Ατομική Διπλωματική Εργασία

ΠΡΟΒΛΕΨΗ ΔΕΥΤΕΡΟΤΑΓΟΥΣ ΔΟΜΗΣ ΤΩΝ ΠΡΩΤΕΪΝΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ CONVOLUTIONAL NEURAL NETWORKS ΓΙΑ ΟΠΤΙΚΗ ΑΝΑΓΝΩΡΙΣΗ ΑΝΤΙΚΕΙΜΕΝΩΝ

Παναγιώτης Παυλίδης

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάιος 2016

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόβλεψη Δευτεροταγούς Δομής Πρωτεϊνών Με Τη Χρήση Των Convolutional Neural Networks Για Οπτική Αναγνώριση Αντικειμένων

Παναγιώτης Παυλίδης

Επιβλέπων Καθηγητής

Δρ. Χρίστος Χριστοδούλου

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2016

Ευχαριστίες

Με την παρούσα διπλωματική εργασία κλείνει ένας κύκλος. Ο φοιτητικός κύκλος ζωής αναμφίβολα γίνεται σημείο απορίας, δυσανασχετήσεων, ευχαριστιών, δημιουργίας και προπαντός έρευνας. Κύκλος όπου σε κάποιο σημείο του, κανείς, διερωτάται πώς το επιστημονικό πεδίο μπορεί να φανεί χρήσιμο σε μια κοινωνία. Η διπλωματική εργασίας μπορεί να δώσει τα κατάλληλα ερεθίσματα για τη επιστημονική επαγρύπνηση ενός φοιτητή.

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τον καθηγητή Δρ. Χρίστο Χριστοδούλου για την εμπιστοσύνη που έδειξε τόσο σε εμένα, όσο και στην Ειρήνη Παπακώστα για την συνεργασία μας στο θέμα της πρόβλεψης της δευτεροταγούς δομής των πρωτεϊνών, με κάπως τρελές τεχνικές. Η μέχρι στιγμής συνεργασία μας μόνο θετικά μου έχει προσδώσει, αφού το υγιές κλίμα εργασίας επέτρεπε την έκταση της διπλωματικής αυτής.

Θα ήθελα, επίσης, να ευχαριστήσω τον διδακτορικό φοιτητή του κυρίου Χριστοδούλου, Μιχάλη Αγαθοκλέους, που σε στιγμές άγνοιας για τους τρόπους επίλυσης του προβλήματος, ήταν σε θέση να ανατροφοδοτεί την προσπάθειά μας, κάνοντας ανάκληση των μέχρι στιγμής προσπαθειών του, στο παρόν πρόβλημα.

Ευχαριστίες θα ήθελα να δώσω στους συμφοιτητές μου Άντρια Κκουσιή, Ειρήνη Παπακώστα και Ανδρέα Φράγκου, άτομα όπου εμπλέκονται σε όλες τις φοιτητικές μου εμπειρίες. Επιπλέον ευχαριστώ το Θεατρικό Όμιλο του Π.Κ., όπως και όλους μου τους φίλους, που ήταν σε ετοιμότητα να βοηθήσουν στην επανάκτηση της χαμένης ενέργειας που αναλωνόταν στην καθημερινότητα του τμήματος.

Τέλος, ίσως οι σημαντικότερες ευχαριστίες κατανέμονται στην οικογένειά μου, που η αμέριστη υποστήριξη της είναι συνοδοιπόρος με ότι και αν καταπιάστηκα στη ζωή μου.

Περίληψη

Οι πρωτεΐνες διαδραματίζουν σημαντικό ρόλο στον ανθρώπινο οργανισμό αφού η παρουσία τους στα κύτταρα είναι απαραίτητη προκειμένου αυτά να εκτελέσουν κανονικά τις λειτουργίες του οργανισμού. Ανήκουν στην κατηγορία των μακρομορίων και αποτελούνται από μια ή περισσότερες αλυσίδες αμινοξέων. Ιδιαίτερης σημασίας είναι ο τρόπος που σχηματίζονται οι πρωτεΐνες, όπου αλληλουχίες είκοσι διαφορετικών αμινοξέων αναδιπλώνονται στο χώρο κατά μοναδικό τρόπο, προσδίδοντας σε αυτές μοναδικό σχήμα και κατ' επέκταση μοναδική λειτουργία. Το σχήμα και η λειτουργία τους αυτή εκφράζεται με το σχηματισμό της τριτοταγούς τους δομής, με αποτέλεσμα ο άνθρωπος να μπορεί να μελετήσει σε ιατροφαρμακευτικό επίπεδο, τις πρωτεΐνες, σε αυτή τους τη δομή. Οι πειραματικές μέθοδοι προσδιορισμού της τριτοταγούς δομής είναι χρονοβόρες και ασύλληπτα δαπανηρές, αφήνοντας την επιστημονική κοινότητα με μερική μόνο γνώση των λειτουργιών των πρωτεϊνών, αφού μόλις μερικές χιλιάδες (από τις εκατομμύρια που υπάρχουν) πρωτεΐνες έχουν μελετηθεί. Ως ενδιάμεσο βήμα, για την μελέτη της τριτοταγούς δομής, είναι η εκμάθηση της δευτεροταγούς δομής όπου αυτή, με τη σειρά της, κάνει χρήση της πρωτοταγούς δομής (αλληλουχία αμινοξέων) για να προβλέψει το σχηματισμό των δομικών στοιχείων (α-έλικες, β-κλώνοι κ.α.) που προκύπτουν από αυτή [7].

Στην προσπάθεια επίλυσης του προβλήματος της πρόβλεψης της δευτεροταγούς δομής των πρωτεϊνών (Protein Secondary Structure Prediction - PSSP), η ερευνητική ομάδα του Π.Κ που ασχολείται με το πρόβλημα αυτό, κατάφερε να πετύχει ακρίβεια πρόβλεψης 78% με το δίκτυο BRNN (Bidirectional Recurrent Neural Network), όπως αυτό περιγράφεται από τον Baldi (1999). Η διπλωματική, αυτή, εργασία έχει σκοπό την επέκταση των ερευνητικών δοκιμών στο θέμα αυτό, με τη χρήση των δικτύων Convolutional Neural Networks (CNN). Το δίκτυο CNN, που ανήκει στην οικογένεια των Deep Neural Networks, έχει ανθίσει στο τομέα αναγνώρισης εικόνας, φωνής και γενικά σε προβλήματα που σχετίζονται με την αναγνώριση χαρακτηριστικών, προκειμένου να κατηγοριοποιήσει τα δεδομένα που δίδονται σε αυτό. Κάνοντας οπτικοποίηση των δεδομένων εισόδου, δηλαδή της αλληλουχίας αμινοξέων, προσπαθούμε να εκπαιδεύσουμε το δίκτυο για να βρίσκει χαρακτηριστικά που να μας δίνουν την πιθανή κατηγορία που ανήκει το δεδομένο. Οι καλύτερη κατηγοριοποίηση που δόθηκε από το δίκτυο, είχε ποσοστό ακρίβειας 42.89%.

Περιεχόμενα

Contents

Κεφάλαιο 1

Εισαγωγή

1.1 Βιολογικό υπόβαθρο

- 1.1.1 Πρωτεΐνες και πολυπεπτίδια
- 1.1.2 Δομή των Πρωτεϊνών

1.2 Σχετική έρευνα του PSSP προβλήματος

1.3 Τεχνητά Νευρωνικά δίκτυα

- 1.3.1 Μάθηση
- 1.3.2 Συναρτήσεις ενεργοποίησης
- 1.3.3 Πολυστρωματικά δίκτυα perceptron εμπρόσθιου περάσματος
- 1.3.4 Αλγόριθμος μάθησης

1.4 Σχετική έρευνα νευρωνικών δικτύων

- 1.4.1 Deep Neural Networks
- 1.4.2 Αρχιτεκτονική
- 1.4.3 Σχετική έρευνα βασισμένη στο NLP

Εισαγωγή

Το πρόβλημα

Οι πρωτεΐνες διαδραματίζουν σημαντικό ρόλο στον ανθρώπινο οργανισμό αφού η παρουσία τους στα κύτταρα είναι απαραίτητη προκειμένου αυτά να εκτελέσουν κανονικά τις λειτουργίες του οργανισμού.

Η μελέτη της δομής των πρωτεϊνών αποτελείται από τέσσερα επίπεδα και περιλαμβάνει την πρωτοταγή, δευτεροταγή, τριτοταγή και τεταρτοταγή δομή. Επιπλέον, δεν είναι αρκετή η γνώση των αμινοξέων που αποτελείται μια πρωτεΐνη. Μεγάλης σημασίας χρίζει ο προσδιορισμός της σειράς των αμινοξέων αφού η αλληλουχία αυτή καθορίζει τις ιδιότητες της πρωτεΐνης. Είναι, επίσης, γνωστό ότι το περιβάλλον που εκτίθενται τα διάφορα αμινοξέα επηρεάζει τη σειρά αυτή. Επομένως, η αλληλουχία των αμινοξέων αποτελεί την πρωτοταγή δομή. Ξέρουμε επίσης ότι η αλληλουχία των αμινοξέων δεν είναι μια ευθεία γραμμή, αλλά ότι αυτή αναδιπλώνεται στο χώρο, βάσει της σειράς με την οποία συντάσσεται οι αλληλουχία αυτή. Μελέτες που διεξήχθησαν, με χρήση κρυσταλλογραφιών ακτινών Χ, έδειξαν ότι οι αναδιπλώσεις αυτές χωρίζονται σε δυο *κύριες* κατηγορίες: την α-έλικας και την β-πτυχωτή. Συμπερασματικά, η δευτεροταγής δομή ασχολείται με τις αναδιπλώσεις αυτές και κατ' επέκταση η πρόβλεψη της δευτεροταγούς δομής αφορά την πρόβλεψη των αναδιπλώσεων αυτών. Η έρευνα που διεξάγεται αυτή τη στιγμή, από το πανεπιστήμιό μας, αφορά την πρόβλεψη της δευτεροταγούς δομής, συνεπώς προκειμένου να πραγματοποιήσουμε μια τέτοια πρόβλεψη θα πρέπει να εστιάσουμε και στη δομή της πρωτοταγούς δομής.

Η τριτοταγής δομή ασχολείται με τη σειρά των διαφόρων αναδιπλώσεων αφού, στην ουσία, μια πρωτεΐνη αποτελείται από μια σειρά από α-ελικοειδές και β-πτυχωτές αναδιπλώσεις που προσδίδουν στην πρωτεΐνη το τελικό της σχήμα και λειτουργία. Η τριτοταγής δομή, παρέχει όλη την πληροφορία που χρειαζόμαστε για μια πρωτεΐνη, αλλά η ανάλυσή της αποτελεί ακόμη δυσκολότερο κομμάτι μελέτης για τη βιολογία. Η πρόβλεψη της δευτεροταγούς δομής των πρωτεϊνών αποτελεί ένα από τα σημαντικότερα προβλήματα της βιοπληροφορικής, λόγω του ότι μια πετυχημένη πρόβλεψη θα αποτελέσει κυρίαρχο βήμα στην πρόβλεψη της τριτοταγούς δομής. Η γνώση αυτή θα μας

προσδιορίσει τα περιβάλλοντα που καθορίζουν τη δημιουργία των πρωτεϊνών με αποτέλεσμα να βοηθήσει τη φαρμακοβιομηχανία, και την ιατρική γενικότερα, να εμποδίσει την ανάπτυξη πρωτεϊνών που κουβαλούν μαζί τους ανίατες ασθένειες.

Για το λόγο ότι η επεξεργασία και μελέτη των διαφόρων δομών των πρωτεϊνών είναι μια πολύ δαπανηρή μέθοδος, προσπαθούμε να προβλέψουμε τη δευτεροταγή δομή, βοηθώντας την περαιτέρω ανάλυση του πώς μια αλληλουχία αμινοξέων μπορεί να συμβάλει στη δημιουργία κάποιας πρωτεΐνης.

Deep Networks

Τα "βαθιά" νευρωνικά δίκτυα (Deep Networks) εμπίπτουν στην κατηγορία των αλγορίθμων μηχανική μάθησης και είναι αλγόριθμοι που αναπτύχθηκαν τις προηγούμενες δεκαετίες αλλά λόγω έλλειψης του μεγάλου όγκου δεδομένων που χρειάζονταν, δεν παρουσίασαν περαιτέρω άνθηση.

Η εποχή των Big Data είναι πλέον γεγονός, ενδεχομένως η ανάγκη για καλύτερο feature extraction (εξαγωγή χαρακτηριστικών) εμφανίζεται έντονα στις μέρες μας. Τα πεδία εφαρμογής που μας οδηγούν στη χρήση τέτοιων δικτύων έχουν να κάνουν με προβλήματα αναγνώρισης χαρακτηριστικών (όπως αναγνώριση φωνής, αναγνώριση προσώπου από εικόνες), λόγω του ότι προσδίδουν καλύτερα αποτελέσματα.

Convolutional Neural Networks

Η εκπόνηση της ατομικής διπλωματικής εργασίας στοχεύει στην εξέταση του πώς οι Convolutional Neural Networks (CNN) αλγόριθμοι μπορούν να βοηθήσουν το πρόβλημα της πρόβλεψης της δευτεροταγούς δομής των πρωτεϊνών. Αναλυτικότερα, τέτοια δίκτυα εκμεταλλεύονται τη χωρική δομή των δεδομένων εισόδου (Κεφάλαιο 3), γεγονός που μας γεμίζει ελπίδα για καλύτερα αποτελέσματα από ότι τα BRNN δίκτυα.

Επιπλέον, θεωρητικά, τα Convolutional Neural Networks κάνουν καλύτερη διαχείριση των δεδομένων εισόδου σε προβλήματα που έχουν να κάνουν με αλληλουχίες ή γενικότερα αυτά που λαμβάνουν ως παράμετρο το χώρο (όπως επεξεργασία εικόνας).

Web Application

Το δεύτερο μέρος της διπλωματικής μου εργασίας αφορά την από κοινού ανάπτυξη διαδικτυακής εφαρμογής που αποσκοπεί στην απομακρυσμένη κλήση των αλγορίθμων που υλοποιήθηκαν μέχρι στιγμής, από την ομάδα, σε συνεργασία με την συμφοιτήτριά μου Ειρήνη Παπακώστα. Η δημοσιοποίηση της μέχρι στιγμής έρευνας που έγινε, θα επιτρέψει τη διαλειτουργικότητα μεταξύ διαφορετικών frameworks, αφού η πλατφόρμα κάνει χρήση του HTTP πρωτοκόλλου, με αποτέλεσμα να είναι πολύ πιο προσιτή προς το κοινό. Επιπλέον, θα αποτελέσει σημείο αναφοράς της προόδου που γίνεται στο ερευνητικό κομμάτι όσον αφορά το πρόβλημα της πρόβλεψης της δευτεροταγούς δομής των πρωτεϊνών.

Για την τεκμηρίωση της διαδικτυακής εφαρμογής επιλέξαμε να έχουμε τα ίδια αποσπάσματα στις διπλωματικές μας εργασίες, ως ένδειξη σεβασμού προς τον αναγνώστη, αφού μπορεί να θεωρηθεί αντιεπαγγελματική η συνεχής εναλλαγή διπλωματικών εργασιών, προκειμένου να αναγνωστεί όλη η τεκμηρίωση.

Η τεκμηρίωση της διαδικτυακής εφαρμογής, βρίσκεται στο Κεφάλαιο 2, και γράφτηκε σε συνεργασία με την Ειρήνη Παπακώστα.

1.1 Βιολογικό υπόβαθρο

1.1.1 Πρωτεΐνες και πολυπεπτίδια

Πρόκειται για μακρομοριακές ενώσεις που αποτελούν το 50% της ξηρής κυτταρικής μάζας. Πέραν από τη δομική σημασία τους, οι πρωτεΐνες συμβάλουν στην διεκπεραίωση των λειτουργιών του κυττάρου, όπως κίνηση, άμυνα, μεταφορά ουσιών κ.α. (Πίνακας 1.1).

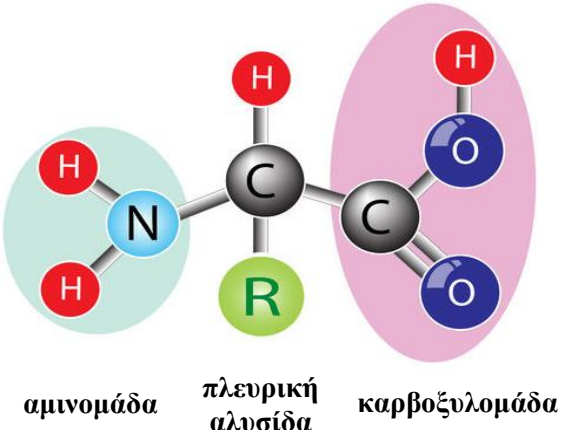
Είδος πρωτεΐνης	**Ρόλος**
Δομικές πρωτεΐνες	Στήριξη
Αποταμιευτικές πρωτεΐνες	Μεταφορά άλλων ουσιών
Πρωτεΐνες – Ορμόνες	Έλεγχος δραστηριοτήτων
Πρωτεΐνες – Υποδοχείς	Αντίδραση στα χημικά ερεθίσματα
Συσταλτικές πρωτεΐνες	Κίνηση
Αμυντικές πρωτεΐνες	Προστασία κατά ασθενειών
Πρωτεΐνες – Ένζυμα	Επιτάχυνση και έλεγχος των πρωτεϊνών
Ρυθμιστικά διαλύματα	Ρύθμιση του pH του πρωτοπλάσματος
Καύσιμα υλικά	Απελευθέρωση ενέργειας

Πίνακας 1.1: Βιολογικοί ρόλοι των πρωτεϊνών

Σχηματίζονται στα ριβοσώματα, με τη διαδικασία της πρωτεινοσύνθεσης, με τη διαδοχική συνένωση απλούστερων χημικών ενώσεων, των αμινοξέων. Υπάρχει μεγάλος αριθμός αμινοξέων αλλά μόνο είκοσι συμμετέχουν στη δομή των πρωτεϊνών (Πίνακας 1.2). Τα περισσότερα αμινοξέα αποτελούνται από ένα ασύμμετρο άτομο άνθρακα συνδεδεμένο ομοιοπολικά με ένα άτομο υδρογόνου, μια καρβοξυλομάδα, μια αμινομάδα και μια πλευρική ομάδα (Εικόνα 1.1). Η μεταβλητότητα των αμινοξέων εμφανίζεται στην πλευρική ομάδα (R), προσδίδοντας στο αμινοξύ διαφορετικές ιδιότητες και κατ' επέκταση κατηγοριοποίηση του σε μια από τις επτά ομάδες (αμινοξέα με αλειφατικές - μη πολικές πλευρικές αλυσίδες, αμινοξέα με αλειφατικές - πολικές πλευρικές αλυσίδες, αμινοξέα με αρωματικές πλευρικές αλυσίδες, αμινοξέα με βασικές πλευρικές αλυσίδες, αμινοξέα με όξινες πλευρικές αλυσίδες, αμινοξέα με καρβοξυλαμίδια στις πλευρικές αλυσίδες τους, ειδικά αμινοξέα).

Δυπεπτίδιο ονομάζεται ο πεπτιδικός δεσμός που σχηματίζουν δυο αμινοξέα όταν ενώνονται μεταξύ τους, αποβάλοντας ένα μόριο νερού. Πολλά αμινοξέα που ενώνονται με αυτό τον τρόπο σχηματίζουν ένα πολυπεπτίδιο ή πολυπεπτιδική αλυσίδα. Η ποικιλομορφία που μπορεί να εμφανιστεί, στο σχηματισμό της πρωτεΐνης, από μια ή

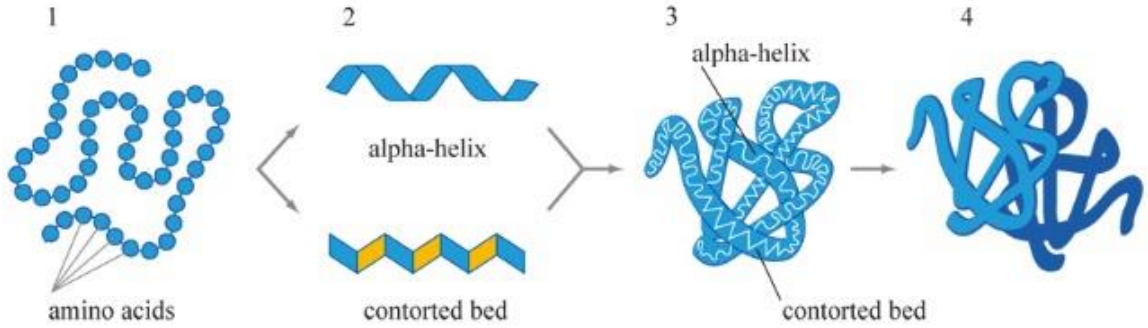
περισσότερες πολυπεπτιδικές αλυδίδες αυξάνουν σημαντικά την αριθμό διαφορετικών πρωτεϊνών. Εν συνεχεία, ο μοναδικός τρόπος που αναδιπλώνονται οι πολυπεπτιδικές αλυσίδες στο χώρο, προσδίδουν στην πρωτεΐνη τη μοναδική τρισδιάστατη μορφή και σχήμα. Η αλληλουχία, επομένως, των αμινοξέων στο πολυπεπτίδιο διαδραματίζει σημαντικό ρόλο ως προς το τελικό σχήμα που θα πάρει η πρωτεΐνη στο χώρο.



Εικόνα 1.1: Δομή αμινοξέως

Ένα πεπτίδιο, αμέσως μετά τη σύνθεσή του είναι ανενεργό, ανίκανο δηλαδή να εκδηλώσει το βιολογικό του ρόλο. Για να γίνει ενεργό πρέπει πρώτα να αποκτήσει την οριστική του στερεοδιάταξη, η οποία οργανώνεται σε τέσσερα επίπεδα: την πρωτοταγή, τη δευτεροταγή, τριτοταγή και τεταρτοταγή δομή (Εικόνα 1.2).

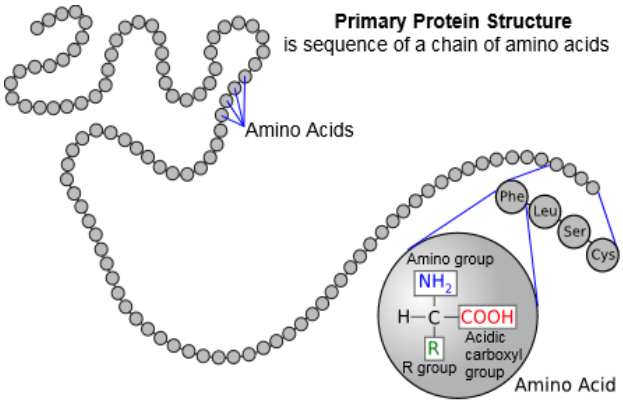
1.1.2 Δομή των Πρωτεϊνών



Εικόνα 1.2: Τέσσερα επίπεδα δομής των πρωτεϊνών

Πρωτοταγής Δομή

Η πρωτοταγής δομή (Εικόνα 1.3) χαρακτηρίζεται από την αλληλουχία των αμινοξέων στην πολυπεπτιδική αλυσίδα. Ανεπαρκής θεωρείται η απλή γνώση των αμινοξέων που απαντώνται στα μόρια μιας πρωτεΐνη. Μεγάλης σημασίας χρίζει ο προσδιορισμός της σειράς των αμινοξέων αφού η αλληλουχία αυτή καθορίζει τις ιδιότητες της πρωτεΐνης. Οι θέσεις που κατέχουν τα αμινοξέα δεν είναι τυχαίες, αλλά καθορίζονται από το γενετικό υλικό, καθώς επίσης είναι γνωστό ότι το περιβάλλον που εκτίθενται τα διάφορα αμινοξέα επηρεάζει τη σειρά αυτή.

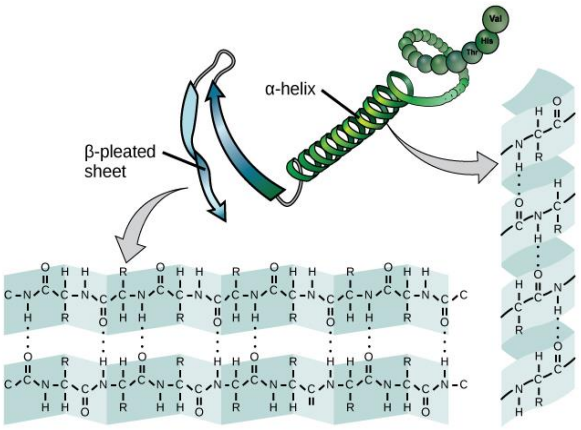


Εικόνα 1.3: Πρωτοταγής δομή

Δευτεροταγής δομή

Οι πολυπεπτιδικές αλυσίδες δεν είναι ευθύγραμμες αλλά συσπειρώνονται κατά συγκεκριμένο (μοναδικό) τρόπο. Μελέτες που διεξήχθησαν, με χρήση κρυσταλλογραφιών ακτινών Χ, έδειξαν ότι οι αναδιπλώσεις αυτές χωρίζονται, συνήθως, σε δυο κύριες κατηγορίες: την α-έλικας και την β-πτυχωτή (Εικόνα 1.4) [8, 9]. Υπεύθυνοι για τις αναδιπλώσεις της δευτεροταγούς δομής είναι οι δεσμοί υδρογόνου. Στις β-πτυχώσεις, για παράδειγμα, τμήματα της αλυσίδας που είναι παράλληλα, συγκροτούνται με δεσμούς υδρογόνου, που αναπτύσσονται μεταξύ τους, σταθεροποιώντας την πτυχωτή δομή [10]. Συμπερασματικά, η δευτεροταγής δομή ασχολείται με τις αναδιπλώσεις αυτές

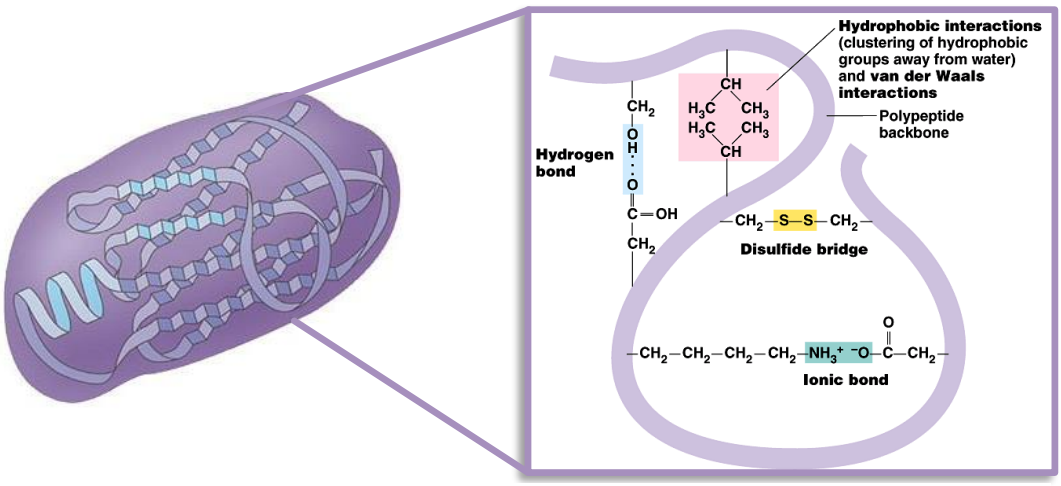
και κατ' επέκταση η πρόβλεψη της δευτεροταγούς δομής αφορά την πρόβλεψη των αναδιπλώσεων αυτών.



Εικόνα 1.4: Δευτεροταγής δομή *(Πάνω: παράδειγμα αναδίπλωσης β-πτυχωτής επιφάνειας. Κάτω: παράδειγμα αναδίπλωσης α έλικας)*

Τριτοταγής δομή

Η τριτοταγής δομή (Εικόνα 1.5) ασχολείται με τη σειρά των διαφόρων αναδιπλώσεων της πολυπεπτιδικής αλυσίδας, αφού ουσιαστικά, η δομή αυτή είναι το αποτέλεσμα της ένωσης υδρογόνου μεταξύ των πλευρικών ομάδων των αμινοξέων. Μια πρωτεΐνη αποτελείται από μια σειρά από α-ελικοειδείς και β-πτυχωτές αναδιπλώσεις που προσδίδουν στην πρωτεΐνη το τελικό της σχήμα και λειτουργία. Παράγοντες που επηρεάζουν την τελική μορφή της πρωτεΐνης είναι: οι υδρόφοβες αλληλεπιδράσεις, οι δεσμοί υδρογόνου που αναπτύσσονται μεταξύ πλευρικών ομάδων, ιοντικές έλξεις και η παρουσία δισουλφιδικών δεσμών [10].



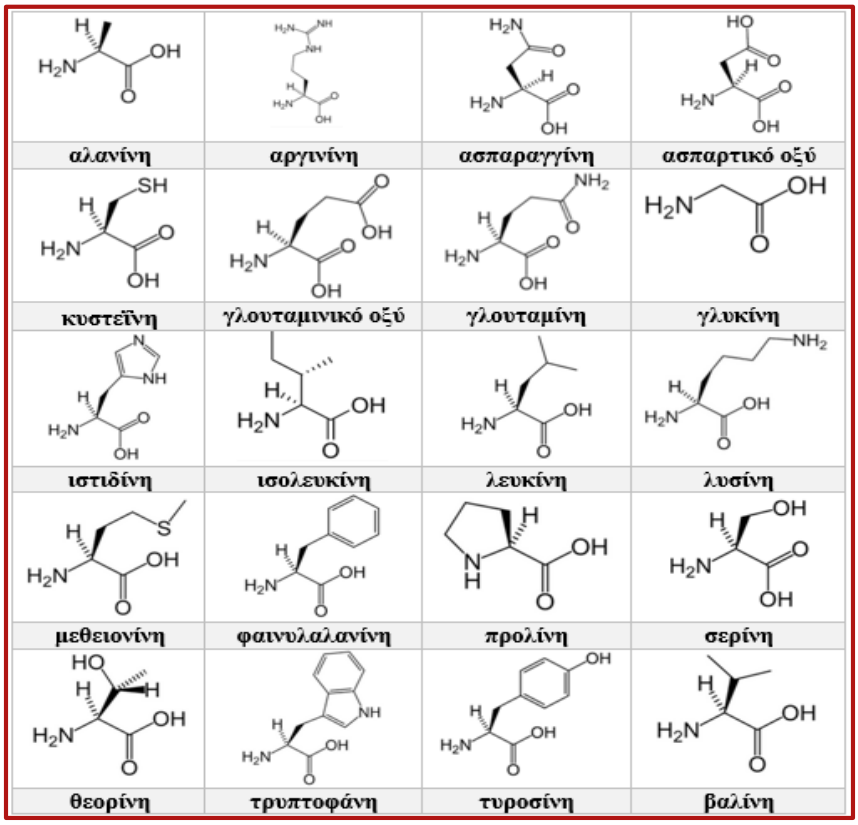
Εικόνα 1.5: Τριτοταγής δομή

Τεταρτοταγής Δομή

Τέλος, μπορεί ο συνδυασμός διαφόρων πολυπεπτιδικών αλυσίδων, που σχηματίστηκαν, να συνενωθούν. Το πρωτεϊνικό σύμπλοκο αποτελεί την τεταρτοταγή δομή της πρωτεΐνης.

Γενικά

Η αλληλουχία των αμινοξέων (πρωτοταγής δομή) καθορίζει και την τριτοταγή δομή της πρωτεΐνης. Πέραν από την ακολουθία αυτή, η τρισδιάστατη μορφή επηρεάζεται από παράγοντες όπως θερμοκρασία, pH κ.α.. Αλλάζοντας το περιβάλλον ύπαρξης της πρωτεΐνης, αυτή αποδιατάσσεται, χάνεται δηλαδή η βιολογική της λειτουργία αφού η τρισδιάστατή της μορφή αλλάζει. Όπως προαναφέρθηκε, για να φτάσουμε στη μελέτη της τριτοταγούς δομής, γνωρίζοντας μόνο την πρωτοταγή, πρέπει να γνωρίζουμε τον ακριβή τρόπο δόμησης της δευτεροταγούς δομής.



Πίνακας 1.2: Τα είκοσι αμινοξέα που σχηματίζουν τις πρωτεΐνες

1.2 Σχετική έρευνα του PSSP προβλήματος

(Σημείωση: PSSP = Protein Secondary Structure Prediction = πρόβλεψη δευτεροταγούς δομής των πρωτεϊνών)

Το 2005, το πρόβλημα της μελέτης του τρόπου αναδίπλωσης των πρωτεϊνών κατατάχθηκε από το περιοδικό Science [11] ως ένα από τα 125 μεγαλύτερα άλυτα προβλήματα της επιστήμης. Με την πάροδο των χρόνων έχουν μελετηθεί πρακτικά και θεωρητικά θέματα που αφορούν το πρόβλημα αυτό. Εντούτοις, ο πρωτεϊνικός κώδικας εξακολουθεί να υποδιαιρείται σε τρία μεγάλα υπό-προβλήματα [12]: το πώς οι διατομικές δυνάμεις των αμινοξεών οδηγούν στην αναδίπλωση της πρωτεΐνης, το υπολογιστικό πρόβλημα που αφορά την πρόβλεψη της δευτεροταγούς δομής των πρωτεϊνών (Protein Secondary Structure Prediction - PSSP) και η απάντηση στο "Levinthal's paradox" [13] σχετικά με την ταχύτητα αναδίπλωσης μιας πρωτεΐνης.

Ιστορική αναδρομή προσεγγίσεων του PSSP πρόβλημα

Με την πάροδο των χρόνων, έχει αναπτυχθεί πληθώρα μεθόδων με σκοπό την επίλυση του προβλήματος. Οι μέθοδοι αυτοί χωρίζονται σε στατιστικές και υπολογιστικές.

Αρχίζοντας με τις στατιστικές μεθόδους, η Chou-Fasman [15], υπολογίζει την πιθανότητα ύπαρξης ενός αμινοξέως στις α-έλικες ή στις β-πτυχωτές βασισμένη στη συχνότητα εμφάνισης του αμινοξέως. Η μετρική αυτή αναπτύχθηκε τα μέσα της δεκαετίας του ’70 και κατάφερε να προβλέψει μόλις το 50-60% της δευτεροταγούς δομής. Αργότερα, η GOR μέθοδος, εξέφρασε την πιθανότητα ένα αμινοξύ να ανήκει σε μια από τις κατηγορίες helix (H), coil (C) και extended (E) κάνοντας χρήση των γειτονικών αμινοξέων. Η αρχική προσέγγιση της GOR μεθόδου έδωσε ποσοστό 65%, ενώ αργότερα, με κάποιες επεκτάσεις, δόθηκε αποτέλεσμα 73,5% [9].

Η μηχανική μάθηση (τεχνητά νευρωνικά δίκτυα - ΤΝΔ), που ανήκει στη κατηγορία των υπολογιστικών μεθόδων, έχει γνωρίσει άνθιση, τον τελευταίο μισό αιώνα, σε προβλήματα που αφορούν τη αναγνώριση χαρακτηριστικών (pattern recognition). Τα ΤΝΔ, μπορούν να μάθουν κάνοντας χρήση των δεδομένων που χρησιμοποιούνται για να τροφοδοτείται το δίκτυο, αντιστοιχώντας τα με το ανάλογο επιθυμητό αποτέλεσμα. Περισσότερα για τα ΤΝΔ καλύπτονται στο Υποκεφάλαιο 1.3.

Βιοπληροφορική και το PSSP πρόβλημα

Η εκπόνηση της διπλωματικής, αυτής, εργασίας έχει σκοπό τη μελέτη του προβλήματος της πρόβλεψης της δευτεροταγούς δομής των πρωτεϊνών. Επί της ουσίας, η προσπάθεια που γίνεται από τον τομέα της βιοπληροφορικής είναι η εύρεση αλγορίθμου που να δέχεται ως είσοδο μια πολυπεπτιδική αλυσίδα (αλυσίδα αμινοξέων) και να εξάγει τη δευτεροταγή δομή της πρωτεΐνης αυτής.

$$Q_3 = \frac{αριθμός\ καταλείπων\ που\ προβλέφθηκαν\ ορθά}{συνολικό\ αριθμός\ καταλείπων} * 100$$

Εξίσωση 1.1: Μετρική ποσοστού επιτυχίας [8].

Σχετική έρευνα του Π.Κ.

Η μέχρι στιγμής σχετική έρευνα που διεξάγεται στο Π.Κ. (Πίνακας 1.3) έχει επιφέρει ως βέλτιστο αποτέλεσμα ποσοστό επιτυχίας της τάξης του 76% [17] *(Q3 μετρική – εξίσωση 1.1)*, μεταξύ άλλων πρακτικών που μελετήθηκαν. Το ποσοστό αφορά το δίκτυο αμφίδρομης ανάδρασης, όπως περιγράφεται από τον Baldi [14].

