ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΣΥΝΕΛΙΚΤΙΚΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΕΥΦΥΪΑΣ ΣΜΗΝΟΥΣ

Διπλωματική Εργασία

Δημήτριος Κουκουγιάννης

Επιβλέπων: Δημήτριος Κατσαρός

Ιούνιος 2022

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΣΥΝΕΛΙΚΤΙΚΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΕΥΦΥΪΑΣ ΣΜΗΝΟΥΣ

Διπλωματική Εργασία

Δημήτριος Κουκουγιάννης

Επιβλέπων: Δημήτριος Κατσαρός

Ιούνιος 2022

UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

OPTIMIZATION OF CONVOLUTIONAL NEURAL NETWORKS ARCHITECTURE USING SWARM INTELLIGENCE

Diploma Thesis

Dimitrios Koukougiannis

Supervisor: Dimitrios Katsaros

June 2022

Εγκρίνεται από την Επιτροπή Εξέτασης:

Επιβλέπων **Δημήτριος Κατσαρός**
Αναπληρωτής καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Δημήτριος Ραφαηλίδης**
Αναπληρωτής καθηγητής, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Μέλος **Γεώργιος Θάνος**
Ε.ΔΙ.Π., Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Αρχικά θα ήθελα να ευχαριστήσω τον κύριο Δημήτριο Κατσαρό για την καθοδήγηση και τις συμβουλές που μου παρείχε καθόλη τη διάρκεια της εκπόνησης της διπλωματικής, καθώς επίσης και τα μέλη της επιτροπής εξέτασης. Επίσης θα ήθελα να εκφράσω ευχαριστίες στην οικογένεια και τους φίλους μου για την υποστήριξη που μου παρείχαν όλα αυτά τα χρόνια.

ΥΠΕΥΘΥΝΗ ΔΗΛΩΣΗ ΠΕΡΙ ΑΚΑΔΗΜΑΪΚΗΣ ΔΕΟΝΤΟΛΟΓΙΑΣ ΚΑΙ ΠΝΕΥΜΑΤΙΚΩΝ ΔΙΚΑΙΩΜΑΤΩΝ

«Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ρητά ότι η παρούσα διπλωματική εργασία, καθώς και τα ηλεκτρονικά αρχεία και πηγαίοι κώδικες που αναπτύχθηκαν ή τροποποιήθηκαν στα πλαίσια αυτής της εργασίας, αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης. Τα σημεία όπου έχω χρησιμοποιήσει ιδέες, κείμενο, αρχεία ή/και πηγές άλλων συγγραφέων, αναφέρονται ευδιάκριτα στο κείμενο με την κατάλληλη παραπομπή και η σχετική αναφορά περιλαμβάνεται στο τμήμα των βιβλιογραφικών αναφορών με πλήρη περιγραφή. Δηλώνω επίσης ότι τα αποτελέσματα της εργασίας δεν έχουν χρησιμοποιηθεί για την απόκτηση άλλου πτυχίου. Αναλαμβάνω πλήρως, ατομικά και προσωπικά, όλες τις νομικές και διοικητικές συνέπειες που δύναται να προκύψουν στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δεν μου ανήκει διότι είναι προϊόν λογοκλοπής».

Ο Δηλών

Δημήτριος Κουκουγιάννης

Διπλωματική Εργασία

ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΣΥΝΕΛΙΚΤΙΚΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΕΥΦΥΪΑΣ ΣΜΗΝΟΥΣ

Δημήτριος Κουκουγιάννης

Περίληψη

Η εύρεση της κατάλληλης αρχιτεκτονικής νευρωνικών δικτύων για την κατηγοριοποίηση εικόνων από υπολογιστή έχει εξελιχθεί σε ένα πολύ δύσκολο αντικείμενο και έχει κεντρίσει το ενδιαφέρον πολλών ερευνητών. Σκοπός αυτής της διπλωματικής είναι η ανάπτυξη μίας μεθόδου η οποία θα εντοπίζει βασικές επαναλαμβανόμενες δομές, που ονομάζονται κελιά, και μπορούν να σχηματίσουν σύνθετες αρχιτεκτονικές αν τοποθετηθούν σειριακά. Για αυτό το λόγο θα βασιστούμε στη συμπεριφορά των μυρμηγκιών στη φύση, αντί για τη χρήση άλλων νευρωνικών δικτύων. Όπως θα γίνει σαφές η μέθοδος μας παράγει αρχιτεκτονικές συγκρίσιμες σε ακρίβεια με άλλες μεθόδους που έχουν προταθεί στο παρελθόν για δύο από τα τρία ΣΔ που θα χρησιμοποιηθούν, ενώ στο τρίτο αν και η ακρίβεια είχε μεγάλη απόκλιση σε σχέση με άλλα δίκτυα, η μέθοδος μας παρήγαγε πολύ μικρότερες αρχιτεκτονικές όσον αφορά τον αριθμό των μεταβλητών, ενώ οι δομές μπορούν εύκολα να συνδυαστούν για να δημιουργήσουν καλύτερα ΤΝΔ.

Λέξεις-κλειδιά:

αναζήτηση αρχιτεκτονικής νευρωνικών δικτύων, ευφυΐα σμηνους, όραση υπολογιστή

Diploma Thesis

OPTIMIZATION OF CONVOLUTIONAL NEURAL NETWORKS ARCHITECTURE USING SWARM INTELLIGENCE

Dimitrios Koukougiannis

Abstract

Finding the suitable architecture in order to categorize pictures by a computer has evolved into a very demanding task and has gathered many researchers' attention. This thesis aims at developing a method, which creates basic iterating structures called cells, which in turn can form complex architectures if they are stacked one after the other. For this reason we are going to be based on the behavior of ants, instead of other neural networks. As it is going to be shown our method produces architectures equally good for two out of three of the datasets with others proposed in different articles and as for the third one, although it was not as good as other techniques, the architectures created by our method have much less variables. Moreover these cells are scalable and can be easily combined, in order to produce better neural networks.

Keywords:

neural architecture search, swarm intelligence, computer vision

Πίνακας περιεχομένων

Κατάλογος σχημάτων

Κατάλογος πινάκων

Συντομογραφίες

BP	Backpropagation
CNN	Convolutional neural networks
ΤΝΔ	Τεχνητά νευρωνικά δίκτυα
MLP	Multi-layer perceptron
ΣΔ	Σύνολο δεδομένων
ACO	Ant colony optimization
PSO	Particle swarm optimization

Κεφάλαιο 1

Εισαγωγή

Η ανάπτυξη της μηχανικής μάθησης έχει οδηγήσει στην ευρεία χρήση των τεχνητών νευρωνικών δικτύων (ΤΝΔ) για επίλυση διαφόρων προβλημάτων από υπολογιστή. Ωστόσο η εύρεση της κατάλληλης αρχιτεκτονικής για κάθε σύνολο δεδομένων μπορεί να είναι ιδιαίτερα απαιτητική και για αυτό το λόγο τα τελευταία χρόνια γίνεται προσπάθεια από πλήθος ερευνητών για την αυτοματοποίηση της διαδικασίας αυτής. Η αναζήτηση αρχιτεκτονικής ΤΝΔ απαιτεί όμως τεράστια υπολογιστική ισχύ, ενώ τα δίκτυα τα οποία προκύπτουν από τις περισσότερες μεθόδους που έχουν προταθεί μέχρι τώρα δεν είναι εύκολο να κλιμακωθούν ώστε να προκύψουν πιο σύνθετες δομές.

1.1 Αντικείμενο της διπλωματικής

Στο πλαίσιο της παρούσας διπλωματικής λοιπόν θα αναπτυχθεί μία μέθοδος η οποία θα κατασκευάζει απλές επαναλαμβανόμενες δομές (κελιά) που τοποθετούνται σειριακά, ώστε να δημιουργήσουν μία κλιμακώσιμη αρχιτεκτονική, της οποίας η πολυπλοκότητα εξαρτάται από τον αριθμό αυτών των δομών. Για την ανάπτυξη της θα βασιστούμε σε τεχνικές ευφυΐας σμήνους (ΕΣ), οι οποίες έχουν προταθεί στο παρελθόν και είναι εμπνευσμένες από την παρατήρηση της συμπεριφοράς μεγάλων πλυθησμών ενός είδους στη φύση, που συνεργάζονται μεταξύ τους για να επιβιώσουν.

1.1.1 Συνεισφορά

Η συνεισφορά της διπλωματικής συνοψίζεται ως εξής:

1. Υλοποιήθηκε ένας αλγόριθμος ο οποίος δημιουργεί τις δομές που αναφέρθηκαν παραπάνω και μία τεχνική για την ενημέρωση της τιμής μίας εκ των παραμέτρων του πρώτου.

2. Αξιολογήθηκε η επίδοση του αλγορίθμου σε τρία σύνολα δεδομένων και διαπιστώθηκε ότι δημιουργεί αρχιτεκτονικές με ακρίβεια συγκρίσιμη με άλλες τεχνικές που έχουν προταθεί για τα δύο από τα τρία, ενώ για το τελευταίο τα ΤΝΔ που προέκυπταν είχαν πολύ λιγότερες μεταβλητές.

3. Διατυπώθηκαν προτάσεις για τη βελτίωση της επίδοση της τεχνικής, τις οποίες δε μπορούσαμε να εκτελέσουμε καθώς δε διαθέταμε αρκετή υπολογιστική ισχύ.

1.2 Οργάνωση του τόμου

Στο κεφάλαιο 2 αναλύονται τα ΤΝΔ και περιγράφονται οι βασικές αρχές από τη θεωρία τους, οι οποίες χρησιμοποιούνται στο πλαίσιο αυτής της διπλωματικής, ενώ στο κεφάλαιο 3 περιγράφονται κάποιες αρχές μεθόδων ευφυΐας σμήνους. Στο κεφάλαιο 4 γίνεται μελέτη και ανάλυση άρθρων τα οποία σχετίζονται με το αντικείμενο της διπλωματικής, στα οποία βασιστήκαμε ώστε να γίνει η ανάπτυξη του δικού μας αλγορίθμου, ο οποίος περιγράφεται στο κεφάλαιο 5. Για τον έλεγχο της ακρίβειας των αρχιτεκτονικών που παράγονται από τη μέθοδο υλοποιήσαμε τα πειράματα που αναφέρονται στο κεφάλαιο 6, στο οποίο μάλιστα παρέχονται και τα αποτελέσματα που λάβαμε. Τέλος στο κεφάλαιο 7 γίνεται σχολιασμός των επιδόσεων του αλγορίθμου και προτείνονται τρόποι για τη βελτιστοποίηση των δομών που παράγει. Αξίζει να σημειωθεί ότι παρέχονται και δύο παραρτήματα, στο πρώτο εκ των οποίων απεικονίζονται οι δομές που παρείχαν τις καλύτερες επιδόσεις, ενώ στο δεύτερο αναφέρονται κάποιες λεπτομέριες για τη δομή και την εκπαίδευση των δικτύων που σχηματίζονταν.

Κεφάλαιο 2

Τεχνητά νευρωνικά δίκτυα

2.1 Εισαγωγή

Ο ανθρώπινος εγκέφαλος αποτελεί ένα αξιοθαύμαστο όργανο, το οποίο εξαιτίας της εξέλιξης έχει τη δυνατότητα να εκτελεί περίπλοκες λειτουργίες σε πολύ μικρά χρονικά διαστήματα. Η ικανότητα του αυτή τον έχει καταστήσει αντικείμενο μελέτης από πλήθος επιστημόνων, οι οποίοι γρήγορα διαπίστωσαν ότι ο τρόπος με τον οποίον το επιτυγχάνει είναι εξαιρετικά περίπλοκος. Η λειτουργία του βασίζεται στους νευρώνες, οι οποίοι επικοινωνούν μεταξύ τους με ηλεκτρικά σήματα των οποίων η τιμή μεταβάλλεται ανάλογα τη σύναψη από την οποία διέρχονται. Ωστόσο η συμπεριφορά των νευρώνων δεν είναι αρκετή για να εξηγήσει, πως ο εγκέφαλος μπορεί να επεξεργαστεί με ακρίβεια τόσες πολλές πληροφορίες. Η απάντηση σε αυτό το ερώτημα βρίσκεται στην ίδια του την ανάπτυξη. Κατά τη γέννηση ο εγκέφαλος έχει μία βασική δομή, η οποία του επιτρέπει να εξάγει κανόνες από προηγούμενες καταστάσεις και με την πάροδο του χρόνου εξελίσσεται ώστε να επιτελεί δυσκολότερες εργασίες. Με αυτόν τον τρόπο είναι σε θέση να εκτελέσει όλες τις απαραίτητες λειτουργίες για έναν άνθρωπο, όπως η όραση, η ακοή αλλά και άλλα.

2.2 Πρώτα μοντέλα νευρωνικών δικτύων

Όλες οι παραπάνω δυνατότητες ώθησαν πλήθος ερευνητών στην προσπάθεια να μοντελοποιήσουν νευρωνικά δίκτυα, που συναντώνται στη βιολογία, ώστε να μπορούν να περιγραφούν μαθηματικά και να χρησιμοποιηθούν για την επίλυση ανθρώπινων προβλημάτων.

2.2.1 Ο νευρώνας

Η πρώτη απόπειρα έγινε με τη μοντελοποίηση ενός νευρώνα από τους McCulloch και Pits. Ο νευρώνας αυτός λαμβάνει ένα σύνολο από εισόδους, οι οποίες πολλαπλασιάζονται με πραγματικούς αριθμούς, οι οποίοι ονομάζονται βάρη. Ο νευρώνας δηλαδή υπολογίζει το άθροισμα

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

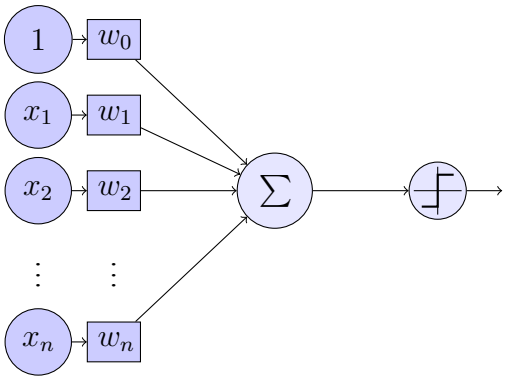
Αν το παραπάνω άθροισμα είναι μεγαλύτερο από ένα κατώφλι θ, τότε η έξοδος y είναι ίση με 1, δηλαδή ο νευρώνας πυροβολεί. Σε διαφορετική περίπτωση η έξοδος είναι ίση με 0 και τότε λέγεται ότι ο νευρώνας δεν πυροβολεί. Η παραπάνω ιδιότητα ορίζεται από τη σχέση $y = step(u - \theta)$, όπου $step$ είναι η βηματική συνάρτηση, η οποία ορίζεται ως

$$step(x) = \begin{cases} 1, \ x > 0 \\ 0, \ x \leq 0 \end{cases}$$

Ο νευρώνας αποτελεί σημαντικό στοιχείο στην ανάπτυξη ΤΝΔ, καθώς είναι το συστατικό τους στοιχείο.

2.2.2 Το δίκτυο Perceptron

Ο ορισμός του νευρώνα άνοιξε το δρόμο για την ανάπτυξη μοντέλων νευρωνικών δικτύων. Έτσι το 1958 δημιουργήθηκε από τον Rosenblatt το δίκτυο Perceptron. Απoντελούνταν από ένα νευρώνα και η είσοδος του ήταν ένα διάνυσμα $x = [x_1, x_2, \ldots, x_n]$ το οποίο πολλαπλασιάζεται με ένα διάνυσμα βαρών $w = [w_1, w_2, \ldots, w_n]$ και στη συνέχεια αφαιρούνταν ένα κατώφλι θ. Τέλος ορίστηκε και η συνάρτηση ενεργοποίησης, η οποία δίνει στην τελική έξοδο y μία δυαδική μορφή, είτε 1/0, είτε 1/-1. Στο σχήμα 2.1 απεικονίζεται το μοντέλο σύμφωνα με το [11]. Αξίζει να σημειωθεί ότι το διάνυσμα βαρών μπορεί να γραφεί ως $w = [w_0, w_1, \ldots, w_n]$, όπου $w_0 = -\theta$ και ονομάζεται πόλωση (bias), ενώ το διάνυσμα εισόδων μπορεί να γραφεί ως $x = [x_0, x_1, ..., x_n]$, με $x_0 = 1$. Πως μπορεί να χρησιμοποιηθεί όμως αυτό το εργαλείο; Το ΤΝΔ αυτό λειτουργεί ως δυαδικός κατηγοριοποιητής, δηλαδή χωρίζει σε δύο κλάσεις την είσοδο η οποία του δόθηκε. Βασική βέβαια προϋπόθεση να είναι τα δεδομένα γραμμικά διαχωρίσιμα.



Σχήμα 2.1: Το δίκτυο Perceptron

Εκπαίδευση

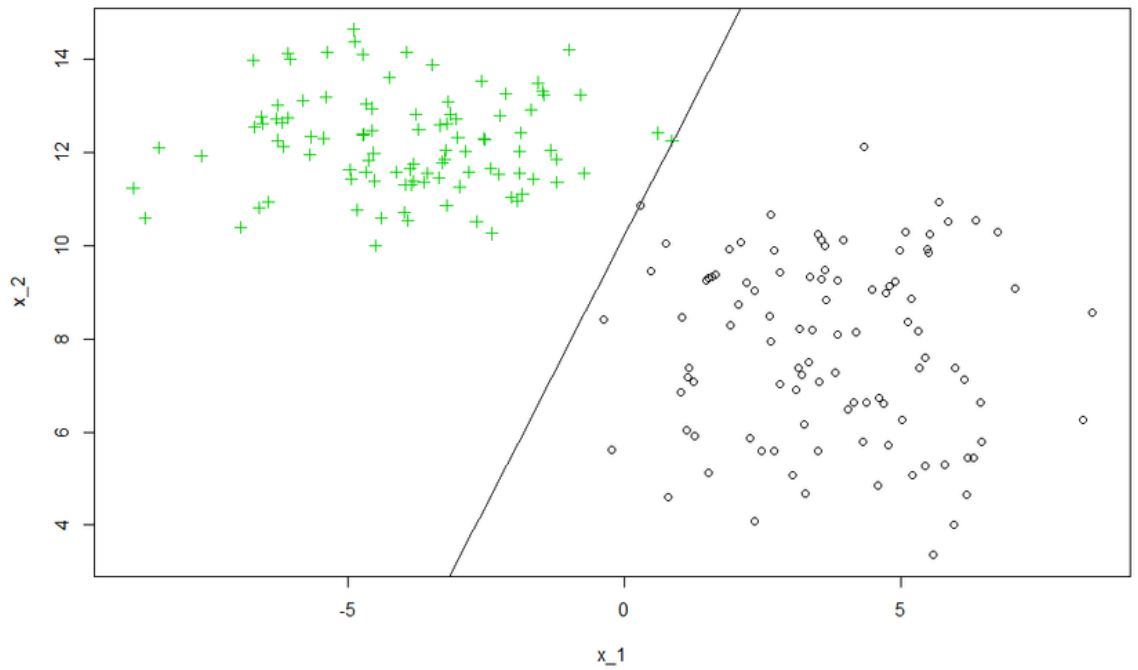
Για να καταφέρει το νευρωνικό δίκτυο να προσαρμόσει τα βάρη του ώστε να κάνει σωστές προβλέψεις χρησιμοποιείται ο κανόνας σταθερής αύξησης. Δηλαδή παρουσιάζονται με κυκλικό τρόπο τα δεδομένα στην είσοδο (πρότυπα) και ανάλογα την έξοδο y για μία δεδομένη είσοδο στο βήμα k τότε τα βάρη ανανεώνονται βάσει της παρακάτω εξίσωσης [11]

$$\mathbf{w}(k) = \mathbf{w}(k-1) + \beta(d^{(p)} - y)\mathbf{x}^{(p)}$$

Το β είναι ένας μικρός θετικός αριθμός ο οποίος ονομάζεται ρυθμός εκπαίδευσης, ενώ το $d^{(p)}$ είναι η κλάση στην οποία ανήκει πραγματικά το δεδομένο πρότυπο. Ο αλγόριθμος εκπαίδευσης που περιεγράφηκε εγγυάται τη σύγκλιση αν τα δεδομένα εισόδου είναι γραμμικά διαχωρίσιμα, όπως απεικονίζεται στο σχήμα 2.2, διαφορετικά η εκπαίδευση του νευρωνικού μπορεί να τερματιστεί όταν [12]:

- Ξεπεραστεί ένας μέγιστος αριθμός επαναλήψεων.
- Δεν υπάρχει κάποια αλλαγή στον αριθμό των σημείων που κατηγοριοποιούνται λανθασμένα.

