ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ Η/Υ

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΠΡΟΒΛΕΨΗ ΑΠΟΔΟΣΗΣ ΚΥΚΛΩΜΑΤΟΣ ΧΡΗΣΙΜΟΠΟΙΩΝΤΑΣ DEEP LEARNING

Πολίτης Ευάγγελος

Επιβλέποντες: **Γ. Σταμούλης, Φ. Πλέσσας, Δ. Μπαργιώτας**

Βόλος, **Σεπτέμβριος** 2019

UNIVERSITY OF THESSALY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

UNIVERSITY OF THESSALY

PREDICTION OF CIRCUIT PERFORMANCE USING DEEP LEARNING

By:

Politis Evangelos

Supervisors: G. Stamoulis, F. Plessas, D. Bargiotas

Volos, September 2019

Στην οικογένεια & τους φίλους μου

Ευχαριστίες

Με την περάτωση της διπλωματικής μου εργασίας στα πλαίσια της ολοκλήρωσης των προπτυχιακών σπουδών μου θα ήθελα να ευχαριστήσω τους επιβλέποντες καθηγητές μου, κ. Σταμούλη Γεώργιο, τον κ. Πλέσσα Φώτιο, καθώς και τον κ. Μπαργιώτα Δημήτριο για την εμπιστοσύνη που μου έδειξαν.

Επίσης θα ήθελα να ευχαριστήσω θερμά τον Αντωνιάδη Χαράλαμπο για την καθοδήγηση και την άψογη συνεργασία που είχαμε, αλλά και για την βοήθεια του σε όλα τα στάδια της εργασίας.

Τέλος οφείλω ένα μεγάλο ευχαριστώ στην οικογένεια και τους φίλους μου που για την υποστήριξη και την ανεκτίμητη βοήθεια που μου προσέφεραν τόσο κατά την διάρκεια των σπουδών μου, όσο και κατά την εκπόνηση της διπλωματικής εργασίας.

Πρόβλεψη απόδοσης κυκλώματος χρησιμοποιώντας Deep Learning

Περίληψη

Η παρούσα εργασία εστιάζει στην αντιμετώπιση του προβλήματος υπολογισμού χειρότερης καθυστέρησης κυκλώματος chip κατά το σχεδιασμό. Λόγω της ολοένα και μικρότερης τεχνολογίας κατασκευής transistor εμφανίζονται και περισσότερα σφάλματα λόγω κατασκευαστικών αστοχιών. Παρατηρείται μεταβλητότητα στις παραμέτρους κατασκευής. Αυτό που μπορούμε να κάνουμε, είναι να ποσοτικοποιήσουμε τη μεταβλητότητα κατασκευής και να βρούμε την καθυστέρηση των κυκλωμάτων. Με αυτόν τον τρόπο θα κατασκεύασουμε λειτουργικά και αξιόπιστα chip. Άρα θα κάνουμε μία στατιστική ανάλυση, που με τις συμβατικές μεθόδους δεν μπορεί να ανταποκριθεί στο ολοένα και αυξανόμενες παραμέτρους, χρησιμοποιώντας Deep Learning.

ΠΕΡΙΕΧΟΜΕΝΑ

Κεφάλαιο 1

1.1 Περιγραφή Του Προβλήματος.....9

Κεφάλαιο 2: Neural Networks

2.1 **Εισαγωγή**.....11

2.2 Λειτουργία ένος Neural Network**.....**.....12

2.3 **Εκπαίδευση ενός** Neural Network**.....**14

2.4 **Τύποι** Neural Network.....14

2.5 Deep Learning**.....**..18

Κεφάλαιο 3: Ανάλυση του αλγορίθμου

3.1 Εισαγωγή.....19

3.2 **Δεδομένα εισόδου.....20**

3.3 Επεξεργασία δεδομένων.....20

3.4 Ανάπτυξη του Deep Learning **δικτύου.....21**

3.5 Χρήση του μοντέλου που υλοποιήθκε.....23

Κεφάλαιο 4: Συμπεράσματα

4.1 Συμπεράσματα.....25

Κεφάλαιο 5: Βιβλιογραφία

5.1 **Βιβλιογραφία.....26**

ΚΕΦΑΛΑΙΟ 1

ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

1.1 Εισαγωγή

Κατά τη διαδικασία παραγωγής chip σε κάτω από 20nm παρατηρείται μία αυξανόμενη επιβάρυνση στη στατική ανάλυση χρονισμού, προκειμένου να διασφαλιστεί η αξιόπιστη λειτουργία του. Οι σχεδιαστές chip εφαρμόζουν ανάλυση cornered-based και προσθέτουν περιθώρια ασφάλειας στις σχεδιαστικές παραμέτρους προκειμένου να ληφθούν υπόψη οι επιπτώσεις της διακύμανσης της διαδικασίας χρονισμού. Ωστόσο, οι προαναφερθείσες τεχνικές είτε είναι πολύ αργές, καθώς ο αριθμός των παραμέτρων σχεδιασμού πολλαπλασιάζεται με την ενσωμάτωση περισσότερων συστατικών σε ένα chip ή ανακριβής λόγω της παραδοχής ότι η χειρότερη περίπτωση καθυστέρησης βρίσκεται στις ακραίες τιμές των σχεδιαστικών παραμέτρων. Η εργασία αποτελεί επέκταση μιας προσφατα δημοσιευμένης έρευνας που βασίζεται στη Θεωρία Ακραίων Τιμών(EVT), η οποία προσπαθεί για να εκτιμήσει τη χειρότερη περίπτωση καθυστέρησης των κυκλωμάτων VLSI υπό διακύμανσης των τιμών στις παραμέτρους πύλης / διασύνδεσης.