Μέθοδος	Q3 (%)
Feedforward Fully Connected NN (Qian και Sejnowski, 1988)	63.300
PHD (Rost, 2001; Rost και Sander, 1993)	71.400
DSC (King και Sternberg, 1996)	71.950
NNSSP (Salamov και Soloveyev, 1997)	68.413
PREDATOR (Frishman και Argos, 1997)	68.602
Consensus (Cuff και Barton, 1999)	72.707
BRNN – Backpropagation (Baldi,et al., 1999)	76.000
LAD (Jacek et al., 2005)	70.600
MASSP3 (Giuliano A. et al., 2005)	76.100
Two-Stage method (Fadime U. Et al., 2007)	74.100
Cascade BRNN (Jinmiao C. και Narendra S.C., 2007)	74.380

Πίνακας 1.3: Δίκτυα που μελετήθηκαν [8], και τα ποσοστά επιτυχίας τους

Η προσπάθεια διεύρυνσης της παρούσας έρευνας του πανεπιστημίου, έχει ως αποτέλεσμα την αναζήτηση δικτύων που πιθανών να βελτιώσουν τα μέχρι στιγμής αποτελέσματα. Με το σκεπτικό αυτό, προτάθηκε η χρήση δικτύων που ανήκουν στην οικογένεια των Deep Neural Networks (Υποκεφάλαιο 1.2) και συγκεκριμένα το δίκτυο Convolutional Neural Networks.

1.3 Τεχνητά Νευρωνικά Δίκτυα

Νευρωνικά δίκτυα ονομάζονται τα μαθηματικά κομβικά μοντέλα προσομοίωσης του ανθρώπινου εγκεφάλου, που αναπτύσσονται για την επίλυση πολύπλοκων προβλημάτων. Παρά τις διαφορές τους, ο τομέας έρευνας και ανάπτυξης των ΤΝΔ βρίσκεται κάτω από την ομπρέλα της Τεχνητής Νοημοσύνη. Ο όρος "τεχνητά" τονίζει την προσπάθεια των δικτύων να μιμηθούν ένα τμήμα νευρικού ιστού του εγκεφάλου, που αποτελείται από βιολογικούς νευρώνες. Οι αλγόριθμοι ΤΝΔ αφορούν την εύρεση ενός αλγορίθμου για την επίλυση ενός υπολογιστικού προβλήματος ή της υπολογιστικής νευροεπιστήμης (προσομοίωση της λειτουργίας των βιολογικών νευρώνων).

Το γεγονός ότι τα δίκτυα αυτά είναι εμπνευσμένα από τη λειτουργία και τους τρόπους εκμάθησης του ανθρώπινου εγκεφάλου, σε αντίθεση με τους υπολογιστικούς κανόνες, τα έχει βοηθήσει να ακμάσουν έντονα τα τελευταία χρόνια. Εκμεταλλευόμενα την ικανότητα του εγκεφάλου να ανταποκρίνεται σε μια πληθώρα δεδομένων και παράλληλα να διατηρεί κάποια ανεκτικότητα σε λάθη, τα ΤΝΔ έχουν δώσει απαντήσεις σε προβλήματα των οποίων τα δεδομένα εισόδου θεωρούνταν ακατανόητα.

1.3.1 Μάθηση

Μαθαίνουμε. Μια λέξη κοινότοπη αλλά παράλληλα καθημερινή διαδικασία για τον ανθρώπινο εγκέφαλο. Επιτρέψτε μου να εκφράσω μια αδόκιμη παρατήρηση: οφείλω να ομολογήσω ότι λόγο την δύσκολης εναλλαγής των συναπτικών βαρών των νευρωνικών δικτύων, μετά από πολλές επαναλήψεις, αρχίζω να καταλαβαίνω γιατί οι άνθρωποι είναι τόσο ξεροκέφαλα όντα! Αντιγράφοντας την ικανότητα της μάθησης, βασισμένη σε εμπειρίες και βιώματα, ο κλάδος της μηχανικής μάθησης έχει κατηγοριοποιήσει τα είδη εκπαίδευσης στα ακόλουθα: την επιβλεπόμενη μάθηση, τη μη επιβλεπόμενη μάθηση και την ενισχυτική μάθηση.

Επιβλεπόμενη μάθηση

Παραλληλίζοντας έναν υπολογιστικό αλγόριθμο με την καθημερινότητα, μπορούμε να προσδιορίσουμε την επιβλεπόμενη μάθηση, ως τη σχέση μαθητή και διδασκάλου, όπου για κάθε πράξη του μαθητή δίδεται σε αυτόν συγκεκριμένη επιθυμητή ανατροφοδότηση.

Αντίστοιχα, ένα ΤΝΔ που χρησιμοποιεί αυτή τη μέθοδο μάθησης περιμένει ανατροφοδότηση για κάθε δεδομένο εξόδου, που παράγει. Για να επιτευχθεί αλγοριθμικά, το παραπάνω, θα πρέπει να έχουμε στη διάθεσή μας, για κάθε δεδομένο εισόδου, μια αντιστοίχιση του επιθυμητού αποτελέσματος με την πραγματική έξοδο που λαμβάνουμε, πράγμα το οποίο είναι κατορθωτό σε μερικές, μόνο, κατηγορίες προβλημάτων.

Μη Επιβλεπόμενη μάθηση

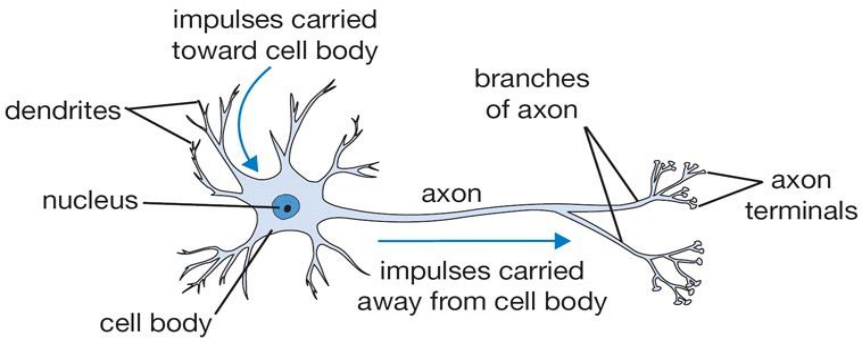
Χωρίς την ύπαρξη κάποιου δασκάλου για ανατροφοδότηση, το δίκτυο προσπαθεί μέσω των δεδομένων εισόδου να αναγνωρίσει κοινά χαρακτηριστικά μεταξύ αυτών, ούτως ώστε να τα κατηγοριοποιήσει.

Ενισχυτική Μάθηση

Ίσως η ρεαλιστικότερη μάθηση, αφού ανταποκρίνεται περισσότερο στον πραγματικό τρόπο εκμάθησης των ανθρώπων, θέτει ένα κριτή που γνωρίζει την απάντηση στο πρόβλημα και η μόνη ανατροφοδότηση που δίνεται στο μαθητή είναι η επιβράβευση ή η τιμωρία. Ένα παράδειγμα ενισχυτικής μάθησης από την πραγματική ζωή είναι το άγγιγμα στη φωτιά, όπου ανυποψίαστοι νεαροί μπορεί να την αγγίξουν, αλλά η τιμωρία που λαμβάνουν, τους αποτρέπει από τη σκόπιμη επανάληψη του συμβάντος. Στην περίπτωση αυτή ο μαθητής προσπαθεί να μειώσει τις ποινές που λαμβάνει αλλά ταυτοχρόνως να αυξάνει την επιβράβευσή του.

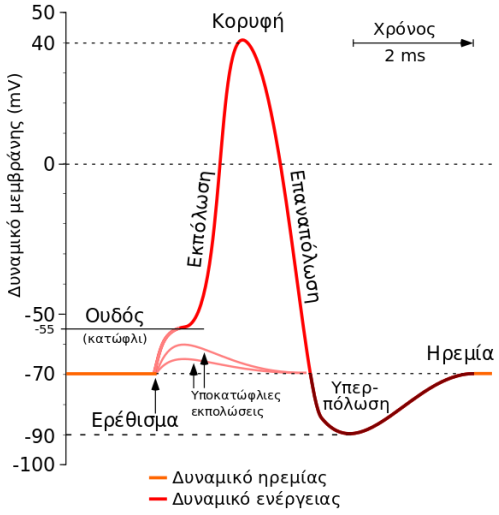
Σύγκριση βιολογικών και τεχνητών νευρώνων

(Σημείωση: οι τεχνητοί νευρώνες οι αρχιτεκτονική MLP που θα σχολαστούν αφορούν την επιβλεπόμενη μάθηση)



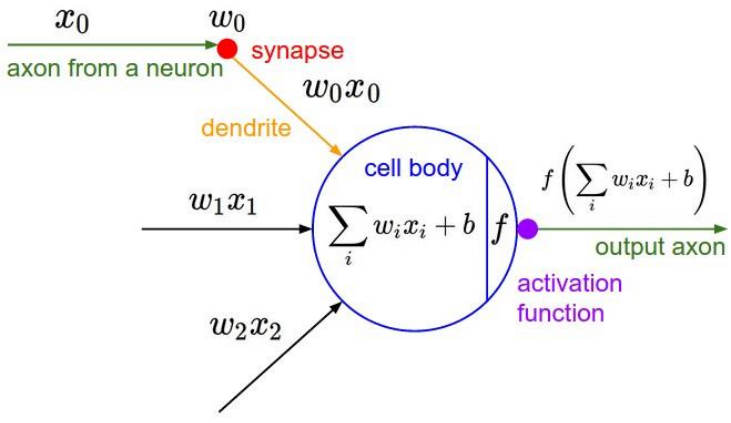
Εικόνα 1. 6 [37]: Βιολογικός Νευρώνας

Ένας βιολογικός νευρώνας (Εικόνα 1.6) επικοινωνεί με άλλους νευρώνες με νευρικές απολήξεις (συναπτικά δυναμικά σήματα – διεγερτικά ή κατασταλτικά). Φτάνοντας στους δενδρίτες του νευρώνα, τα δυναμικά σήματα εισόδου συναθροίζονται στο σώμα του νευρώνα, και όταν τα σήματα αυτά ξεπεράσουν κάποιο κατώφλι (Εικόνα 1.7), αλλάζει η κατάσταση του νευρώνα γίνεται στιγμιαία ενεργή, παράγοντας ένα ηλεκτρικό παλμό ο οποίος μεταφέρεται σε άλλους νευρώνες.



Εικόνα 1.7: Δυναμικό ενός βιολογικού νευρώνα κατά τη διάρκεια ενεργοποίησης του.

Με τρόπο παρόμοιο, ο νευρώνας τεχνητού νευρωνικού δικτύου (Εικόνα 1.8) συναθροίζει τις τιμές των εισόδων του, για να προκληθεί η τυχών πυροδότησή του και παράλληλα η μετάδοση του σήματός τους, στο υπόλοιπο δίκτυο.



Εικόνα 1.8 [37]: Τεχνητός νευρώνας

Η Εικόνα 1.8 παρουσιάζει τον τεχνητό νευρώνα, όπως αυτός προτάθηκε από τον McCulloch and Pitts το 1943 [18]. Η κάθε ακμή που φτάνει στο νευρώνα, παραδίδει σε αυτόν πληροφορία, ένα σήμα από το δεδομένο εισόδου – ή από κάποιο άλλο νευρώνα - και ακολούθως πολλαπλασιάζεται με την αντίστοιχη τιμή βάρους, που έχει η ακμή τη δεδομένη στιγμή. Ο πολλαπλασιασμός έχει ως αποτέλεσμα τη σχετική ενίσχυση ή αποδυνάμωση του σήματος αυτού. Στη συνέχεια, οι τιμές που αφήνουν οι πολλαπλασιασμοί συναθροίζονται για να δώσουν την τελική τιμή του νευρώνα, όπως φαίνεται από την Εξίσωση 1.1. Για να εξεταστεί το κατά πόσο ο νευρώνας θα πυροδοτήσει ή όχι, πρέπει να περάσουμε το αποτέλεσμα της εξίσωσης από μια συνάρτηση ενεργοποίησης.

$$u = \sum_{i=1}^{n} w_i x_i + b$$

Εξίσωση 1.1: Εξίσωση εισόδου στον τεχνητό νευρώνα.

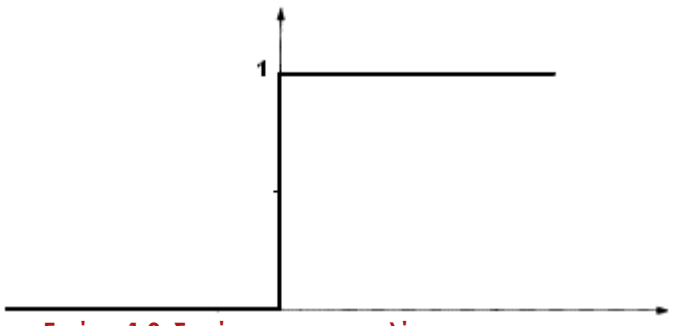
1.3.2 Συναρτήσεις ενεργοποίησης

Συνάρτηση κατωφλίου (Βηματική συνάρτηση - Step function)

Έστω θ η τιμή κατωφλίου, τότε όπως ο βιολογικός νευρώνας, έτσι και ο τεχνητός, εάν η συνάθροιση ξεπερνά την τιμή του κατωφλίου, ο νευρώνας πυροδοτεί διαφορετικά παραμένει ανενεργός (Εξίσωση 1.2, Εικόνα 1.9).

$$\psi = \begin{cases} 1 & \text{εάν } u \geq \theta \\ 0 & \text{εάν } u < \theta \end{cases}$$

Εξίσωση 1.2: Εξίσωση συνάρτησης κατωφλίου



Εικόνα 1.9: Συνάρτηση κατωφλίου

Η συνάρτηση κατωφλίου είναι χρήσιμη μόνο όταν δεδομένα εισόδου, είναι γραμμικά διαχωρίσιμα αφού εκτελεί γραμμικό διαχωρισμό στο χώρο των δεδομένων εισόδου. Δηλαδή, εάν αναπαραστήσουμε τα δεδομένα εισόδου στο χώρο, η βηματική συνάρτηση, καταφέρνει να βρει ένα γραμμικό διαχωρισμό, μεταξύ αυτών.

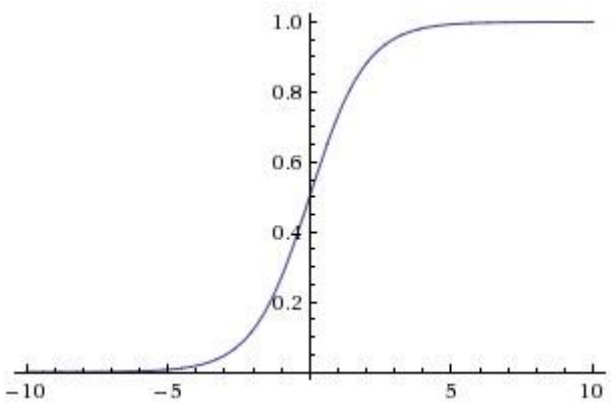
Δυστυχώς, η μόνη πληροφορία που μπορούμε να λάβουμε από ένα δεδομένο εισόδου, κάνοντας χρήση της βηματικής συνάρτησης, είναι η ενεργοποίηση ή μη-πυροδότηση του νευρώνα. Αυτό επηρεάζει την απόδοση του δικτύου μας αφού δεν γνωρίζουμε πόσο απέχουμε από το επιθυμητό αποτέλεσμα. Δεύτερον, οι παράμετροι που αλλάζουν κατά την εκπαίδευση του δικτύου είναι τα συναπτικά βάρη. Επομένως, σε περίπτωση λάθους θα θέλαμε να κάνουμε αλλαγή βαρών τέτοια ώστε στην επόμενη επανάληψη του δεδομένου εισόδου, το δίκτυο να μπορεί να ανταποκριθεί σωστά. Ψάχνουμε, συνεπώς, συνάρτηση που να μπορεί να μας δώσει διαφοροποίηση βαρών που να μειώνει την απόκλιση του δικτύου από το επιθυμητό αποτέλεσμα. Για να μπορεί μια συνάρτηση να μας δώσει την πληροφορία αυτή θα πρέπει να είναι παραγωγίσιμη σε όλο το πεδίο ορισμού, κάτι το οποίο δεν πληροί η συνάρτηση κατωφλίου. Τα δυο, παραπάνω, σοβαρά μειονεκτήματα μας οδηγούν στη χρήση άλλων συναρτήσεων ενεργοποίησης.

Σιγμοειδής συνάρτηση

Η σιγμοειδής (Εικόνα 1.10) είναι μη γραμμική συνάρτηση και έχει τη μαθηματική μορφή:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Εξίσωση 1.2: Εξίσωση σιγμοειδούς συνάρτησης



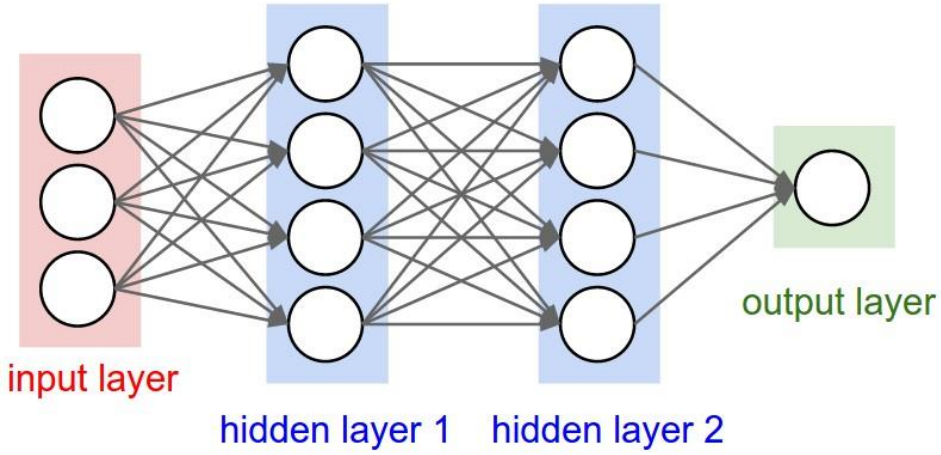
Εικόνα 1.10: Σιγμοειδής συνάρτηση

Όπως υπαινίχθηκε από τη γραφική παράσταση, η συνάρτηση αυτή λαμβάνει ως είσοδο ένα πραγματικό αριθμό και τον "στριμώχνει" μέσα στο πεδίο ορισμού [0, 1]. Ιδιαίτερα, η συνάρτηση αυτή δίνει τιμή 0 σε μεγάλες αρνητικές τιμές εισόδου, όπου και ο νευρώνας δεν πυροδοτεί καθόλου και τιμή 1 σε μεγάλες θετικές.

Αρνητικό της σιγμοειδούς

Όταν οι τιμές των βαρών της είναι κοντά στο 0 ή στο 1, τότε η παράγωγος είναι σχεδόν μηδέν, επομένως η εναλλαγή των βαρών δεν εκτελείται, νεκρώνοντας έτσι ένα κομμάτι του νευρώνα.

1.3.3 Πολυστρωματικά δίκτυα perceptron εμπρόσθιου περάσματος (Feed Forward) – MLP (Multi-Layer Perceptron)



Εικόνα 1.11 [37]: MLP αρχιτεκτονική

Προκειμένου να σχηματίσουμε ένα νευρωνικό δίκτυο πρέπει να δημιουργήσουμε τα διάφορα επίπεδα του, ενώνοντας τεχνητούς νευρώνες μεταξύ τους. Όπως φαίνεται από την Εικόνα 1.11 οι νευρώνες του επιπέδου εισόδου (input layer) λαμβάνουν τα δεδομένα εισόδου, οι νευρώνες των κρυφών επιπέδων (hidden layer) κρατούν τη "σοφία" του δικτύου εφαρμόζοντας μια συνάρτηση ενεργοποίησης και τέλος το επίπεδο εξόδου (output layer) υπολογίζει και δίνει ως έξοδο την τιμή που αφήνει η συνάρτηση ενεργοποίησης του νευρώνα (ή των νευρώνων) του.

Επίσης θα πρέπει να σημειωθεί ότι η αρχιτεκτονική που επιλέγεται, δηλαδή ο αριθμός νευρώνων ανά επίπεδο, αριθμός hidden layers κ.τ.λ., είναι βασισμένη πάντα στο πρόβλημα που προσπαθούμε να επιλύσουμε. Δυστυχώς, δεν υπάρχει ένας αλγόριθμος

που να μας προσδιορίζει την αρχιτεκτονική που θα επιλέξουμε, έχοντας ως συμπέρασμα την εμπειρική επιλογή της. Τέλος, δεν είναι απαραίτητο, αλλά για την ευκολότερη εκπαίδευση του δικτύου, προτιμούμε να κανονικοποιήσουμε τα δεδομένα εισόδου στο εύρος το τιμών [0, 1].

Η λέξη εμπρόσθιο προσδίδει τη μονόπλευρη κατεύθυνση των δεδομένων μέσα στο δίκτυο. Αυτό επαληθεύεται από την Εικόνα 1.11. αφού η αρχιτεκτονική του δικτύου μοιάζει έντονα με ένα κατευθυνόμενο γράφο. Αυτό που δε φαίνεται στο γράφο είναι η δράση του δικτύου μετά την εξαγωγή του τελικού αποτελέσματος.

1.3.4 Ανάστροφη μετάδοση λάθους

Με τα όσα αναλύθηκαν μέχρι στιγμής, όσες φορές και να παρουσιαστεί ένα δεδομένο εισόδου στο δίκτυο, θα λαμβάνουμε πάντα τις ίδιες τιμές εξόδου, αφού αυτό δεν εκπαιδεύεται.

Τα δίκτυα μηχανικής μάθησης πρέπει κατά κάποιο τρόπο να μαθαίνουν, κάνοντας χρήση των δεδομένων εκπαίδευσης, τα καλύτερα δυνατά βάρη, που θα οδηγήσουν στη γενίκευση των δεδομένων ελέγχου. Σε γενικές γραμμές, ένα δίκτυο υπολογίζει την τιμή εξόδου, για κάθε δεδομένο εισόδου, τέτοιο ώστε να αποτιμάται σε:

$$Y^p = F(Z^p, W)$$

Εξίσωση 1.3: Γενικευμένη εξίσωση εξόδου του δικτύου.

όπου Z^p είναι το p-οστό δεδομένο εισόδου (input pattern) και W οι προσαρμόσιμες παράμετροι του δικτύου. Η συνάρτηση λάθους, που παρουσιάζει την απόκλιση από το επιθυμητό αποτέλεσμα ενός input pattern είναι η ακόλουθη:

$$E^p = D\big(G^p, F(Z^p, W)\big)$$

Εξίσωση 1.4: Γενικευμένη εξίσωση σφάλματος για ένα δεδομένο εισόδου.

όπου G^p είναι το επιθυμητό αποτέλεσμα. Ψάχνουμε, λοιπόν, συνάρτηση Y^p και τιμές παραμέτρων W, τέτοιες ώστε το συνολικό σφάλμα του δικτύου

$$E_{test} = \{(Z^1, G^1), (Z^2, G^2), \dots, (Z^k, G^k)\}$$

Εξίσωση 1.5: Σύνολο σφάλματος δικτύου για όλα τα δεδομένα εισόδου.

στο σύνολο εισόδου Z και G, να είναι όσο το δυνατόν μικρότερο.

Ανάστροφη μετάδοση λάθους στα MLP

Σε ένα δίκτυο MLP, το σφάλμα για κάθε δεδομένο εισόδου δίνεται από την εξίσωση:

$$E^p = \frac{1}{2}\sum_j (t_{pj} - o_{pj})^2$$

Εξίσωση 1.6: Mean Square Error

Η προσαρμογή των βαρών βασίζεται στο γενικό κανόνα δέλτα, τη μέθοδο κατάβασης κλίσης, τη μέθοδο τετραγωνικού σφάλματος και τη συνάρτηση ενεργοποίησης [19].

Παραγωγίζοντας το αρνητικό της συναρτήσεως του σφάλματος ως προς τα βάρη - μέθοδος κατάβασης κλίσης [19] – φτάνουμε στις ακόλουθες εξισώσεις για αναβάθμιση των βαρών σε κάθε νευρώνα.

$$\delta_{pj} = o_{pj}(1 - o_{pj})(t_{pj} - o_{pj})$$

Εξίσωση 1.7: αναβάθμιση βαρών του νευρώνα εξόδου, όπου o_{pj} η πραγματική έξοδος του νευρώνα και t_{pj} η επιθυμητή.

$$\delta_{pj} = o_{pj}(1 - o_{pj})\sum \delta_{pк} W_{jк}$$

Εξίσωση 1.8: αναβάθμιση βαρών του νευρώνα κρυφού επιπέδου, όπου o_{pj} η πραγματική έξοδος του νευρώνα, $\delta_{pк}$ το σφάλμα του νευρώνα κ, που είναι συνδεδεμένος με την έξοδο j και $W_{jк}$ τα ανάλογα βάρη.

1.3.5 Αλγόριθμος μάθησης

"Επανάληψη μήτηρ πάσης μαθήσεως". Το δίκτυο θα πρέπει να εκπαιδευτεί για έναν αριθμό επαναλήψεων, κάνοντας χρήση των ίδιων δεδομένων εισόδου προκειμένου να καταφέρει καλύτερη γενίκευση στα δεδομένα ελέγχου.

Συνοψίζοντας τα παραπάνω μπορούμε να ορίσουμε τον αλγόριθμο μάθησης ενός νευρωνικού δικτύου ως τον ακόλουθο:

1. Αρχικοποίηση των βαρών και του κατωφλίου κάθε νευρώνα με μικρές τυχαίες τιμές.
2. Τροφοδότηση τιμών εισόδου στο επίπεδο εισόδου του δικτύου
3. Υπολογισμός τιμών εξόδου σε κάθε νευρώνα και ακολούθως στους νευρώνες εξόδου
4. Υπολογισμός της μεταβολής των βαρών των νευρώνων εξόδου – εξίσωση 1.7
5. Ανάστροφη μετάδοση λάθους – αλλαγή βαρών νευρώνων κρυφού επιπέδου – εξίσωση 1.8
6. Επανάληψη από βήμα 2 μέχρι να φτάσει το σφάλμα σε ικανοποιητικό επίπεδο.

1.4Σχετική έρευνα νευρωνικών δικτύων

1.4.1 Deep Neural Networks

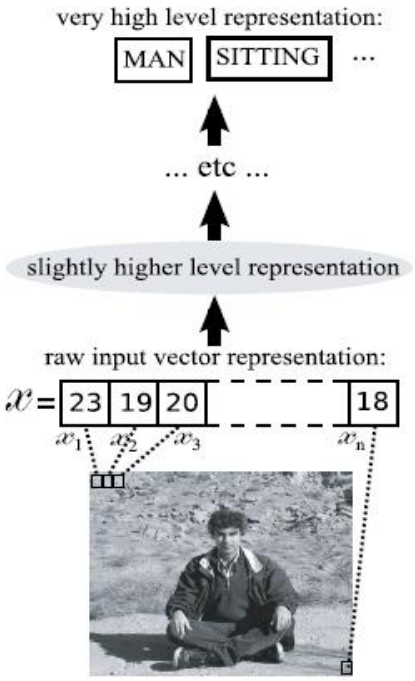
Ο Minsky (1963) ορίζει το θεμελιώδες Credit Assignment Problem (πρόβλημα κατανομής αμοιβής) ως εξής: Ποια προσαρμόσιμα συστατικά ενός συστήματος μηχανικής μάθησης είναι υπεύθυνα για την επιτυχία ή αποτυχία τους; Ποιες αλλαγές βελτιστοποιούν την απόδοσή τους; [24].

Σε ένα συνηθισμένο νευρωνικό δίκτυο οι νευρώνες συμβάλουν στην εξαγωγή τιμών που αντιπροσωπεύουν την έξοδο του δικτύου για ένα δεδομένο εισόδου. Credit assignment είναι το πρόβλημα που αφορά την εύρεση των καλύτερων τιμών στα βάρη, προκειμένου να πετύχουμε τα επιθυμητά αποτελέσματα. Ανάλογα πάντα με το πρόβλημα, το μήκος αυτής της υπολογιστικής "αλυσίδας" μπορεί να διαφέρει κατά πολύ. Το μικρότερο μονοπάτι Credit Assignment που δίνει λύση σε κάποιο πρόβλημα, ορίζει και το βάθος του προβλήματος [16]. *Deep Learning* είναι η ορθή ανάθεση των credits (προσαρμόσιμων μεταβλητών) σε όλο το μήκος της αλυσίδας. Την ανάθεση αναλαμβάνουν αλγόριθμοι που επιχειρούν την άντληση πληροφορίας από δεδομένα με ψηλά επίπεδα αφαιρετικότητας, συνθέτοντας πολύπλοκα μοντέλα αποτελούμενα από πολλαπλούς μη-γραμμικούς μετασχηματιστές [16]. Το βάθος της αρχιτεκτονικής υπόρρητα δηλώνει τον αριθμό των μη-γραμμικών συστατικών στοιχείων του συστήματος. Για παράδειγμα σε shallow αρχιτεκτονικές (όπως MLP) χρησιμοποιούνται 1, 2 ή 3 ενεργά επίπεδα.

1.4.2 Αρχιτεκτονική

Στόχος της βαθιάς μηχανικής μάθησης (Deep Learning) είναι η εύρεση υψηλών-επιπέδου (high-level) χαρακτηριστικών, βασισμένα στην πληροφορία που θα λάβουν, από τη σύνθεση πολλών χαμηλού-επιπέδου χαρακτηριστικών. Εκπαιδεύοντας, ταυτόχρονα, πολλά επίπεδα επιτρέπεται στο δίκτυο να κάνει αντιστοίχιση των δεδομένων εισόδου με το επιθυμητό αποτέλεσμα, χωρίς να χρειαστεί η ρητή βοήθεια από τον προγραμματιστή [25].

Ο ανθρώπινος εγκέφαλος, κατά την ακοή και όραση, αποικοδομεί και επεξεργάζεται την εισερχόμενη πληροφορία σε πολλαπλά επίπεδα, το καθένα σε διαφορετικό φλοιό [26]. Τις τελευταίες δεκαετίες έχουν γίνει μεγάλες προσπάθειες εκπαίδευσης δικτύων όμοιες με τον τρόπο λειτουργίας του εγκεφάλου των θηλαστικών [25], χωρίς ιδιαίτερα θετικά αποτελέσματα (εκτός από τα CNNs), μέχρι το 2006 όπου δημοσιεύτηκε από τον Hinton [27] η πετυχημένη "βαθιά" εκπαίδευση δημιουργώντας το δίκτυο RBM – Restricted Baltzmann Machine. Ο άπληστος αυτός αλγόριθμος εκπαιδεύει το κάθε επίπεδο ξεχωριστά, χρησιμοποιώντας μη επιβλεπόμενη μάθηση για κάθε επίπεδο. Την ίδια φιλοσοφία υιοθέτησαν μετά από μικρό χρονικό διάστημα οι auto-encoders αλγόριθμοι



Εικόνα 1.12 [25]: Προσπαθούμε να αναγάγουμε τα χαμηλού επιπέδου δεδομένα, από απλούς αριθμούς σε ακμές, αντικείμενα και τέλος σε γλωσσολογικό επίπεδο.

[26]. Παράλληλα υλοποιήθηκαν αλγόριθμοι που καταφέρνουν να λύσουν εξίσου καλά προβλήματα που έδωσαν απαντήσεις οι παραπάνω αλγόριθμοι, χρησιμοποιώντας επιβλεπόμενη μάθηση.

Παρόμοια εξέλιξη είχαν τα ConvNets (CNNs), αρκετά χρόνια πριν, όπου το δίκτυο του Fukushima (1980) που αφορούσε το πρόβλημα Neocognition [28] αρχικά εκπαιδεύτηκε κάνοντας χρήση μη επιβλεπόμενης μάθησης και 9 χρόνια αργότερα, το 1989, ο Yann LeCun [1] έδωσε λύση χρησιμοποιώντας επιβλεπόμενη μάθηση. Περισσότερα για την αρχιτεκτονική των ConvNets, στο κεφάλαιο 3.

Τι μας οδηγεί στα ConvNets;

Δύο από τα σημαντικότερα προβλήματα που έτυχαν αντικατάσταση της "χειροποίητης" δημιουργίας πυρήνων εύρεσης χαρακτηριστικών (feature detectors), όπως για παράδειγμα τα φίλτρα Gabor, ήταν η αναγνώριση αντικειμένων στις εικόνες (image recognition) και η επεξεργασία φυσικού λόγου (NLP – Natural Language Processing), όπου ουσιαστικά γίνεται προσπάθεια εξεύρεσης χαρακτηριστικών σε γλωσσολογικό επίπεδο. Το γεγονός ότι προσεγγίζοντας τα δύο προβλήματα με deep networks, συγκεκριμένα CNN, έδωσαν πολύ καλά αποτελέσματα, μας οδήγησε στην προσπάθεια να τα μελετήσουμε στο πρόβλημα της πρόβλεψης της δευτεροταγούς δομής των πρωτεϊνών.

Τα θετικά χαρακτηριστικά των δικτύων αυτών υπήρξαν αφορμή για "εκμετάλλευσή" τους μέσα από μια σειρά εφαρμογών που αναπτύχθηκαν. Οι κυριότερες εφαρμογές αφορούσαν την εξαγωγή χαρακτηριστικών μέσα από τα δεδομένα εισόδου. Σαφώς, η πρώτη εφαρμογή που θα μπορούσε να εφαρμοστεί το χαρακτηριστικό αυτό είναι η οπτική αναγνώριση αντικειμένων μέσα από φωτογραφίες καθώς επίσης και κατά τη διάρκεια κίνησης (π.χ. ρομπότ). Εφαρμογές που δεν είναι εμφανής ο λόγος χρήσης των ConvNets είναι το NLP πρόβλημα καθώς επίσης και τα Recommendation Systems [41][42] (αλγόριθμοι που προτείνουν προϊόντα στους πελάτες). Η αξιοποίηση των δικτύων για τα προβλήματα αυτά, αφορούσε την εύρεση μη-εμφανών χαρακτηριστικών που βρίσκονται σε αυτά. Για παράδειγμα, στο NLP πρόβλημα οι σχέσεις των λέξεων αποτελούν ένα εκ των χαρακτηριστικών που οφείλει το δίκτυο να εντοπίσει. Τέλος, η αξιοποίησή τους από την εταιρία Spotify προκειμένου να λύσει το πρόβλημα του αραιού

πίνακα γειτνίασης του Collaborative filtering algorithm [43] αφορά την εύρεση κοινών χαρακτηριστικών μεταξύ χρηστών προκειμένου να προτείνει σε αυτούς μη δημοφιλοί ακούσματα.

