
Πανεπιστήμιο Θεσσαλίας

Πολυτεχνική Σχολή

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών


# “Τεχνικές Βαθιάς Γνώσης σε Πολυπρακτορικά Συστήματα”

Διπλωματική Εργασία


**Σαρδέλης Σπυρίδων**

Επιβλέποντες Καθηγητές:

Δασκαλοπούλου Ασπασία
Επίκουρη Καθηγήτρια
Πανεπιστήμιο Θεσσαλίας

Αργυρίου Αντώνιος
Επίκουρος Καθηγητής
Πανεπιστήμιο Θεσσαλίας



Βόλος, Ελλάδα

Οκτώβριος 2017


University of Thessaly

School of Engineering

Department of Electrical and Computer Engineering


# "Deep Learning Techniques in Multi-Agent Systems"

Diploma Thesis Project


**Sardelis Spyridon**


Supervisors:


Daskalopoulou Aspasia

Assistant Professor

University of Thessaly

Argyriou Antonios

Assistant Professor

University of Thessaly


*(Signature)* *(Signature)*

..... .....

Volos, Greece

October 2017

# Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω την οικογένεια μου για την στήριξη που μου παρείχε κατά τη διάρκεια των φοιτητικών μου χρόνων. Εν συνεχεία, να ευχαριστήσω τους επιβλέποντες καθηγητές μου την κυρία Δασκαλοπούλου Ασπασία και τον κύριο Αργυρίου Αντώνιο που μου εμπιστεύτηκαν την εν λόγω διπλωματική εργασία και με έφεραν σε επαφή με το άκρως ενδιαφέρον αντικείμενο των Τεχνικών Βαθιάς Μάθησης και των Πολυπρακτορικών Συστημάτων. Επίσης, να ευχαριστήσω τους φίλους μου για τη συμπαράσταση που μου έδειξαν όλα τα χρόνια των σπουδών μου και ιδιαίτερα τους προπτυχιακούς φοιτητές του τμήματος Σκετόπουλο Νικόλαο και Γεωργιάδη Κωνσταντίνο με τους οποίους είχα και την τιμή να συνεργαστώ άψογα σε πολλές εργασίες.

*Σαρδέλης Π. Σπυρίδων*
*Βόλος*
*Οκτώβριος 2017*

# Περίληψη

Η παρούσα διπλωματική εργασία εξετάζει τις Τεχνικές Βαθιάς Μάθησης (Deep Learning Techniques) σε σχέση με τα Πολυπρακτορικά Συστήματα (Multi-Agent Systems). Συγκεκριμένα, παρουσιάζει 5 τεχνικές (Deep Boltzmann Machines (DBM), Restricted Boltzmann Machines (RBM), Deep Belief Networks (DBN), Deep Convolutional Neural Networks (CNN) και Stacked Auto-Encoders (SAE)) και συγκρίνει τις 4 από αυτές ως προς την ποιότητα των αποτελεσμάτων που εξάγουν, το χρόνο εκτέλεσής τους και τις απαιτήσεις μνήμης. Μετά, εξετάζει κατά πόσο αυτές μπορούν να ενσωματωθούν με επιτυχία σε ένα πολυπρακτορικό σύστημα και καταλήγει ότι υπό συγκεκριμένες προϋποθέσεις κοινωνικής αρχιτεκτονικής το σενάριο αυτό είναι εφικτό.

# Abstract

This diploma thesis examines Deep Learning Techniques in relation to Multi-Agent Systems. It presents 5 techniques (Deep Boltzmann Machines (DBM), Restricted Boltzmann Machines (RBM), Deep Belief Networks (DBN), Deep Convolutional Neural Networks (CNN), and Stacked Auto-Encoders (SAE)) and compares 4 of them in terms of the quality of the outputs they export, their execution time and memory requirements. Then, it examines whether these can be successfully integrated into a multi-agent system and concludes that under certain conditions of social architecture this will be feasible.

# Πίνακας Περιεχομένων

# Κεφάλαιο 1ο: Εισαγωγή

## 1.1. Γενική Περιγραφή Προβλήματος

Στην παρούσα διπλωματική εργασία εξετάζονται οι Τεχνικές Βαθιάς Μάθησης (Deep Learning Techniques) σε σχέση με τα Πολυπρακτορικά Συστήματα (Multi-Agent Systems). Στα επόμενα κεφάλαια θα παρουσιαστούν οι εξής τεχνικές: Deep Boltzmann Machines (DBM), Restricted Boltzmann Machines (RBM), Deep Belief Networks (DBN), Deep Convolutional Neural Networks (CNN) και Stacked Auto-Encoders (SAE). Στη συνέχεια, θα γίνει σύγκριση τεσσάρων από αυτές ως προς την ποιότητα των αποτελεσμάτων που εξάγουν, το χρόνο εκτέλεσής τους και τις απαιτήσεις χώρου αποθήκευσης. Μετά, εξετάζεται κατά πόσο αυτές μπορούν να ενσωματωθούν με επιτυχία σε ένα πολυπρακτορικό σύστημα και καταλήγει ότι υπό συγκεκριμένες προϋποθέσεις κοινωνικής αρχιτεκτονικής και με τη βοήθεια μιας αρχιτεκτονικής cloud, που θα αναλάβει την εκτέλεση των αλγορίθμων των παραπάνω τεχνικών, το σενάριο αυτό είναι εφικτό.

## 1.2. Τι είναι το Deep Learning;

Η Βαθιά Μάθηση (Deep Learning) είναι μια τεχνική Μηχανικής Μάθησης (Machine Learning) που διδάσκει τους υπολογιστές να κάνουν αυτό που φαντάζει φυσικό στον άνθρωπο, δηλαδή να μαθαίνουν με τη χρήση παραδειγμάτων. Είναι μια βασική τεχνολογία πίσω από τα μηχανοκίνητα αυτοκίνητα, που τους επιτρέπει να αναγνωρίσουν ένα σημάδι στάσης ή να διακρίνουν έναν πεζό από ένα φωτεινό σηματοδότη. Είναι το κλειδί για τον φωνητικό έλεγχο σε συσκευές ευρείας κατανάλωσης όπως τηλέφωνα, tablets, τηλεοράσεις και ηχεία ανοιχτής ακρόασης. Το Deep Learning αποτελεί το επίκεντρο των τεχνολογικών εξελίξεων τα τελευταία χρόνια καθώς προσφέρει νέες δυνατότητες και αποφέρει αποτελέσματα που δεν ήταν δυνατό να επιτευχθούν νωρίτερα.

Για παράδειγμα, στο Deep Learning, ένα μοντέλο υπολογιστή μαθαίνει να εκτελεί εργασίες ταξινόμησης απευθείας από εικόνες, κείμενο ή ήχο. Τα μοντέλα Deep Learning μπορούν να επιτύχουν πολύ υψηλή ακρίβεια, ενώ μερικές φορές να υπερβαίνουν τις επιδόσεις την ανθρώπινη σκέψη. Τα μοντέλα εκπαιδεύονται χρησιμοποιώντας ένα μεγάλο σύνολο ετικετών δεδομένων (labeled data set) και Αρχιτεκτονικές Νευρωνικών Δικτύων (Neural Network Architectures) που περιέχουν πολλαπλά στρώματα (layers).

## 1.3. Η Σημασία του Deep Learning

Ο λόγος για τον οποίο το Deep Learning επιτυγχάνει εντυπωσιακά αποτελέσματα είναι η ακρίβεια. Επιτυγχάνει την ακρίβεια αναγνώρισης σε υψηλότερα επίπεδα από ποτέ. Αυτό βοηθά τα ηλεκτρονικά είδη ευρείας κατανάλωσης να ανταποκρίνονται στις προσδοκίες των χρηστών και είναι ζωτικής σημασίας για εφαρμογές κρίσιμης σημασίας για την ασφάλεια. Οι πρόσφατες εξελίξεις στο Deep Learning έχουν βελτιωθεί στο σημείο όπου ξεπερνά τους ανθρώπους σε ορισμένες εργασίες, όπως είναι η ταξινόμηση αντικειμένων σε εικόνες.

Παρόλο που το Deep Learning υπήρχε αρχικά στη θεωρία από τη δεκαετία του 1980, υπάρχουν δύο κύριοι λόγοι για τους οποίους μόλις πρόσφατα έγινε χρήσιμο:

- Το Deep Learning απαιτεί μεγάλες ποσότητες από labeled data. Για παράδειγμα, η ανάπτυξη αυτοκινήτου χωρίς οδηγό απαιτεί εκατομμύρια εικόνες και χιλιάδες ώρες βίντεο.
- Το Deep Learning απαιτεί σημαντική υπολογιστική ισχύ. Οι GPU υψηλής απόδοσης έχουν μια παράλληλη αρχιτεκτονική που είναι αποτελεσματική. Όταν συνδυάζεται με clusters ή cloud computing, αυτό δίνει τη δυνατότητα στις ομάδες ανάπτυξης να μειώσουν το χρόνο εκπαίδευσης για ένα δίκτυο Deep Learning από εβδομάδες σε ώρες ή λιγότερο.

## 1.4. Το Deep Learning στη Βιομηχανία

Οι εφαρμογές Deep Learning χρησιμοποιούνται ευρέως στο βιομηχανικό χώρο από αυτοματοποιημένη οδήγηση μέχρι και σε ιατρικές συσκευές.

- Αυτοματοποιημένη οδήγηση:

  Οι ερευνητές της αυτοκινητοβιομηχανίας χρησιμοποιούν το Deep Learning για την αυτόματη ανίχνευση αντικειμένων όπως σημάδια στάσης και φανάρια. Επιπλέον, χρησιμοποιείται για την ανίχνευση πεζών, γεγονός που συμβάλλει στη μείωση των ατυχημάτων.

- Αεροναυπηγική και άμυνα:

  Το Deep Learning χρησιμοποιείται για τον εντοπισμό αντικειμένων από δορυφόρους που εντοπίζουν περιοχές ενδιαφέροντος και τον εντοπισμό ασφαλών ή μη ασφαλών ζωνών για στρατεύματα.

- Ιατρική έρευνα:

  Οι ερευνητές του καρκίνου χρησιμοποιούν το Deep Learning για την αυτόματη ανίχνευση των καρκινικών κυττάρων. Για παράδειγμα, ομάδες στο UCLA δημιούργησαν ένα προηγμένο μικροσκόπιο που αποδίδει ένα σύνολο δεδομένων μεγάλης διαστάσεως που χρησιμοποιείται για να εκπαιδεύσει μια εφαρμογή Deep Learning για την ακριβή αναγνώριση των καρκινικών κυττάρων.

- Βιομηχανικός Αυτοματισμός:

  Το Deep Learning συμβάλλει στη βελτίωση της ασφάλειας των εργαζομένων γύρω από τα βαριά μηχανήματα, ανιχνεύοντας αυτόματα πότε οι άνθρωποι ή τα αντικείμενα βρίσκονται σε μια μη ασφαλή απόσταση από τα μηχανήματα.

- Ηλεκτρονικά:

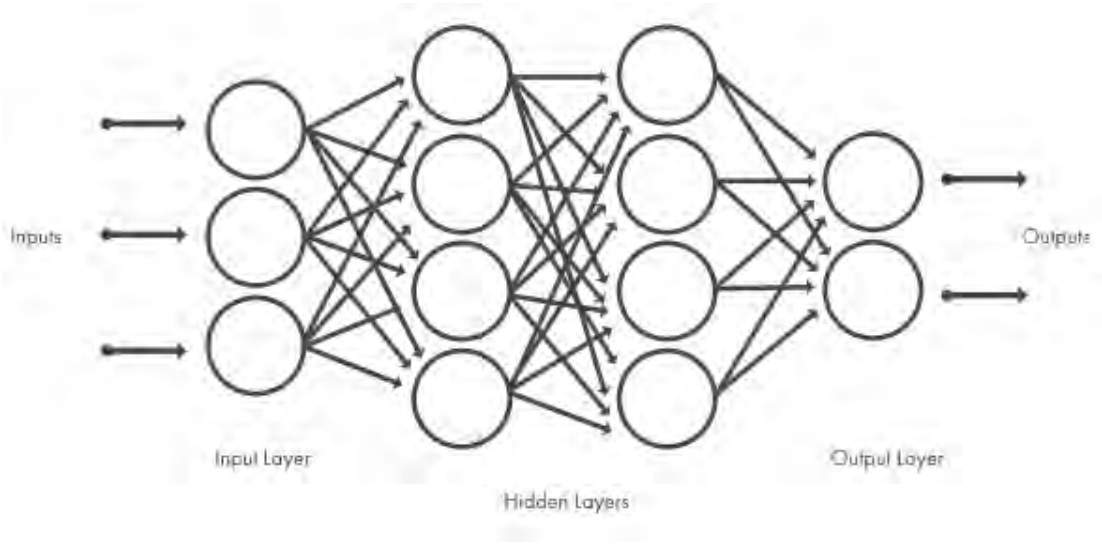
  Το Deep Learning χρησιμοποιείται σε αυτοματοποιημένη μετάφραση και ακοή. Για παράδειγμα, οι συσκευές οικιακής βοήθειας που ανταποκρίνονται στη φωνή σας και γνωρίζουν τις προτιμήσεις σας τροφοδοτούνται από εφαρμογές Deep Learning.

## 1.5. Πώς λειτουργεί το Deep Learning;

Οι περισσότερες μέθοδοι Deep Learning χρησιμοποιούν Αρχιτεκτονικές Νευρωνικών Δικτύων (Neural Network Architectures), γι' αυτό τα μοντέλα Deep Learning συχνά αναφέρονται ως Δίκτυα Νευρώνων (Neural Networks).

Ο όρος "Deep" συνήθως αναφέρεται στον αριθμό των κρυφών στρωμάτων (hidden layers) στο Neural Network. Τα παραδοσιακά Neural Networks περιέχουν μόνο 2-3 κρυμμένα στρώματα (layers), ενώ τα Βαθιά Δίκτυα (Deep Networks) μπορούν να έχουν έως 150.

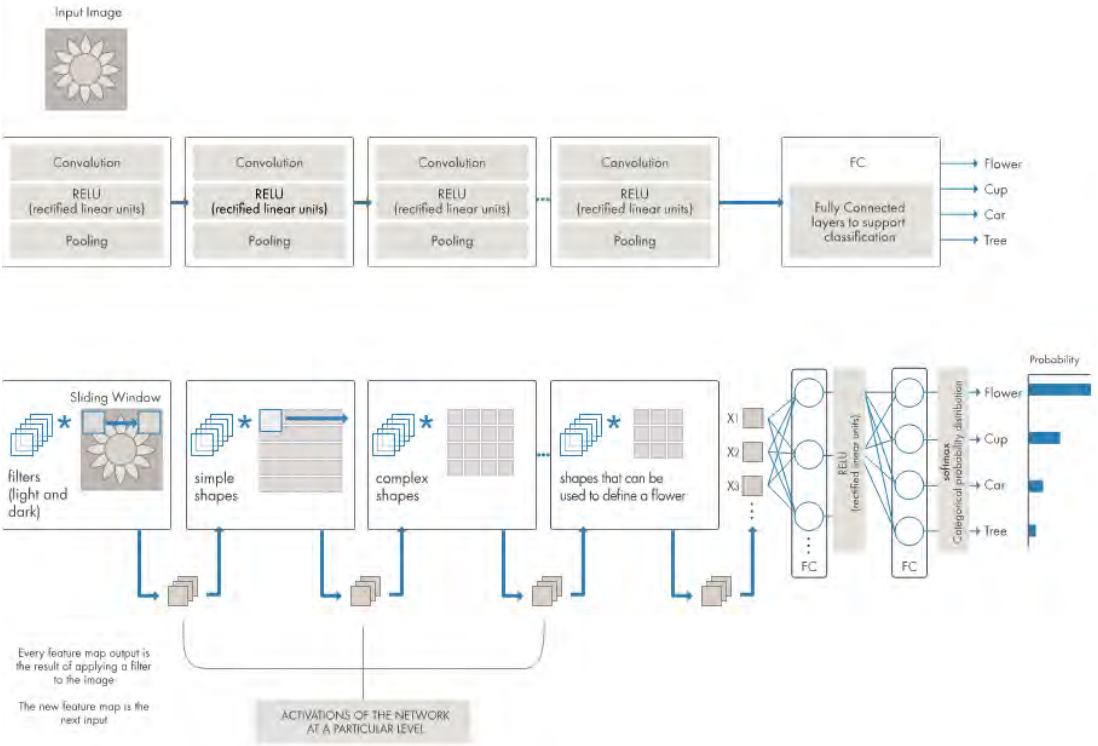
Τα μοντέλα Deep Learning εκπαιδεύονται με τη χρήση μεγάλων labeled data sets και Αρχιτεκτονικών Νευρωνικών Δικτύων που μαθαίνουν λειτουργίες απευθείας από τα δεδομένα χωρίς την ανάγκη χειροκίνητης εξαγωγής χαρακτηριστικών (Εικόνα 1.5.i).




*Εικόνα 1.5.i: Νευρωνικά Δίκτυα, τα οποία είναι οργανωμένα σε στρώματα που αποτελούνται από ένα σύνολο διασυνδεδεμένων κόμβων. Τα δίκτυα μπορούν να έχουν δεκάδες ή εκατοντάδες κρυμμένα στρώματα (hidden layers). (Πηγή: https://www.mathworks.com)*

Ένας από τους πιο δημοφιλείς τύπους Deep Neural Networks είναι γνωστός ως Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks – CNN ή ConvNet). Τα CNNs περιγράφουν τα στοιχεία που μαθαίνουν με δεδομένα εισόδου και χρησιμοποιούν δισδιάστατα στρώματα περιστροφής (convolutional layers), καθιστώντας αυτή την αρχιτεκτονική κατάλληλη για την επεξεργασία δεδομένων δύο διαστάσεων, όπως είναι οι εικόνες.

Τα CNNs εξαλείφουν την ανάγκη για χειροκίνητη εξαγωγή χαρακτηριστικών, οπότε δεν χρειάζεται να προσδιορίζονται οι λειτουργίες που χρησιμοποιούνται για την ταξινόμηση εικόνων. Τα CNNs λειτουργούν εξάγοντας χαρακτηριστικά απευθείας από τις εικόνες. Τα σχετικά χαρακτηριστικά δεν έχουν προρυθμιστεί. Μαθαίνονται ενώ το δίκτυο εκπαιδεύεται σε μια συλλογή εικόνων. Αυτή η αυτόματη εξαγωγή χαρακτηριστικών καθιστά τα μοντέλα Deep Learning ιδιαίτερα ακριβή για τα καθήκοντα όρασης υπολογιστή όπως η ταξινόμηση αντικειμένων.




*Εικόνα 1.5.ii: Παράδειγμα δικτύου με πολλά στρώματα συνέλιξης (convolutional layers). Τα φίλτρα εφαρμόζονται σε κάθε εικόνα προπόνησης με διαφορετικές αναλύσεις και η έξοδος κάθε συνεστραμμένης εικόνας χρησιμεύει ως είσοδος στην επόμενη στρώση. (Πηγή: https://www.mathworks.com)*

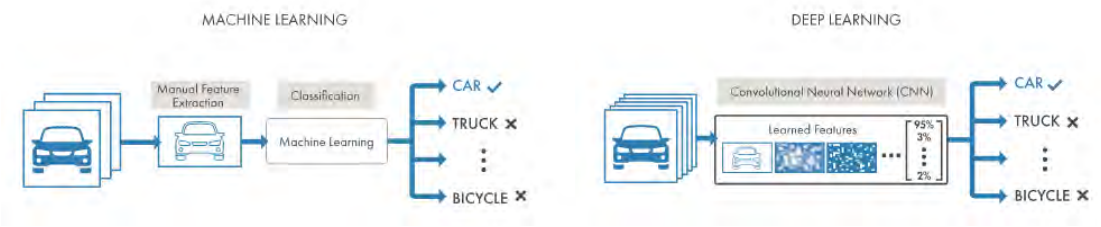
Τα CNNs μαθαίνουν να ανιχνεύουν διαφορετικά χαρακτηριστικά μιας εικόνας χρησιμοποιώντας δεκάδες ή εκατοντάδες κρυμμένα στρώματα. Κάθε κρυμμένη στρώση αυξάνει την πολυπλοκότητα των χαρακτηριστικών της εικόνας που έχουν μάθει. Για παράδειγμα, το πρώτο κρυφό στρώμα θα μπορούσε να μάθει πώς να ανιχνεύει τις άκρες και το τελευταίο μαθαίνει πώς να ανιχνεύει πιο πολύπλοκα σχήματα, ειδικά για το σχήμα του αντικειμένου που προσπαθούμε να αναγνωρίσουμε (Εικόνα 1.5.ii).

## 1.6. Machine Learning vs Deep Learning

Το Machine Learning προσφέρει μια ποικιλία τεχνικών και μοντέλων που μπορούν να χρησιμοποιηθούν ανάλογα με την εφαρμογή που πρέπει να υλοποιηθεί, το μέγεθος των δεδομένων που επεξεργάζονται και το είδος του προβλήματος που επιλύεται. Μια επιτυχημένη εφαρμογή εκμάθησης Deep Learning απαιτεί πολύ μεγάλο όγκο δεδομένων (χιλιάδες εικόνες) για την εκπαίδευση του μοντέλου, καθώς και μονάδες GPU ή μονάδες επεξεργασίας γραφικών για την ταχεία επεξεργασία των δεδομένων.

Προκειμένου να επιλέξουμε ανάμεσα στο Machine Learning και στο Deep Learning, οφείλουμε να εξετάσουμε αν έχουμε μια GPU υψηλής απόδοσης και πολλά labeled data. Αν δεν είναι διαθέσιμο κανένα από αυτά, είναι πιο λογικό να χρησιμοποιηθεί Machine Learning αντί για Deep Learning. Το Deep Learning είναι γενικά πιο περίπλοκο, επομένως θα χρειαστούν τουλάχιστον μερικές χιλιάδες εικόνες για να προκύψουν αξιόπιστα αποτελέσματα. Έχοντας

μια GPU υψηλής απόδοσης σημαίνει ότι το μοντέλο θα πάρει λιγότερο χρόνο για να αναλύσει όλες αυτές τις εικόνες (Εικόνα 1.6.i).




*Εικόνα 1.6.i: Συγκρίνοντας μια προσέγγιση Machine Learning στην κατηγοριοποίηση των οχημάτων (αριστερά) με το Deep Learning (δεξιά). (Πηγή: https://www.mathworks.com)*

## 1.7. Δημιουργία & Εκπαίδευση μοντέλων Deep Learning

Οι τρεις πιο συνηθισμένοι τρόποι με τους οποίους χρησιμοποιείται το deep learning για την ταξινόμηση αντικειμένων είναι οι εξής:

- Εκπαίδευση από την αρχή:

  Για να εκπαιδευτεί ένα βαθύ δίκτυο από την αρχή, συγκεντρώνεται ένα πολύ μεγάλο σετ δεδομένων και σχεδιάζεται μια αρχιτεκτονική δικτύου που θα μάθει τα χαρακτηριστικά και το μοντέλο. Αυτό είναι καλό για νέες εφαρμογές ή εφαρμογές που θα έχουν μεγάλο αριθμό κατηγοριών εξόδου. Αυτή είναι μια λιγότερο συνηθισμένη προσέγγιση, επειδή με το μεγάλο όγκο δεδομένων και το ρυθμό εκμάθησης, αυτά τα δίκτυα τυπικά χρειάζονται ημέρες ή εβδομάδες για να εκπαιδευτούν.

- Μεταφορά μάθησης:

  Οι περισσότερες εφαρμογές Deep Learning χρησιμοποιούν την προσέγγιση της μεταφοράς εκμάθησης (transfer learning), μια διαδικασία που περιλαμβάνει την τελειοποίηση ενός προρυθμισμένου μοντέλου. Ξεκινώντας από ένα υπάρχον δίκτυο, όπως το AlexNet και το GoogLeNet, τροφοδοτείται με νέα δεδομένα που περιέχουν προηγουμένως άγνωστες κατηγορίες (classes). Μετά από κάποιες τροποποιήσεις στο δίκτυο, μπορεί να εκτελεστεί μια νέα εργασία, όπως είναι για παράδειγμα η κατηγοριοποίηση μόνο σκύλων ή γατών αντί για 1000 διαφορετικά αντικείμενα. Αυτό έχει επίσης το πλεονέκτημα ότι χρειάζονται πολύ λιγότερα δεδομένα (επεξεργασία χιλιάδων εικόνων, αντί εκατομμυρίων), οπότε ο χρόνος υπολογισμού πέφτει σε λεπτά ή ώρες. Το transfer learning απαιτεί μια διεπαφή στα εσωτερικά του προϋπάρχοντος δικτύου, ώστε να μπορεί να τροποποιηθεί "χειρουργικά" και να ενισχυθεί για τη νέα εργασία.

- Εξαγωγή χαρακτηριστικών:

  Μια ελαφρώς λιγότερο κοινή, πιο εξειδικευμένη προσέγγιση στο Deep Learning είναι η χρήση του δικτύου ως εξαγωγέα χαρακτηριστικών (feature extractor). Δεδομένου ότι όλα τα επίπεδα έχουν επιφορτιστεί με την εκμάθηση ορισμένων χαρακτηριστικών από εικόνες, μπορούμε να τραβήξουμε αυτά τα χαρακτηριστικά από το δίκτυο ανά πάσα στιγμή

κατά τη διάρκεια της διαδικασίας κατάρτισης. Αυτά τα χαρακτηριστικά μπορούν στη συνέχεια να χρησιμοποιηθούν ως είσοδος σε ένα Μοντέλο Μηχανικής Μάθησης (Machine Learning Model), όπως Μηχανές Φορέα Υποστήριξης (Support Vector Machines – SVM).