Συγκεκριμένα γίνεται εστίαση στην εκτίμηση της μέγιστης καθυστέρησης. Θα εργαστούμε πάνω στην εργασία [1] στον διαχωρισμό των κυκλωμάτων και των αντίστοιχων καθυστερήσεων τους σε σχέση με ένα σημείο "*threshold u*". Ταξινομούμε τα κυκλώματα βάσει των καθυστερήσεων. Όποια συστήματα βρίσκονται μετά από το *u* λέμε ότι βρίσκονται στην ουρά του δείγματος, ενώ τα υπόλοιπα στο σώμα. Στην περίπτωση μας το *u* αποτέλεσαι το σημείο που βρίσκεται στο 97% των συνολικών καθυστερίσεων. Άρα όσα κυκλώματα είχαν καθυστερήσεις μεγαλύτερες από εκείνη που βρισκόταν στο *u* ήταν οι θεμιτές για προσομοίωση.

Ο αλγόριθμος Deep Learning που αναπτύξαμε μας βοηθάει, έχοντας εκπαιδευτεί σε υπάρχοντα δεδομένα, να κάνουμε μία καλή

εκτίμηση της καθυστέρησης ενός νέου κυκλώματος. Δηλαδή εάν αυτή υπαρβαίνει το *u*, τότε μπορούμε να το χρησιμοποιήσουμε για προσομοίωση, ή το αντίθετο.

Κατά συνέπεια έχουμε σημαντική ελαχιστοποίηση του χρόνου στην αναζήτηση των επιθυμητών κυκλωμάτων, καθώς και χρήση κυκλωμάτων για κυκλώματα με μεγάλο αριθμό παραμέτρων. Στην εργασία εργαστήκαμε σε κυκλώματα με 10 και 245 παραμέτρους.

ΚΕΦΑΛΑΙΟ 2

NEURAL NETWORKS

2.1 **Εισαγωγή**

Ένα Νευρωνικό Δίκτυο (Neural Network) είναι ένα πρόγραμμα υλοποιημένο σε ηλεκτρονικό υπολογιστή που λειτουργεί παρόμοια με τον ανθρώπινον εγκέφαλο. Ο στόχος των νευρωνικών δικτύων είναι να εκτελέσουν τις λειτουργίες που μπορεί να κάνει ο εγκέφαλός μας όπως η επίλυση προβλημάτων και η δυνατότητα εκμάθησης.

Το πρώτο νευρωνικό δίκτυο αναπτύχθηκε από τους Warren McCulloch και τον Walter Pitts το 1943, που όμως δεν κατάφεραν να λειτουργήσουν λόγω της τεχνολογίας της εποχής. Το 1954, ωστόσο, οι Wesley Clark και Belmont Farley έφεραν επιτυχώς σε λειτουργία το πρώτο απλό νευρωνικό δίκτυο.

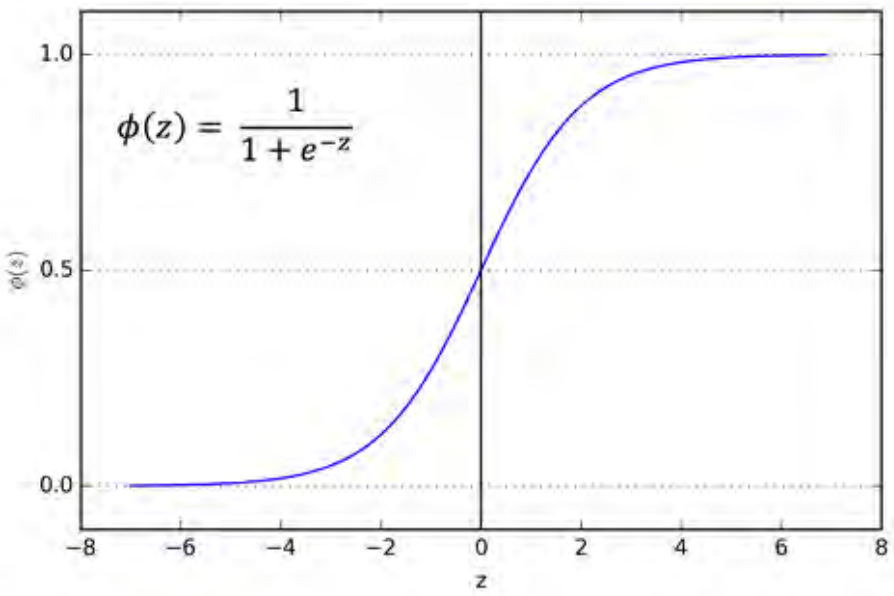
Η πρωταρχική έφεση των νευρωνικών δικτύων είναι η ικανότητά τους να μιμούνται τις δεξιότητες αναγνώρισης μοτίβου του εγκεφάλου. Τα νευρωνικά δίκτυα έχουν χρησιμοποιηθεί για ένα ευρύ φάσμα εφαρμογών, όπως για την πρόβλεψη των αποτελεσμάτων των επενδυτικών αποφάσεων, μπορούν να βρουν μοτίβα σε χειρόγραφα και μπορούν ακόμη και να ανιχνεύσουν εκτάσεις γης για ανωμαλίες, κάνοντας τα να μπορούν να επισημάνουν αντικείμενα όπως νάρκες ή βόμβες.