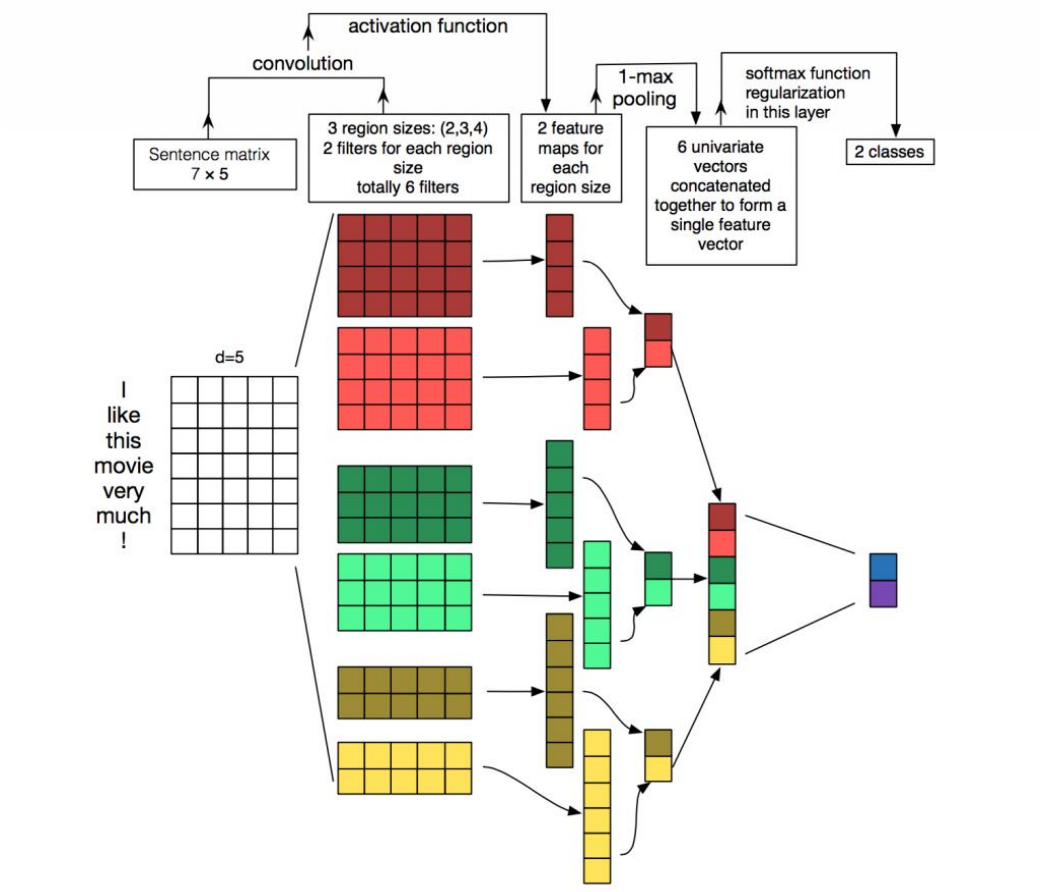
Πιστεύεται ότι όσο το δυνατό μεγαλύτερο απόσπασμα από το σύνολο δεδομένων παρουσιαστεί στο δίκτυο, μια χρονική στιγμή, τόσο το δυνατό καλύτερα ποσοστά πρόβλεψης θα δώσει. Η άποψη βασίζεται στο γεγονός ότι η δευτεροταγής δομή επηρεάζεται τόσο από κοντινές όσο από μακρινές αλληλεπιδράσεις των αμινοξέων μεταξύ τους. Για παράδειγμα, μπορεί μια β-πτυχωτή να σταθεροποιείται, με δεσμούς υδρογόνου, με μια άλλη β-πτυχωτή που να βρίσκεται αρκετά μακριά. Προσεγγίσεις όπως BRNN είναι σε θέση να αντιληφθούν χωρική συσχέτιση των αμινοξέων, αλλά σε πολύ πιο μικρό επίπεδο, από ότι θα μπορούσε να αντιληφθεί ένα deep δίκτυο. Μοντέλα που ακολουθούν deep αρχιτεκτονική, έχει αποδειχτεί ότι καταφέρνουν να δημιουργήσουν αυτόματα μια αναπαράσταση (representation) [27][35], που μερικές φορές ούτε ο ανθρώπινος νους δεν μπορεί να την εξηγήσει, για να δώσουν απαντήσεις στα προβλήματα.

1.4.3 Σχετική έρευνα βασισμένη στο NLP

Η επεξεργασία του φυσικού λόγου είναι ένας τομές της πληροφορικής, τεχνητής νοημοσύνης και της υπολογιστικής γλωσσολογίας που ασχολείται με την αλληλεπίδραση της φυσικής γλώσσας με τον ηλεκτρονικό υπολογιστή. Μερικά από τα ζητήματα που αφορούν τον τομέα αυτό είναι η χρήση των Η/Υ για κατανόηση της φυσικής μας γλώσσας καθώς επίσης και η ορθά συντακτικά και σημασιολογικά παραγωγή φυσικής γλώσσας.

CNN και NLP

(Σημείωση: Για την κατανόηση του υποκεφαλαίου, καλύτερα να ανατρέξετε πρώτα στο κεφάλαιο 3, όπου και εξηγούνται τα CNNs.)



Εικόνα 1.13 [38]: Αρχιτεκτονική δικτύου CNN για NLP το πρόβλημα.

Τα δεδομένα εισόδου αποτελούνται από προτάσεις. Η κάθε λέξη κωδικοποιώταν, αρχικά, με την one hot vector [20] τεχνική· τεχνική που εγκαταλείφθηκε όταν έρευνες, από την Google, έδειξαν ότι η χρήση του "shallow" νευρωνικού δικτύου word2vec [30] μπορούσε να επιφέρει καλύτερα αποτελέσματα, δημιουργώντας ξεχωριστή κωδικοποίηση λέξεων

προτού εισαχθούν ως δεδομένο εισόδου στο σύστημα [31]. Στο σημείο αυτό είναι σημαντικό να γίνει μια γρήγορη αναφορά στον τρόπο λειτουργίας του word2vec, αφού μπορεί να βοηθήσει την έρευνα του Π.Κ. σε μεταγενέστερο στάδιο.

Word2Vec

> *"You shall know a word by the company it keeps"*
>
> Firth, J.R. 1957:11

Επί της ουσίας, το *word2vec* προσπαθεί να αναγνωρίσει τη συντακτική και σημασιολογική σχέση μιας δοθείσας λέξης, συγκρίνοντάς την με τις παραπλήσιες της.

government debt problems turning into **banking** crises as has happened in
saying that Europe needs unified **banking** regulation to replace the hodgepodge

Εικόνα 1.14: Η λέξη "banking" αντιπροσωπεύεται από τις λέξεις δεξιά και αριστερά της.

Για παράδειγμα εάν έχουμε τις προτάσεις: *I like deep learning*, *I like NLP* και *I enjoy flying*, ο πίνακας που γειτνίασης των λέξεων είναι ο ακόλουθος.

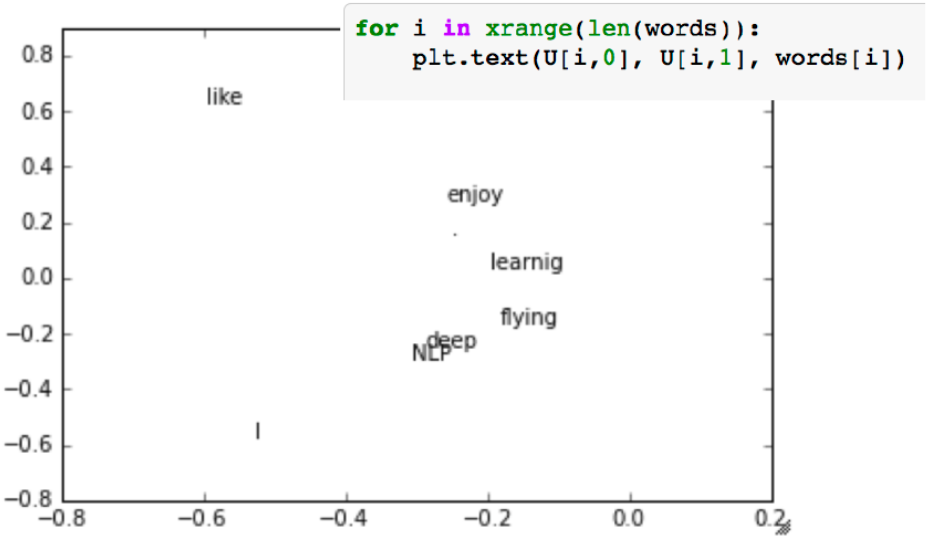
counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

Πίνακας 1.4 [37]: Πίνακας γειτνίασης.

Εύκολα μπορεί κάποιος να παρατηρήσει ότι όσο το κείμενο, που δίνεται ως είσοδος στο word2vec, μεγαλώνει τόσο πιο αραιός (sparse) γίνεται ο πίνακας γειτνίασης. Οδηγούμαστε, στο βήμα "κλειδί" του αλγορίθμου, όπου με στόχο τη μείωση των

διαστάσεων (dimensionality reduction) γίνεται χρήση ενός πολύ γνωστού αλγορίθμου στο χώρο της εξόρυξης δεδομένων, του SVD (Singular value decomposition) [32].

Ο SVD αλγόριθμος επιτρέπει την αναπαράσταση οποιουδήποτε δυσδιάστατου πίνακα με συγκεκριμένο τρόπο (Εικόνα 1.15), μειώνοντας την αναπαράσταση των λιγότερο σημαντικών πληροφοριών που τηρούνται στο πίνακα. Το σημαντικότερο σημείο του αλγορίθμου είναι ο ορισμός των "concepts" όπου ουσιαστικά προσδιορίζουν τη σχέση των γραμμών με των στηλών. Για παράδειγμα, στο NLP πρόβλημα, μπορεί η σχέση δύο λέξεων να είναι η συντακτική σχέση υποκειμένου και ρήματος, ενώ στο PSSP πρόβλημα οι πιθανές αλληλεπιδράσεις ανάμεσα στα αμινοξέα (π.χ. ηλεκτροστατικές αλληλεπιδράσεις, όπως δυνάμεις Van de Waals) [9].



Εικόνα 1.15 [37]: Οπτικοποίηση του αποτελέσματος του SVD αλγορίθμου.

Τέλος, η αναπαράσταση της λέξης (Εικόνα 1.16), η είσοδος δηλαδή του CNN δικτύου, είναι ένα διάνυσμα με τιμές, όσα και "concepts".

$$linguistics = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Εικόνα 1.16: Το διάνυσμα τιμών για τη λέξη "linguistics".

Επιστρέφοντας στο CNN δίκτυο, αναλογιζόμαστε τον τρόπο λειτουργίας τους στο πρόβλημα αναγνώρισης της εικόνας και διαπιστώνουμε ότι δεν θα γίνεται τοπική σάρωση (local receptive fields) αλλά το φίλτρο θα έχει σταθερό μέγεθος στο πλάτος (διάσταση του διανύσματος αναπαράστασης) και μεταβλητό στο μήκος [34].

Εφαρμόζοντας το NLP στο PSSP πρόβλημα

Όπως αναλύθηκε στο Υποκεφάλαιο 1.1, οι πολυπεπτιδικές αλυσίδες αποτελούνται από μια σειρά αμινοξέων. Όπως με το DNA και το RNA έτσι και για τις πρωτεΐνες ένας τρόπος αναπαράστασής τους είναι η χρήση γραμμάτων του λατινικού αλφαβήτου, αντιπροσωπεύοντας το καθένα από αυτά ένα δομικό υλικό (στην περίπτωση των πρωτεϊνών, ένα αμινοξύ). Ο παραλληλισμός του "κειμένου" της αλληλουχίας των αμινοξέων με το NLP, έχει ως στόχο την εκμετάλλευση των τριών κύριων χαρακτηριστικών που έχουν αναπτυχθεί για το NLP πρόβλημα: ανάκτηση πληροφοριών (από μη δομημένα ή ημιδομημένα δεδομένα), τη σημασιολογική μετάφραση και την εξαγωγή πληροφοριών [17].

Η παρόμοια αντιμετώπιση του PSSP προβλήματος με το NLP (Υποκεφάλαιο 1.2) έχει οδηγήσει στη χρήση της απλούστερης μορφής της πρωτοταγούς δομής· την αλληλουχία χαρακτήρων. Η μορφή αυτή είναι ιδιαίτερα χρήσιμη όταν ενδιαφερόμαστε να σπάσουμε την αλληλουχία σε n-grams, δημιουργώντας "λέξεις" για εισαγωγή τους στο word2vec. Η n-gram τεχνική αφορά την κατάτμηση ενός κειμένου σε n-άδες μεγέθους n:

Field	Unit	Sample sequence	1-gram sequence	2-gram sequence	3-gram sequence
Protein sequencing	amino acid	… Cys-Gly-Leu-Ser-Trp …	…, Cys, Gly, Leu, Ser, Trp, …	…, Cys-Gly, Gly-Leu, Leu-Ser, Ser-Trp, …	…, Cys-Gly-Leu, Gly-Leu-Ser, Leu-Ser-Trp, …
DNA sequencing	base pair	…AGCTTCGA…	…, A, G, C, T, T, C, G, A, …	…, AG, GC, CT, TT, TC, CG, GA, …	…, AGC, GCT, CTT, TTC, TCG, CGA, …
Computational linguistics	character	…to_be_or_not_to_be…	…, t, o, _, b, e, _, o, r, _, n, o, t, _, t, o, _, b, e, …	…, to, o_, _b, be, e_, _o, or, r_, _n, no, ot, t_, _t, to, o_, _b, be, …	…, to_, o_b, _be, be_, e_o, _or, or_, r_n, _no, not, ot_, t_t, _to, to_, o_b, _be, …
Computational linguistics	word	… to be or not to be …	…, to, be, or, not, to, be, …	…, to be, be or, or not, not to, to be, …	…, to be or, be or not, or not to, not to be, …

Πίνακας 1.2: Παραδείγματα n-gram (Wikipedia)

Βασισμένη σε όλα τα παραπάνω η έρευνα των Asgari και Mofrad (2015) [33] αφορούσε τη δημιουργία πρωτεϊνικών διανυσμάτων (protein vectors ή ProtVec) κάνοντας χρήση του συνόλου δεδομένων SwissProt. Το δίκτυο εκπαιδεύτηκε κάνοντας χρήση 546,790 ακολουθιών του συνόλου δεδομένων, σχηματίζοντας 7,027 πρωτεϊνικές οικογένειες και κατά τον έλεγχο των 324,018 πρωτεϊνικών ακολουθιών επιτεύχθηκε το ποσοστό της τάξεως του 94%.

Σύνοψη

Είναι σημαντικό να γίνει κατανοητό ότι ίσως το κυριότερο συστατικό ενός επιτυχημένου τεχνητού νευρωνικού δικτύου είναι η επιλογή ενός καλού συνόλου δεδομένων (dataset). Η σύγκριση ποσοστών αποτελεσμάτων μεταξύ δικτύων, που αναπτύχθηκαν, κάνοντας χρήση διαφορετικών datasets, έχει λογική στην περίπτωση που ο μόνος παράγοντας που διαφοροποιεί τα αποτελέσματα είναι τα δεδομένα εισόδου, ενώ η αρχιτεκτονική να παραμένει άθικτη. Σε οποιαδήποτε άλλη περίπτωση, η εξαγωγή συμπερασμάτων αποσκοπεί στο να δώσει κατευθυντήριες γραμμές για την πορεία της έρευνας, αφού η σύγκριση μπορεί να θεωρηθεί άτοπη.

Επηρεαζόμενοι από τα παραπάνω, έχουμε πλέον την υποψία για την κατεύθυνση που μπορεί να πάρει η διπλωματική εργασία, αξιοποιώντας το σύνολο δεδομένων που έχει το Π.Κ στη διάθεσή του (Υποκεφάλαιο 2.2).

Κεφάλαιο 2

Σχεδίαση και Υλοποίηση

2.1 Απαιτήσεις και σχεδιασμός διαδικτυακής εφαρμογής

- 2.1.1 Σκοπός
- 2.1.2 Product Perspective
  - 2.1.2.1 User Interfaces
  - 2.1.2.2 Hardware Interfaces
  - 2.1.2.3 Software Interfaces
  - 2.1.2.4 Memory Constraints
  - 2.1.2.5 Product Functions
- 2.1.3 Constraints
- 2.1.4 Performance Requirements
- 2.1.5 Maintainability
- 2.1.6 Logical Database Requirement

2.2 Υλοποίηση

- 2.2.1 Χρήση πακέτου ανοικτού κώδικα XAMMP
- 2.2.2 Χρήση γλώσσας προγραμματισμού PHP
- 2.2.3 Χρήση phpMyAdmin – MySQL
- 2.2.4 Δημιουργία διεπαφών διαδικτυακής εφαρμογής
- 2.2.5 Back End

2.3 Δεδομένα εισόδου

- 2.3.1 Δεδομένα εισόδου και νευρωνικά δίκτυα
- 2.3.2 MSA profiles
- 2.3.3 Χρησιμοποιώντας τα MSA PROFILES για image recognition
- 2.3.4 Δημιουργία του συνόλου δεδομένων

2.1 Απαιτήσεις και Σχεδιασμός Διαδικτυακής εφαρμογής

2.1.1 Σκοπός

Τα τελευταία χρόνια έχουν μελετηθεί και υλοποιηθεί αρκετοί αλγόριθμοι, όπως και τεχνικές βελτίωσής τους, γεγονός που δημιούργησε την ανάγκη για δημοσιοποίηση τους. Επομένως, σκοπός της εφαρμογής είναι η δημοσιοποίηση της μέχρι στιγμής έρευνας του Π.Κ. στο πρόβλημα πρόβλεψης της δευτεροταγούς δομής των πρωτεϊνών ενώ παράλληλα να δημιουργηθεί ένα διαδικτυακό εργαλείο όπου ο χρήστης να έχει τη δυνατότητα να το χρησιμοποιεί για εξαγωγή των δικών του αποτελεσμάτων, χωρίς αυτός να πρέπει να δημιουργήσει το δικό του δίκτυο. Το σύστημα μας απευθύνεται σε όλους τους ερευνητές που ασχολούνται με το PSSP πρόβλημα, ενώ σε μελλοντικό επίπεδο, τα ενδιαφερόμενα μέρη ενδέχεται να αυξηθούν.

Ακρωνύμια

ΒΔ: Βάση δεδομένων
Χρήστης: πιθανό μέλος του συστήματος
Σύστημα: Η προς ανάπτυξη εφαρμογή

2.1.2 Product Perspective

2.1.2.1 User Interfaces – Απαιτήσεις

Διασύνδεση εισόδου

Μέσω της διασύνδεσης εισόδου ο χρήστης επιχειρεί την εισαγωγή του, στο σύστημα μας. Θα παρουσιάζεται στο χρήστη μια φόρμα με δυο βασικές επιλογές. Η πρώτη επιλογή θα είναι το "Sign Up", το οποίο παραπέμπει το χρήστη στη διασύνδεση εγγραφής, όπου και θα κάνει εγγραφή για πρώτη φορά. Η δεύτερη επιλογή θα είναι το "Log In", μέσω του οποίου θα συνδέεται, ο χρήστης, στο σύστημα αφού δώσει τα στοιχεία του για επιβεβαίωση μέσω της δικής μας ΒΔ.

Διασύνδεση εγγραφής

Ο χρήστης θα κάνει την εγγραφή του στο σύστημα μέσω αυτής της διασύνδεσης. Θα

παραπέμπεται να συμπληρώσει τη φόρμα που παρουσιάζεται στην οθόνη για να γίνει καταγραφή των προσωπικών του δεδομένων στην ΒΔ.

Διασύνδεση εκτέλεσης εκπαιδευμένου δικτύου

Με τη διασύνδεση αυτή, δίνεται στο χρήστη η ευκαιρία να τρέξει ένα ήδη εκπαιδευμένο δίκτυο, κάνοντας χρήση του συνόλου δεδομένων που επιθυμεί. Αρχικά, ο χρήστης θα καλείται να επιλέξει το δίκτυο με το οποίο θέλει να τρέξει τα δεδομένα του και στη συνέχεια θα πρέπει να δώσει το ηλεκτρονικό του ταχυδρομείο ούτως ώστε να του σταλούν τα αποτελέσματα. Σε περίπτωση που δεν επιθυμεί να καταχωρήσει το ηλεκτρονικό του ταχυδρομείο, το σύστημα θα παράγει έναν μοναδικό κωδικό, βάσει του οποίου θα μπορέσει να κατεβάσει, μόνο μια φορά, τα αποτελέσματα του δικτύου στον προσωπικό του λογαριασμό, όταν αυτά είναι έτοιμα.

Διασύνδεση εκπαίδευσης δικτύου

Πέραν από την εκτέλεση ενός ήδη εκπαιδευμένου δικτύου, επιθυμούμε την παροχή της υπηρεσίας εκπαίδευσης δικτύου από το χρήστη, δικαίωμα που θα μπορούν να έχουν μόνο οι εγγεγραμμένοι χρήστες. Πιο συγκεκριμένα, θα πρέπει να παρέχεται στο χρήση η επιλογή του δικτύου που επιθυμεί να δημιουργήσει (π.χ. BRNN, Echo-State, CNN) και βάσει της επιλογής αυτή, να εμφανίζεται σε αυτόν μια φόρμα για να δηλώσει την αρχιτεκτονική των προσαρμόσιμων παραμέτρων (αριθμό κρυφών δικτύων, αριθμό νευρώνων, learning rate κ.τ.λ).

2.1.2.2 Hardware Interfaces

Απαραίτητη προϋπόθεση για τη λειτουργία του συστήματός μας, είναι η σύνδεση του χρήστη με το διαδίκτυο. Σε πρωταρχικό επίπεδο η διαδικτυακή εφαρμογή που θα σχεδιάζουμε πρέπει να τρέχει σε όλους τους φυλλομετρητές. Δεν έχει συμφωνηθεί λειτουργία της εφαρμογή σε κινητές πλατφόρμες, αλλά οι σχεδιαστικές αρχές της εφαρμογής θα πρέπει να επιτρέπουν μια τέτοια προσαρμογή.

2.1.2.3 Software Interfaces

Εφαρμογή για δημιουργία ιστοσελίδων:

Όνομα: IntelliJ IDEA 14

Documentation: https://www.jetbrains.com/idea/

Θεωρήσαμε το IntelliJ IDEA, ως το ιδανικότερο εργαλείο για την ανάπτυξη της διαδικτυακής μας εφαρμογής, λόγω της υποστήριξης όλων των προγραμματιστικών γλωσσών που χρειάζονται για την ανάπτυξη μιας διαδικτυακής εφαρμογής (HTML5, Javascript, jQuery, CSS, PHP – διερμηνέα).

Σύστημα διαχείρισης ΒΔ

Όνομα: Microsoft SQL Server

Documentation: http://msdn.microsoft.com/en-us/library/ms130214.aspx

Ως εργασία πανεπιστημίου, το σύστημα που θα αναπτύξουμε, δεν υποστηρίζεται από οικονομικούς πόρους. Επομένως οδηγούμαστε στην επιλογή του Microsoft SQL Server αφού παρέχεται στην ομάδα μας δωρεάν από το πανεπιστήμιο. Στην περίπτωση που ο πελάτης θελήσει να χρηματοδοτήσει το έργο μας, τότε θα πρέπει να γίνει υλοποίηση της βάση μας σε cloud.

2.1.2.4 Memory Constraints

Η μέγιστη μνήμη RAM που θα απαιτείται για τη λειτουργία της εφαρμογής σε ένα φυλλομετρητή είναι 128 Mb, αφού η ανάπτυξη βασίζεται σε lightweight – client [22].

Η ελάχιστη μνήμη RAM που χρειάζεται ο server είναι 16 Gb, καθώς θα επιτρέπεται να εκτελούνται πέντε δίκτυα παράλληλα.

2.1.2.5 Product Functions

Διαχείριση Δεδομένων

Η εφαρμογή μας, επιτρέπει στο χρήστη να δημιουργήσει τον δικό του λογαριασμό. Όταν ο χρήστης επιλέξει την δημιουργία λογαριασμού τότε θα του δίνεται η δυνατότητα να εισάγει όλα τα προσωπικά του στοιχεία προκειμένου να κάνει έναν αυτόνομο λογαριασμό, τα οποία αποθηκεύονται στην βάση δεδομένων.

Λειτουργία Εκτέλεσης Δικτύων

Η λειτουργία αυτή, τρέχει από τη στιγμή που ο χρήστης επιλέξει την εκτέλεση κάποιου δικτύου, μέχρι να παραδοθεί σε αυτόν το αποτέλεσμα. Λαμβάνει ως είσοδο την επιλογή του χρήστη για το δίκτυο που επιθυμεί να εκτελέσει καθώς επίσης το ηλεκτρονικό του ταχυδρομείο. Αναλαμβάνει την εκτέλεση του εκπαιδευμένου δικτύου στον εξυπηρετητή και ακολούθως αποστολή των αποτελεσμάτων. Τυγχάνει διακλάδωσης όταν ο χρήστης δεν δώσει το ηλεκτρονικό του ταχυδρομείο και δημιουργεί έναν μοναδικό MD5 κωδικό μέσω του οποίο θα γίνει ανάκτηση των δεδομένων.

Λειτουργία Ανάκτησης Αποτελεσμάτων

Χρησιμοποιώντας τη λειτουργία ανάκτησης αποτελεσμάτων, ο χρήστης παρέχει στο σύστημα τον κωδικό που του δόθηκε, εξετάζεται ακολούθως εάν το αρχείο αποτελεσμάτων έχει δημιουργηθεί. Εάν δεν υπάρχει, ζητείται από το χρήστη να επιστρέψει αργότερα ενώ σε αντίθετη περίπτωση το αρχείο κατεβαίνει στον προσωπικό υπολογιστή.

Λειτουργία Εκπαίδευσης Δικτύων

Μια λειτουργία που ο μεγάλος αριθμός παραμέτρων, που μπορεί να επηρεάσουν τη λειτουργία της, την θέτει ταυτόχρονα ως το πιο ευάλωτο χαρακτηριστικό του συστήματος. Θα δέχεται ως είσοδο μια σειρά από παραμέτρους που προσδιορίζουν την αρχιτεκτονική ενός δικτύου, το σύνολο δεδομένων του χρήστη και ακολούθως θα είναι υπεύθυνη για την ανάθεση της δημιουργίας του δικτύου από τον εξυπηρετητή.

2.1.3 Constraints

Το σύστημα σχεδιάστηκε χρησιμοποιώντας το αυξητικό μοντέλο [23], αφού γνωρίζουμε εξ’ αρχής όλες τις απαιτήσεις, αλλά λόγο του περιορισμένου χρόνου, προσπαθήσαμε να καλύψουμε όσο το δυνατό περισσότερες από τις απαιτήσεις που προαναφέραμε.

2.1.4 Performance Requirements

Θεωρείται απαραίτητο να είναι δυνατή η ταυτόχρονη πρόσβαση των χρηστών σε όλες τις

λειτουργιές του συστήματος. Αν οι αριθμός των διεργασιών που εμπλέκουν επικοινωνία με τη ΒΔ, την ίδια χρονική στιγμή, ενδέχεται να παρουσιαστεί μείωση στην απόδοση του συστήματος.

Το λογισμικό θα ανταποκρίνεται στις εντολές του χρήστη σε χρόνο λιγότερο των 5 δευτερολέπτων με μέγιστη καθυστέρηση 7 δευτερόλεπτα [22].

Τέλος, για τη ορθή διαχείριση των υπολογιστικών πόρων, δεν θα επιτρέπεται η εκτέλεση περισσότερων από πέντε διεργασιών, την ίδια χρονική στιγμή, στον διακομιστή.

2.1.5 Maintainability

Οι φάσεις που προηγούνται αποσκοπούν στην αποδοτική λειτουργία του συστήματος ούτως ώστε να μεγιστοποιήσουν τον κύκλο ζωής του. Παρόλα αυτά, η φάση της συντήρησης μπορεί να προκληθεί από πολλαπλούς παράγοντες, όπως:

Διόρθωση των bugs

Το μέγεθος του συστήματος δεν πρέπει να υποτιμηθεί. Για την αντιμετώπιση του κινδύνου αυτού, οι απαιτήσεις του συστήματος υλοποιούνται με σειρά προτεραιότητας της λειτουργικότητας τους.

Εμφάνιση νέων απαιτήσεων

Για να διασφαλίσουμε τη δια βίου εξέλιξη του συστήματος, θα πρέπει να είναι αρκετά ευπροσάρμοστο στην εμφάνιση νέων απαιτήσεων. Η εμφάνιση νέας απαίτησης μπορεί να επηρεάσει τόσο τις διεπαφές του χρήστη, όσο και τον λογικό σχεδιασμό του συστήματος.

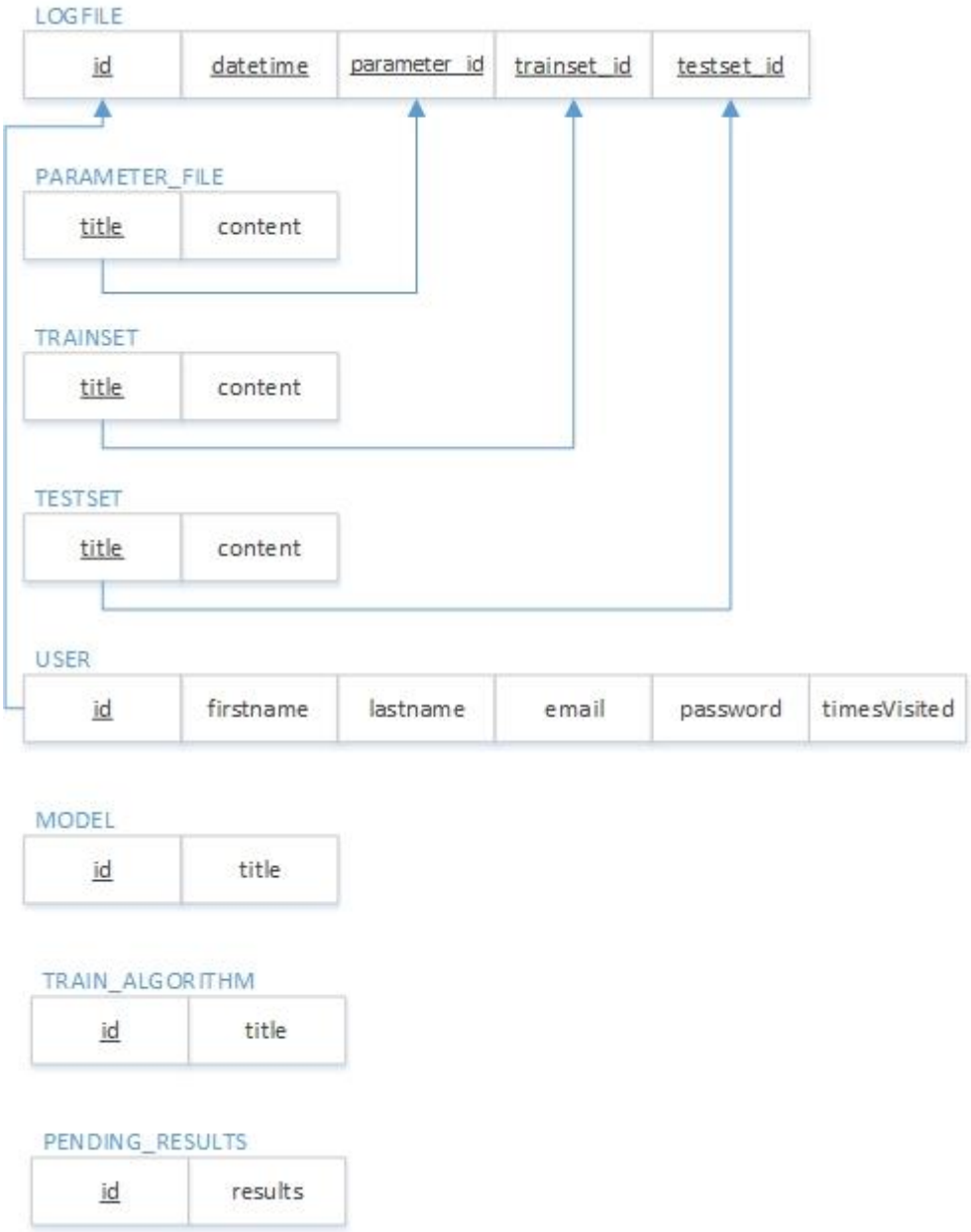
Απόδοση ΒΔ

Η εξελικτική πορεία της τεχνολογίας μπορεί να οδηγήσει στην ανάγκη για αλλαγή της ΒΔ, λόγω της ανικανότητας της τρέχουσας να εξυπηρετήσει τις αναδυόμενες ανάγκες.

Ασφάλεια

Η λειτουργία εκπαίδευσης δικτύων εγείρει νέα ζητήματα ασφάλειας και αξιοπιστίας του συστήματος, τομέας που χρίζει ιδιαίτερης προσοχής.

2.1.6 Logical Database Requirements



2.2 Υλοποίηση

2.2.1 Χρήση πακέτου ανοικτού κώδικα XAMMP

Η διαδικτυακή εφαρμογή που κληθήκαμε να υλοποιήσουμε προϋποθέτει τη δημιουργία βάσης δεδομένων για την αποθήκευση δεδομένων απαραίτητων για τη φόρτωση της ιστοσελίδας και τα δεδομένα που ο χρήστης καταχωρεί στην εφαρμογή (στοιχεία εγγραφής του χρήστη, αποτελέσματα των δικτύων που τρέχει).

Χρησιμοποιήσαμε το πακέτο ανοικτού κώδικα XAMMP το οποίο μας προφέρει την δυνατότητα δημιουργίας Apache server ως localhost στον προσωπικό μας υπολογιστή.

2.2.2 Χρήση γλώσσας προγραμματισμού PHP

Το πακέτο ανοικτού κώδικα XAMPP μας παρέχει τη δυνατότητα γραφής στην γλώσσας PHP. Η PHP είναι γλώσσα προγραμματισμού που χρησιμοποιείται για την δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών με δυναμικό περιεχόμενο. Ουσιαστικά μας βοήθησε στην επικοινωνία μεταξύ της διεπαφής και της βάσης δεδομένων. Οι ακόλουθες λειτουργίες που υλοποιήθηκαν στη διαδικτυακή εφαρμογή εκτελούνται με κώδικα γραμμένο σε αυτή την γλώσσα:

2.2.2.1 Φόρτωση της διαδικτυακής εφαρμογής στο φυλλομετρητή του χρήστη:

Η διαδικτυακή εφαρμογή είναι αποθηκευμένη σε ένα διακομιστή (server). Όταν ο επισκέπτης αιτηθεί την ιστοσελίδα δηλαδή την καλέσει με την απαραίτητη διεύθυνση – URL - αυτή περνά από κάποια επεξεργασία. Αρχικά από τον PHP κώδικα παράγεται το περιεχόμενό της σε κείμενο HTML και ακολούθως στέλνεται σε πραγματικό χρόνο στο πρόγραμμα περιήγησης του χρήστη (user’s browser).

2.2.2.2 Εγγραφή του χρήστη στην διαδικτυακή εφαρμογή:

Ο χρήστης είναι απαραίτητο να καταχωρίσει κάποια προσωπικά του στοιχεία στο σύστημα για να μπορεί να εγγραφεί. Όταν η διαδικασία αυτή ολοκληρωθεί, στα στοιχεία που καταχωρήθηκαν ανατίθενται κάποιες μεταβλητές οι οποίες στέλνονται με την μέθοδο

POST στον διακομιστή. Ακολούθως με τον κώδικα PHP δημιουργούνται ερωτήματα (queries) σε μορφή MySql για να καταχωρηθούν τα δεδομένα στην Βάση Δεδομένων.

2.2.2.3 Σύνδεση χρήστη στο σύστημα (Log In)

Ο χρήστης για να ενωθεί στην διαδικτυακή εφαρμογή θα πρέπει να καταχωρήσει το χαρακτηριστικό όνομα χρήστη (Username) και τον κωδικό πρόσβασης (Password). Αρχικά όπως και για τα στοιχεία εγγραφής ο κωδικός πρόσβασης και το χαρακτηριστικό όνομα του χρήστη μετατρέπονται σε μεταβλητές και με την μέθοδο Post μεταφέρονται στον διακομιστή. Ακολούθως δημιουργείται με την γλώσσα PHP ένα ερώτημα σε μορφή MySql το οποίο αναζητεί τον χαρακτηριστικό όνομα χρήστη και σε περίπτωση που περιέχεται στην βάση ελέγχει τον κωδικό. Η απάντηση αποστέλνεται επίσης με κώδικα PHP.

2.2.2.4 Δοκιμή εκτέλεσης Δικτύου

Όταν ο χρήστης επιλέξει να τρέξει ένα δίκτυο με κάποια συγκεκριμένη είσοδο - πρωτεϊνική ακολουθία - που ο ίδιος καταχωρεί, τότε όπως και στις προηγούμενες δύο περιπτώσεις επικοινωνούμε με την βάση με PHP. Έτσι θα αναγνωριστεί πιο δίκτυο επιθυμεί ο χρήστης να τρέξει και θα στείλουμε την πρωτεϊνική ακολουθία στην είσοδο του δικτύου για να αυτού.

2.2.3 Χρήση phpMyAdmin- MySQL

Το πακέτου ανοικτού κώδικα XAMPP επίσης μας παρέχει ένα περιβάλλον το οποίο ονομάζεται phpMyAdmin. Με την βοήθεια του περιβάλλοντος αυτού μπορούμε να διαχειριστούμε με ευκολία τις MySQL βάσεις δεδομένων. Χρησιμοποιήσαμε το περιβάλλον αυτό για να δημιουργήσουμε τη Βάση Δεδομένων για τη διαδικτυακή εφαρμογή.

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων και τη χρησιμοποιήσαμε για την εύρεση, την εισαγωγή και την διαγραφή δεδομένων από και προς την βάση. Η κάθε μια λειτουργία μεταφράστηκε σε ερωτήματα (queries) γραμμένα σε MySQL τα οποία εκτελούνται με την βοήθεια της PHP .