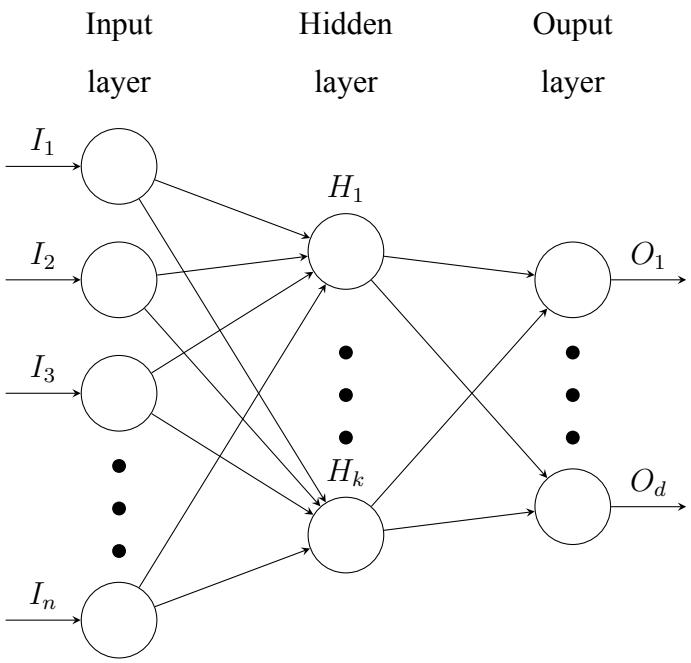
Πως όμως γίνεται η ανάθεση σε κλάσεις; Τα βάρη ορίζουν ένα όριο που ικανοποιεί τη σχέση $\mathbf{w} \cdot x = \theta$, το οποίο είναι κάθετο ως προς το βάρος. Όσα σημεία βρίσκονται στη θετική πλευρά που ορίζεται από το όριο κατηγοριοποιούνται στην κλάση +1, ενώ τα υπόλοιπα στην κλάση -1. Το όριο είναι η γραμμή που διακρίνεται στο σχήμα.



Σχήμα 2.2: Πιθανός διαμερισμός χώρου από ένα δίκτυο Perceptron [1]

2.3 Perceptron πολλών στρωμάτων

Το νευρωνικό δίκτυο το οποίο περιέγραψε ο Rosenblatt είχε κάποιους περιορισμούς. Για παράδειγμα δε μπορούσε να κατηγοριοποιήσει μη γραμμικά διαχωρίσιμα δεδομένα, ενώ αδυνατούσε να περατώσει ακόμα και απλές λειτουργίες όπως η προσομοίωση μίας πύλης XOR (το γνωστό XOR-problem). Αυτούς τους περιορισμούς μπορεί να ξεπεράσει ένα Multi-layer perceptron(MLP), το οποίο αποτελείται από πολλούς νευρώνες. Αυτοί μπορούν να ανήκουν είτε στο στρώμα εισόδου (input layer), το οποίο λαμβάνει την είσοδο και τη μεταβιβάζει με κατάλληλα βάρη στα επόμενα στρώματα, σε κρυφά στρώματα (hidden layers), τα οποία είναι υπεύθυνα για υπολογισμούς, είτε στο στρώμα εξόδου το οποίο παράγει την κατάλληλη έξοδο. Ένα MLP μπορεί να χρησιμοποιηθεί σε πολλές εφαρμογές, όπως η αναγνώριση προτύπων, η κατηγοριοποίηση αλλά και και η προσέγγιση συναρτήσεων [13], ενώ είναι απλό και μπορεί να κλιμακωθεί εύκολα [14]. Στο σχήμα 2.3 απεικονίζεται ένα MLP με n εισόδους, d εξόδους κι ένα hidden layer. Κάθε νευρώνας μπορεί να έχει και bias, όπως ο απλός Perceptron, το οποίο ωστόσο θα μπορούσε να αναπαρασταθεί ως είσοδος του.



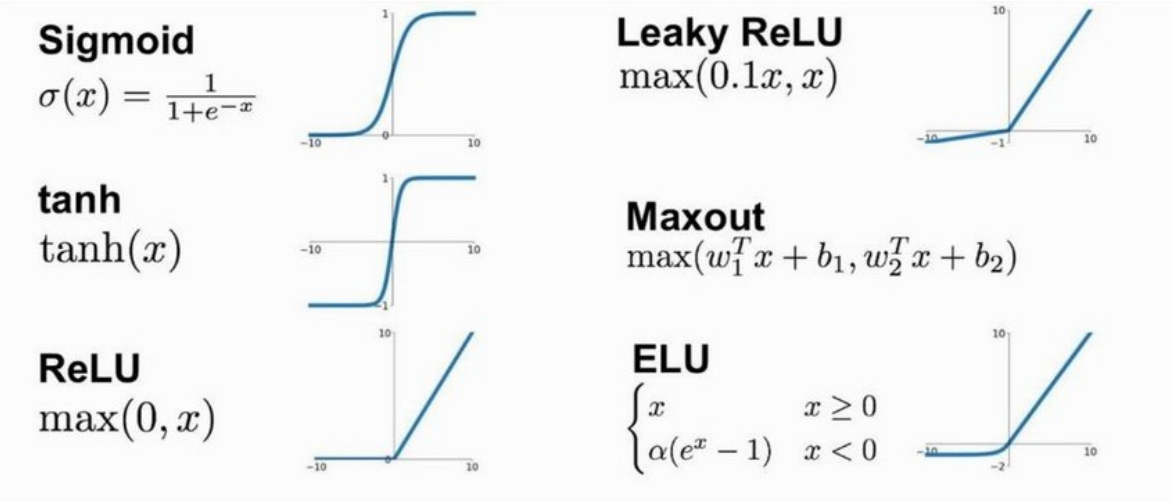
Σχήμα 2.3: MLP με 1 hidden layer

2.3.1 Έξοδος κόμβων

Για κάθε σύνδεση που υπάρχει μεταξύ νευρώνων ορίζεται και ένα βάρος που επηρεάζει το μέγεθος της εξόδου του πρώτου νευρώνα σε αυτήν του δεύτερου. Για κάθε κόμβο σε κάποιο hidden layer ή στο output layer μπορούμε να ορίσουμε το διάνυσμα $\mathbf{w}_k = [w_1, w_2, \ldots, w_n]$, όπου περιλαμβάνει τα βάρη με τις προηγούμενες εισόδους. Η σημειογραφία αυτή θα φανεί χρήσιμη στην περαιτέρω ανάλυση έκαστου κόμβου. Όμως κατά τη μελέτη της εξόδου ενός νευρώνα είναι απαραίτητο να ληφθεί υπόψιν μία ακόμα σημαντική παράμετρος. Αυτή είναι η συνάρτηση ενεργοποίησης (activation function), η οποία επηρεάζει την τελική έξοδο έκαστου νευρώνα και έτσι έχει ενεργό ρόλο στη λειτουργία του. Η επιλογή της μάλιστα επηρεάζει άμεσα τον τρόπο με τον οποίον μαθαίνει το νευρωνικό δίκτυο [15]. Αν αυτή η συνάρτηση συμβολιστεί ως $f(\cdot)$ και οι γραμμές του πίνακα W είναι το διάνυσμα κάθε κόμβου που απαρτίζει το hidden layer τότε η έξοδος ενός νευρώνα k θα είναι $h_k = f(\mathbf{w}_k\mathbf{x} + b_i)$ ή σε μορφή διανύσματος για όλους τους νευρώνες ως

$$\mathbf{h} = f(W\mathbf{x} + \mathbf{b})$$

Για να ολοκληρωθεί η μελέτη για την έξοδο κάθε κόμβου θα γίνει αναφορά σε κάποιες από τις πιο γνωστές συναρτήσεις ενεργοποίησης.



Σχήμα 2.4: Γνωστές συναρτήσεις ενεργοποίησης [2]

Sigmoid

Μία από τις πιο συνηθισμένες συναρτήσεις ενεργοποίησησς είναι η logistic sigmoid, η οποία είναι συνεχής και παραγωγίσιμη και έτσι δε δημιουργεί προβλήματα στην εκτέλεση της εκπαίδευσης του ΤΝΔ που θα αναλυθεί παρακάτω. Ωστόσο βασικό της μειονέκτημα είναι ότι τείνει στο 0, όσο η απόλυτη τιμή της εισόδου είναι μεγαλύτερη του μηδενός. Μπορεί να εκφραστεί από τον τύπο

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

ReLU

Η ReLU είναι παντού παραγωγίσιμη εκτός από το σημείο στο μηδέν. Αποτελεί την πιο επιτυχημένη συνάρτηση και χρησιμοποιείται ευρέως σε αρκετές αρχιτεκτονικές [15]. Η ReLU προσφέρει γρηγορότερη εκπαίδευση εξαιτίας της κλίσης της, ενώ η παράγωγος της υπολογίζεται ευκολότερα με μαθηματικές ιδιότητες. Ορίζεται από τον τύπο

$$f(x) = max(0, x)$$

Υπάρχουν αρκετές συναρτήσεις ενεργοποίησης ακόμα οι οποίες δε θα αναφερθούν σε αυτήν τη διπλωματική και εφαρμόζονται σε διαφορετικούς τύπους προβλημάτων. Σε κάθε περίπτωση η χρήση τους επηρεάζει άμεσα την εκπαίδευση και την απόδοση κάθε αρχιτεκτονικής, ενώ η επιλογή της κατάλληλης activation function είναι κρίσιμη και απαιτεί εμπειρία αλλά και χρόνο. Στο σχήμα 2.4 διακρίνονται κάποιες από τις πιο γνωστές.

2.3.2 Εκπαίδευση MLP

Στην υποενότητα 2.2.2 παρουσιάστηκε ο τρόπος με τον οποίον εκπαιδεύεται το δίκτυο Perceptron ώστε να δύναται να ανταποκριθεί με τον επιθυμητό τρόπο στα δεδομένα τα οποία λαμβάνει. Στο σημείο αυτό θα αναλύσουμε τον τρόπο με τον οποίον ένα MLP προσαρμόζει τα βάρη του, δηλαδή μαθαίνει. Ο αλγόριθμος Backpropagation (BP) χρησιμοποιείται στην εκπαίδευση Perceptron πολλών στρωμάτων εξαιτίας του μαθηματικού του υποβάθρου. Πριν όμως από αυτό θα πρέπει να ορίσουμε μία μετρική η οποιά θα αντιπροσωπεύει την απόδοση του ΤΝΔ. Για αυτό το λόγο ορίζεται το μέσο τετραγωνικό σφάλμα (MSE), το οποίο ορίζεται ως [12]

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (\mathbf{y}_i - \tilde{\mathbf{y}}_i)^2$$

Στην παραπάνω φόρμουλα ως n ορίζεται ο αριθμός σημειών για την εκπαίδευση του ΤΝΔ, ως $\mathbf{y}_i$ ορίζεται η έξοδος του νευρωνικού δικτύου για είσοδο x_i και ως $\tilde{\mathbf{y}}_i$ η επιθυμητή έξοδος για αυτή την είσοδο. Βάσει του MSE μπορούμε να ξεκινήσουμε την ανάλυση του BP ο οποίος χωρίζεται σε δύο μέρη: Το forward pass και το backward pass. Έστω λοιπόν ένα MLP με τα παρακάτω χαρακτηριστικά:

- L layers
- n εισόδους
- m εξόδους

Στόχος του αλγορίθμου εκπαίδευσης είναι να μειώσει το σφάλμα σε έναν πεπερασμένο αριθμό επαναλήψεων [16]. Ο τρόπος με τον οποίον θα επιτευχθεί αυτό είναι με την αναβάθμιση των διανυσμάτων των βαρών, τα οποία άλλωστε έχουν άμεση επίδραση στην έξοδο.

Ο αλγόριθμος

Έστω ένας νευρώνας i. Αν αυτός ο νευρώνας έχει μία συνάρτηση ενεργοποίησης $f_i(\cdot)$ και συνδέεται με k νευρώνες με βάρη $w_{i,0}, w_{i,1}, \ldots$ από το προηγούμενο layer τότε η έξοδος του ορίζεται ως

$$o_i = f_i(\sum_{j=0}^{k} (w_{i,j} \cdot o_j))$$

Με αυτόν τον τρόπο υπολογίζεται η έξοδος σε κάθε νευρώνα του νευρωνικού μέχρι το layer εξόδου. Αφού υπολογιστεί και η τελική έξοδος του νευρωνικού είμαστε σε θέση να ξεκινήσουμε την επόμενη φάση του αλγορίθμου. Στόχος είναι η διόρθωση των βαρών με τέτοιο τρόπο ώστε να ελαχιστοποιείται το τετραγωνικό σφάλμα $E(n)$ στο n-οστό βήμα. Η διόρθωση των βαρών θα γίνει χρησιμοποιώντας τον κανόνα δέλτα

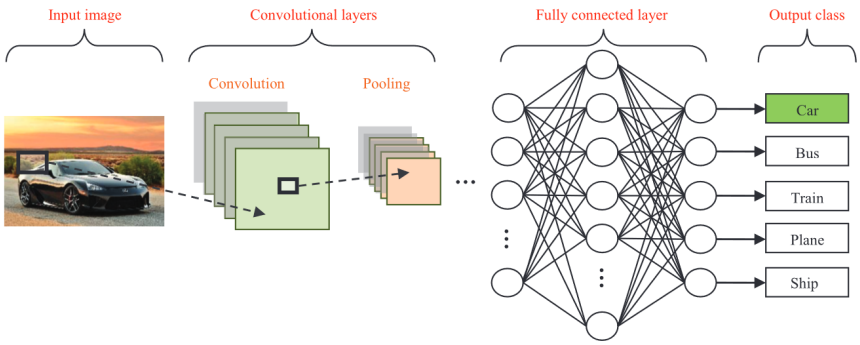
$$\Delta w_{i,j} = -\eta \frac{\partial E(n)}{\partial w_{i,j}(n)}$$

Για να υπολογιστεί η παράγωγος στο δεξί μέρος της εξίσωσης θα χρησιμοποιηθεί ο κανόνας αλυσιδωτής παραγώγισης σύμφωνα με τον οποίον είναι δυνατόν να εκφραστεί η παραπάνω παράγωγος ως γινόμενο διαφορετικών παραγώγων. Εφαρμόζοντας αυτούς τους κανόνες αν l είναι κάποιο στρώμα και $w_{i,j}(l,k)$ είναι κάποιο βάρος σε εκείνο την k-οστή επανάληψη υπολογίζουμε [10]

$$\begin{aligned}
\delta_i(L) &= o_i'(L) \cdot \textstyle\sum_k (\tilde{y}_i - y_i) \\
\delta_i(l) &= o_i'(l) \textstyle\sum_{\mu=1}^{N(l+1)} w_\mu(l+1)\delta_\mu(l+1) \\
w_{i,j}(l,k+1) &= w_{i,j}(l,k) + \eta\delta_i(l)o_j(l-1)
\end{aligned}$$

Αφού ορίσαμε τους παραπάνω τύπους καθίσταται πλέον εφικτό να υπολογιστούν οι έξοδοι σε κάθε layer $l = 1, 2, \ldots, L$. Σημειώνεται ότι τα βάρη έχουν αρχικοποιηθεί με τυχαίες τιμές πριν ξεκινήσει αυτή η διαδικασία. Μόλις ολοκληρωθεί η παραπάνω διαδικασία υπολογίζονται τα σφάλματα δ_i ξεκινώντας από το τελευταίο στρώμα μέχρι το πρώτο. Τέλος υπολογίζονται τα νέα βάρη εφόσον έχουν υπολογιστεί οι παράμετροι δ_i. Σημειώνεται ότι η παράμετρος η είναι ένας μικρός θετικός αριθμός τον οποίον ονομάζουμε ρυθμό εκμάθησης (learning rate). Ο αλγόριθμος σταματάει την εκτέλεση του στις παρακάτω περιπτώσεις:

- Δεν υπάρχει σημαντική μείωση του σφάλματος σε δύο συνεχόμενες επαναλήψεις (οι οποίες ονομάζονται εποχές).
- Το σφάλμα σε μία εποχή είναι μικρότερο από μία προκαθορισμένη τιμή ϵ.
- Ένας αριθμός μέγιστων επαναλήψεων (μέγιστες εποχές) έχει συμπληρωθεί, χωρίς να πληρείται κάποιο από τα παραπάνω κριτήρια.



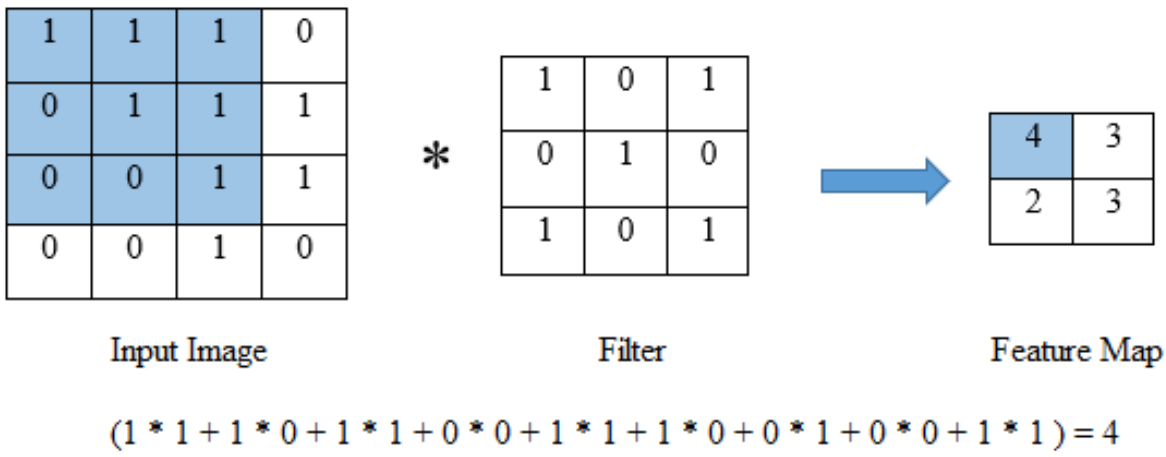
Σχήμα 2.5: Η αρχιτεκτονική ενός CNN [3]

Εκπαίδευση σε batches

Ο αλγόριθμος που παρουσιάστηκε προηγουμένως προσεγγίζει τη μέθοδο κατάβασης δυναμικού (gradient descent). Ωστόσο είναι αλήθεια ότι η εκτέλεση του αποτελεί μία ακριβή υπολογιστικά εργασία, οπότε είναι επιθυμητό να επιταχυνθεί η διαδικασία αυτή. Ένας από τους τρόπους να γίνει αυτό είναι η εκπαίδευση σε batches. Όταν το νευρωνικό δίκτυο εκπαιδεύεται με αυτόν τον τρόπο υπάρχει μία ομαδοποίηση δεδομένων και αυτά παρουσιάζονται όλα μαζί σε μία εποχή, μετά το πέρας της οποίας γίνεται η ανανέωση των βαρών [17].

