biomarkers of Disease. *In proceedings INTERSPEECH 2016: Understanding Speech Processing in Humans and Machines*, 2016.

Λίστα άπο Κώδικες

Κεφάλαιο 8

Κώδικες

8.1 Download και οργάνωση των ηχητικών δεδομένων σε φακέλους.

```
import shutil
import synapseclient
import numpy as np
from definitions import SYNAPSE_DIR
import csv
import os
import time

if __name__ == '__main__':

    Download_Dir_Full_Path = "/home/polaras/Diploma_PD/newFiles/"
    Csv_Full_Path = SYNAPSE_DIR + "/Survey.csv"  # Demographic Survey csv file
    synapseclient.cache.CACHE_ROOT_DIR = Download_Dir_Full_Path  # set the
    download directory

    """Credentials"""
    username = 'ckotro'
    password = 'ckotro%^&8'

    syn = synapseclient.Synapse()
    syn.login(username, password)

    tableID = "syn5511444"
    column = ["audio_audio.m4a"]

    healthID = []
    with open(Csv_Full_Path, 'rb') as csv_file:
        reader = csv.reader(csv_file)
        for row in reader:
            healthID.append(row[3])
        healthID = np.asarray(healthID)
        healthID = healthID[1:len(healthID)]
    start = time.time()
```

```
    for id in healthID:

        if not os.path.exists(Download_Dir_Full_Path+id+'.alac'):
            result = syn.tableQuery("SELECT 'audio_audio.m4a' "
                                    "FROM syn5511444 WHERE healthCode = '%s'
    LIMIT 2" % id)

            file = syn.downloadTableColumns(result, column)
            try:
                item = file.items()
                print item[0][9] , item[1][0]
                new_name = os.path.join(Download_Dir_Full_Path, id + '.alac')
                shutil.move(item[0][1], new_name)
            except (ValueError, IndexError):
                print("Empty Directory")
        else:
            print("File {0}.alac already exists. .i. \nTime up until now : {1}
    ".format(id,time.time()-start))
    print("Final script time: {0}".format(time.time()-start))
```

Κώδικας 8.1: Downloader

```
'''Ta arxeia einai sthn arxikh toys morfh opws katevhkan apo to downloader.py
    '''
import os
import sys
rootDir = '.'
directory = '/home/polaras/Desktop/edw' #<------ to arxiko directory poy 8a
    tre3ei to script
os.chdir(directory)

def main():
    counter = 0
    for folderName, subfolders, filenames in os.walk(rootDir):
        print('The current folder is ' + folderName)
        for subfolder in subfolders:
            print(subfolder)
        for filename in filenames:
            print('FILE INSIDE ' + folderName + ': ' + filename)
            if filename.endswith('.tmp'): #gia na 3exwrisw apola ta upoloipa
    arxeia
                counter = counter + 1
                name = subfolder + filename[71:] #71 #o teleutaios subfolder
    einai kai to id toy hxhtikou kai 8elw na to valw brosta toy
                folderName = folderName[2:] #diwxnw to ./ pou einai to rootDir
     kai ftiaxnw to old_path kai new_path gia na ta metaonomasw kai metakinisw
     parallhla
                old_path = os.path.join(directory, folderName)
                old_path = os.path.join(old_path, filename)
                new_path = os.path.join(directory, name)
                print(folderName)
                print(old_path)
```

```
                print(new_path)
                os.rename(old_path,new_path) #<------- edw ginetai h doyleia
        print('')
    print(counter)
    return 0

if __name__ == '__main__':
    status = main()
    sys.exit(status)
```

Κώδικας 8.2: Extractor

```
'''Sta arxeia prepei prwta na exei trexei to script file_extractor.py etsi
    wste na einai se morfh id.tmp '''
import os
import csv
from definitions import AUDIO_DIR

if __name__ == '__main__':
    filenames = os.listdir(AUDIO_DIR)
    with open('Survey.csv', 'rb') as csv_file:
        reader = csv.DictReader(csv_file)
        for row in reader:
            for idx, filename in enumerate(filenames):
                if row['healthCode'] == filename[:-5]:
                    os.rename(AUDIO_DIR + '/' + filename, AUDIO_DIR + '/'
                              + filename[:-5] + '_' + row['professional-
    diagnosis'] + '.alac')
```

Κώδικας 8.3: Organizer

```
'''Sta arxeia prepei prwta na exei trexei to script file_organizerr.py etsi
    wste na einai se morfh id_bla_bla.m4a '''
import os
import sys
import subprocess
import shutil
from definitions import AUDIO_DIR

# <--- to directory poy trexei to script

working_directory = AUDIO_DIR
os.chdir(working_directory)


def main():
    filenames = []  # <------- pinakas me ola ta filename sto working directory
     poy teleiwnoun se .m4a
    for filename in os.listdir(working_directory):
        if filename.endswith('.m4a'):
            filenames.append(filename)

```

```
    for filename in filenames:
        subprocess.call([
            "ffmpeg", "-i",
            os.path.join(working_directory, filename),
            "-ar", "16000",
            os.path.join(working_directory, '%s.wav' % filename[:-4])
            #<======= metatroph toy m4a arxeiou se wav kai rixnw to samplerate sta 16kHz
        ])

    filenames = []
    for filename in os.listdir(working_directory):  #<======== xana ftiaxnw ena pinaka apo ta ftiagmena wav pleon
        if filename.endswith('.wav'):
            filenames.append(filename)

    for filename in filenames:  #<======= stelnw ta arxeia ston kainourio directory
        if filename.endswith('.wav'):
            old_path = os.path.join(working_directory, filename)
            new_path = []
            print (old_path)
            print (new_path)
            os.rename(old_path, new_path)

    shutil.rmtree(working_directory)
    return 0


if __name__ == '__main__':
    sub_dirs = os.listdir(AUDIO_DIR)
    for idx1, sub_dir in enumerate(sub_dirs):
        filenames = os.listdir(sub_dir)
        print (len(filenames))
        for idx2, filename in enumerate(filenames):
            full_path =  AUDIO_DIR + '/' + sub_dir + '/' + filename
```