2.2 Λειτουργία ένος Neural Network

Φανταστείτε ένα νευρωνικό δίκτυο ως ένα μαύρο κουτί το οποίο παίρνει εισόδους, όπως τα ερεθίσματα των αισθητήρων ενός αυτοδηγούμενου αυτοκινήτου και έχει για εξόδους στοιχεία που καθορίζουν τον χειρισμό του. Το ίδιο το νευρωνικό δίκτυο έχει πολλές μικρές μονάδες που ονομάζονται νευρώνες (Neurons), οι οποίοι ομαδοποιούνται σε επίπεδα (Layers). Τα επίπεδα είναι στήλες που με τη σειρά τους συνδέονται μέσω των νευρώνων τους.

Κάθε νευρώνας συνδέεται με τον νευρώνα κάποιου άλλου επιπέδου με συνδέσεις που αποκαλούνται βεβαρημένες (weighted connections). Στις συνδέσεις αυτές τοποθετείται μία πραγματική αριθμητική τιμή (βάρος). Έπειτα η τιμή που έχει ο νευρώνας πολλαπλασιάζεται με το βάρος της σύνδεσης (νευρώνες από επίπεδο σε επίπεδο). Στο άθροισμα όλων των συνδεμένων νευρώνων προστίθεται η τιμή “bias”. Το αποτέλεσμα αυτής της πράξης, που θυμίζει ένα πολυώνυμο πρώτου βαθμου ¨a.W + b¨, αποτελεί την είσοδο στην συνάρτηση ενεργοποίησης (activation function) [f(x)] (Εικόνα 1), η οποια μετασχηματίζει μαθηματικά την τιμή αυτή και την αναθέτει στο επόμενο επίπεδο. Αυτή η διαδικασία γίνεται κατά μήκος ολόκληρου του νευρωνικού δικτύου.

Εικόνα 1: Activation Function [f(x)] -- Sigmoid



Ουσιαστικά, το δίκτυο είναι σαν ένα φίλτρο που χρησιμοποιεί όλες τις πιθανότητες, και ο υπολογιστής να είναι σε θέση να μπορεί να βρει τη σωστή απάντηση. Η πραγματική πρόκληση είναι να βρεθούν τα σωστά βάρη (τιμή νευρώνων) προκειμένου να υπολογιστούν τα σωστά αποτελέσματα. Η εύρεση των σωστών βαρών γίνεται μέσω της μηχανικής μάθησης (Machine Learning), και γι 'αυτό τα νευρικά δίκτυα είναι τόσο αλληλένδετα με την εξέλιξη της τεχνητής νοημοσύνης.

Υπάρχουν περιπτώσεις όπου το δίκτυο είναι λάθος και εξάγει λάθος απάντηση. Το δίκτυο είναι πάντοτε σε θέση να είναι σωστό, επειδή ψάχνει για συγκεκριμένα χαρακτηριστικά προκειμένου να εξάγει μια απάντηση.

Εάν ένα αντικείμενο μοιάζει με άλλο αντικείμενο, το δίκτυο θα μπορούσε να "μπερδευτεί" και να εξάγει μια εσφαλμένη απάντηση. Για να αποφευχθεί αυτό, μπορούμε πρώτα να εξοπλίσουμε το δίκτυο με κάποιο είδος μηχανισμού «προς τα πίσω» διάδοσης, που είναι γνωστός ως αλγόριθμος Back-Propagation. Χρησιμοποιώντας αυτόν τον αλγόριθμο, το δίκτυο μπορεί να επιστρέψει και να «ελέγξει ξανά» το δίκτυό του για να βεβαιωθεί ότι όλα τα biases είναι σωστά, όλες οι συνδέσεις έχουν τα σωστά βάρη για την εφαρμογή Machine Learning.

Δεύτερον, μπορούμε να κάνουμε το νευρικό δίκτυο επαναλαμβανόμενο (Recurrent Neural Network ή RNN), που περιλαμβάνει σήματα που προχωρούν και στις δύο κατευθύνσεις καθώς και μεταξύ των Layers. Τα RNN είναι συνήθως σχεδιασμένα για να αναγνωρίζουν τα διαδοχικά χαρακτηριστικά των δεδομένων και χρησιμοποιούν μοτίβα για να προβλέψουν το επόμενο πιθανό σενάριο.