2.4 Συνελικτικά Νευρωνικά Δίκτυα

Όπως προαναφέρθηκε, τα MLP χρησιμοποιούνται ευρέως σε εφαρμογές των ΤΝΔ, ωστόσο η συγκεκριμένη αρχιτεκτονική δεν είναι η μοναδική που έχει δημιουργηθεί. Τα συνελικτικά νευρωνικά δίκτυα (Convolutional Neural Networks/CNN) αποτελούν μία ακόμα διάσημη αρχιτεκτονική ΤΝΔ. Τα CNN εφαρμόζονται συνήθως σε προβλήματα κατηγοριοποίησης εικόνας, στην όραση υπολογιστών (computer vision) αλλά και στην επεξεργασία φυσικής γλώσσας (Natural Language Processing/NLP), εξαιτίας της υψηλής ακρίβειας που μπορούν να επιτύχουν [18]. Αξίζει να σημειωθεί ότι τα CNN οφείλουν το όνομα τους στη μαθηματική πράξη της συνέλιξης (convolution), η οποιά ωστόσο δε θα αναλυθεί. Στο σχήμα 2.5 διακρίνεται ένα παράδειγμα αρχιτεκτονικής συνελικτικών νευρωνικών δικτύων, η οποία θα μπορούσε να χρησιμοποιηθεί για την κατηγοριοποίηση εικόνων.



Σχήμα 2.6: Υπολογισμός συνέλιξης φίλτρου $3x3$ και $stride = 1$ [4]

2.4.1 Convolutional layers

Το πρώτο είδος στρώματος το οποίο συναντάται σε ένα CNN είναι το convolutional. Αποτελείται από έναν αριθμό πινάκων, τους οποίους ονομάζουμε φίλτρα ή kernels, καθένας από τους οποίους απαρτίζεται από ένα σύνολο βαρών και υπολογίζει τη συνέλιξη $W_k * x$, όπου x είναι η είσοδος του στρώματος και W_k ένας πίνακας που περιγράφει το φίλτρο k. Αν $f(\cdot)$ είναι η συνάρτηση ενεργοποίησης τότε η τελική έξοδος για ένα φίλτρο ορίζεται ως [3]

$$Y_k = f(W_k * x)$$

Για να γίνει κατανοητή η πράξη της συνέλιξης θα πρέπει να εισαχθεί ακόμα η έννοια του stride. Όταν γίνεται αυτή η πράξη ο πίνακας που αντιπροσωπεύει το φίλτρο μετακινείται από αριστερα προς τα δεξιά κι έπειτα από πάνω προς τα κάτω και υπολογίζει τα στοιχεία του πίνακα εξόδου, με τον τρόπο που διακρίνεται στο σχήμα 2.6. Το $stride$ επηρεάζει τον τρόπο με τον οποίον γίνεται η μετακίνηση του πίνακα του φίλτρου στην αρχική είσοδο. Για παράδειγμα για $stride = 1$ γίνεται μετακίνηση στην αρχική είσοδο για ένα pixel μέχρι να έχουν καλυφθεί οι αντίστοιχες , για $stride = 2$ κατά 2 κ.ο.κ.

2.4.2 Pooling layers

Μετά από ένα convolutional layer ακολουθεί συνήθως ένα στρώμα pooling, του οποίου σκοπός είναι να μειώσει τις διαστάσεις της εξόδου του προηγούμενου στρώματος. Για να το κάνει αυτό χωρίζει την είσοδο που έλαβε σε μικρότερους πίνακες και στη συνέχεια εκτελεί κάποια πράξη συνένωσης των δεδομένων. Πιθανές τέτοιες πράξεις είναι το μέγιστο μεταξύ

των στοιχείων κάθε περιοχής και αποθηκεύεται σε μία θέση του νέου πίνακα.

2.4.3 Fully connected layer

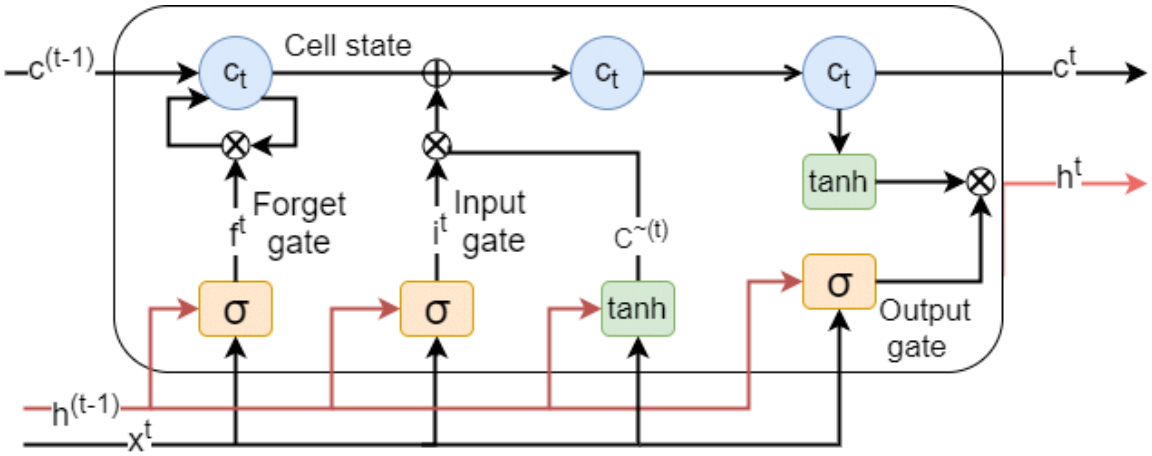
Τελευταίο είδος στρώματος στην αρχιτεκτονική ενός συνελικτικού δικτύου είναι το πλήρως συνδεδεμένο, το οποίο αποτελείται από ένα σύνολο νευρώνων, οι οποίοι είναι συνδεδεμένοι με βάρη ώστε να λαμβάνουν την προηγούμενη είσοδο. Για να είναι εφικτή η σύνδεση μεταξύ τους θα πρέπει να έχει προηγηθεί η επιπεδοποίηση της εξόδο του pooling layer (Flatten). Με αυτόν τον τρόπο τα δεδομένα, που ήταν αποθηκευμένα σε μορφή πίνακα προηγουμένως, είναι πλέον αποθηκευμένα σε μορφή διανύσματος. Για παράδειγμα το στοιχείο της θέσης (0, 0) του πίνακα αποθηκεύεται στη θέση 0 του διανύσματος, της θέσης (0, 1) στην 1 κ.ο.κ. Το πλήρως συνδεδεμένο στρώμα έχει τόσες εξόδους όσες οι επιθυμητές κλάσεις.

2.4.4 Εκπαίδευση συνελικτικών νευρωνικών δικτύων

Η μάθηση σε CNN αποτελεί ένα πολύ σημαντικό στοιχείο της θεωρίας τους, που ωστόσο δε θα αναλυθεί εκτενώς σε αυτή τη διπλωματική. Όπως και τα MLP έτσι και τα συνελικτικά νευρωνικά δίκτυα βασίζονται στον αλγόριθμο backpropagation. Ωστόσο στη δεύτερη αρχιτεκτονική αντί να εκπαιδεύονται βάρη μεταξύ νευρώνων, εκπαιδεύονται τα βάρη των φίλτρων. Τα φίλτρα που βρίσκονται στα πρώτα στρώματα συνήθως ανιχνεύουν απλά μοτίβα, ενώ αυτά που βρίσκονται σε βαθύτερα ανιχνεύουν πιο σύνθετα μοτίβα. Οι συνδέσεις αυτές προσθέτουν σε ένα απλό ΤΝΔ την ικανότητα αποθήκευσης προηγούμενων καταστάσεων, ωστόσο δημιουργούν και προβλήματα κατά την εκπαίδευση. Για αυτό το λόγο έχουν αναπτυχθεί καινούργιες αρχιτεκτονικές, όπως τα δίκτυα Long Short Term Memory/LSTM, τα οποία χρησιμοποιούν μία καινούργια μονάδα που ονομάζεται κελί και επιτρέπει στο δίκτυο να διατηρεί αρχείο των καταστάσεων εξόδου του παρελθόντος.

2.5 Αναδρομικά νευρωνικά δίκτυα

Αν και οι αρχιτεκτονικές που αναφέρθηκαν προηγουμένως χρησιμοποιούνται σε μεγάλο βαθμό μέχρι και σήμερα ωστόσο υπάρχει ακόμα ένας τύπος νευρωνικών δικτύων που βρίσκει εφαρμογή σε πληθώρα εφαρμογών. Αυτά είναι τα αναδρομικά νευρωνικά δίκτυα (recurrent neural networks/RNN) τα οποία έχουν τη δυνατότητα να εντοπίζουν μη γραμμικές σχέσεις μεταξύ γνωρισμάτων τα οποία σχετίζονται χρονικά μεταξύ τους [19]. Η μεγάλη καινοτομία



Σχήμα 2.7: Η αρχιτεκτονική ενός κελιού LSTM [5]

σε αυτά είναι ότι απομνημονεύουν προηγούμενες καταστάσεις, τις οποίες και χρησιμοποιούν στη συνέχεια για να προσαρμόσουν την έξοδο τους, με τη μνήμη τους να επιτυγχάνεται με τη δημιουργία κύκλων από συνδέσεις μεταξύ μονάδων [20].

2.5.1 LSTM

Τα αναδρομικά νευρωνικά δίκτυα είναι ένας πολύ σημαντικός τύπος αρχιτεκτονικής, ωστόσο αντιμετωπίζουν προβλήματα κατά την εκπαίδευση τους εξαιτίας της φύσης των κλίσεων τους. Έτσι δε δύνανται να μοντελοποιήσουν σχέσεις μεταξύ εισόδων και εξόδων για πάνω από 10 διακριτά βήματα [21] και ως αποτέλεσμα έχουν προταθεί νέες αρχιτεκτονικές αναδρομικών νευρωνικών δικτύων. Ένας τύπος αυτών είναι τα Long Short Term Memory/LSTM τα οποία αποτελούνται από μονάδες μνήμης που ονομάζονται κελιά, των οποίων οι είσοδοι και έξοδοι ελέγχονται από πύλες [22]. Η συνδεσμολογία μίας τέτοιας μονάδας διακρίνεται στο σχήμα 2.7 και αποτελείται από την πύλη εισόδου, την πύλη εξόδου και την πύλη απώλειας μνήμης (forget gate).

Πύλη εισόδου

Η πύλη εισόδου χρησιμοποιείται για να ληφθεί η απόφαση ποιες πληροφορίες θα διατηρηθούν από τις προηγούμενες καταστάσεις [23]. Η έξοδος της i_t ποσοτικοποιεί τη σημαντικότητα της νέας πληροφορίας μεταξύ 0 και 1, με την πρώτη τιμή να ορίζεται για τις τιμές όπου η νέα πληροφορία είναι ασήμαντη ενώ η τελευταία πολύ σημαντική. Η συγκεκριμένη μεταβλητή ορίζεται από την εξίσωση 2.1, όπου x^t είναι η τρέχουσα είσοδος, U_i ο πίνακας βαρών

που σχετίζεται με αυτήν, $h^{(t-1)}$ είναι η κατάσταση της προηγούμενης χρονικής στιγμής και W_i ένας πίνακας βαρών που συνδέεται με αυτήν. Ως $\sigma(\cdot)$ ορίζεται η σιγμοειδής συνάρτηση.

$$i^t = \sigma(x^t \cdot U_i + h^{(t-1)} \cdot W_i) \tag{2.1}$$

Πύλη απώλειας μνήμης

Η πύλη αυτή είναι υπεύθυνη για τον προσδιορισμό εκείνων των καταστάσεων οι οποίες δεν είναι απαραίτητες να διατηρηθούν την επόμενη χρονική στιγμή οπότε και διαγράφονται. Αν U_f και W_f είναι δύο πίνακες βαρών, οι οποίοι σχετίζονται με την είσοδο και την προηγούμενη κατάσταση τότε η έξοδος της πύλης απώλειας μνήμης είναι

$$f^t = \sigma(x^t \cdot U_f + h^{(t-1)} \cdot W_f) \tag{2.2}$$

Πύλη εξόδου

Η τελευταία πύλη η οποία απαρτίζει ένα LSTM κελί είναι αυτή η οποία θα προσδιορίσει και την έξοδο του. Η έξοδος της ίδιας είναι o^t και ορίζεται από την εξίσωση 2.3.

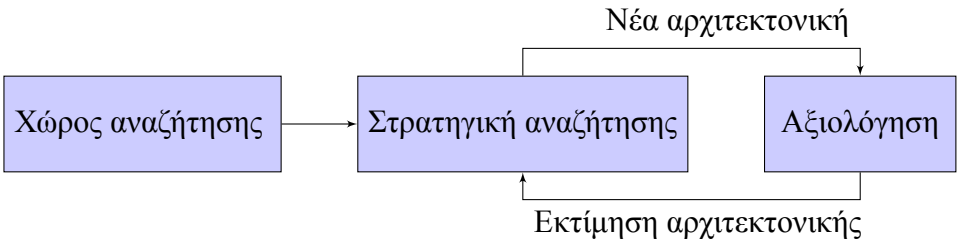
$$o^t = \sigma(x^t \cdot U_o + h^{(t-1)} \cdot W_o) \tag{2.3}$$

Για τον υπολογισμό της τελικής εξόδου του κελιού πρέπει να οριστεί μία ακόμα μεταβλητή. Αυτή είναι η $N^t = tanh(x^t \cdot U_c + h^{(t-1)} \cdot W_c)$ που ορίζει τη νέα πληροφορία του κελιού. Μέσω αυτής δύναται να υπολογιστεί η νέα κατάσταση $C^t = f^t \cdot C^{(t-1)} + i^t \cdot N^t$. Τελικά μπορεί να βρεθεί και η συνολική έξοδος του κελιού η οποία ορίζεται από την εξίσωση 2.4.

$$out = softmax(H^t) = softmax(o^t \cdot tanh(C^t)) \tag{2.4}$$

2.6 Προβλήματα στην εκπαίδευση νευρωνικών δικτύων

Ο αλγόριθμος BP αποτελεί ένα εξαιρετικό εργαλείο για την ανανέωση των βαρών ενός ΤΝΔ, ώστε να ελαχιστοποιείται η συνάρτηση απώλειας. Ωστόσο όταν αυτό εκπαιδεύεται είναι πιθανό τα βάρη του να προσαρμοστούν σε τέτοιο βαθμό στα δεδομένα που του εισάγονται ώστε να αδυνατούν να εκτελέσουν σωστά οι επιθυμητές λειτουργίες. Αυτό ονομά-



Σχήμα 2.8: Τα στάδια αναζήτησης αρχιτεκτονικής

ζεται overfitting και έχουν προταθεί διάφορες λύσεις για να το αντιμετωπίσουν, με μία από τις βασικότερες να είναι το dropout. Όταν εφαρμόζεται αυτή η τεχνική κάποιοι νευρώνες αφαιρούνται κατά τη διάρκεια μίας εποχής από την αρχιτεκτονική που έχει επιλεγεί και συνεχίζεται η εκπαίδευση του ΤΝΔ χωρίς αυτούς. Η επιλογή της διατήρησης ενός νευρώνα μία χρονική στιγμή στο δίκτυο γίνεται με τυχαίο τρόπο και καθορίζεται από μία πιθανότητα p. Συνήθως επιλέγεται μία τιμή κοντά στο 1 για τα στρώματα εισόδου [24]. Ένα ακόμα πρόβλημα που καλούμαστε να αντιμετωπίσουμε κατά την εκπαίδευση ενός νευρωνικού δικτύου είναι η απόδοση: Ο αλγόριθμος που χρησιμοποιείται εκτελεί πολλές πράξεις και έτσι καθίσταται απαγορευτική η χρήση του για πολλά δεδομένα εισόδου ή για βαθιές αρχιτεκτονικές, οι οποίες διαθέτουν μεγάλο αριθμό στρωμάτων.

2.7 Αναζήτηση αρχιτεκτονικής νευρωνικών δικτύων

Όλα τα παραπάνω ΤΝΔ αποτελούν πολύ χρήσιμα εργαλεία και συναντώνται σε πληθώρα εφαρμογών. Ωστόσο η επιλογή της αρχιτεκτονικής για κάθε πρόβλημα αποτελεί μία δύσκολη συνήθως διαδικασία, που χρειάζεται χρόνο αλλά και εμπειρία του άτομο, το οποίο θα κληθεί να τη σχεδιάσει. Για αυτό τον λόγο υπάρχει μεγάλο ενδιαφέρον στην αυτοματοποίηση της διαδικασίας αυτής, η χρήση δηλαδη αλγορίθμου ο οποίος θα διαλέγει χωρίς ανθρώπινη παρέμβαση την αρχιτεκτονική του ΤΝΔ που θα χρησιμοποιηθεί για την επίλυση ενός δεδομένου προβλήματος. Το συγκεκριμένο πεδίο έρευνας είναι υποκατηγορία του AutoML, που αποτελεί υποσύνολο της μηχανικής μάθησης και ασχολείται με την αυτοματοποίηση συγκεκριμένων πεδίων του [25]. Οι αλγόριθμοι που χρησιμοποιούνται στην αναζήτηση της αρχιτεκτονικής χωρίζονται σε τρία μέρη, τα οποία διακρίνονται στο σχήμα 2.8.

2.7.1 Χώρος αναζήτησης

Σε πρώτη φάση είναι αναγκαίο να καθοριστεί τι είδους στρώματα θα εισαχθούν στην αρχιτεκτονική. Ο χώρος αναζήτησης προσδιορίζει τα είδη των στρωμάτων που θα χρησιμοποιηθούν στις αρχιτεκτονικές που θα κατασκευαστούν άλλα και τη μεταξύ τους συνδεσμολογία, παράμετροι οι οποίοι συνήθως προσδιορίζονται από τον χρήστη. Ως αποτέλεσμα ο χώρος αυτός είναι πιθανό να εξαρτάται από τον χρήστη του αλγορίθμου, οπότε και η τελική αρχιτεκτονική δεν παρέχει καλές λύσεις [26].

2.7.2 Στρατηγική αναζήτησης

Για να δημιουργηθεί το μοντέλο πρέπει να βρεθεί ένας τρόπος προσθήκης των στρωμάτων. Η στρατηγική αναζήτησης αποτελείται από ένα σύνολο κανόνων, σύμφωνα με τους οποίους γίνεται η πλοήγηση στον χώρο αναζήτησης.