Κώδικας 8.4: Transformer

```
'''Prepei na einai exei trexei to script file_transformer.py dhladh ta arxeia na einai se morfh 5827528_idont_take...._audio.wav gia na doulepsei
   Prepei epishs na vriskontai ston fakelo Files '''
import os
import sys
from definitions import AUDIO_DIR
import shutil

if __name__ == '__main__':
    if not os.path.exists(AUDIO_DIR + '/PD'):
        os.mkdir(AUDIO_DIR + '/PD')
    if not os.path.exists(AUDIO_DIR + '/HC'):
        os.mkdir(AUDIO_DIR + '/HC')
```

```
    filenames = os.listdir(AUDIO_DIR)
    for idx, filename in enumerate(filenames):
        if filename[37:-5] == 'true' and not(dir(filename)):
            shutil.move(AUDIO_DIR + '/' + filename, AUDIO_DIR + '/PD')
        else :
            shutil.move(AUDIO_DIR + '/' + filename, AUDIO_DIR + '/HC')
```

Κώδικας 8.5: Individual Sorter

8.2 Εξαγωγή χαρακτηριστικών και δημιουργία των train/test set.

```
from __future__ import print_function
from shutil import copy2
from definitions import FILES_DIR
import csv
import librosa
import librosa.display
import os
import numpy as np
import sys
from librosa.feature import mfcc
from librosa.feature import delta

''' This script can produce 3 possible dataset types so far:

        1) Spectro: spectrogram input of size (:,:,128,400)
        2) StackPerRow: Mfcc Delta Delta2 stacked per row input of size (:,:,60,400)
        3) StackPlain: Mfcc Delta Delta2 plain stacked input of size (:,:,60,400)

    Also it creates an input csv with the corresponding name'''


def _get_class_names(path):  # class names are subdirectory names in Samples/ directory
    class_names = os.listdir(path)
    return class_names


def createCsv(data):
    with open("Inputs.csv", "wb") as csv_file:
        writer = csv.writer(csv_file)
        for row in data:
            writer.writerow([row])


def moveSamples(list, source, dest):
    for filename in list:
        if not os.path.exists(dest + '/' + filename):
```

```
37              copy2(source + '/' + filename, dest)
38          else:
39              print("File already exists.")
40
41
42  def createAudioSamplesDir(nb_files):
43      '''Create Audio Sample directory
44          Create Csv with shuffled inputs'''
45
46      os.chdir(FILES_DIR)
47      audio_sample_path = "Audio_Files_Samples/"
48      audio_origin_path = "Audio_Files/"
49      inputFiles = []
50      if not os.path.exists(audio_sample_path):
51          print("There is no samples directory. \nCreating new one..")
52          os.mkdir(audio_sample_path)
53
54      class_names = _get_class_names(audio_origin_path)
55
56      print("class_names = ", class_names)
57
58      for idx, classname in enumerate(class_names):  # go through the subdirs
59
60          class_files = os.listdir(audio_origin_path + classname)
61
62          if classname.startswith("H"):
63              choice = np.random.choice(class_files, 556,
64                                        replace=False)  # randomly select n
    files from a class
65          elif classname.startswith("P"):
66              choice = np.random.choice(class_files, 278,
67                                        replace=False)
68          else:
69              sys.exit("Unable to determine audio class : {:14s} ".format(
    classname))
70
71          if not os.path.exists(audio_sample_path + classname):
72              print("There is no {:s} sub-directory. \nCreating new one..".
    format(classname))
73              os.mkdir(audio_sample_path + classname)
74
75          moveSamples(choice, audio_origin_path + classname,
76                      audio_sample_path + classname)  # copy from audio origin
    to audio samples
77          inputFiles.extend(choice)
78
79      inputFiles = np.array(inputFiles)
80      np.random.shuffle(inputFiles)
81
82      createCsv(inputFiles)
83
```

```
84      return audio_sample_path
85
86
87  def _LogMelSpectrogram(aud, sr):
88      melgram = librosa.logamplitude(
89          librosa.feature.melspectrogram(
90              aud, sr=sr, n_mels=96), ref_power=1.0)  # load log melspectrogram
    and save it with 128 mels
91      # also reashape to [:,:,:,400]
92      return melgram
93
94
95  #
96  # def extract_feature_array_CNN(aud, sr, n_mels=256):
97  #     melspec = librosa.feature.melspectrogram(aud, n_mels=n_mels, power=1.0)
98  #     logs_spectrogram = librosa.logamplitude(melspec)
99  #     logs_spectrogram = logs_spectrogram[:, :, np.newaxis]
100 #
101 #     feature = np.concatenate((logs_spectrogram, np.zeros(logs_spectrogram.
    shape)), axis=2)
102 #     feature = np.concatenate((feature, np.zeros(logs_spectrogram.shape)),
    axis=2)
103 #
104 #     for i in range(len(feature)):
105 #         feature[i, :, 1] = librosa.feature.delta(feature[i, :, 0])
106 #         feature[i, :, 2] = librosa.feature.delta(feature[i, :, 1], order=2)
107 #
108 #     return feature
109 #
110 #
111 # def extract_feature_array_RNN(aud, sr, n_mels=256):
112 #     melspec = librosa.feature.melspectrogram(aud, n_mels=n_mels, sr=sr, power
    =2.0)
113 #     logs_spectrogram = librosa.logamplitude(melspec)
114 #     delta = librosa.feature.delta(logs_spectrogram)
115 #     delta2 = librosa.feature.delta(logs_spectrogram, order=2)
116 #
117 #     logs_spectrogram = logs_spectrogram.T
118 #     delta = delta.T
119 #     delta2 = delta2.T
120 #     features = []
121 #     for row in range(len(logs_spectrogram)):
122 #         stack = np.vstack(
123 #             (logs_spectrogram[row], delta[row], delta2[row]))  # stack with
    row of each mel delta and delta2
124 #         features.extend(stack)  # append to array
125 #     features = np.array(features).T
126 #
127 #     return features
128
129
```

```
130 def extract_feature_array(aud, sr):
131     mfcc_feat = mfcc(aud, sr, n_mfcc=20)
132     delta_feat = delta(mfcc_feat)
133     delta2_feat = delta(delta_feat, order=2)
134     feature = np.concatenate((mfcc_feat, delta_feat, delta2_feat))
135     # feature = np.mean(feature, axis=1)
136     return feature
137 
138 
139 class preprocess_dataset(object):
140     def __init__(self, nb_files, path):
141         os.chdir(FILES_DIR)
142         if os.path.exists(path):
143             self.AudioSamplePath = "Audio_Files_Samples/"
144         else:
145             print("Creating new Audio Sample Files..\n")
146             self.AudioSamplePath = createAudioSamplesDir(nb_files)
147 
148     def _createDataset(self, printevery, outpath):
149 
150         if not os.path.exists(outpath):
151             os.mkdir(outpath)  # make a new directory for preproc'd files
152         else:
153             print("A dataset with extracted features already exists.")
154             #    print("Path: {:s} alraedy exists.".format(outpath))
155             return outpath
156 
157         data = []
158         class_names = _get_class_names(path=self.AudioSamplePath)  # get the
    names of the subdirectories
159         nb_classes = len(class_names)
160         print("class_names = ", class_names)
161         for idx, classname in enumerate(class_names):  # go through the
    subdirs
162 
163             if not os.path.exists(outpath + classname):
164                 os.mkdir(outpath + classname)  # make a new subdirectory for
    preproc class
165 
166             class_files = os.listdir(self.AudioSamplePath + classname)
167             n_files = len(class_files)
168             n_load = n_files
169             print(' class name = {:14s} - {:3d}'.format(classname, idx),
170                   ", ", n_files, " files in this class", sep="")
171 
172             for idx2, infilename in enumerate(class_files):  # fore each file
173                 audio_path = self.AudioSamplePath + classname + '/' +
    infilename
174                 if (0 == idx2 % printevery):
175                     print('\r Creating class: {:14s} ({:2d} of {:2d} classes)'
    .format(classname, idx + 1, nb_classes),
```

```
176                         ", file ", idx2 + 1, " of ", n_load, ": ",
    audio_path, sep="")
177                 aud, sr = librosa.load(audio_path)
178
179                 if outpath == "Extracted_features/":
180                     if len(aud) == 0: # if empty or corrupted file
181                         print("File remove!! : ", audio_path)
182                         os.remove(audio_path)
183                     else:
184                         data = extract_feature_array(aud, sr)
185                 else:
186                     sys.exit("Unable to specify outpath")
187                 outfile = outpath + classname + '/' + infilename + '.npy'
188
189                 np.save(outfile, data)  # save output to file
190
191         return outpath
192
193
194 if __name__ == '__main__':
195     audio = FILES_DIR + "/" + "Audio_Files" + "/HC/0a21ce6f-a5e2-413c-a1a1-12
    dce0eb8826_fals.wav"
196     aud, sr = librosa.load(audio)
197     feature = extract_feature_array(aud, sr)
198     # feature = np.matrix(feature)
199
200     import matplotlib.pyplot as plt
201
202     plt.figure(figsize=(7, 5.2))
203     plt.show()
204     print(feature.shape)
205
206     # feature = np.vstack((mfcc, fdelta, fdelta2))
```