2.3 Εκπαίδευση ενός Neural Network

Όπως αναφερθήκαμε στην εισαγωγή, τα νευρωνικά δίκτυα μιμούνται τον ανθρώπινο εγκέφαλο και σε αυτή την ενότητα θα εξηγήσουμε τον τρόπο με τον οποίο εκπαιδεύονται.

Κατά την εκπαίδευση του δικτύου χρησιμοποιείται η λεγόμενη εποπτευόμενη εκπαίδευση (Supervised Learning). Ουσιαστικά για κάθε παράδειγμα που χρησιμοποιούμε, γνωρίζουμε τόσο την τιμή εισόδου του, αλλά και την τιμή εξόδου. Επαναλαμβάνουμε τη διαδικασία μέχρι τα αποτελέσματα εξόδου του νευρωνικού δικτύου να είναι ικανοποιητικά και έπειτα μπορούμε να προσθέσουμε νέα παραδείγματα προς εκπαίδευση.

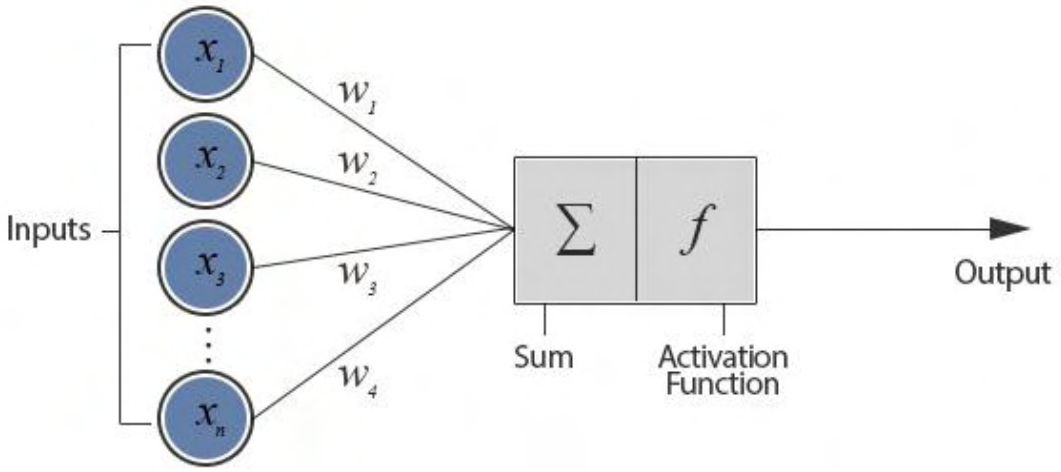
Επίσης υπάρχει και η μη εποπτευόμενη εκπαίδευση (Unsupervised Learning), όπου το νευρωνικλο δίκτυο δέχεται εισόδους και δίνει μοτίβα που ανακαλύτει ως εξόδους χωρίς να είναι ξεκάθαρο τι πρέπει να αναζητήσουμε. Τέτοια νευρωνικά δίκτυα χρησιμοποιούνται στην εξόρυξη δεδομένων(data mining).

2.4 Τύποι Neural Network

Σε αυτό το σημείο θα ήταν σημαντικό να διαχωρίσουμε και να πούμε μερικά λόγια για τους τύπους νευρωνικών δικτύων που είναι περισσότερο διαδεδομένοι, καθώς και μερικά σχόλια και χρήσεις για τον καθένα.

Έχουμε το Feedforward Neural Network, το οποίο είναι και το πιο απλό. Κατά τη χρήση του τα δεδομένα μας ‘προχωρουν’ μόνο προς μία κατεύθυνση, από την είσοδο προς την έξοδο. Δεν υπάρχει κάποιου είδους διάδοση δεδομένων προς τα πίσω, και είναι δύσκολη η διόρθωση μετά από λάθος αποτελέσματα(Εικόνα 2).