2.7.3 Αξιολόγηση

Το τελευταίο στάδιο ενός αλγορίθμου αναζήτησης αρχιτεκτονικής νευρωνικών δικτύων περιλαμβάνει την εκτίμηση του μοντέλου το οποίο σχεδιάστηκε. Μία αρκετά απλή λύση είναι η δημιουργία μοντέλου και στη συνέχεια η εκπαίδευση του με έλεγχο των επιδόσεων του στο τέλος, αν και προτιμάται να αποφεύγεται καθώς είναι πολύ απαιτητική υπολογιστικά [26]. Αφού ολοκληρωθεί ο έλεγχος των μετρικών του μοντέλου, δημιουργείται μία νέα αρχιτεκτονική η οποία υποβάλλεται για άλλη μία φορά σε αξιολόγηση. Είναι επιθυμητό αυτές οι αρχιτεκτονικές που παράγονται να βελτιώνονται σε διαδοχικές επαναλήψεις, ώστε να βελτιώνονται συνεχώς τα ΤΝΔ που προκύπτουν.

Κεφάλαιο 3

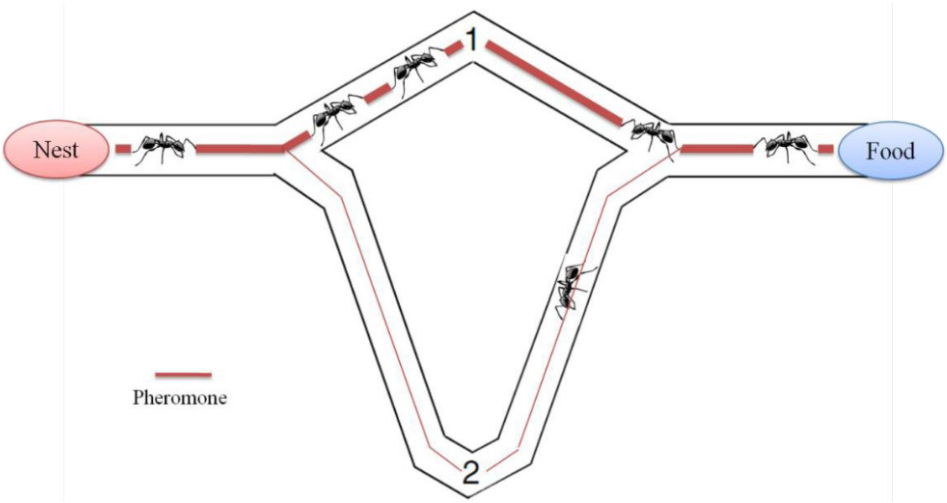
Ευφυΐα σμήνους

3.1 Εισαγωγή

Η φύση αποτελούσε πάντα έμπνευση για τον άνθρωπο, ο οποίος τείνει να μιμηθεί συμπεριφορές και πρότυπα που εμφανίζονται σε αυτή, ώστε να τα εφαρμόσει για την επίλυση δικών του προβλημάτων. Στο κεφάλαιο 2 έγινε αναφορά στα ΤΝΔ, τα οποία οφείλουν την ύπαρξη τους σε βιολογικά συστήματα, όπως ο ανθρώπινος εγκέφαλος, ωστόσο αυτά δεν αποτελούν τις μόνες τεχνικές που εφαρμόζονται σε υπολογιστή και έχουν βιολογικό υπόβαθρο. Η ευφυΐα σμήνους (Swarm Intelligence/SI) είναι ένα σύνολο τεχνικών, που οφείλουν την ύπαρξη τους σε συμπεριφορές πληθυσμών ζώων στο φυσικό τους περιβάλλον. Όπως θα δούμε αυτή αποτελείται από οικογένειες αλγορίθμων, οι οποίες χρησιμοποιούν ένα σύνολο από πράκτορες, οι οποίοι μπορούν να εκτελούν απλές διαδικασίες μόνοι τους αλλά όταν εργάζονται με τους υπόλοιπους πράκτορες (σμήνος) είναι σε θέση να περατώσουν δύσκολα προβλήματα.

3.2 Ant Colony Optimization

Η πρώτη τεχνική στην οποία θα αναφερθούμε είναι μία μεταευρετική που ονομάζεται Ant Colony Optimization(ACO) και αναπτύχθηκε από τον Marco Dorigo και συνεργάτες του [28] για να λύσει το πρόβλημα του πλανόδιου πωλητή (Travelling Salesman Problem/TSP). Πριν γίνει όμως η περιγραφή της πρέπει να κάνουμε ένα βήμα πίσω και να εξετάσουμε την πηγή έμπνευσης του συγκεκριμένου αλγορίθμου: Τη συμπεριφορά των μυρμηγκιών στη φύση, όταν αυτά αναζητούν τροφή.



Σχήμα 3.1: Αναζήτηση τροφής από τα μυρμήγκια [6]

3.2.1 Η βιολογία

Τα μυρμήγκια οργανώνονται σε αποικίες, οι οποίες αποτελούνται από ένα σύνολο οργανισμών, οι οποίοι είναι να περατώσουν απλές εργασίες και επικοινωνούν έμμεσα μεταξύ τους, αφήνοντας πίσω τους μία ουσία που ονομάζεται φερομόνη. Αν λοιπόν υπάρχουν δύο μονοπάτια από τη φωλιά, τα οποία οδηγούν στο σημείο που βρίσκεται η τροφή, τα μυρμήγκια διαλέγουν τυχαία έναν από τους δύο δρόμους και ξεκινούν τη διαδρομή που θα τα οδηγήσει στην επιθυμητή θέση. Ο πληθυσμός που θα διαλέξει το συντομότερο μονοπάτι θα γυρίσει και νωρίτερα στο σημείο εκκίνησης, οπότε και θα έχει αφήσει μεγαλύτερη ποσότητα φερομόνης, την οποία μπορούν στη συνέχεια να εντοπίσουν τα υπόλοιπα μέλη του σμήνους και να ακολουθήσουν εκείνο το μονοπάτι. Η συμπεριφορά αυτή διακρίνεται στο σχήμα 3.1. Μία τελευταία παράμετρος που πρέπει να ληφθεί υπόψιν είναι η εξάτμιση της φερορμόνης, η οποία αποτρέπει τη συγκεκριμένη ουσία να παραμείνει για μεγάλο χρονικό διάστημα σε μία διαδρομή και έτσι να αποτραπεί η επιλογή κακών διαδρομών που έτυχε να επιλεγούν στο παρελθόν. Με αυτόν τον τρόπο το σμήνος καταφέρνει να συγκλίνει στην επιλογή της λύσης που βελτιστοποιεί το πρόβλημα του.

3.2.2 Η ευρετική

Η τεχνική ACO μιμείται τη συμπεριφορά των μυρμηγκιών στη φύση και ορίζει ένα σύνολο πρακτόρων (μυρμήγκια) που αναζητούν μία λύση για ένα δεδομένο πρόβλημα. Στη

γενικότερη του μορφή ένας τέτοιος αλγόριθμος ορίζεται από τρεις μεταβλητές [29].

Σύνολο αναζήτησης

Οι πράκτορες-μυρμήγκια αναζητούν πιθανές διακριτές λύσεις X_i, από τις οποίες θα προκύψει η βέλτιστη. Το σύνολο που τις περιλαμβάνει όλες ονομάζεται σύνολο αναζήτησης και συμβολίζεται με S.

Περιορισμοί

Μία άλλη σημαντική παράμετρος, η οποία ορίζεται στο ACO είναι το σύνολο περιορισμών Ω. Το σύνολο αυτό περιλαμβάνει όλους τους κανόνες τους οποίους μπορούν να ακολουθήσουν οι πράκτορες που έχουν οριστεί για να αναζητήσουν μία λύση s από το σύνολο αναζήτησης. Για παράδειγμα στους αλγορίθμους που θα μελετηθούν στα επόμενα κεφάλαια υπάρχουν κανόνες που ορίζουν τη μετάβαση των πρακτόρων από μία κορυφή i σε μία κορυφή j ενός γραφήματος, υπολογίζοντας μία πιθανότητα μετάβασης που εξαρτάται από μία ευρετική συνάρτηση $\eta_{i,j}$ και την τιμή της φερομόνης $\tau_{i,j}$ για το συγκεκριμένο μονοπάτι, η οποία ανήκει σε ένα σύνολο T_i^j. Ένας άλλος κανόνας που απαρτίζει αυτό το σύνολο είναι αυτός της ανανέωσης των φερομονών με το πέρασμα των επαναλήψεων της εκτέλεσης.

Συνάρτηση στόχου

Τελευταία μεταβλητή η οποία είναι απαραίτητη για τον ορισμό του αλγορίθμου είναι η συνάρτηση στόχου f. Οι πράκτορες καλούνται να κατασκευάσουν λύσεις $s \in S$ οι οποίες ελαχιστοποιούν (ή μεγιστοποιούν) αυτή τη συνάρτηση.
Χρησιμοποιώντας τις παραπάνω παραμέτρους ξεκινάει η λειτουργία του αλγορίθμου, η οποία απαρτίζεται από τα εξής βήματα

- Αρχικοποίηση τιμών φερομονών: Ανάθεση δηλαδή αρχικών τιμών $c > 0$ για τις φερομόνες.

- Κατασκευή λύσης: Εκκίνηση της λειτουργίας των πρακτόρων, οι οποίοι καλούνται να αναζητήσουν πιθανές λύσεις του συνόλου αναζήτησης χρησιμοποιώντας τις τιμές των φερομονών αλλά και το σύνολο περιορισμών.

- Τοπική αναζήτηση: Αυτό το στάδιο δεν είναι απαραίτητο και χρησιμοποιείται για την περαιτέρω βελτιστοποίηση των λύσεων που έχουν κατασκευαστεί από τους πράκτορες.

- Ανανέωση τιμών φερομονών με τη χρήση των κανόνων που έχουν οριστεί.

3.3 Particle Swarm Optimization

Οι πληθυσμοί μυρμηγκιών δεν είναι οι μόνοι οι οποίοι έχουν μελετηθεί διεξοδικά από τους βιολόγους. Σμήνη πουλιών, ψαριών αλλά κ.α. αποτελούν πηγή έμπνευσης για την ανάπτυξη αλγορίθμων βελτιστοποίησης με τους Kennedy και Eberhart [30] να είναι οι πρώτοι οι οποίοι να την εισάγουν.

3.3.1 Η βιολογία

Στη φύση διάφοροι οργανισμοί συνεργάζονται για να πετύχουν το μέγιστο δυνατό όφελος. Για παράδειγμα τα ψάρια και τα πτηνά προσαρμόζουν τις κινήσεις τους σύμφωνα με τη συμπεριφορά των υπόλοιπων οργανισμών του σμήνους για να βρουν τροφή και σύντροφο ή να αποφύγουν τους θυρευτές τους. Οι άνθρωποι έχοντας πιο ανεπτυγμένο εγκέφαλο δεν προσαρμόζουν μόνο την κίνηση τους στον χώρο αλλά και τη συμπεριφορά τους, ώστε να μην αποκλίνει από αυτή των υπολοίπων και έτσι να βελτιστοποιεί το προσωπικό του όφελος. Έχει διαπιστωθεί λοιπόν ότι η θέση των παραπάνω οργανισμών δεν επηρεάζεται αποκλειστικά από την προσωπική αντίληψη που έχουν αλλά και από αυτή των υπολοίπων μελών του σμήνους ή της κοινωνίας που ανήκουν και έτσι είναι σε θέση να βελτιστοποιήσουν την κατάσταση τους.

3.3.2 Η ευρετική

Οι συμπεριφορές που περιεγράφηκαν προηγουμένως έχουν οδηγήσει στην ανάπτυξη του αλγορίθμου βελτιστοποίησης πλήθους σωματιδίων (Particle Swarm Optimization/PSO) για την επίλυση προβλημάτων από υπολογιστή. Σε αυτήν την ευρετική βασική μονάδα αποτελεί το σωματίδιο το οποίο περιγράφεται από τη θέση του $x_{i,j}$ στον n-διάστατο χώρο και την ταχύτητα του $V_{i,j}$, η οποία έχει μέγιστη τιμή V_{max}. Στον PSO τα σωματίδια αντιπροσωπεύουν λύσεις σε ένα δεδομένο πρόβλημα [31]. Όπως συμβαίνει και με τα σμήνη τα οποία συναν-

τώνται στη φύση έτσι και στον PSO ένα σωματίδιο επηρεάζεται από τη συμπεριφορά των υπόλοιπων μελών και ανανεώνει τη θέση του στον χώρο τη χρονική στιγμή $t + 1$ σύμφωνα με τις εξισώσεις

$$V_{i,j}(t+1) = wV_i(t) + c_1 r_1(p_{best} - x_{i,j}(t)) + c_2 r_2(g_{best} - x_{i,j}(t)) \tag{3.1}$$

$$x_{i,j}(t+1) = x_{i,j}(t) + V_{i,j}(t+1) \tag{3.2}$$

Ως p_{best} ορίζεται η καλύτερη θέση στην οποία έχει βρεθεί το ίδιο το σωματίδιο, ενώ ως g_{best} ορίζεται η καλύτερη θέση του χώρου στην οποία έχει βρεθεί κάποιο μέλος του σμήνους. Οι μεταβλητές c_1, c_2 είναι παράμετροι του αλγορίθμοι, ενώ οι μεταβλητές r_1, r_2 είναι τυχαίοι αριθμοί μεταξύ 0 και 1. Οι συνθήκες τερματισμού της εκτέλεσης ορίζονται από τον χρήστη και μπορεί να είναι η μη εύρεση μίας νέας βέλτιστης καλύτερης θέσης ή η ολοκλήρωση ενός προκαθορισμένου αριθμού επαναλήψεων.

Κεφάλαιο 4

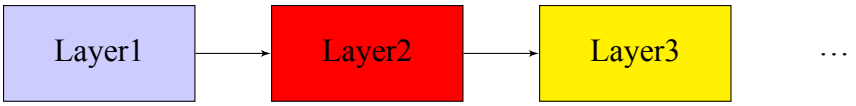
Σχετική έρευνα

4.1 Εισαγωγή

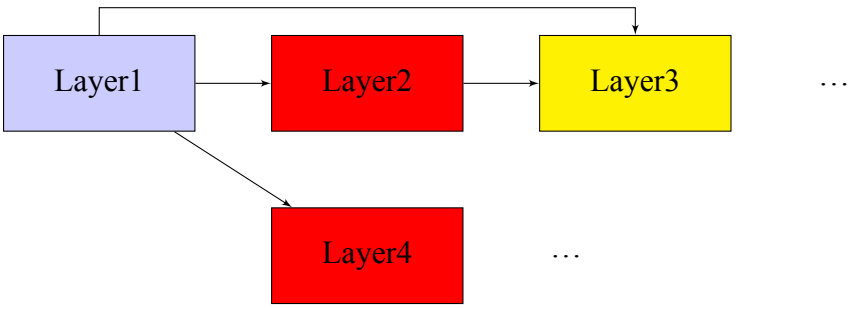
Η αναζήτηση αρχιτεκτονικής είναι ένα αντικείμενο έρευνας με ιστορία τουλάχιστον τριάντα ετών, με την πρώτη προσέγγιση στην εύρεση αρχιτεκτονικών ΤΝΔ να γίνεται με τη χρήση ενός γενετικού αλγορίθμου [32]. Στο παρόν κεφάλαιο θα γίνει ανασκόπηση βιβλιογραφίας καθώς και αναφορά σε μεθόδους, που έχουν χρησιμοποιηθεί από ερευνητές για την εύρεση της ιδανικής τοπολογίας ΤΝΔ.

4.2 Χώρος αναζήτησης

Όπως αναφέρθηκε στο κεφάλαιο 1 πριν εφαρμοστεί ο οποιοσδήποτε αλγόριθμος αναζήτησης θα πρέπει να καθοριστεί ο χώρος αναζήτησης. Ο χώρος αναζήτησης μπορεί να θεωρηθεί και ως το σημαντικότερο κομμάτι για την εύρεση της κατάλληλης αρχιτεκτονικής, καθώς επηρεάζει άμεσα τη φύση του τελικού ΤΝΔ. Ο απλούστερος χώρος αναζήτησης είναι αυτός που αποτελείται από ΤΝΔ σε μορφή αλυσίδας (chain structured neural networks), ο οποίος αποτελείται από νευρωνικά δίκτυα, των οποίων τα στρώματα είναι συνδεδμένα σε σειρά. Έτσι κάθε στρώμα δέχεται μία είσοδο από το αμέσως προηγούμενο και η έξοδος του είναι η είσοδος στο αμέσως επόμενο. Παράδειγμα ενός τέτοιου δικτύου απεικονίζεται στο σχήμα 4.1, με τον τύπο του κάθε στρώματος να εξαρτάται από το είδος του αμέσως προηγούμενου. Εναλλακτικά ο χώρος αναζήτησης μπορεί να αποτελείται από ΤΝΔ στα οποία ένα στρώμα μπορεί να συνδέεται με στρώματα που βρίσκονται βαθύτερα στην αρχιτεκτονική. Επίσης ένας κόμβος μπορεί να συνδέεται με περισσότερα από ένα στρώματα του επόμενου



Σχήμα 4.1: Αρχιτεκτονική δικτύου αλυσίδας



Σχήμα 4.2: Σύνθετα ΤΝΔ

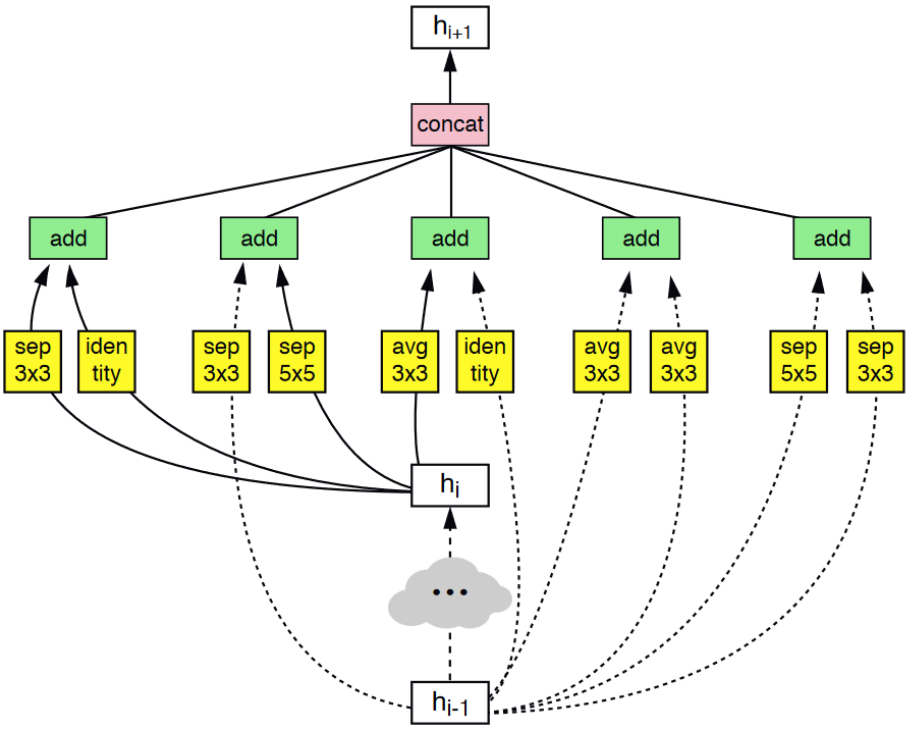
επιπέδου. Στο σχήμα 4.2 απεικονίζεται ένα παράδειγμα τέτοιου ΤΝΔ.