2.2.3.1 Σύνδεση στην Βάση δεδομένων:

Για να συνδεθούμε με την βάση δεδομένων θα πρέπει να ορίσουμε κάποιες μεταβλητές που αντιπροσωπεύουν αυτή την βάση. Οι μεταβλητές που πρέπει να οριστούν είναι το servername στην δική μας περίπτωση είναι localhost εφόσον μετατρέψαμε τον προσωπικό μας υπολογιστή σε διακομιστή, το username και password για περιορισμένη πρόσβαση στην βάση και το dbname (data Base Name) το όνομα της βάση Δεδομένων. (Εικόνα 2.1)

```php
<?php

$servername = "xxxx";
$username = "xxxx";
$password = "xxxx";
$dbname = "xxxx";

try {
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);
    // set the PDO error mode to exception
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
}
catch(PDOException $e)
{
    echo "Connection failed: " . $e->getMessage();
}

```

Εικόνα 2.1: Κώδικας γραμμένος στην γλώσσα php που επιτρέπει την σύνδεση στην Βάση Δεδομένων. Βρίσκεται στο MySqlConnect.php.

2.2.3.2 Εισαγωγή δεδομένων στην βάση:

Για να εισάξουμε τα δεδομένα στην βάση προϋποθέτεται να συνδεθούμε στην Βάση Δεδομένων. Για την σύνδεση στην βάση χρησιμοποιούμε τον πιο πάνω κώδικα που βρίσκεται στο αρχείο MySqlConnect.php. Στην εικόνα 2.2 φαίνεται ο κώδικας που εισάγει μια γραμμή στον πίνακα model όπου στα πεδία ‘id’και ‘title’ καταχωρούνται αντίστοιχα το ‘3’ και το ‘Bidirectional Recurrent Neural Network’.

```php
require "MySqlConnect.php";
$Model = $conn->prepare("INSERT INTO `model` (`id`, `title`)
                    VALUES ('3', 'Bidirectional Recurrent Neural Network')");
$Model->execute();

$ResultModel = $Model->fetchAll(PDO::FETCH_ASSOC);

```

Εικόνα 2.2: Εισαγωγή μιας γραμμής στον πίνακα model στην βάση δεδομένων.

2.2.3.3 Εύρεση δεδομένων από την βάση:

Για την εύρεση δεδομένων από την βάση θα πρέπει πρώτα να συνδεθούμε σε αυτή χρησιμοποιόντας τον κώδικα στην σελίδα MySqlConnect.php. Στην εικόνα 2.3 φαίνεται ο κώδικας που αποθηκεύει στη μεταβλητή $ResultModel τα αποτελέσματα από την εύρεση όλων των δεδομένων από το πεδίο title από τον πίνακα model.

```
 8    require "MySqlConnect.php";
 9    $Model = $conn->prepare("SELECT title FROM model");
10    $Model->execute();
11
12    $ResultModel = $Model->fetchAll(PDO::FETCH_ASSOC);
13
```

Εικόνα 2.3: Εύρεση όλων των δεδομένων από το πεδίου title από τον πίνακα model.

2.2.4 Διεπαφές Διαδικτυακής Εφαρμογής (Front - End)

2.2.4.1 Εργαλεία και γλώσσες που χρησιμοποιήθηκαν για την διαμόρφωση των διεπαφών

Για τη δημιουργία των διεπαφών χρησιμοποιήσαμε ένα αποκριτικό πλαίσιο διεπαφής από την ιστοσελίδα http://html5up.net/. Επίσης για την διαμόρφωση των διεπαφών στιλιστικά, λειτουργικά και για την δημιουργία αντικείμενα που θα περιέχουν χρησιμοποιήσαμε τις ακόλουθες γλώσσες και εργαλεία:

Χρήση γλώσσας HTML

Για τη δημιουργία των διεπαφών χρησιμοποιήσαμε την γλώσσα HTML (HyperText Markup Language) η οποία είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες. Τα αντικείμενα και τα στοιχεία τα οποία ονομάζονται *ετικέτες* (tags) είναι τα βασικά δομικά

στοιχεία των ιστοσελίδων όπως για παράδειγμα κουμπί, παράγραφος, κεφαλίδα, σύνδεσμος κ.α.

Χρήση της γλώσσας CSS

Η CSS *(Cascading Style Sheets*) γλώσσα χρησιμοποιείται για τον έλεγχο της εμφάνισης ενός αρχείου HTML. Διαμορφώνει στιλιστικά την ιστοσελίδα δηλαδή διαμορφώνει χαρακτηριστικά αντικειμένων, χρώματα, στοίχιση κ.α.

Χρήση της γλώσσας Java Script και JQuery

Για την λειτουργικότητα των διεπαφών χρησιμοποιήσαμε την διερμηνευμένη γλώσσα προγραμματισμού JavaScript. Ουσιαστικά η γλώσσα αυτή βοήθησε στην επικοινωνία μεταξύ των αντικειμένων της ιστοσελίδας και του χρήστη. Δηλαδή για να εναλλάσουν δεδομένα ασύγχρονα και να συνεπώς να αλλάζουν δυναμικά το περιεχόμενο του εγγράφου που εμφανίζεται. Για παράδειγμα η υλοποίηση της λειτουργικότητας ενός κουμπιού.

JQuery είναι βιβλιοθήκη της Java Script και χρησιμοποιείται για τον ίδιο σκοπό όπως και η Java Script.

Χρήση Ajax

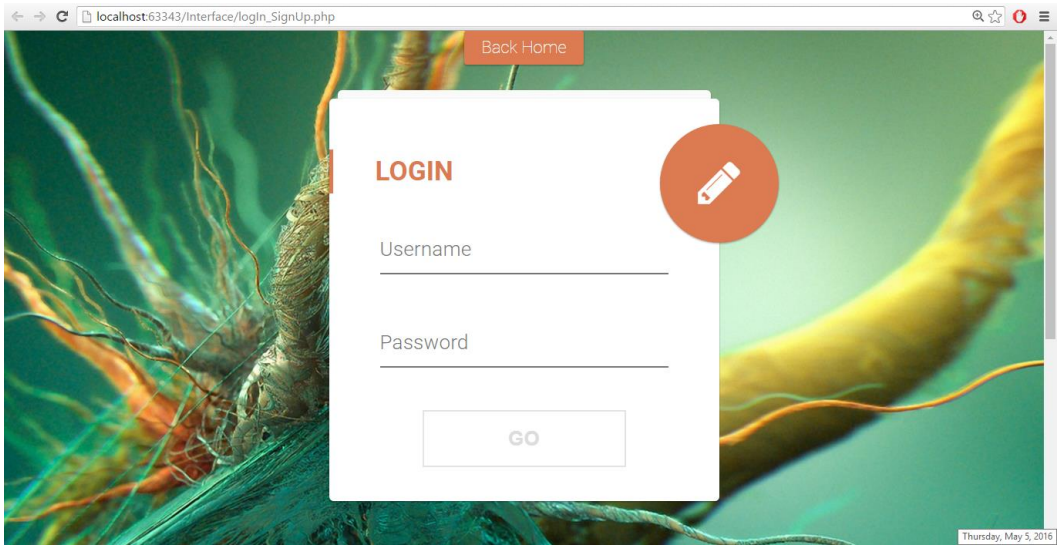
Χρησιμοποιήσαμε Ajax για να μπορούμε να ανταλλάζουμε δεδομένα με τον διακομηστή χωρίς να ξαναφορτώνεται η σελίδα. Ajax χρησιμοποιήσαμε στην λειτουργία όπου έπρεπε να σταλεί ένα email και να παραχθεί ένας κωδικός.

Χρήση Bootstrap framework

Το Bootstrap framework είναι μια βιβλιοθήκη από την οποία μπορείς να επιλέξεις αντικείμενα για την ιστοσελίδα σου ούτως ώστε αυτή να είναι αποκριτική (responsive). Χρησιμοποιήσαμε το Bootstrap framework κυρίως όταν θέλαμε να δημιουργήσουμε modals.(Contact us modal, Email-Generate Code modal)

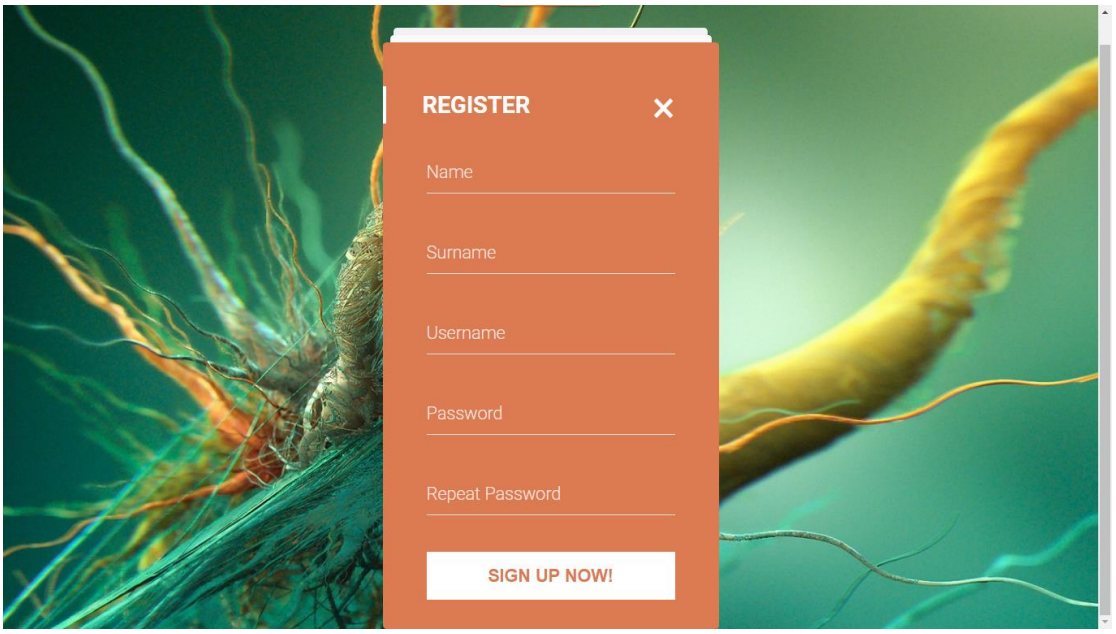
2.2.4.2 Διεπαφή Σύνδεσης και Εγγραφής του Χρήστη

Η Διεπαφή Σύνδεσης Χρήστη (Εικόνα 2.4) επιτρέπει στο χρήστη να συνδεθεί στη διαδικτυακή εφαρμογή χρησιμοποιόντας το χαρακτηριστικό όνομα χρήστση (Username) και τον κωδικό πρόσβασής (Password) του.



Εικόνα 2.4: Διεπαφή Σύνδεσης Χρήστη

Πατώντας το πορτοκαλί κουμπί με το μολύβι ο χρήστης έχει τη δυνατότητα εάν δεν είναι εγγεγραμμένος στο σύστημα να εγγραφεί. Όπως φαίνεται και στην εικόνα 2.5, ο χρήστης θα πρέπει να παρέχει κάποια από τα προσωπικά του στοιχεία προκειμένου να μπορεί να εγγραφεί. Με το πάτημα του κουμπιού Sign Up Now! θα γίνεται έλεγχος στη φόρμα αν όλα τα παιδία είναι συμπληρωμένα και με κατάλληλα στοιχεία και ο χρήστης θα μεταφέρεται στη διεπαφή σύνδεσης χρήστη. Σε αντίθετη περίπτωση θα εμφανίζεται μήνυμα λάθους.



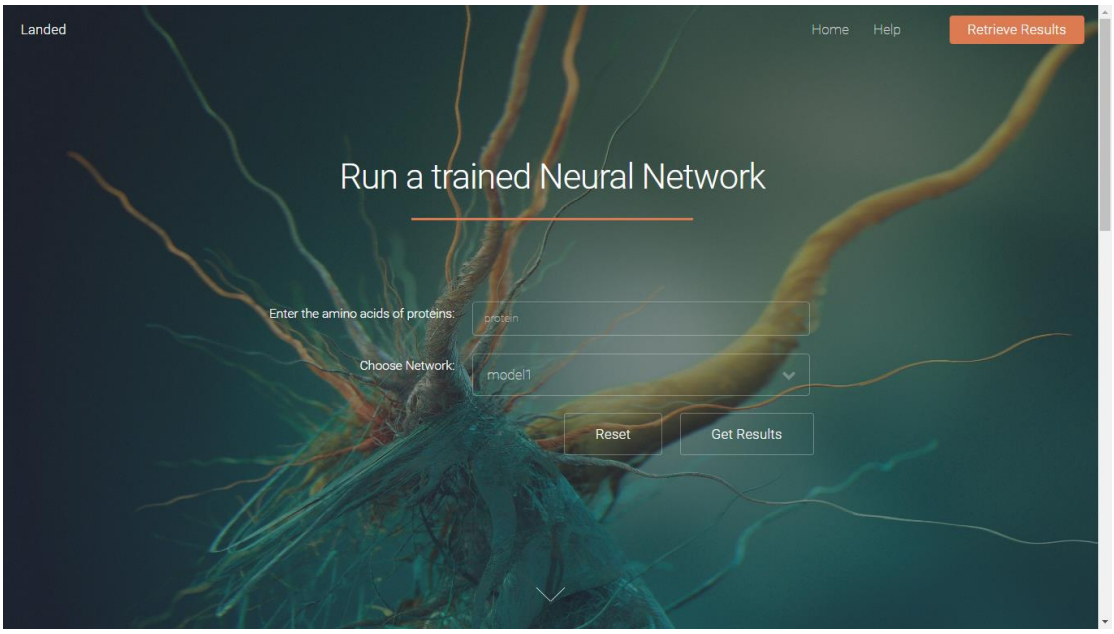
Εικόνα 2.5: Διεπαφή Εγγραφής Χρήστη

2.2.4.3 Διεπαφή Κεντρικής Σελίδας

Η κεντρική σελίδα χωρίζεται σε 3 κύρια παράθυρα στα οποία ο χρήστης μεταβαίνει με μετακινόντας τον καίρσορά του προς τα κάτω ή πατώντας στο τοξο που εμφανίζεται στο κάτω μέρος κάθε παραθύρου.

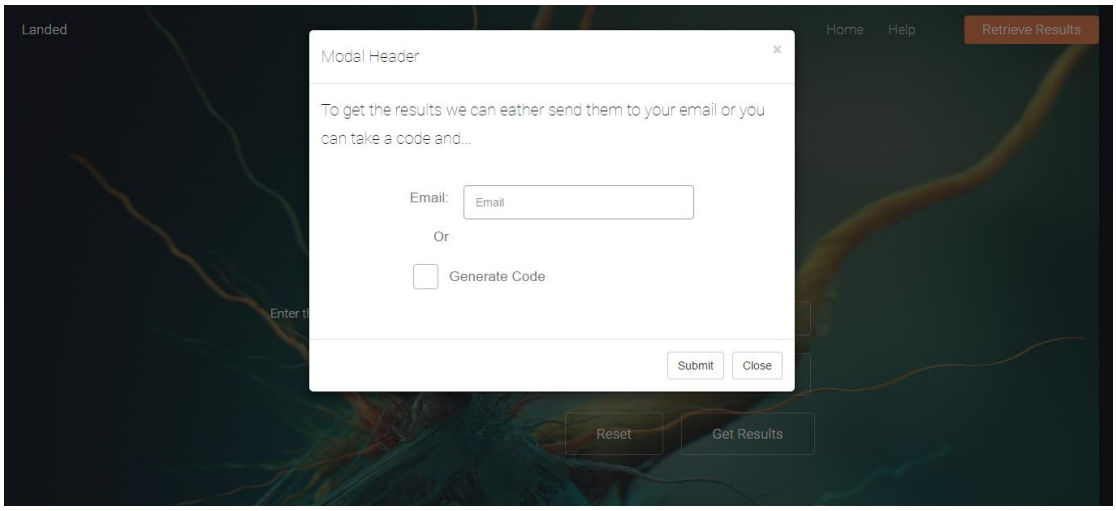
Πρώτο Παράθυρο

Στο πρώτο παράθυρο (Εικόνα 2.6) ο χρήστης θα μπορεί να τρέξει ένα ήδη εκπεδευμένο Τεχνητό Νευρωνικό Δίκτυο εισάγοντας την πρωτεϊνική ακολουθία με την οποία θέλει να τρέξει το δίκτυο και το επιλέγοντας ένα μοντέλο δικτύου.



Εικόνα 2.6: Κύρια σελίδα - Πρώτο Παράθυρο

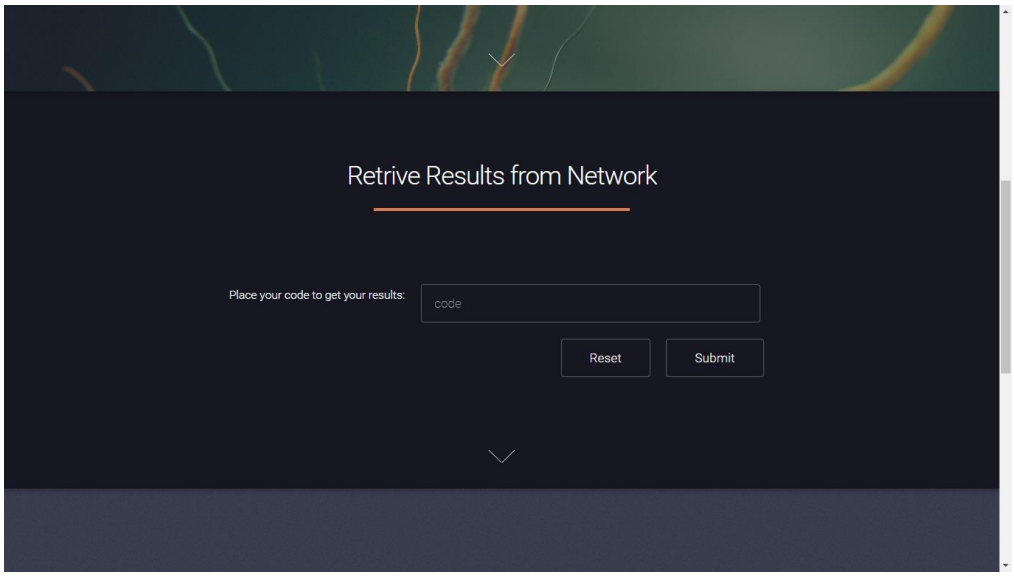
Ακολούθως αν τα δύο πεδία δεν είναι άδεια πατώντας το κουμπί Get Results μεταφέρεται στο modal όπως φαίνεται στην Εικόνα 2.7. Σε αντίθετη περίπτωση εμφανίζεται μήνυμα λάθους. Ο χρήστης σε αυτή τη φόρμα καλείται αν θέλει να δώσει τη διεύθυνση του προσωπικού του ταχυδρομίου για να σταλούν τα αποτελέσματα του δικτύου. Αν δεν επιθυμεί να δώσει κάποιο email τότε με το πάτημα του κουμπιού Submit παράγεται ένας κωδικός για να έχει πρόσβαση μέσω αυτού στα αποτελέσματα του δικτύου που έτρεξε.



Εικόνα 2.7: Καταχώρηση email ή παραγωγή κωδικού

Δεύτερο Παράθυρο

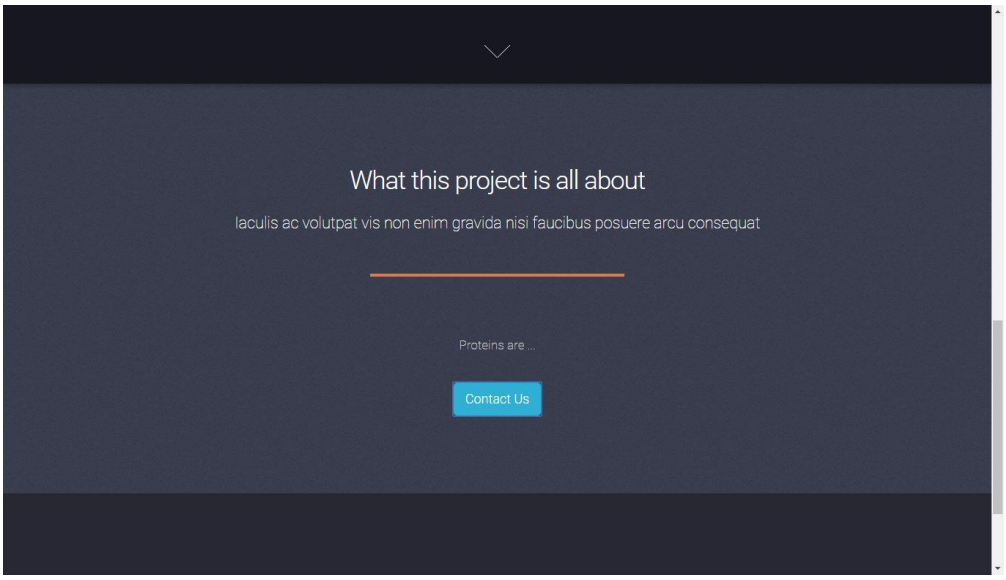
Το παράθυρο αυτό αφορά την περίπτωση όπου ο χρήστης δεν δίνει το e-mail του και λαμβάνει μοναδικό κωδικό για λήψη των δεδομένων. Στο παράθυρο αυτό (Εικόνα 2.8) θα καταχωρεί τον κωδικό και θα παίρνει τα αποτελέσματα. Σε περίπτωση που τα αποτελέσματα δεν είναι έτοιμα θα εμφανίζεται μήνυμα που να τον ενημερώνει.



Εικόνα 2.8: Παραλαβή Αποτελεσμάτων - Παράθυρο δεύτερο

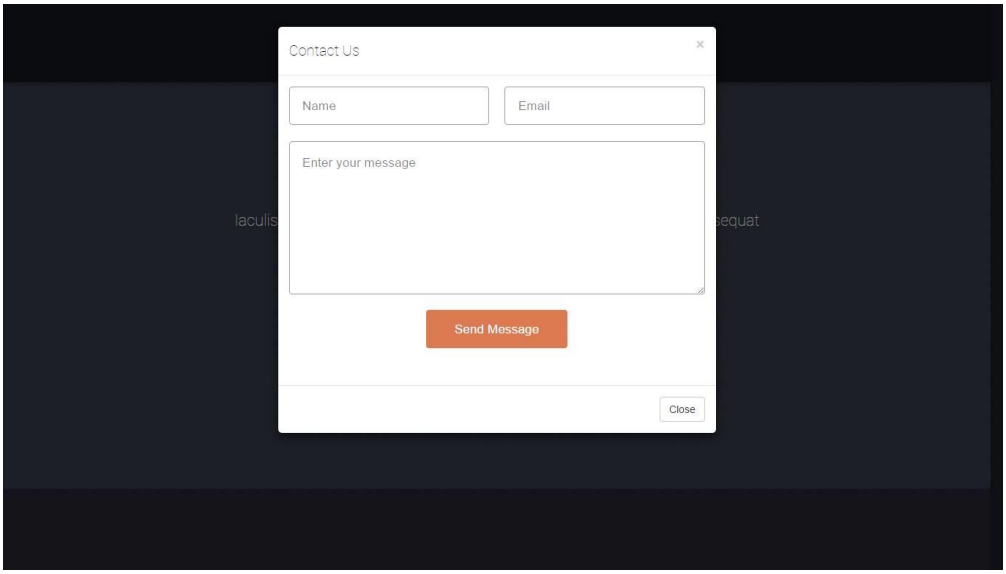
Τρίτο Παράθυρο

Στο τρίτο παράθυρο (Εικόνα 2.9) ο χρήστης θα έχει την δυνατότητα να μάθει περισσότερες πληροφορίες για την διαδικτυακή εφαρμογή και για το PSSP πρόβλημα.



Εικόνα 2.9: Πληροφορίες για το PSSP πρόβλημα - Τρίτο Παράθυρο

Επίσης με το πάτημα του κουμπιού Contact Us θα εμφανίζεται ένα modal (Εικόνα 2.10) στο οποίο ο χρήστης συμπληρώνοντας τα παιδία name,email και γράφοντας ένα κείμενο θα μπορεί να επικοινωνήσει μαζί μας.



Εικόνα 2.10: Contact Us

2.2.5 Back End

2.2.5.2 Αποστολή email

Εκτός από την αλληλεπίδραση του χρήστη με τις διεπαφές της διαδικτυακής εφαρμογής, οι οποίες σχολιάστηκαν πιο πάνω, η επικοινωνία του χρήστη και διαδικτυακής εφαρμογής μπορεί να επιτευχθεί και μέσω της αποστολής email.

Από την πλευρά του χρήστη η διαδικασία αυτή προστέθηκε για να μπορεί ο ίδιος ανά πάσα χρονική στιγμή να επικοινωνήσει με τους συντηρητές της διαδικτυακής εφαρμογής για την επίλυση οποιονδήποτε απορίων ή για την αποστολή σχολίων του. Με βάση το δεύτερο παράθυρο της κεντρικής σελίδας ο χρήστης έχει την δυνατότητα να μεταφερθεί σε ένα model το οποίο περιέχει μια φόρμα. Με την συμπλήρωση της φόρμας (Εικόνα 2.10) ο χρήστης θα μπορεί να στείλει email με το κείμενο που επιθυμεί.

Από την πλευρά της διαδικτυακής εφαρμογής η λειτουργία αυτή αποτελεί μια από τις δύο μεθόδους αποστολής των αποτελεσμάτων της εκτέλεσης των εκπαιδευμένων Τεχνητών Νευρωνικών Δικτύων που είναι διαθέσιμα στην εφαρμογή. Ο χρόνος που απαιτείται για να αποσταλεί το email με τα αποτελέσματα εξαρτάται απόλυτα από τον χρόνο που χρειάζεται το επιλεγμένο από τον χρήστη τεχνητό νευρωνικό δίκτυο για να τρέξει την πρωτεϊνική ακολουθία που εισήγαγε ο χρήστης. Ακολούθως αυτόματα θα στέλνεται ένα email στο χρήστη με τα αποτελέσματα.

Για την υλοποίηση της διαδικασίας αποστολής του email χρησιμοποιήθηκε το PHPMailer, πακέτο το οποίο αποτελείται από κλάσεις που διαθέτουν τα πλήρη χαρακτηριστικά για την δημιουργία και αποστολή email με την βοήθεια της γλώσσας προγραμματισμού PHP [39]. Η επιλογή του πακέτου αυτού έγινε επειδή η γλώσσα προγραμματισμού PHP έχει χρησιμοποιηθεί κατά κόρον για την επικοινωνία των διεπαφών της διαδικτυακής εφαρμογής και της Βάσης Δεδομένων. Δηλαδή την επικοινωνία μεταξύ Frond-End και Back-End.

Με βάση το πακέτο αυτό δημιουργείται ένα αυτοματοποιημένο μήνυμα το οποίο περιλαμβάνει τα αποτελέσματα του ΤΝΔ που εκτελέστηκε. Επίσης να σημειωθεί ότι για την επιτυχή αποστολή του email έπρεπε πρώτα να αποφευχθούν ορισμένα προβλήματα που είχαμε με τις ρυθμίσεις του localhost. Προβλήματα που έχουν σχέση με την προσβασιμότητα μας σε πλατφόρμες αποστολής email όπως για παράδειγμα Gmail.

2.2.5.2 Εκτέλεση δικτύου με δημιουργία MD5 κωδικού

Η εξήγηση των back-end λειτουργιών θα περιγραφούν με ψευδοκώδικα για σαφέστερη κατανόηση των βημάτων στο παρόν αλλά και σε μεταγενέστερο στάδιο.

Στην περίπτωση που χρήστης δεν δώσει το email του, για αποστολή αποτελεσμάτων γίνονται τα ακόλουθα:

1. Γίνεται έλεγχος στη βάση δεδομένων για το κατά πόσο η αίτηση θα ξεπεράσει το μέγιστο αριθμό παράλληλων διεργασιών στο διακομιστή. Εάν η απάντηση είναι αρνητική, τότε ζητείται από το χρήστη να δοκιμάσει αργότερα.
2. Ασύγχρονο ajax call στη PHP για λήψη του μοναδικού MD5 κωδικού. Στο σημείο αυτό παράγεται ο κωδικός. Παράλληλα, δημιουργείται εγγραφή στη ΒΔ με μορφή

(κωδικός, αποτέλεσμα), όπου στο πεδίο κωδικός εισάγεται ο MD5 κωδικός ενώ το πεδίο των αποτελεσμάτων είναι κενό.

Όταν τα αποτελέσματα είναι έτοιμα, γίνεται κλήση στη ΒΔ, για συμπλήρωση της εγγραφής (κωδικός, αποτέλεσμα).

Κατά την προσπάθεια του χρήστη να λάβει τα αποτελέσματα, γίνονται τα παρακάτω:

1. Έλεγχος εάν η εγγραφή στη ΒΔ έχει ολοκληρωθεί. Σε περίπτωση που το πεδίο "αποτελέσματα" είναι κενό, ζητείται από το χρήστη να δοκιμάσει αργότερα.
2. Εάν το πεδίο είναι συμπληρωμένο, τότε δημιουργείται αρχείο που να περιέχει τα αποτελέσματα και ο φυλλομετρητής ζητά από το χρήστη να κατεβάσει το αρχείο.
3. Αμέσως, η εγγραφή του αποτελέσματος από τη ΒΔ μας, διαγράφεται.

2.3 Δεδομένα εισόδου

2.3.1 Δεδομένα εισόδου και νευρωνικά δίκτυα

Χωρίς αμφιβολία, ένα από τα σημεία που χρίζει ιδιαίτερης σημασίας, κατά την προσπάθεια επίλυσης κάποιου προβλήματος κάνοντας χρήση των ΤΝΔ, είναι η επιλογή του συνόλου δεδομένων εισόδου (dataset). Το δίκτυο θα εκπαιδευτεί βάσει των δεδομένων που θα επιλέξουμε, αφού στην προσπάθειά του για μερική αφομοίωσή τους θα αλλάζει τα συναπτικά του βάρη. Γενικός κανόνας των νευρωνικών δικτύων είναι ο διαχωρισμός των δεδομένων εισόδου σε δύο ομάδες. Τα δεδομένα εκπαίδευσης (training data) και τα δεδομένα επαλήθευσης (test data). Ο διαχωρισμός αυτός αποσκοπεί στην προσαρμογή των βαρών, του δικτύου, κάνοντας χρήση των training data και ακολούθως στον έλεγχο του κατά πόσο το δίκτυο μπορεί να προβλέψει σωστά δεδομένα τα οποία δεν του παρουσιάστηκαν προηγουμένως. Όπως είναι αντιληπτό, η μέθοδος ανάστροφης μετάδοσης λάθος εφαρμόζεται μόνο στα δεδομένα εκπαίδευσης.

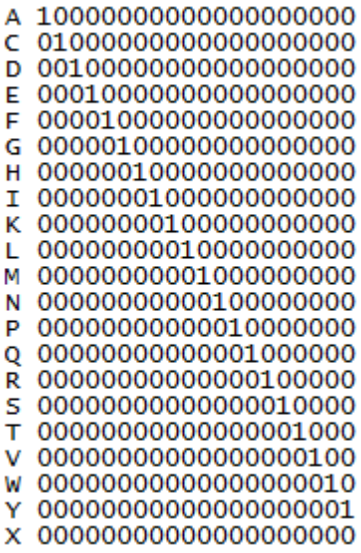
Στόχος μας είναι η εύρεση dataset που να μας παρέχει για κάθε πρωτεΐνη, την πρωτοταγή της δομή (δηλ. την αλληλουχία των αμινοξέων που την αποτελούν) και την αντίστοιχη δευτεροταγή δομή, που θα λειτουργεί ως επιθυμητό αποτέλεσμα. Οι δευτεροταγείς δομές χωρίζονται συνήθως σε 8 κατηγορίες [7] αλλά από το 2005 συνηθίζεται να διαχωρίζονται σε τρεις [29].

2.3.2 MSA profiles

Στο Π.Κ., χρησιμοποιείται τα τελευταία χρόνια η προσέγγιση Multiple Sequence Alignment (MSA) profiles [9]. Η στοίχιση ακολουθιών είναι μια ευρέως γνωστή προσέγγιση στον τομέα της βιοπληροφορικής και αφορά ακολουθίες DNA, RNA και πρωτεϊνών. Επί της ουσίας, γίνεται προσπάθεια εύρεσης ομοιοτήτων μεταξύ των προαναφερθέντων βιολογικών ακολουθιών αφού οι ομοιότητες ορίζουν συνήθως κάποια βιολογική συσχέτιση, οδηγώντας στην καλύτερη κατανόηση των βιολογικών μηχανισμών.

Κατά την στοίχιση θα πρέπει να γίνει επιλογή πρωτεϊνών που να καλύπτουν όλο το φάσμα των διαφορετικών πρωτεϊνών, εκτείθοντας το δίκτυο σε ένα πιο άρτια δομημένο

σύνολο δεδομένων, ούτως ώστε να μπορεί το δίκτυο να γενικεύει καλύτερα τα δεδομένα ελέγχου. Η στοίχιση στο σύνολο δεδομένων που χρησιμοποιείται παρουσιάζει μια μορφή κωδικοποίησης, διαφορετική από την one-hot vector [20] (Εικόνα 2.11), παρέχοντας σε αυτό περισσότερη πληροφορία.



Εικόνα 2.11: One Hot vector [20]

Αναλυτικότερα, για το σχηματισμό του dataset έγινε, αρχικά, πολλαπλή στοίχιση της εξετάζουσας πρωτεΐνης με άλλες που έχουν έντονη ομοιότητα με αυτή. Επιτελείται ευθυγράμμιση των εξελικτικά σχετιζόμενων πρωτεϊνών, λόγω του ότι αφού υπάρχει εξελικτική συσχέτιση μεταξύ τους, τότε θα έχουν την ίδια δομή στο χώρο [9][21]. Αποτέλεσμα της στοίχισης είναι τα MSA profiles των διαφόρων πρωτεϊνικών οικογενειών που παρά την διαφορά των ακολουθιών τους, δίνουν την ίδια δομή στο χώρο.

Εκτελώντας τα παραπάνω, έχουμε πετύχει ένα νέο είδος κωδικοποίησης (Εικόνα 2.12) που αφορά τη θέση του αμινοξέως. Ουσιαστικά, αντί να μας ενδιαφέρει το αμινοξύ που βρίσκεται σε μια θέση, νοιαζόμαστε για την πιθανότητα ύπαρξης συγκεκριμένου αμινοξέως στην εξετάζουσα θέση. Το δίκτυο λαμβάνει δεδομένο εισόδου 20 αμινοξέων,

	V	L	I	M	F	W	Y	G	A	P	S	T	C	H	R	K	Q	E	N	D
N	0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	16	7	16	40	10
K	1	1	1	1	0	0	0	4	0	0	0	4	0	1	5	77	1	0	3	0
C	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0
P	1	1	0	0	1	0	2	22	5	48	4	2	1	7	1	2	0	1	1	1

Εικόνα 2.12: Στο παράδειγμα MSA profile φαίνονται οι πιθανότητες εμφάνισης των αμινοξέων σε μια θέση.

αλλά τώρα η τιμή που τροφοδοτείται σε αυτό αφορά την πιθανότητα εμφάνισης του αμινοξέως σε συγκεκριμένη θέση.

2.3.3 Χρησιμοποιώντας τα MSA profiles για image recognition

Τι είναι ψηφιακή εικόνα;

Η ψηφιακή εικόνα είναι η αριθμητική αναπαράσταση, σε ένα πίνακα, μιας εικόνας (Εικόνα 2.13). Η μαυρόασπρη ψηφιακή εικόνα σχηματίζεται με ένα δισδιάστατο πίνακα ο οποίος σε κάθε κελί του λαμβάνει έναν 8-bit διαδικό αριθμό, επομένως τιμές [0, 255], ενώ η πολύχρωμη ψηφιακή εικόνα αποτελείται από τρείς πίνακες (με τιμές [0 255]), έναν για κάθε κύριο χρώμα (RGB – Red, Green, Blue).

34	45	…	177
56	88	…	250
…	…	…	…
57	100	…	86

Εικόνα 2.13: Ψηφιακή εικόνα σε μορφή δισδιάστατου πίνακα

Κύρια Ιδέα

Στην προσπάθεια να διατηρήσουμε το σύνολο δεδομένων που έχουμε στην κατοχή μας (MSA files), σκεφτήκαμε ένα "τρελό" τρόπο χρήσης των αρχείων. Η κύρια ιδέα, του τρόπου λειτουργείας, ήταν να οπτικοποιήσουμε (Εικόνα 2.20) τα MSA files καταχωρώντας τις τιμές τους σε δυσδιάστατους πίνακες (Εικόνα 2.14) και στη συνέχεια να τους εξάγουμε προκειμένου να τύχουν επεξεργασίας από το Convolutional δίκτυο.