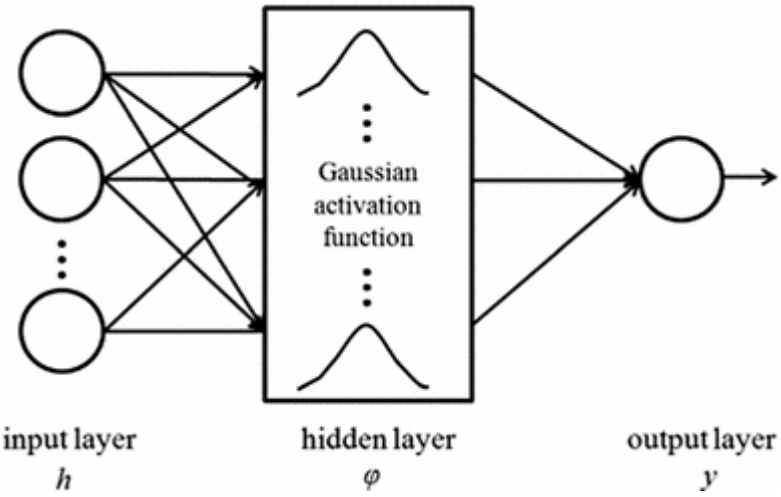
Εικόνα 2.Feedforward Neural Network



Ο δεύτερος τύπος νευρωνικών δικτύων είναι τα Radial Basis Function Neural Networks (RBF), τα οποία λειτουργούν με συναρτήσεις που σχετίζονται με την απόσταση δεδομένων πάνω στο επίπεδο από κάποιο κέντρο. Οι λειτουργίες RBF έχουν δύο layers. Το πρώτο όπου τα χαρακτηριστικά συνδυάζονται με τη λειτουργία Radial Basis στο εσωτερικό layer και στη συνέχεια λαμβάνεται υπόψη η έξοδος αυτών των χαρακτηριστικών, όσο υπολογίζεται η ίδια έξοδος στην επόμενη συνάρτηση(Εικόνα 3). Χαρακτηρηστικό τους είναι η activation function τους που είναι η Γκαουσιανή συνάρτηση με τύπο:

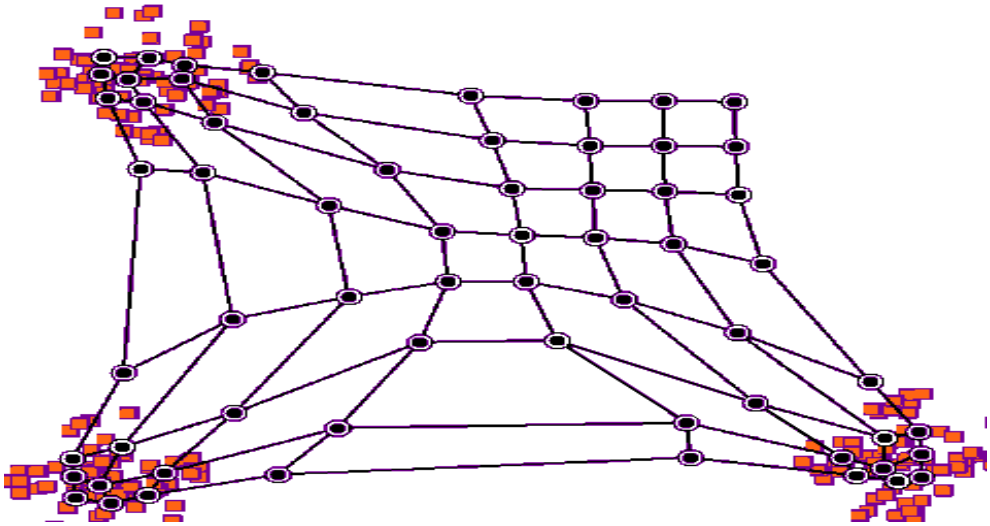
$$\phi(v_i) = \exp\left(-\frac{\|v_i - c_i\|^2}{2\sigma^2}\right)$$

Εικόνα 3. Radial Basis Function Neural Network



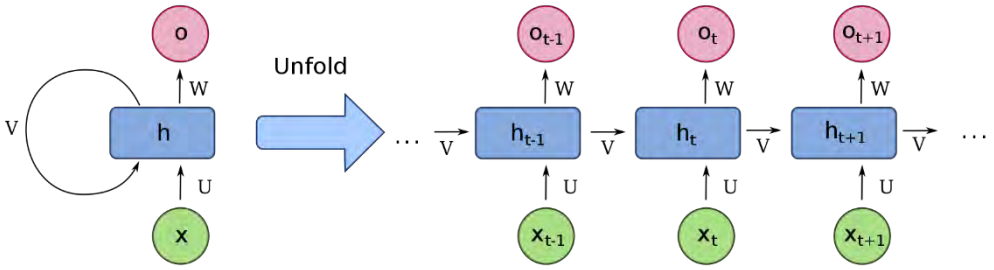
Ο τρίτος τύπος νευρωνικών δικτύων είναι ο αυτορρυθμιζόμενος χάρτης του Kohonen, ο οποίος είναι ένας τύπος τεχνητού νευρωνικού που εκπαιδεύεται χρησιμοποιώντας Unspurvised Learning και παράγει μία διακριτική αντιπροσώπευση χαμηλότερης διάστασης, που ονομάζεται χάρτης (map). Διαφέρει από άλλα νευρωνικά δίκτυα, καθώς εφαρμόζει ανταγωνιστική μάθηση(Competitve Learning) σε αντίθεση με τη μάθηση διόρθωσης σφαλμάτων, και χρησιμοποιεί τη λειτουργία της γειτονιάς για να διατηρεί τις τοπολογικές ιδιότητες του χώρου εισόδου(Εικόνα 4). Χαρακτηριστικό παράδειγμα είναι τα βιολογικά μοντέλα μορφωγένεσης.