4.2.1 Αναζήτηση σε χώρο κελιών

Ο συγκεκριμένος χώρος αποτελεί μία ιδιαίτερη περίπτωση, από τον οποίον μπορούν να προκύψουν αρχιτεκτονικές ΤΝΔ. Οι αρχιτεκτονικές αυτές αποτελούνται από επαναλαμβανόμενες δομές που ονομάζονται κελιά (cells) [33] και αποτελούνται από ένα σύνολο μονάδων (blocks) που περιέχουν δύο φίλτρα που λαμβάνουν την είσοδο τους από προηγούμενα κελιά και εκτελούν κάποια πράξη, η οποία στη συνέχεια ορίζεται ως είσοδος σε κάποιο άλλο κελί. Είναι σημαντικό να επισημανθεί ότι ένα φίλτρο μπορεί να λαμβάνει και ως είσοδο, την έξοδο άλλον blocks. Όταν ολοκληρωθεί ο υπολογισμός των εξόδων όλων των blocks γίνεται κάποια πράξη συγχώνευσης τους, της οποίας το αποτέλεσμα είναι και η έξοδος του συγκεκριμένου κελιού. Τα κελιά μπορούν να είναι είτε normal, δηλαδη το τελικό τους αποτέλεσμα να έχει ίδιες διαστάσεις με τις εισόδους. Αλλιώς ένα κελί ονομάζεται reduction και οι διαστάσεις της εξόδου του είναι υποδιπλάσιες από αυτές των εισόδων του. Τα κελιά αυτά συνδέονται σε μορφή αλυσίδας. Στο σχήμα 4.3 απεικονίζεται ένα normal cell.

Overfitting σε κελιά

Το overfittting αποτελεί μεγάλο πρόβλημα και στις αρχιτεκτονικές κελιών οπότε έχουν αναπτυχθεί προσαρμοσμένες τεχνικές για να το αντιμετωπίσουν. Ο αλγόριθμος DropPath [8]



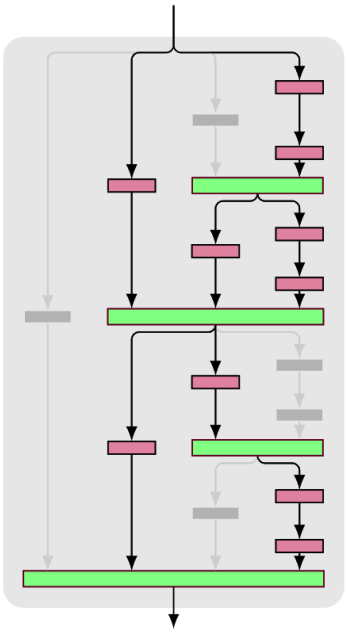
Σχήμα 4.3: Normal Cell [7]

αποτελεί εναλλακτική του dropout σε ΤΝΔ με κελιά και εφαρμόζεται κατά τη διάρκεια της εκπαίδευσης. Όταν αυτός εκτελείται γίνεται δειγματοληψία μονοπατιών του κελιού, διαγράφοντας προσωρινά τους τελεστές μίας πράξης συνένωσης. Μπορούν να εφαρμοστούν οι δύο παρακάτω στρατηγικές εναλλάξ

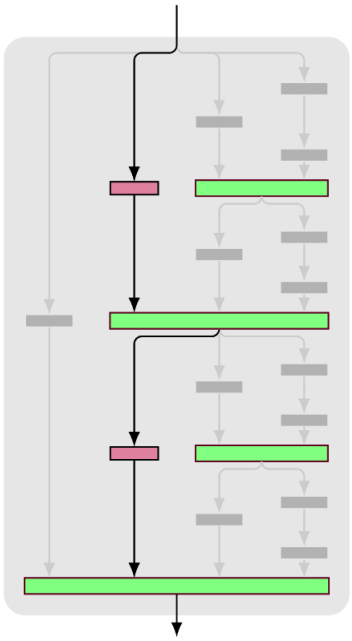
- Τοπική: Η είσοδος μίας συνδυαστικής πράξης διαγράφεται προσωρινά με μία σταθερή πιθανότητα.

- Ολική: Μόνο ένα συγκεκριμένο μονοπάτι παραμένει στο ΤΝΔ.

Οι δύο αυτοί διαφορετικοί τρόποι απεικονίζονται στα σχήματα 4.4 και 4.5 αντίστοιχα. Όπως διαπιστώθηκε μεταγενέστερα ο αλγόριθμος αυτός συμβάλλει στην αποφυγή του overfitting, ωστόσο όταν εφαρμόζεται είναι πιθανό να μειώσει την ακρίβεια της αρχιτεκτονικής. Έτσι έχει προταθεί και ο ScheduledDropPath [7] στον οποίον τα μονοπάτια δεν απορρίπτονται με μία σταθερή πιθανότητα αλλά με γραμμικά αυξανόμενη.



Σχήμα 4.4: Τοπική διαγραφή [8]



Σχήμα 4.5: Ολική διαγραφή [8]

4.3 Στρατηγικές αναζήτησης

Στο παρελθόν έχει προταθεί πληθώρα αλγορίθμων για την αναζήτηση καλών αρχιτεκτονικών ΤΝΔ, μεγάλο μειονέκτημα των οποίων όπως θα δούμε αποτελεί το γεγονός ότι καταναλώνουν πολλούς υπολογιστικούς πόρους [34].

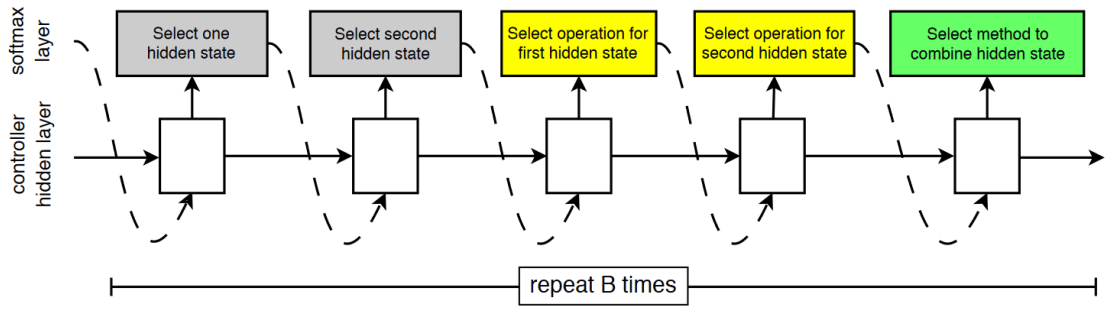
4.3.1 Reinforcement learning

Μία από τις πιο διαδεδομένες τεχνικές για την αναζήτηση αρχιτεκτονικής είναι το reinforcement learning, το οποίο εφαρμόζεται σε πολλούς τομείς της μηχανικής μάθησης και να μπορεί να χρησιμοποιηθεί και στην εύρεση μοντέλων ΤΝΔ. Σύμφωνα με αυτήν ένας πράκτορας βρίσκεται σε ένα δυναμικό περιβάλλον και καλείται να προσαρμόσει τη συμπεριφορά του κάνοντας διάφορες δοκιμές οι οποίες θα τον οδηγήσουν στο βέλτιστο αποτέλεσμα [35]. Το reinforcement learning έχει εφαρμοστεί στην αναζήτηση αρχιτεκτονικής ως εξής

1. Ένα αναδρομικό νευρωνικό δίκτυο που ονομάζεται ελεγκτής παράγει μία αρχιτεκτονική A με πιθανότητα p.

2. Ένα ΤΝΔ εκπαιδεύεται με αυτήν την αρχιτεκτονική και εξετάζεται η ακρίβεια του.

3. Ο ελεγκτής ενημερώνεται για την επίδοση της αρχιτεκτονικής και παράγει μία νέα αρχιτεκτονική.



Σχήμα 4.6: LSTM για δημιουργία αρχιτεκτονικής κελιών [7]

Όπως μπορεί να γίνει εύκολα αντιληπτό η επιλογή της αρχιτεκτονικής του ελεγκτή είναι κρίσιμη για την παραγωγή ενός ΤΝΔ με ικανοποιητική επίδοση. Μία τέτοια αρχιτεκτονική προτάθηκε από τον Zoph και άλλους [36], όπου ένα δίκτυο LSTM εκπαιδευόταν κατάλληλα και παρήγαγε ολοένα και καλύτερα νευρωνικά δίκτυα. Η έξοδος του ήταν τα χαρακτηριστικά ενός συνελικτικού νευρωνικού δικτύου. Σε συνέχεια αυτής της έρευνας προτάθηκε ένα LSTM δίκτυο, το οποίο παράγει την αρχιτεκτονική ενός κελιού δημιουργώντας την αρχιτεκτονική των normal, reduction κελιών με προκαθορισμένο σύνολο από blocks [7] και στη συνέχεια επιλεγόταν ένας αριθμός από κελιά τα οποία έμπαιναν σε σειρά. Σκοπός ήταν η εκπαίδευση των κελιών σε μικρότερα σύνολα δεδομένων και η αρχιτεκτονική αυτή να μπορούσε να εισαχθεί ως μέρος ενός ΤΝΔ που χρησιμοποιείται σε πιο μεγάλα σύνολα. Η αρχιτεκτονική του LSTM διακρίνεται στο σχήμα 4.6.

4.3.2 Ευφυΐα σμήνους

Ένα ακόμα σύνολο τεχνικών που χρησιμοποιούνται για την αναζήτηση αρχιτεκτονικής νευρωνικών δικτύων είναι οι αλγόριθμοι βασισμένοι σε ευφυΐα σμήνους, οι οποίοι έχουν προσαρμοστεί κατάλληλα για τη βελτιστοποίηση της εύρεσης.

Ο αλγόριθμος deepswarm

Ο αλγόριθμος deepswarm χρησιμοποιεί ACO για να αναζητήσει αρχιτεκτονικές. Πιο συγκεκριμένα δημιουργείται αρχικά ένα γράφημα το οποίο περιέχει μόνο τον κόμβο εισόδου. Στη συνέχεια παράγεται ένας διακριτός αριθμός από μυρμήγκια τα οποία περιμένουν σε μία ουρά σε αυτόν τον κόμβο και στη συνέχεια επιλέγουν ένα-ένα τη μετακίνηση τους από τον κόμβο-στρώμα s στο οποίο βρίσκονται κάποιο στρώμα r από τα διαθέσιμα, χρησιμοποιώντας

τις εξισώσεις του Ant Colony System [37].

$$s = \begin{cases} argmax_{u \in J_k(r)}(\tau(r,u) \cdot \eta(r,u)^{\beta}), \ q \leq q_0 \\ S, q > q_0 \end{cases} \tag{4.1}$$

Στην παραπάνω εξίσωση το $J_k(r)$ συμβολίζει το σύνολο που περιέχει όλους τους κόμβους που μπορεί να επισκεφθεί ένα μυρμήγκι από τη θέση που βρίσκεται, η μεταβλητή $\tau(r,u)$ συμβολίζει τη φερορμόνη του μονοπατιού που συνδέει δύο κόμβους u και r, ενώ η μεταβλητή $\eta(r,u)$ είναι η τιμή της ευρετικής συνάρτησης για αυτό το μονοπάτι. Το q_0 είναι μία μεταβλητή η οποία ρυθμίζει μαζί με το β το πόσο άπληστος είναι ο αλγόριθμος και ισχύει ότι $0 < q_0 \leq 1$ και $\beta \geq 0$. Η μεταβλητή S είναι μία τυχαία τιμή η οποία όταν χρησιμοποιείται ωθεί το μυρμήγκι να επιλέξει σύντομα μονοπάτια με μεγάλη συγκέντρωση φερορμόνης. Ορίζεται από την κατανομή

$$p_k(r,u) = \begin{cases} \frac{(\tau(r,u) \cdot \eta(r,u)^{\beta})}{\sum_{u \in J_k(r)(\tau(r,u) \cdot \eta(r,u)^{\beta})}}, \ s \in J_k(r) \\ 0 \end{cases} \tag{4.2}$$

Αφού ολοκληρωθεί η επιλογή του κόμβου το μυρμήγκι ελέγχει αν ο κόμβος υπάρχει στο γράφημα. Αν όχι τον προσθέτει και συνεχίζει να εκτελεί τη διαδικασία που περιεγράφηκε μεχρι να φτάσει στο μέγιστο επιτρεπτό βάθος της αρχιτεκτονικής για τη δεδομένη επανάληψη. Τότε ενημερώνει την τιμή της φερορμόνης για τα μονοπάτια τα οποία επέλεξε βάσει του παρακάτω κανόνα

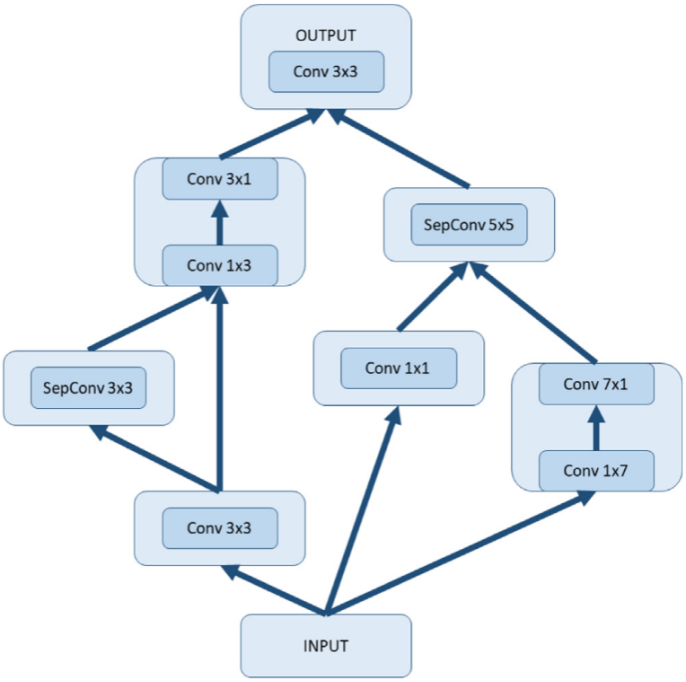
$$\tau(r,s) = (1-\rho) \cdot \tau(r,s) + \rho\tau_0, \ \rho \in \mathbb{R} \tag{4.3}$$

Στο τέλος όλες οι αρχιτεκτονικές οι οποίες κατασκευάστηκαν από τα μυρμήγκια ελέγχονται και το μυρμήγκι που παρήγαγε την καλύτερη αρχιτεκτονική ενημερώνει την τιμή της φερορμόνης για τα μονοπάτια τα οποία επισκέφτηκε σύμφωνα με την εξίσωση 4.4, όπου η μεταβλητή α ρυθμίζει την εξάτμιση της φερορμόνης και το $\Delta\tau(r,s)$ είναι ίσο με την απόδοση της αρχιτεκτονικής που κατασκευάστηκε για όσα μονοπάτια επιλέχτηκαν από το μυρμήγκι.

$$\tau(r,s) = (1-\alpha) \cdot \tau(r,s) + \alpha \cdot \Delta\tau(r,s), \ \alpha \in \mathbb{R} \tag{4.4}$$

IntelliSwAS

Εκτός από αλγορίθμους ACO έχουν προταθεί και αλγόριθμοι αναζήτησης PSO. Ο αλγόριθμος IntelliSwAS [9] κατασκευάζει κελιά όπως αυτό του σχήματος 4.7 χρησιμοποιώντας



Σχήμα 4.7: Τοπολογία κελιού μεγέθους 7 στον IntelliSwAS [9]

βελτιστοποίηση πλήθους σωματιδιών. Αρχικά ορίζεται ένας συγκεκριμένος αριθμός N σωματιδίων με τυχαίες αρχικές θέσεις p_i μεταξύ 0 και 2^{S_N-1} και ταχύτητες u_i. Σε κάθε επανάληψη m υπολογίζεται η επόμενη θέση p_{i+1}^m κάθε σωματιδίου σύμφωνα με την εξίσωση 4.5. Σε αντίθεση με τον συνηθισμένο PSO ένα σωματίδιο δε χρησιμοποιεί την καλύτερη θέση που βρέθηκε μέχρι στιγμής από όλα τα σωματίδια αλλά μόνο από συγκεκριμένα τα οποία ονομάζονται φίλοι. Η καλύτερη θέση όλων των φίλων συμβολίζεται με $p_{i,f}^m$, ενώ η καλύτερη θέση του ίδιου του σωματιδίου συμβολίζεται με $p_{i,b}^m$. Οι μεταβλητές s, f και w είναι τυχαίοι αριθμοί μικρότεροι ή ίσοι του 1 και τέλος τα j_s, j_f ορίζονται από τον χρήστη και ρυθμίζουν τη βαρύτητα της καλύτερης θέσης του σωματιδίου και της καλύτερης θέσης των φιλικών σωματιδίων.

$$\begin{aligned} p_{i+1}^m &= p_i^m + u_i^m \\ u_{i+1}^m &= w \cdot u_i^m + s \cdot j_s \cdot (p_{i,b}^m - p_i^m) + f \cdot j_f \cdot (p_{i,f}^m - p_i^m) \end{aligned} \tag{4.5}$$

Σε κάθε βήμα τα σωματίδια μετακινούνται σε θέσεις οι οποίες βελτιστοποιούν την απόδοση και έτσι όταν ολοκληρωθεί η εκτέλεση του αλγορίθμου έχουν κατασκευαστεί κελιά τα οποία μπορούν να συνδεθούν το ένα πίσω από το άλλο και έτσι να δημιουργηθούν βαθιές αρχιτεκτονικές.

4.4 Αξιολόγηση

Για την επιλογή του κατάλληλου ΤΝΔ απαιτείται η σύγκριση των παραγόμενων αρχιτεκτονικών μεταξύ τους. Ο πιο συνηθισμένος τρόπος για να γίνει αυτό είναι η εκπαίδευση των μοντέλων και η σύγκριση των τελικών συναρτήσεων απώλειας [38], ωστόσο έχει αποδειχθεί ότι αυτή η προσέγγιση απαιτεί πολύ μεγάλους χρόνους εκτέλεσης. Χαρακτηριστικό παράδειγμα ο deepswarm ο οποίος μπορούσε να εκτελεστεί το πολύ για 16 μυρμήγκια καθώς απαιτούσε τεράστια υπολογιστική ισχύ.