Κώδικας 8.6: Pre Process Data

```
1  from __future__ import print_function
2
3  from sklearn.model_selection import train_test_split
4
5  from definitions import FILES_DIR
6  import numpy as np
7  import sys
8  import os
9
10
11 def get_class_names(path):
12     print(path)  # class names are subdirectory names in Preproc directory
13     class_names = os.listdir(path)
14     return class_names
15
16
```

```
17 def get_total_files(path, train_percentage):
18     sum_total = 0
19     sum_train = 0
20     sum_test = 0
21     subdirs = os.listdir(path)
22     for subdir in subdirs:
23         files = os.listdir(path + subdir)
24         n_files = len(files)
25         sum_total += n_files
26         n_train = int(train_percentage * n_files)
27         n_test = n_files - n_train
28         sum_train += n_train
29         sum_test += n_test
30 
31     return sum_total, sum_train, sum_test
32 
33 
34 def get_sample_dimensions(path):
35     classname = os.listdir(path)[0]
36     files = os.listdir(path + classname)
37     infilename = files[0]
38     audio_path = path + classname + '/' + infilename
39     mel = np.load(audio_path)
40     return mel.shape
41 
42 
43 def encode_class(class_name, class_names):  # makes a "one-hot" vector for
    each class name called
44     try:
45         idx = class_names.index(class_name)
46         vec = np.zeros(len(class_names))
47         vec[idx] = 1
48         return vec
49     except ValueError:
50         return None
51 
52 
53 def shuffle_XY_paths(X, Y, paths):  # generates a randomized order, keeping X&
    Y(&paths) together
54     assert (X.shape[0] == Y.shape[0])
55     idx = np.array(range(Y.shape[0]))
56     np.random.shuffle(idx)
57     newX = np.copy(X)
58     newY = np.copy(Y)
59     newpaths = paths
60     for i in range(len(idx)):
61         newX[i] = X[idx[i], :, :]
62         newY[i] = Y[idx[i], :]
63         newpaths[i] = paths[idx[i]]
64     return newX, newY, newpaths
65 
```

```
66 def get_min_dimensions(path):
67     class_names = os.listdir(path)  # get the names of the subdirectories
68     min = 80000
69     for idx, classname in enumerate(class_names):  # go through the subdirs
70         class_files = os.listdir(path + classname)
71         for idx2, infilename in enumerate(class_files):  # fore each file
72             mel_path = path + classname + '/' + infilename
73             mel = np.load(mel_path)
74             if min > mel.shape[1]:
75                 min = mel.shape[1]
76                 print(infilename)
77                 print("Minimum shape is : ", mel.shape)
78     mel = mel[:, 0:min]
79     print("Minimum shape is : ", mel.shape)
80     return mel.shape
81 
82 '''To make sure statistics in training & testing are as similar
83 as possible I create train and test dataset separately'''
84 
85 
86 def build_dataset(train_percentage, path):
87     os.chdir(FILES_DIR)
88 
89     class_names = get_class_names(path=path)
90     print("class_names = ", class_names)
91 
92     total_files, total_train, total_test = get_total_files(path=path,
    train_percentage=train_percentage)
93     print("total files = ", total_files)
94 
95     nb_classes = len(class_names)
96 
97 
98 
99     mel_dims = get_min_dimensions(path)  # Find out the 'shape' of smallest
    data file
100 
101     # pre-allocate memory for speed (old method used np.concatenate, slow)
102     paths_train = []
103     paths_test = []
104     X = np.zeros((total_files, mel_dims[0], mel_dims[1]))
105     Y = np.zeros((total_files, nb_classes))
106     train_count = 0
107     test_count = 0
108     count = 0
109     for idx, classname in enumerate(class_names):
110         this_Y = np.array(encode_class(classname, class_names))  # makes one
    hot vector
111         this_Y = this_Y[np.newaxis, :]  # reshape
112         print(this_Y)
113         class_files = os.listdir(path + classname)
```

```
114         n_files = len(class_files)
115         n_load = n_files
116         n_train = int(train_percentage * n_load)
117         printevery = 50
118         print("")
119         for idx2, infilename in enumerate(class_files[0:n_load]):
120             mel_path = path + classname + '/' + infilename
121             if (0 == idx2 % printevery):
122                 print('\r Loading class: {:14s} ({:2d} of {:2d} classes)'.
    format(classname, idx + 1, nb_classes),
123                       ", file ", idx2 + 1, " of ", n_load, ": ", mel_path, sep
    ="")
124             # start = timer()
125             mel = np.load(mel_path)
126             mel = mel[:, 0:mel_dims[1]]
127             # because files may be differnt size: clip to smallest file size
    and reshape
128             # end = timer()
129             # print("time = ",end - start)
130             X[count, :, :] = mel
131             Y[count, :] = this_Y
132             count += 1
133
134         print("")
135     print(X.shape)
136     print("Shuffling order of data...")
137     X_train, X_test, y_train, y_test = train_test_split(X, Y, random_state
    =123, train_size=train_percentage)
138
139     # X_train, Y_train, paths_train = shuffle_XY_paths(X_train, Y_train,
    paths_train)
140     # X_test, Y_test, paths_test = shuffle_XY_paths(X_test, Y_test, paths_test
    )
141
142     return X_train, y_train, paths_train, X_test, y_test, paths_test,
    class_names
```