0	0	0	0	0	0	0	0	0	0	10	0	0	0	0	16	7	16	40	10
1	1	1	1	0	0	0	4	0	0	0	4	0	1	5	77	1	0	3	0
0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0
1	1	0	0	1	0	2	22	5	48	4	2	1	7	1	2	0	1	1	1
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…

Εικόνα 2.14: Δημιουργία της ψηφιακής εικόνας κάνοντας χρήση το MSA (Εικόνα 2.2)

2.3.4 Δημιουργία του συνόλου δεδομένων

Όπως προ ειπώθηκε, το σύνολο δεδομένων των νευρωνικών δικτύων επιβλεπόμενης μάθησης αποτελείται από τα δεδομένα εισόδου και τις επιθυμητές εξόδους του δικτύου, για κάθε ένα από αυτά. Συμπεραίνουμε ότι το δίκτυο που θα προβλέπει τη δευτεροταγή δομή της πρωτεΐνης θα λαμβάνει ως είσοδο την πρωτοταγή δομή και θα του δίνεται ως επιθυμητό αποτέλεσμα η δευτεροταγής (στα δεδομένα εκπαίδευσης). Βάση της κύριας ιδέας, αποφασίστηκε η εξής προεπαιξεργασία των δεδομένων.

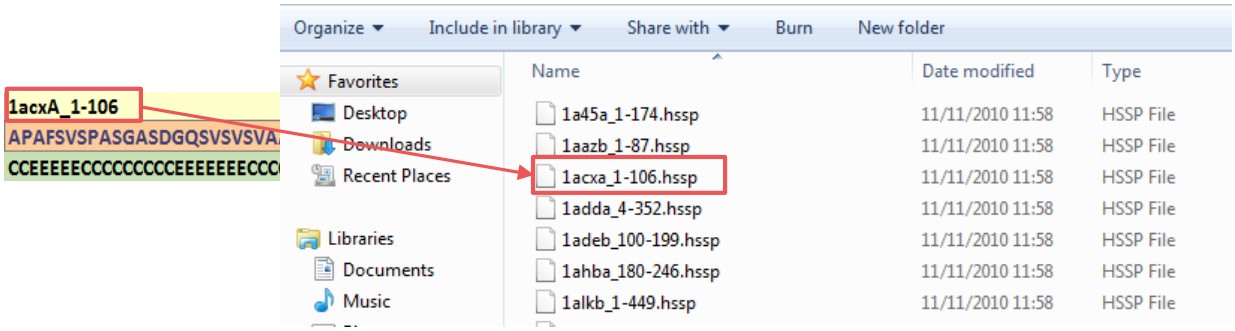
Ξεκινώντας, έχουμε στη διάθεσή μας αρχείο που να περιέχει εγγραφές πρωτεϊνών της παρακάτω μορφής:

```
1acxA_1-106
APAFSVSPASGASDGQSVSVSVAAAGETYYIAQCAPVGGQDACNPATATSFTTDASGAASFSFTVRKSYAGQTPSGTPVGSVDCATDACNLGAGNSGLNLGHVALTF
CCEEEEECCCCCCCCCEEEEEECCCCEEEEEECEECCEECCCCCCCCEEECCCCCCCEEEECCCEEEECCCCCEEEEECCCCCCEEEEECCCCCCCCCCCCC
```

Εικόνα 2.15: Παράδειγμα εγγραφής μιας πρωτεΐνης στο αρχείο πρωτεϊνών.

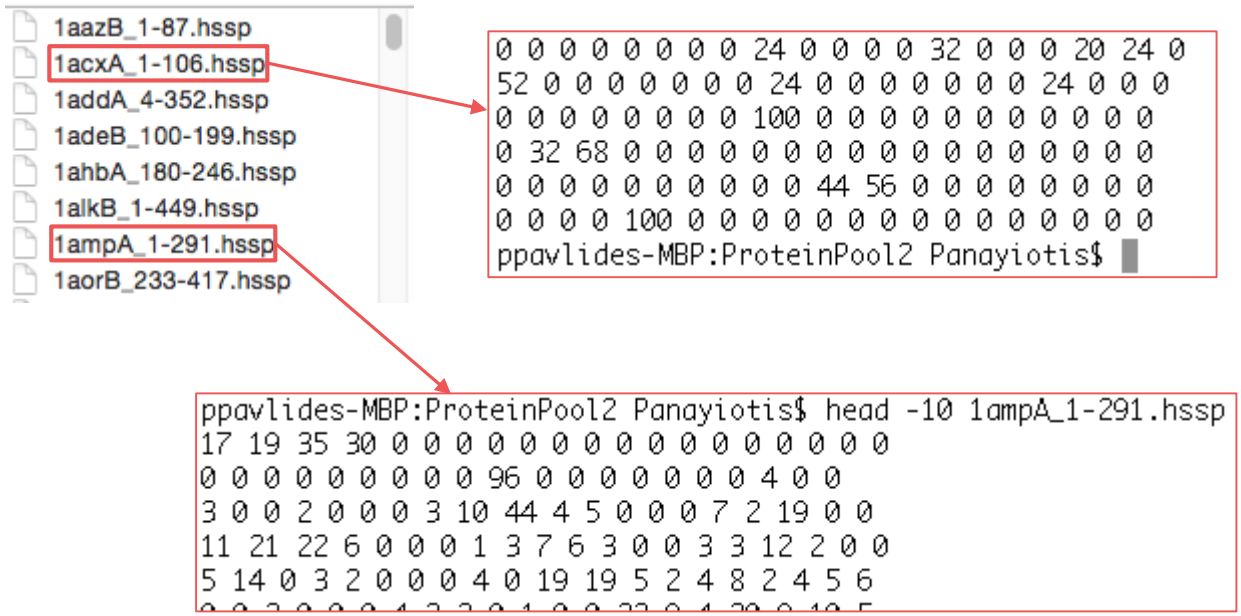
Η πρώτη γραμμή της Εικόνας 2.15 αναγράφει το όνομα της πρωτεΐνης, η δεύτερη την πρωτοταγή δομή και η τρίτη την δευτεροταγή δομή, το επιθυμητό, ουσιαστικά, αποτέλεσμα. Όπως υπογραμμίστηκε στο Υποκεφάλαιο 2.3.2, η χρήση των MSA files παρέχει στο δίκτυο περισσότερη πληροφορία. Θα πρέπει επομένως, να αντικαταστήσουμε την πρωτοταγή δομή, Εικόνα 2.15, με την αντίστοιχη ακολουθία των MSA files [9].

Εν συνεχεία, για κάθε πρωτεΐνη που βρίσκεται στο αρχείο της Εικόνας 2.15, υπάρχει αντίστοιχο αρχείο που τηρεί την πρωτοταγή της δομή στη μορφή των MSA files.

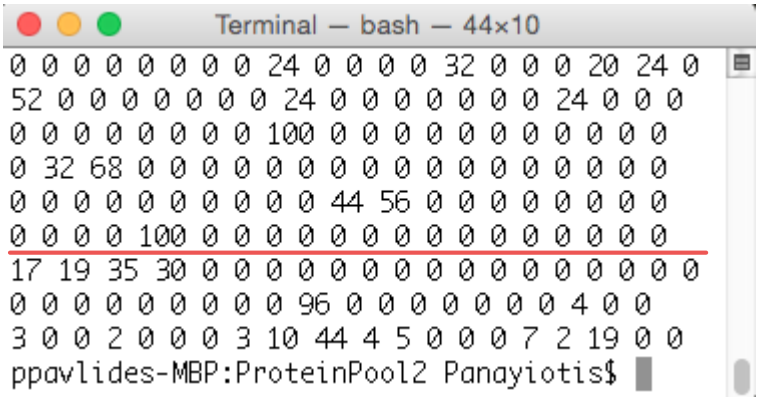


Εικόνα 2.16: Αντικατάσταση της πρωτοταγούς δομής (Εικόνας 2.5) με την MSA.

Ακολούθως, συμπτύξαμε όλα τα αρχεία *.hssp (MSA αρχεία) σε ένα μόνο αρχείο ενώ παράλληλα έγινε σύμπτυξη και των επιθυμητών δεδομένων εξόδου.

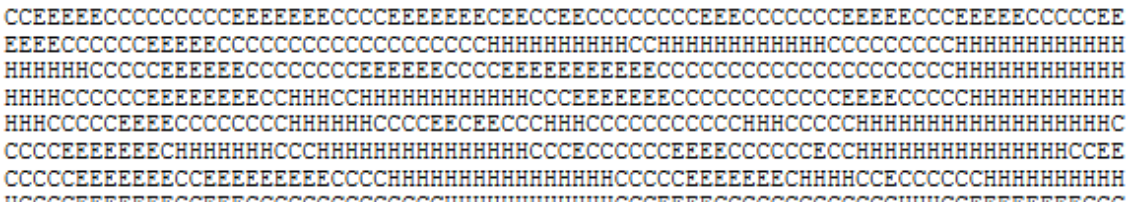


Εικόνα 2.17: Αρχεία πριν την σύμπτυξη



Εικόνα 2.18: Αρχείο μετά τη σύμπτυξη

Ομοίως, γίνεται σύμπτυξη της δευτεροταγούς δομής όλων των πρωτεϊνών:



Εικόνα 2.19: Δευτεροταγής δομή

Για κάθε γραμμή της Εικόνας 2.18 γίνεται 1-1 αντιστοίχιση με γράμμα της Εικόνας 2.19.

Σε προηγούμενες προσεγγίσεις του προβλήματος, χρησιμοποιήθηκε κινητό παράθυρο μεγέθους 30 [8][9][14], πράγμα που μας οδήγησε στον κατακερματισμό του MSA αρχείου σε εικόνες διαστάσεων 31×20 (ο αριθμός 20 αντιπροσωπεύει τον αριθμό των αμινοξέων που συμμετέχουν στη δημιουργία της πρωτεΐνης).

Εικόνα 2.20: Οπτικοποιημένη εικόνα τυχαίου δείγματος.

Κεφάλαιο 3

Convolutional Neural Network

3.1 Pattern Recognition

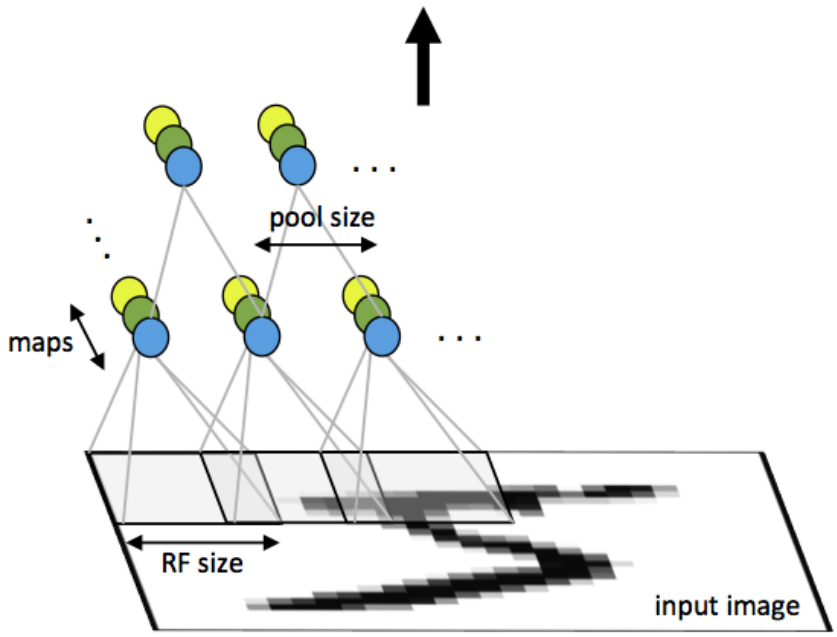
3.2 Εισαγωγή στα ConvNets

3.3 Αρχιτεκτονική των Convolutional Neural Networks

3.1 Pattern Recognition

Το pattern recognition αποτελεί κύριο τομέα μελέτης για τη μηχανική μάθηση, τις τελευταίες δεκαετίες. Κατά τη μελέτη της όρασης των ζώων, στα τέλη της δεκαετίας του ’60, ανακαλύφθηκαν δύο ειδών νευρώνες. Ο πρώτος, ο “απλός” νευρώνας, ανταποκρίνεται σε συγκεκριμμένα μοτίβα, όπως ακμές, που βρίσκονται εντός ενός παραθύρου υποδοχής, receptive field, και ο δεύτερος νευρώνας, ο “σύνθετος”, είναι ευαίσθητος σε μεγαλύτερα receptive fields και έχει την ικανότητα να αναγνωρίζει τα μοτίβα, στο πεδίο όρασης, ανεξαρτήτου τοποθεσίας.

Η πιο πάνω ανακάλυψη πυροδότησε μια σειρά από έρευνες και δοκιμές στον τομέα της μηχανικής μάθησης, αφού ήταν δεδομένο πλέον στην επιστημονική κοινότητα η χρήση παραθύρων, feature maps, για περεταίρω εξέταση των δεδομένων εισόδου. Οι προσπάθειες για τη δημιουργία των πιο αποδοτικών feature maps ξεκίνησαν με χειροτεχνία. Το κυριότερο μειονέκτημα της προσέγγισης αυτής ήταν η ικανότητα του σχεδιαστή να δημιουργήσει τα καταλληλότερα feature maps που θα εξήγαγαν τα καλύτερα δυνατά αποτελέσματα. Έτσι, έγινε προσπάθεια για συνδιασμό αυτών των feature maps με τεχνικές αυτοματοποίησης της παραγωγής τους.



Εικόνα 3.1 [37]: Επισκόπηση του τρόπου λειτουργίας των ConvNets.

3.2 Εισαγωγή στα ConvNets

Η ικανότητα των δικτύων με πολλαπλά επίπεδα, που εκπαιδεύονται με πολύπλοκα - high dimensional δεδομένα και κάνουν χρήση της μεθόδου ανάστροφης μετάδοσης λάθους, τα προσδιορίζει ως κατάλληλα στον τομέα της αναγνώρισης εικόνας. Όπως προαναφέρθηκε, επιθυμούμε κάποιο δίκτυο που να μπορεί, μέσω της εκπαίδευσης του, να σχηματίζει τα καλύτερα feature maps, χωρίς έτσι να κρίνεται η απόδοση του δικτύου από τις ικανότητες του σχεδιαστή.

Η άπληστη λογική, όσον αφορά την αρχιτεκτονική του δικτύου, δεν ενδιαφέρει τον τομέα της μηχανικής μάθησης, μιας και η χρήση πολλών πόρων σε ένα δίκτυο αυξάνει τον αριθμό των προσαρμόσιμων παραμέτρων (free – adjustable parameters). Αποτέλεσμα αυτού είναι η εκθετική αύξηση των πιθανών συμπεριφορών του δικτύου. Μια τέτοια προσέγγιση απαιτεί ένα πολύ μεγάλο training set το οποίο μπορεί να μεν να μας δώσει καλά αποτελέσματα στην εκπαίδευση, παρόλα αυτά έχει μεγάλη πιθανότητα να υπερεκπαιδευτεί. Το κυριότερο, όμως, αρνητικό ενός τέτοιου δικτύου είναι η ανικανότητα του να παραμείνει άθικτο στην παρουσία θορύβου ή φθαρμένων δεδομένων εισόδου. Εκλαϊκεύοντας το πρόβλημα αυτό, αντιλαμβανόμαστε ότι ουσιαστικά το δίκτυο είναι επιρρεπές στην ποικιλομορφία, αφού αδυνατεί να γενικεύσει τα δεδομένα με αποτέλεσμα να προσπαθεί να τα κατηγοριοποιήσει σε ξεχωριστή ομάδα. Όσον αφορά το πρόβλημα των πρωτεϊνών, ένα τέτοιο χαρακτηριστικό μπορεί να επηρεάσει σε αρνητικό βαθμό τα αποτελέσματα λόγω της ποικιλομορφίας που παρουσιάζει το PSSP από μόνο του.

Εν συνεχεία, ένα κλασικό fully-connected δίκτυο αγνοεί την τοπολογία των δεδομένων εισόδου, μιας και η σειρά παρουσίασης των μεταβλητών εισόδου, στο επίπεδο εισόδου, δεν επηρεάζει το τελικό αποτέλεσμα [1]. Εξ’ ορισμού, οι εικόνες έχουν αυστηρή δομή. Αυτό εξ υπακούει ότι μια μεταβλητή εισόδου – pixel – επηρεάζεται από τα γειτονικά του pixels. Για παράδειγμα, μια ακμή δεν μπορεί να θεωρηθεί ακμή εάν δεν συγκριθεί με τα γειτονικά της pixels. Έτσι εμφανίζεται η ανάγκη για τη δημιουργία feature maps (υποδοχέων δεδομένων) που να μπορούν να συσχετίζουν και να αντιλαμβάνονται τη χωρική διάταξη των δεδομένων εισόδου.

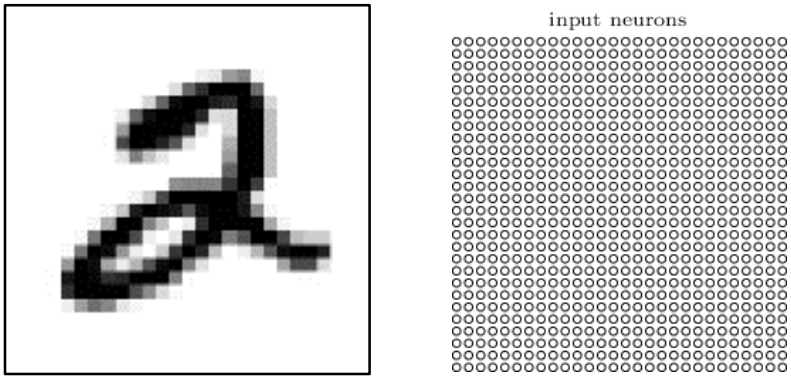
3.3 Αρχιτεκτονική των Convolutional Neural Networks

Η αμεταβλητότητα στην μετατόπιση, θόρυβο αλλά και κλίμακα των δεδομένων εισόδου επιτυγχάνεται από τα Convolutional Neural Networks (CNN) κάνοντας χρήση τριών κύριων αρχιτεκτονικών στοιχείων τους:

Local Receptive Fields

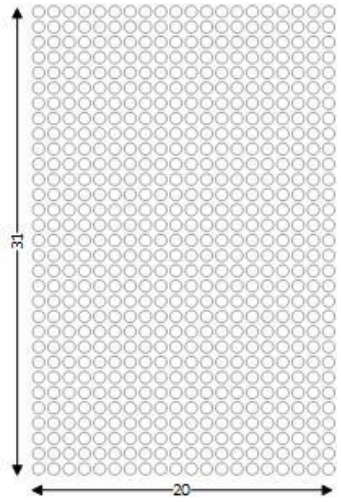
Για ευκολία στην επεξήγηση της αρχιτεκτονικής του δικτύου, θα χρησιμοποιήσουμε το πρόβλημα αναγνώρισης γραφικού χαρακτήρα (MNIST), όπως αυτό περιγράφεται από τον Yann LeCun (1989) [1] και παράλληλα θα γίνεται αναφορά στην *πρώτη* προσπάθεια εκτέλεσης των CNN δικτύων στο PSSP πρόβλημα.

Όπως είναι γνωστό, μια ασπρόμαυρη εικόνα δεν είναι άλλο παρά ένας δυσδιάστατος πίνακας που λαμβάνει τιμές από 0 (άσπρο στην Python) μέχρι 255 (μαύρο στην Python). Όλες οι ενδιάμεσες τιμές (0, 255) αποτελούν γκρίζες αποχρώσεις. Οι δυαδικές εικόνες λαμβάνουν μόνο τιμές 0 - για άσπρο - ή 1 - για μαύρο - στα pixels τους. Το δίκτυο (MNIST) λαμβάνει ως είσοδο δυαδικές εικόνες διαστάσεων 28×28, όπως φαίνεται παρακάτω (Εικόνα 3.2).



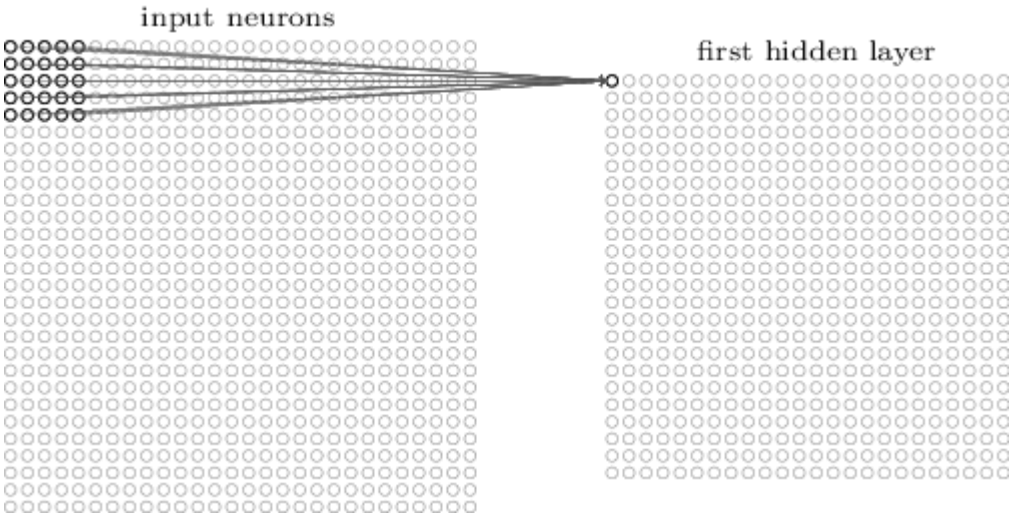
Εικόνα 3.2: Δυαδική εικόνα

Για τις ανάγκες του PSSP προβλήματος, χρησιμοποιούμε εικόνες μεγέθους 31×20 (Εικόνα 3.3).



Εικόνα 3.1: Διαστάσεις των εικόνων εισόδου στο PSSP πρόβλημα.

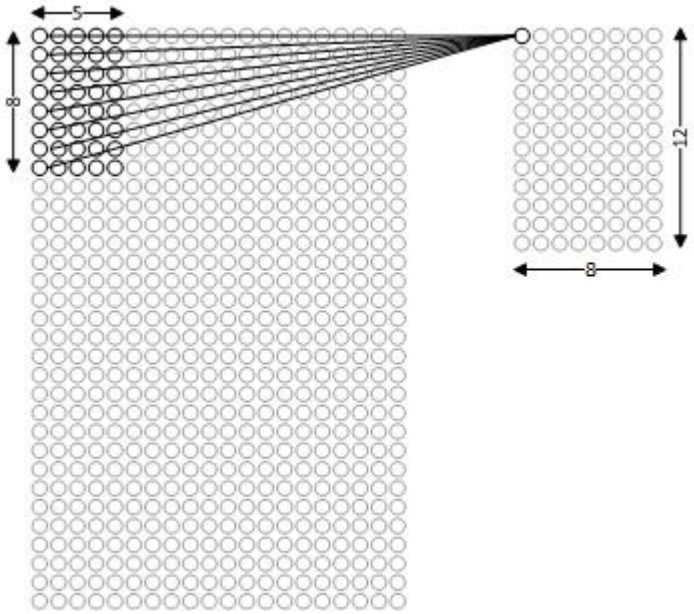
Η τιμή του κάθε pixel αναπαριστά, όπως προαναφέρθηκε στο Υποκεφάλαιο 2.3.4, το ποσοστό που έχει ένα αμινοξύ σε μια εγγραφή των msa αρχείων μας. Είναι σαφές ότι στο σημείο αυτό θα πρέπει να προσδιορίσουμε τον τρόπο σύνδεσης του επιπέδου εισόδου (input layer) με το πρώτο κρυφό επίπεδο (hidden layer). Η έναν προς έναν αντιστοίχιση των νευρώνων του input layer και του hidden layer θα εξάλειφε, ίσως τη σημαντικότερη ιδιότητα των CNNs, που είναι η ύπαρξη των local receptive fields. Αντ' αυτού, θα ενώσουμε μια γειτονιά (πυρήνα - kernel) από το επίπεδο εισόδου, σε ένα νευρώνα του κρυφού επιπέδου. Αυτή η γειτονιά ονομάζεται *local receptive field*.



Εικόνα 3.2 [5]: Receptive Field στο αναγνώρισης γραφικού χαρακτήρα

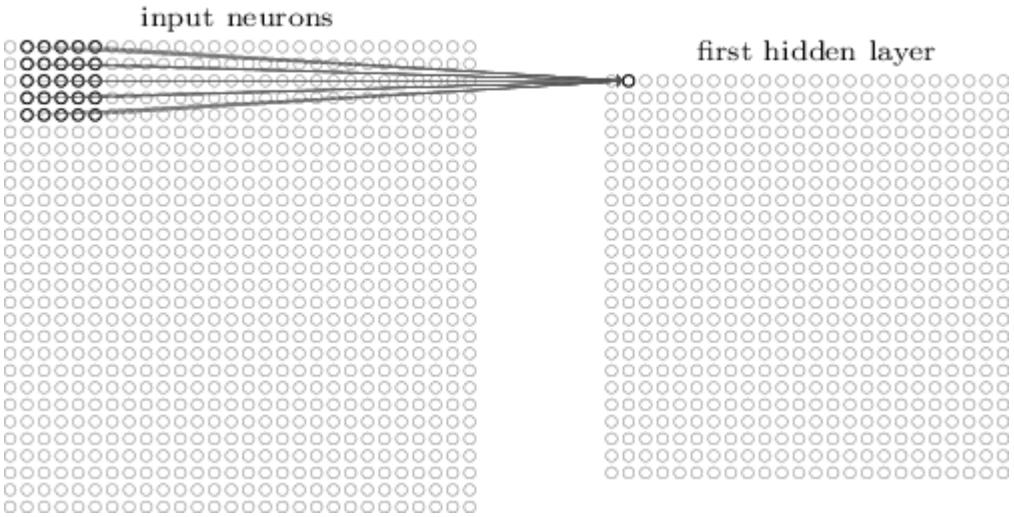
Από την εικόνα 3.4 φαίνεται ότι το μέγεθος του kernel που ψάχνει για χαρακτηριστικά, είναι 5×5. Εάν θεωρήσουμε τις γραμμές που εμφανίζονται στο σχήμα ως τα βάρη που μαθαίνει ένας νευρώνας για έναν kernel, τότε εύκολα μπορεί να εννοηθεί ότι ο κρυφός

νευρώνας "μαθαίνει" 25 συνολικά παραμέτρους, για τα βάρη. Αντίστοιχα, στο PSSP πρόβλημα χρησιμοποιούμε πυρήνα διαστάσεων 8×5 (Εικόνα 3.5).



Εικόνα 3.3: Receptive Field όπως χρησιμοποιήθηκε στο PSSP πρόβλημα

Μετακινώντας τον πυρήνα κατά μια θέση κάθε φορά σχηματίζονται καινούριες γειτονιές (receptive fields). Για κάθε μετατόπιση, άρα και κάθε γειτονιά, είναι υπεύθυνος ο αμέσως επόμενος κρυφός νευρώνας (Εικόνα 3.6). Συμπεραίνουμε, ότι ο κάθε κρυφός νευρώνας αναλύει τη δική του 12×8 γειτονιά στο επίπεδο εισόδου.



Εικόνα 3.4 [5]: Σάρωση γειτονικού receptive field.

Shared Weights and Biases

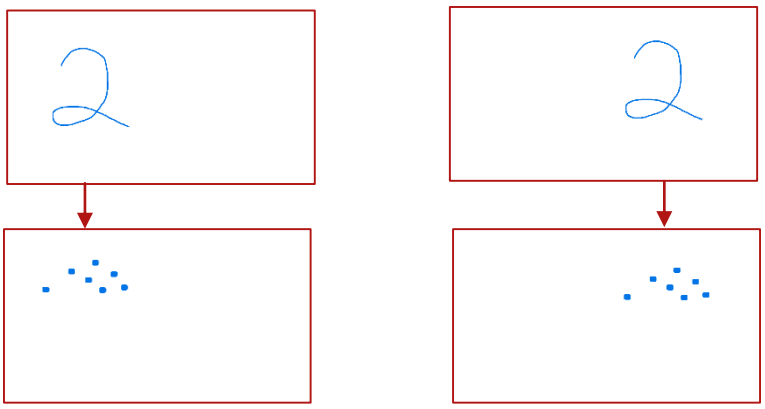
Βασισμένο στην ιδέα των replicated features: "Elementary feature detectors that are useful on one part of the image are likely to be useful across the entire image", Yann LeCun (1989). Η ιδέα αυτή δηλώνει ότι εάν έχουμε έναν feature detector που είναι χρήσιμος σε ένα σημείο, στην εικόνα, μπορεί να χρησιμεύσει και στην υπόλοιπη εικόνα. Επομένως, θέλουμε να δημιουργήσουμε "αντίγραφα" του ανιχνευτή αυτού, καλύπτοντας έτσι όλη την εικόνα.

Με την παραπάνω δήλωση, ο LeCun επισημαίνει τη σημασία και συνάμα την ιδιαιτερότητα των CNNs. Όπως αναλύσαμε πιο πάνω, ο κάθε κρυφός νευρώνας είναι υπεύθυνος για μια γειτονιά, στην οποία ψάχνει ένα χαρακτηριστικό. Παραλήφθηκε, όμως, το γεγονός ότι όλοι οι κρυφοί νευρώνες ενός κρυφού επιπέδου έχουν ακριβώς τα ίδια βάρη. Έστω ένας νευρώνας που βρίσκεται στη θέση j,k του κρυφού επιπέδου· η έξοδος του νευρώνα αυτού είναι:|

$$\sigma\left(b + \sum_{l=0}^{11}\sum_{m=0}^{7} w_{l,m} a_{j+l,k+m}\right)$$

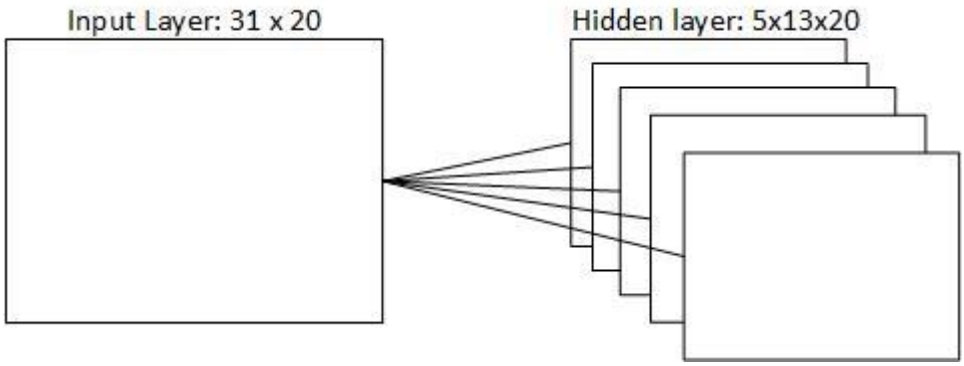
Εξίσωση 3.1: Έξοδος νευρώνα στο δίκτυο.

Εξυπακούεται, λοιπόν, ότι όλοι οι νευρώνες σε ένα κρυφό επίπεδο ψάχνουν για το ίδιο χαρακτηριστικό (feature) σε όλη την εικόνα, σε διαφορετικές, όμως, τοποθεσίες (Εικόνα 3.7). Υπεραπλουστεύοντας, μπορούμε να προσδιορίσουμε το feature ως ένα input pattern που προκαλεί την πυροδότηση ενός κρυφού νευρώνα, όταν αυτό αναγνωριστεί στην εικόνα (για παράδειγμα μια ακμή ή οποιοδήποτε άλλο σχήμα). Εάν υποθέσουμε ότι ένας κρυφός νευρώνας βρίσκει μια κάθετη ακμή, εντός του αντίστοιχου receptive field, τότε είναι πιθανόν η ακμή αυτή να βρίσκεται σε αρκετά σημεία στις εικόνες εισόδου.



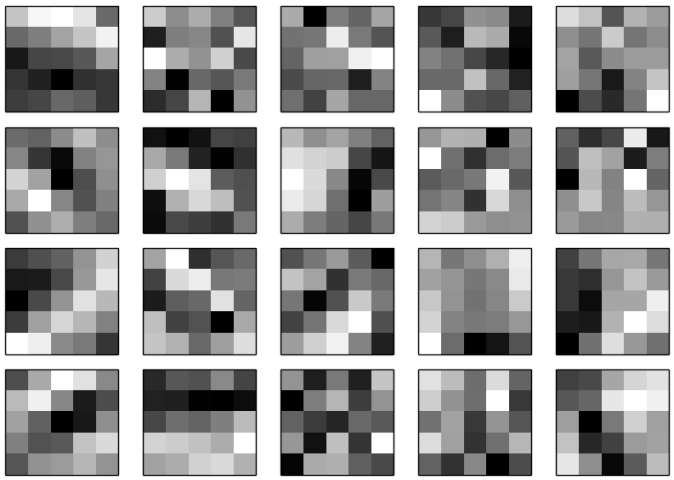
Εικόνα 3.7: Αναπαράσταση ενεργών νευρώνων στο κρυφό επίπεδο

Από το σχεδιάγραμμα διακρίνουμε ότι η γνώση είναι καθολική, μεταξύ των νευρώνων, αφού εντοπίζουν τον αριθμό σε όλη την επέκταση της εικόνας. Όλα τα παραπάνω οδήγησαν στην ονομασία των κρυφών επιπέδων σε *feature maps*, τα βάρη του δικτύου ως *shared weights* και τα biases ως *shared biases*.



Εικόνα 2.5: Αρχιτεκτονική κρυφού επιπέδου στο PSSP πρόβλημα.

Ένα αντικείμενο δεν αποτελείται μόνο από κάθετες γραμμές αλλά από μια σειρά χαρακτηριστικών που συμβάλουν στη δημιουργία του τελικού σχήματός του. Προκειμένου να αναγνωρίζουμε το ίδιο το αντικείμενο, και όχι μόνο ένα του χαρακτηριστικό, είναι αναγκαίο να προσθέσουμε έναν αριθμό από feature maps (Εικόνα 2.8). Η αρχική προσέγγιση του MNIST προβλήματος, με το LeNet-5 δίκτυο, έκανε χρήση 6 feature maps, ενώ στο, δικό μας, PSSP πρόβλημα μόλις 5. Με την πάροδο του χρόνου, σε ακαδημαϊκό επίπεδο γίνεται χρήση του LeNet-5 δικτύου με περισσότερα feature maps, συνήθως 20 μέχρι 40. Πρόσφατα ανακοινώθηκε από την Microsoft ότι κάνει χρήση των CNNs με 700 feature maps. [40].



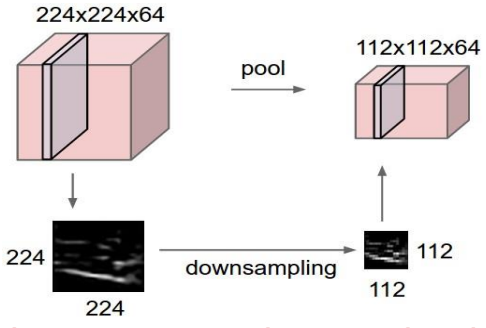
Εικόνα 2.6: Τα βάρη που έμαθε το δίκτυο για την αναγνώριση γραφικού χαρακτήρα.

Το γεγονός ότι το αποτέλεσμα των CNNs δεν επηρεάζεται από τη χωρική διάταξη της εικόνας, προσδίδει σε αυτά ένα από τα κυριότερα πλεονεκτήματα τους· το μειωμένο αριθμό προσαρμόσιμων παραμέτρων. Ας υποθέσουμε ότι έχουμε να επιλέξουμε ανάμεσα σε ένα fully-connected MLP και ένα CNN δίκτυο, για να λύσει το PSSP πρόβλημα. Εάν στο πρώτο κρυφό επίπεδο του MLP χρησιμοποιήσουμε 20 νευρώνες τότε ο αριθμός των παραμέτρων που θα αναβαθμίζονται είναι 31×20×20=12420 (συμπεριλαμβανομένου των biases). Αντίθετα σε ένα CNN δίκτυο θα έχουμε 20×13=260 βάρη για κάθε feature map, επομένως για 5 feature maps οι παράμετροι που αλλάζουν γίνονται 1305 (συμπεριλαμβανομένου των biases).

Στην εικόνα 3.8 εμφανίζονται 20 feature maps (MNIST), εκ των οποίων το καθένα είναι υπεύθυνο για έναν 5×5 kernel. Τα pixels αντιπροσωπεύουν τις διακυμάνσεις των βαρών, έτσι ώστε οι πιο ανοικτές αποχρώσεις του γκρίζου να αντιστοιχούν σε μικρές τιμές βαρών, ενώ οι σκούρες σε μεγάλες.