## 1.8. Επιτάχυνση μοντέλων Deep Learning με χρήση GPU

Η εκπαίδευση ενός μοντέλου Deep Learning είναι αρκετά χρονοβόρα και μπορεί να διαρκέσει από μερικές μέρες έως και εβδομάδες. Με τη ραγδαία ανάπτυξη των χώρο των GPUs, η χρήση των τελευταίων μπορεί να επιταχύνει σημαντικά τη διαδικασία, να μειώσει τον χρόνο που απαιτείται για την κατάρτιση ενός δικτύου και μπορεί να μειώσει το χρόνο εκπαίδευσης για ένα πρόβλημα ταξινόμησης εικόνων από ημέρες σε ώρες. Επομένως, το Deep Learning γίνεται όλο και πιο εύκολο να εφαρμοστεί με την πάροδο του χρόνου έχοντας υπόψη μας ότι θα τρέχει πάνω σε GPUs και όχι τόσο συχνά σε τυπικές CPUs, τόσο για λόγους παραλληλοποίησης της εκτέλεσης όσο για λόγους ταχύτητας εκτέλεσης, ταχύτητας μνήμης κτλ.

## 1.9. Χάρτης

Στο Κεφάλαιο 2 ορίζονται κάποιες βασικές έννοιες που χρησιμοποιούνται εκτενώς στην παρούσα διπλωματική και στη συνέχεια παρουσιάζεται μία θεωρητική ανάλυση των 5 μεθόδων deep learning αναφέροντας συμπληρωματικά τα θετικά και τα αρνητικά στοιχεία της κάθε μεθόδου.

Στο Κεφάλαιο 3 παρουσιάζεται η βασική δομή του λογισμικού που χρησιμοποιήθηκε και τροποποιήθηκε καταλλήλως για την εκτέλεση των αλγορίθμων από τις μεθόδους που παρουσιάστηκαν στο Κεφάλαιο 2.

Στο Κεφάλαιο 4 παρατίθενται οι εκτελέσεις των αλγορίθμων που θεωρητικά αναπτύχθηκαν στο Κεφάλαιο 2 μαζί με τις παρεμβάσεις που πραγματοποιήθηκαν σε ορισμένες από τις παραμέτρους των αλγορίθμων, ελέγχονται τα αποτελέσματα και σχολιάζονται ανά περίπτωση τα ποιοτικά χαρακτηριστικά τους. Επίσης, παρουσιάζονται με μορφή πινάκων τα αποτελέσματα των αλγορίθμων και δίνεται έμφαση στα σφάλματα κατά την αναγνώριση/κατηγοριοποίηση. Επίσης, δίνονται οι εικόνες από το τελικό στάδιο των τρεξιμάτων και οι χρόνοι εκτέλεσης για κάθε μία από τις μεθόδους.

Στο Κεφάλαιο 5 γίνεται μία συζήτηση πάνω στα συμπεράσματα που προκύπτουν από τα αποτελέσματα του Κεφαλαίου 4 και γίνεται μία πρόταση για την αποδοτική χρήση των μεθόδων Deep Learning σε μία κοινωνία πρακτόρων.

# Κεφάλαιο 2ο: Βιβλιογραφία/Υπόβαθρο

## 2.1. Ορισμοί Εννοιών:

Σε αυτή την ενότητα παρατίθενται οι ορισμοί των βασικών εννοιών που θεωρήθηκαν σημαντικές και χρησιμοποιούνται εκτενώς στα επόμενα κεφάλαια.

### 2.1.1. Βαθιά Γνώση (Deep Learning):

Η Βαθιά Γνώση ή Βαθιά Μάθηση ή Βαθιά Εκμάθηση είναι μια κλάση των αλγορίθμων Μηχανικής Μάθησης που:

- Χρησιμοποιούν μία αλληλουχία πολλών στρωμάτων μη γραμμικών μονάδων επεξεργασίας για την εξαγωγή και τον μετασχηματισμό χαρακτηριστικών. Κάθε διαδοχική στρώση χρησιμοποιεί ως είσοδο την έξοδο από το προηγούμενο επίπεδο. Οι αλγόριθμοι μπορούν να εποπτεύονται ή να μην εποπτεύονται και οι εφαρμογές να περιλαμβάνουν ανάλυση προτύπων (χωρίς εποπτεία) και ταξινόμηση (με εποπτεία).

- Βασίζονται στην (μη επιτηρούμενη) εκμάθηση πολλαπλών επιπέδων χαρακτηριστικών ή παραστάσεων των δεδομένων. Χαρακτηριστικά υψηλότερου επιπέδου προέρχονται από χαρακτηριστικά χαμηλότερου επιπέδου για να σχηματίσουν μια ιεραρχική αναπαράσταση.

- Αποτελούν μέρος του ευρύτερου τομέα εκμάθησης μηχανών των μαθησιακών αναπαραστάσεων των δεδομένων.

- Μαθαίνουν πολλαπλά επίπεδα παραστάσεων που αντιστοιχούν σε διαφορετικά επίπεδα αφαίρεσης. Τα επίπεδα αποτελούν μια ιεραρχία των εννοιών.

### 2.1.2. Πολυπρακτορικό Σύστημα (Multi-Agent System):

Ένα Πολυπρακτορικό Σύστημα (Multi-Agent System – MAS) είναι ένα υπολογιστικό σύστημα που αποτελείται από πολλαπλούς αλληλοεπιδρώντες ευφυείς πράκτορες μέσα σε ένα περιβάλλον. Τα συστήματα πολλών πρακτόρων μπορούν να χρησιμοποιηθούν για την επίλυση προβλημάτων που είναι δύσκολο ή αδύνατο για ένα μεμονωμένο πράκτορα ή ένα μονολιθικό σύστημα να επιλυθούν.

Η νοημοσύνη μπορεί να περιλαμβάνει κάποια μεθοδολογική, λειτουργική, διαδικαστική προσέγγιση, αλγοριθμική αναζήτηση ή ενίσχυση μάθησης. Παρόλο που υπάρχει σημαντική επικάλυψη, ένα σύστημα πολλαπλών πρακτόρων δεν είναι πάντα το ίδιο με ένα μοντέλο που βασίζεται σε έναν πράκτορα.

Ο στόχος ενός τέτοιου μοντέλου είναι να αναζητήσει μια επεξηγηματική εικόνα της συλλογικής συμπεριφοράς των πρακτόρων (οι οποίοι δεν χρειάζεται απαραίτητα να είναι "έξυπνοι") υπακούοντας σε απλούς κανόνες, συνήθως σε φυσικά συστήματα, παρά στην επίλυση συγκεκριμένων πρακτικών ή μηχανικών προβλημάτων.

### 2.1.3. Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks)

Τα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks – ANN) είναι υπολογιστικά συστήματα εμπνευσμένα από τα βιολογικά νευρωνικά δίκτυα που αποτελούν έναν τυπικό ζωικό εγκέφαλο. Αυτά τα συστήματα μαθαίνουν και σταδιακά βελτιώνουν την απόδοσή τους ώστε να εκτελούν διεργασίες εξετάζοντας παραδείγματα, γενικά χωρίς να έχει προηγηθεί προγραμματισμός για συγκεκριμένες εργασίες. Για παράδειγμα, στην αναγνώριση εικόνων μπορεί να μάθουν να εντοπίζουν εικόνες που περιέχουν γάτες αναλύοντας παραδείγματα εικόνων που έχουν επισημανθεί με το χέρι ως "γάτα" ή "όχι γάτα" και χρησιμοποιώντας τα αναλυτικά αποτελέσματα για τον εντοπισμό των γατών σε άλλες εικόνες. Χρησιμοποιούνται συχνά σε εφαρμογές που είναι δύσκολο να εκφραστούν σε έναν παραδοσιακό αλγόριθμο υπολογιστή, χρησιμοποιώντας προγραμματισμό βασισμένο σε κανόνες.

Ένα Νευρωνικό Δίκτυο βασίζεται σε μια συλλογή συνδεδεμένων μονάδων που ονομάζονται Τεχνητοί Νευρώνες (Artificial Neurons), αντίστοιχοι με τους άξονες σε έναν βιολογικό εγκέφαλο. Κάθε σύνδεση/σύναψη μεταξύ των νευρώνων μπορεί να μεταδώσει ένα σήμα σε άλλο νευρώνα. Ο υποδοχέας νευρώνας μπορεί να επεξεργαστεί το σήμα και στη συνέχεια να σηματοδοτήσει τους προς τα κάτω νευρώνες που είναι συνδεδεμένοι με αυτόν. Οι νευρώνες μπορεί να έχουν κατάσταση που γενικά αναπαρίσταται με πραγματικούς αριθμούς, αλλά τυπικά με τιμές μεταξύ των 0 και 1. Οι νευρώνες και οι συνάψεις μπορεί επίσης να έχουν ένα βάρος που ποικίλλει ανάλογα με την εκμάθηση, που μπορεί να αυξήσει ή να μειώσει τη δύναμη του σήματος που στέλνει στους προς τα κάτω νευρώνες. Περαιτέρω, μπορεί να έχουν ένα κατώτατο όριο τέτοιο ώστε αν το συνολικό σήμα είναι κάτω από (ή παραπάνω από) αυτό το επίπεδο να είναι το μεταγενέστερο σήμα που αποστέλλεται.

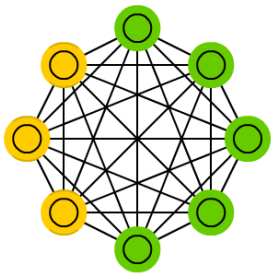
Τυπικά, οι νευρώνες οργανώνονται σε στρώματα. Τα διαφορετικά στρώματα μπορούν να εκτελούν διαφορετικά είδη μετασχηματισμών στις εισόδους τους. Τα σήματα ταξιδεύουν από την πρώτη (είσοδο) έως την τελευταία στρώση (έξοδο), ενδεχομένως μετά από την πολλαπλή διασταύρωση των στρωμάτων.

Ο αρχικός στόχος της προσέγγισης των νευρωνικών δικτύων ήταν να λυθούν τα προβλήματα με τον ίδιο τρόπο που θα μπορούσε να επιλύσει ένας ανθρώπινος εγκέφαλος. Με την πάροδο του χρόνου, η προσοχή επικεντρώθηκε στην αντιστοίχιση συγκεκριμένων πνευματικών ικανοτήτων, οδηγώντας σε αποκλίσεις από τη βιολογία, όπως είναι η οπίσθια προώθηση ή η μεταφορά πληροφοριών προς την αντίθετη κατεύθυνση και η προσαρμογή του δικτύου ώστε να αντικατοπτρίζει αυτές τις πληροφορίες.

## 2.2. Μέθοδοι:

Σε αυτή την ενότητα θα αναλύσουμε και θα περιγράψουμε τις επικρατέστερες μεθόδους που χρησιμοποιούνται στο Deep Learning ενώ παράλληλα θα γίνονται αναφορές για τα πλεονεκτήματα και τα μειονεκτήματα της καθεμίας.

### 2.2.1. Deep Boltzmann Machine (DBM)

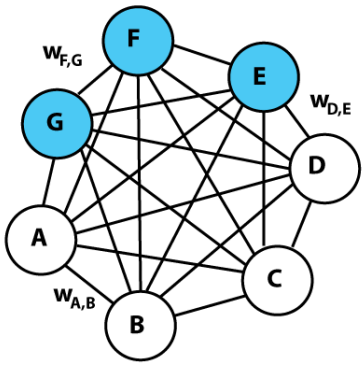


*Εικόνα 2.2.1.i: Δομή ενός Boltzmann Machine*

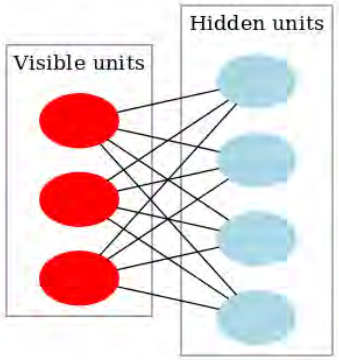
Η Βαθιά Μηχανή Boltzmann (Deep Boltzmann Machine – DBM) είναι ένας τύπος δυαδικών ζευγών ενός τυχαίου πεδίου Markov με πολλαπλά στρώματα κρυφών τυχαίων μεταβλητών με ένα δίκτυο συμμετρικά συζευγμένων στοχαστικών δυαδικών μονάδων. Περιλαμβάνει επίσης ένα σύνολο ορατών μονάδων (visible units) $v_i \in \{0,1\}^c$ και μια σειρά κρυφών μονάδων (hidden units) $h_i^{(1)} \in \{0,1\}^{C_1}, h_i^{(2)} \in \{0,1\}^{C_2}, \cdots h_i^{(2)} \in (0,1)^{C_L}$. Η πιθανότητα που αποδίδεται στο διάνυσμα $v_i$ είναι η εξής:

$$p(v_i) = \frac{1}{Z}\sum_{h_i} e^{\sum_{ab} W_{ab}^{(1)} v_i h_b^{(1)} + \sum_{bc} W_{bc}^{(2)} h_b^{(1)} h_i^{(2)} + \sum_{cd} W_{cd}^{(3)} h_c^{(2)} h_d^{(3)}}$$

όπου $h = \left\{h_i^{(1)}, h_i^{(2)}, h_i^{(3)}\right\}$ το σύνολο hidden units και $\vartheta = \left\{W_i^{(1)}, W_i^{(2)}, W_i^{(3)}\right\}$ είναι οι αντίστοιχες παράμετροι του μοντέλου που αντιπροσωπεύουν τις αλληλεπιδράσεις μεταξύ visible–hidden και hidden–hidden units. Αν $w_i^{(2)} = w_i^{(3)} = 0$, τότε το δίκτυο είναι γνωστό ως Restricted Boltzmann Machine [1][7][9].




*Εικόνα 2.2.1.ii: Γραφική αναπαράσταση ενός Boltzmann Machine. Με μπλε χρώμα απεικονίζονται οι κρυφές μονάδες και με άσπρο χρώμα οι ορατές μονάδες.*




*Εικόνα 2.2.1.iii: Γραφική αναπαράσταση ενός Restricted Boltzmann Machine*

Το σχήμα της Εικόνας 2.2.1.ii αναπαριστά τη γραφική αρχιτεκτονική ενός Boltzmann Machine. Είναι προφανές ότι κάθε μη κατευθυνόμενη ακμή αντιπροσωπεύει εξάρτηση. Εδώ υπάρχουν τρία hidden units και τέσσερα visible units. Το σχήμα της Εικόνας 2.2.1.iii αναπαριστά την γραφική αναπαράσταση ενός Restricted Boltzmann Machine που θα αναλυθεί στην παράγραφο 2.2.2. Από το σχήμα της Εικόνας 2.2.1.iii συμπεραίνουμε ότι οι τέσσερις μπλε

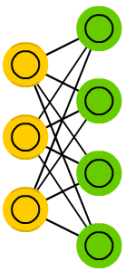
μονάδες αντιπροσωπεύουν hidden units και οι τρεις κόκκινες μονάδες αντιπροσωπεύουν visible units. Αυτό αποδεικνύει ότι το Restricted Boltzmann Machine έχει συνδέσεις ή εξαρτήσεις μόνο μεταξύ των hidden units και των visible units και δεν υπάρχει σύνδεση μεταξύ των hidden–hidden μονάδων και των visible–visible units.

Το DBM μαθαίνει σύνθετες και αφηρημένες εσωτερικές αναπαραστάσεις της εισόδου σε διάφορα μοντέλα όπως αναγνώριση αντικειμένου ή αναγνώριση ομιλίας, χρησιμοποιώντας επαρκή labeled data για την τελειοποίηση (fine-tune) των αναπαραστάσεων που έχουν δημιουργηθεί με τη χρήση μιας μεγάλης ποσότητας μη επισημασμένων αισθητηριακών δεδομένων εισόδου (unlabeled sensory input data). Τα DBMs υιοθετούν επίσης τη διαδικασία συμπερασμάτων και κατάρτισης και προς τις δύο κατευθύνσεις, δηλαδή από πάνω προς τα κάτω (top-down) και από κάτω προς τα πάνω (bottom-up), τα οποία επιτρέπουν στα DBM να αποκαλύψουν καλύτερα τις αναπαραστάσεις των διφορούμενων και πολύπλοκων δομών εισόδου. Η ταχύτητα των DBM περιορίζει την απόδοση και τη λειτουργικότητά τους.

Ένα από πλεονεκτήματα των DBMs είναι η ικανότητά τους να μαθαίνουν αποτελεσματικές αναπαραστάσεις σύνθετων δεδομένων με αποτελεσματική τεχνική προ-εκπαίδευσης ανά επίπεδο. Το μεγαλύτερο πλεονέκτημα των DBMs είναι ότι θα μπορούσαν να εκπαιδευτούν ακόμη και με μη επισημασμένα δεδομένα (unlabeled data) και να ρυθμιστεί (fine-tuned) με τα πιθανά δεδομένα ορίων για μια συγκεκριμένη εφαρμογή. Τα DBM θα μπορούσαν επίσης να προβλέψουν την αβεβαιότητα της αμφιλεγόμενης εισόδου με τον τρόπο της ανάλυσης της προσεγγιστικής διαδικασίας υπολογισμού που εντοπίστηκε σε αυτά. Εφαρμόζοντας τη διαδικασία της κατά προσέγγισης κλίσης σε όλα τα επίπεδα, οι παράμετροι σε αυτό θα μπορούσαν να βελτιστοποιηθούν, πράγμα που με τη σειρά του διευκολύνει την εκμάθηση καλύτερης παραγωγής μοντέλων.

Το μειονέκτημα των DBMs είναι ότι είναι καλά για θεωρητικό σκοπό και όχι για γενικό υπολογιστικό μέσο και σταματούν να μαθαίνουν σωστά όταν το μηχάνημα κλιμακώνεται σε κάτι μεγαλύτερο από ένα μικρό μηχάνημα. Η προσεγγιστική διαδικασία υπολογισμού που ακολουθείται στα DBMs είναι σχεδόν 50 φορές πιο αργή από ότι ακολουθείται στα Deep Belief Networks (DBNs). Ως εκ τούτου, τα DBM δεν είναι κατάλληλα για μεγάλα σύνολα δεδομένων και περιορίζεται τη χρήση τους για εργασίες όπως είναι η αναπαράσταση χαρακτηριστικών [1][4].

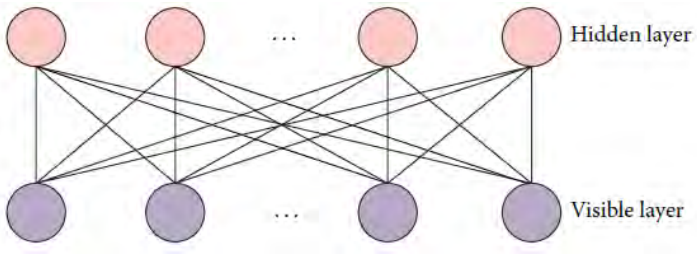
### 2.2.2. Restricted Boltzmann Machine (RBM)



*Εικόνα 2.2.2.i: Δομή ενός Restricted Boltzmann Machine*

Η περιορισμένη μηχανή Boltzmann (Restricted Boltzmann Machine – RBM) είναι ένα είδος τυπικού νευρωνικού δικτύου. Το RBM αποτελείται από τη σύνδεση του visible layer και του hidden layer, αλλά δεν υπάρχει σύνδεση μεταξύ των μονάδων του hidden layer καθώς επίσης και μεταξύ των μονάδων του visible layer [9]. Στο σχήμα της Εικόνας 2.2.2.ii, η εκπαίδευση του RBM χρησιμοποίησε τη μη επιτηρούμενη άπληστη μέθοδο βήμα προς βήμα. Δηλαδή, κατά την εκπαίδευση, η χαρακτηριστική τιμή του visible layer αντιστοιχίζει (maps) στο κρυμμένο στρώμα, τότε το visible layer μπορεί να ανακατασκευαστεί μέσω του hidden layer. Αυτή η νέα ορατή χαρακτηριστική τιμή αντιστοιχίζεται ξανά στο hidden layer και έπειτα αποκτά ένα νέο hidden layer [5][11][14].

Ο κύριος σκοπός του είναι να αποκτήσει τη γενετική ισχύ (generative power value). Έτσι, τα κύρια χαρακτηριστικά του RBM είναι τα χαρακτηριστικά ενεργοποίησης του στρώματος που εισάγεται στο επόμενο στρώμα ως δεδομένα εκπαίδευσης και ως εκ τούτου η ταχύτητα εκμάθησης είναι γρήγορη. Αυτή είναι μία επίπεδο προς επίπεδο αποτελεσματική θεωρία στρατηγικής μάθησης.




*Εικόνα 2.2.2.ii: Γραφική αναπαράσταση ενός Restricted Boltzmann Machine*

Στο σχήμα της Εικόνας 2.2.2.ii παρατηρούμε ότι τα DBN στοιβάζονται από κάτω προς τα πάνω από στο RBM. Με τη χρήση του Gauss-Bernoulli RBM και του Bernoulli-Bernoulli RBM για σύνδεση, η έξοδος του κάτω στρώματος είναι η είσοδος για το ανώτερο στρώμα [6].

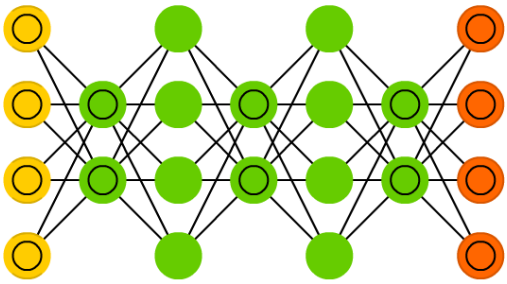
Στο Bernoulli RBM, τα visible και hidden units των στρωμάτων είναι δυαδικά: $v \in \{0,1\}^D$ και $h \in \{0,1\}^K$. Τα $D$ και $K$ παρουσιάζουν αριθμούς μονάδων visible και hidden στρωμάτων. Στο Gaussian RBM, τα visible units είναι ένας πραγματικός αριθμός: $v \in R^D$. Η κοινή πιθανότητα (joint probability) των $V$ και $h$ εκφράζεται ως εξής: $P(v,h) = \frac{1}{Z} exp\big(-E(v,h)\big)$. Το $Z$ είναι μία κανονικοποιημένη σταθερά και το $E(v,h)$ είναι η εξίσωση ενέργειας. Στο Bernoulli RBM, η εξίσωση ενέργειας είναι η εξής:

$$E(v,h) = -\sum_{i=1}^{D}\sum_{j=1}^{K} w_{ij} v_i h_j - \sum_{i=1}^{D} b_i v_i - \sum_{j=1}^{K} a_j h_j$$

όπου $w_{i,j}$ εκφράζει την ποιότητα των μη συνδεδεμένων ορατών κόμβων του στρώματος $V_i$ και των hidden units του στρώματος $h_j$ και $a$ και $b$ είναι τα implicit bias των visible και των hidden units. Για Gaussian RBM, η εξίσωση ενέργειας είναι η εξής:

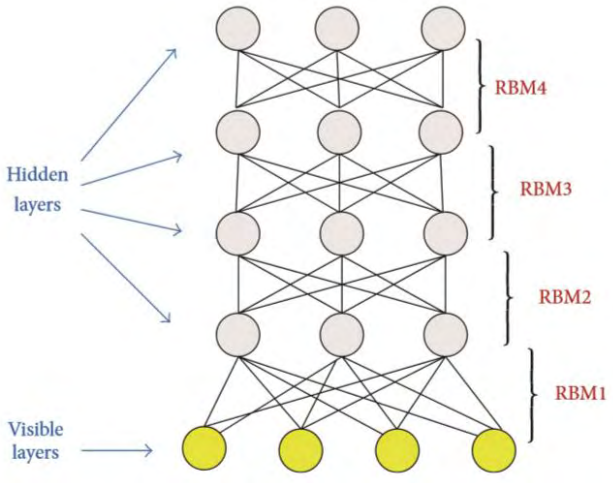
$$E(v,h) = \sum_{i=1}^{D} \frac{(v_i - b_i)^2}{2} - \sum_{i=1}^{D}\sum_{j=1}^{K} w_{ij} v_i h_j - \sum_{j=1}^{K} a_j h_j$$

### 2.2.3. Deep Belief Networks (DBN)



*Εικόνα 2.2.3.i: Δομή ενός Deep Belief Network*

Τα Δίκτυα Βάθους Πεποιθήσεων (Deep Belief Networks – DBN) είναι πολύπλοκοι κατευθυνόμενοι άκυκλοι γράφοι, οι οποίοι σχηματίζονται από μια σειρά αρχιτεκτονικές Περιορισμένων Μηχανών Boltzmann (Restricted Boltzmann Machines – RBM). Το DBN θα μπορούσε να εκπαιδευτεί με την εκπαίδευση των RBMs ανά επίπεδο από κάτω προς τα πάνω. Δεδομένου ότι τα RBMs μπορούν να εκπαιδευτούν ταχέως μέσω του αλγόριθμου πολυεπίπεδης αντίθεσης απόκλισης (layered contrast divergence algorithm), η εκπαίδευσή τους αποφεύγει έναν υψηλό βαθμό πολυπλοκότητας, η οποία απλοποιεί τη διαδικασία εκπαίδευσης καθενός RBM. Μελέτες στα DBNs έδειξαν ότι μπορεί να επιλύσουν την χαμηλή ταχύτητα σύγκλισης και τα τοπικά προβλήματα βελτιστοποίησης στον παραδοσιακό αλγόριθμο της προς τα πίσω διάδοσης στην εκπαίδευση πολυεπίπεδων Νευρωνικών Δικτύων [1][8]. Το σχήμα της Εικόνας 2.2.3.ii αναπαριστά την αρχιτεκτονική του δικτύου DBN, στο οποίο τα RBMs εκπαιδεύονται ανά επίπεδο από κάτω προς τα πάνω.