Κώδικας 8.7: Build Dataset

8.3 Δημιουργια μοντέλων για εκπαίδευση και παραγωγή αποτελεσμάτων.

```
1 from __future__ import print_function
2 import keras
3 import shutil
4 import csv
5 from keras.layers import *
6 from keras.optimizers import *
7 from keras.regularizers import *
8 from keras.models import *
9 from keras.engine import Input
```

```
from keras.layers import Conv2D, Flatten, AveragePooling2D
from keras.layers.advanced_activations import ELU
from keras.optimizers import Adadelta
from  keras.losses import categorical_crossentropy
from keras.layers import MaxPooling2D
from scipy import interp
import matplotlib.pyplot as plt
import numpy as np
from keras.models import Sequential
from sklearn.metrics import precision_recall_fscore_support, roc_auc_score
from keras.callbacks import EarlyStopping, ModelCheckpoint,
    LearningRateScheduler, ReduceLROnPlateau
from os.path import isfile
from sklearn.metrics import roc_curve, auc
from itertools import cycle, product
from keras.layers.core import Dense, Activation, Dropout, Reshape
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.cross_decomposition import CCA

from sklearn import tree
from definitions import MODEL_RUN_DIR, DISS_CODE_FIG_DIR
import time

from matplotlib.colors import ListedColormap


class build_model(object):
    def __init__(self, nb_classes, X_train, Y_train, X_test, Y_test, nb_epoch,
     batch_size=64, model_name='',
                 monitor=''):
        shutil.copy(__file__, MODEL_RUN_DIR)
        self.X_train = X_train
        self.Y_train = Y_train
        self.X_test = X_test
        self.Y_test = Y_test
        self.nb_epoch = nb_epoch
        self.batch_size = batch_size
        self.nb_classes = nb_classes
        self.monitor = monitor
        self.model_name = model_name
        if model_name == "deepLSTM":
            self.model_type = self.deep_lstm()
        elif model_name == "deepCNN":
            self.model_type = self.deep_cnn()
        elif model_name == "resNET":
            self.model_type = self.resNET()
        elif model_name == "CRNN":
            self.model_type = self.crnn()
        elif model_name == "exTraTree":
```

```
59              self.model_type = self.exTraTree()
60          elif model_name == "BiLSTM":
61              self.model_type = self.BiLSTM()
62          else:
63              print("No proper model is defined")
64              return
65
66      def deep_lstm(self):
67          xShape = self.X_train.shape[1]
68          yShape = self.X_train.shape[2]
69          model = Sequential()
70          model.add(LSTM(512, input_shape=(xShape, yShape), return_sequences=
    True))
71          model.add(Dropout(0.2))
72          model.add(LSTM(256, return_sequences=True))
73          model.add(Dropout(0.2))
74          model.add(LSTM(128, return_sequences=True))
75          model.add(Dropout(0.2))
76          model.add(LSTM(64, return_sequences=False))
77          model.add(Dropout(0.2))
78          model.add(Dense(32, activation='relu'))
79          model.add(Dropout(0.3))
80          model.add(Dense(2, activation='softmax'))
81          model.compile(loss="categorical_crossentropy", optimizer="adam",
    metrics=[self.monitor])
82          model.summary()
83          return model
84
85      def deep_cnn(self):
86          nb_filters = 60  # number of convolutional filters to use
87          pool_size = (2, 2)  # size of pooling area for max pooling
88          kernel_size = (3, 3)  # convolution kernel size
89          nb_layers = 5
90          input_shape = (self.X_train.shape[1], self.X_train.shape[2], 1)
91
92          self.X_train = self.X_train.reshape(self.X_train.shape[0], self.
    X_train.shape[1], self.X_train.shape[2], 1)
93          self.X_test = self.X_test.reshape(self.X_test.shape[0], self.X_test.
    shape[1], self.X_test.shape[2], 1)
94
95          model = Sequential()
96          conv = Conv2D(nb_filters, input_shape=input_shape, kernel_size=
    kernel_size)
97          model.add(conv)
98          model.add(BatchNormalization(axis=1))
99          model.add(Activation('relu'))
100
101         for layer in range(nb_layers - 1):
102             model.add(Conv2D(nb_filters, kernel_size=kernel_size))
103             model.add(BatchNormalization(axis=1))
104             model.add(ELU(alpha=1.0))
```

```
105             model.add(MaxPooling2D(pool_size=pool_size))
106             model.add(Dropout(0.3))
107 
108         model.add(Flatten())
109         model.add(Dense(32))
110         model.add(Activation('relu'))
111         model.add(Dropout(0.2))
112         model.add(Dense(len(self.nb_classes)))
113         model.add(Activation("softmax"))
114 
115         model.summary()
116         model.compile(loss=categorical_crossentropy,
117                       optimizer=Adam(lr=0.0004),
118                       metrics=[self.monitor])
119 
120         return model
121 
122     def crnn(self):
123         input_shape = (self.X_train.shape[1], self.X_train.shape[2], 1)
124         cnn_layer_num_filters = 180
125         cnn_layer_kernel_size = (3, 3)
126         cnn_layer_strides = (4, 4)
127         rnn_cell = Bidirectional(LSTM(25, return_sequences=True))
128         full_connected_layer_num_units = 120
129         dropout = 0.2
130         opt = Adam
131         learning_rate = 0.0005
132 
133         self.X_train = self.X_train.reshape(self.X_train.shape[0], self.
    X_train.shape[1], self.X_train.shape[2], 1)
134         self.X_test = self.X_test.reshape(self.X_test.shape[0], self.X_test.
    shape[1], self.X_test.shape[2], 1)
135 
136         model = Sequential()
137         model.add(Conv2D(filters=cnn_layer_num_filters, kernel_size=
    cnn_layer_kernel_size,
138                          strides=cnn_layer_strides, padding="valid",
    input_shape=input_shape))
139         print(model.output_shape)
140         model.add(BatchNormalization(axis=2))
141         model.add(Activation("relu"))
142         model.summary()
143         print(model.output_shape)
144         model.add(Reshape((963 * 2 * 3, 50)))
145 
146         model.add(rnn_cell)
147 
148         model.add(Bidirectional(LSTM(60, return_sequences=False)))
149 
150         model.add(Dense(units=full_connected_layer_num_units))
151         model.add(Activation("relu"))
```

```
152         model.add(Dropout(dropout))
153         model.add(Dense(units=2, activation="softmax"))
154 
155         optimizer = opt(lr=learning_rate)
156 
157         model.compile(optimizer=optimizer, loss="mean_squared_error", metrics
    =[self.monitor])
158         model.summary()
159         return model
160 
161     def exTraTree(self):
162         from sklearn.ensemble import ExtraTreesRegressor
163         pca = PCA(n_components=60)
164         estimator = ExtraTreesRegressor(n_estimators=834, max_features=60)
165         self.X_train = self.X_train.reshape(self.X_train.shape[0], self.
    X_train.shape[1] * self.X_train.shape[2])
166         self.X_test = self.X_test.reshape(self.X_test.shape[0], self.X_test.
    shape[1] * self.X_test.shape[2])
167         self.X_train = pca.fit_transform(self.X_train)
168         self.X_test = pca.fit_transform(self.X_test)
169         return estimator
170 
171     def BiLSTM(self):