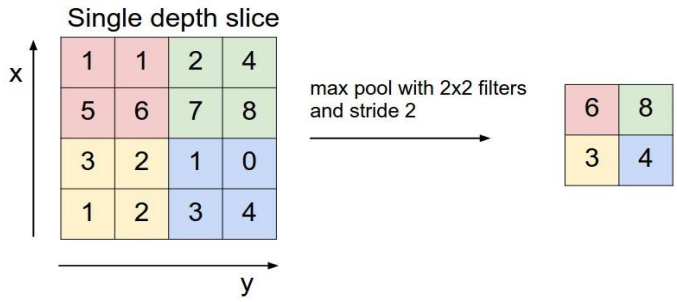
Pooling Layer

Εφόσον, στην εικόνα, έχει βρεθεί κάποιο χαρακτηριστικό, παύει να μας ενδιαφέρει πλέον η τοποθεσία του σε αυτήν. Τα pooling layers, που συνιστούν το τρίτο σημαντικότερο αρχιτεκτονικό στοιχείο των CNNs, αποκτούν σημασία λόγω της παραπάνω δήλωσης. Παραδείγματος χάρη, στο πρόβλημα αναγνώρισης γραφικού χαρακτήρα, εάν το δίκτυο γνωρίζει ότι στην πάνω αριστερή περιοχή, της εικόνας εισόδου, υπάρχει μια οριζόντια ακμή, στην πάνω δεξιά περιοχή βρίσκεται μια γωνία και στην κατώτερη περιοχή υπάρχουν κάθετες γραμμές, το δίκτυο μπορεί να κατηγοριοποιήσει την εικόνα ως τον αριθμό 7 [1]. Στο πλαίσιο αυτό κατανοούμε ότι η χρήση τους, μειώνει σημαντικά την ποσότητα των δεδομένων που πρέπει να επεξεργαστεί το δίκτυο αφού δειγματοληπτούν (Εικόνα 3.10) από το αμέσως προηγούμενο επίπεδο και δημιουργούν μια πιο συμπαγής εικόνα, όπως φαίνεται στην εικόνα 3.9.



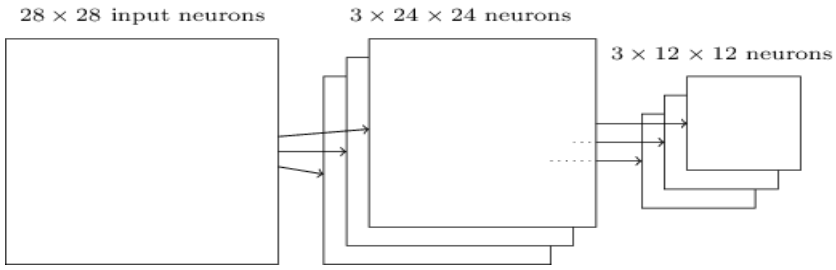
Εικόνα 3.7 [37]: Δημιουργία συμπαγούς εικόνας από

Η πιο συνηθισμένη πρακτική για εφαρμογή των pooling layers είναι η "περιγραφή" μίας μη επικαλυπτόμενης 2×2 περιοχής του hidden layer με τη χρήση της *max* συνάρτησης. Εξ' ορισμού είναι προφανές ότι το max-pooling λαμβάνει ως είσοδο 4 τιμές (της 2×2 περιοχής) και έχει ως έξοδο τη μέγιστη από αυτές (Εικόνα 3.11). Εκτός από τη max συνάρτηση, τα pooling layers, μπορούν να κάνουν χρήση και των συναρτήσεων *average* αλλά και *L2 – norm polling* (ρίζα του αθροίσματος των τετραγώνων της γειτονιάς). Στο LeNet-5 [1], έγινε χρήση του average pooling, αλλά με την πάροδο του χρόνου, πειράματα έδειξαν το καλύτερα αποτελέσματα δίνονται με τη χρήση της max.



Εικόνα 3.8 [37]: MAX Sampling

Η εφαρμογή των παραπάνω αφορούν μόνο ένα χαρακτηριστικό της εικόνας, και κατ' επέκταση ένα feature map. Αποτέλεσμα αυτού είναι η εφαρμογή ενός επιπέδου pooling για κάθε feature map, όπως φαίνεται παρακάτω (Εικόνα 3.12).

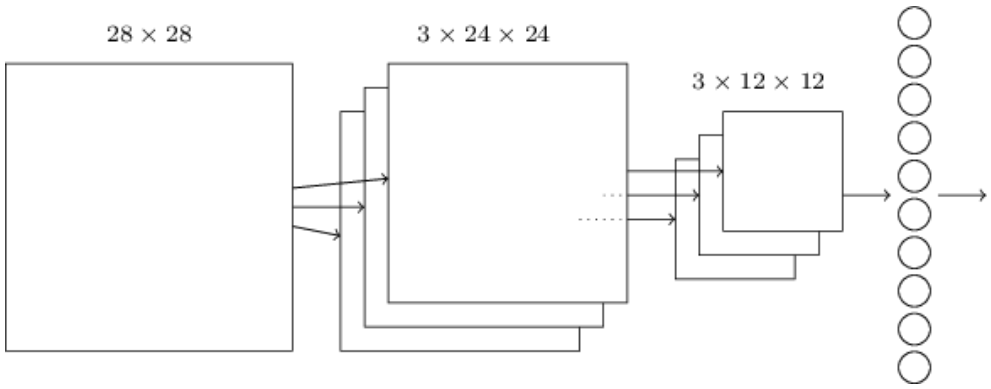


Εικόνα 3.12 [5]: Χρήση τριών feature maps και τριών pooling layers.

Εξάγοντας τα αποτελέσματα

Συνοψίζοντας, προκειμένου να κατηγοριοποιήσουμε την εικόνα που αναγνώρισε το δίκτυό μας, προσθέτουμε ένα fully-connected MLP επίπεδο (Εικόνα 3.13). Ενώνουμε, δηλαδή, όλους τους νευρώνες του MLP layer μαζί με τους νευρώνες του pooling layer.

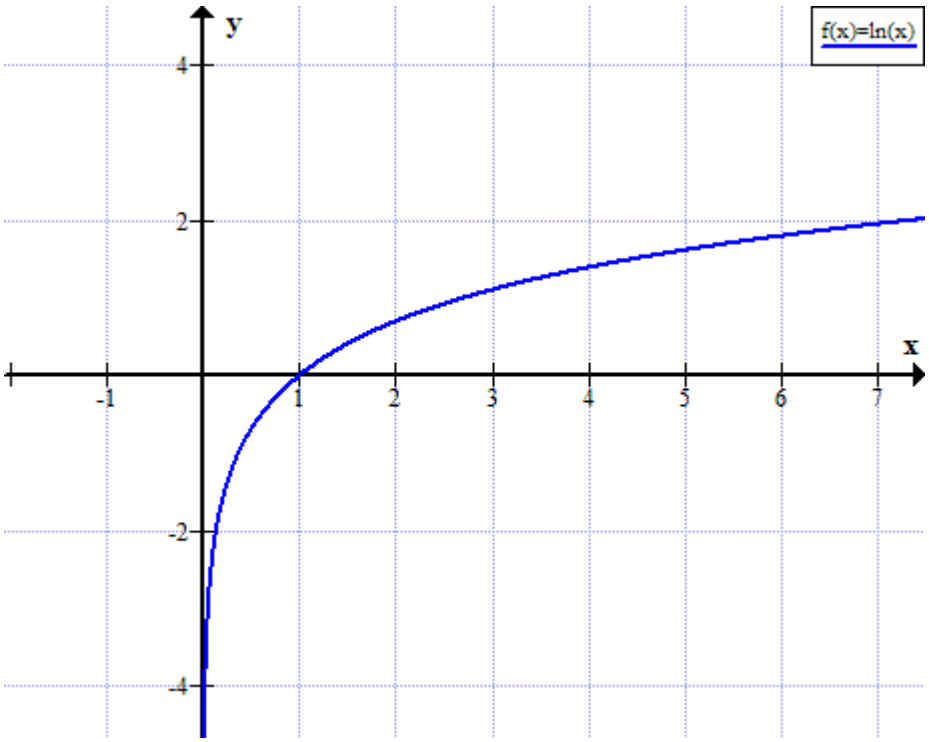
Ο κάθε νευρώνας του επιπέδου αυτού έχει τόσες εξόδους όσες και οι κατηγορίες του προβλήματος (10 κατηγορίες στο MNIST, ενώ μόλις 3 κατηγορίες στο PSSP).



Εικόνα 3.13 [5]: Τελική αρχιτεκτονική ενός ρηχού CNN.

Cost function – Συνάρτηση σφάλματος

Όπως περιγράφεται στο Υποκεφάλαιο 4.1.1, γίνεται χρήση ενός επιπέδου Softmax το οποίο με τη σειρά του περνά τις εισόδους των νευρώνων από την Softmax συνάρτηση. Οδηγούμαστε λογικά στην ερώτηση που αφορά τον υπολογισμό του σφάλματος του δικτύου όταν χρησιμοποιούμε μια τέτοια συνάρτηση.



Γραφική 3.1: Γραφική παράσταση της συναρτήσεως $f(x) = \ln(x)$. (Log- likelihood)

Σύμφωνα με την εξίσωση 4.1, η τιμή του νευρώνα a_j που δίνει τη μεγαλύτερη τιμή, αποτελεί και την έξοδο του δικτύου. Θέλουμε, επομένως, συνάρτηση του σφάλματος που να δίνει μικρό σφάλμα όταν η έξοδος ανταποκρίνεται στο επιθυμητό αποτέλεσμα και μεγάλο σφάλμα σε αντίθετη περίπτωση. Επιπλέον, έχοντας εις γνώση μας ότι το αποτέλεσμα της εξίσωσης 4.1 φράσσεται στο διάστημα [0,1] μπορούμε να παρατηρήσουμε ότι η συνάρτηση $f(x) = \ln(x)$ (Γραφική 3.1), δίνει μεγάλο σφάλμα όταν η έξοδος είναι κοντά στο μηδέν και μικρότερο σφάλμα όταν βρίσκεται κοντά στο ένα. Επομένως η συνάρτηση σφάλματος που χρησιμοποιεί το δίκτυο είναι η ακόλουθη (Εξίσωση 3.1):

$$Cost = -\ln a_j$$

Εξίσωση 3.1: Συνάρτηση σφάλματος.

Back propagation στα CNNs

Σε αυτό το σημείο, εύλογα, μας δημιουργείται η απορία για το πώς το δίκτυο μαθαίνει τα βάρη που χρειάζονται στα feature maps. Η απάντηση βρίσκεται στις θεμελιώδεις έννοιες της μηχανικής μάθησης, τη μέθοδο ανάστροφης μετάδοσης λάθους (back-propagation). Κάνοντας χρήση των shared weights προστίθεται στο δίκτυο ο πιο κάτω γραμμικός περιορισμός (linear constrain).

$$w_1 = w_2$$

Κατά τη διάρκεια εκπαίδευσης του δικτύου, θέλουμε αλλαγή βαρών τέτοια ώστε να ικανοποιείται το ακόλουθο:

$$\Delta w_1 = \Delta w_2$$

άρα:

$$\frac{\partial E}{\partial w_1} = \frac{\partial E}{\partial w_2}$$

θέτουμε τις νέες τιμές των βαρών:

$$w_1 = w_2 = \frac{\partial E}{\partial w_1} + \frac{\partial E}{\partial w_2}$$

Σημείωση: Μπορεί να χρησιμοποιηθεί και ο μέσος όρος των δυο αυτών τιμών. $w_1 = w_2 = \frac{\frac{\partial E}{\partial w_1} + \frac{\partial E}{\partial w_2}}{2}$.

Εν κατακλείδι, τα CNNs με τη χρήση της μεθόδου ανάστροφης μετάδοσης λάθους μπορεί να συνθέσει από μόνο του όλα τα feature maps.

Κεφάλαιο 4

Αποτελέσματα και Συμπεράσματα

4.1 Πειράματα

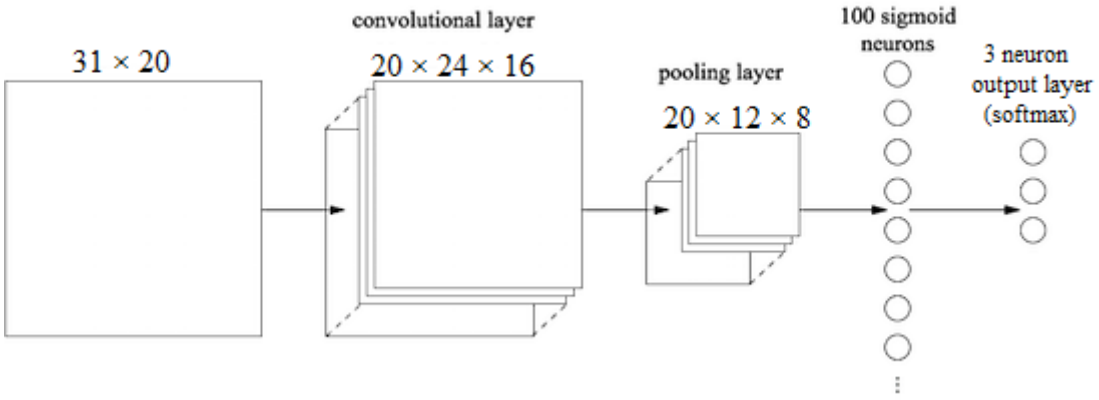
Όπως προαναφέρθηκε, όταν ο αριθμός των προσαρμόσιμων παραμέτρων (free parameters), που συνθέτουν ένα δίκτυο, αυξάνεται τότε ταυτόχρονα μεγαλώνει το και εύρος των πιθανών συμπεριφορών του. Ακόμη και στην περίπτωση όπου ένα δίκτυο δίνει καλά ποσοστά ακρίβειας, δεν μπορούμε να δηλώσουμε με σιγουριά ότι το μοντέλο είναι εκπαιδευμένο και μπορεί να γενικεύσει οτιδήποτε του δοθεί σαν είσοδος. Το μόνο που μπορούμε να θεωρήσουμε είναι ότι το δίκτυο αυτό είναι ικανό να εκπαιδευτεί και να γενικεύσει κάνοντας χρήση του συνόλου δεδομένων που έχουμε στη διάθεσή μας.

Θα πρέπει, λοιπόν αφού δημιουργήσαμε το dataset, να πράξουμε μια σειρά πειραμάτων ούτως ώστε να καταλήξουμε σε ένα σύνολο πιθανών τιμών για τις προσαρμόσιμες παραμέτρους (όπως learning rate, αριθμό των feature maps κ.τ.λ.). Η τακτική που θα ακολουθηθεί, για τη διεξαγωγή των πειραμάτων, είναι η σταδιακή βελτίωση του δικτύου προσθέτοντας μια σειρά από πρακτικές, που χρησιμοποιούνται στον κλάδο της μηχανικής μάθησης, και όταν το δίκτυο φτάσει σε "ώριμο" στάδιο, για γίνουν σε αυτό δοκιμές με διαφορετικά σύνολα δεδομένων. Σημειώνεται ότι το ποσοστό επιτυχίας που αναφέρεται στα παρακάτω πειράματα είναι βασισμένο στην Q3 μετρική (εξίσωση 1.1)

4.1.1 Εξελίσσοντας το δίκτυο

4.1.1.1 Τα ConvNets της βιβλιογραφίας

Ανακαλώντας τα παραπάνω, σχετικά με τον τρόπο λειτουργίας των ConvNets, η Εικόνα 4.1 παρουσιάζει την αρχιτεκτονική που θα χρησιμοποιήσουμε στην πρώτη προσπάθεια εκτέλεσης του δικτύου.



Εικόνα 4.1: Αρχιτεκτονική πρώτης προσπάθειας.

Softmax Function

Σύμφωνα με την Εικόνα 4.1, το επίπεδο εξόδου του δικτύου κάνει χρήση της συνάρτησης softmax, και όχι της λογιστικής σιγμοειδούς. Ένας softmax νευρώνας, αρχικά, χρησιμοποιεί την εξίσωση 1.1 ($u_j = \sum_{i=1}^{n} w_{ji} x_{ji} + b_j$) για να υπολογίσει το weighted input. Στη συνέχεια, χρησιμοποιεί το αποτέλεσμα της εξίσωσης 1.1 ως παράμετρο για τη softmax συνάρτηση, όπως φαίνεται στην εξίσωση 5.1.

$$a_j = \frac{e^{u_j}}{\sum_k e^{u_k}}$$

Εξίσωση 4.1: Softmax συνάρτηση, όπου j ο αριθμός του νευρώνα στο επίπεδο, u_j το αποτέλεσμα της εξίσωσης 1.1 και $\sum_k e^{u_k}$ το άθροισμα όλων των νευρώνων.

Το χαρακτηριστικό της συναρτήσεως, που μας οδηγεί στη χρήση της, είναι ότι το άθροισμα a_j όλων των νευρώνων του επιπέδου εξόδου, δίνει την τιμή 1 (Εξίσωση 5.1):

$$\sum_j a_j = \frac{e^{u_j}}{\sum_k e^{u_k}} = 1$$

Εξίσωση 4.2: Άθροισμα της εξόδου των νευρώνων στο επίπεδο εξόδου.

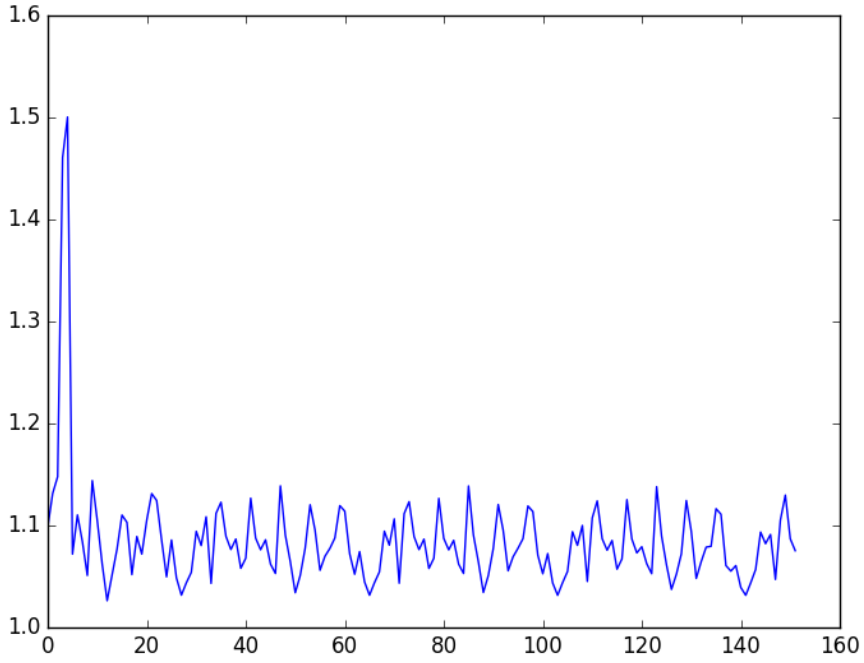
Λόγω της εξίσωσης 5.2, είναι αυτονόητο, ότι οι τιμές εξόδου των νευρώνων χαρακτηρίζουν την πιθανότητα ενός νευρώνα να είναι το επιθυμητό αποτέλεσμα (Εικόνα 4.3). Κατά την υλοποίηση, η έξοδος του δικτύου θεωρείται η μεγαλύτερη τιμή από όλους τους νευρώνες.

Εκτελώντας το δίκτυο με τις ακόλουθες τιμές των προσαρμόσιμων παραμέτρων (Πίνακας 4.1), πετυχαίνουμε ποσοστό ακρίβειας, στα δεδομένα ελέγχου 42.40%.

Παράμετρος	**Τιμή**
mini batch size	1000
input image size	31 × 20
activation function	sigmoid
Feature Detectors	
αριθμός των feature detectors	20
μέγεθος των feature detectors	8 × 5

Pooling Layer	
μέγεθος των pooling detectors	2 × 2
softmax neurons	100
learning rate	0.4
epochs	20

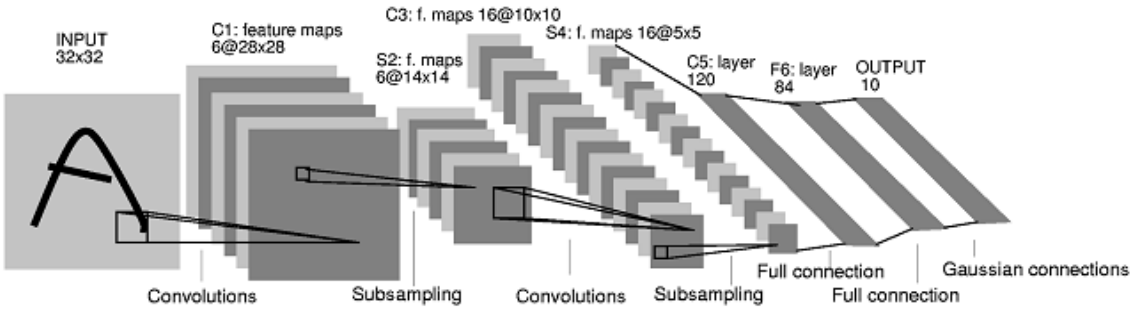
Πίνακας 4.1: Τιμές παραμέτρων του δικτύου.



Γραφική 4.1: Αποτελέσματα συναρτήσεως σφάλματος πρώτης προσπάθειας.

Χρήση 2ου ConvNet

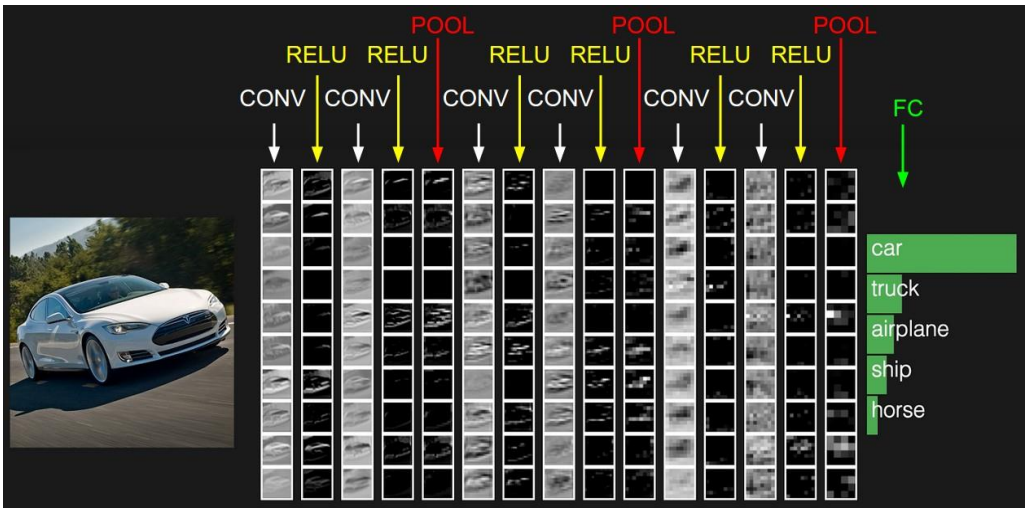
Μελετώντας το άρθρο του LeCun [1], συναντούμε μια πιο "βαθιά" προσέγγιση που χρησιμοποιήθηκε στο MNIST πρόβλημα (Εικόνα 4.2).



Εικόνα 4.2: Αρχιτεκτονική των LeNets [1].

Προκειμένου να καταλάβουμε την έννοια της εισαγωγής δευτέρου Convolutional επιπέδου, αρκεί να κάνουμε ανάκληση στη γενική ιδέα των ConvNets· ότι δηλαδή προσπαθούν να αναγνωρίσουν τη χωρική αναπαράσταση των δεδομένων, υποδιαιρώντας την πληροφορία σε χαμηλότερα επίπεδα.

Στην πρώτη προσπάθεια εκτέλεσης του δικτύου, ένας νευρώνας του pooling layer μετά την αυτό-οργάνωση, μπορούμε να πούμε, ότι εκφράζει – συμπιεσμένα - κάποιο χαρακτηριστικό σε συγκεκριμένη τοποθεσία, στο δεδομένο εισόδου. Η εισαγωγή ενός επιπλέον Convolutional επιπέδου, μπορεί να μας βοηθήσει στην περεταίρω αφαιρετική κατανόηση του δεδομένου αυτού. Είναι λογικό, ότι λόγω της ύπαρξης των 20 feature maps (Πίνακας 4.1) στο pooling layer, η είσοδος του επιπέδου θα λαμβάνει ως δεδομένο εισόδου τα 20 feature maps. Παράδειγμα των αυτό-οργανωμένων feature detectors φαίνεται στην Εικόνα 4.3.



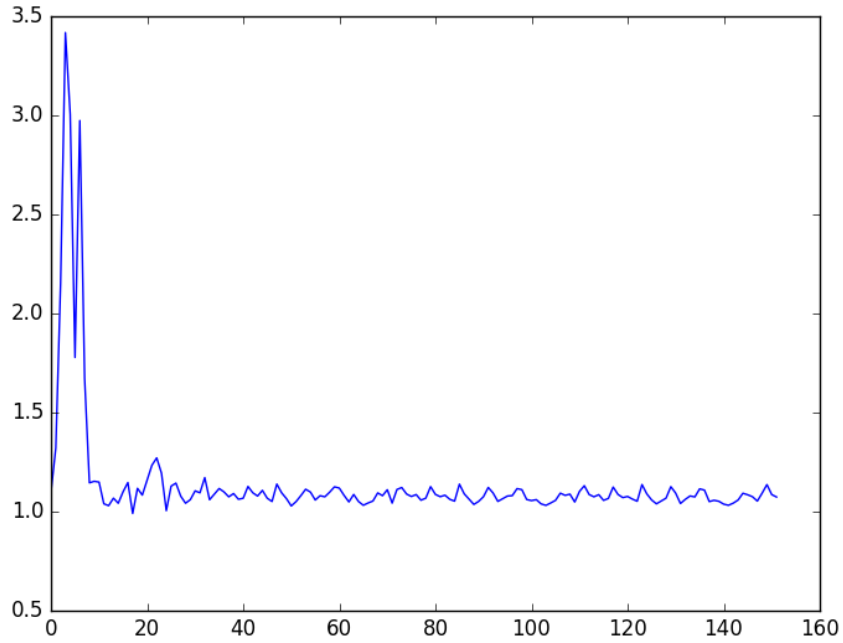
Εικόνα 4.3 [37]: Αποτέλεσμα αυτο-οργάνωσης των feature detectors.

Παράμετρος	Τιμή
mini batch size	1000
input image size	31 × 20
activation function	sigmoid
Conv Layer 1	
αριθμός των feature detectors	20
μέγεθος των feature detectors	8 × 5
μέγεθος των pooling detectors	2 × 2
Conv Layer 2	
αριθμός των feature detectors	40
μέγεθος των feature detectors	5 × 3

μέγεθος των pooling detectors	2 × 2
softmax neurons	100
learning rate	0.4
epochs	20

Πίνακας 4.2: Τιμές παραμέτρων του δικτύου κατά τη δεύτερη προσπάθεια.

Το παραπάνω μοντέλο παρά τις αλλαγές στην αρχιτεκτονική του είχε ως τελικό αποτέλεσμα, ποσοστό ακρίβειας 42.40 %. Υπάρχει, όμως μικρή διαφοροποίηση στη γραφική παράσταση της συναρτήσεως του σφάλματος (Γραφική 4.2), όπου τώρα το σφάλμα δεν έχει τόσο μεγάλες διακυμάνσεις, όπως στη Γραφική 4.1.



Γραφική 4.2: Αποτέλεσμα συναρτήσεως του σφάλματος (δεύτερη προσπάθεια).

Δυστυχώς, το δίκτυο με αυτή τη διάταξη συνεχίζει να δίνει αποτελέσματα 42.40 % καθώς επίσης η μορφή της γραφικής παράστασης της συναρτήσεως του σφάλματος διαφέρει από την επιθυμητή. Επομένως προτού συνεχίσουμε στην αλλαγή συνάρτησης ενεργοποίησης, θα ήταν καλό να εφαρμοστούν μερικές αλλαγές στο δίκτυο, στοχεύοντας σε καλύτερα αποτελέσματα.

L2 Regularization form

Το L2 regularization form (L2 κανονικοποίηση) βασίζεται στην πρόσθεση του όρου κανονικοποίησης (regularisation term) στην εξίσωση του σφάλματος (εξίσωση 5.3):

$$Cost = Cost + \frac{\lambda}{2n} \sum_{w} w^2$$

Εξίσωση 4.2: Συνάρτηση σφάλματος με L2 regularisation

w: τα συναπτικά βάρη του δικτύου
n: το μέγεθος του συνόλου δεδομένων εκπαίδευσης

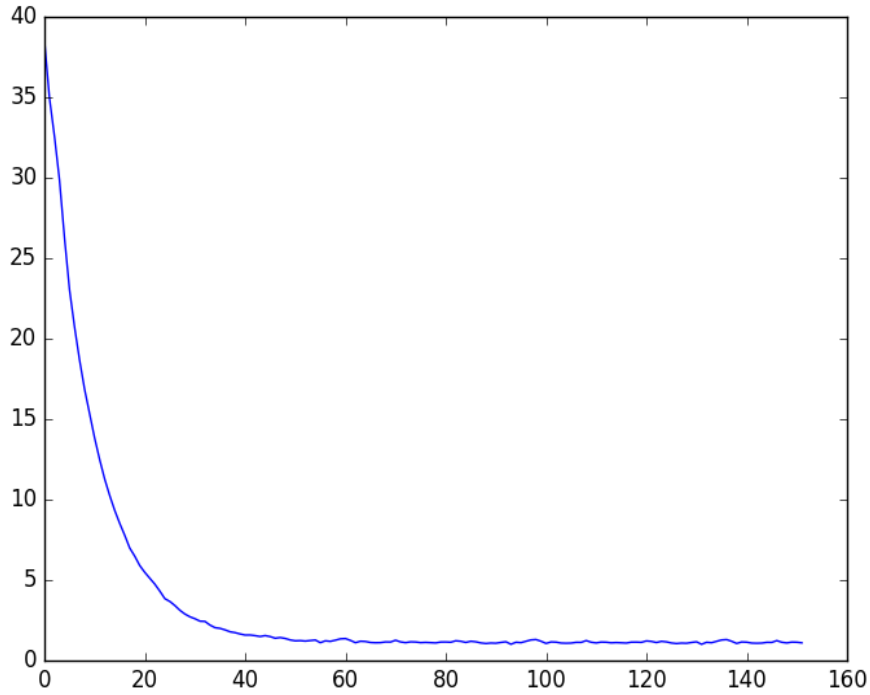
Το **λ** θεωρείται το "κλειδί" ενεργοποίησης του όρου. Ειδικότερα, μεγάλη τιμή στο λ επιτρέπει στο δίκτυο να μαθαίνει μικρές τιμές για τα βάρη, κατά τη μέθοδο κατάβασης κλίσης, ενώ όταν το **λ** είναι μικρό, μειώνεται περισσότερο η αποτίμηση της συναρτήσεως του σφάλματος [5].

Επομένως, λόγω του ότι στην περίπτωση του δικτύου μας, στην παρούσα φάση, δεν θέλουμε να μειώσουμε περεταίρω τη συνάρτηση του σφάλματος, θα προσπαθήσουμε να το εκπαιδεύσουμε με μεγάλη τιμή του όρου **λ.**

Παράμετρος	**Τιμή**
mini batch size	1000
input image size	31 × 20
activation function	sigmoid
learning rate	0.4
epochs	20
λ	5
Conv Layer 1	
αριθμός των feature detectors	20
μέγεθος των feature detectors	8 × 5
μέγεθος των pooling detectors	2 × 2
Conv Layer 2	
αριθμός των feature detectors	40
μέγεθος των feature detectors	5 × 3
μέγεθος των pooling detectors	2 × 2
softmax neurons	100

Πίνακας 4.3: Τιμές παραμέτρων του δικτύου με χρήση του L2 regularisation.

Ακόμη μια φορά, το δίκτυο παραμένει σταθερό, όπως τις προηγούμενες φορές, με ποσοστό επιτυχίας 42.40 %. Παρόλα αυτά, παρατηρούμε ότι η συνάρτηση σφάλματος (Γραφική 4.3) είναι πιο ρεαλιστική ως προς τον τρόπο που μαθαίνει το δίκτυο. Λόγω του ότι η τιμή του **λ** είναι αρκετά μεγάλη, η αρχική τιμή του σφάλματος είναι επίσης μεγάλη.



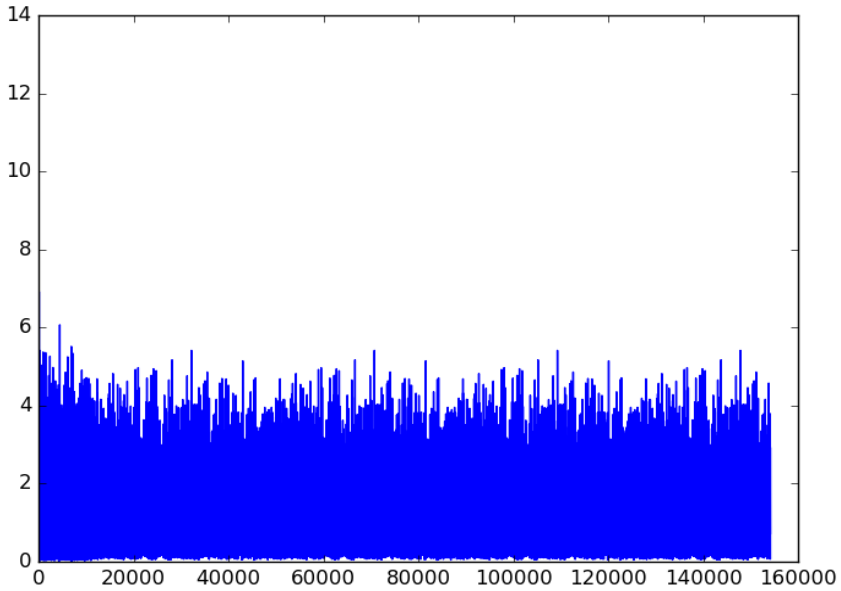
Γραφική 4.3: Γραφική παράσταση σφάλματος με χρήση του L2 reaularization.

Μέγεθος του mini – batch

Το μέγεθος του mini-batch για εκπαίδευση του δικτύου, σαφώς διαδραματίζει σημαντικό ρόλο στο ποσοστό επιτυχίας. Συγκεκριμένα, όταν το mini-batch έχει μέγεθος 1, τότε έχουμε online-training, δηλαδή με κάθε δεδομένο εισόδου, έχουμε αλλαγή των βαρών. Αυτό μπορεί να επηρεάσει αρνητικά το δίκτυο μας στην περίπτωση όπου τα δεδομένα εκπαίδευσης διαφέρουν κατά πολύ μεταξύ τους, πράγμα που οδηγεί στην ταλάντωση των συναπτικών βαρών και συνεπώς του σφάλματος.

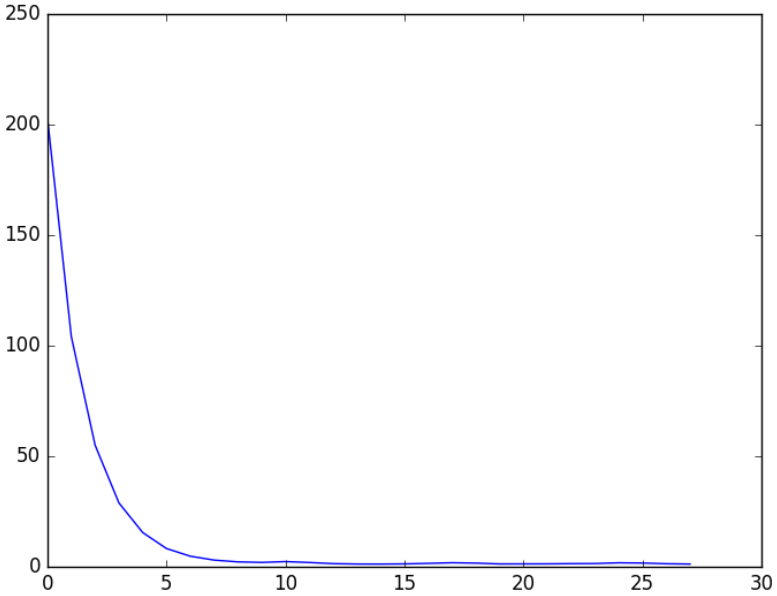
Για να ελέγξουμε εάν κάτι τέτοιο συμβαίνει στο δίκτυο μας, θα χρησιμοποιήσουμε την τοπολογία του δικτύου, όπως φαίνεται στον Πίνακα 4.3 με αλλαγή μόνο στο μέγεθος του mini-batch (όπου θα το θέσουμε 1).

Όντως, από τη γραφική παράσταση 4.4 είναι ξεκάθαρο ότι τα βάρη του δικτύου διαφοροποιούνται συνεχώς, δίνοντάς μας ποσοστό επιτυχίας 35.5%.



Γραφική 4.4: Συνάρτηση σφάλματος με online training.