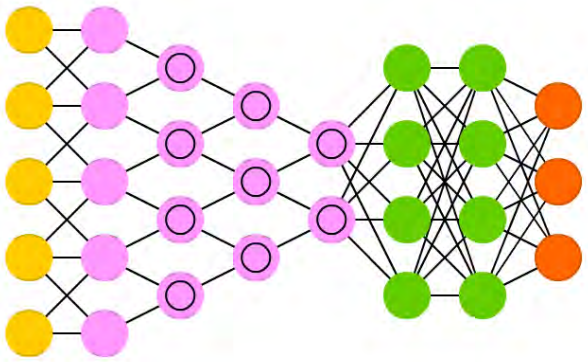

*Εικόνα 2.2.3.ii: Γραφική αναπαράσταση ενός Deep Belief Network*

Στα πλεονεκτήματα του μοντέλου Deep Belief Network περιλαμβάνεται η ικανότητα να μαθαίνουν ένα βέλτιστο σύνολο παραμέτρων γρήγορα ακόμα και για τα μοντέλα που περιέχουν πολύ μεγάλο πλήθος παραμέτρων και τα επίπεδα με μη γραμμικότητα μέσω του άπληστου αλγόριθμου ανά επίπεδο. Τα DBNs χρησιμοποιούν μια μέθοδο προ-εκπαίδευσης χωρίς επιτήρηση (unsupervised pre-training) ακόμη και για πολύ μεγάλες μη επισημασμένες βάσεις

δεδομένων. Τα DBNs θα μπορούσαν επίσης να υπολογίσουν τις τιμές εξόδου των μεταβλητών στο χαμηλότερο επίπεδο χρησιμοποιώντας προσεγγιστική διαδικασία υπολογισμού.

Στα μειονεκτήματα των DBN περιλαμβάνεται ο περιορισμός της προσέγγισης σε ένα ενιαίο βήμα προς τα πάνω. Η άπληστη διαδικασία μαθαίνει μόνο τα χαρακτηριστικά ενός επιπέδου κάθε φορά και ποτέ δεν αναπροσαρμόζεται εκ νέου με τα άλλα επίπεδα ή τις παραμέτρους του δικτύου. Ο αλγόριθμος Wake-Sleep που προτείνεται από την Hinton για τα DBNs είναι πολύ αργός και αναποτελεσματικός, παρόλο που πραγματοποιεί fine-tuning καθολικά [1][5][6].

### 2.2.4. Deep Convolutional Neural Networks (CNN)



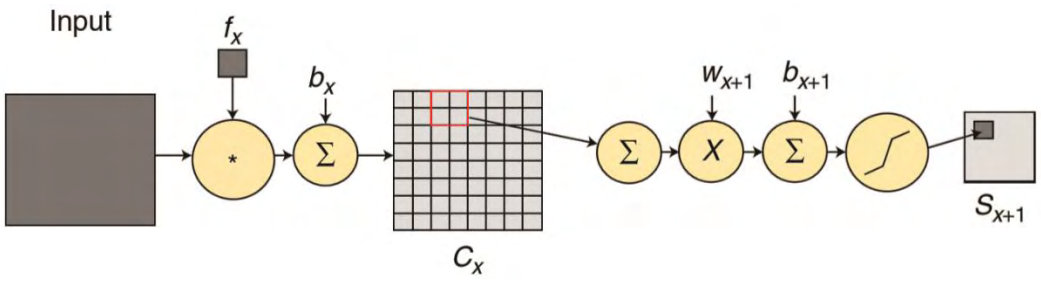
*Εικόνα 2.2.4.i: Δομή ενός Deep Convolutional Neural Network*

Τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks – CNNs) είναι μια οικογένεια Νευρωνικών Δικτύων πολλαπλών επιπέδων ειδικά σχεδιασμένων για χρήση σε δισδιάστατα δεδομένα, όπως οι εικόνες και το βίντεο. Τα CNNs επηρεάζονται από προηγούμενες εργασίες σε Νευρωνικά Δίκτυα Χρονικής Καθυστέρησης (Time-delay Neural Networks – TDNN), τα οποία μειώνουν τις απαιτήσεις της υπολογιστικής μάθησης με το να μοιράζονται τα βάρη σε μία χρονική διάσταση και προορίζονται για επεξεργασία ομιλίας και χρονοσειρών. Τα CNNs είναι η πρώτη πραγματικά επιτυχημένη προσέγγιση Deep Learning όπου πολλά επίπεδα ιεραρχίας εκπαιδεύονται με επιτυχία με έναν εύρωστο τρόπο.

Το CNN είναι μια επιλογή τοπολογίας ή αρχιτεκτονικής που αξιοποιεί τις χωρικές σχέσεις για να μειώσει τον αριθμό των παραμέτρων που πρέπει να γνωστοποιηθούν και επομένως να βελτιώνει τη γενική εκπαίδευση προς τα εμπρός τροφοδοσίας και προς τα πίσω προώθησης/διάδοσης (feed-forward back propagation training). Τα CNNs προτείνονταν ως ένα βαθύ πλαίσιο μάθησης που βασίζεται σε ελάχιστες απαιτήσεις προεπεξεργασίας δεδομένων.

Στα CNNs, μικρά τμήματα της εικόνας (που μεταγλωττίζονται από ένα τοπικό πεδίο υποδοχής) αντιμετωπίζονται ως εισροές στο χαμηλότερο στρώμα της ιεραρχικής δομής. Οι πληροφορίες γενικώς διαδίδονται μέσω των διαφόρων επιπέδων του δικτύου, όπου σε κάθε στρώμα εφαρμόζεται ψηφιακό φιλτράρισμα προκειμένου να αποκτηθούν τα κυριότερα χαρακτηριστικά των δεδομένων που παρατηρήθηκαν. Η μέθοδος παρέχει ένα επίπεδο αμετάβλητης μετατόπισης, κλίμακας και περιστροφής καθώς το τοπικό πεδίο δεκτικότητας επιτρέπει στο νευρώνα ή την μονάδα επεξεργασίας να έχει πρόσβαση σε στοιχειώδη χαρακτηριστικά όπως προσανατολισμένες άκρες ή γωνίες.

Πιο συγκεκριμένα, η εικόνα εισόδου συμπλέκεται με ένα σύνολο N μικρών φίλτρων των οποίων οι συντελεστές είτε έχουν εκπαιδευτεί είτε έχουν προκαθοριστεί χρησιμοποιώντας ορισμένα κριτήρια. Έτσι, το πρώτο (χαμηλότερο) στρώμα του δικτύου αποτελείται από "χάρτες χαρακτηριστικών" (feature maps) που είναι το αποτέλεσμα των διαδικασιών συνέλιξης, μαζί με επιπρόσθετα bias και ενδεχομένως με συμπίεση ή με εξομάλυνση των χαρακτηριστικών. Αυτό το αρχικό στάδιο ακολουθείται από μια υποδειγματοληψία (τυπικά μια διαδικασία μέτρησης 2x2) που μειώνει περαιτέρω τη διαστατικότητα (dimensionality) και προσφέρει κάποια ευρωστία στις χωρικές μετατοπίσεις (Εικόνα 2.2.4.ii). Ο χάρτης χαρακτηριστικών εφόσον έχει υποστεί δειγματοληψία λαμβάνει στη συνέχεια ένα βάρος και εκπαιδεύσιμα bias και τελικά διαδίδεται μέσω μιας λειτουργίας ενεργοποίησης. Ορισμένες παράμετροι υπάρχουν με ένα μόνο χάρτη ανά στρώμα ή αθροίσεις πολλαπλών χαρτών.

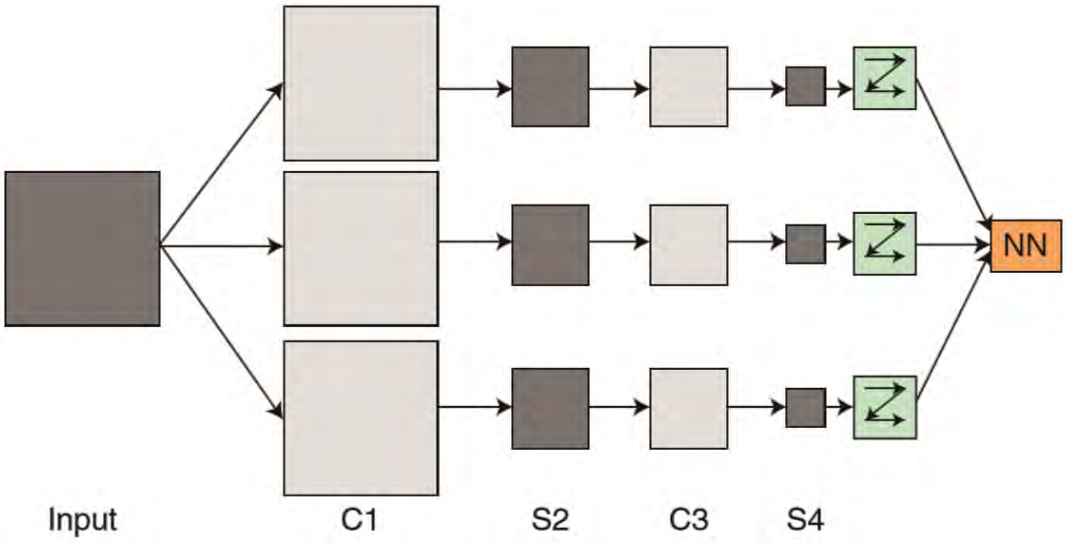



*Εικόνα 2.2.4.ii: Η διαδικασία συνέλιξης και υποδειγματοληψίας: η διαδικασία συνέλιξης συνίσταται από τη συνέλιξη μιας εισόδου (εικόνα για το πρώτο στάδιο ή το χάρτη χαρακτηριστικών για τα μεταγενέστερα στάδια) με ένα εκπαιδεύσιμο φίλτρο $f_x$ και στη συνέχεια την προσθήκη ενός εκπαιδεύσιμου bias $b_x$ για την παραγωγή του στρώματος συνέλιξης $C_x$. Η υποδειγματοληψία συνίσταται από το άθροισμα μιας γειτονιάς τεσσάρων εικονοστοιχείων, σταθμίζοντας με κλίμακα $w_{x+1}$, προσθέτοντας εκπαιδεύσιμα bias $b_{x+1}$ και με τη χρήση μιας σιγμοειδούς συνάρτησης να παραχθεί ένας κατά προσέγγιση 2x μικρότερος χάρτης χαρακτηριστικών $S_{x+1}$.*

Όταν η βαρύτητα είναι μικρή, η λειτουργία ενεργοποίησης είναι σχεδόν γραμμική και το αποτέλεσμα είναι θόλωμα της εικόνας. Άλλα βάρη μπορούν να προκαλέσουν την ενεργοποίηση εξόδου ώστε να μοιάζει με μια συνάρτηση AND ή OR. Αυτές οι έξοδοι σχηματίζουν έναν νέο χάρτη χαρακτηριστικών ο οποίος στη συνέχεια περνάει μέσω μιας άλλης ακολουθίας ροής συνέλιξης, υποδειγματοληψίας και ενεργοποίησης, όπως απεικονίζεται στο σχήμα της Εικόνας 2.2.4.iii.

Αυτή η διαδικασία μπορεί να επαναληφθεί σε έναν αυθαίρετο αριθμό φορών. Θα πρέπει να σημειωθεί ότι τα επόμενα στρώματα μπορούν να συνδυάσουν ένα ή περισσότερα από τα προηγούμενα στρώματα.

Τα CNNs δημιουργούν την αμετάβλητη διατύπωση αντικειμένων μεταφράσεων με μια μέθοδο που ονομάζεται "συγκέντρωση χαρακτηριστικών" (τα στρώματα S στο σχήμα της Εικόνας 2.2.4.iii). Ωστόσο, η συγκέντρωση χαρακτηριστικών είναι χειροποίητη από τον οργανωτή του δικτύου, που δεν έχει εκπαιδευτεί ή έχει μάθει από το σύστημα. Στα CNNs η συγκέντρωση "συντονίζεται" από παραμέτρους στη διαδικασία εκμάθησης αλλά ο βασικός μηχανισμός (ο συνδυασμός των εισροών στα στρώματα S) καθορίζεται από τον σχεδιαστή δικτύου. Τέλος, στο τελικό στάδιο της διαδικασίας, οι ενεργοποιήσεις εξόδων προωθούνται σε ένα συμβατικό ανεστραμμένο δίκτυο που παράγει την τελική έξοδο του συστήματος.




*Εικόνα 2.2.4.iii: Εννοιολογικό παράδειγμα Συνελικτικού Νευρωνικού Δικτύου. Η εικόνα εισόδου συμπλέκεται με τρία εκπαιδεύσιμα φίλτρα και bias (όπως στο Σχήμα 8) για την παραγωγή τριών χαρτών χαρακτηριστικών στο επίπεδο C1. Κάθε ομάδα τεσσάρων εικονοστοιχείων στους χάρτες χαρακτηριστικών προστίθενται, σταθμίζονται, συνδυάζονται με bias και περνούν μέσω μιας σιγμοειδούς συνάρτησης για να παράγουν τους τρεις χάρτες χαρακτηριστικών στο S2. Αυτά φιλτράρονται ξανά για να παράγουν το επίπεδο C3. Η ιεραρχία παράγει τότε το S4 κατά τρόπο ανάλογο με το S2. Τελικά, αυτές οι τιμές εικονοστοιχείων είναι ραστεροποιημένες και αναπαριστάμενες ως ένα διάνυσμα εισόδου στο "συμβατικό" Νευρωνικό Δίκτυο στην έξοδο.*

Η στενή σχέση μεταξύ των επιπέδων και των χωρικών πληροφοριών σε CNNs τις καθιστά κατάλληλες για επεξεργασία και κατανόηση της εικόνας και γενικά αποδίδουν καλά στην αυτόνομη εξαγωγή χαρακτηριστικών από εικόνες. Σε μερικές περιπτώσεις τα φίλτρα Gabor έχουν χρησιμοποιηθεί ως αρχικό στάδιο προ-επεξεργασίας για να μιμηθούν την ανθρώπινη οπτική απόκριση στην οπτική διέγερση. Σε πιο πρόσφατες εργασίες, οι ερευνητές έχουν εφαρμόσει τα CNNs σε διάφορα προβλήματα μηχανικής μάθησης, όπως ανίχνευση προσώπου, ανάλυση εγγράφων και ανίχνευση ομιλίας. Τα CNNs πρόσφατα εκπαιδεύτηκαν με στόχο τη χρονική συνοχή για τη χειρισμό της συνοχής μεταξύ πλαισίων σε βίντεο, αν και αυτός ο στόχος δεν χρειάζεται να είναι συγκεκριμένος για τα CNNs [3][13].

Μία προσέγγιση για τη δομή των CNNs θα μπορούσε να είναι η ακόλουθη με την κατάλληλη τοποθέτηση των στρωμάτων που αναφέρονται παρακάτω:

- <u>Ένα στρώμα περιστροφής (convolutional layer):</u>

  Σε αυτό το στρώμα, κατά τη διάρκεια της φάσης εμπρόσθιου υπολογισμού, τα δεδομένα εισόδου είναι συνωστισμένα με ορισμένα φίλτρα. Η έξοδος της συνέλιξης ονομάζεται συνήθως χάρτης χαρακτηριστικών (feature map). Δείχνει που μπορούν να εντοπιστούν τα χαρακτηριστικά που ανιχνεύονται από το φίλτρο στα δεδομένα εισόδου.

  Στο σχήμα της Εικόνας 2.2.4.iv, παρέχουμε ένα παράδειγμα συνελικτικού επιπέδου όπου το διάνυσμα εισόδου X αντιπροσωπεύεται ως πίνακας (δηλαδή $X = (x_{i,j}) \in \mathbb{R}^{t \times t}$, όπου t είναι ο μικρότερος τετραγωνικός ακέραιος μεγαλύτερος από το μέγεθος n του X ως διάνυσμα) και γεμάτος με μηδενικά γύρω από τα όρια. Στόχος είναι να ελέγξουμε το μέγεθος της εξόδου. Οι τιμές εξόδου μπορούν να εκφράζονται ως $\sum_{a=1}^{m} \sum_{b=1}^{m} w_{a,b} x_{i+a,j+b}$, όπου $w_{a,b}$ δηλώνει τα βάρη του φίλτρου που θεωρούνται ως πίνακας m x m.

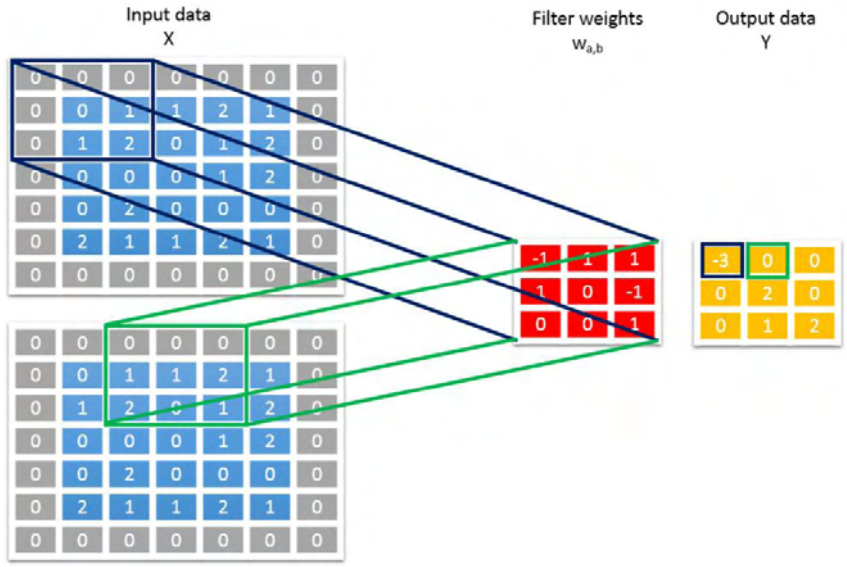
Κατά τη διάρκεια του προς τα πίσω υπολογισμού, τα βάρη του φίλτρου “μαθαίνονται/γνωστοποιούνται”, προσπαθώντας να ελαχιστοποιήσουν τη συνολική απώλεια.

- Ένα στρώμα μέγιστης συγκέντρωσης (Max Pooling layer):

  Αυτό είναι ένα επίπεδο υποδειγματοληψίας. Ο χάρτης χαρακτηριστικών χωρίζεται σε περιοχές και η έξοδος αυτού του στρώματος είναι η συνένωση των μέγιστων τιμών όλων αυτών των περιοχών. Αυτά τα στρώματα μπορούν να βοηθήσουν στη μείωση της πολυπλοκότητας του υπολογισμού και να ενισχύσουν την ευρωστία του μοντέλου σε σχέση με τη μετάφραση της εισόδου.

- Ένα στρώμα SoftMax (SoftMax layer):

  Προστίθεται στο επάνω μέρος των προηγούμενων στοιβαγμένων στρωμάτων. Μετατρέπει τις βαθμολογίες από το προηγούμενο στρώμα σε κατανομή πιθανότητας στις κλάσεις.

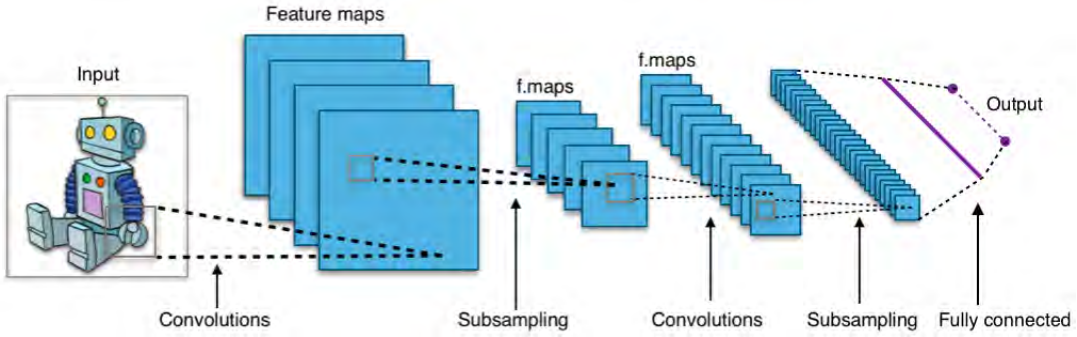



*Σχήμα 2.2.4.iv: Παράδειγμα ενός συνελικτικού επιπέδου όπου n = 25, t = 5 και m = 3.*

Η εκμάθηση των φίλτρων επιτρέπει την εξαγωγή χαρακτηριστικών υψηλού επιπέδου από τα δεδομένα. Αυτό το βήμα μπορεί επομένως να χρησιμοποιηθεί ως μείωση της διαστασιολόγησης ή σε μια τεχνική επιλογής σημείων ενδιαφέροντος (Points Of Interest – POI). Με βάση αυτή την παρατήρηση, θα ήταν ενδιαφέρον να εκτιμηθεί η αποδοτικότητα της λειτουργίας εξαγωγής εσωτερικών χαρακτηριστικών του CNN στην επιλογή των πιο πληροφοριακών σημείων για την πραγματοποίηση επιτυχούς επίθεσης ανάκτησης κλειδιών [2].

Ένα Συνεργατικό Νευρωνικό Δίκτυο (Convolutional Neural Network – CNN) είναι ένας τύπος αρχιτεκτονικής νευρωνικών δικτύων εμπρόσθιας προώθησης στη μηχανική μάθηση. Τα CNNs έχουν μια συλλογή μικρών νευρώνων σε πολλαπλά στρώματα που επεξεργάζονται την εικόνα εισόδου σε τμήματα που ονομάζονται ως δεκτικά πεδία (receptive fields). Η έξοδος

αυτών των συλλογών είναι κατασκευασμένη με τέτοιο τρόπο ώστε υπάρχει αλληλοεπικάλυψη των περιοχών εισόδου που δίνει μια σαφή αναπαράσταση της αρχικής εικόνας εισόδου. Η διαδικασία επαναλαμβάνεται για όλα τα επίπεδα. Τα CNNs χρησιμοποιούνται ως επί το πλείστον σε συστήματα αναγνώρισης βίντεο και εικόνας, συστήματα επεξεργασίας φυσικών γλωσσών και συστηματοποιητές. Το σχήμα της Εικόνας 2.2.4.v δείχνει το αρχιτεκτονικό διάγραμμα του CNN στο οποίο γίνεται η υποδειγματοληψία για κάθε στρώμα και οι συνδέσεις παρατάσσονται για να σχηματίσουν την πλήρως συνδεδεμένη αναπαράσταση της εικόνας.




*Εικόνα 2.2.4.v: Γραφική αναπαράσταση ενός Convolutional Neural Network*

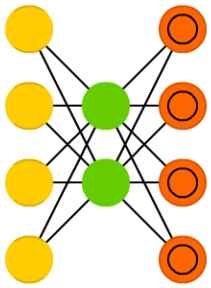
Τα πλεονεκτήματα του CNN είναι τα εξής: Πρώτον, η χρήση κοινών βαρών σε συνελικτικά στρώματα ανοίγει το δρόμο για τη χρήση του ίδιου φίλτρου για κάθε εικονοστοιχείο στο στρώμα. Στη συνέχεια, το CNN χρησιμοποιεί σχετικά μικρή προεπεξεργασία, πράγμα που σημαίνει ότι το δίκτυο CNN είναι υπεύθυνο για την εκμάθηση των φίλτρων όπου οι παραδοσιακοί αλγόριθμοι κατασκευάζονται χειροκίνητα. Τρίτον, τα CNN είναι εύκολο να εκπαιδευτούν και εξαρτώνται λιγότερο από την ανθρώπινη κατανόηση και προσπάθεια καθώς και από προηγούμενη γνώση στη σχεδίαση των χαρακτηριστικών για το μοντέλο.

Τα CNNs μπορούν επίσης να σχεδιάσουν την 2D δομή της εικόνας εισόδου, χρησιμοποιώντας τοπικές συνδέσεις και βάρη ακολουθούμενα από τεχνική συγκέντρωσης (pooling technique) η οποία με τη σειρά της έχει ως αποτέλεσμα στη μετάφραση αμετάβλητα χαρακτηριστικά.

Το κύριο πλεονέκτημα του CNN είναι ότι έχει λιγότερες παραμέτρους σε σύγκριση με τα πλήρως συνδεδεμένα δίκτυα με τον ίδιο αριθμό κρυφών μονάδων. Το πιο χαρακτηριστικό γνώρισμα του CNN είναι ότι έχει όγκο 3D νευρώνων στον οποίο οι νευρώνες είναι διατεταγμένοι σε 3 διαστάσεις, συγκεκριμένα το βάρος, το ύψος και το βάθος.