4.4.1 Επιτάχυνση αναζήτησης

Η επιτάχυνση της αναζήτησης είναι κρίσιμο σημείο και συνδέεται άμεσα με την αξιολόγηση της αρχιτεκτονικής. Στόχος είναι η ελάττωση του χρόνου εκτέλεσης όσο το δυνατόν περισσότερο χωρίς όμως να επηρεάζεται η ακρίβεια των αρχιτεκτονικών που δημιουργούνται. Για αυτό το λόγο έχουν προταθεί άλλες προσεγγίσεις όπως η αναζήτηση αρχιτεκτονικής σε μικρότερα σύνολα δεδομένων και στη συνέχεια η χρήση της καλύτερης σε μεγαλύτερα [7], [36], ώστε να μειωθεί ο χρόνος εκτέλεσης σε μέρες GPU. Επίσης η επαναχρησιμοποίηση των βαρών από προηγούμενες εκτελέσεις [34], [39] ή ο διαμοιρασμός τους μεταξύ συγγενικών αρχιτεκτονικών φαίνεται να επιταχύνει σημαντικά τη διαδικασία της αναζήτησης. Μία άλλη λύση είναι ο τερματισμός της εκπαίδευσης πριν τη σύγκλιση, καθώς όταν οι άνθρωποι αναπτύσσουν κάποια αρχιτεκτονική κρίνουν εαν πρέπει να συνεχιστεί η εκπαίδευση της κρίνοντας από την καμπύλη απόδοσης τους [40]. Έτσι η εκπαίδευση πρέπει να σταματήσει όσο το δυνατόν νωρίτερα ώστε να εξοικονομηθούν υπολογιστικοί πόροι. Στο NAO [41] χρησιμοποιείται μία συνάρτηση f, η οποία πρέπει να μεγιστοποιηθεί ώστε να βρεθεί η καλύτερη αρχιτκτονική. Τέλος αξίζει να αναφερθεί ότι έχουν προταθεί και άλλοι τρόποι για την επιτάχυνση της αναζήτησης αρχιτεκτονικών. Για παράδειγμα στο [42] χρησιμοποιήθηκε θεωρία πιθανοτήτων για τον τερματισμό της εκπαίδευσης νωρίτερα.

Κεφάλαιο 5

Προτεινόμενες τεχνικές

5.1 Εισαγωγή

Σε αυτό το κεφάλαιο περιγράφεται ο αλγόριθμος ο οποίος χρησιμοποιήθηκε για την αναζήτηση αρχιτεκτονικής ΤΝΔ. Ο αλγόριθμος βασίζεται σε έννοιες οι οποίες αναφέρθηκαν προηγουμένως και χρησιμοποιεί ευφυΐα σμήνους για την αναζήτηση αρχιτεκτονικών για κελιά όπως αυτά του [7], οι οποίες θα μεγιστοποιούν την ακρίβεια κατηγοριοποίησης για σύνολα δεδομένων εικόνας. Παράλληλα αναλύεται και μία ακόμα τεχνική η οποία εφαρμόζεται στα πλαίσια της μεθόδου για βελτίωση της επίδοσης της.

5.2 Ο αλγόριθμος AntCell

Για τη δημιουργία των κελιών θα χρησιμοποιηθεί ACO. Πριν όμως αναλυθεί ο τρόπος με τον οποίον εργάζονται τα μυρμήγκια είναι απαραίτητη η περιγραφή του χώρου στον οποίον κινούνται. Στην πραγματικότητα θα δημιουργηθούν δύο διαφορετικοί γράφοι από τους οποίους προκύπτουν το normal και reduction cell.

5.2.1 Δημιουργία γράφων

Βασική παράμετρος της μεθόδου είναι τα $maxBlocks$, η οποία καθορίζει πόσες πράξεις πρόσθεσης υπάρχουν στο κάθε κελί. Αφού καθοριστεί αυτή από τον χρήστη δημιουργείται ένας γράφος ο οποίος αποτελείται από τους παρακάτω τύπους κόμβων

- Κόμβος εκκίνησης: Εκεί τοποθετείται κάθε μυρμήγκι κατά την εκκίνηση της εκτέλε-

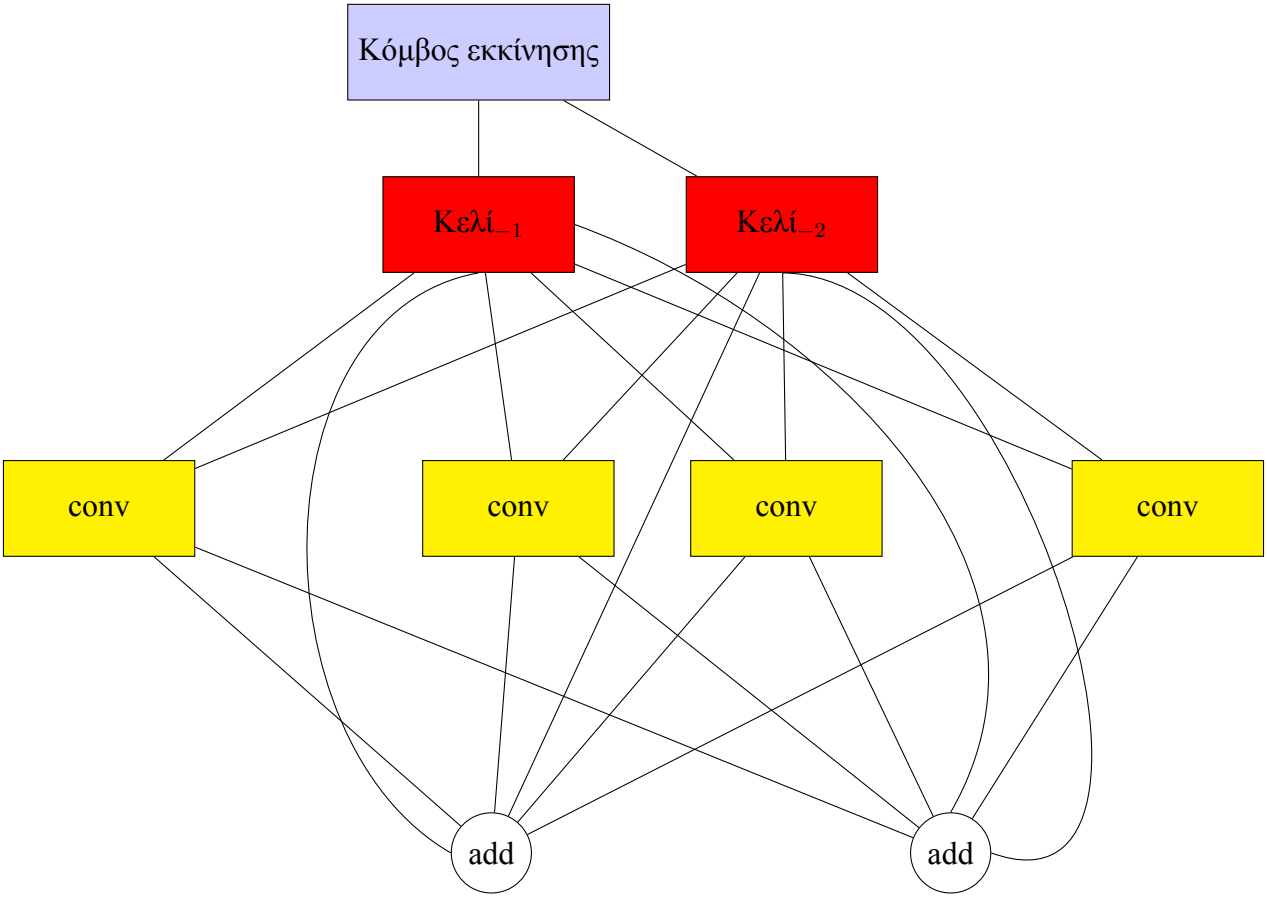
σης.

- Κόμβοι εισόδου: Είναι δύο κόμβοι οι οποίοι αντιπροσωπεύουν τις εισόδους του κελιού, οι οποίες είναι η έξοδος του αμέσως προηγούμενου κελιού και η έξοδος του κελιού δύο εισόδων πίσω. Αν τουλάχιστον ένα από αυτά τα δύο κελιά δεν υπάρχει τότε η αντίστοιχη είσοδος είναι η είσοδος της αρχιτεκτονικής.
- Κόμβοι πρόσθεσης: Το πλήθος τους είναι $maxBlocks$ και αντιπροσωπεύουν τις πράξεις πρόσθεσης που θα χρησιμοποιηθούν στην αρχιτεκτονική που θα κατασκευαστεί.
- Κόμβοι στρωμάτων: Οι συγκεκριμένες μονάδες αντιπροσωπεύουν τα στρώματα τα οποία θα χρησιμοποιηθούν μέσα σε ένα κελί.

Οι κόμβοι στρώματων ορίζονται εκ των προτέρων και υπάρχουν διάφοροι συνδυασμοί για τις παραμέτρους τους. Για παράδειγμα τα συνελικτικά στρώματα αποτελούνται από τράπεζες φίλτρων των οποίων ο αριθμός αλλάζει ριζικά τις αρχιτεκτονικές που δημιουργούνται. Για αυτόν τον λόγο χρησιμοποιούνται οι μεταβλητές $max\_filters$ και $min\_filters \leq max\_filters$ οι οποίες καθορίζουν το διάστημα στο οποίο ανήκει ο αριθμός των φίλτρων για κάθε διακριτό κόμβο στρώματος. Έτσι δημιουργούνται κόμβοι συνέλιξης με τον αριθμό των φίλτρων τους να έχει αρχική τιμή $min\_filters$ και στη συνέχεια να αυξάνεται εκθετικά μέχρι να φτάσει την τιμή $max\_filters$. Συνολικά οι κόμβοι στρωμάτων που θα αποτελούν τον γράφο είναι οι παρακάτω

- Convolution (3, 3)
- Depthwise conv (3, 3)
- Max pool (3, 3)
- Convolution (5, 5)
- Depthwise conv (5, 5)
- Max pool (5, 5)
- Convolution (1, 1)
- Separable conv (3, 3)
- Max pool (7, 7)
- Convolution (1, 7)
- Separable conv (5, 5)
- Average pool (3, 3)

Αφού ορίστηκε το είδος του κάθε κόμβου δημιουργούνται δύο πανομοιότυποι γράφοι για κάθε είδος κελιού (αν και οι μεταβλητές τους θα μπορούσαν να είναι διαφορετικές), στους οποιους ο κόμβος εκκίνησης είναι συνδεδεμένος και με τους δύο κόμβους εισόδου. Οι τελευταίοι συνδέονται με όλους τους κόμβους στρωμάτων, των οποίων το πλήθος είναι $maxBlocks+2$ (για κάθε διαφορετικό είδος στρώματος) και με όλους τους κόμβους πρόσθεσης. Ένας τέτοιος γράφος για $maxBlocks = 2$ και έναν κόμβο στρώματος (για παράδειγμα Convolution (3, 3) με δύο φίλτρα ή εν συντομία conv) απεικονίζεται στο σχήμα 5.1.



Σχήμα 5.1: Παράδειγμα γράφου για $maxBlocks = 2$ και ένα είδος στρώματος

5.2.2 Δημιουργία αρχιτεκτονικής κελιών

Αφού ολοκληρωθεί η δημιουργία των γράφων και αρχικοποιηθούν οι φερομόνες των ακμών ένα μυρμήγκι τοποθετείται στον κόμβο εκκίνησης έκαστου. Αυτό ξεκινάει να διασχίζει τον γράφο χρησιμοποιώντας τις ακμές του κόμβου στον οποίον βρίσκεται, τους κανόνες επιλογής του Ant System όπως αυτοί περιεγράφηκαν στο προηγούμενο κεφάλαιο και ένα σύνολο κανόνων U για να μετακινηθεί σε μία θέση η οποία είναι διαθέσιμη από το σημείο στο οποίο βρίσκεται. Οι κανόνες του συνόλου εμποδίζουν τον πράκτορα να μετακινηθεί σε κόμβους που δε μπορούν να ανήκουν στην αρχιτεκτονική εξαιτίας της θέσης που βρισκόταν προηγουμένως το μυρμήγκι και είναι οι παρακάτω

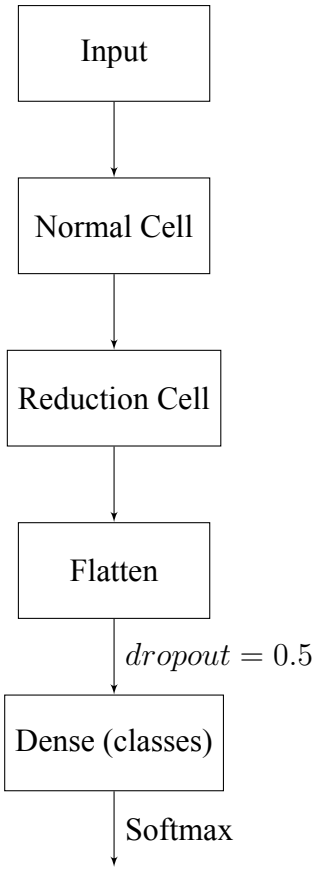
- Το μυρμήγκι δε μπορεί να μετακινηθεί σε κάποιον κόμβο στρώματος αν βρίσκεται σε κόμβο πρόσθεσης ο οποίος έχει αριθμό εισόδων ίσο με τον μέγιστο. Αξίζει να σημειωθεί ότι οι είσοδοι προσδιορίζουν τον αριθμό των φορών τις οποίες ένα μυρμήγκι μπορεί να χρησιμοποιήσει τον συγκεκριμένο κόμβο (δύο για κόμβο πρόσθεσης, μία για κόμβο στρώματος και ∞ για τους κόμβους εκκίνησης και κελιών).

- Το μυρμήγκι μπορεί να μετακινηθεί μόνο σε κόμβο εισόδου αν η αμέσως προηγούμενη θέση ήταν ένας κόμβος πρόσθεσης και βρίσκεται σε κόμβο στρώματος.

- Ο κόμβος εκκίνησης είναι προσπελάσιμος από τους κόμβους εισόδων, μόνο αν δεν ήταν η προηγούμενη θέση του πράκτορα.

- Αν η τρέχουσα θέση που βρίσκεται ο πράκτορας ανήκει στους κόμβους στρώματος και η αμέσως προηγούμενη θέση του ήταν κάποιος κόμβους εισόδου, τότε μπορεί να μετακινηθεί μόνο σε κόμβο πρόσθεσης.

Συγκεκριμένα ο πράκτορας μετακινείται σε κάποιον από τους κόμβους κελιών και συνεχίζει την πορεία του σε κάποιον από τους κόμβους στρώματος και τελικά φτάνει σε κάποιον από τους κόμβους πρόσθεσης. Από εκεί ξεκινάει την πορεία του προς τα πίσω και χρησιμοποιεί κάποιον εναλλακτικό κόμβο στρώματος ώστε να επιστρέψει στη θέση που βρισκόταν κατά την εκκίνηση της εκτέλεσης. Κάθε φορά που επιλέγεται ένας κόμβος το μυρμήγκι αυξάνει τον αριθμό εισόδων του. Επίσης γίνεται αύξηση των εισόδων ενός κόμβου πρόσθεσης εάν το μυρμήγκι επέλεξε έναν κόμβο κελιού και πριν από δύο θέσεις βρισκόταν στον πρώτο. Η διαδικασία αυτή επαναλαμβάνεται μέχρι όλοι οι κόμβοι πρόσθεσης να έχουν δύο εισόδους. Μόλις ολοκληρωσει τη διαδρομή του, το μυρμήγκι ενημερώνει τις φερομόνες των ακμών



Σχήμα 5.2: Αρχιτεκτονική ΤΝΔ προς αξιολόγηση

τις οποίες διαπέρασε, χρησιμοποιώντας τον τοπικό κανόνα του AntSystem και το υπογράφημα το οποίο προέκυψε μετατρέπεται σε αρχιτεκτονική. Για να αποφευχθεί το overfitting χρησιμοποιήθηκε η τοπική διαγραφή της τεχνικής ScheduledDropPath.

5.2.3 Αξιολόγηση κελιών

Για να γίνει η αξιολόγηση των αρχιτεκτονικών των κελιών που δημιουργήθηκαν κατασκευάζεται ένα μοντέλο, το οποίο έχει τη δομή που διακρίνεται στο σχήμα 5.2. Αυτό εκπαιδεύεται για k επαναλήψεις και στη συνέχεια ελέγχεται η ακρίβεια του στα άγνωστα δεδομένα. Τέλος οι φερομόνες των μονοπατιών που ανήκουν στην καλύτερη αρχιτεκτονική που έχει βρεθεί ως εκείνη τη στιγμή ενημερώνονται σύμφωνα με τον γενικό κανόνα του AntSystem με $\Delta\tau(r,s) = acc$, ενώ για τις ακμές που δεν ανήκουν στην καλύτερη αρχιτεκτονική ισχύει ότι $\Delta\tau(r,s) = 0$. Μετά την αναζήτηση m αρχιτεκτονικών ολοκληρώνεται η εκτέλεση του αλγορίθμου και τα κελιά που βρέθηκαν μπορούν να χρησιμοποιηθούν για την κατασκευή βαθύτερων αρχιτεκτονικών.

5.2.4 Ενημέρωση απληστίας

Στα πλαίσια του AntCell αναπτύχθηκε μία ακόμα τεχνική για την εύρεση καλύτερων αρχιτεκτονικών κελιών. Στον αλγόριθμο AntSystem το greediness μένει σταθερό. Αντιθέτως στη μέθοδο που παρουσιάστηκε στο παρόν κεφάλαιο η μεταβλητή αυτή έχει μία αρχική τιμή $min\_greediness$ και όταν το μυρμήγκι εντοπίζει καλύτερη αρχιτεκτονική η μεταβλητή μειώνεται γραμμικά (εφόσον η νέα τιμή δεν είναι μικρότερη από $min\_greediness$) κατά $step$. Όταν το μυρμήγκι δεν εντοπίσει μία δεδομένη επανάληψη καλύτερη αρχιτεκτονική τότε η μεταβλητή $greediness$ αυξάνεται γραμμικά κατά $step$ (εφόσον δεν ξεπερνάει μία τιμή $max\_greediness$). Με αυτόν τον τρόπο ο πράκτορας ωθείται να αναζητείσει διαφορετικές αρχιτεκτονικές όσο αυτές που δημιουργεί έχουν μεγαλύτερη ακρίβεια, ενώ αν αδυνατεί να βρει καλύτερα κελιά τότε ξεκινάει να δημιουργεί αρχιτεκτονικές οι οποίες μοιάζουν με τις καλύτερες που έχει εντοπίσει μέχρι τη συγκεκριμένη επανάληψη.

Κεφάλαιο 6

Πειράματα

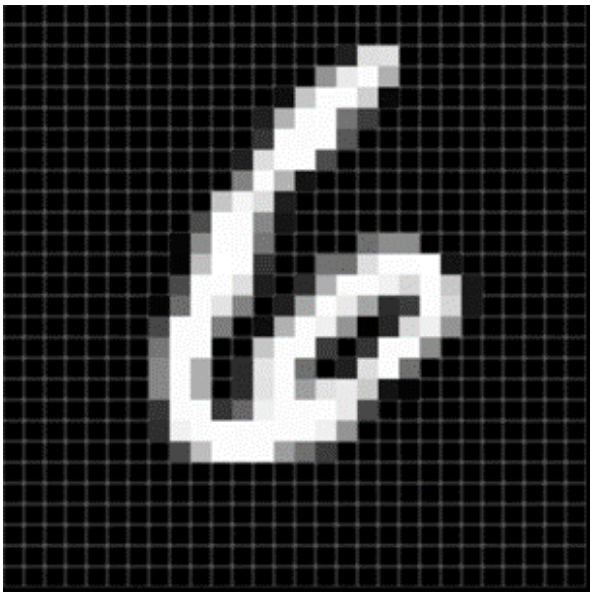
Σε αυτό το κεφάλαιο θα γίνει αναφορά στις επιδόσεις του αλγορίθμου. Για αυτό το σκοπό θα ελεγχθούν οι αρχιτεκτονικές οι οποίες δημιουργούνται από τη μέθοδο AntCell που περιεγράφηκε στο κεφάλαιο 5. Ως καλύτερη αρχιτεκτονική θεωρούμε αυτή την οποία πέτυχε τη μεγαλύτερη ακρίβεια στα δεδομένα ελέγχου του αντίστοιχου συνόλου, ωστόσο λαμβάνεται υπόψιν και ο αριθμός των παραμέτρων κάθε αρχιτεκτονικής, τον οποίον προσπαθούμε να μειώσουμε όσο το δυνατόν περισσότερο.