172 
173         self.X_train = self.X_train.reshape(self.X_train.shape[0], self.
    X_train.shape[2], self.X_train.shape[1])
174         self.X_test = self.X_test.reshape(self.X_test.shape[0], self.X_test.
    shape[2], self.X_test.shape[1])
175         xShape = self.X_train.shape[1]
176         yShape = self.X_train.shape[2]
177 
178         model = Sequential()
179         model.add(Bidirectional(LSTM(120, return_sequences=True), input_shape
    =(xShape, yShape)))
180         model.add(BatchNormalization())
181         model.add(Dropout(0.2))
182         model.add(Bidirectional(LSTM(60, return_sequences=True)))
183         model.add(BatchNormalization())
184         model.add(Dropout(0.2))
185         model.add(Bidirectional(LSTM(30, return_sequences=True)))
186         model.add(BatchNormalization())
187         model.add(Dropout(0.2))
188         model.add(Bidirectional(LSTM(15, return_sequences=False)))
189         model.add(Dropout(0.2))
190         model.add(Dense(30, activation='relu'))
191         model.add(Dropout(0.2))
192         model.add(Dense(2, activation='softmax'))
193 
194         # try using different optimizers and different optimizer configs
195         model.compile(optimizer='sgd', loss='mean_squared_error', metrics=[
    self.monitor])
```

```
196         model.summary()
197         return model
198 
199     def resNET(self):
200 
201         def resnet_layer(inputs,
202                          num_filters=16,
203                          kernel_size=3,
204                          strides=1,
205                          activation='relu',
206                          batch_normalization=True,
207                          conv_first=True):
208             """2D Convolution-Batch Normalization-Activation stack builder
209             # Arguments
210                 inputs (tensor): input tensor from input image or previous
    layer
211                 num_filters (int): Conv2D number of filters
212                 kernel_size (int): Conv2D square kernel dimensions
213                 strides (int): Conv2D square stride dimensions
214                 activation (string): activation name
215                 batch_normalization (bool): whether to include batch
    normalization
216                 conv_first (bool): conv-bn-activation (True) or
217                     bn-activation-conv (False)
218             # Returns
219                 x (tensor): tensor as input to the next layer
220             """
221             conv = Conv2D(num_filters,
222                           kernel_size=kernel_size,
223                           strides=strides,
224                           padding='same',
225                           kernel_initializer='he_normal',
226                           kernel_regularizer=l2(1e-4))
227             x = inputs
228             if conv_first:
229                 x = conv(x)
230                 if batch_normalization:
231                     x = BatchNormalization()(x)
232                 if activation is not None:
233                     x = Activation(activation)(x)
234             else:
235                 if batch_normalization:
236                     x = BatchNormalization()(x)
237                 if activation is not None:
238                     x = Activation(activation)(x)
239                 x = conv(x)
240             return x
241 
242         def resnet_v1(input_shape, depth, num_classes=2):
243 
244             if (depth - 2) % 6 != 0:
```

```
245                 raise ValueError('depth should be 6n+2 (eg 20, 32, 44 in [a])'
    )
246             # Start model definition.
247             num_filters = 16
248             num_res_blocks = int((depth - 2) / 6)
249 
250             inputs = Input(shape=input_shape)
251             x = resnet_layer(inputs=inputs)
252             # Instantiate the stack of residual units
253             for stack in range(3):
254                 for res_block in range(num_res_blocks):
255                     strides = 1
256                     if stack > 0 and res_block == 0:  # first layer but not
    first stack
257                         strides = 2  # downsample
258                     y = resnet_layer(inputs=x,
259                                      num_filters=num_filters,
260                                      strides=strides)
261                     y = resnet_layer(inputs=y,
262                                      num_filters=num_filters,
263                                      activation=None)
264                     if stack > 0 and res_block == 0:  # first layer but not
    first stack
265                         # linear projection residual shortcut connection to
    match
266                         # changed dims
267                         x = resnet_layer(inputs=x,
268                                          num_filters=num_filters,
269                                          kernel_size=1,
270                                          strides=strides,
271                                          activation=None,
272                                          batch_normalization=False)
273                     x = keras.layers.add([x, y])
274                     x = Activation('relu')(x)
275                 num_filters *= 2
276 
277             # Add classifier on top.
278             # v1 does not use BN after last shortcut connection-ReLU
279             x = AveragePooling2D(pool_size=8)(x)
280             y = Flatten()(x)
281             outputs = Dense(num_classes,
282                             activation='softmax',
283                             kernel_initializer='he_normal')(y)
284 
285             # Instantiate model.
286             model = Model(inputs=inputs, outputs=outputs)
287             return model
288 
289         def resnet_v2(input_shape, depth=29, num_classes=2):
290 
291             if (depth - 2) % 9 != 0:
```

```
292                 raise ValueError('depth should be 9n+2 (eg 56 or 110 in [b])')
293             # Start model definition.
294             num_filters_in = 16
295             num_res_blocks = int((depth - 2) / 9)
296 
297             inputs = Input(shape=input_shape)
298             # v2 performs Conv2D with BN-ReLU on input before splitting into 2
    paths
299             x = resnet_layer(inputs=inputs,
300                              num_filters=num_filters_in,
301                              conv_first=True)
302 
303             # Instantiate the stack of residual units
304             for stage in range(3):
305                 for res_block in range(num_res_blocks):
306                     activation = 'relu'
307                     batch_normalization = True
308                     strides = 1
309                     if stage == 0:
310                         num_filters_out = num_filters_in * 4
311                         if res_block == 0:  # first layer and first stage
312                             activation = None
313                             batch_normalization = False
314                     else:
315                         num_filters_out = num_filters_in * 2
316                         if res_block == 0:  # first layer but not first stage
317                             strides = 2  # downsample
318 