Το μειονέκτημα των CNNs είναι ότι απαιτούν τεράστια ποσότητα μνήμης για να διατηρήσουν όλα τα ενδιάμεσα αποτελέσματα του συνελικτικού στρώματος για να δώσουν ως είσοδο στο στρώμα back-propogation.

### 2.2.5. Stacked Auto-Encoders (SAE)

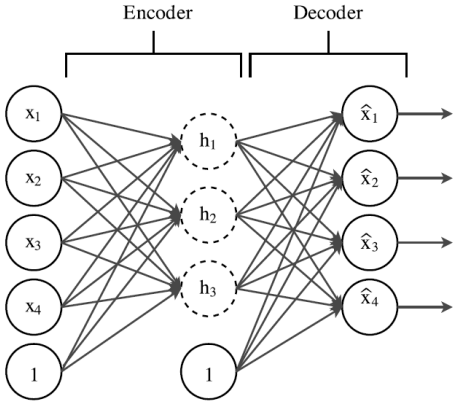


*Εικόνα 2.2.5.i: Δομή ενός δικτύου Stacked Auto-Encoders*

Οι στοιβαγμένοι αυτόματοι κωδικοποιητές (Stacked Auto-Encoders – SAE) είναι τεχνητά νευρωνικά δίκτυα με πολλά στρώματα που εκπαιδεύονται ακολουθώντας μια πολύ ειδική διαδικασία. Αυτή η διαδικασία συνίσταται στην κατάρτιση κάθε στρώματος ανεξάρτητα, χρησιμοποιώντας την έξοδο του προηγούμενου στρώματος ως είσοδο για το τρέχον. Κάθε στρώμα αποτελείται από έναν κωδικοποιητή και έναν αποκωδικοποιητή, και οι δύο είναι ένα πυκνό στρώμα (δηλαδή πλήρως συνδεδεμένο στρώμα, όπως είναι τα Restricted Boltzmann Machines που αναλύθηκαν στην παράγραφο 2.2.2) [2][9].

Ο ρόλος του κωδικοποιητή είναι να παράγει χαρακτηριστικά υψηλότερου επιπέδου από τις εισόδους. Ενώ ο ρόλος του αποκωδικοποιητή είναι να ανοικοδομήσει τις εισόδους από τα ενδιάμεσα χαρακτηριστικά γνωστά από τον κωδικοποιητή (Εικόνα 2.2.5.ii). Ένα αδιάφορο δίκτυο θα μάθει τη λειτουργία ταυτότητας (identity function). Για να αποφευχθεί μια τέτοια συμπεριφορά, μπορούμε να θεωρήσουμε ότι κάθε στρώμα πρέπει να είναι μικρότερο από το προηγούμενο. Ωστόσο, αυτό δεν είναι υποχρεωτικό. Ορισμένα εμπειρικά αποτελέσματα, έδειξαν ότι θα ήταν καλύτερα να έχουμε περισσότερες φορές νευρώνες στο πρώτο κρυμμένο στρώμα από ότι στην έξοδο ως στάδιο “προ-μάθησης” (pre-training). Με αυτόν τον τρόπο το δίκτυο θα αναγκαστεί να μάθει μια συμπιεσμένη αναπαράσταση της εισόδου.

Μόλις ολοκληρωθεί η εκπαίδευση, αφαιρείται ο αποκωδικοποιητής, ο νεοσύστατος κωδικοποιητής στοιβάζεται με τους προηγουμένως εκπαιδευμένους και η διαδικασία μπορεί να επαναληφθεί χρησιμοποιώντας την έξοδο του νέου εκπαιδευμένου στρώματος.



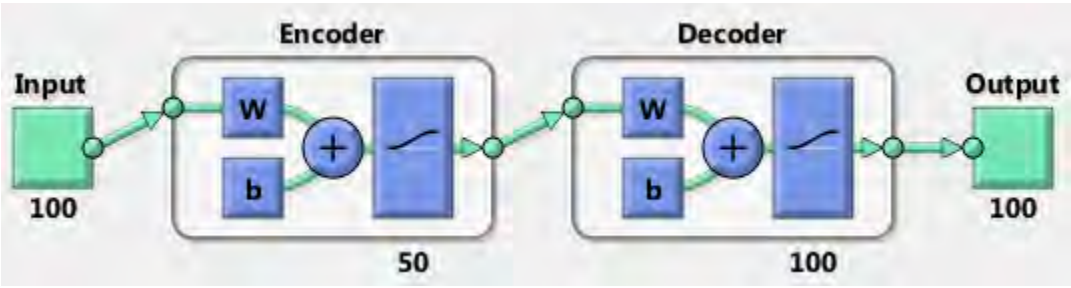

*Εικόνα 2.2.5.ii: Η εκμάθηση ενός επιπέδου Stacked Auto-Encoder. Αρχικά, η είσοδος $X = (x_0, x_1, x_2, x_3, x_4) \in \mathbb{R}^5$ κωδικοποιείται. Στη συνέχεια, το αποτέλεσμα που προκύπτει $H = (h_0, h_1, h_2, h_3, h_4) \in \mathbb{R}^5$ αποκωδικοποιείται χρησιμοποιώντας το δεύτερο στρώμα του διαγράμματος για να ανακατασκευάσει την είσοδο $\hat{X} = (\hat{x}_0, \hat{x}_1, \hat{x}_2, \hat{x}_3, \hat{x}_4) \in \mathbb{R}^5$. Έπειτα υπολογίζεται η διαφορά $(X - \hat{X})$ και τροφοδοτείται στον αλγόριθμο της προς τα πίσω διάδοσης (back-propagation) προκειμένου να εκτιμηθούν τα βέλτιστα βάρη που ελαχιστοποιούν τη συνάρτηση απώλειας (loss function).*

Στην κορυφή των στρωμάτων του Stacked Auto-Encoder, προστίθεται συνήθως ένας SoftMax ταξινομητής για να προβλέψει την κλάση της εισόδου χρησιμοποιώντας τις υψηλού επιπέδου εξαγόμενα χαρακτηριστικά της τελευταίας στρώσης. Κάθε ένα από αυτά τα στρώματα (συμπεριλαμβανομένου του στρώματος SoftMax) έχει εκπαιδευτεί διαδοχικά. Εφόσον το τελευταίο στρώμα είναι εκπαιδευμένο, πραγματοποιείται μία καθολική εκπαίδευση, χρησιμοποιώντας το γνωστό αλγόριθμο της προς τα πίσω διάδοσης (back-propagation). Αυτό είναι γνωστό ως τεχνική fine tuning.

Όπως στο CNN, οι κωδικοποιητές είναι εξαγωγείς χαρακτηριστικών (feature extractors). Ο ρόλος τους είναι να οικοδομήσουν χαρακτηριστικά υψηλού επιπέδου που είναι πιο εύκολο στη χρήση σε εργασίες profiling [2].

Η ιδέα πίσω από τον Αυτόματο Κωδικοποιητή (Auto-Encoder) βασίζεται στην ιδέα μιας καλά δομημένης αναπαράστασης οποιουδήποτε μοντέλου. Ένας κωδικοποιητής είναι μια αντιστοίχιση (mapping) $f_\vartheta$ που μετατρέπει ένα διάνυσμα εισόδου $i$ σε μια κρυφή απεικόνιση στρώματος $h$, όπου $\vartheta = \{W_t, c\}$, $W_t$ είναι το βάρος του πίνακα (matrix weight) και $c$ είναι ο φορέας παραμόρφωσης/μετατόπισης (bias/the offset vector). Ο αποκωδικοποιητής αντιστοιχίζει (κάνει mapping) την κρυφή αναπαράσταση $h$ στην ανακατασκευασμένη είσοδο $d$ μέσω του $e_0$. Η διαδικασία του αυτόματου κωδικοποιητή είναι να συγκρίνει την ανακατασκευασμένη είσοδο με το πρωτότυπο ελαχιστοποιώντας το σφάλμα έτσι ώστε η ανακατασκευασμένη τιμή να πλησιάσει όσο το δυνατόν περισσότερο στην αρχική τιμή.

Σε αυτή την τεχνική, η έξοδος που είναι εν μέρει αλλοιωμένη ξεκαθαρίζεται. Εφόσον εκπαιδευτεί η λειτουργία κωδικοποίησης $f_\vartheta$ του πρώτου αυτόματου κωδικοποιητή αποκοπής, στη συνέχεια χρησιμοποιείται για να αποκαταστήσει την αλλοιωμένη είσοδο ώστε να μπορεί να εκπαιδευτεί το δεύτερο επίπεδο. Αφού εκπαιδεύονται όλα τα στρώματα στον αυτόματο κωδικοποιητή, η έξοδος μπορεί να χρησιμοποιηθεί ως είσοδος σε οποιονδήποτε αλγόριθμο εποπτευόμενης μάθησης, όπως Support Vector Machine, Multiclass logistic regression κλπ. Το σχήμα της εικόνας 2.2.5.iii αντιπροσωπεύει το διάγραμμα ενός στοιβαγμένου αυτόματου κωδικοποιητή.




*Εικόνα 2.2.5.iii: Γραφική αναπαράσταση ενός Stacked Auto-Encoder*

Τα πλεονεκτήματα των στοιβαγμένων κωδικοποιητών είναι τα εξής: Πρώτον, είναι μια μέθοδος ευφυούς μάθησης με επίπεδα. Είναι συμβατό με τα Τεχνητά Νευρωνικά Δίκτυα. Επίσης, λαμβάνει υπόψη του όλες τις εισόδους πραγματικών αριθμών, τις δυαδικές εισόδους, την πιθανοτική κατανομή μόνο με απλή αλλαγή της λειτουργίας απώλειας (loss function) και της λειτουργίας ενεργοποίησης (activation function). Οι Stacked Auto-Encoders μπορούν επίσης να συντονιστούν από μόνοι τους χρησιμοποιώντας τον αλγόριθμο της προς τα πίσω διάδοσης (back-propagation), ο οποίος μειώνει τη συνολική απώλεια αναδόμησης [1].

# Κεφάλαιο 3ο: Σχεδιασμός & Ανάλυση Συστήματος

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τη βασική δομή του λογισμικού που χρησιμοποιήθηκε και τροποποιήθηκε καταλλήλως για την εκτέλεση των αλγορίθμων που παρουσιάστηκαν στο κεφάλαιο 2.

Για να μπορέσουμε να οπτικοποιήσουμε τις παραπάνω μεθόδους θα χρησιμοποιήσουμε τα αρχεία κώδικα των Ruslan Salakhutdinov και Geoff Hinton που βρίσκονται στην ιστοσελίδα: http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html.

Υπάρχουν διαθέσιμα τρία πακέτα κώδικα. Ένα για κάθε μία από τις εξής μεθόδους: DBM, RBM, DBN. Και στις τρεις περιπτώσεις το προγράμματα τροφοδοτούνται με ορισμένα αρχεία που περιέχουν σε raw μορφή χειρόγραφους αριθμητικούς χαρακτήρες όπου με τη χρήση μίας συνάρτησης μετατρέπονται σε μορφή κατανοητή/διαχειρίσιμη από το Matlab. Στη συνέχεια για κάθε μέθοδο πραγματοποιείται εκπαίδευση (pre-training) και μάθηση (learning) ενώ παράλληλα γίνεται καταγραφή των σφαλμάτων αναγνώρισης/κατηγοριοποίησης των αριθμών.

Γενικά σε κάθε εκτέλεση ισχύουν οι παρακάτω τιμές των εξής παραμέτρων:

```
Size of the training dataset = 60000
Size of the test dataset = 10000
```

## 3.1. Deep Boltzmann Machines

Για τα DBMs έχουμε 2 εκτελέσεις την “demo_small” και την “demo”. Η πρώτη κατά σειρά αποτελεί μία μικρογραφία της εκτέλεσης της δεύτερης. Στον πίνακα που ακολουθεί, αναλύεται το περιεχόμενο του κάθε αρχείου που χρειάστηκε για τις εκτελέσεις.

| Όνομα Αρχείου | Περιγραφή |
|---|---|
| *demo_small.m* | Main file for training and fine-tuning a toy DBM model. |
| *demo.m* | Main file for training and fine-tuning a DBM model (reproduces results of the DBM paper). |
| *converter.m* | Converts raw MNIST digits into matlab format. |
| *rbm.m* | Training RBM with binary hidden and binary visible units. |
| *rbm_l2.m* | Training 2nd layer RBM with binary hidden and visible units. |
| *dbm_mf.m* | Joint training of all layers in a DBM. |
| *mf.m* | Implements mean-field inference. |
| *backprop.m* | Backpropagation for fine-tuning a DBM. |
| *mf_class.m* | Helper function used by backprop.m |
| *CG_MNIST.m* | Conjugate Gradient optimization for fine-tuning a DBM. |
| *CG_MNIST_INIT.m* | Conjugate Gradient optimization for fine-tuning a DBM (training top-level parameters, while holding low-level parameters fixed). |
| *makebatches.m* | Creates minibatches for DBM training. |
| *testerr.m* | Computes misclassification error on the MNIST dataset. |
| *dispims.m* | Displays progress during DBM training. |
| *minimize.m* | Conjugate gradient code. |

## 3.2. Restricted Boltzmann Machines

Για τα RBMs έχουμε 2 εκτελέσεις την “demo_toy” και την “demo_rbm”. Η πρώτη κατά σειρά αποτελεί μία μικρογραφία της εκτέλεσης της δεύτερης. Στον πίνακα που ακολουθεί, αναλύεται το περιεχόμενο του κάθε αρχείου που χρειάστηκε για τις εκτελέσεις.

| Όνομα Αρχείου | Περιγραφή |
|---|---|
| *demo_toy.m* | Main file for training and evaluating a toy RBM model. |
| *demo_rbm.m* | Main file for evaluating an RBM model (mnistvh_CD25.mat). |
| *converter.m* | Converts raw MNIST digits into matlab format. |
| *rbm.m* | Training RBM with binary hidden and binary visible units. |
| *calculate_true_partition.m* | Calculates the true log-partition function of the toy mode |
| *RBM_AIS.m* | Estimates partition function using AIS. |
| *base_rate.m* | Estimates biases of the base-rate model by maximum likelihood. |
| *logsum.m* | Helper function used by RBM_AIS.m. |
| *logdiff.m* | Helper function used by RBM_AIS.m. |
| *free_energy.m* | Helper function used by RBM_AIS.m. |
| *makebatches.m* | Creates minibatches for RBM training. |
| *mnistdisp.m* | Displays progress during AIS run. |
| *mnistvh_CD25.mat* | Parameters of the carefully trained RBM model using CD25. |
| *calculate_logprob.m* | Calculates log-probability of data by summing out hidden units. |
| *demo_toy.m* | Main file for training and evaluating a toy RBM model. |

## 3.3. Deep Belief Networks

Για τα DBNs έχουμε 2 εκτελέσεις την “ mnistdeepauto” και την “mnistclassify”. Η πρώτη είναι για training ενός deep autoencoder και η δεύτερη για training ενός classification model.

| Όνομα Αρχείου | Περιγραφή |
|---|---|
| *mnistdeepauto.m* | Main file for training deep autoencoder |
| *mnistclassify.m* | Main file for training classification model |
| *converter.m* | Converts raw MNIST digits into matlab format |
| *rbm.m* | Training RBM with binary hidden and binary visible units |
| *rbmhidlinear.m* | Training RBM with Gaussian hidden and binary visible units |
| *backprop.m* | Backpropagation for fine-tuning an autoencoder |
| *backpropclassify.m* | Backpropagation for classification using "encoder" network |
| *CG_MNIST.m* | Conjugate Gradient optimization for fine-tuning an autoencoder |
| *CG_CLASSIFY_INIT.m* | Conjugate Gradient optimization for classification (training top-layer weights while holding low-level weights fixed) |
| *CG_CLASSIFY.m* | Conjugate Gradient optimization for classification (training all weights) |
| *makebatches.m* | Creates minibatches for RBM training |
| *mnistdisp.m* | Displays progress during fine-tuning stage |

# Κεφάλαιο 4ο: Υλοποίηση Συστήματος

Σε αυτό το κεφάλαιο παρουσιάζονται οι εκτελέσεις των αλγορίθμων που θεωρητικά αναπτύχθηκαν στο Κεφάλαιο 2 μαζί με τις παρεμβάσεις που πραγματοποιήθηκαν σε ορισμένες από τις παραμέτρους των αλγορίθμων (κυρίως στις επαναλήψεις), ελέγχονται τα αποτελέσματα και σχολιάζονται ανά περίπτωση τα ποιοτικά χαρακτηριστικά τους. Οι τροποποιήσεις που έγιναν ήταν αναγκαίες προκειμένου να ολοκληρωθούν τα tests σε εύλογο χρονικό διάστημα.

Στις επόμενες υποενότητες παρατίθενται σε πίνακες τα αποτελέσματα των αλγορίθμων και θα δοθεί έμφαση στα σφάλματα κατά την αναγνώριση/κατηγοριοποίηση των χειρόγραφων χαρακτήρων. Επίσης, δίνονται οι εικόνες από το τελικό στάδιο των τρεξιμάτων και οι χρόνοι εκτέλεσης για κάθε μία από τις μεθόδους.

## 4.1. Deep Boltzmann Machines

### 4.1.1. demo_small

Οι παράμετροι που χρησιμοποιήθηκαν σε αυτή τη μέθοδο είναι οι εξής:

```
%%%%% Training 1st layer %%%%%
numhid = 10;
maxepoch = 10;

%%%%% Training 2st layer %%%%%
numpen = 10;
maxepoch = 10;

%%%%% Fine-tuning two-layer Boltzmann machine for classification %%%%%
maxepoch = 10;
```

Στον επόμενο πίνακα παρουσιάζονται τα σφάλματα κατά τη διαδικασία του pre-training:

| EPOCH | PRETRAINING LAYER 1 WITH RBM: 784-10 (ERROR) | PRETRAINING LAYER 2 WITH RBM: 10-10 (ERROR) |
|---|---|---|
| **1** | 8400123.0 | 208544.0 |
| **2** | 7033473.0 | 177596.0 |
| **3** | 6724717.0 | 151642.0 |
| **4** | 6579357.0 | 132015.0 |
| **5** | 6514131.0 | 117912.0 |
| **6** | 6506378.0 | 98719.0 |
| **7** | 6460473.0 | 84467.0 |
| **8** | 6378999.0 | 80909.0 |
| **9** | 6291550.0 | 79250.0 |
| **10** | 6229382.0 | 78356.0 |

Παρατηρούμε ότι καθώς προχωράνε οι επαναλήψεις το σφάλμα μειώνεται συνεχώς.

Στον επόμενο πίνακα παρουσιάζονται τα Reconstruction Errors και τα Misclassified Training Cases για κάθε *epoch* κατά τη διαδικασία του learning του DBM:

| EPOCH | RECONSTRUCTION ERROR | MISCLASSIFIED TRAINING CASES (OUT OF 60000) |
|---|---|---|
| **1** | 3084091.9 | 5456 |
| **2** | 3049610.0 | 5797 |
| **3** | 3034201.7 | 6151 |
| **4** | 3029653.5 | 5643 |
| **5** | 3025151.1 | 5478 |
| **6** | 3017884.1 | 4974 |
| **7** | 3011645.7 | 4861 |
| **8** | 3004518.8 | 4509 |
| **9** | 2997781.1 | 4107 |
| **10** | 2992220.8 | 3522 |

Number of misclassified test examples: 3519 out of 10000

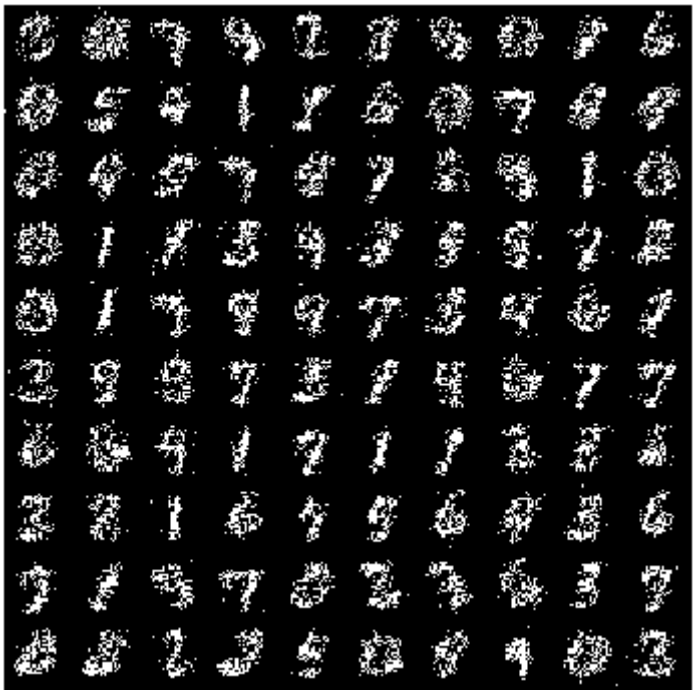
Training discriminative model on MNIST by minimizing cross entropy error.

60 batches of 1000 cases each.

| EPOCH | TEST MISCLASSIFICATION ERROR (OUT OF 10000) | TEST CROSS ENTROPY ERROR |
|---|---|---|
| **1** | 9013 | 23563.168381 |
| **2** | 3260 | 9425.017144 |
| **3** | 3154 | 9046.298875 |
| **4** | 3057 | 8817.049130 |
| **5** | 2965 | 8676.788028 |
| **6** | 2953 | 8633.532026 |
| **7** | 2728 | 7922.900521 |
| **8** | 2545 | 7498.999014 |
| **9** | 2458 | 7209.735065 |
| **10** | 2360 | 7004.906865 |

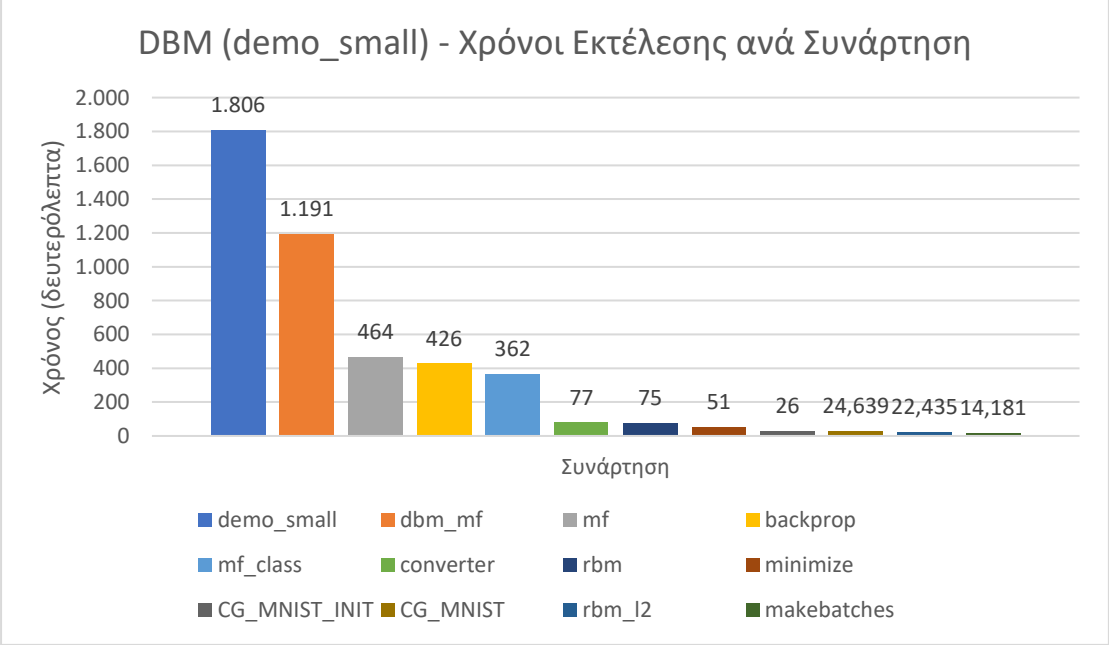
| EPOCH | TRAIN MISCLASSIFICATION ERROR (OUT OF 60000) | TRAIN CROSS ENTROPY ERROR |
|---|---|---|
| **1** | 54112 | 141514.083216 |
| **2** | 19292 | 56929.626112 |
| **3** | 18859 | 54696.604871 |
| **4** | 18279 | 53471.626283 |
| **5** | 17867 | 52725.469460 |
| **6** | 17786 | 52411.542107 |
| **7** | 16171 | 47449.458264 |
| **8** | 15107 | 45038.516594 |
| **9** | 14530 | 43223.833357 |
| **10** | 13903 | 41746.304398 |

Από τους δύο παραπάνω πίνακες παρατηρούμε ότι σε κάθε επόμενο *epoch* τόσο το Test Misclassification Error όσο και το Train Misclassification Error μειώνονται συνεχώς.



*Εικόνα 4.1.1.i: Αποτέλεσμα εκτέλεσης του αλγορίθμου DBM demo_small*

Παρατηρούμε στην Εικόνα 4.1.1.i ότι το αποτέλεσμα του τρεξίματος του demo_small είναι αρκετά κακό. Οι χαρακτήρες που εμφανίζονται δεν είναι σε κάθε περίπτωση αναγνωρίσιμοι.