Εικόνα 4. Self-Organizing Feature Map



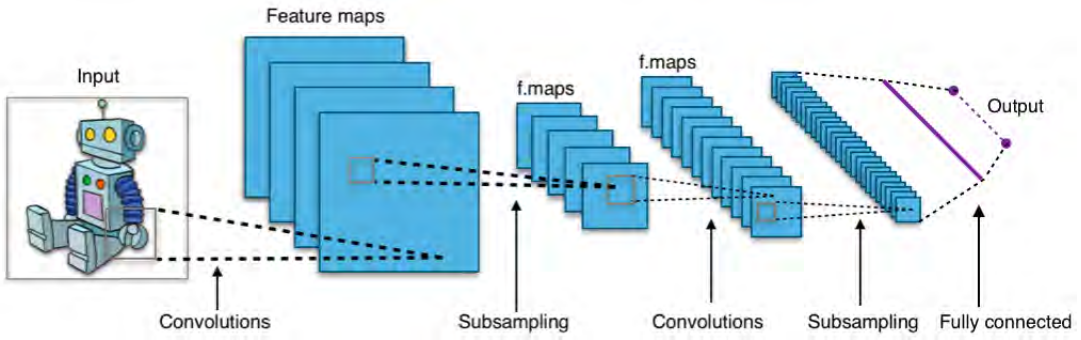
Η επόμενη κατηγορία είναι τα επαναλαμβανόμενα νευρωνικά δίκτυα (Recurrent Neural Network ή RNN), τα οποία αποτελούν μία τάξη τεχνητών νευρωνικών δικτύων όπου οι συνδέσεις μεταξύ των κόμβων σχηματίζουν ένα κατευθυνόμενο γράφημα κατά μήκος μιας χρονικής αλληλουχίας. Αυτό τους δίνει τη δυνατότητα να παρουσιάζουν μία χρονική δυναμική συμπεριφορά, δηλαδή έχουν εσωτερική μνήμη για να μπορούν να επεξεργάζονται τις ακολουθίες εισόδων(Εικόνα 5). Τα συναντάμε συχνά σε εργασίες αναγνώρισης ομιλίας.

Εικόνα 5. Basic RNN



Μία ακόμη μεγάλη κατηγορία αποτελούν τα Convolutional Neural Network(ConvNet/CNN), τα οποία είναι νευρωνικά δίκτυλα Deep Learning που μπορούν να δεχτούν σαν είσοδο μία εικόνα, να αναδείξουν τη σημασία σε διάφορα αντικείμενά της και είναι σε θέση να διαφοροποιήσουν το ένα από το αλλο. Η προεπεξεργασία που απαιτείται σε ένα CNN είναι αρκετά λιγότερη σε σύγκριση με άλλους αλγόριθμους. Το δίκτυο είναι σε θέση να μάθει τα φίλτρα που σε παραδοσιακούς αλγόριθμους ήταν χειροκίνητοι. Αυτή η ανεξαρτησία από τις προηγούμενες γνώσεις και την αθρώπινη προσπάθεια στο σχεδιασμό αποτελεί σημαντικό πλεονέκτημα τους(Εικόνα 6). Έχουν εφαρμογές στην αναγνώριση εικόνων και βίντεο, ταξινόμηση εικόνων, ανάλυση ιατρικών εικόνων και επεξεργασία φυσικής γλώσσας.

Εικόνα 6. Typical CNN architecture



2.5 Deep Learning

Η βαθιά εκμάθηση(Deep Learning) επέτρεψε πολλές πρακτικές εφαρμογές του Machine Learning και κατ 'επέκταση του συνολικού πεδίου της Τεχνητής Νοημοσύνης(AI). Το Deep Learning αναλύει λειτουργίες με τρόπους που καθιστούν την πιθανή βοήθεια απο τα μηχανήματα, ακόμη καλύτερα. Τα αυτοκίνητα χωρίς οδηγό, η καλύτερη προληπτική υγειονομική περίθαλψη, ακόμη καλύτερες συστάσεις ταινιών. Το AI είναι το παρόν και το μέλλον. Με τη βοήθεια του Deep Learning, το AI μπορεί να φτάσει σε εκείνη την κατάσταση επιστημονικής φαντασίας που τόσο πολύ φανταζόμασταν.

ΚΕΦΑΛΑΙΟ 3

ΑΝΑΛΥΣΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ

3.1 Εισαγωγή

Όπως είπαμε και στην εισαγωγή, στόχος της εργασίας ήταν να βρούμε έναν τρόπο, κατά τον οποίο να μπορούμε να καταλαβαίνουμε γρήγορα εάν ένα κύκλωμα έχει καθυστέρηση μεγαλύτερη εκείνης που βρίσκεται στο σημείο *u*, δηλαδή μεγαλύτερη από το 97% των καθυστερήσεων που χρησιμοποιήσαμε από τα δεδομένα που αργότερα θα τροφοδοτήσουμε το νευρωνικό μας δίκτυο για εκπαίδευση. Άρα να ελέγξουμε εάν αυτό μπορεί να χρησιμοποιηθεί για προσομοίωση.

Για την επίλυση του προβλήματος της εργασίας αναπτύξαμε ένα αλγόριθμο Deep Learning, ο οποίος είναι σε θέση να διαχωρίζει κατάλληλα την τελική καθυστέρηση των ομάδων λογικών πυλών σε δύο κατηγορίες. Η μία κατηγορία είναι εκείνη των μικρών καθυστερήσεων και δεύτερη των μεγάλων, με σημείο διαχωρισμού την καθυστέρηση στο σημείο *u*. Οπότε μιλάμε για ένα Binary Classification πρόβλημα. Έτσι δοσμένου ενός νέου κυκλώματος να μπορούμε αντιλειφθούμε εξ' αρχής σε ποια κατηγορία ανήκει. Για τη δημιουργία του αλγόριθμου επιλέξαμε τη γλώσσα προγραμματισμού Python 3.0.