319                     # bottleneck residual unit
320                     y = resnet_layer(inputs=x,
321                                      num_filters=num_filters_in,
322                                      kernel_size=1,
323                                      strides=strides,
324                                      activation=activation,
325                                      batch_normalization=batch_normalization,
326                                      conv_first=False)
327                     y = resnet_layer(inputs=y,
328                                      num_filters=num_filters_in,
329                                      conv_first=False)
330                     y = resnet_layer(inputs=y,
331                                      num_filters=num_filters_out,
332                                      kernel_size=1,
333                                      conv_first=False)
334                     if res_block == 0:
335                         # linear projection residual shortcut connection to
    match
336                         # changed dims
337                         x = resnet_layer(inputs=x,
338                                          num_filters=num_filters_out,
339                                          kernel_size=1,
340                                          strides=strides,
```

```
                                        activation=None,
                                        batch_normalization=False)
                    x = keras.layers.add([x, y])

                num_filters_in = num_filters_out

            # Add classifier on top.
            # v2 has BN-ReLU before Pooling
            x = BatchNormalization()(x)
            x = Activation('relu')(x)
            x = AveragePooling2D(pool_size=8)(x)
            y = Flatten()(x)
            outputs = Dense(num_classes,
                            activation='softmax',
                            kernel_initializer='he_normal')(y)

            # Instantiate model.
            model = Model(inputs=inputs, outputs=outputs)
            return model

        input_shape = (self.X_train.shape[1], self.X_train.shape[2], 1)

        self.X_train = self.X_train.reshape(self.X_train.shape[0], self.
    X_train.shape[1], self.X_train.shape[2], 1)
        self.X_test = self.X_test.reshape(self.X_test.shape[0], self.X_test.
    shape[1], self.X_test.shape[2], 1)
        model = resnet_v1(input_shape=input_shape, depth=20, num_classes=2)
        model.compile(loss='binary_crossentropy',
                      optimizer=Adam(lr=0.001), metrics=[self.monitor])
        model.summary()
        return model

    # pca
    def plot_PCA(self):
        def plot_hyperplane(clf, min_x, max_x, linestyle, label):

            # get the separating hyperplane
            w = clf.coef_[0]
            a = -w[0] / w[1]
            xx = np.linspace(min_x - 5, max_x + 5)  # make sure the line is
    long enough
            yy = a * xx - (clf.intercept_[0]) / w[1]
            plt.plot(xx, yy, linestyle, label=label)

        def plot_figure(X, Y, title, transform):
            if transform == "pca":
                X = PCA(n_components=2).fit_transform(X)
            elif transform == "cca":
                X = CCA(n_components=2).fit(X, Y).transform(X)
            else:
                raise ValueError
```

```
389
390             min_x = np.min(X[:, 0])
391             max_x = np.max(X[:, 0])
392
393             min_y = np.min(X[:, 1])
394             max_y = np.max(X[:, 1])
395
396             classif = OneVsRestClassifier(SVC(kernel='linear'))
397             classif.fit(X, Y)
398
399             plt.title(title)
400
401             zero_class = np.where(Y[:, 0])
402             one_class = np.where(Y[:, 1])
403             plt.scatter(X[:, 0], X[:, 1], s=40, c='gray', edgecolors=(0, 0, 0)
    )
404             plt.scatter(X[zero_class, 0], X[zero_class, 1], s=160, edgecolors=
    'b',
405                         facecolors='none', linewidths=2, label='HC')
406             plt.scatter(X[one_class, 0], X[one_class, 1], s=80, edgecolors='
    orange',
407                         facecolors='none', linewidths=2, label='PD')
408
409             plot_hyperplane(classif.estimators_[0], min_x, max_x, 'k--',
410                             'Boundary\nfor HC')
411             plot_hyperplane(classif.estimators_[1], min_x, max_x, 'k-.',
412                             'Boundary\nfor PD')
413             plt.xticks(())
414             plt.yticks(())
415
416             plt.xlim(min_x - .5 * max_x, max_x + .5 * max_x)
417             plt.ylim(min_y - .5 * max_y, max_y + .5 * max_y)
418
419             plt.xlabel('First principal component')
420             plt.ylabel('Second principal component')
421             plt.legend(loc="upper left")
422             plt.savefig(DISS_CODE_FIG_DIR + '/' + 'pca.png')
423
424         plt.figure()
425         print("Plotting PCA")
426         print("=" * 65)
427
428         X_test = self.X_test.reshape(self.X_test.shape[0], self.X_test.shape
    [1] * self.X_test.shape[2])
429         plot_figure(X_test, self.Y_test, "PCA", "pca")
430
431     # ROC
432     def plot_ROC(self):
433         plt.figure()
434         fpr = dict()
435         tpr = dict()
```

```
436         lw = 2
437         roc_auc = dict()
438         for i in range(len(self.nb_classes)):
439             fpr[i], tpr[i], _ = roc_curve(self.Y_test[:, i], self.pred[:, i])
440             roc_auc[i] = auc(fpr[i], tpr[i])
441 
442         # Compute micro-average ROC curve and ROC area
443         fpr["micro"], tpr["micro"], _ = roc_curve(self.Y_test.ravel(), self.
    pred.ravel())
444         roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])
445 
446         # Compute macro-average ROC curve and ROC area
447 
448         # First aggregate all false positive rates
449         all_fpr = np.unique(np.concatenate([fpr[i] for i in range(len(self.
    nb_classes))]))
450 
451         # Then interpolate all ROC curves at this points
452         mean_tpr = np.zeros_like(all_fpr)
453         for i in range(len(self.nb_classes)):
454             mean_tpr += interp(all_fpr, fpr[i], tpr[i])
455 
456         # Finally average it and compute AUC
457         mean_tpr /= len(self.nb_classes)
458 
459         fpr["macro"] = all_fpr
460         tpr["macro"] = mean_tpr
461         roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])
462 
463         # Plot all ROC curves
464         color = 'darkorange'
465         plt.plot(fpr["micro"], tpr["micro"],
466                  label='ROC curve (area = {0:0.2f})'
467                        ''.format(roc_auc["micro"]),
468                  color=color, linestyle=':', linewidth=4)
469 
470         plt.plot([0, 1], [0, 1], 'k--', lw=lw)
471         plt.xlim([0.0, 1.0])
472         plt.ylim([0.0, 1.05])
473         plt.xlabel('False Positive Rate')
474         plt.ylabel('True Positive Rate')
475         plt.legend(loc="lower right")
476         plt.savefig(DISS_CODE_FIG_DIR + '/' + self.model_name + '_ROC.png')
477 
478     # Confusion Matric
479 
480     def plot_confusion_matrix(self, normalize=False, cmap=plt.cm.Blues):
481         """
482         This function prints and plots the confusion matrix.