*Εικόνα 4.1.1.ii: Χρόνοι τρεξίματος για DBM demo_small ανά συνάρτηση*

### 4.1.2. demo

Οι παράμετροι που χρησιμοποιήθηκαν σε αυτή τη μέθοδο είναι οι εξής:

```
%%%%% Training 1st layer %%%%%
numhid = 500;
maxepoch = 100;

%%%%% Training 2st layer %%%%%
numpen = 1000;
maxepoch = 200;

%%%%% Training two-layer Boltzmann machine %%%%%
numhid = 500;
numpen = 1000;
maxepoch = 300;

%%%%% Fine-tuning two-layer Boltzmann machine for classification %%%%%
maxepoch = 100;
```

Παρατήρηση:

Έπειτα από 11 ώρες τρεξίματος πραγματοποιήθηκε βίαιη διακοπή της εκτέλεσης μετά την ολοκλήρωση της *epoch* = 40, δηλαδή κατά τη διάρκεια της επανάληψης για *epoch* = 41 για την οποία δε λάβαμε υπόψη μας τα αποτελέσματα.

Στον επόμενο πίνακα παρουσιάζονται τα σφάλματα κατά τη διαδικασία του pre-training:

| EPOCH | PRETRAINING LAYER 1 WITH RBM: 784-500 (ERROR) | PRETRAINING LAYER 2 WITH RBM: 500-1000 (ERROR) | NUMBER OF MISCLASSI-FIED TEST EXAMPLES (OUT OF 10000) |
|---|---|---|---|
| **1** | 4576513.0 | 4307508.0 | |
| **2** | 3099715.0 | 2864871.0 | |
| **3** | 2739298.0 | 2537504.0 | |
| **4** | 2543804.0 | 2380925.0 | |
| **5** | 2424677.0 | 2287725.0 | |
| **6** | 2383835.0 | 2342920.0 | |
| **7** | 2265579.0 | 2230057.0 | |
| **8** | 2220720.0 | 2185625.0 | |
| **9** | 2201567.0 | 2165819.0 | |
| **10** | 2182438.0 | 2154326.0 | 278 |
| **11** | 2173337.0 | 2152392.0 | |
| **12** | 2171183.0 | 2148120.0 | |
| **13** | 2168306.0 | 2147336.0 | |
| **14** | 2164663.0 | 2145777.0 | |
| **15** | 2163008.0 | 2143092.0 | |
| **16** | 2158138.0 | 2144703.0 | |
| **17** | 2158697.0 | 2142308.0 | |
| **18** | 2155112.0 | 2141812.0 | |

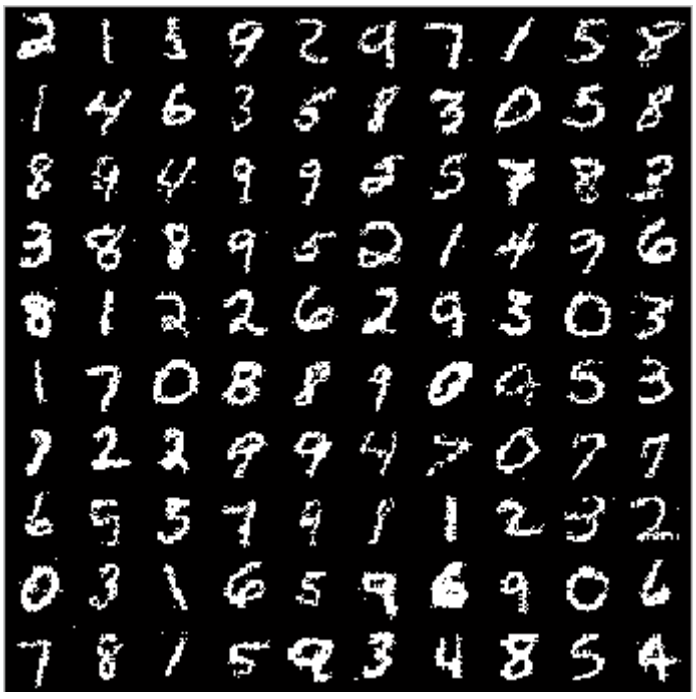
| | | | |
|---|---|---|---|
| **19** | 2157590.0 | 2140987.0 | |
| **20** | 2152326.0 | 2134542.0 | 258 |
| **…** | … | … | … |
| **30** | 2150946.0 | 2482984.0 | 211 |
| **…** | … | … | … |
| **40** | 2147363.0 | 2475490.0 | 206 |
| **…** | … | … | … |
| **50** | 2144256.0 | 2691550.0 | 199 |
| **…** | … | … | … |
| **60** | 2142355.0 | 2685979.0 | 196 |
| **…** | … | … | … |
| **70** | 2144191.0 | 2850508.0 | 184 |
| **…** | … | … | … |
| **80** | 2141451.0 | 2847133.0 | 205 |
| **…** | … | … | … |
| **90** | 2144448.0 | 2983197.0 | 196 |
| **…** | … | … | … |
| **100** | 2143051.0 | 2979216.0 | 187 |
| **…** | … | … | … |

Παρατηρούμε ότι καθώς προχωράνε οι επαναλήψεις το σφάλμα μειώνεται συνεχώς στο Layer 1 που ολοκληρώθηκε επιτυχώς το τρέξιμο. Στο Layer 2 παρατηρούμε σε αντίθεση με το πρώτο ότι ενώ αρχικά μειώνεται το σφάλμα στη συνέχεια αυξάνεται. Δε μπορούμε να λάβουμε σοβαρά υπόψη μας αυτά τα αποτελέσματα διότι δεν έχει ολοκληρωθεί η διαδικασία του pre-training και κατά συνέπεια του learning του παρόντος DBM. Στον πίνακα έχουν τοποθετηθεί αποσιωπητικά διότι έχουν αφαιρεθεί σε εκείνα τα σημεία 9 γραμμές του πίνακα. Ο πλήρης πίνακας βρίσκεται στο Παράρτημα Α με όνομα Πίνακας Α.1 (DBM (demo)).

Στον επόμενο πίνακα παρουσιάζονται τα Reconstruction Errors και τα Misclassified Training Cases για κάθε *epoch* κατά τη διαδικασία του learning του DBM:

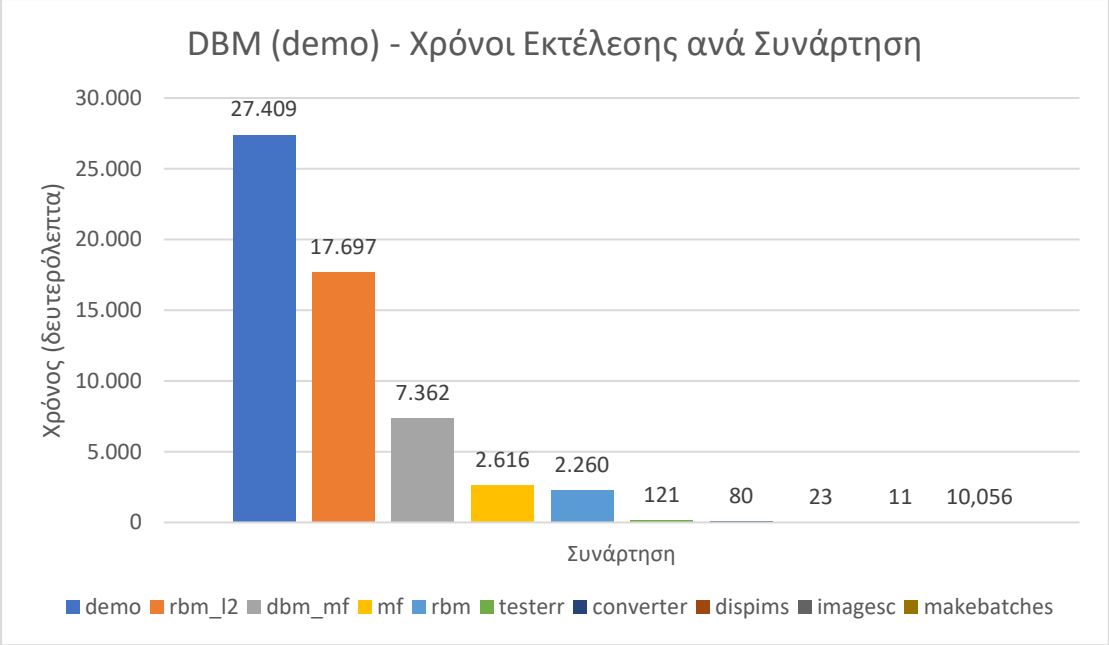
| EPOCH | RECONSTRUCTION ERROR | NUMBER OF MISCLASSIFIED TRAINING CASES (OUT OF 60000) |
|---|---|---|
| **1** | 1008721.0 | 291 |
| **2** | 983321.5 | 270 |
| **3** | 975016.3 | 274 |
| **4** | 969448.9 | 264 |
| **5** | 963975.8 | 241 |
| **6** | 1015241.0 | 286 |
| **7** | 1005944.7 | 300 |
| **8** | 990354.6 | 294 |
| **9** | 974955.7 | 282 |
| **10** | 971377.1 | 271 |
| **11** | 962407.4 | 306 |
| **12** | 957169.5 | 292 |
| **13** | 950063.0 | 290 |
| **14** | 950317.3 | 289 |
| **15** | 943155.7 | 264 |
| **16** | 948629.6 | 279 |

| | | |
|---|---|---|
| **17** | 937799.4 | 257 |
| **18** | 940056.8 | 272 |
| **19** | 931048.3 | 236 |
| **20** | 932007.5 | 255 |
| **21** | 925570.0 | 237 |
| **22** | 927277.9 | 240 |
| **23** | 922772.9 | 244 |
| **24** | 920012.5 | 229 |
| **25** | 921434.2 | 219 |
| **26** | 918746.5 | 219 |
| **27** | 917304.3 | 229 |
| **28** | 912789.1 | 205 |
| **29** | 908942.9 | 218 |
| **30** | 912577.9 | 205 |
| **31** | 909974.3 | 209 |
| **32** | 908153.3 | 219 |
| **33** | 908799.4 | 213 |
| **34** | 904876.7 | 201 |
| **35** | 904620.5 | 197 |
| **36** | 904042.8 | 208 |
| **37** | 900299.4 | 188 |
| **38** | 902054.0 | 182 |
| **39** | 901445.1 | 181 |
| **40** | 900185.5 | 163 |



*Εικόνα 4.1.2.i: Αποτέλεσμα εκτέλεσης του αλγορίθμου DBM demo*

Παρατηρούμε στην Εικόνα 4.1.2.i ότι το αποτέλεσμα του τρεξίματος του demo είναι αρκετά καλύτερο από αυτό του demo_small που είδαμε στην παράγραφο 4.1.1 και συγκεκριμένα στην Εικόνα 4.1.1.i. Οι χαρακτήρες που εμφανίζονται είναι στην πλειοψηφία τους αναγνωρίσιμοι.




*Εικόνα 4.1.2.ii: Χρόνοι τρεξίματος για DBM demo ανά συνάρτηση*

## 4.2. Restricted Boltzmann Machines

### 4.2.1. demo_toy

Οι παράμετροι που χρησιμοποιήθηκαν σε αυτή τη μέθοδο είναι οι εξής:

```
maxepoch = 10;
numhid = 15;
CD = 3;
numruns = 100;
```

| EPOCH | ERROR |
|---|---|
| **1** | 8192190.0 |
| **2** | 6546489.0 |
| **3** | 6191338.0 |
| **4** | 6019993.0 |
| **5** | 5888824.0 |
| **6** | 5721758.0 |
| **7** | 5659454.0 |
| **8** | 5643367.0 |
| **9** | 5634455.0 |
| **10** | 5631675.0 |

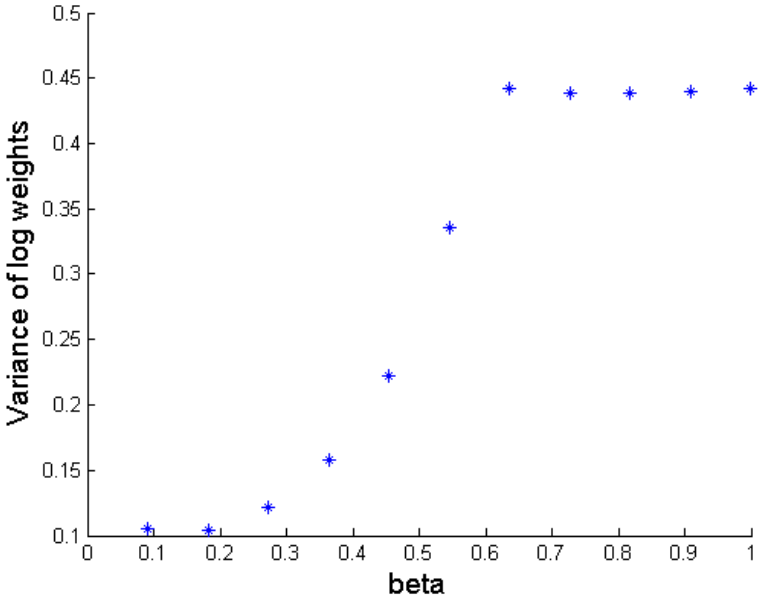
Στον παραπάνω πίνακα έπειτα από training του RBM, παρατηρούμε τη μείωση του σφάλματος σε κάθε επόμενο *epoch*.

Παρακάτω εμφανίζεται το output του Matlab:

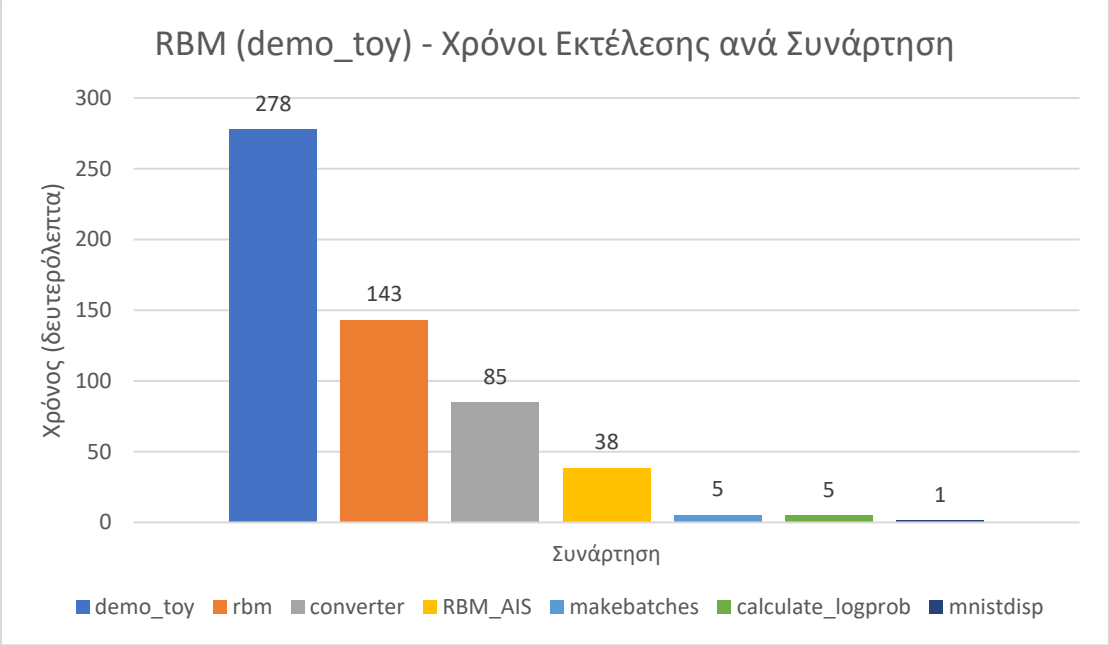
```
Calculating the true log-partition function by brute force.

True log-partition function: 216.899631

Average log-prob on the test data: -172.816200

Estimating partition function by running 100 AIS runs.

Estimated log-partition function (+/- 3 std): 216.907675 (216.696050
217.082243)

Average estimated log-prob on the test data: -172.824244
```

*Εικόνα 4.2.1.i: Αποτέλεσμα εκτέλεσης του αλγορίθμου RBM demo_toy*

Παρατηρούμε στην Εικόνα 4.2.1.i ότι το αποτέλεσμα του τρεξίματος του demo_toy είναι το χειρότερο μέχρι στιγμής. Οι χαρακτήρες που εμφανίζονται δεν είναι αναγνωρίσιμοι.




*Εικόνα 4.2.1.ii: Ακολουθία ενδιάμεσων κατανομών πιθανότητας*




*Εικόνα 4.2.1.iii: Χρόνοι τρεξίματος για RBM demo_toy ανά συνάρτηση*

### 4.2.2. demo_rbm

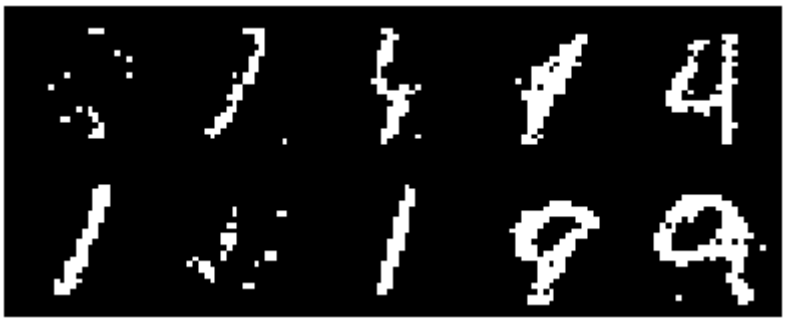
Η παράμετρος που χρησιμοποιήθηκε σε αυτή τη μέθοδο είναι η εξής:

```
numruns = 100;
```

Στην προκειμένη περίπτωση δεν υπάρχει περιορισμός με τη χρήση της μεταβλητής maxepoch. Παρακάτω εμφανίζεται το τμήμα από το output του Matlab:

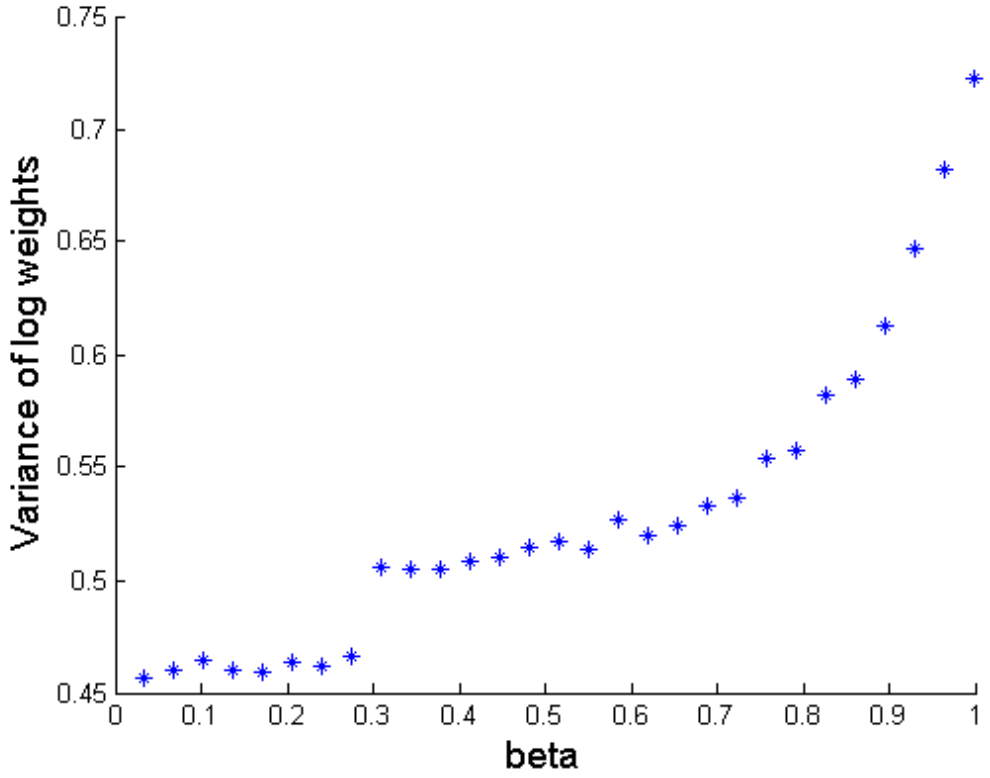
Estimated log-partition function (+/- 3 std): 451.104469 (450.804795 451.334740)

Average estimated log_prob on the test data: -86.178536

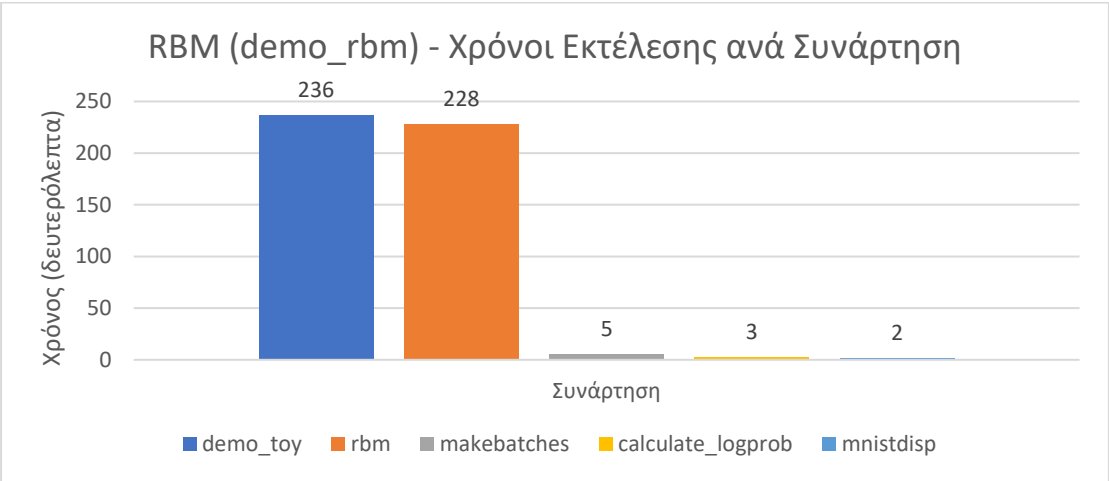


*Εικόνα 4.2.2.i: Αποτέλεσμα εκτέλεσης του αλγορίθμου RBM demo_rbm*

Παρατηρούμε στην Εικόνα 4.2.2.i ότι στο αποτέλεσμα του τρεξίματος του demo_rbm σε σχέση με το demo_toy που είδαμε στην παράγραφο 4.2.1 και συγκεκριμένα στην Εικόνα 4.2.1.i υπάρχει μία σχετική βελτίωση, ωστόσο οι αριθμοί δεν είναι αναγνωρίσιμοι σε αρκετές περιπτώσεις.




*Εικόνα 4.2.2.ii: Ακολουθία ενδιάμεσων κατανομών πιθανότητας*




*Εικόνα 4.2.1.iii: Χρόνοι τρεξίματος για RBM demo_rbm ανά συνάρτηση*

Οι χρόνοι τρεξίματος είναι μέχρι στιγμής οι καλύτεροι που έχουν σημειωθεί στα τρεξίματα συγκριτικά και με τις 3 μεθόδους ωστόσο τα αποτελέσματα είναι τα χειρότερα, τόσο για το demo_toy όσο και για το demo_rbm για αντίστοιχο πλήθος επαναλήψεων (Εικόνα 4.2.1.iii).

## 4.3. Deep Belief Networks

### 4.3.1. mnistdeepauto

Οι παράμετροι που χρησιμοποιήθηκαν σε αυτή τη μέθοδο είναι οι εξής:

```
maxepoch = 10;
numhid = 10;
numpen = 5;
numpen2 = 2;
numopen = 3;
```

Στον επόμενο πίνακα παρουσιάζονται τα σφάλματα κατά τη διαδικασία του pre-training ενός deep autoencoder πάνω σε MNIST digits:

| EPOCH | PRETRAINING LAYER 1 WITH RBM: 784-10 (ERROR) | PRETRAINING LAYER 2 WITH RBM: 10-5 (ERROR) | PRETRAINING LAYER 3 WITH RBM: 5-2 (ERROR) | PRETRAINING LAYER 4 WITH RBM: 2-3 (ERROR) |
|---|---|---|---|---|
| **1** | 2586644.1 | 58059.4 | 48191.8 | 22650.866729 |
| **2** | 2301503.7 | 35080.9 | 37329.9 | 18901.766914 |
| **3** | 2306704.4 | 29458.8 | 34887.1 | 17017.181858 |
| **4** | 2327030.1 | 26466.8 | 34589.6 | 16463.699047 |
| **5** | 2335726.1 | 24919.4 | 34392.8 | 16387.544813 |
| **6** | 2352444.1 | 20680.0 | 34017.4 | 15926.785098 |
| **7** | 2319903.5 | 19042.5 | 33459.7 | 14863.153009 |
| **8** | 2325225.5 | 18978.9 | 33196.8 | 12458.494654 |
| **9** | 2334655.0 | 18965.1 | 33292.3 | 9692.702885 |
| **10** | 2351887.8 | 19046.3 | 33313.3 | 7771.134145 |

Παρατηρούμε ότι καθώς προχωράνε οι επαναλήψεις το σφάλμα μειώνεται συνεχώς σε κάθε Layer.

Fine-tuning deep autoencoder by minimizing cross entropy error.

60 batches of 1000 cases each.