3.2 **Δεδομένα Εισόδου**

Τα δεδομένα εισόδου που είχαμε στη διάθεση μας ήταν επιμέρους πλάτη από transistors, ως είσοδοι, καθώς και την συνολική τους καθυστέρηση ανά συστοιχία ως έξοδο. Για την κάθε συστοιχία πήραμε δεδομένα από πολλές δοκιμές (trials). Τα δεδομένα μας δημιουργήθηκαν μέσα στον ίδιο Η/Υ. Τα τοποθετήσαμε σε τέσσερα αρχεία, τα οποία είναι:

c17_gate_widths_1000_MC: για κύκλωμα με 10 παραμέτρους

c17_N22_ATfall_1000_MC : οι αντίστοιχες καθυστερήσεις τους

c432_gate_widths_100_MC: για κύκλωμα με 245 παραμέτρους

c432_N223_ATfall_100_MC: οι αντίστοιχες καθυστερήσεις τους.

3.3 **Επεξεργασία Δεδομένων**

Πριν τροφοδοτήσουμε τα δεδομένα μας στο νευρωνικό δίκτυο Deep Learning έπρεπε αρχικά να τα επεξεργαστούμε, για να μπορούμε να έχουμε μία καλύτερη εικόνα τόσο για εμάς, όσο και μία σωστή ομαδοποίηση για το ίδιο το νευρωνικό που δέχεται σαν εισόδους διανύσματα. Για το λόγο αυτό δημιουργήσαμε έναν πίνακα ο οποίος περιέχει τα πλάτη σε στήλες και σαν τελική στήλη τις τελικές καθυστερήσεις των αντίστοιχων κυκλωμάτων, με κάθε γραμμή να αντιστοιχεί σε ένα trial. Έπειτα ταξινομήσαμε αυτόν τον πίνακα ως προς την τελευταία στήλη, εκείνη δηλαδή με τις τελικές καθυστερήσεις του κάθε trial. Ο αλγόριθμος που κάνει την παραπάνω διαδικασία βρίσκεται στο αρχείο:

Parser.py

Αποτέλεσμα του είναι η δημιουργία ενός νέου αρχείου, των:

SortedData1000.csv & SortedData100.csv.

Το πρώτο για την περίπτωση των 10 παραμέτρων και ένα για την περίπτωση των 245 παραμέτρων. Στη συνέχεια υπολογίσαμε την καθυστέρηση που βρίσκεται στο *u* του ταξινομημένου πίνακα και μετατρέψαμε τις τιμές των καθυστερήσεων σε 0 εάν η καθυστέρηση ήταν <*u* και σε 1 εάν ήταν >*u*, έτσι ώστε δοσμένης μίας συστοιχίας από πλάτη transistor να γνωρίζουμε εάν ανήκει στις μεγαλύτερες καθυστερήσεις ή όχι. Τα αποτελέσματα τα τοποθετήσαμε σε ένα νέο αρχείο για την κάθε περίπτωση με όνομα:

SortedData0-1.csv.

Για την επεξεργασία των δεδομένων μας ώστε να τα φέρουμε στην επιθυμητή μορφή χρησιμοποιήσαμε τις βιβλιοθήκες *Pandas & Numpy* της *Python*.

3.4 **Ανάπτυξη του** Deep Learning **δικτύου**

Όπως είπαμε και στην προηγούμενη ενότητα δημιουργήσαμε ένα πίνακα τον οποίο θα τροφοδοτήσουμε στο Deep Learning δίκτυο, προκειμένου να το εκπαιδεύσουμε. Τόσο για τη δημιουργία του μοντέλου μας όσο κ για την εκπαίδευσή του χρησιμοποιήσαμε τη βιβλιοθήκη *Keras* η οποία έχει αναπτυχθεί από το MIT και περιέχει συναρτήσεις για τη δημιουργία του μοντέλου και την εκπαίδευση του. Η βιβλιοθήκη *Keras* λειτουργεί πάνω στο *Tensorflow* που είναι open-source πλατφόρμα *Machine Learning*.

Τα δεδομένα του πίνακα χωρίστηκαν με τυχαίο τρόπο σε εκείνα που απαιτούνται για να γίνει η εκπαίδευση και σε αυτά που θα χρησιμοποιηθούν για να γίνουν επαληθεύσεις των αποτελεσμάτων. Έτσι από τον πίνακα μας έχουμε δύο κατηγορίες δεδομένων μέσω της συνάρτησης *train_test_split(X,Y)*. Αποτέλεσμα αυτής είναι ο διαχωρισμός των δεδομένων μας στις παρακάτω υποκατηγορίες:

- X_train, Y_train:

Όπου X_train είναι trials με πλάτη transistor και Y_train οι αντίστοιχες καθυστερήσεις τους. Τα δύο δεδομένα θα εκπαιδεύσουν το μοντέλο.