483         Normalization can be applied by setting `normalize=True`.
484         """
```

```
485
486         def get_confusion_matrix_one_hot():
487             '''model_results and truth should be for one-hot format, i.e, have
    >= 2 columns,
488             where truth is 0/1, and max along each row of model_results is
    model result
489             '''
490             assert self.pred.shape == self.Y_test.shape
491             num_outputs = self.Y_test.shape[1]
492             confusion_matrix = np.zeros((num_outputs, num_outputs), dtype=np.
    int32)
493             predictions = np.argmax(self.pred, axis=1)
494             assert len(predictions) == self.Y_test.shape[0]
495
496             for actual_class in range(num_outputs):
497                 idx_examples_this_class = self.Y_test[:, actual_class] == 1
498                 prediction_for_this_class = predictions[
    idx_examples_this_class]
499                 for predicted_class in range(num_outputs):
500                     count = np.sum(prediction_for_this_class ==
    predicted_class)
501                     confusion_matrix[actual_class, predicted_class] = count
502             assert np.sum(confusion_matrix) == len(self.Y_test)
503             assert np.sum(confusion_matrix) == np.sum(self.Y_test)
504             return confusion_matrix
505
506         plt.figure(figsize=(7, 5.2))
507         cm = get_confusion_matrix_one_hot()
508
509         if normalize:
510             cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
511             print("Normalized confusion matrix")
512         else:
513             print('Confusion matrix, without normalization')
514
515         print(cm)
516         np.save(MODEL_RUN_DIR + '_' + self.model_name + '_ConfusionMatrixNumpy
    ', cm)
517         plt.imshow(cm, interpolation='nearest', cmap=cmap)
518
519         plt.colorbar()
520         tick_marks = np.arange(len(self.nb_classes))
521         plt.xticks(tick_marks, self.nb_classes, rotation=45)
522         plt.yticks(tick_marks, self.nb_classes)
523
524         fmt = '.2f' if normalize else 'd'
525         thresh = cm.max() / 2.
526         for i, j in product(range(cm.shape[0]), range(cm.shape[1])):
527             plt.text(j, i, format(cm[i, j], fmt),
528                      horizontalalignment="center",
529                      color="white" if cm[i, j] > thresh else "black")
```

```
530
531         plt.tight_layout()
532         plt.ylabel('True label')
533         plt.xlabel('Predicted label')
534         plt.savefig(DISS_CODE_FIG_DIR + '/' + self.model_name + '_Confusion.
    png')
535
536     # fscore accuracy loss, precision, recall, support
537     def evaluate(self, history):
538         print(history.history.keys())
539
540         # summarize history for accuracy
541         plt.figure()
542         plt.plot(history.history['acc'])
543         plt.plot(history.history['val_acc'])
544         plt.title('model accuracy')
545         plt.ylabel('accuracy')
546         plt.xlabel('epoch')
547         plt.legend(['train', 'test'], loc='upper left')
548         plt.savefig(DISS_CODE_FIG_DIR + '/' + self.model_name + '_Accuracy.png
    ')
549
550         # summarize history for loss
551         plt.figure()
552         plt.plot(history.history['loss'])
553         plt.plot(history.history['val_loss'])
554         plt.title('model loss')
555         plt.ylabel('loss')
556         plt.xlabel('epoch')
557         plt.legend(['train', 'test'], loc='upper left')
558         plt.savefig(DISS_CODE_FIG_DIR + '/' + self.model_name + '_Loss.png')
559
560         self.pred = self.model_type.predict(self.X_test, verbose=0)
561
562         predict = np.argmax(self.pred, 1)
563         y_true = np.argmax(self.Y_test, 1)
564
565         # evaluate the model
566         _, accuracy = self.model_type.evaluate(self.X_test, self.Y_test,
    batch_size=32)
567         print("\nAccuracy = {:.2f}".format(accuracy))
568
569         # the F-score gives a similiar value to the accuracy score, but useful
     for cross-checking
570         p, r, f, s = precision_recall_fscore_support(y_true, predict, average=
    'micro')
571         print("Outputs from Sklearn Lib")
572         print("F-Score: ", round(f, 3))
573         print("Precision: ", round(p, 3))
574         print("Recall: ", round(r, 3))
575         header = ["model", 'fscore', 'roc', 'accuracy', 'precision', 'recall',
```

```
    'epochs']
576         data = [self.model_name, round(f, 3),
577                 round(accuracy, 3), round(p, 3), round(r, 3), self.nb_epoch]
578         with open("outputs.csv", "a") as csv_file:
579             writer = csv.writer(csv_file)
580             writer.writerow(header)
581             writer.writerow(data)
582 
583     def train(self, load_checkpoint=False, ifearlystop=False):
584         # Initialize weights using checkpoint if it exists. (Checkpointing
    requires h5py)
585 
586         print("X train shape = ", self.X_train.shape)
587         print("X test shape = ", self.X_test.shape)
588 
589         checkpoint_filepath = self.model_name + '_weights.hdf5'
590         checkpointer = ModelCheckpoint(filepath=checkpoint_filepath, monitor="
    val_acc", verbose=1,
591                                        save_best_only=True)
592         earlystop = EarlyStopping(monitor="val_loss", patience=3, verbose=1,
    mode='auto')
593 
594         if (load_checkpoint):
595             print("Looking for previous weights...")
596             if (isfile(checkpoint_filepath)):
597                 print('Checkpoint file detected. Loading weights.')
598                 self.model_type.load_weights(checkpoint_filepath)
599             else:
600                 print('No checkpoint file detected.  Starting from scratch.')
601         else:
602             print('Starting from scratch (no checkpoint)')
603 
604         if (ifearlystop):
605             callbacks = [checkpointer, earlystop]
606             print('Training with earlystop')
607         else:
608             callbacks = [checkpointer]
609             print('Training without earlystop')
610 
611         startTime = time.time()
612         if (self.model_name == 'exTraTree'):
613             estimator = self.model_type
614             estimator.fit(self.X_train, self.Y_train)
615             y_pred = estimator.predict(self.X_test)
616             from sklearn.metrics import mean_squared_error
617             score = mean_squared_error(self.Y_test, y_pred)
618             print(round(score, 6))
619         else:
620             # train and score the model
621             fit = self.model_type.fit(self.X_train, self.Y_train, batch_size=
    self.batch_size, epochs=self.nb_epoch,
```

```
                                    shuffle=True,

                                    validation_data=[self.X_test, self.
    Y_test], callbacks=callbacks)

        endTime = time.time()
        totalTime = endTime - startTime
        print("-" * 65)
        print("The total time for Training is : ", round(totalTime, 4))
        print("-" * 65)
        print("The average time for each epoch is : ", round(totalTime / self.
    nb_epoch, 4))
        print("-" * 65)
        return fit


if __name__ == '__main__':
    pass
```