Στον επόμενο πίνακα παρουσιάζονται τα Train Squared Errors και τα Test Squared Errors για κάθε *epoch* κατά τη διαδικασία του Fine-tuning ενός deep autoencoder:

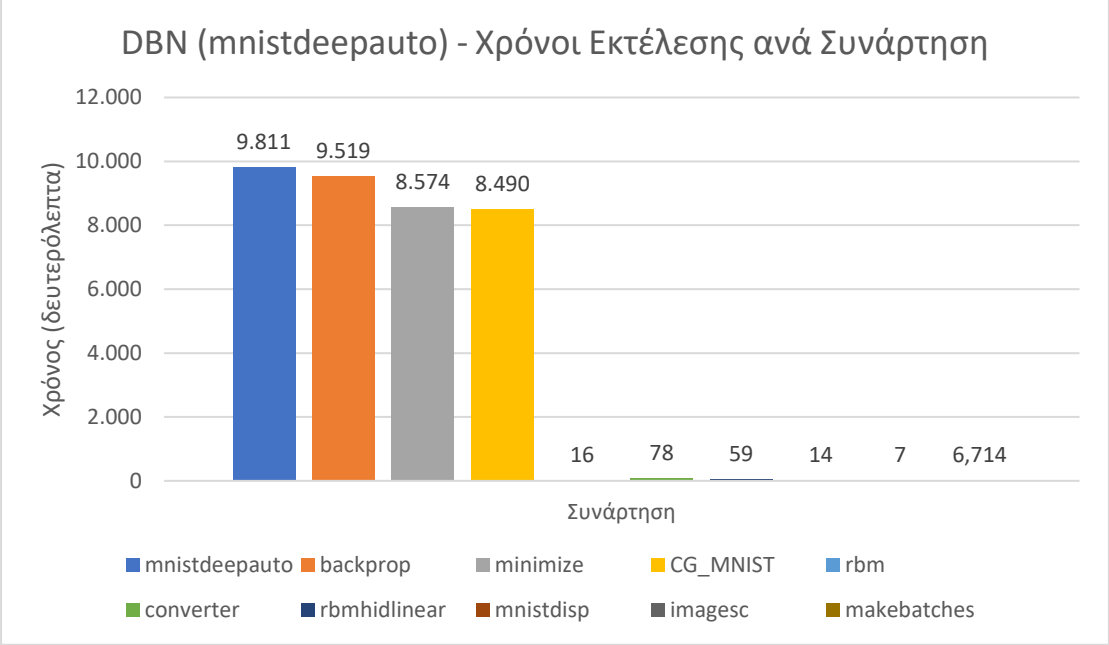
| BEFORE EPOCH | TRAIN SQUARED ERROR | TEST SQUARED ERROR |
|---|---|---|
| **1** | 49.708 | 49.706 |
| **2** | 45.937 | 45.830 |
| **3** | 45.390 | 45.255 |
| **4** | 44.891 | 44.754 |
| **5** | 44.432 | 44.286 |
| **6** | 44.147 | 43.982 |

| | | |
|---|---|---|
| **7** | 43.900 | 43.748 |
| **8** | 43.704 | 43.528 |
| **9** | 43.537 | 43.397 |
| **10** | 43.408 | 43.264 |
| **11** | 43.307 | 43.192 |
| **12** | 43.206 | 43.072 |
| **13** | 43.096 | 42.943 |
| **14** | 43.015 | 42.877 |
| **15** | 42.929 | 42.802 |
| **16** | 42.867 | 42.733 |
| **17** | 42.804 | 42.676 |
| **18** | 42.724 | 42.595 |
| **19** | 42.682 | 42.557 |
| **20** | 42.614 | 42.516 |
| **…** | … | … |
| **30** | 42.280 | 42.190 |
| **…** | … | … |
| **40** | 42.088 | 42.033 |
| **…** | … | … |
| **50** | 41.950 | 41.914 |
| **…** | … | … |
| **60** | 41.817 | 41.804 |
| **…** | … | … |
| **70** | 41.729 | 41.740 |
| **…** | … | … |
| **80** | 41.659 | 41.682 |
| **…** | … | … |
| **90** | 41.598 | 41.638 |
| **…** | … | … |
| **100** | 41.551 | 41.585 |
| **…** | … | … |
| **110** | 41.501 | 41.541 |
| **…** | … | … |
| **120** | 41.457 | 41.505 |
| **…** | … | … |
| **130** | 41.412 | 41.463 |
| **…** | … | … |
| **140** | 41.377 | 41.419 |
| **…** | … | … |
| **150** | 41.334 | 41.377 |
| **…** | … | … |
| **160** | 41.294 | 41.333 |
| **…** | … | … |
| **170** | 41.264 | 41.303 |
| **…** | … | … |
| **180** | 41.234 | 41.272 |
| **…** | … | … |
| **190** | 41.204 | 41.243 |
| **…** | … | … |
| **200** | 41.174 | 41.216 |
| **…** | … | … |

Από τον παραπάνω πίνακα παρατηρούμε ότι σε κάθε επόμενο *epoch* τόσο το Train Squared Errors και τα Test Squared Errors μειώνονται συνεχώς. Έχουν τοποθετηθεί αποσιωπητικά διότι έχουν αφαιρεθεί σε εκείνα τα σημεία 9 γραμμές του πίνακα. Ο πλήρης πίνακας βρίσκεται στο Παράρτημα Α με όνομα Πίνακας Α.2 (DBN (mnistdeepauto)).

*Εικόνα 4.3.1.i: Αποτέλεσμα εκτέλεσης του αλγορίθμου για DBN mnistdeepauto*

Γενικά παρατηρούμε μία πολύ καλή απεικόνιση των ψηφίων στην Εικόνα 4.3.1.i. Είναι μέχρι στιγμής η καλύτερη μέθοδος και από τις 3, ωστόσο λόγω των περιορισμών στις παραμέτρους που θέσαμε για την εκτέλεση του αλγορίθμου σε λογικό χρονικό διάστημα δεν μας επιτρέπουν να το δούμε με απόλυτη σαφήνεια. Αν είχαμε αφήσει τις αρχικές παραμέτρους θα ολοκληρωνόταν σε πολύ μεγαλύτερο χρονικό διάστημα αλλά με σαφώς ευκρινέστερα αποτελέσματα στην απεικόνιση των χειρόγραφων ψηφίων.




*Εικόνα 4.3.1.ii: Χρόνοι τρεξίματος για DBN mnistdeepauto ανά συνάρτηση*

Σε αυτό το τρέξιμο οι χρόνοι εκτέλεσης ήταν αρκετά μεγάλοι παρόλες τις τροποποιήσεις που πραγματοποιήθηκαν για να περιοριστούν όσο γίνεται περισσότερο, προφανώς με κόστος στην ποιότητα των αποτελεσμάτων που όπως είδαμε στην Εικόνα 4.3.1.i ήταν πολύ ικανοποιητικά.

### 4.3.2. mnistclassify

Οι παράμετροι που χρησιμοποιήθηκαν σε αυτή τη μέθοδο είναι οι εξής:

```
maxepoch = 10;
numhid = 5;
numpen = 5;
numpen2 = 20;
```

Στον επόμενο πίνακα παρουσιάζονται τα σφάλματα κατά τη διαδικασία του pretraining ενός deep auto encoder για classification σε MNIST digits:

| EPOCH | PRETRAINING LAYER 1 WITH RBM: 784-5 (ERROR) | PRETRAINING LAYER 2 WITH RBM: 5-5 (ERROR) | PRETRAINING LAYER 3 WITH RBM: 5-20 (ERROR) |
|---|---|---|---|
| **1** | 3006587.9 | 29207.9 | 25258.5 |
| **2** | 2784088.9 | 14584.5 | 9434.9 |
| **3** | 2760407.5 | 9700.0 | 6617.9 |
| **4** | 2760103.0 | 7520.0 | 5520.2 |
| **5** | 2722399.4 | 6553.8 | 4926.0 |
| **6** | 2688914.5 | 4861.3 | 4153.4 |
| **7** | 2686759.2 | 3561.7 | 3663.9 |
| **8** | 2686058.9 | 3376.2 | 3482.8 |
| **9** | 2684975.9 | 3257.4 | 3361.9 |
| **10** | 2677644.5 | 3206.9 | 3313.5 |

Training discriminative model on MNIST by minimizing cross entropy error.

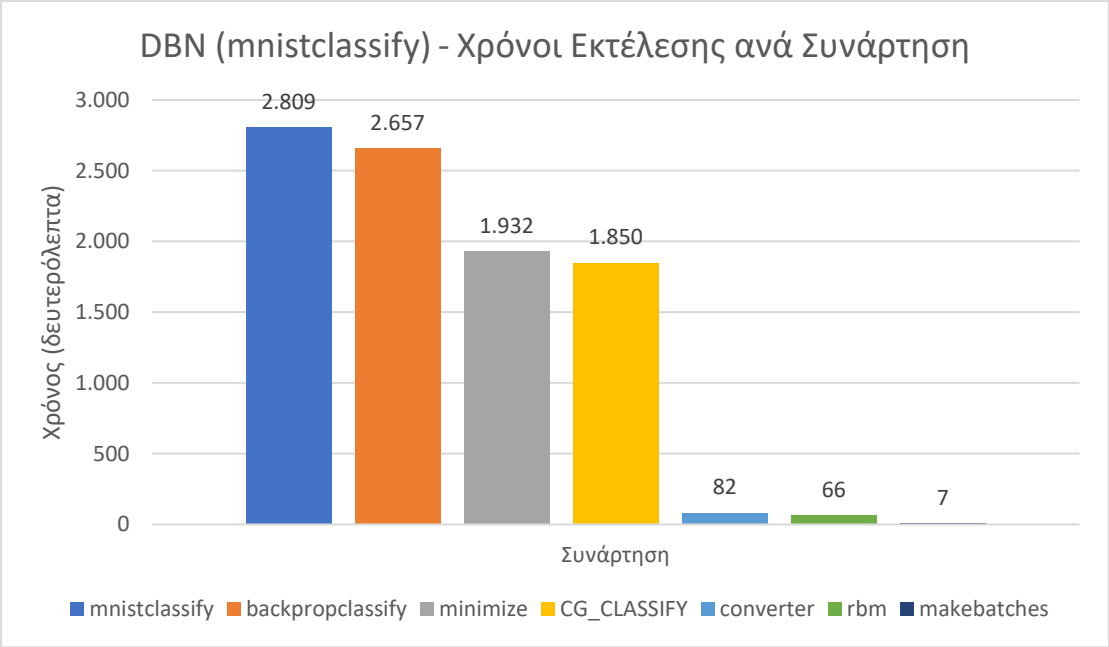
60 batches of 1000 cases each.

Στον επόμενο πίνακα παρουσιάζονται τα πλήθη των αριθμών που δεν κατάφεραν να κατηγοριοποιηθούν πριν από κάθε epoch τόσο κατά το training (Train # Misclassified) όσο και κατά το learning (Test # Misclassified):

| BEFORE EPOCH | TRAIN # MISCLASSIFIED (FROM 60000) | TEST # MISCLASSIFIED (FROM 10000) |
|---|---|---|
| **1** | 51142 | 8468 |
| **2** | 38242 | 6351 |
| **3** | 38154 | 6336 |
| **4** | 38155 | 6338 |
| **5** | 38157 | 6338 |
| **6** | 38160 | 6338 |
| **7** | 32796 | 5400 |
| **8** | 29548 | 4876 |
| **9** | 27548 | 4523 |
| **10** | 26646 | 4362 |
| **11** | 24990 | 4136 |
| **12** | 23598 | 3905 |

| | | |
|---|---|---|
| **13** | 22246 | 3716 |
| **14** | 21456 | 3585 |
| **15** | 20931 | 3547 |
| **16** | 19787 | 3315 |
| **17** | 19392 | 3268 |
| **18** | 18903 | 3174 |
| **19** | 18610 | 3138 |
| **20** | 18606 | 3072 |
| **…** | … | … |
| **30** | 16720 | 2827 |
| **…** | … | … |
| **40** | 16113 | 2740 |
| **…** | … | … |
| **50** | 15637 | 2706 |
| **…** | … | … |
| **60** | 11263 | 1965 |
| **…** | … | … |
| **70** | 10614 | 1946 |
| **…** | … | … |
| **80** | 10339 | 1884 |
| **…** | … | … |
| **90** | 9902 | 1793 |
| **…** | … | … |
| **100** | 9879 | 1807 |
| **…** | … | … |
| **110** | 9525 | 1747 |
| **…** | … | … |
| **120** | 9455 | 1738 |
| **…** | … | … |
| **130** | 9591 | 1777 |
| **…** | … | … |
| **140** | 9725 | 1798 |
| **…** | … | … |
| **150** | 9647 | 1798 |
| **…** | … | … |
| **160** | 9396 | 1736 |
| **…** | … | … |
| **170** | 9326 | 1735 |
| **…** | … | … |
| **180** | 9168 | 1715 |
| **…** | … | … |
| **190** | 9066 | 1708 |
| **…** | … | … |
| **200** | 8995 | 1691 |
| **…** | … | … |

Σύμφωνα με τον παραπάνω πίνακα παρατηρούμε ότι έπειτα από 200 *epochs* το σφάλμα κατηγοριοποίησης των αριθμών μειώθηκε σημαντικά σε σχέση με το *epoch* 1. Έχουν τοποθετηθεί αποσιωπητικά διότι έχουν αφαιρεθεί σε εκείνα τα σημεία 9 γραμμές του πίνακα. Ο πλήρης πίνακας βρίσκεται στο Παράρτημα Α με όνομα Πίνακας Α.3 (DBN (mnistclassify)).




*Εικόνα 4.3.2.i: Χρόνοι τρεξίματος για DBN mnistclassify ανά συνάρτηση*

Οι χρόνοι εκτέλεσης είναι σχετικά μεγάλοι παρόλες τις τροποποιήσεις των παραμέτρων (Εικόνα 4.3.2.i). Ωστόσο, ο αλγόριθμος φάνηκε αρκετά αποδοτικός στην κατηγοριοποίηση (και έμμεσα στην αναγνώριση) των αριθμών.

## 4.4. Παρατηρήσεις & Συγκέντρωση Στοιχείων:

Έπειτα από στρογγυλοποιήσεις των αποτελεσμάτων των χρόνων εκτέλεσης των παραπάνω αλγορίθμων προκύπτει ο παρακάτω πίνακας με τα συγκεντρωτικά στοιχεία για κάθε μέθοδο:

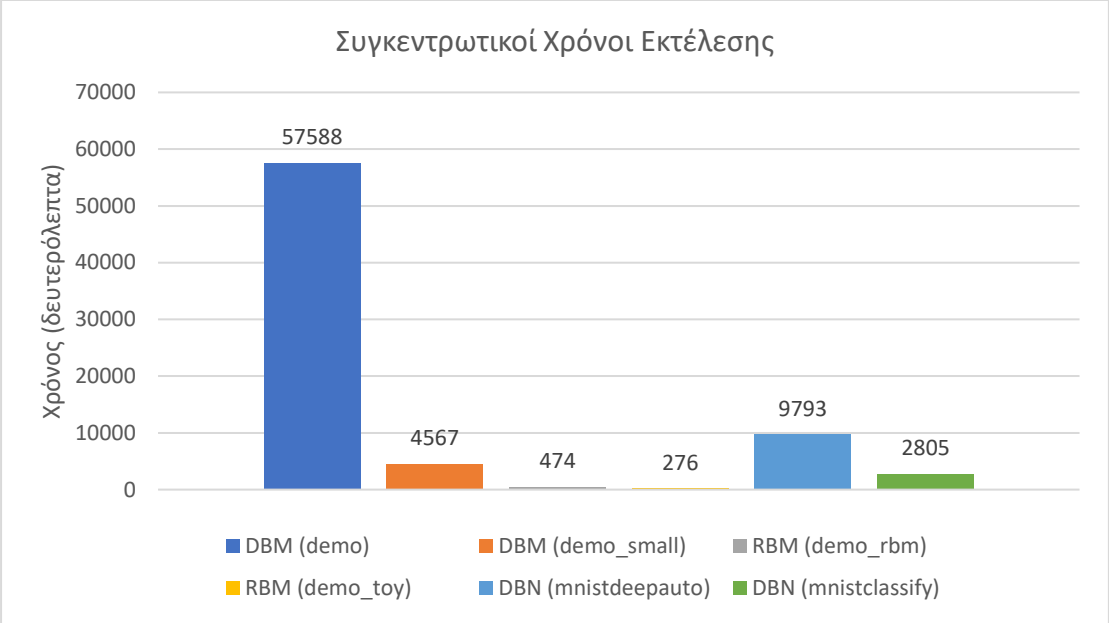
Χρόνος Εκτέλεσης:

Όπως ήταν αναμενόμενο η πιο αργή μέθοδος από αυτές τις τρεις που εξετάσαμε είναι η Deep Boltzmann Machine. Σύμφωνα με τον ορισμό του Deep Boltzmann Machine υπάρχει επικοινωνία μεταξύ hidden-visible units αλλά και μεταξύ των hidden-hidden units και των visible-visible units. Η επικοινωνία αυτή μεταφράζεται σε επιπρόσθετο χρόνο για την ολοκλήρωση της εκτέλεσης.

Στην περίπτωση του DBM_demo (παράγραφο 4.1.2) αναγκαστήκαμε να διακόψουμε την εκτέλεση του προγράμματος έπειτα από 11 ώρες συνεχούς εκτέλεσης (με 100% χρήση CPU σε AMD FX-8350 Eight-Core Processor χρονισμένος στα 4.00 GHz).

Σημαντική μείωση στο χρόνο εκτέλεσης παρατηρήσαμε στο demo_rbm (παράγραφο 4.2.2) το οποίο για περισσότερες από 100 εποχές κατάφερε και τερμάτισε πρώτο απ' όλα λόγω της μειωμένης επικοινωνίας μεταξύ των units σε σχέση με τα DBMs (Εικόνα 4.4.i).

| Μέθοδος | Χρόνος Εκτέλεσης (σε δευτερόλεπτα) |
|---|---|
| DBM (demo) [Interrupted at epoch = 40] | 57588 [143970 εκτίμηση για epoch = 100] |
| DBM (demo_small) | 4567 |
| RBM (demo_rbm) | 474 |
| RBM (demo_toy) | 276 |
| DBN (mnistdeepauto) | 9793 |
| DBN (mnistclassify) | 2805 |



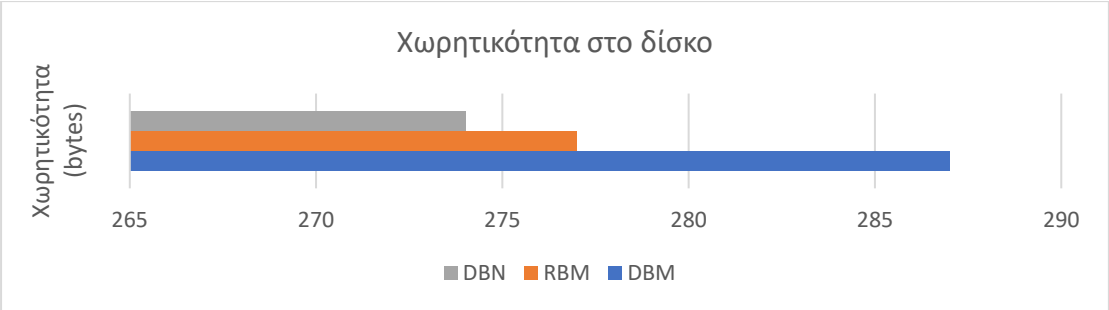

*Εικόνα 4.4.i: Συγκεντρωτικοί χρόνοι εκτέλεσης ανά μέθοδο*

Χώρος Αποθήκευσης:

Ο χώρος που κατέλαβαν τα tests για να ολοκληρώσουν την εκτέλεσή τους ήταν περίπου στα ίδια επίπεδα. Η παρακάτω λίστα δείχνει με αύξουσα σειρά το MBs που χρειάστηκε η κάθε μέθοδος:

1. DBN: 274 MB (288.264.558 bytes)
2. RBM: 277 MB (291.154.247 bytes)
3. DBM: 287 MB (301.896.254 bytes)

Με μικρή διαφορά είναι πρώτη η DBN τόσο για την deepmnistauto όσο και για την mnistclassify, επομένως δεν συντρέχει λόγος περαιτέρω σχολιασμού (Εικόνα 4.4.ii).




*Εικόνα 4.4.ii: Χώρος που δεσμεύεται στο δίσκο ανά μέθοδο*

Ποιότητα Αποτελεσμάτων:

Δεδομένων των εξαγόμενων φωτογραφιών του Matlab (Εικόνες 4.1.1.i, 4.1.2.i, 4.2.1.i, 4.2.2.i, 4.3.1.i) καθώς επίσης και των πινάκων με τα σφάλματα μπορούμε να κρίνουμε την ποιότητα των αποτελεσμάτων με αύξουσα σειρά προτίμησης ως εξής:

1. DBN
   Παρατηρήσαμε ότι παρόλο που περιορίσαμε αρκετά την εκτέλεση του αλγορίθμου με την αλλαγή των αρχικών παραμέτρων του για να καταφέρουμε να ολοκληρώσουμε την εκτέλεσή του σε λογικά χρονικά πλαίσια, τα αποτελέσματα ήταν ικανοποιητικά καλά. Αν επιτρέπαμε το τρέξιμο με τις αντίστοιχες επαναλήψεις των υπολοίπων μεθόδων τα αποτελέσματα τα ήταν καλύτερα από κάθε άλλη μέθοδο. Αυτό οφείλεται στη συνδυαστική χρήση των προηγούμενων μεθόδων (κυρίως των RBMs σε πολυεπίπεδη (multi-layered) διάταξη).

2. DBM
   Όπως εξηγήσαμε και παραπάνω, σύμφωνα με τον ορισμό του Deep Boltzmann Machine υπάρχει επικοινωνία μεταξύ hidden-visible units αλλά και μεταξύ των hidden-hidden units και των visible-visible units. Η επικοινωνία αυτή έχει μία σημαντική επιβάρυνση στο χρόνος εκτέλεσης, γεγονός που σε μεγάλα προβλήματα η μέθοδος αυτή καθίσταται ανίκανη να εξάγει συμπέρασμα σε λογικό υπολογιστικό χρόνο.

3. RBM
   Τα RBMs σε σχέση με τα DBMs είναι κατά πολύ ταχύτερα λόγω του ότι επιτρέπουν την επικοινωνία μόνο μεταξύ των hidden-visible units και όχι μεταξύ των hidden-hidden units και των visible-visible units όπως συμβαίνει στα DBNs. Χάρη σε αυτή την κατασκευαστική αρχή, η επικοινωνία μειώνεται σημαντικά και η εκτέλεση του αλγορίθμου όπως παρατηρήσαμε στις υποενότητες 4.2.1 και 4.2.2 είναι η ταχύτερη. Ωστόσο, τα αποτελέσματα δεν ήταν ικανοποιητικά και ήταν αναγκαία η χρήση πολλών επιπέδων RBMs όπως χρησιμοποιήθηκαν στα DBNs με μεγάλη επιτυχία.

# Κεφάλαιο 5ο: Συμπεράσματα & Μελλοντική Έρευνα

## 5.1. Συμπεράσματα

Στην προσπάθειά μας να προσαρμόσουμε τις εκτελέσεις των παραπάνω αλγορίθμων σε μία κοινωνία πρακτόρων που θα λειτουργούν με τον ίδιο τρόπο με αυτόν που ενεργούν τα threads ενός προγράμματος σε ένα πολυεπεξεργαστικό/πολυπύρηνο υπολογιστικό σύστημα, επιφορτισμένα με κατάλληλους αισθητήρες για τη λήψη φωτογραφιών των αριθμητικών χαρακτήρων, αναγκαζόμαστε να πάρουμε κάποιες αποφάσεις και να συμβιβαστούμε ώστε να είναι επιτεύξιμη η εκτέλεση σε έναν πραγματικό πράκτορα σε εύλογο χρονικό διάστημα.

Όσα αναφέρθηκαν στα προηγούμενα κεφάλαια από τη θεωρητική ανάπτυξη των μεθόδων για Deep Learning μέχρι τα τρεξίματα των αλγορίθμων και τα αποτελέσματά τους, είχαν σαν σκοπό την απάντηση στο ερώτημα: «Μπορούν να προσαρμοστούν οι μέθοδοι Deep Learning πάνω σε ένα ρεαλιστικό πολυπρακτορικό σύστημα;» Κρίνοντας από τη μέχρι στιγμής πορεία των αποτελεσμάτων, θεωρείται απαγορευτικό για έναν τυπικό πράκτορα να τρέξει απροβλημάτιστα και αποδίδοντας ένα καλό επίπεδο αποκρισιμότητας (responsiveness) σε μία πραγματική υλοποίηση.

Ένας μέσος πράκτορας που θα αναλάβει να κάνει μία τέτοια εργασία θα έχει πεπερασμένους και περιορισμένους πόρους τόσο στην επεξεργαστική ισχύ του όσο και στη διαθεσιμότητα επαρκούς ποσότητας ενέργειας για να καταφέρει να ολοκληρώσει ένα πλήρες τρέξιμο. Για να επιτύχουμε την αποκρισιμότητα των πρακτόρων είναι σχεδόν αδύνατον να τρέξουν τοπικά οι παραπάνω αλγόριθμοι διότι λόγω της εκτεταμένης χρήσης του επεξεργαστή θα δημιουργήσει λιμοκτονία (starvation) του πράκτορα ενώ παράλληλα θα εξαντληθούν ταχύτατα τα αποθέματα ενέργειάς του εφόσον μιλάμε για φορητές συσκευές.