6.1 Σύνολα δεδομένων

Τα σύνολα δεδομένων που θα χρησιμοποιηθούν είναι το MNIST [44], Fashion MNIST [45] και CIFAR-10 [46]. Επιλέγησαν τα συγκεκριμένα σύνολα καθώς απαιτούν διαφορετικά είδη αρχιτεκτονικών που να βελτιστοποιούν την ακρίβεια κατηγοριοποίησης [34], ενώ έχουν χρησιμοποιηθεί στο παρελθόν από πληθώρα ερευνητών για να εξετάσουν την επίδοση των αλγορίθμων τους.

MNIST

Το σύνολο αυτό αποτελείται από 70.000 εικόνες σε κλίμακα του γκρι (60.000 για εκπαίδευση του δικτύου και 10000 για τον έλεγχο της ακρίβειας του) διαστάσεων $28x28$. Κάθε μία από αυτές απεικονίζει ένα χειρόγραφο ψηφίο από το 0 έως και το 9 όπως αυτό του σχήματος 6.1, τα οποία θα προσπαθήσουμε να αναγνωρίσουμε ,δηλαδή θα γίνει κατηγοριοποίηση της εικόνας για την εύρεση του είδους στο οποίο ανήκε, με τη χρήση μίας αρχιτεκτονικής συνελικτικού νευρωνικού δικτύου, το οποίο δημιουργήθηκε από τον AntCell.



Σχήμα 6.1: Εικόνα από το MNIST

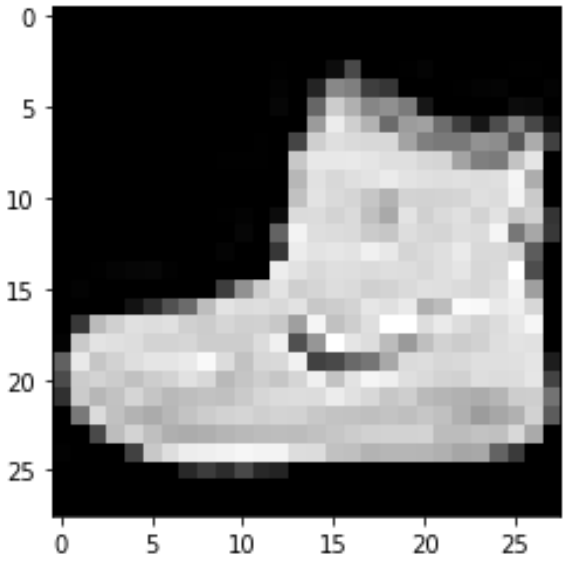
Fasion-MNIST

Όπως και το MNIST έτσι και το Fashion MNIST αποτελείται από 60.000 εικόνες για εκπαίδευση ενός δικτύου και 10.000 για έλεγχο της ακρίβειας του, οι οποίες έχουν διαστάσεις $28x28$. Κάθε εικόνα περιλαμβάνει ένα αντικείμενο μόδας όπως αυτό στο σχήμα 6.2, για την αναγνώριση του οποίου θα χρησιμοποιηθεί ΤΝΔ. Πιο συγκεκριμένα μία εικόνα του ΣΔ μπορεί να απεικονίζει ένα από τα παρακάτω

- T-shirt
- Παντελόνι
- Πουλόβερ
- Φόρεμα
- Παλτό
- Σανδάλι
- Πουκάμισο
- Αθλητικό παπούτσι
- Τσάντα
- Μπότα

CIFAR-10

Το CIFAR-10 περιλαμβάνει συνολικά 60.000 εικόνες (50.000 για εκπαίδευση και 10.000 για έλεγχο) διαστάσεων $32x32$, όπως αυτή στο σχήμα 6.3. Κάθε φωτογραφία μπορεί να δείχνει ένα αντικείμενο το οποίο ανήκει σε μία από τις παραπάνω κλάσεις



Σχήμα 6.2: Εικόνα από το Fashion-MNIST

- Αεροπλάνο
- Αυτοκίνητο
- Πτηνό
- Γάτα
- Ελάφι
- Σκύλος
- Βάτραχος
- Άλογο
- Πλοίο
- Φορτηγό

6.2 Παράμετροι

Για την εύρεση των επιδόσεων του AntCell θα πρέπει να προκαθοριστούν οι τιμές των παραμέτρων για την εκτέλεση των πειραμάτων. Θα γίνουν πέντε εκτελέσεις της μεθόδου για κάθε σύνολο δεδομένων με τη μεταβλητή $maxBlocks$ να λαμβάνει τιμές μεταξύ 1 και 5. Για κάθε σύνολο θα γίνει η αναζήτηση 5 αρχιτεκτονικών, ενώ το βήμα με το οποίο αυξομειώνεται η απληστεία (greediness) και τα α, ρ είναι ίσα με 0.1, ενώ αν V είναι το σύνολο κορυφών κάποιου από τους δύο γράφους τότε

$$\eta(r, u) = 1, \ \forall r, u \in V$$

Τέλος αξίζει να σημειωθεί ότι σε κάθε προσπάθεια γίνεται εκπαίδευση του κάθε ΤΝΔ

Σχήμα 6.3: Εικόνα από το Cifar-10

Μεταβλητή	Τιμή
min_filters	8
max_filters	32
Αρχικές εποχές	5
Τελικές εποχές	5

Πίνακας 6.1: Οι τιμές των παραμέτρων σε κάθε εκτέλεση για τα δύο ΣΔ

για k εποχές και μόλις βρεθεί το καλύτερο ΤΝΔ (μετά το πέρας των αναζητήσεων) αυτό εκπαιδεύεται για άλλες m επαναλήψεις. Για το $dropout$ ισχύει η σχέση 6.1.

$$dropout = \begin{cases} 0.015 \cdot epoch & 1 \leq epoch \leq k \\ 0.015 \cdot (epoch - k) & k + 1 \leq epoch \leq m + k \end{cases} \tag{6.1}$$

MNIST και Fashion-MNIST

Για τα δύο αυτά σύνολα θα χρησιμοποιηθούν οι ίδιες παράμετροι για κάθε εκτέλεση. Στον πίνακα 6.1 δίνονται οι τιμές των υπόλοιπων παραμέτρων, οι οποίες παραμένουν ίδιες σε κάθε εκτέλεση με διαφορετικό $maxBlocks$.

Μεταβλητή	Τιμή
min_filters	32
max_filters	128
Αρχικές εποχές	10
Τελικές εποχές	10

Πίνακας 6.2: Οι τιμές των παραμέτρων σε κάθε εκτέλεση για το CIFAR-10

CIFAR-10

Επειδή όπως διαπιστώθηκε οι τιμές των παραμέτρων που χρησιμοποιήθηκαν για τα δύο προηγούμενα σύνολα δεν αρκούσαν για να δημιοργήσει ο AntCell αρχιτεκτονικές με υψηλή ακρίβεια στο CIFAR-10 θα χρησιμοποιηθούν οι τιμές παραμέτρων που δίνονται στον πίνακα 6.2.

6.3 Αποτελέσματα

Σε αυτήν την ενότητα παρουσιάζονται συγκριτικά αποτελέσματα της μεθόδου μας με άλλες μεθόδους οι οποίες έχουν προταθεί σε άλλες ερευνητικές εργασίες. Θα παρουσιαστεί η καλύτερη αρχιτεκτονική που προέκυψε βάσει ακρίβειας (AntCell-best) αφού έχουν ολοκληρωθεί οι $m + k$ εποχές, η οποία παρέχεται στο παράρτημα Α, αλλά και το μέσο error rate της μεθόδου από όλα τα πειράματα για κάθε ΣΔ (AntCell-avg), όπου η ακρίβεια μετράται στο τέλος κάθε εποχής για κάθε υποαρχιτεκτονική.

6.3.1 MNIST και Fasion-MNIST

Στον πίνακα 6.3 παρουσιάζονται τα συγκριτικά αποτελέσματα του error rate για τα δύο ΣΔ, όπου

$$error\ rate = 1 - accuaracy$$

Ο αριθμός που υπάρχει εντός της παρένθεσης στην καλύτερη αρχιτεκτονική αναφέρεται στον αριθμό των $maxBlocks$.

Μέθοδος	MNIST	Fashion-MNIST
RANDOM	1.79%	11.36%
GRID	1.68%	10.28%
SPMT	1.36%	9.62%
SMAC	1.43%	10.87%
SEAS	1.07%	8.05%
AutoKeras BFS	1.56%	9.13%
AutoKeras BO	1.83%	7.99%
AutoKeras BFS	0.55%	7.42%
DeepSwarm Average	0.46%	6.75%
DeepSwarm Best	0.39%	6.44%
AntCell-avg	1.20%	9.66%
AntCell-best	0.92% (5)	8.46% (4)

Πίνακας 6.3: Σύγκριση Error rate (%) του AntCell με άλλες μεθόδους

6.3.2 CIFAR-10

Όσον αφορά το CIFAR-10 στον πίνακα 6.4 γίνεται η σύγκριση με άλλες αρχιτεκτονικές ή μεθόδους. Παρόλο που η μέθοδος μας δεν παράγει τόσο χαμηλά error rate όσο τα NASNet, για την εύρεση των τελευταίων χρειάστηκε τεράστιος αριθμός GPUs (500) και οι αρχιτεκτονικές που βρέθηκαν τελικά αποτελούνταν από σημαντικά περισσότερες παραμέτρους. Παράλληλα αν και η μέθοδος μας υστερεί όσον αφορά το error rate, οι αρχιτεκτονικές που προέκυψαν αποτελούνταν από αρκετά λιγότερες μεταβλητές, όπου μάλιστα η κάθε μία από αυτές συμβάλλει περισσότερο στην αύξηση της ακρίβειας της αρχιτεκτονικής, σε σύγκριση με όλα τα NASNETs με τα οποία έγινε σύγκριση.

Μέθοδος/Μοντέλο	Αριθμός μεταβλητών	Error rate (%)	Ακρίβεια (%)/Μεταβλητή
NASNet-A (6 @ 768)	3.3M	3.41	$2.926969 \cdot 10^{-5}$
NASNet-A (6 @ 768) + cutout	3.3M	2.65	$2.95 \cdot 10^{-5}$
NASNet-A (7 @ 2304)	27.6M	2.97	$3.51557 \cdot 10^{-6}$
NASNet-A (7 @ 2304) + cutout	27.6M	2.40	$3.53623 \cdot 10^{-6}$
NASNet-B (4 @ 1152)	2.6M	3.73	$3.702692 \cdot 10^{-5}$
NASNet-C (4 @ 640)	3.1M	3.59	$3.11 \cdot 10^{-5}$
AntCell-avg	658.984K	32.69	$1.0214208 \cdot 10^{-4}$
AntCell-best	542K	26.49	$1.356273 \cdot 10^{-4}$

Πίνακας 6.4: Σύγκριση Error rate (%) και ακρίβειας ανά μεταβλητή του AntCell με τα NASNETs [7] για το Cifar-10

Κεφάλαιο 7

Συμπεράσματα

Ακολουθεί ανασκόπηση των αποτελεσμάτων της μεθόδου που αναπτύχθηκε, καθώς επίσης και προτάσεις για τη βελτίωση της.

7.1 Σύνοψη και συμπεράσματα

Αν και η αναζήτηση αρχιτεκτονικής ΤΝΔ είναι μία διαδικασία η οποία απαιτεί τεράστια υπολογιστική ισχύ, ο AntCell παράγει αρχιτεκτονικές οι οποίες μπορούν να εκπαιδευτούν σε μικρό χρονικό διάστημα εξαιτίας του μικρού αριθμού των μεταβλητών τους. Επίσης αξίζει να σημειωθεί ότι με την αύξηση της μεταβλητής $maxBlocks$ παρατηρήθηκε και μία αύξηση στη μέση ακρίβεια των ΤΝΔ που προέκυψαν. Αυτό συνέβη γιατί όσο πιο πολλά είναι τα blocks τόσο αυξάνεται ο αριθμός των μεταβλητών και η πιθανότητα η κάθε αρχιτεκτονική να έχει μεγαλύτερο βάθος. Παρόλα αυτά θα θέλαμε να επισημάνουμε ότι η αύξηση των $blocks$ συνεπάγεται την αύξηση των Batch Normalization στρωμάτων, οπότε απαιτείται η χρήση μεγαλύτερου ρυθμού εκπαίδευσης γιατί αυτά έχουν την τάση να σταθεροποιούν την εκπαίδευση των ΤΝΔ. Όσον αφορά τα ΣΔ τα οποία χρησιμοποιήθηκαν για την εξέταση της επίδοσης της μεθόδου μας παρατηρήθηκε ότι

- Το error rate ήταν συγκρίσιμο με αυτό των άλλων μεθόδων για τα MNIST και Fasion-MNIST, αν και τα δίκτυα που κατασκευάστηκαν είχαν πολύ μικρό βάθος. Επίσης οι κόμβοι φίλτρων διέθεταν μικρό αριθμό μέγιστων φίλτρων (32), ενώ για παράδειγμα ο DeepSwarm [34] χρησιμοποιούσε στις καλύτερες αρχιτεκτονικές 256 φίλτρα. Επίσης τα κελιά δύναται να χρησιμοποιηθούν για τη δημιουργία ΤΝΔ μεγαλύτερου βάθους που κλιμακώνονται εύκολα, οπότε είναι πιθανόν η ακρίβεια της μεθόδου μας να μπορεί

να αυξηθεί με την αύξηση του βάθους της αρχιτεκτονικής.

- Ο AntCell παράγει αρχιτεκτονικές με πολύ λιγότερες μεταβλητές για το Cifar-10 και έτσι η αναζήτηση δύναται να πραγματοποιηθεί πολύ γρηγορότερα σε σχέση με τις υπόλοιπες μεθόδους, αν και το error rate είχε μεγάλη απόκλιση. Ωστόσο παρατηρήθηκε ότι οι μεταβλητές των ΤΝΔ που προέκυπταν από τον AntCell είχαν μεγαλύτερη αξία σε σχέση με αυτές των μοντέλων που προέκυψαν από τις μεθόδους με τις οποίες έγινε σύγκριση, καθώς το πηλίκο Ακρίβεια/μεταβλητή ήταν σαφώς μεγαλύτερο στη δική μας μέθοδο.

- Εξαιτίας της στοχαστικής φύσης του αλγορίθμου μπορεί να χρειαστούν πολλές δοκιμές για την εύρεση ΤΝΔ με ικανοποιητικό error rate.

Εν κατακλείδει, ο AntCell παράγει βασικές δομές, τις οποίες ονομάζουμε κελιά, και τις οποίες μπορούμε να τοποθετήσουμε σειριακά για να δημιουργήσουμε κλιμακώσιμες αρχιτεκτονικές. Για την αξιολόγηση των κελιών χρησιμοποιήθηκέ το πιο απλό ΤΝΔ που θα μπορούσε να κατασκευαστεί από ένα normal και ένα reduction cell, αρχιτεκτονική η οποία δεν προσφέρει υψηλή ακρίβεια για όλα τα ΣΔ, στα οποία εξετάστηκε ο αλγόριθμος. Για αυτό το λόγο συμπεράναμε ότι ανάλογα τα δεδομένα μπορεί να χρειαστεί η αύξηση του αριθμού των κελιών ώστε να δημιουργηθούν πιο περίπλοκες αρχιτεκτονικές οι οποίες είναι σε θέση να κατηγοριοποιήσουν εικόνες με μεγαλύτερη ακρίβεια. Η μέθοδος μας θα μπορούσε να χρησιμοποιηθεί σε συσκευές με χαμηλή μνήμη και υπολογιστική ισχύ, στις οποίες δεν είναι απαραίτητη η κατηγοριοποίηση εικόνων με μεγάλη ακρίβεια.

7.2 Μελλοντικές επεκτάσεις

Για την εύρεση αρχιτεκτονικών με μεγαλύτερη ακρίβεια προτείνονται οι παρακάτω βελτιώσεις

- Χρήση του AntCell για αναζήτηση κελιών που θα χρησιμοποιηθούν σε ΤΝΔ που αποτελούνται από μεγαλύτερο αριθμό παραμέτρων, ώστε να γίνει διαπιστωθεί αν η αύξηση των τελευταίων θα μειώσει το error rate (ιδιαίτερα στο Cifar-10).

- Έλεγχος των επιδόσεων του AntCell κατά τη μεταφορά αρχιτεκτονικών (transferable architectures).

- Διεξαγωγή πειραμάτων με αυξημένα $min\_filters$ και $max\_filters$ για παραγωγή αρχιτεκτονικών που αποτελούνται από μεγαλύτερο αριθμό μεταβλητών.
- Κοινή χρήση βαρών μεταξύ συγγενικών αρχιτεκτονικών.

Βιβλιογραφία

[1] Stuart Baumann and Margaryta Klymak. Fixed point acceleration in r. *The R Journal*, 11:359, 01 2019.

[2] Rahul Jayawardana and Thusitha Bandaranayake. Analysis of optimizing neural networks and artificial intelligent models for guidance, control, and navigation systems. Technical report, 04 2021.

[3] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29(9):2352–2449, 2017.

[4] Arjun Haridas and Martin Dyrba. Comparison of convolutional neural network training parameters for detecting alzheimers disease and effect on visualization, 08 2020.

[5] Ian Jenkins, Ludvig Gee, Alessia Knauss, Hang Yin, and Jan Schroeder. Accident scenario generation with recurrent neural networks. pages 3340–3345, 11 2018.

[6] Mohammad Jabbarpour, Houman Zarrabi, Jason Jung, and Pankoo Kim. A green ant-based method for path planning of unmanned ground vehicles. *IEEE Access*, PP:1–1, 01 2017.

[7] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8697–8710, 2018.

[8] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *ArXiv*, abs/1605.07648, 2017.

[9] Sergiu Nistor and Gabriela Czibula. Intelliswas: Optimizing deep neural network architectures using a particle swarm-based approach. *Expert Systems with Applications*, 187:115945, 09 2021.

[10] Simon Haykin. *Neural Networks and Learning Machines*. Pearson, Upper Saddle River, New Jersey 07458, third edition, 2009.

[11] Κωνσταντίνος Διαμαντάρας. *Τεχνητά Νευρωνικά Δίκτυα*. Κλειδάριθμος, Αθήνα, 2007.

[12] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2 edition, 2014.

[13] S. Abirami and P. Chitra. Chapter fourteen - energy-efficient edge based real-time healthcare support system. In Pethuru Raj and Preetha Evangeline, editors, *The Digital Twin Paradigm for Smarter Systems and Environments: The Industry Use Cases*, volume 117 of *Advances in Computers*, pages 339–368. Elsevier, 2020.

[14] Stephan Trenn. Multilayer perceptrons: Approximation order and necessary number of hidden units. *IEEE transactions on neural networks / a publication of the IEEE Neural Networks Council*, 19:836–44, 06 2008.

[15] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *ArXiv*, abs/1710.05941, 2018.