Αντίθετα, όταν εκτελέσουμε το δίκτυο με την ίδια αρχιτεκτονική αλλά με αριθμό δεδομένων σε κάθε mini-batch ίσο με 3000, για πρώτη φορά το ποσοστό επιτυχίας μεγαλώνει πολύ λίγο και γίνεται 42.68% ενώ εάν αυξήσουμε τον αριθμός στις 5000



Γραφική 4.5: Συνάρτηση σφάλματος με 5000 δεδομένα για μέγεθος του mini-batch.

δεδομένα, λαμβάνουμε τη Γραφική παράσταση 4.5, πετυχαίνουμε το ποσοστό επιτυχίας 42.89%. Παρατηρούμε επίσης, ότι το σφάλμα στις πρώτες επαναλήψεις είναι πάρα πολύ μεγάλο, αλλά το δίκτυο μαθαίνει αρκετά γρήγορα.

Τέλος, όταν αυξήσουμε δραματικά τον αριθμό των δεδομένων σε ένα mini-batch, όπως για παράδειγμα 8000, η ακρίβεια του δικτύου πέφτει στο 34.51%, ποσοστό που είναι χειρότερο και από την online εκπαίδευση. Αυτό μπορεί να οφείλεται στην υπεργενίκευση της μεταβολής των βαρών με αποτέλεσμα το δίκτυο να μην προσαρμόζει σωστά τα συναπτικά του βάρη.

Learning Rate

Μια από τις θεμελιώδεις ανησυχίες της μηχανικής μάθησης, είναι αποφυγή του δικτύου να κολλήσει σε τοπικό ελάχιστο. Για το λόγο ότι τα πειράματα με αλλαγή του όρου μάθησης (learning rate), δεν έδειξαν καλύτερα αποτελέσματα, δεν θα παραθέσουμε γραφικές παραστάσεις.

4.1.1.2 Συνάρτηση ενεργοποίησης

"Η παραμετρική αλλαγή σε ένα αλγόριθμο αποτελεί περισσότερο τέχνη παρά επιστήμη" – Yann LeCun. Ένα από τα πολυσυζητημένα ζητήματα στο χώρο των νευρωνικών δικτύων, είναι η επιλογή της καταλληλότερης συνάρτησης ενεργοποίησης των νευρώνων.

Tanh function – Συνάρτηση υπερβολικής εφαπτομένης

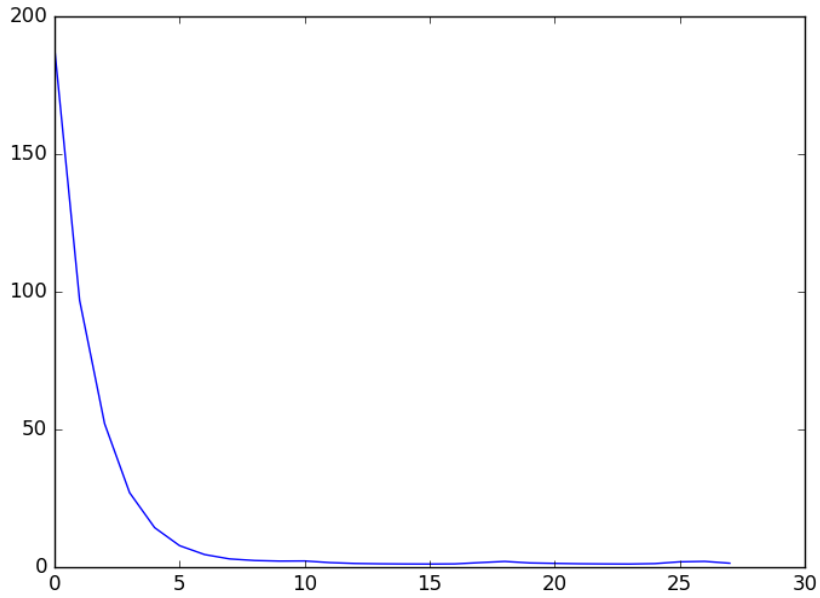
Υπάρχουν, επιστημονικά και μερικά εμπειρικά επιχειρήματα σχετικά με το ότι η συνάρτηση υπερβολικής εφαπτομένης (Tanh) αφήνει καλύτερα αποτελέσματα από ότι η λογιστική σιγμοειδής συνάρτηση. Σύμφωνα με τον Yann LeCun [1], η συνάρτηση υπερβολικής εφαπτομένης συγκλίνει γρηγορότερα λόγω του ότι το αποτέλεσμα της συνάρτησης είναι κεντρισμένο στο μηδέν. Αναλυτικότερα, η λογιστική σιγμοειδής συνάρτηση έχει πάντα θετικό αποτέλεσμα (output) στους νευρώνες, έτσι κατά την ανάστροφη μετάδοση λάθους όλα τα βάρη που φτάνουν σε ένα νευρώνα, θα αλλάξουν μόνο θετικά είτε μόνο αρνητικά. Αντίθετα, λόγω του ότι η συνάρτηση υπερβολικής εφαπτομένης έχει ως έξοδο [-1, 1], επιτρέπει στα βάρη που φτάνουν σε ένα νευρώνα να αλλάζουν ανάλογα θετικά ή αρνητικά, σε κάθε μετάδοση λάθους.

Παράμετρος	**Τιμή**
mini batch size	1000
input image size	31 × 20
activation function	tanh

learning rate	0.4
epochs	20
λ	5
Conv Layer 1	
αριθμός των feature detectors	20
μέγεθος των feature detectors	8 × 5
μέγεθος των pooling detectors	2 × 2
Conv Layer 2	
αριθμός των feature detectors	40
μέγεθος των feature detectors	5 × 3
μέγεθος των pooling detectors	2 × 2
softmax neurons	100

Πίνακας 4.3: Τιμές παραμέτρων του δικτύου, χρησιμοποιώντας τη Tanh συνάρτηση.

Η μόνη αλλαγή που παρατηρήθηκε, με τη συνάρτηση αυτή, είναι ότι το αρχικό σφάλμα του δικτύου είναι ελαφρώς πιο κάτω, από ότι με τη σιγμοειδή συνάρτηση· αλλαγή από την οποία δεν μπορούμε να εξάγουμε κάποιο συμπέρασμα και μπορούμε μόνο να τη θεωρήσουμε ως παρατήρηση.



Γραφική 4.6: Συνάρτηση σφάλματος με συνάρτηση ενεργοποίησης τη Tanh.

ReLU – Rectified Linear Units function

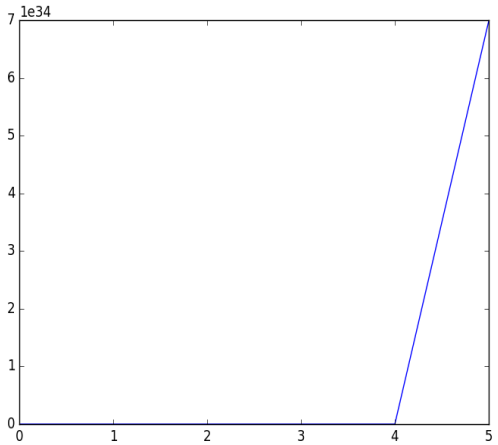
Πέραν από τις προαναφερθείσες συναρτήσεις, η ReLU αποτελεί ακόμη μια ευρέως διαδεδομένη συνάρτηση στον τομέα της αναγνώρισης αντικειμένων στην ψηφιακή εικόνα. Το ίδιο αμφιλεγόμενα, στην επιστημονική κοινότητα, είναι τα θετικά χαρακτηρίστηκα την συνάρτησης αυτής.

Ο Alex Krizhevsky [6], ισχυρίζεται ότι η ReLU συγκλίνει ακόμη γρηγορότερα από τις δυο προαναφερθείσες συναρτήσεις κάτι που οφείλεται στο γραμμικό, non-saturating form. Επιπλέον, η ReLU υπερτερεί έναντι των άλλων συναρτήσεων σχετικά με την υπολογιστική ισχύ που καταναλώνει. Αντίθετα, υποστηρίζεται ότι κατά την εκπαίδευση, με χρήση της ReLU, μπορεί να γίνει αναβάθμιση βαρών, τέτοια ώστε να προκαλέσει "θάνατο" σε περισσότερους νευρώνες, όταν το learning rate είναι αρκετά μεγάλο.

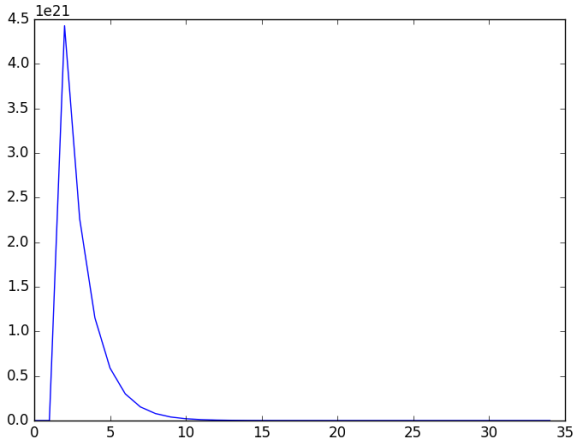
Παράμετρος	**Τιμή**
mini batch size	5000
input image size	31 × 20
activation function	ReLU
learning rate	0.4
epochs	20
λ	5
Conv Layer 1	
αριθμός των feature detectors	20
μέγεθος των feature detectors	8 × 5
μέγεθος των pooling detectors	2 × 2
Conv Layer 2	
αριθμός των feature detectors	40
μέγεθος των feature detectors	5 × 3
μέγεθος των pooling detectors	2 × 2
softmax neurons	100

Πίνακας 4.4: Τιμές παραμέτρων του δικτύου, χρησιμοποιώντας την ReLU συνάρτηση.

Λαμβάνοντας υπόψιν τις παραπάνω γραφικές παραστάσεις, παρατηρούμε ότι το δίκτυο εκπαιδεύεται πολύ γρήγορα, και όταν ο αριθμός των επαναλήψεων είναι αρκετά μεγάλος, δεν παρουσιάζει κάποια βελτίωση στα αποτελέσματα. Επομένως κατά την εκτέλεση αυτή, μειώσαμε τον αριθμό των εποχών σε μόνο 10.



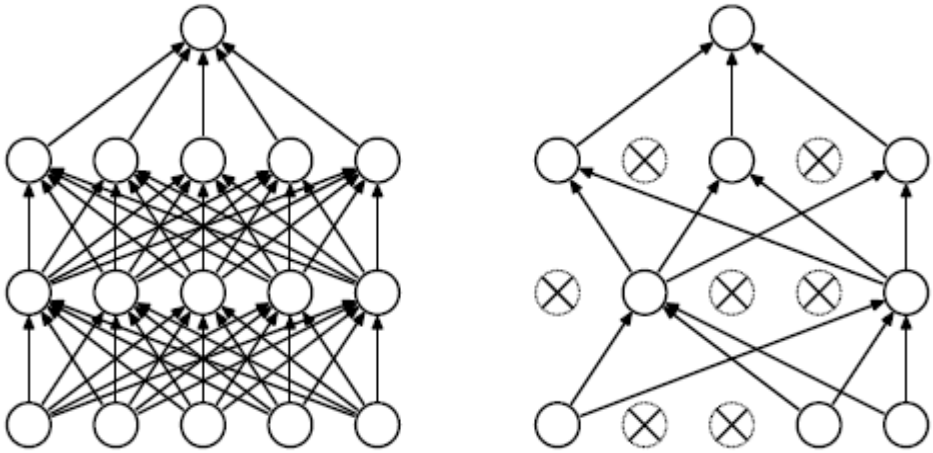
Γραφική 4.7: Συνάρτηση σφάλματος με ReLU.



Γραφική 4.8: Συνάρτηση σφάλματος ReLU με λ=0.4.

Και πάλι, τα αποτελέσματα δεν βελτιώνονται (ποσοστό επιτυχίας 42.89%), καθώς επίσης η γραφική παράσταση 4.7 μας δημιουργεί τον προβληματισμό για τη χρήση της ReLU. Έτσι, ξανατρέξαμε το δίκτυο θέτοντας τον όρο **λ** ίσο με 0.4, για να μειώσουμε το σφάλμα της συνάρτησης (Γραφική 4.8).

Dropout



(α) Πριν την εφαρμογή του dropout (β) Μετά την εφαρμογή του dropout

Εικόνα 4.4: Αρχιτεκτονική του δικτύου με τη χρήση του dropout.

Για να διασαφηνιστεί η Εικόνα 4.4, ας θεωρήσουμε ότι έχουμε πέντε ανεξάρτητα δίκτυα αναγνώρισης γραφικού χαρακτήρα και όλα προσπαθούν να αναγνωρίσουν τον αριθμό "2", αλλά μόνο τα τρία από αυτά τα δίκτυα καταφέρνουν να κατηγοριοποιήσουν σωστά το δεδομένο εισόδου. Ένας καλός τρόπος να εξάγουμε σωστά αποτελέσματα, συνδυάζοντας τα πέντε δίκτυα για να δημιουργήσουμε ένα πιο σύνθετο, είναι να υπολογίζεται ο μέσος όρος των δικτύων για κάθε δεδομένο εισόδου. Το σημαντικό μειονέκτημα, αυτού του τρόπου σύνθεσης δικτύων προκαλεί σοβαρή καθυστέρηση στην εκπαίδευση του δικτύου. Ο τρόπος για να πετύχουμε τα θετικά αποτελέσματα, της παραπάνω πρακτικής είναι η χρήση του dropout. [3][5].

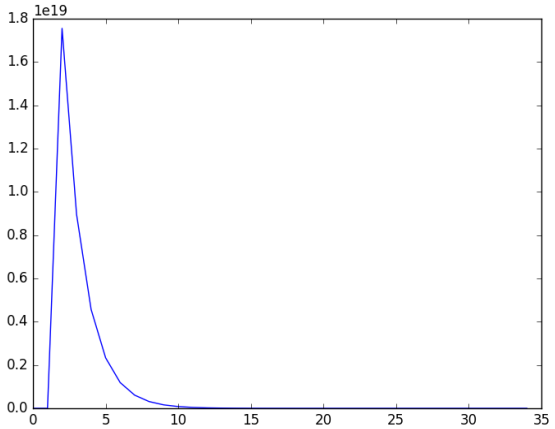
Η ετυμολογία της dropout τεχνικής, την ορίζει η αγγλική γλώσσα, αφού πηγάζει από τις λέξεις drop + out, που σημαίνει ότι κάποιοι νευρώνες αγνοούνται [2][3]. Η τεχνική αυτή δηλώνει ότι κατά τη διάρκεια ενός mini-batch (υποσύνολο δεδομένων) απενεργοποιούνται, με ένα τυχαίο τρόπο κατά τη διάρκεια της εκπαίδευσης του

συστήματος, οι μισοί νευρώνες, οι οποίοι απέχουν τόσο από την εξαγωγή του αποτελέσματος Y^p, όσο και από την αναβάθμιση των βαρών τους. Οι αναβαθμίσεις των βαρών εκτελούνται κανονικά, στους ενεργούς νευρώνες, μέχρι να τελειώσει ο αριθμός επαναλήψεων της εκπαίδευσης των δεδομένων. Η εκμάθηση νέου mini-batch έχει ως αποτέλεσμα την απενεργοποίηση άλλου 50% των κρυφών νευρώνων [5].

Παράμετρος	**Τιμή**
mini batch size	5000
input image size	31 × 20
activation function	ReLU
learning rate	0.4
epochs	20
λ	0.4
dropout	50 %
Conv Layer 1	
αριθμός των feature detectors	20
μέγεθος των feature detectors	8 × 5
μέγεθος των pooling detectors	2 × 2
Conv Layer 2	
αριθμός των feature detectors	40
μέγεθος των feature detectors	5 × 3
μέγεθος των pooling detectors	2 × 2
softmax neurons	100

Πίνακας 4.5: Τιμές παραμέτρων του δικτύου, χρησιμοποιώντας την dropout τεχνική.

Έχει γίνει γνωστό, πλέον, ότι το δίκτυο αδυνατεί να μάθει αφού το ποσοστό επιτυχίας που δίνει χρησιμοποιώντας το dropout είναι και πάλι 42.89%. Συγκριτικά με τη γραφική παράσταση 4.8, διακρίνεται αρκετή μείωση στο σφάλμα, κατά τις πρώτες επαναλήψεις.



Γραφική 4.9: Συνάρτηση σφάλματος με dropout.

4.1.2 Αλλαγές βασισμένες στα δεδομένα εισόδου

Είναι ξεκάθαρο, ότι παρά τη δύναμη του δικτύου να υπεργενικεύει τα δεδομένα εισόδου, στο παρόν πρόβλημα, οι ιδιότητες αυτές δεν καρποφόρησαν. Ξεκινούμε λοιπόν να στρέφουμε το βλέμμα προς το σύνολο δεδομένων και να ασχοληθούμε με την αιτία που το δίκτυο αδυνατεί να εκπαιδευτεί με το σύνολο δεδομένων που έχουμε στην κατοχή μας.

Μέγεθος φίλτρου

0	5	0	0	0	0	3	3	9	3	4	0	0	0	9	3	14	34	9	2
6	5	17	5	3	0	0	1	13	0	0	6	0	14	7	8	3	0	6	3
6	1	3	0	0	0	0	1	1	0	1	0	0	0	0	3	0	1	2	80
6	33	5	8	2	13	0	1	3	0	6	0	0	0	13	0	0	4	3	0
7	1	3	0	0	0	0	4	11	25	6	2	0	4	10	12	6	2	3	3
0	0	0	0	0	0	15	2	13	0	9	0	0	0	35	9	3	0	12	0
1	39	15	19	3	0	2	4	11	2	0	0	0	0	0	0	0	0	2	0
0	22	2	0	0	0	2	0	5	1	1	3	0	0	13	7	8	26	4	5
0	2	0	1	0	0	0	2	5	15	7	1	0	0	23	29	5	4	3	4
5	10	4	0	11	3	66	0	0	0	0	0	0	0	0	0	0	0	0	0
2	3	0	2	1	1	0	1	7	0	10	1	0	0	13	14	21	14	1	6
4	0	4	0	0	0	0	1	2	25	18	7	1	6	1	2	4	12	7	5
7	36	15	4	2	0	5	0	4	1	3	6	1	0	0	4	7	0	1	3
18	6	54	6	4	0	0	0	6	0	0	4	0	0	1	0	0	0	0	0
1	3	1	0	2	0	0	1	6	0	7	4	0	6	22	14	13	12	7	2
0	**3**	**0**	**1**	**0**	**0**	**1**	**0**	**8**	**1**	**9**	**12**	**0**	**0**	**7**	**26**	**6**	**21**	**3**	**0**
30	2	7	2	0	1	10	0	38	0	1	0	2	3	0	2	0	1	0	0
0	2	0	0	0	0	0	33	42	0	21	0	1	0	0	0	0	0	0	0
1	4	0	1	0	0	0	2	5	2	2	4	0	0	16	12	29	11	6	4
1	0	3	0	0	0	0	0	8	0	1	5	0	1	21	39	9	12	0	0
0	8	0	1	5	0	46	0	1	0	2	0	5	17	0	8	5	0	2	0
0	0	0	0	0	0	0	35	3	0	3	0	17	3	3	1	4	6	14	12
36	27	21	14	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	9	4	3	0	0	0	0	1	12	2	69
1	2	0	0	0	5	5	0	4	61	2	1	0	4	2	2	2	10	0	1
1	11	0	0	0	0	3	7	43	0	6	0	0	1	5	3	2	3	13	1
16	67	3	0	7	3	0	0	2	0	0	0	0	0	0	0	0	0	3	0
25	19	55	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	11	1	5	1	0	5	2	41	0	2	0	2	2	11	13	3	1	1	0
0	0	0	0	0	0	0	21	75	0	3	0	0	0	1	0	0	0	0	0
55	12	28	3	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0

πρώτο feature detector

δεύτερο feature detector

τρίτο feature detector

σειρά του target output

Πίνακας 4.6: Στιγμιότυπο τιμών της εικόνας για το αρχείο 18743.

Μια από τις ανησυχίες που μπορεί να μας δημιουργηθούν, προσέχοντας τον Πίνακα 4.6, είναι το μέγεθος του φίλτρου που ορίσαμε για την εικόνα. Για παράδειγμα, ορίζοντας ένα μεγάλο φίλτρο, που περιέχει αρκετά μηδενικά οδηγεί τη συνάρτηση ενεργοποίησης στην εξαγωγή μιας αρκετά μικρής τιμής, ο οποία με τη σειρά της περνά στο pooling layer με αποτέλεσμα εάν το επόμενο φίλτρο δώσει μεγαλύτερη τιμή να επιλεγεί αυτό αντί του πρώτου. Προσπαθούμε τώρα, να αλλάξουμε το μέγεθος του φίλτρου, ευελπιστώντας στη βελτίωση των αποτελεσμάτων.

Παράμετρος	**Τιμή**
mini batch size	5000
input image size	31 × 20
activation function	ReLU
learning rate	0.4
epochs	20
λ	0.4
dropout	50 %
Conv Layer 1	
αριθμός των feature detectors	20
μέγεθος των feature detectors	4 × 3
μέγεθος των pooling detectors	2 × 2
Conv Layer 2	
αριθμός των feature detectors	40
μέγεθος των feature detectors	3 × 2
μέγεθος των pooling detectors	2 × 2
softmax neurons	100

Πίνακας 4.6: Τιμές παραμέτρων του δικτύου, χρησιμοποιώντας την μικρά feature maps.

Δεν θα παραθέσω γραφική παράσταση για το πείραμα αυτό, αφού καμία αλλαγή δεν παρατηρήθηκε στην εκτέλεση του δικτύου.

Μια διαφορετική προσέγγιση για μέγεθος του φίλτρου είναι ο ορισμός του μεγέθους του ίσο με μια γραμμή ούτως ώστε να γενικεύει την κάθε γραμμή των MSA files, που και πάλι δεν έδωσε καλύτερα αποτελέσματα.

Μέγεθος αρχείων

Η υπερπήδηση της χωρικής τοποθέτησης των αντικειμένων στην εικόνα, είναι ακόμη μια ιδιότητα των ConvNets. Ιδιότητα που μάλλον στρέφεται εναντίων μας, κατά την εκτέλεση του δικτύου.

Έστω μια εικόνα διαστάσεων 31 × 20, όπως το μέγεθος που χρησιμοποιούμε στο πρόβλημά μας, αναπαριστά μια γάτα και ένα σκουλήκι (Εικόνα 4.5). Σύμφωνα με τον τρόπο που προσδιορίζουμε στις εικόνες το επιθυμητό αποτέλεσμα, επειδή η γάτα βρίσκεται στο κέντρο της εικόνας, ορίζεται αυτόματα το επιθυμητό αποτέλεσμα.

Το πρόβλημα παρουσιάζεται με την Εικόνα 4.6, όπου η γάτα έχει προχωρήσει πιο πάνω στην εικόνα και στο κέντρο βρίσκεται το σκουληκάκι. Το δίκτυο αναγνωρίζει τόσο τη γάτα όσο και το σκουλήκι αλλά η γάτα έχει μεγαλύτερη βαρύτητα, λόγω του μεγέθους της, επομένως εξάγεται αυτή ως το τελικό αποτέλεσμα του δικτύου. Αντίθετα, εμείς υποδηλώνουμε στο δίκτυο ότι έκανε λάθος και πως το επιθυμητό αποτέλεσμα είναι στο σκουλήκι.

Εικόνα 4.5: Τυχαία εικόνα γάτας και σκουληκιού με επιθυμητό αποτέλεσμα τη γάτα.

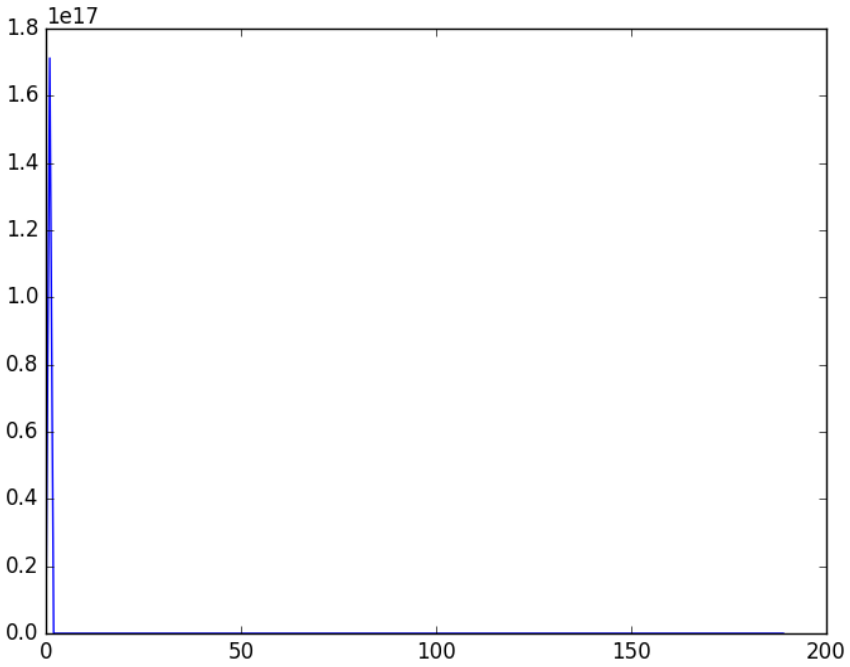
Εικόνα 4.6: Τυχαία εικόνα γάτας και σκουληκιού με επιθυμητό αποτέλεσμα το σκουλήκι.

Δημιουργήσαμε, λοιπόν, αρχεία μεγέθους 17 × 20, που και πάλι δεν λάβαμε καλύτερα αποτελέσματα.

Μέγεθος συνόλου δεδομένων

Όπως είναι γνωστό, τα ConvNets ανήκουν στην οικογένεια των Deep Networks πράγμα που μας υποδηλώνει την ανάγκη για ύπαρξη μεγάλου συνόλου δεδομένων, έτσι ώστε το δίκτυο να μπορεί να αποδώσει καλύτερα. Η αύξηση του συνόλου δεδομένων εφαρμόστηκε και στο MNIST πρόβλημα, χρησιμοποιώντας κώδικα από αποθετήριο του github, τετραπλασιάζοντας τον αριθμό των δεδομένων του δικτύου. Επί της ουσίας, η επέκταση των δεδομένων προσθέτει μια σειρά από μηδενικά σε όλες τις πρώτες και τελευταίες γραμμές καθώς επίσης στις πρώτες και τελευταίες στήλες του συνόλου δεδομένων (77092 * 4 = 308368 δεδομένα).

Για ακόμη μια φορά, το ποσοστό επιτυχίας παραμένει 42.89%.

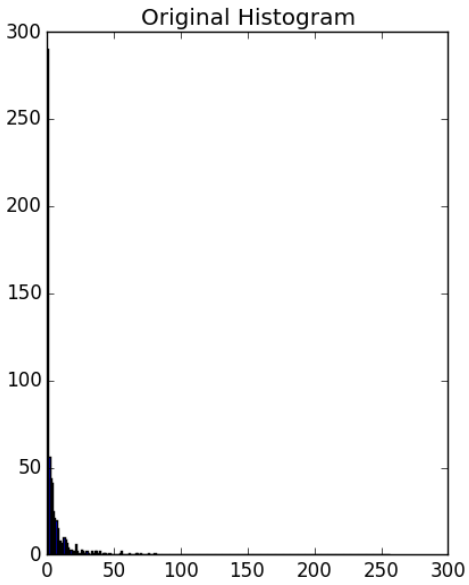


Γραφική 4.10: Χρήση μεγάλου συνόλου δεδομένων.

4.1.3 Ψηφιακή επεξεργασία εικόνας

Συνεπακόλουθο των παραπάνω, είναι η στροφή προς τη δειγματοληψία εικόνων των πρωτεϊνών με σκοπό να τροποποιήσουμε τη μορφή τους, με τρόπο που να μπορεί να την εκμεταλλευτεί το δίκτυο. Σημειώνεται ότι η οποιαδήποτε μετατροπή των εικόνων των πρωτεϊνών το πιο πιθανό διαφωνεί με τους βιολογικούς περιορισμούς που βασίστηκαν τα αρχεία MSA.

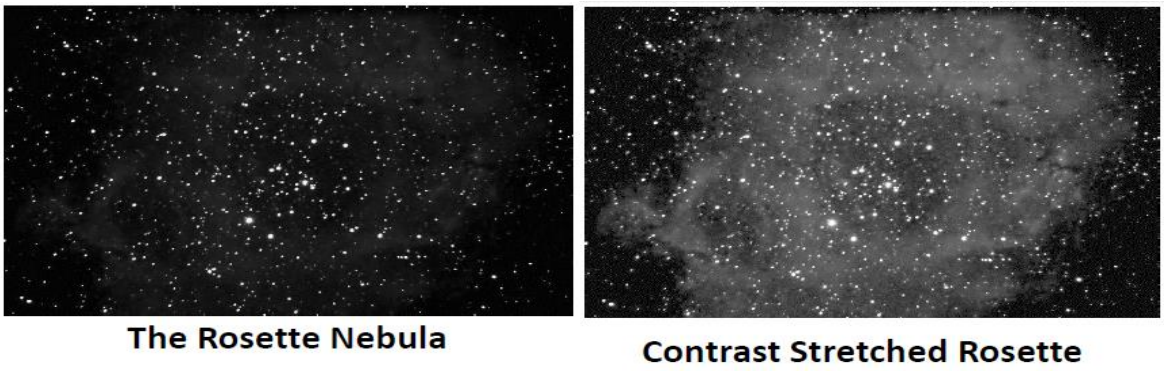
Εικόνα 4.7: Στιγμιότυπο του αρχείου 18743.



Εικόνα 4.8: Ιστόγραμμα του αρχείου 18743.

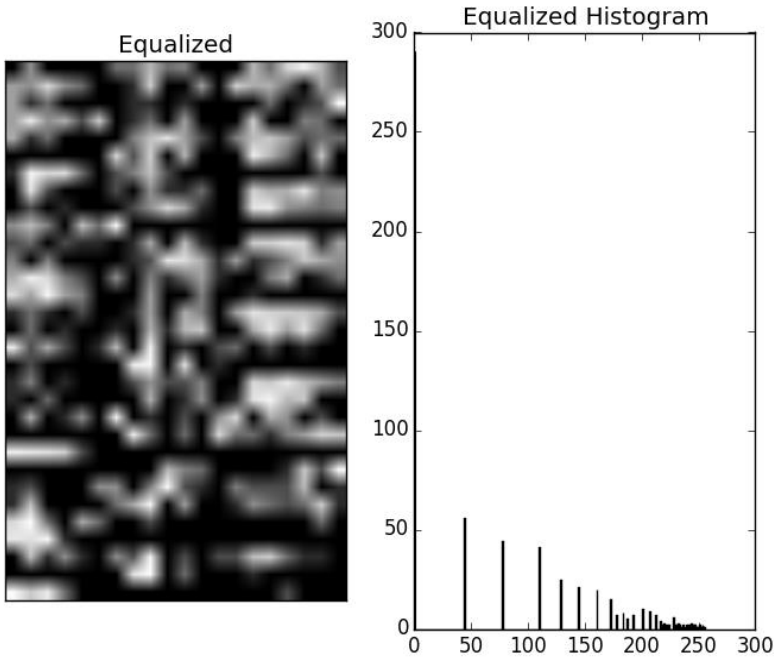
Στην Εικόνα 4.7 παρουσιάζεται ένα τυχαίο οπτικοποιημένο στιγμιότυπο από το σύνολο δεδομένων μας και αντιλαμβανόμαστε ότι η εικόνα έχει στην πλειονότητά της την τιμή 0, χαρακτηριστικό που η Python το ερμηνεύει ως μαύρο. Στην επιστήμη της ψηφιακής επεξεργασίας εικόνας, για τη μελέτη της κατανομής των τιμών στα pixels, χρησιμοποιείται η εμφάνιση μιας γραφικής παράστασης που ονομάζεται ιστόγραμμα. Το ιστόγραμμα του στιγμιοτύπου 18743 (Εικόνα 4.8), καθρεφτίζει το γεγονός ότι τα στιγμιότυπα εικόνων σχηματίστηκαν με τα MSA files και οι τιμές που χρησιμοποιήθηκαν ανήκουν στο διάστημα [0-100]. Αρχίζουμε, λοιπόν, να υποψιαζόμαστε ότι στο δίκτυο δίνονται πολλές παραπλήσιες τιμές, στοιχείο που μπορεί να προκαλεί την μειωμένη απόδοση του δικτύου μας.

Τέντωμα Αντίθεσης – Equalise Histogram



Εικόνα 4.9: Rosette Nebula με τέντωμα αντίθεσης [38].

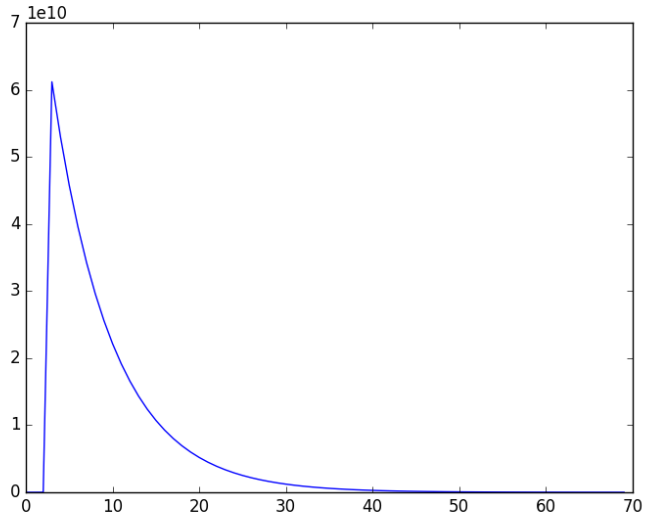
Η Εικόνα 4.9 γίνεται αφορμή για την εξαγωγή της ισοσταθμισμένης εικόνας του τυχαίου δείγματος 18743 (Εικόνα 4.10), το ιστόγραμμα της οποίας μας παρουσιάζει την κατανομή των τιμών στα pixels της.



Εικόνα 4.10: Ισοσταθμισμένη εικόνα του αρχείου 18743.

Παράμετρος	Τιμή
dataset	equalized images
mini batch size	5000
input image size	31 × 20
activation function	ReLU

learning rate	0.4
epochs	20
λ	5
dropout	50 %
Conv Layer 1	
αριθμός των feature detectors	20
μέγεθος των feature detectors	4 × 3
μέγεθος των pooling detectors	2 × 2
Conv Layer 2	
αριθμός των feature detectors	40
μέγεθος των feature detectors	3 × 2
μέγεθος των pooling detectors	2 × 2
softmax neurons	100



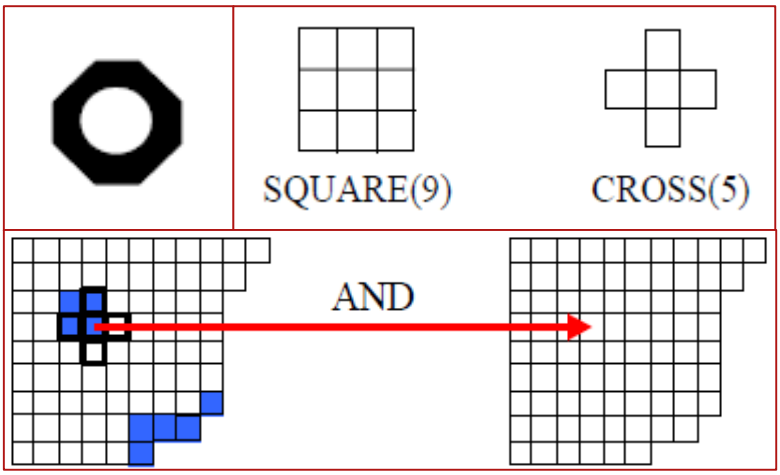
Γραφική 4.11: Χρήση ισοσταθμισμένου συνόλου δεδομένων.

Η εκτέλεση του δικτύου δεν κατάφερε να προσδώσει καλύτερα αποτελέσματα από τις προηγούμενες προσπάθειες, αφού και πάλι το ποσοστό επιτυχίας είναι 42.89%.