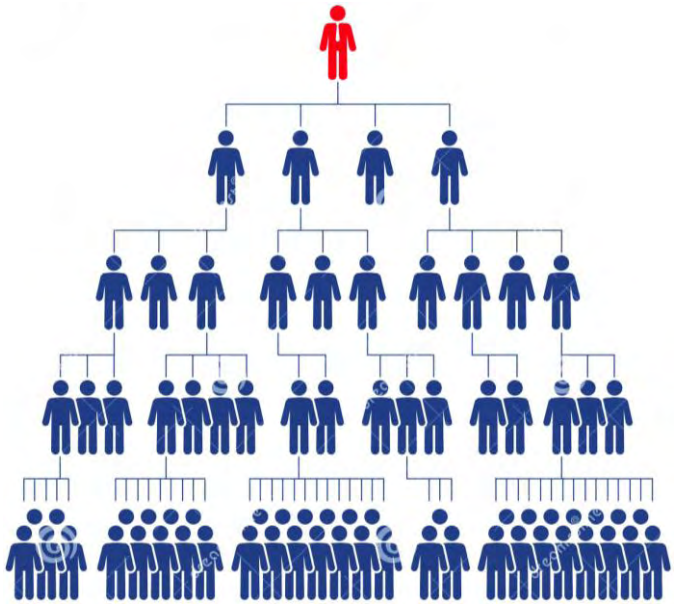
Για να αποφύγουμε τον υπολογιστικό φόρτο τοπικά σε κάθε πράκτορα μπορούμε να προτείνουμε την ανάθεση αυτού του φόρτου σε έναν τρίτο. Μία πιθανή προσπάθεια θα ήταν να υιοθετήσει κανείς τη ιεραρχική αρχιτεκτονική για την κοινωνία πρακτόρων προκειμένου να πραγματοποιείται η λήψη της απόφασης, ενώ το τρέξιμο των αλγορίθμων να πραγματοποιηθεί από μία Αρχιτεκτονική Cloud. Αυτό θα συζητηθεί πιο αναλυτικά στην υποκεφάλαιο 5.2 που ακολουθεί.

## 5.2. Μελλοντική Δουλειά

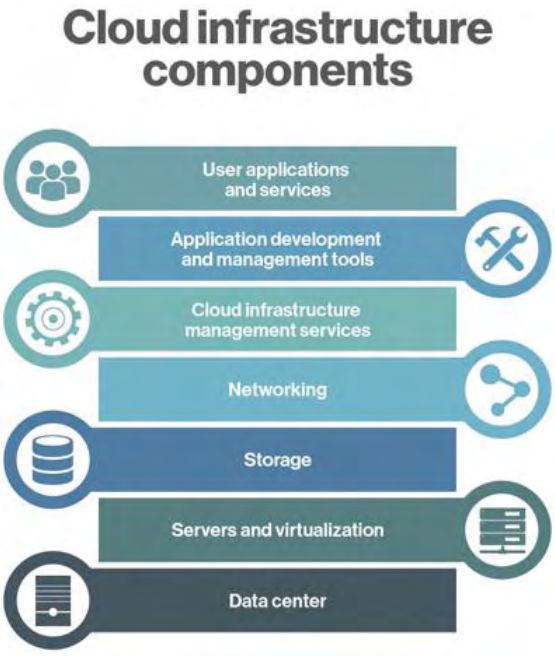
Σε αυτή την ενότητα θα προσπαθήσουμε να επιλύσουμε σε θεωρητικό επίπεδο τα βασικά προβλήματα που αντιμετωπίσαμε στη μέχρι στιγμής πορεία μας σε σχέση με έναν τυπικό πράκτορα. Στο υποκεφάλαιο 5.1 προτείναμε να αναθέσουμε το τρέξιμο των αλγορίθμων Deep Learning σε μία Αρχιτεκτονική Cloud (Cloud Architecture).

Θα ξεκινήσουμε την προσέγγισή μας από την οργάνωση της κοινωνίας των πρακτόρων. Λαμβάνοντας υπόψη το πλήθος των πρακτόρων και την ταχύτητα της μεταξύ τους επικοινωνίας θα πρέπει να υπάρξει μία πολυεπίπεδη ιεραρχική δομή για την οργάνωσή τους με έναν βασικό coordinator προκειμένου να γίνει η τελική λήψη απόφασης για την αναγνώριση και συγκέντρωση του εκάστοτε χειρόγραφου κειμένου. Πιο συγκεκριμένα, ανά Ν πράκτορες θα

υπάρχει ένας άμεσα συνδεδεμένος πράκτορας πατέρας ο οποίος θα συγκεντρώνει τα συμπεράσματα των Ν πρακτόρων και θα τα προωθεί στο πιο πάνω επίπεδο της ιεραρχίας. Η τελική απόφαση θα λαμβάνεται από τον βασικό coordinator που θα είναι στην κορυφή της ιεραρχίας (Εικόνα 5.2.i) [12].



*Εικόνα 5.2.i: Ιεραρχική δομή των πρακτόρων (Πηγή: https://fr.dreamstime.com)*




*Εικόνα 5.2.ii: Δομή μιας αρχιτεκτονικής cloud (Πηγή: http://searchcloudcomputing.techtarget.com/)*

Στη συνέχεια, θα μελετήσουμε πως ο κάθε πράκτορας επικοινωνεί με την αρχιτεκτονική cloud. Στην Εικόνα 5.2.ii βλέπουμε μία τυπική δομή μιας Αρχιτεκτονικής Cloud όπου στο πρώτο επίπεδο (User applications and services) μπορούμε να θεωρήσουμε ότι έχουμε τους

πράκτορες και στο τρίτο επίπεδο (Cloud infrastructure management services) και παρακάτω θα οργανώσουμε το σύστημά μας το οποίο θα αναλαμβάνει να τρέξει τους αλγορίθμους Deep Learning για τα δεδομένα που έστειλε ο κάθε πράκτορας και για την αντίστοιχη μέθοδο που ζητήθηκε.

Παράδειγμα:

Θα προσπαθήσουμε με ένα απλό παράδειγμα αποτελούμενο από 5 πράκτορες εκ των οποίων ο ένας είναι ο coordinator, να εξηγήσουμε πιο απλοϊκά το πως προτείνουμε το σχεδιασμό του συστήματος. Ο καθένας από τους 4 βασικούς πράκτορες διαθέτει αισθητήρες για την λήψη φωτογραφίας του χειρόγραφου κειμένου.

Πιο αναλυτικά, έστω ότι:

- ο Agent 1.1 λαμβάνει το πρώτο ψηφίο προς ανάγνωση και το στέλνει στο cloud με αναγνωριστικό εκτέλεσης DBM,
- ο Agent 1.2 λαμβάνει το πρώτο ψηφίο προς ανάγνωση και το στέλνει στο cloud με αναγνωριστικό εκτέλεσης RBM,
- ο Agent 2.1 λαμβάνει το δεύτερο ψηφίο προς ανάγνωση και το στέλνει στο cloud με αναγνωριστικό εκτέλεσης DBM,
- ο Agent 2.2 λαμβάνει το δεύτερο ψηφίο προς ανάγνωση και το στέλνει στο cloud με αναγνωριστικό εκτέλεσης RBM.

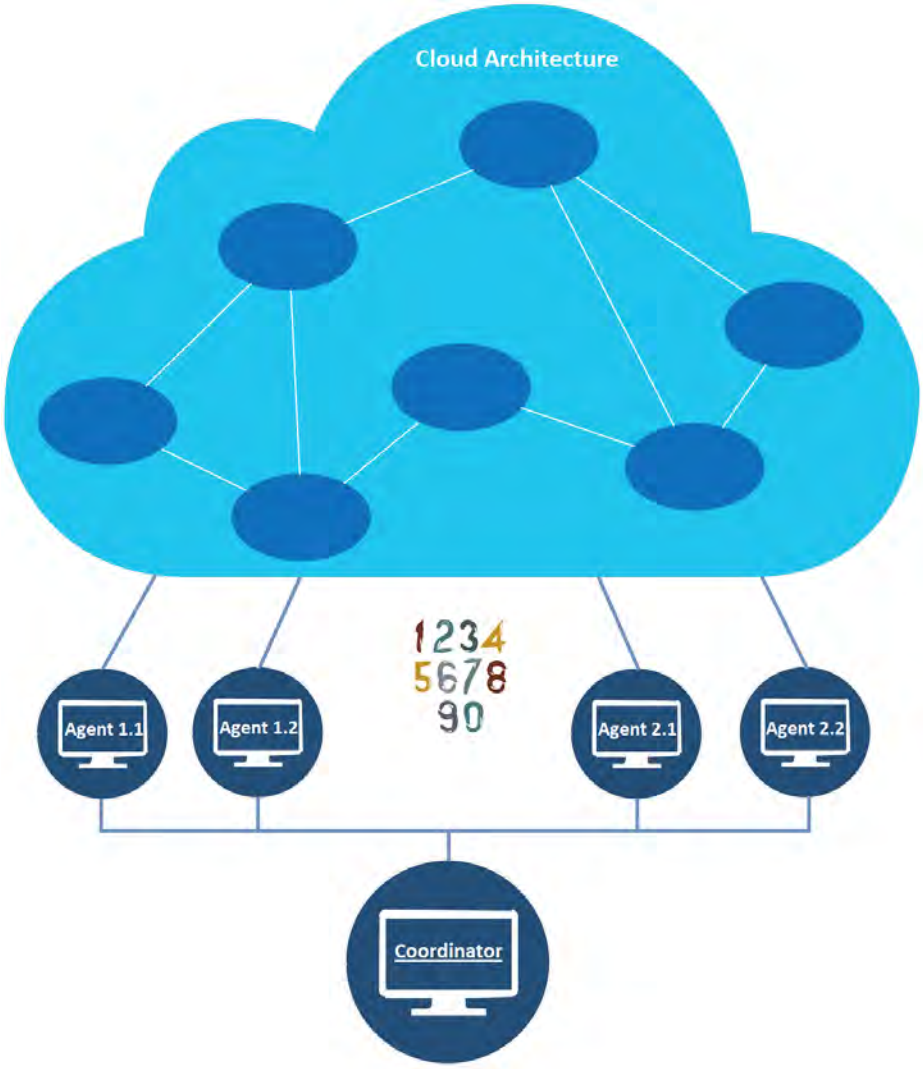
Έπειτα, το κατάλληλο cloud service αναλαμβάνει να εξυπηρετήσει τον αντίστοιχο Agent X,Y λαμβάνοντας τα δεδομένα, τρέχει τη ζητούμενη μέθοδο. Επιστρέφει στον πράκτορα τα αποτελέσματα με το αντίστοιχο σφάλμα και αυτός με τη σειρά του τα προωθεί στον coordinator.

Τέλος, Ο coordinator βρίσκεται σε θέση να αποφασίσει/αναγνωρίσει πλέον ποιο είναι το κάθε ψηφίο. Η διαδικασία θα συνεχιστεί μέχρι να τελειώσει το κείμενο.

Εναλλακτική Λύση:

Μπορούν να υπάρξουν πολλές ακόμα παραλλαγές αυτής της υλοποίησης για να είναι ταχύτερη και πιο παραλληλοποιήσιμη η πορεία της αναγνώρισης, όπως το να ζητάει ο κάθε πράκτορας την αναγνώριση ενός ψηφίου με δύο διαφορετικές μεθόδους και να επιλέγει ο καθένας ξεχωριστά (τοπικά) ανάλογα με το επιστρεφόμενο σφάλμα ποιο αποτέλεσμα (δηλαδή το αποτέλεσμα από ποια από τις δύο μεθόδους) θα στείλει στον coordinator. Εξετάζεται ποια από τις δύο μεθόδους επέστρεψε αποτέλεσμα με μικρότερη πιθανότητα λάθους αναγνώρισης. Αυτό όμως, απαιτεί περισσότερους πόρους από το cloud.

Στην Εικόνα 5.2.iii απεικονίζουμε τη δομή της αρχιτεκτονικής του παραδείγματος και των πιθανών παραλλαγών που μπορεί να υπάρξουν δίνοντας έμφαση στην ιεραρχία και τη συνδεσμολογία των επιμέρους συστημάτων.




*Εικόνα 5.2.iii: Σχηματική απεικόνιση της υλοποίησης του παραδείγματος*

Στόχος μας σε αυτό το εγχείρημα είναι να αφήσουμε μια ανοιχτή πρόταση για μελλοντική δουλειά και μία λύση στο πρόβλημα της έλλειψης πόρων ενός ρεαλιστικού πολυπρακτορικού συστήματος που θα πρέπει να είναι σε θέση να τρέξει αντίστοιχους αλγορίθμους υψηλών απαιτήσεων σε επεξεργαστική ισχύ που είναι αδύνατον να πραγματοποιηθεί τοπικά σε έναν τυπικό πράκτορα [10][15][16][17][18][19][20][21][22][23].

# Κεφάλαιο 6ο: Αναφορές


[1] Latha, C. P., & Priya, M. (2016). A Review on Deep Learning Algorithms for Speech and Facial Emotion Recognition. *APTIKOM Journal on Computer Science and Information Technologies*, *1*(3), 88-104.

[2] Maghrebi, H., Portigliatti, T., & Prouff, E. (2016, December). Breaking Cryptographic Implementations Using Deep Learning Techniques. In *International Conference on Security, Privacy, and Applied Cryptography Engineering* (pp. 3-26). Springer International Publishing.

[3] Arel, I., Rose, D. C., & Karnowski, T. P. (2010). Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *IEEE computational intelligence magazine*, *5*(4), 13-18.

[4] Salakhutdinov, R., & Hinton, G. (2009, April). Deep boltzmann machines. In *Artificial Intelligence and Statistics* (pp. 448-455).

[5] Salakhutdinov, R., & Murray, I. (2008, July). On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine learning* (pp. 872-879). ACM.

[6] Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, *18*(7), 1527-1554.

[7] Ackley, D. H., Hinton, G. E., & Sejnowski, T. J. (1985). A learning algorithm for Boltzmann machines. *Cognitive science*, *9*(1), 147-169.

[8] Huang, C., Gong, W., Fu, W., & Feng, D. (2014). A research of speech emotion recognition based on deep belief network and SVM. *Mathematical Problems in Engineering*, *2014*.

[9] Deng, L. (2014). A tutorial survey of architectures, algorithms, and applications for deep learning. APSIPA Transactions on Signal and Information Processing, 3.

[10] Deng, L. ACHIEVEMENTS AND CHALLENGES OF DEEP LEARNING.

[11] Fischer, A., & Igel, C. (2012). An introduction to restricted Boltzmann machines. *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, 14-36.

[12] Panait, L., & Luke, S. (2005). Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, *11*(3), 387-434.

[13] Glauner, P. O. (2015). Deep convolutional neural networks for smile recognition. *arXiv preprint arXiv:1508.06535*.

[14] Fischer, A., & Igel, C. (2014). Training restricted Boltzmann machines: An introduction. *Pattern Recognition*, *47*(1), 25-39.

[15] Choi, K., Fazekas, G., Cho, K., & Sandler, M. (2017). A Tutorial on Deep Learning for Music Information Retrieval. *arXiv preprint arXiv:1709.04396*.

[16] Choudhury, P. D., Bora, A., & Sarma, K. K. (2017). Big Spectrum Data and Deep Learning Techniques for Cognitive Wireless Networks. *Deep Learning Innovations and Their Convergence With Big Data*, 33.

[17] Chaudhari, P., & Agarwal, H. (2017). Progressive Review Towards Deep Learning Techniques. In *Proceedings of the International Conference on Data Engineering and Communication Technology* (pp. 151-158). Springer Singapore.

[18] Potluri, S., & Henry, N., Diedrich, C. (2017). Evaluation of Hybrid Deep Learning Techniques for Ensuring Security in Networked Control Systems. *22nd IEEE Conference on Emerging Technologies and Factory Automation*

[19] Zhang, W., Wulan, G., Zhai, J., Xu, L., Zhao, D., Liu, X., ... & Zhou, J. (2017). An Intelligent Power Distribution Service Architecture using Cloud Computing and Deep Learning Techniques. *Journal of Network and Computer Applications*.

[20] Danglade, F., Pernot, J. P., Véron, P., & Fine, L. (2017). A priori evaluation of simulation models preparation processes using artificial intelligence techniques. *Computers in Industry*, *91*, 45-61.

[21] Kumar, K. N., & Christopher, T. (2017). NEURAL NETWORK BASED DEEP LEARNING AND ENSEMBLE TECHNIQUES FOR DATA CLASSIFICATION. *International Journal*, *8*(7).

[22] You, Y., Buluc, A., & Demmel, J. (2017). Scaling Deep Learning on GPU and Knights Landing clusters. *arXiv preprint arXiv:1708.02983*.

[23] Xie, X., Wu, D., Liu, S., & Li, R. (2017). IoT Data Analytics Using Deep Learning. *arXiv preprint arXiv:1708.03854*.

# Κεφάλαιο 7ο: Παράρτημα

## Παράρτημα Α: Αναλυτικοί Πίνακες Αποτελεσμάτων

Στο Παράρτημα Α παρουσιάζονται οι πλήρεις πίνακες του Κεφαλαίου 4 για λόγους πληρότητας της παρούσας διπλωματικής εργασίας.

### Πίνακας Α.1 (DBM (demo)):

Στον επόμενο πίνακα παρουσιάζονται τα σφάλματα κατά τη διαδικασία του pre-training για το τρέξιμο DBM (demo) (παράγραφος 4.1.2):

| EPOCH | PRETRAINING LAYER 1 WITH RBM: 784-500 (ERROR) | PRETRAINING LAYER 2 WITH RBM: 500-1000 (ERROR) | NUMBER OF MISCLASSIFIED TEST EXAMPLES (OUT OF 10000) |
|---|---|---|---|
| **1** | 4576513.0 | 4307508.0 | |
| **2** | 3099715.0 | 2864871.0 | |
| **3** | 2739298.0 | 2537504.0 | |
| **4** | 2543804.0 | 2380925.0 | |
| **5** | 2424677.0 | 2287725.0 | |
| **6** | 2383835.0 | 2342920.0 | |
| **7** | 2265579.0 | 2230057.0 | |
| **8** | 2220720.0 | 2185625.0 | |
| **9** | 2201567.0 | 2165819.0 | |
| **10** | 2182438.0 | 2154326.0 | 278 |
| **11** | 2173337.0 | 2152392.0 | |
| **12** | 2171183.0 | 2148120.0 | |
| **13** | 2168306.0 | 2147336.0 | |
| **14** | 2164663.0 | 2145777.0 | |
| **15** | 2163008.0 | 2143092.0 | |
| **16** | 2158138.0 | 2144703.0 | |
| **17** | 2158697.0 | 2142308.0 | |
| **18** | 2155112.0 | 2141812.0 | |
| **19** | 2157590.0 | 2140987.0 | |
| **20** | 2152326.0 | 2134542.0 | 258 |
| **21** | 2153130.0 | 2519624.0 | |
| **22** | 2152660.0 | 2505163.0 | |
| **23** | 2153075.0 | 2503259.0 | |
| **24** | 2153442.0 | 2498952.0 | |
| **25** | 2151887.0 | 2494729.0 | |
| **26** | 2151184.0 | 2493084.0 | |
| **27** | 2150747.0 | 2489380.0 | |
| **28** | 2152221.0 | 2489014.0 | |
| **29** | 2150795.0 | 2488058.0 | |
| **30** | 2150946.0 | 2482984.0 | 211 |
| **31** | 2150378.0 | 2482678.0 | |
| **32** | 2150031.0 | 2483226.0 | |
| **33** | 2146706.0 | 2481826.0 | |
| **34** | 2148152.0 | 2476949.0 | |

| | | | |
|---|---|---|---|
| **35** | 2147522.0 | 2480221.0 | |
| **36** | 2149523.0 | 2479020.0 | |
| **37** | 2148938.0 | 2475510.0 | |
| **38** | 2148369.0 | 2476181.0 | |
| **39** | 2146476.0 | 2476858.0 | |
| **40** | 2147363.0 | 2475490.0 | 206 |
| **41** | 2148525.0 | 2709605.0 | |
| **42** | 2147015.0 | 2701869.0 | |
| **43** | 2146315.0 | 2699182.0 | |
| **44** | 2145490.0 | 2699152.0 | |
| **45** | 2148543.0 | 2699420.0 | |
| **46** | 2148976.0 | 2697131.0 | |
| **47** | 2145871.0 | 2697794.0 | |
| **48** | 2147670.0 | 2694971.0 | |
| **49** | 2147300.0 | 2691253.0 | |
| **50** | 2144256.0 | 2691550.0 | 199 |
| **51** | 2147839.0 | 2689009.0 | |
| **52** | 2144426.0 | 2690055.0 | |
| **53** | 2145300.0 | 2686250.0 | |
| **54** | 2146046.0 | 2688462.0 | |
| **55** | 2146848.0 | 2688079.0 | |
| **56** | 2145094.0 | 2686501.0 | |
| **57** | 2147098.0 | 2686733.0 | |
| **58** | 2145782.0 | 2684702.0 | |
| **59** | 2144562.0 | 2683577.0 | |
| **60** | 2142355.0 | 2685979.0 | 196 |
| **61** | 2142456.0 | 2862503.0 | |
| **62** | 2144209.0 | 2856192.0 | |
| **63** | 2144187.0 | 2855748.0 | |
| **64** | 2140922.0 | 2858112.0 | |
| **65** | 2139504.0 | 2853870.0 | |
| **66** | 2143623.0 | 2853720.0 | |
| **67** | 2144118.0 | 2851485.0 | |
| **68** | 2143875.0 | 2852658.0 | |
| **69** | 2143374.0 | 2853275.0 | |
| **70** | 2144191.0 | 2850508.0 | 184 |
| **71** | 2142567.0 | 2849959.0 | |
| **72** | 2142563.0 | 2848044.0 | |
| **73** | 2138841.0 | 2846741.0 | |
| **74** | 2141401.0 | 2848757.0 | |
| **75** | 2141590.0 | 2847162.0 | |
| **76** | 2141607.0 | 2845179.0 | |
| **77** | 2140963.0 | 2844623.0 | |
| **78** | 2145029.0 | 2846389.0 | |
| **79** | 2142560.0 | 2845547.0 | |
| **80** | 2141451.0 | 2847133.0 | 205 |
| **81** | 2143040.0 | 2988634.0 | |
| **82** | 2142775.0 | 2991370.0 | |
| **83** | 2142573.0 | 2987239.0 | |
| **84** | 2143560.0 | 2985768.0 | |
| **85** | 2140801.0 | 2990940.0 | |

| | | | |
|---|---|---|---|
| **86** | 2143011.0 | 2984008.0 | |
| **87** | 2142116.0 | 2981351.0 | |
| **88** | 2140626.0 | 2986895.0 | |
| **89** | 2140886.0 | 2983712.0 | |
| **90** | 2144448.0 | 2983197.0 | 196 |
| **91** | 2144400.0 | 2985344.0 | |
| **92** | 2143262.0 | 2983239.0 | |
| **93** | 2141076.0 | 2985908.0 | |
| **94** | 2144548.0 | 2984367.0 | |
| **95** | 2143808.0 | 2982153.0 | |
| **96** | 2141858.0 | 2983326.0 | |
| **97** | 2143557.0 | 2980668.0 | |
| **98** | 2142599.0 | 2978453.0 | |
| **99** | 2142940.0 | 2977419.0 | |
| **100** | 2143051.0 | 2979216.0 | 187 |
| **101** | | 3099595.0 | |
| **102** | | 3103568.0 | |
| **103** | | 3101661.0 | |
| **104** | | 3101553.0 | |
| **105** | | 3095755.0 | |
| **106** | | 3095817.0 | |
| **107** | | 3098846.0 | |
| **108** | | 3098728.0 | |
| **109** | | 3095797.0 | |
| **110** | | 3095406.0 | 187 |
| **111** | | 3100653.0 | |
| **112** | | 3098218.0 | |
| **113** | | 3099282.0 | |
| **114** | | 3099614.0 | |
| **115** | | 3096629.0 | |
| **116** | | 3097291.0 | |
| **117** | | 3096027.0 | |
| **118** | | 3095256.0 | |
| **119** | | 3098598.0 | |
| **120** | | 3098036.0 | 191 |
| **121** | | 3205387.0 | |
| **122** | | 3201829.0 | |
| **123** | | 3203170.0 | |
| **124** | | 3204358.0 | |
| **125** | | 3202871.0 | |
| **126** | | 3201627.0 | |
| **127** | | 3205244.0 | |
| **128** | | 3203515.0 | |
| **129** | | 3198830.0 | |
| **130** | | 3201887.0 | 181 |
| **131** | | 3198716.0 | |
| **132** | | 3197075.0 | |
| **133** | | 3201524.0 | |
| **134** | | 3204767.0 | |
| **135** | | 3198148.0 | |
| **136** | | 3197155.0 | |

| | | |
|---|---|---|
| **137** | 3196652.0 | |
| **138** | 3197546.0 | |
| **139** | 3198399.0 | |
| **140** | 3199172.0 | 191 |
| **141** | 3298768.0 | |
| **142** | 3292790.0 | |
| **143** | 3299591.0 | |
| **144** | 3293160.0 | |
| **145** | 3296309.0 | |
| **146** | 3292887.0 | |
| **147** | 3293384.0 | |
| **148** | 3293018.0 | |
| **149** | 3290979.0 | |
| **150** | 3288018.0 | 186 |
| **151** | 3293259.0 | |
| **152** | 3293351.0 | |
| **153** | 3292922.0 | |
| **154** | 3292782.0 | |
| **155** | 3293224.0 | |
| **156** | 3289584.0 | |
| **157** | 3287446.0 | |
| **158** | 3291380.0 | |
| **159** | 3293459.0 | |
| **160** | 3291374.0 | 183 |
| **161** | 3381177.0 | |
| **162** | 3381624.0 | |
| **163** | 3379964.0 | |
| **164** | 3380639.0 | |
| **165** | 3379381.0 | |
| **166** | 3377407.0 | |
| **167** | 3378063.0 | |
| **168** | 3373119.0 | |
| **169** | 3383926.0 | |
| **170** | 3377919.0 | 179 |
| **171** | 3372260.0 | |
| **172** | 3380163.0 | |
| **173** | 3377498.0 | |
| **174** | 3376670.0 | |
| **175** | 3375756.0 | |
| **176** | 3376259.0 | |
| **177** | 3373182.0 | |
| **178** | 3377639.0 | |
| **179** | 3373432.0 | |
| **180** | 3375733.0 | 177 |
| **181** | 3460745.0 | |
| **182** | 3457319.0 | |
| **183** | 3457112.0 | |
| **184** | 3458841.0 | |
| **185** | 3459849.0 | |
| **186** | 3455636.0 | |
| **187** | 3454022.0 | |

| | | |
|---|---|---|
| **188** | 3454672.0 | |
| **189** | 3453975.0 | |
| **190** | 3457558.0 | 184 |
| **191** | 3454103.0 | |
| **192** | 3456491.0 | |
| **193** | 3454933.0 | |
| **194** | 3452998.0 | |
| **195** | 3453208.0 | |
| **196** | 3450184.0 | |
| **197** | 3456703.0 | |
| **198** | 3458099.0 | |
| **199** | 3453237.0 | |
| **200** | 3452624.0 | 181 |