[16] Ε. Σκέντα. ΠΡΟΣΕΓΓΙΣΗ ΚΑΙ ΑΝΑΠΑΡΑΣΤΑΣΗ ΔΕΔΟΜΕΝΩΝ ΜΙΑΣ ΔΙΑΣΤΑΣΗΣ ΜΕ ΣΥΝΔΥΑΣΜΟ MLP ΚΑΙ RBF ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ. Master's thesis, Πανεπιστήμιο Ιωαννίνων, Ιούλιος 2018.

[17] D.Randall Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451, 2003.

[18] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[19] Benjamin Lindemann, Timo Müller, Hannes Vietz, Nasser Jazdi, and Michael Weyrich. A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, 99:650–655, 2021. 14th CIRP Conference on Intelligent Computation in Manufacturing Engineering, 15-17 July 2020.

[20] Kanchan M.Tarwani and Swathi Edem. Survey on recurrent neural network in natural language processing. *International Journal of Engineering Trends and Technology*, 48:301–304, 06 2017.

[21] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. 02 2014.

[22] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. Recent advances in recurrent neural networks. 12 2017.

[23] Greg Van Houdt, Carlos Mosquera, and Gonzalo Nápoles. A review on the long short-term memory model. *Artificial Intelligence Review*, 53, 12 2020.

[24] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[25] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. *Automated Machine Learning - Methods, Systems, Challenges*. 01 2019.

[26] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *J. Mach. Learn. Res.*, 20:55:1–55:21, 2019.

[27] Gerardo Beni. *Swarm Intelligence*, pages 791–818. Springer US, New York, NY, 2020.

[28] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Positive feedback as a search strategy. *Tech rep., 91-016, Dip Elettronica, Politecnico di Milano, Italy*, 04 1999.

[29] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2):243–278, 2005.

[30] James Kennedy and Russell C. Eberhart. Particle swarm optimization. *Proceedings of ICNN'95 - International Conference on Neural Networks*, 4:1942–1948 vol.4, 1995.

[31] Adham Atyabi and Sepide Samadzadegan. *Particle Swarm Optimization : A Survey*, pages 167 –179. 01 2011.

[32] Geoffrey Miller, Peter Todd, and Shailesh Hegde. Designing neural networks using genetic algorithms. pages 379–384, 01 1989.

[33] Martin Wistuba, Ambrish Rawat, and Tejaswini Pedapati. A survey on neural architecture search. *ArXiv*, abs/1905.01392, 2019.

[34] Edvinas Byla and Wei Pang. *DeepSwarm: Optimising Convolutional Neural Networks Using Swarm Intelligence*, pages 119–130. 01 2020.

[35] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *J. Artif. Int. Res.*, 4(1):237–285, may 1996.

[36] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *ArXiv*, abs/1611.01578, 2017.

[37] M. Dorigo and L.M. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[38] George Kyriakides and Konstantinos G. Margaritis. An introduction to neural architecture search for convolutional networks. *CoRR*, abs/2005.11074, 2020.

[39] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Understanding and simplifying one-shot architecture search. In *ICML*, 2018.

[40] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. A comprehensive survey of neural architecture search: Challenges and solutions. *ACM Comput. Surv.*, 54(4), may 2021.

[41] Renqian Luo, Fei Tian, Tao Qin, Enhong Chen, and Tie-Yan Liu. Neural architecture optimization. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 7827–7838, Red Hook, NY, USA, 2018. Curran Associates Inc.

[42] Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *IJCAI*, 2015.

[43] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[44] Yann LeCun and Corinna Cortes. The mnist database of handwritten digits. 2005.

[45] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

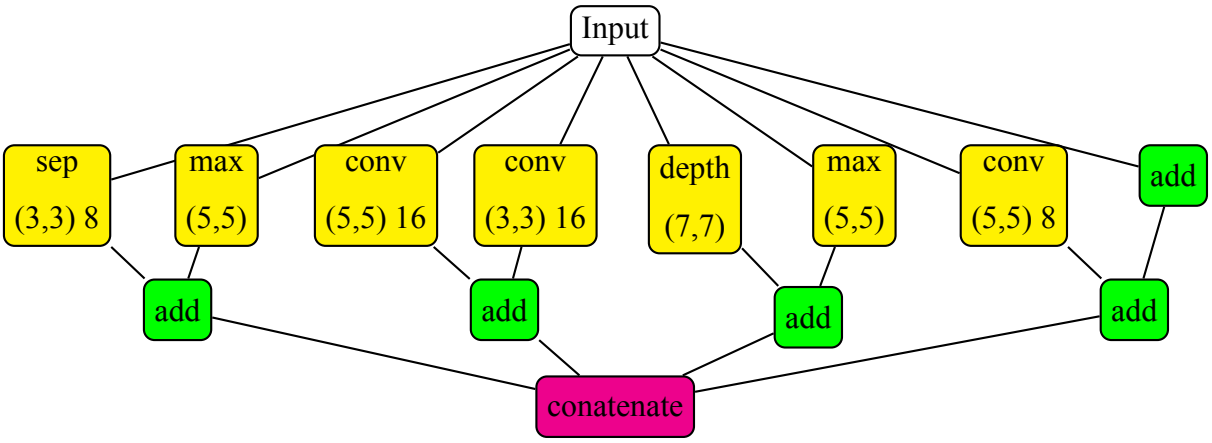
[46] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

ΠΑΡΑΡΤΗΜΑΤΑ

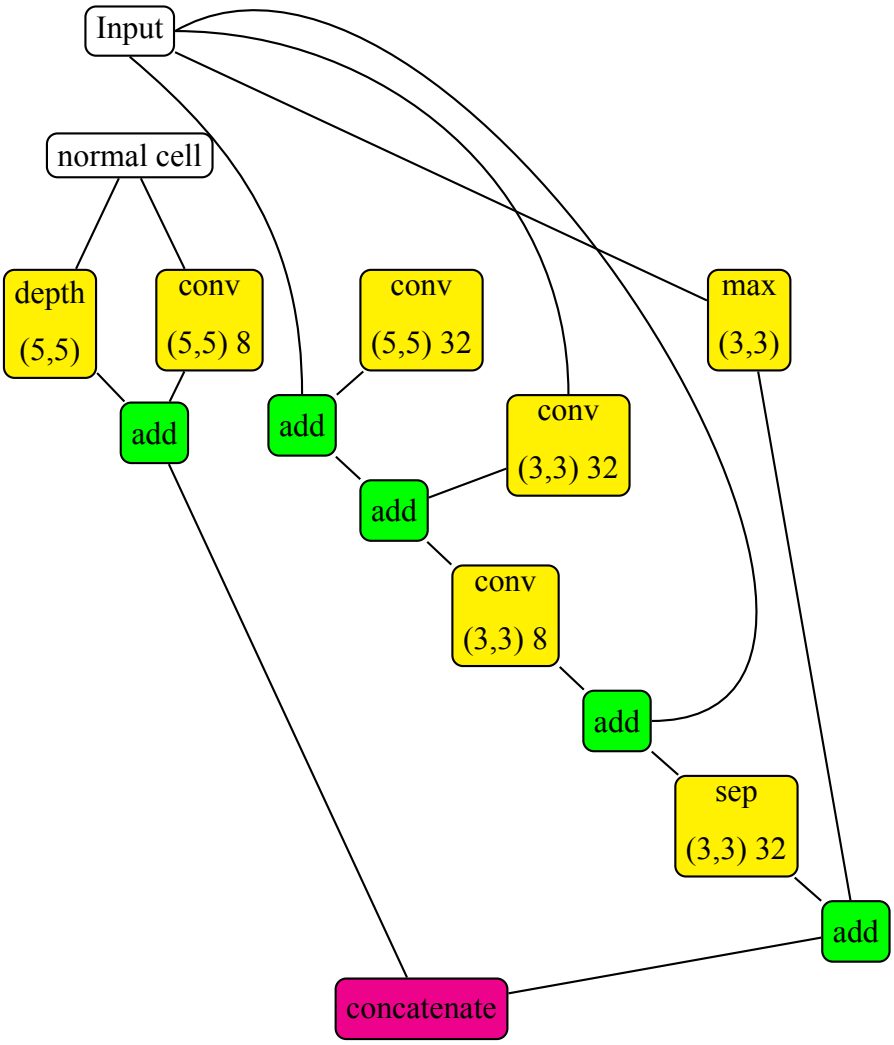
Παράρτημα Α

Καλύτερες αρχιτεκτονικές

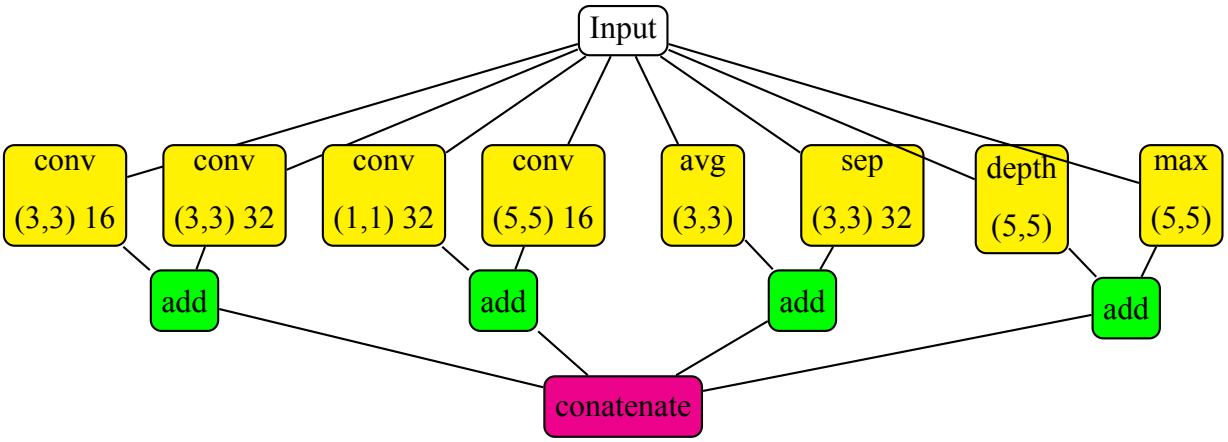
Στο παρόν παράρτημα παρέχονται οι καλύτερες αρχιτεκτονικές κελιών που προέκυψαν από τον AntCell για όλα τα κελιά.



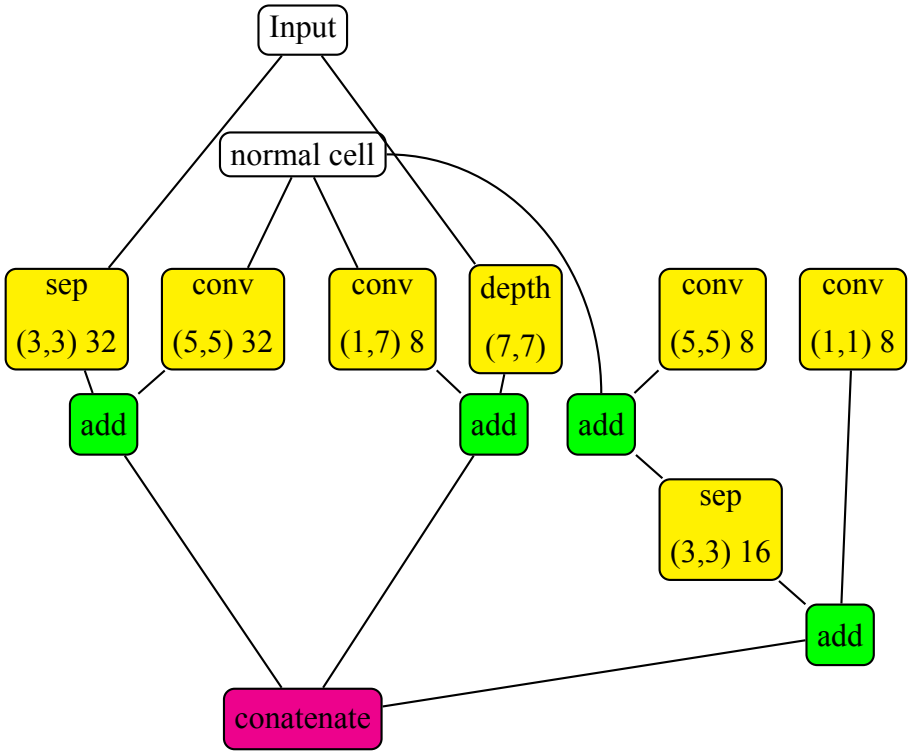
Σχήμα Α.1: Το καλύτερο normal cell που προέκυψε για το MNIST



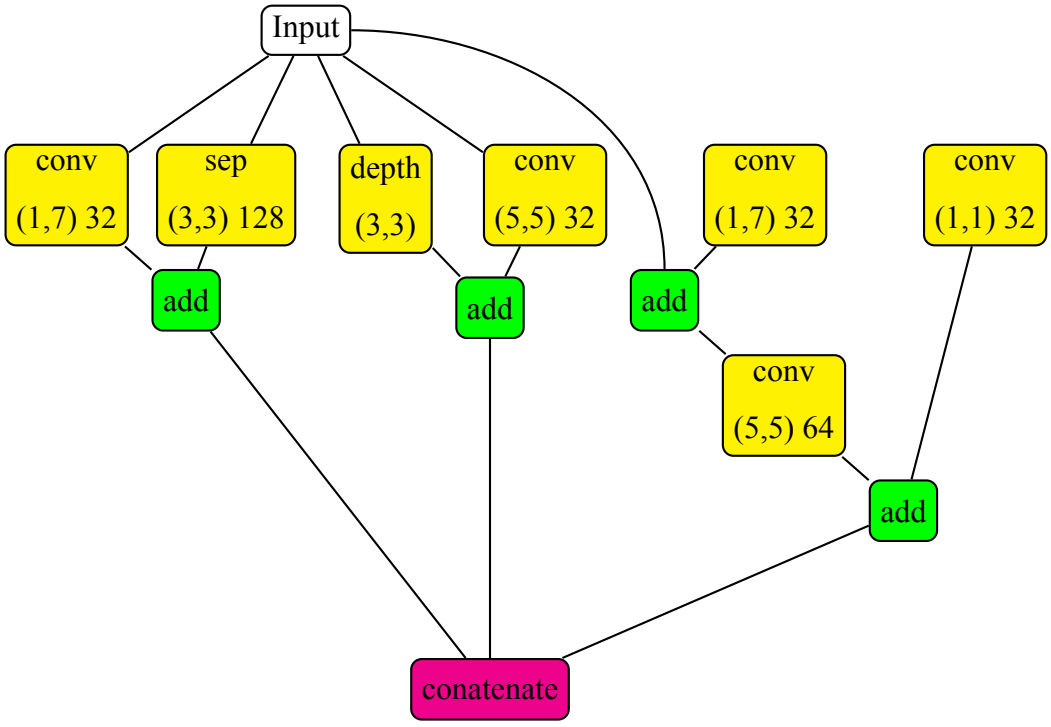
Σχήμα Α.2: Το καλύτερο reduction cell για το MNIST



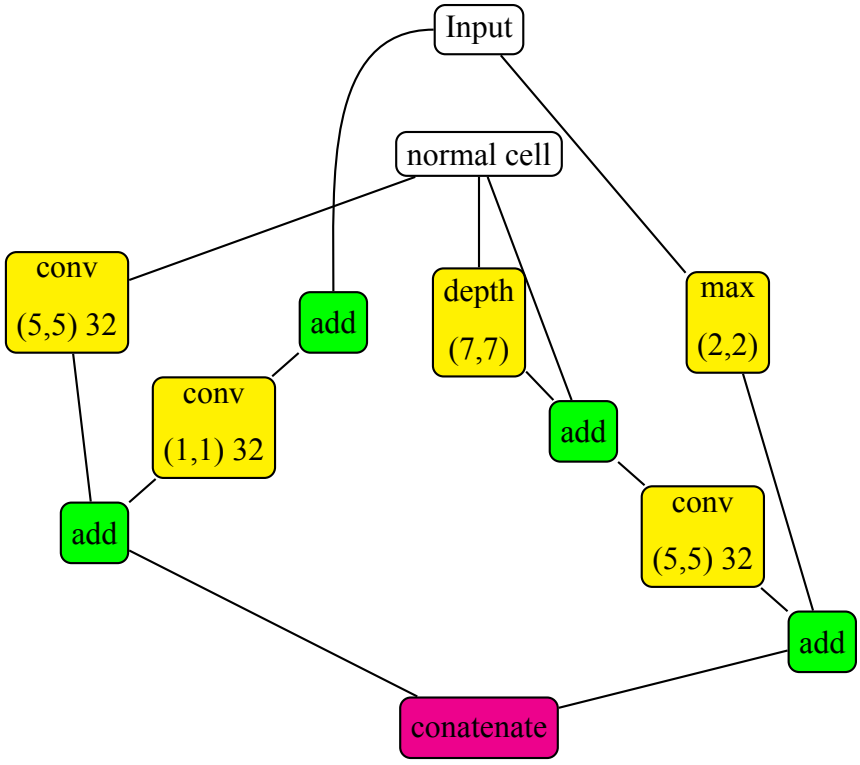
Σχήμα Α.3: Το καλύτερο normal cell για το Fashion-MNIST



Σχήμα Α.4: Το καλύτερο reduction cell για το Fashion-MNIST



Σχήμα Α.5: Το καλύτερο normal cell για το Cifar-10



Σχήμα Α.6: Το καλύτερο reduction cell για το Cifar-10

Παράρτημα Β

Λεπτομέρειες αρχιτεκτονικών και εκπαίδευσης τους

Σε αυτό το παράρτημα θα περιγράψουμε βασικές λεπτομέρειες των αρχιτεκτονικών που δημιουργήθηκαν από τον AntCell, καθώς επίσης και τις παραμέτρους που χρησιμοποιήθηκαν για την εκπαίδευση τους.

Β.1 Αρχιτεκτονικές

Μετά από κάθε Convolutional, Separable ή Depthwise κόμβο χρησιμοποιείται μία ReLU συνάρτηση ενεργοποποίησης, η οποία προσθέτει μη γραμμικότητα στο ΤΝΔ που προέκυψε. Μετά από αυτήν τοποθετείται ένα Batch Normalization στρώμα, το οποίο κανονικοποιεί τις εισόδους των επόμενων στρωμάτων και έτσι [43]

- Δεν επηρεάζεται σημαντικά η επίδοση της αρχιτεκτονικής από την αρχικοποίηση των μεταβλητών.
- Μειώνεται ο αριθμός των επαναλήψεων που απαιτούνται για εκπαίδευση του δικτύου, ενώ δύναται να χρησιμοποιηθεί μεγαλύτερος ρυθμός εκμάθησης.
- Το ΤΝΔ είναι σε θέση να γενικεύει καλύτερα σε άγνωστα δεδομένα και έτσι να μειώνεται η ανάγκη για χρήση dropout.

Επίσης εαν δύο είσοδο ενός στρώματος πρόσθεσης έχουν διαφορετικό αριθμό φίλτρων τότε τοποθετούνται κατάλληλα $1x1$ conv στρώματα, ενώ αν οι διαστάσεις δε συμφωνούν τότε τοποθετούνται ZeroPadding2D.

Β.2 Εκπαίδευση

Κάθε ΤΝΔ της μεθόδου μας εκπαιδεύεται χρησιμοποιώντας τον Adam για ρυθμό εκμάθησης ίσο με 0.001. Επίσης για τη μείωση των διαστάσεων από το reduction cell χρησιμοποιούνται συνελικτικά στρώματα (3,3) με $strides = 2$.