Κώδικας 8.8: Models

```
from __future__ import print_function

import csv

import numpy as np
import shutil

from definitions import MODEL_RUN_DIR, DISS_CODE_FIG_DIR

import matplotlib.pyplot as plt
from keras.utils import plot_model
from Data.Code import build_dataset, preprocess_dataset
from Build_Model import build_model as model


def main():
    models = ["deepLSTM", "deepCNN", "resNET"]  # "CRNN","deepLSTM", "deepCNN
    ", "resNET"
    header = ["model", 'fscore', 'roc', 'accuracy', 'precision', 'recall', '
    epochs']

    for modelaki in models:
        shutil.copy(__file__, MODEL_RUN_DIR)

        np.random.seed(1234)
        nb_files = 278 + 556
        batch_size = 32
        nb_epoch = 1
        train_percentage = 0.8
        model_name = modelaki
        monitor = 'accuracy'  # deepLSTM , deepCNN  , resNET , CRNN, BLSTM
```

```
30
31
32          dataset = preprocess_dataset(nb_files=nb_files, path="
    Audio_Files_Samples/")
33
34          # get the data
35          X_train, Y_train, paths_train, X_test, Y_test, paths_test, class_names
     = \
36              build_dataset(train_percentage=train_percentage,
37                            path=dataset._createDataset(printevery=10,
38                                                        outpath="
    Extracted_features/"))
39
40          # make the model
41          test_model = model(class_names, X_train, Y_train, X_test, Y_test,
    nb_epoch=nb_epoch, batch_size=batch_size,
42                             model_name=model_name, monitor=monitor)
43
44          print("-------------------------> Training for model : ", test_model.
    model_name, " <-------------------------")
45
46          # train the model
47          history = test_model.train(load_checkpoint=True, ifearlystop=True)
48
49          # Plot the model
50          plot_model(test_model.model_type, to_file=DISS_CODE_FIG_DIR + '/' +
    model_name + '.png')
51
52          # Evaluate the model
53          test_model.evaluate(history)
54
55          # plot data with PCA
56          # test_model.plot_PCA()
57
58          # Plot the average receiver operating characteristic for each class
59          test_model.plot_ROC()
60
61          np.set_printoptions(precision=2)  # set the precision of floats in 2
    decimeter
62          ## Plot non-normalized confusion matrix
63
64          test_model.plot_confusion_matrix(normalize=False)
65
66      plt.tight_layout()
67      plt.show()
68      plt.close()
69
70
71  if __name__ == '__main__':
```

```
72      main()
```

Κώδικας 8.9: Run File