### Πίνακας Α.2 (DBN (mnistdeepauto)):

Στον επόμενο πίνακα παρουσιάζονται τα Train Squared Errors και τα Test Squared Errors για κάθε *epoch* κατά τη διαδικασία του Fine-tuning ενός deep autoencoder (παράγραφος 4.3.1):

| BEFORE EPOCH | TRAIN SQUARED ERROR | TEST SQUARED ERROR |
|---|---|---|
| **1** | 49.708 | 49.706 |
| **2** | 45.937 | 45.830 |
| **3** | 45.390 | 45.255 |
| **4** | 44.891 | 44.754 |
| **5** | 44.432 | 44.286 |
| **6** | 44.147 | 43.982 |
| **7** | 43.900 | 43.748 |
| **8** | 43.704 | 43.528 |
| **9** | 43.537 | 43.397 |
| **10** | 43.408 | 43.264 |
| **11** | 43.307 | 43.192 |
| **12** | 43.206 | 43.072 |
| **13** | 43.096 | 42.943 |
| **14** | 43.015 | 42.877 |
| **15** | 42.929 | 42.802 |
| **16** | 42.867 | 42.733 |
| **17** | 42.804 | 42.676 |
| **18** | 42.724 | 42.595 |
| **19** | 42.682 | 42.557 |
| **20** | 42.614 | 42.516 |
| **21** | 42.581 | 42.471 |
| **22** | 42.540 | 42.433 |
| **23** | 42.505 | 42.394 |
| **24** | 42.461 | 42.329 |
| **25** | 42.410 | 42.300 |
| **26** | 42.435 | 42.340 |
| **27** | 42.352 | 42.266 |
| **28** | 42.324 | 42.250 |
| **29** | 42.306 | 42.234 |
| **30** | 42.280 | 42.190 |
| **31** | 42.255 | 42.167 |

| | | |
|---|---|---|
| **32** | 42.240 | 42.148 |
| **33** | 42.217 | 42.127 |
| **34** | 42.218 | 42.124 |
| **35** | 42.187 | 42.103 |
| **36** | 42.148 | 42.067 |
| **37** | 42.198 | 42.163 |
| **38** | 42.136 | 42.080 |
| **39** | 42.131 | 42.065 |
| **40** | 42.088 | 42.033 |
| **41** | 42.097 | 42.063 |
| **42** | 42.069 | 42.014 |
| **43** | 42.053 | 42.000 |
| **44** | 42.031 | 41.989 |
| **45** | 42.010 | 41.977 |
| **46** | 42.002 | 41.962 |
| **47** | 41.981 | 41.949 |
| **48** | 41.966 | 41.943 |
| **49** | 41.954 | 41.920 |
| **50** | 41.950 | 41.914 |
| **51** | 41.945 | 41.901 |
| **52** | 41.914 | 41.886 |
| **53** | 41.895 | 41.869 |
| **54** | 41.890 | 41.861 |
| **55** | 41.869 | 41.843 |
| **56** | 41.860 | 41.845 |
| **57** | 41.849 | 41.835 |
| **58** | 41.838 | 41.830 |
| **59** | 41.827 | 41.807 |
| **60** | 41.817 | 41.804 |
| **61** | 41.804 | 41.806 |
| **62** | 41.803 | 41.803 |
| **63** | 41.794 | 41.794 |
| **64** | 41.778 | 41.779 |
| **65** | 41.769 | 41.771 |
| **66** | 41.759 | 41.766 |
| **67** | 41.752 | 41.758 |
| **68** | 41.744 | 41.752 |
| **69** | 41.731 | 41.735 |
| **70** | 41.729 | 41.740 |
| **71** | 41.722 | 41.735 |
| **72** | 41.722 | 41.732 |
| **73** | 41.711 | 41.726 |
| **74** | 41.700 | 41.718 |
| **75** | 41.707 | 41.717 |
| **76** | 41.686 | 41.704 |
| **77** | 41.679 | 41.696 |
| **78** | 41.668 | 41.696 |
| **79** | 41.667 | 41.689 |
| **80** | 41.659 | 41.682 |
| **81** | 41.642 | 41.669 |
| **82** | 41.639 | 41.674 |

| | | |
|---|---|---|
| **83** | 41.631 | 41.667 |
| **84** | 41.632 | 41.660 |
| **85** | 41.622 | 41.646 |
| **86** | 41.617 | 41.650 |
| **87** | 41.615 | 41.648 |
| **88** | 41.605 | 41.641 |
| **89** | 41.600 | 41.635 |
| **90** | 41.598 | 41.638 |
| **91** | 41.592 | 41.624 |
| **92** | 41.581 | 41.617 |
| **93** | 41.583 | 41.619 |
| **94** | 41.582 | 41.615 |
| **95** | 41.572 | 41.606 |
| **96** | 41.565 | 41.602 |
| **97** | 41.564 | 41.599 |
| **98** | 41.563 | 41.594 |
| **99** | 41.558 | 41.585 |
| **100** | 41.551 | 41.585 |
| **101** | 41.547 | 41.581 |
| **102** | 41.543 | 41.577 |
| **103** | 41.543 | 41.572 |
| **104** | 41.531 | 41.568 |
| **105** | 41.524 | 41.558 |
| **106** | 41.517 | 41.553 |
| **107** | 41.519 | 41.558 |
| **108** | 41.513 | 41.556 |
| **109** | 41.507 | 41.544 |
| **110** | 41.501 | 41.541 |
| **111** | 41.495 | 41.536 |
| **112** | 41.490 | 41.533 |
| **113** | 41.490 | 41.535 |
| **114** | 41.487 | 41.526 |
| **115** | 41.476 | 41.519 |
| **116** | 41.478 | 41.518 |
| **117** | 41.466 | 41.514 |
| **118** | 41.468 | 41.517 |
| **119** | 41.462 | 41.508 |
| **120** | 41.457 | 41.505 |
| **121** | 41.450 | 41.495 |
| **122** | 41.449 | 41.497 |
| **123** | 41.442 | 41.488 |
| **124** | 41.439 | 41.484 |
| **125** | 41.438 | 41.481 |
| **126** | 41.435 | 41.481 |
| **127** | 41.425 | 41.472 |
| **128** | 41.424 | 41.468 |
| **129** | 41.422 | 41.468 |
| **130** | 41.412 | 41.463 |
| **131** | 41.412 | 41.459 |
| **132** | 41.411 | 41.456 |
| **133** | 41.405 | 41.453 |

| | | |
|---|---|---|
| **134** | 41.404 | 41.448 |
| **135** | 41.397 | 41.440 |
| **136** | 41.394 | 41.439 |
| **137** | 41.381 | 41.428 |
| **138** | 41.381 | 41.424 |
| **139** | 41.382 | 41.424 |
| **140** | 41.377 | 41.419 |
| **141** | 41.370 | 41.415 |
| **142** | 41.369 | 41.411 |
| **143** | 41.364 | 41.404 |
| **144** | 41.365 | 41.405 |
| **145** | 41.351 | 41.392 |
| **146** | 41.354 | 41.394 |
| **147** | 41.352 | 41.394 |
| **148** | 41.346 | 41.388 |
| **149** | 41.341 | 41.383 |
| **150** | 41.334 | 41.377 |
| **151** | 41.333 | 41.377 |
| **152** | 41.330 | 41.374 |
| **153** | 41.323 | 41.369 |
| **154** | 41.321 | 41.364 |
| **155** | 41.318 | 41.359 |
| **156** | 41.312 | 41.353 |
| **157** | 41.306 | 41.346 |
| **158** | 41.306 | 41.343 |
| **159** | 41.299 | 41.342 |
| **160** | 41.294 | 41.333 |
| **161** | 41.291 | 41.332 |
| **162** | 41.292 | 41.332 |
| **163** | 41.286 | 41.328 |
| **164** | 41.285 | 41.327 |
| **165** | 41.278 | 41.319 |
| **166** | 41.275 | 41.316 |
| **167** | 41.273 | 41.312 |
| **168** | 41.270 | 41.309 |
| **169** | 41.268 | 41.309 |
| **170** | 41.264 | 41.303 |
| **171** | 41.261 | 41.300 |
| **172** | 41.256 | 41.299 |
| **173** | 41.256 | 41.295 |
| **174** | 41.254 | 41.294 |
| **175** | 41.253 | 41.294 |
| **176** | 41.244 | 41.286 |
| **177** | 41.241 | 41.279 |
| **178** | 41.239 | 41.276 |
| **179** | 41.238 | 41.275 |
| **180** | 41.234 | 41.272 |
| **181** | 41.232 | 41.271 |
| **182** | 41.229 | 41.267 |
| **183** | 41.226 | 41.265 |
| **184** | 41.222 | 41.262 |

| | | |
|---|---|---|
| **185** | 41.220 | 41.258 |
| **186** | 41.215 | 41.255 |
| **187** | 41.212 | 41.252 |
| **188** | 41.211 | 41.246 |
| **189** | 41.208 | 41.247 |
| **190** | 41.204 | 41.243 |
| **191** | 41.201 | 41.239 |
| **192** | 41.197 | 41.239 |
| **193** | 41.195 | 41.235 |
| **194** | 41.192 | 41.233 |
| **195** | 41.189 | 41.230 |
| **196** | 41.187 | 41.230 |
| **197** | 41.185 | 41.224 |
| **198** | 41.180 | 41.222 |
| **199** | 41.177 | 41.213 |
| **200** | 41.174 | 41.216 |

### Πίνακας Α.3 (DBN (mnistclassify)):

Στον επόμενο πίνακα παρουσιάζονται τα πλήθη των αριθμών που δεν κατάφεραν να κατηγοριοποιηθούν πριν από κάθε epoch τόσο κατά το training (Train # Misclassified) όσο και κατά το learning (Test # Misclassified) (παράγραφος 4.3.2):

| BEFORE EPOCH | TRAIN # MISCLASSIFIED (FROM 60000) | TEST # MISCLASSIFIED (FROM 10000) |
|---|---|---|
| **1** | 51142 | 8468 |
| **2** | 38242 | 6351 |
| **3** | 38154 | 6336 |
| **4** | 38155 | 6338 |
| **5** | 38157 | 6338 |
| **6** | 38160 | 6338 |
| **7** | 32796 | 5400 |
| **8** | 29548 | 4876 |
| **9** | 27548 | 4523 |
| **10** | 26646 | 4362 |
| **11** | 24990 | 4136 |
| **12** | 23598 | 3905 |
| **13** | 22246 | 3716 |
| **14** | 21456 | 3585 |
| **15** | 20931 | 3547 |
| **16** | 19787 | 3315 |
| **17** | 19392 | 3268 |
| **18** | 18903 | 3174 |
| **19** | 18610 | 3138 |
| **20** | 18606 | 3072 |
| **21** | 17960 | 3015 |
| **22** | 17682 | 3006 |
| **23** | 17454 | 2932 |
| **24** | 17487 | 2957 |
| **25** | 17278 | 2930 |
| **26** | 17112 | 2902 |

| | | |
|---|---|---|
| **27** | 17175 | 2928 |
| **28** | 16736 | 2852 |
| **29** | 16854 | 2874 |
| **30** | 16720 | 2827 |
| **31** | 16669 | 2804 |
| **32** | 16675 | 2820 |
| **33** | 16692 | 2835 |
| **34** | 16543 | 2808 |
| **35** | 16644 | 2829 |
| **36** | 16434 | 2817 |
| **37** | 16144 | 2755 |
| **38** | 16121 | 2754 |
| **39** | 15997 | 2731 |
| **40** | 16113 | 2740 |
| **41** | 16256 | 2782 |
| **42** | 15930 | 2715 |
| **43** | 15936 | 2725 |
| **44** | 15930 | 2745 |
| **45** | 15575 | 2666 |
| **46** | 15796 | 2735 |
| **47** | 15725 | 2724 |
| **48** | 15456 | 2685 |
| **49** | 15460 | 2694 |
| **50** | 15637 | 2706 |
| **51** | 15222 | 2631 |
| **52** | 15361 | 2671 |
| **53** | 15105 | 2593 |
| **54** | 14634 | 2510 |
| **55** | 13992 | 2416 |
| **56** | 13632 | 2346 |
| **57** | 12746 | 2224 |
| **58** | 12182 | 2124 |
| **59** | 11929 | 2081 |
| **60** | 11263 | 1965 |
| **61** | 11116 | 1941 |
| **62** | 11135 | 1947 |
| **63** | 10830 | 1925 |
| **64** | 10904 | 1953 |
| **65** | 10746 | 1950 |
| **66** | 10488 | 1850 |
| **67** | 10637 | 1964 |
| **68** | 10543 | 1903 |
| **69** | 10404 | 1876 |
| **70** | 10614 | 1946 |
| **71** | 10519 | 1913 |
| **72** | 10444 | 1862 |
| **73** | 10324 | 1831 |
| **74** | 10319 | 1835 |
| **75** | 10218 | 1840 |
| **76** | 10535 | 1912 |
| **77** | 10168 | 1827 |

| | | |
|---|---|---|
| **78** | 10467 | 1878 |
| **79** | 10449 | 1895 |
| **80** | 10339 | 1884 |
| **81** | 10105 | 1823 |
| **82** | 10288 | 1887 |
| **83** | 10163 | 1846 |
| **84** | 10066 | 1821 |
| **85** | 10110 | 1813 |
| **86** | 10041 | 1798 |
| **87** | 10215 | 1880 |
| **88** | 9923 | 1798 |
| **89** | 10124 | 1827 |
| **90** | 9902 | 1793 |
| **91** | 10067 | 1838 |
| **92** | 9894 | 1786 |
| **93** | 10002 | 1818 |
| **94** | 9928 | 1822 |
| **95** | 9871 | 1787 |
| **96** | 9809 | 1777 |
| **97** | 9794 | 1764 |
| **98** | 9817 | 1765 |
| **99** | 9982 | 1816 |
| **100** | 9879 | 1807 |
| **101** | 9885 | 1816 |
| **102** | 9740 | 1765 |
| **103** | 9685 | 1778 |
| **104** | 9866 | 1799 |
| **105** | 9807 | 1821 |
| **106** | 9895 | 1832 |
| **107** | 9761 | 1780 |
| **108** | 9670 | 1755 |
| **109** | 9606 | 1766 |
| **110** | 9525 | 1747 |
| **111** | 9622 | 1812 |
| **112** | 9819 | 1825 |
| **113** | 9553 | 1772 |
| **114** | 9665 | 1768 |
| **115** | 9738 | 1806 |
| **116** | 1008 | 1862 |
| **117** | 1008 | 1841 |
| **118** | 9781 | 1804 |
| **119** | 9725 | 1805 |
| **120** | 9455 | 1738 |
| **121** | 9768 | 1800 |
| **122** | 9713 | 1775 |
| **123** | 9916 | 1823 |
| **124** | 9422 | 1729 |
| **125** | 9742 | 1825 |
| **126** | 9715 | 1779 |
| **127** | 9727 | 1777 |
| **128** | 9792 | 1768 |

| | | |
|---|---|---|
| **129** | 9529 | 1742 |
| **130** | 9591 | 1777 |
| **131** | 9622 | 1779 |
| **132** | 9422 | 1715 |
| **133** | 9863 | 1816 |
| **134** | 9456 | 1735 |
| **135** | 9667 | 1779 |
| **136** | 9821 | 1810 |
| **137** | 9655 | 1775 |
| **138** | 9484 | 1738 |
| **139** | 9630 | 1765 |
| **140** | 9725 | 1798 |
| **141** | 9440 | 1721 |
| **142** | 9549 | 1766 |
| **143** | 9748 | 1800 |
| **144** | 9677 | 1800 |
| **145** | 9685 | 1768 |
| **146** | 9624 | 1768 |
| **147** | 9688 | 1808 |
| **148** | 9643 | 1768 |
| **149** | 9471 | 1749 |
| **150** | 9647 | 1798 |
| **151** | 9636 | 1777 |
| **152** | 9562 | 1759 |
| **153** | 9472 | 1775 |
| **154** | 9668 | 1799 |
| **155** | 9604 | 1786 |
| **156** | 9265 | 1709 |
| **157** | 9197 | 1702 |
| **158** | 9458 | 1734 |
| **159** | 9474 | 1772 |
| **160** | 9396 | 1736 |
| **161** | 9376 | 1731 |
| **162** | 9379 | 1728 |
| **163** | 9176 | 1706 |
| **164** | 9373 | 1718 |
| **165** | 9196 | 1694 |
| **166** | 9406 | 1756 |
| **167** | 9197 | 1682 |
| **168** | 9167 | 1682 |
| **169** | 9203 | 1692 |
| **170** | 9326 | 1735 |
| **171** | 9341 | 1717 |
| **172** | 9514 | 1761 |
| **173** | 9253 | 1709 |
| **174** | 9157 | 1678 |
| **175** | 9274 | 1716 |
| **176** | 9094 | 1682 |
| **177** | 9295 | 1723 |
| **178** | 9116 | 1708 |
| **179** | 9046 | 1687 |

| | | |
|---|---|---|
| **180** | 9168 | 1715 |
| **181** | 9259 | 1736 |
| **182** | 9063 | 1707 |
| **183** | 9390 | 1743 |
| **184** | 9391 | 1705 |
| **185** | 9142 | 1721 |
| **186** | 9046 | 1687 |
| **187** | 9033 | 1681 |
| **188** | 9037 | 1704 |
| **189** | 8995 | 1672 |
| **190** | 9066 | 1708 |
| **191** | 8997 | 1659 |
| **192** | 9021 | 1677 |
| **193** | 9018 | 1691 |
| **194** | 8983 | 1690 |
| **195** | 8975 | 1667 |
| **196** | 9103 | 1688 |
| **197** | 9094 | 1693 |
| **198** | 8986 | 1687 |
| **199** | 8950 | 1673 |
| **200** | 8995 | 1691 |

## Παράρτημα Β: Αναλυτικοί Χρόνοι Τρεξίματος (Matlab Profiler)

Στο Παράρτημα Β παρουσιάζονται screenshots από το Profiler του Matlab για να αποσαφηνιστεί σε βάθος ο χρόνος εκτέλεσης των επιμέρους συναρτήσεων των προγραμμάτων για κάθε μέθοδο (Εικόνες 7.B.i, 7.B.ii, 7.B.iii, 7.B.iv, 7.B.v, 7.B.vi)

**Profile Summary**

*Generated 24-Sep-2017 07:14:14 using cpu time.*

| Function Name | Calls | Total Time | Self Time* | Total Time Plot (dark band = self time) |
|---|---|---|---|---|
| demo_small | 1 | 1806.418 s | 0.239 s | |
| dbm_mf | 1 | 1191.091 s | 715.650 s | |
| mf | 6000 | 463.956 s | 463.956 s | |
| backprop | 1 | 425.646 s | 13.429 s | |
| mf_class | 700 | 361.517 s | 361.517 s | |
| converter | 1 | 77.294 s | 77.265 s | |
| rbm | 1 | 75.386 s | 74.315 s | |
| minimize | 60 | 50.700 s | 0.400 s | |
| CG_MNIST_INIT | 263 | 25.632 s | 25.632 s | |
| CG_MNIST | 256 | 24.639 s | 24.639 s | |
| rbm_l2 | 1 | 22.435 s | 22.183 s | |
| makebatches | 4 | 14.181 s | 14.181 s | |

*Εικόνα 7.B.i: Χρόνοι τρεξίματος για DBM demo_small ανά συνάρτηση*

**Profile Summary**

*Generated 25-Sep-2017 18:40:18 using cpu time.*

| Function Name | Calls | Total Time | Self Time* | Total Time Plot (dark band = self time) |
|---|---|---|---|---|
| demo | 1 | 27409.280 s | 0.104 s | |
| rbm_l2 | 1 | 17696.961 s | 17576.177 s | |
| dbm_mf | 1 | 7361.957 s | 4726.552 s | |
| mf | 24246 | 2616.372 s | 2616.372 s | |
| rbm | 1 | 2260.366 s | 2256.067 s | |
| testerr | 20 | 120.784 s | 120.784 s | |
| converter | 1 | 79.692 s | 79.667 s | |
| dispims | 584 | 23.323 s | 5.962 s | |
| imagesc | 584 | 11.065 s | 5.149 s | |
| makebatches | 3 | 10.056 s | 10.056 s | |

*Εικόνα 7.B.ii: Χρόνοι τρεξίματος για DBM demo ανά συνάρτηση*

## Profile Summary

*Generated 23-Sep-2017 18:05:40 using cpu time.*

| Function Name | Calls | **Total Time** | Self Time* | Total Time Plot (dark band = self time) |
|---|---|---|---|---|
| demo_toy | 1 | 278.029 s | 0.148 s | |
| rbm | 1 | 143.097 s | 143.097 s | |
| converter | 1 | 84.746 s | 84.718 s | |
| RBM_AIS | 1 | 38.497 s | 37.062 s | |
| makebatches | 1 | 5.479 s | 5.479 s | |
| calculate_logprob | 2 | 5.167 s | 5.167 s | |
| mnistdisp | 11 | 1.216 s | 0.340 s | |

*Εικόνα 7.B.iii: Χρόνοι τρεξίματος για RBM demo_toy ανά συνάρτηση*

## Profile Summary

*Generated 25-Sep-2017 09:18:06 using cpu time.*

| Function Name | Calls | **Total Time** | Self Time* | Total Time Plot (dark band = self time) |
|---|---|---|---|---|
| demo_rbm | 1 | 236.329 s | 0.112 s | |
| RBM_AIS | 1 | 228.208 s | 226.160 s | |
| makebatches | 1 | 5.227 s | 5.227 s | |
| calculate_logprob | 1 | 2.770 s | 2.770 s | |
| mnistdisp | 29 | 1.814 s | 0.474 s | |

*Εικόνα 7.B.iv: Χρόνοι τρεξίματος για RBM demo_rbm ανά συνάρτηση*

**Profile Summary**

*Generated 23-Sep-2017 22:28:22 using cpu time.*

| Function Name | Calls | **Total Time** | Self Time* | Total Time Plot (dark band = self time) |
|---|---|---|---|---|
| mnistdeepauto | 1 | 9810.940 s | 0.259 s | |
| backprop | 1 | 9518.795 s | 927.706 s | |
| minimize | 12000 | 8573.911 s | 83.759 s | |
| CG_MNIST | 103987 | 8490.152 s | 8490.152 s | |
| rbm | 3 | 150.596 s | 150.596 s | |
| converter | 1 | 78.459 s | 78.433 s | |
| rbmhidlinear | 1 | 59.316 s | 59.316 s | |
| mnistdisp | 200 | 13.822 s | 2.903 s | |
| imagesc | 200 | 7.327 s | 4.130 s | |
| makebatches | 2 | 6.714 s | 6.714 s | |

*Εικόνα 7.B.v: Χρόνοι τρεξίματος για DBN mnistdeepauto ανά συνάρτηση*

**Profile Summary**

*Generated 28-Sep-2017 04:49:30 using cpu time.*

| Function Name | Calls | **Total Time** | Self Time* | Total Time Plot (dark band = self time) |
|---|---|---|---|---|
| mnistclassify | 1 | 2808.641 s | 0.168 s | |
| backpropclassify | 1 | 2656.913 s | 722.041 s | |
| minimize | 12000 | 1931.542 s | 78.006 s | |
| CG_CLASSIFY | 114174 | 1850.259 s | 1850.259 s | |
| converter | 1 | 81.923 s | 81.888 s | |
| rbm | 3 | 66.304 s | 66.304 s | |
| makebatches | 2 | 6.654 s | 6.654 s | |

*Εικόνα 7.B.vi: Χρόνοι τρεξίματος για DBN mnistclassify ανά συνάρτηση*