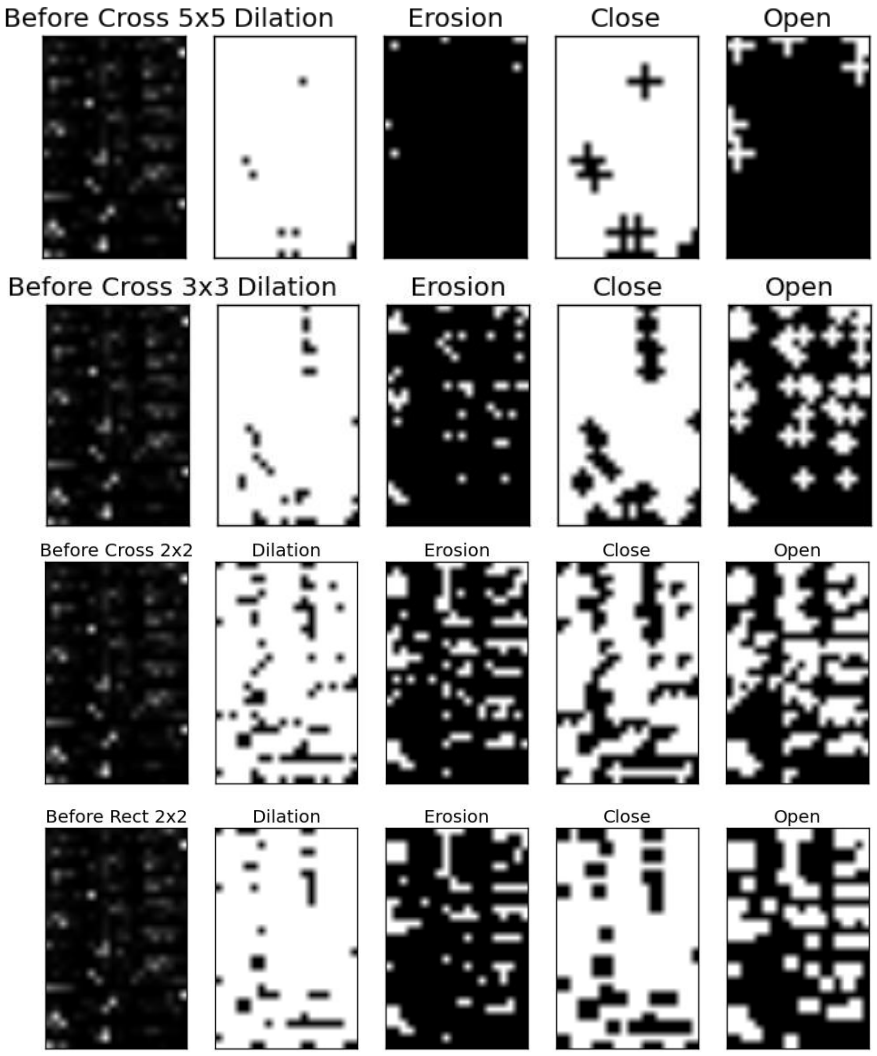
Δυαδικοί τελεστές

Όπως προαναφέρθηκε στο κεφάλαιο 3, εικόνες των οποίων οι τιμές στα εικονοστοιχεία τους αποτελούνται μόνο από "0" ή "1" ονομάζονται δυαδικές εικόνες (Εικόνα 4.11). Οι δυαδικοί τελεστές εφαρμόζονται σε δυαδικές εικόνες. Αποτελούνται από έναν πυρήνα, όπως στην Εικόνα 4.11 (π.χ. τετράγωνο, κύκλο) και διαπερνούν τη δυαδική εικόνα

εφαρμόζοντας τους λογικούς τελεστές AND (EROSION), OR (DILATION), OPEN (DILATION & EROSION), CLOSE (EROSION & DILATION) [36].

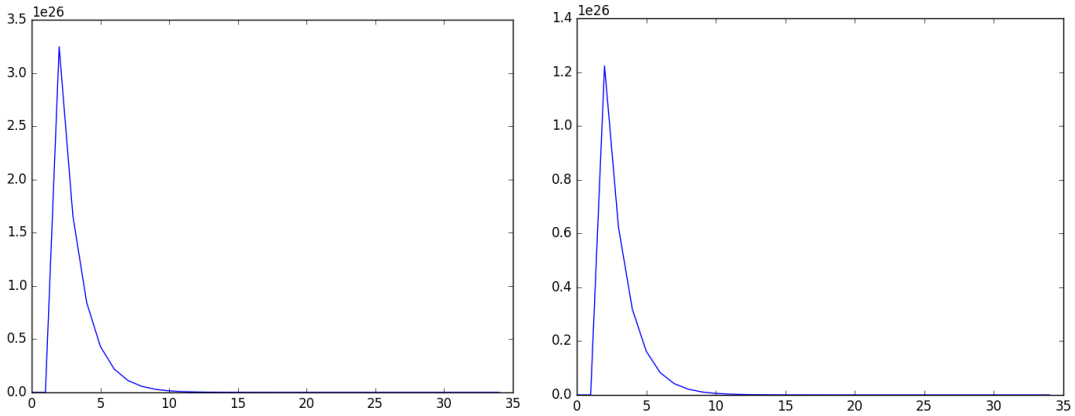


Εικόνα 4.11: Διαδικασία εφαρμογής δυαδικών τελεστών.



Εικόνα 4.12: Εφαρμογή των δυαδικών τελεστών στο αρχείο 18743.

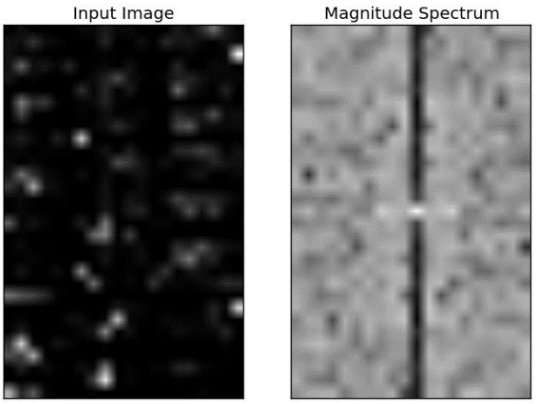
Από την Εικόνα 4.12, όπου βλέπουμε την εφαρμογή των δυαδικών τελεστών στο αρχείο 18743, διαπιστώνουμε ότι οι περιπτώσεις που αξίζουν την εξέταση τους στο δίκτυο είναι η χρήση τετραγωνικού πυρήνα μεγέθους 2×2 με OPEN τελεστή καθώς και του πυρήνα σχήματος σταυρού, μεγέθους 2×2 με OPEN τελεστή.



Γραφική 4.12: Χρήση cross πυρήνα – Αριστερά. Χρήση τετραγωνικού πυρήνα – Δεξιά.

Δυστυχώς, η χρήση και των δύο συνόλων δεδομένων, δεν καρποφόρησαν και το ποσοστό επιτυχίας παρέμεινε σταθερό στο 42.89%.

Μετασχηματισμός Fourier

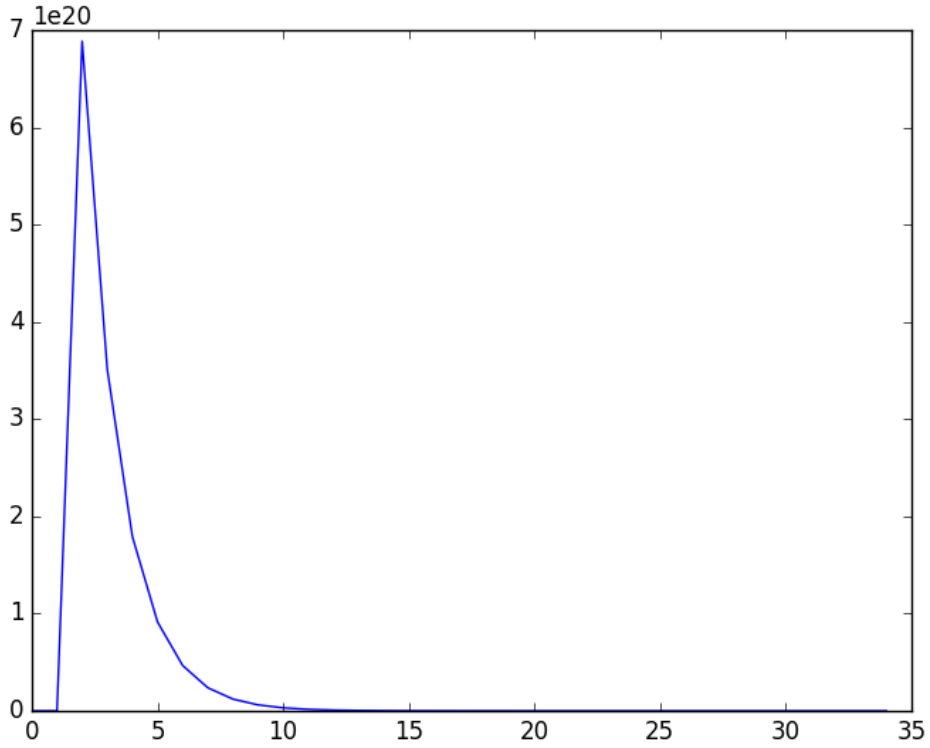


Εικόνα 4.13: Μετασχηματισμός Fourier.

Με την εφαρμογή του μετασχηματισμού Fourier φτάνουμε στο τέλος των πειραμάτων. Μπορεί να φαίνεται τρελή προσέγγιση, αλλά προσπαθώντας να εξαλείψουμε το πρόβλημα που περιγράφεται στο Υποκεφάλαιο "Μέγεθος αρχείου", μετατρέπουμε τις εικόνες των πρωτεϊνών σε μονάδες κύκλου (μονάδα μέτρησης της συχνότητας). Η χρήση

εικόνων που αναπαριστούν το φάσμα συχνοτήτων (Εικόνα 4.13) αγνοούν τη χωρική τοποθέτηση των αντικειμένων στην εικόνα, δημιουργώντας μας ελπίδες για καλύτερα αποτελέσματα.

Κλείνοντας, παρά τις ελπίδες, η εκτέλεση του δικτύου δεν βελτίωσε τα αποτελέσματα παραμένωντας στο 42.89 %.



Γραφική 4.13: Χρήση εικόνων φάσης συχνοτήτων.

4.2 Συμπεράσματα και μελλοντική έρευνα

Το πρόβλημα που αντιμετωπίζουμε σχετικά με τη αμεταβλητότητα του ποσοστού επιτυχίας είναι αρκετά ανησυχητικό γιατί εμφανίζεται ο ίδιος αριθμός με οποιαδήποτε αλλαγή και αν κάνουμε στο δίκτυο.

Οδηγούμαστε, λοιπόν, στην εκτύπωση της πρόβλεψης που κάνει το δίκτυο για το σύνολο των δεδομένων και διαπιστώνουμε ότι σε όλες τις προσπάθειες εκτέλεσης το δίκτυο εξάγει ως πιθανή δευτεροταγή δομή, την κατηγορία coil ("C"). Με μερική μελέτη βρίσκουμε ότι το ποσοστό των coils στο σύνολο των επιθυμητών αποτελεσμάτων είναι 42.53% · ποσοστό πολύ κοντινό στα πραγματικά αποτελέσματα του δικτύου. Συμπεραίνουμε, ότι το δίκτυο καταφέρνει να βρει τη δευτεροταγή δομή μόνο για μια κατηγορία.

Γιατί;

Θεωρούμε ότι η αίτια του προβλήματος είναι τα χαρακτηριστικά του δικτύου. Όπως περιγράφηκε πιο πάνω, το δίκτυο εντοπίζει στην έκταση της εικόνας τα coils και όταν του δίνουμε ως επιθυμητό αποτέλεσμα helixes, αυτό αδυνατεί να το εντοπίσει.

Μελλοντική εργασία

Συνοψίζοντας, μπορούμε να επισημάνουμε ότι το δίκτυο ConvNet είναι ένα πολύ ισχυρό μαθηματικό μοντέλο που ο συνδυασμός της ισχύς του με το σύνολο δεδομένων εισόδου που είχαμε στη διάθεσή μας, δεν έφεραν τα επιθυμητά αποτελέσματα. Κατά την άποψή μου, δεν πρέπει να εγκαταλειφθεί η μελέτη των deep networks από την ομάδα έρευνας του πανεπιστημίου, αλλά θα ήταν καλύτερο να υπάρχει η δυνατότητα προσαρμογής του συνόλου δεδομένων για κάθε μοντέλο. Αναλυτικότερα, όπως έδειξαν οι προηγούμενες έρευνες, η αντιμετώπιση του προβλήματος όταν παρομοιάστηκε με το NLP πρόβλημα έδωσε πολύ καλύτερα αποτελέσματα. Μια μελλοντική προσέγγιση, είναι η αυτούσια χρήση των MSA files, χωρίς το ενδιάμεσο βήμα word2vec για εύρεση κωδικοποίησης, ως είσοδο στην NLP αρχιτεκτονική των ConvNets. Κλείνοντας, στην περίπτωση που αυτό δεν επιφέρει καλά αποτελέσματα, μπορεί τότε να γίνει εκμετάλλευση των MSA files για εύρεση μιας διαφορετικής καλής κωδικοποίησης.

Βιβλιογραφία

[1]: LeCun Y., Bottou L., Bengio Y., Haffner P. *"Gradient-based learning applied to document recognition"*, Proceedings of the IEEE 86 (11), 2278-2324, November 1989.

[2]: Srivastava N., Hinton G. E., Krizhevsky A., Sutskever I., Salakhutdinov R., *"Dropout: A Simple Way to Prevent Neural Networks from Overfitting"*, Toronto, 2014.

[3]: Hinton G.E., Srivastava N., Krizhevsky A., Sutskever I., Salakhutdinov R., *"Improving neural networks by preventing co-adaptation of feature detectors"*, Toronto, 2012.

[4]: Rumelhart D.E., Hinton G.E., Williams R.J, *"Learning representations by back-propagating errors"*, University of California, 1986.

[5]: Neural networks and deep learning, March 22, 2016, http://neuralnetworksanddeeplearning.com/chap3.html.

[6]: Krizhevsky A., Sutskever I., Hinton G.E, *"ImageNet Classification with Deep Convolutional Neural Networks"*, Toronto, 2012.

[7]: Kabsch W., and Sander C., *"Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features"*, Biopolymers, Vol 22 (12), 1983.

[8]: Αγαθοκλέους Μ., *"Πρόβλεψη Δευτεροταγούς Δομής Πρωτεϊνών με Νευρωνικά Δίκτυα Αμφίδρομης Ανάδρασης"*, Προπτυχιακή διπλωματική, Πανεπιστήμιο Κύπρου, Λευκωσία, 2009.

[9]: Χριστοδούλου Γ., *"Διερεύνηση μεθόδων εκπαίδευσης νευρωνικών δικτύων αμφίδρομης ανάδρασης για πρόβλεψη δευτεροταγούς δομής πρωτεϊνών"*, Προπτυχιακή διπλωματική, Πανεπιστήμιο Κύπρου, Λευκωσία, 2010.

[10]: Χριστοδούλου Χρ., Χατζηνεοφύτου Μ., *"Βιολογία. 2nd ed.*", pp. 28-32, Λευκωσία, 2005.

[11]: Editorial, *"So much more to know."*, Science 2005, Vol 309 (78-102).

[12]: Dill K.A., Ozkan S.B., Weikl T.R, Chodera J.D., Voelz V.A., *"The protein folding problem: when will it be solved?"*, California, 2007.

[13]: Levinthal C., "How to Fold Graciously", Illinois, 1969.

[14]: Baldi P., Brunak S., Frasconi P., Soda G., Pollastri G., *"Exploiting the Past and the Future in Protein Secondary Structure Prediction"*, Bioinformatics, Vol. 15, No. 11, pp. 937-946, 1999.

[15]: Chou P.Y., Fasman GD., *"Prediction of protein conformation"*, Biochemistry, Vol. 13(2), pp. 222-45, 1974.

[16]: Schmidhuber J., *"Deep Learning in Neural Networks: An Overview*", Vol. 61, pp. 85–117, 2015.

[17]: Yandell M.D, Majoros W.H., *"*Genomics and natural language processing" Nature Reviews Genetics 3, pp. 601-610, 2002.

[18]: McCulloch W.S. and Pitts W. *"A Logical Calculus of the Ideas Immanent in Nervous Activity"*, Bulletin of Mathematical Biophysics, Vol. 5, pp. 115-133, 1943.

[19]: Rumelhart D. E., Hinton G. E. and Williams R. J., *"Learning internal representations by error propagation"*, Parallel Distributed Processing, Vol. 1, pp. 318-362, 1986.

[20]: Harris D., Harris S., *"Digital design and computer architecture. 2nd ed.*". San Francisco, p. 129, 2012, ISBN 978-0-12-394424-5.

[21]: Rost B., Sander C. *"Improved prediction of protein secondary structure by use of sequence profiles and neural networks"*, PNAS, Vol. 90, pp. 7558-7562, 1993.

[22]: Sommerville I., *"Software Engineering"*, Pearson, 9th edition, 2011, ISBN 0-13-705346-0.

[23]: H. van Vliet, *"Software Engineering: Principles and Practice"*, 3rd edition, 2008.

[24]: Minsky M.L., (1963). *"Steps toward artificial intelligence"*, Computers and Thought, pp. 406-450, New York, 1963.

[25]: Bengio Y., *"Learning Deep Architectures for AI"*, Montreal, 2009

[26]: Serre T., Kreiman G., Kouh M., Cadieu C., Knoblich U., Poggio T., *"A quantitative theory of immediate visual recognition"*, Progress in Brain Research, Computational Neuroscience: Theoretical Insights into Brain Function, Vol. 165, pp. 33–56, 2007.

[27]: Hinton G.E., Osindero S., The Y., *"A fast learning algorithm for deep belief nets"*, Neural Computation, vol. 18, pp. 1527–1554, 2006.

[28]: Fukushima K., *"Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position"*, Biological Cybernetics, Vol. 36, pp. 193–202, 1980.

[29]: Singh M., "*Predicting protein secondary and supersecondary structure",* Handbook of Computational Molecular Biology, Princeton, 2005.

[30]: word2vec - Revision 42: /trunk. 2016. April 22, 2016 https://word2vec.googlecode.com/svn/trunk/.

[31]: Mikolov T., Chen K., Corrado G., Dean J., *"Efficient Estimation of Word Representations in Vector Space"*, 2013.

[*32]:* Leskovec J., Rajaraman A., Ullman J.D., *"Mining of Massive Datasets"*. 1 Edition, pp. 418-427, 2011.

[33]: Asgari E., Mofrad M.R.K., *"ProtVec: A Continuous Distributed Representation of Biological Sequences"*, 2015.

[34]: Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), 1746–1751.

[35]: Ronan C., Weston J, Bottou L., Karln M., Kavukcuoglu K., Kuksa P., "*Natural Language Processing (almost) from Scratch"*, 2011.

[36]: Bovik, A. *"The essential guide to image processing"*, Burlington, Mass. London: Academic Press, 2009.

[37]: Convolutional Neural Networks for Visual Recognition, 27 April 2016, http://cs231n.github.io/neural-networks-1/

[38]: Understanding Convolutional Neural Networks for NLP, 20 March 2016, http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

[39]: PHPMailer, A full-featured email creation and transferclass for PHP, GitHub, 10 May 2016, "https://github.com/PHPMailer/PHPMailer "

[40]: Microsoft Neural Net Shows Deep Learning can get wat deeper, Wired, 18 January 2016, http://www.wired.com/2016/01/microsoft-neural-net-shows-deep-learning-can-get-way-deeper/

[41]: Ricci F., Rokach L., Shapira B., *"Introduction to Recommender Systems Handbook"*, Recommender Systems Handbook, Springer, pp. 1-35, 2011.

[42]: Wang H., Wang N., Yeung D., *"Collaborative Deep Learning for Recommender Systems",* Hong-Kong, 2015

[43]: Recommending music on Spotify with deep learning, 28 May 2016, http://benanne.github.io/2014/08/05/spotify-cnns.html

Appendix

storeProteins.py

```
import os
import sys
from shutil import copyfile
from random import randint

if len(sys.argv) < 3:
   print "Give me a file and a source folder!"
   exit(1)

targetFolder="ProteinPool2"
if not os.path.exists(targetFolder):
   os.makedirs(targetFolder)
targetFolder="ProteinPool2/"

fileName = sys.argv[1]
sourceFolder = sys.argv[2]

open_file = open(fileName,"r")
lines = open_file.readlines()

target_output_file = open("target_output.txt","w")


for i in range(0, len(lines), 3):
   name = lines[i].rstrip()
   name = name+".hssp"
   print name
   index=0
   secondary = lines[i+2].rstrip()
   for character in secondary:
      if (character!="C" and character!="E" and character!="H"):
         continue
         num = randint(0,2)
         print num
         if (num==0):
            character="C"
         elif (num==1):
            character="E"
         else:
            character="H"
      target_output_file.write(character)

   src = sourceFolder+name
   dst = targetFolder+name

   copyfile(src, dst)

open_file.close()
```

newInputData.py

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import misc
import math
import csv

# EXECUTE
# wc -l <filename>
# in terminal to determine the number of lines

fileCounter=1
textName = 'file'+str(fileCounter)

numOfLines = 0
thefilepath = "all.hssp"
inputFile = np.loadtxt('all.hssp')

with open(thefilepath, 'rU') as f:
	for index in f:
		numOfLines+=1

width = 0
windowSize = float(31)
middle = int(math.floor(windowSize/2))

for height in range (0, numOfLines):
	diff = middle - height
	additional = numOfLines - height
	add = height + middle
	if (diff <= middle and diff > 0):
		print "&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&"
		patch = np.zeros((diff, 20))
		patch2 = inputFile[height-height:height+middle+1, width:width+20]
		table = np.concatenate([patch,patch2])
		table.astype(int)
		# if (height==14):
		#	print table

	elif (add >= numOfLines):
		print "##########################################"
		d = add - numOfLines
		end_patch = inputFile[height-middle:height+middle-d, width:width+20]
		end_patch2 = np.zeros((d+1, 20))
		table = np.concatenate([end_patch,end_patch2])
		table.astype(int)

		else:
		table = inputFile[height-middle:height+middle+1, width:width+20]
		table.astype(int)

	textName= 'file'+str(fileCounter)+'.csv'

	# write it
	with open(textName, 'w') as csvfile:
		writer = csv.writer(csvfile)
		[writer.writerow(r) for r in table]

	fileCounter+=1
```

readData_TEXT.py

```
import numpy as np # linear algebra

# from subprocess import check_output
import os, sys, tarfile
import gzip, cPickle
import theano
import csv

def loadTrainCSV(fileName):
	PhotoID_Labels = []
	target_output_file = open(fileName,"r")
	line = target_output_file.readline()

	count =0

	for t in line:
		count +=1
		if(t=="C"):
			PhotoID_Labels.append(2)
		elif (t=="E"):
			PhotoID_Labels.append(1)
		else:
			PhotoID_Labels.append(0)

	return PhotoID_Labels

def split_list(a_list):
   half = len(a_list)/2
   a_half = []

   return a_list[:half], a_list[half:]

def thesis_data():

	if len(sys.argv) < 3:
		print "Give me a folder for the input data and a file for the target outputs.!"
		exit(1)

	fi = sys.argv[1]
	targetFileName = sys.argv[2]

	# open_file = open(fileName,"r")
	# lines = open_file.readlines()


	training_results = loadTrainCSV(targetFileName)

	# fi = "./Data/ExpandedData/"		# TRAIN PHOTOS Directory

	classes = os.listdir(fi)
	counter = 0
```

```
# JUST COMMENTED
        training_inputs = []
        # training_results = []

        # read TRAIN PHOTOS
        for filename in classes:

                if (filename == ".DS_Store"):
                        print "something"
                        continue
                fileCounter = counter + 1
                finalFileName = fi+'file'+str(fileCounter)+'.csv'
                with open(finalFileName, 'r') as csvfile:
                        reader = csv.reader(csvfile)
                        table = [[float(e) for e in r] for r in reader]

                # convert table to 1D array
                arr = np.array(table)
                oneDarray = arr.ravel()

                training_inputs.append(oneDarray)

        training_inputs , validation_test_inputs = split_list(training_inputs)
        training_results , validation_test_results = split_list(training_results)

        validation_inputs , test_inputs = split_list(validation_test_inputs)
        validation_results, test_results = split_list(validation_test_results)

        training_inputs = np.asarray(training_inputs)
        validation_inputs = np.asarray(validation_inputs)
        test_inputs = np.asarray(test_inputs)
        print type(training_inputs)
        print len(training_inputs)
        print training_inputs

        training_results = np.asarray(training_results)
        validation_results = np.asarray(validation_results)
        test_results = np.asarray(test_results)
        print type(training_results)
        print len(training_results)
        print training_results

        print len(validation_results)
        print len(validation_inputs)

        training_data_tuple = training_inputs, training_results
        validation_data_tuple = validation_inputs, validation_results
        test_data_tuple = test_inputs, test_results

        dataset = training_data_tuple, validation_data_tuple, test_data_tuple
        f = gzip.open('protein_text_reversedOutputs.pkl.gz', 'wd')

        cPickle.dump(dataset, f)
        f.close()

        return

thesis_data()
print "ok"
```

network3.py

```
"""network3.py
~~~~~~~~~~~~~~

A Theano-based program for training and running simple neural
networks.

Supports several layer types (fully connected, convolutional, max
pooling, softmax), and activation functions (sigmoid, tanh, and
rectified linear units, with more easily added).

When run on a CPU, this program is much faster than network.py and
network2.py.  However, unlike network.py and network2.py it can also
be run on a GPU, which makes it faster still.

Because the code is based on Theano, the code is different in many
ways from network.py and network2.py.  However, where possible I have
tried to maintain consistency with the earlier programs.  In
particular, the API is similar to network2.py.  Note that I have
focused on making the code simple, easily readable, and easily
modifiable.  It is not optimized, and omits many desirable features.

This program incorporates ideas from the Theano documentation on
convolutional neural nets (notably, http://deeplearning.net/tutorial/lenet.html ), from Misha Denil's
implementation of dropout (https://github.com/mdenil/dropout ), and
from Chris Olah (http://colah.github.io).

"""

#### Libraries
# Standard library
import cPickle
import gzip

# Third-party libraries
import numpy as np
import theano
import theano.tensor as T
from theano.tensor.nnet import conv
from theano.tensor.nnet import softmax
from theano.tensor import shared_randomstreams
from theano.tensor.signal import downsample

# Activation functions for neurons
def linear(z): return z
def ReLU(z): return T.maximum(0.0, z)
from theano.tensor.nnet import sigmoid
from theano.tensor import tanh


#### Constants
GPU = True
if GPU:
    print "Trying to run under a GPU.  If this is not desired, then modify "+\
        "network3.py\nto set the GPU flag to False."
```

```
    try: theano.config.device = 'gpu'
    except: pass # it's already set
    theano.config.floatX = 'float32'
else:
    print "Running with a CPU.  If this is not desired, then the modify "+\
        "network3.py to set\nthe GPU flag to True."

#### Load the MNIST data

def load_data_shared(filename="../data/protein_text.pkl.gz"):
    f = gzip.open(filename, 'rb')
    training_data, validation_data, test_data = cPickle.load(f)
    f.close()
    def shared(data):
        """Place the data into shared variables.  This allows Theano to copy
        the data to the GPU, if one is available.

        """
        shared_x = theano.shared(
            np.asarray(data[0], dtype=theano.config.floatX), borrow=True)
        shared_y = theano.shared(
            np.asarray(data[1], dtype=theano.config.floatX), borrow=True)
        return shared_x, T.cast(shared_y, "int32")
    return [shared(training_data), shared(validation_data), shared(test_data)]

#### Main class used to construct and train networks
class Network(object):

    def __init__(self, layers, mini_batch_size):
        """Takes a list of `layers`, describing the network architecture, and
        a value for the `mini_batch_size` to be used during training
        by stochastic gradient descent.

        """
        self.layers = layers
        self.mini_batch_size = mini_batch_size
        self.params = [param for layer in self.layers for param in layer.params]
        self.x = T.matrix("x")
        self.y = T.ivector("y")
        init_layer = self.layers[0]
        init_layer.set_inpt(self.x, self.x, self.mini_batch_size)
        for j in xrange(1, len(self.layers)):
            prev_layer, layer = self.layers[j-1], self.layers[j]
            layer.set_inpt(
                prev_layer.output, prev_layer.output_dropout, self.mini_batch_size)
        self.output = self.layers[-1].output
        self.output_dropout = self.layers[-1].output_dropout

    def SGD(self, training_data, epochs, mini_batch_size, eta,
            validation_data, test_data, lmbda=0.0):
        """Train the network using mini-batch stochastic gradient descent."""
        training_x, training_y = training_data
        validation_x, validation_y = validation_data
        test_x, test_y = test_data

        # compute number of minibatches for training, validation and testing
```

```
num_training_batches = size(training_data)/mini_batch_size
num_validation_batches = size(validation_data)/mini_batch_size
num_test_batches = size(test_data)/mini_batch_size

# define the (regularized) cost function, symbolic gradients, and updates
l2_norm_squared = sum([(layer.w**2).sum() for layer in self.layers])
cost = self.layers[-1].cost(self)+\
      0.5*lmbda*l2_norm_squared/num_training_batches
grads = T.grad(cost, self.params)
updates = [(param, param-eta*grad)
         for param, grad in zip(self.params, grads)]

# define functions to train a mini-batch, and to compute the
# accuracy in validation and test mini-batches.
i = T.lscalar() # mini-batch index
train_mb = theano.function(
   [i], cost, updates=updates,
   givens={
      self.x:
      training_x[i*self.mini_batch_size: (i+1)*self.mini_batch_size],
      self.y:
      training_y[i*self.mini_batch_size: (i+1)*self.mini_batch_size]
   })
validate_mb_accuracy = theano.function(
   [i], self.layers[-1].accuracy(self.y),
   givens={
      self.x:
      validation_x[i*self.mini_batch_size: (i+1)*self.mini_batch_size],
      self.y:
      validation_y[i*self.mini_batch_size: (i+1)*self.mini_batch_size]
   })
test_mb_accuracy = theano.function(
   [i], self.layers[-1].accuracy(self.y),
   givens={
      self.x:
      test_x[i*self.mini_batch_size: (i+1)*self.mini_batch_size],
      self.y:
      test_y[i*self.mini_batch_size: (i+1)*self.mini_batch_size]
   })
self.test_mb_predictions = theano.function(
   [i], self.layers[-1].y_out,
   givens={
      self.x:
      test_x[i*self.mini_batch_size: (i+1)*self.mini_batch_size]
   })
# Do the actual training
best_validation_accuracy = 0.0
for epoch in xrange(epochs):
   for minibatch_index in xrange(num_training_batches):
      iteration = num_training_batches*epoch+minibatch_index
      if iteration % 1000 == 0:
         print("Training mini-batch number {0}".format(iteration))
      cost_ij = train_mb(minibatch_index)
      # print cost_ij
      if (iteration+1) % num_training_batches == 0:
         print "^^^^^^^^^^^^^^^^^^^^"
```

```
            # tmpp = [validate_mb_accuracy(j) for j in xrange(num_validation_batches)]
            # print tmpp
            # test_mb_predictions

            validation_accuracy = np.mean(
               [validate_mb_accuracy(j) for j in xrange(num_validation_batches)])
            print("Epoch {0}: validation accuracy {1:.2%}".format(
               epoch, validation_accuracy))
            if validation_accuracy >= best_validation_accuracy:
               print("This is the best validation accuracy to date.")
               best_validation_accuracy = validation_accuracy
               best_iteration = iteration
               if test_data:
                  test_accuracy = np.mean(
                     [test_mb_accuracy(j) for j in xrange(num_test_batches)])
                  print('The corresponding test accuracy is {0:.2%}'.format(
                     test_accuracy))
      print("Finished training network.")
      print("Best validation accuracy of {0:.2%} obtained at iteration {1}".format(
         best_validation_accuracy, best_iteration))
      print("Corresponding test accuracy of {0:.2%}".format(test_accuracy))

      print "#################"
      # print len(self.test_mb_predictions)
      xsx = [self.test_mb_predictions(j) for j in xrange(num_validation_batches)]
      print xsx
      print len(xsx[0])
      print len(xsx)

      print "**********************"
      print "FILTERS"
      print "**********************"


#### Define layer types

class ConvPoolLayer(object):
   """Used to create a combination of a convolutional and a max-pooling
   layer.  A more sophisticated implementation would separate the
   two, but for our purposes we'll always use them together, and it
   simplifies the code, so it makes sense to combine them.

   """

   def __init__(self, filter_shape, image_shape, poolsize=(2, 2),
                activation_fn=sigmoid):
      """`filter_shape` is a tuple of length 4, whose entries are the number
      of filters, the number of input feature maps, the filter height, and the
      filter width.

      `image_shape` is a tuple of length 4, whose entries are the
      mini-batch size, the number of input feature maps, the image
      height, and the image width.

      `poolsize` is a tuple of length 2, whose entries are the y and
      x pooling sizes.
```

```
        """
        self.filter_shape = filter_shape
        self.image_shape = image_shape
        self.poolsize = poolsize
        self.activation_fn=activation_fn
        # initialize weights and biases
        n_out = (filter_shape[0]*np.prod(filter_shape[2:])/np.prod(poolsize))
        self.w = theano.shared(
            np.asarray(
                np.random.normal(loc=0.5, scale=np.sqrt(1.0/n_out), size=filter_shape),
                dtype=theano.config.floatX),
            borrow=True)
        self.b = theano.shared(
            np.asarray(
                np.random.normal(loc=0.5, scale=1.0, size=(filter_shape[0],)),
                dtype=theano.config.floatX),
            borrow=True)
        self.params = [self.w, self.b]

    def set_inpt(self, inpt, inpt_dropout, mini_batch_size):
        self.inpt = inpt.reshape(self.image_shape)
        conv_out = conv.conv2d(
            input=self.inpt, filters=self.w, filter_shape=self.filter_shape,
            image_shape=self.image_shape)
        pooled_out = downsample.max_pool_2d(
            input=conv_out, ds=self.poolsize, ignore_border=True)
        self.output = self.activation_fn(
            pooled_out + self.b.dimshuffle('x', 0, 'x', 'x'))
        self.output_dropout = self.output # no dropout in the convolutional layers

class FullyConnectedLayer(object):

    def __init__(self, n_in, n_out, activation_fn=sigmoid, p_dropout=0.0):
        self.n_in = n_in
        self.n_out = n_out
        self.activation_fn = activation_fn
        self.p_dropout = p_dropout
        # Initialize weights and biases
        self.w = theano.shared(
            np.asarray(
                np.random.normal(
                    loc=0.5, scale=np.sqrt(1.0/n_out), size=(n_in, n_out)),
                dtype=theano.config.floatX),
            name='w', borrow=True)
        self.b = theano.shared(
            np.asarray(np.random.normal(loc=0.5, scale=1.0, size=(n_out,)),
                       dtype=theano.config.floatX),
            name='b', borrow=True)
        self.params = [self.w, self.b]

    def set_inpt(self, inpt, inpt_dropout, mini_batch_size):
        self.inpt = inpt.reshape((mini_batch_size, self.n_in))
        self.output = self.activation_fn(
            (1-self.p_dropout)*T.dot(self.inpt, self.w) + self.b)
        self.y_out = T.argmax(self.output, axis=1)
```

```
        self.inpt_dropout = dropout_layer(
            inpt_dropout.reshape((mini_batch_size, self.n_in)), self.p_dropout)
        self.output_dropout = self.activation_fn(
            T.dot(self.inpt_dropout, self.w) + self.b)

    def accuracy(self, y):
        "Return the accuracy for the mini-batch."
        return T.mean(T.eq(y, self.y_out))

class SoftmaxLayer(object):

    def __init__(self, n_in, n_out, p_dropout=0.0):
        self.n_in = n_in
        self.n_out = n_out
        self.p_dropout = p_dropout
        # Initialize weights and biases
        self.w = theano.shared(
            np.zeros((n_in, n_out), dtype=theano.config.floatX),
            name='w', borrow=True)
        self.b = theano.shared(
            np.zeros((n_out,), dtype=theano.config.floatX),
            name='b', borrow=True)
        self.params = [self.w, self.b]

    def set_inpt(self, inpt, inpt_dropout, mini_batch_size):
        self.inpt = inpt.reshape((mini_batch_size, self.n_in))
        self.output = softmax((1-self.p_dropout)*T.dot(self.inpt, self.w) + self.b)
        # print self.output
        self.y_out = T.argmax(self.output, axis=1)
        self.inpt_dropout = dropout_layer(
            inpt_dropout.reshape((mini_batch_size, self.n_in)), self.p_dropout)
        self.output_dropout = softmax(T.dot(self.inpt_dropout, self.w) + self.b)

    def cost(self, net):
        "Return the log-likelihood cost."
        return -T.mean(T.log(self.output_dropout)[T.arange(net.y.shape[0]), net.y])

    def accuracy(self, y):
        "Return the accuracy for the mini-batch."
        # a = T.mean(T.eq(y, self.y_out))
        # print "%%%%%%%%%%%%"
        return T.mean(T.eq(y, self.y_out))


#### Miscellanea
def size(data):
    "Return the size of the dataset `data`."
    return data[0].get_value(borrow=True).shape[0]

def dropout_layer(layer, p_dropout):
    srng = shared_randomstreams.RandomStreams(
        np.random.RandomState(0).randint(999999))
    mask = srng.binomial(n=1, p=1-p_dropout, size=layer.shape)
    return layer*T.cast(mask, theano.config.floatX)
```