- X_test, Y_test:

  Όπου X_test είναι και πάλι trials με πλάτη και Y_test οι αντίσοιχες καθυστερήσεις τους. Τα δεδομένα χρησιμοποιούνται για να καταλαβαίνουμε την εγκυρότητα του αποτελέσματος του μοντέλου μας.

Αφότου διαχωρίσαμε τα δεδομένα μας δημιουργήσαμε το Deep Learning μοντέλο μας το οποίο αποτελείται από τρία Layers με το πρώτο να έχει 160 νευρώνες, το δεύτερο 80 και το τελευταίο 1, που είναι και η έξοδος. Τα δύο πρώτα Layers χρησιμοποιούν ως Activation Function τη συνάρτηση ReLu με μαθηματικό τύπο:

$y = max(0, \chi)$.

Εφόσον το πρόβλημα είναι Binary Classification τότε η Activation Function του τελευταίου Layer είναι η Sigmoid με μαθηματικό τύπο:

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

Και αυτό που κάνει ουσιαστικά είναι ότι έρχεται από τα προηγούμενα Layers του δικτύου να τα τοποθετεί ανάμεσα σε 0 και 1. Άρα το Classification επιτυγχάνεται όταν έχουμε δύο ομάδες, με τη μία να βρίσκεται κοντά στο μηδέν και την άλλη κοντα στο ένα.

Οπότε με το κάλεσμα της *model.fit()* συνάρτησης η εκπαίδευση ξεκινά.

Στο τέλος της και εφόσον είμαστε ικανοποιημένοι τόσο με το accuracy, όσο και με το loss μπορούμε να αποθηκεύσουμε το μοντέλο και να το καλέσουμε ώστε να το χρησιμοποιήσουμε.

Αφότου το μοντέλο μας τελικά εκπαιδευτεί το αποθηκεύσαμε με τη βοήθεια της βιβλιοθήκης *json* και συγκεκριμένα με τη συνάρτηση *model.to_json()*, η οποία αποθηκεύει την αρχιτεκτονική του εκπαιδευμένου μοντέλου μας στο αρχείο *model_in_json.json*. Δεν αρκεί όμως μόνο αυτό. Πρέπει να αποθηκεύσουμε και τις τελικές τιμές των *weights*. Για να γίνει αυτο χρησιμοποιούμε τη συνάρτηση

model.save_weights(), η οποία αποθηκεύει τις τελικές τιμές των *weights* στο αρχείο *model_weights.h5*.

Για να εκμεταλλευτούμε το νευρωνικό δίκτυο εισάγουμε έναν πίνακα που να έχει της διαστάσεις εισόδου που δέχεται το μοντέλο, δηλαδή ένα μονοδιάστατο διάνυσμα με στήλες όσες τα πλάτη transistor, που αποτελούν και τις παραμέτρους μας, πάνω στα οποία εκπαιδεύτηκε. Σαν έξοδο θα πάρουμε μηδέν ή ένα και έτσι θα έχουμε την πληροφορία που χρειαζόμαστε.

Η εκπαίδευση του αλγορίθμου Deep Learning υλοποιείται στο αρχείο *BinaryTrain.py*.

3.5 **Χρήση του μοντέλου που υλοποιήθηκε**

Εφόσον ολοκληρώθηκε η εκπαίδευση του νευρωνικού μας μοντέλου, πλέον μπορούμε να το χρησιμοποιήσουμε για το δικό μας σκοπό. Δηλαδή να το τροφοδοτήσουμε με χαρακτηριστικά πλάτη από transistors, ως παραμέτρους, στη μορφή διανύσματος και να πάρουμε σαν αποτέλεσμα εάν η καθυστέρηση από το συγκεκριμένο κύκλωμα βρίσκεται ή όχι στην περιοχή τιμών μετά από εκείνη του *u*. Αυτό σημαίνει ότι εάν το μοντέλο μας επιστρέψει 0, τότε δεν το χρειαζόμαστε, ενώ εάν μας επιστρέψει 1 τότε κύκλωμα θα μας είναι χρήσιμο προς προσομοίωση.

Το αρχείο μας που εξετάζει τα παραπάνω είναι το *Final.py*. Σε αυτό καλούμε την αρχιτεκτονική του νευρωνικού δικτύου που δημιουργήσαμε νωρίτερα με τη συνάρτηση *json.load()* και επίσης καλούμε και τις τελικές τιμές *weights* που είχαμε αποθηκεύσει με τη συνάρτηση *load_weights("model_weights.h5")*. Τα δύο παραπάνω τα τοποθετούμε σε ένα *model* που έχει τόσο την επιθυμητή αρχιτεκτονική, όσο και τα *weights* που είχαμε υπολογίσει.

Έπειτα τροφοδοτούμε εισάγουμε διάνυσμα με το κύκλωμα μας και προσπαθούμε να ελέγξουμε εάν η καθυστέρηση του θα βρίσκεται

πριν ή μετά από την τιμή του *u*. Έτσι μεταφέρουμε το διάνυσμα στη μεταβλητή:

Xnew

και μετά καλούμε τη συνάρτηση:

model.predict(Xnew)

Η διαδικασία που γίνεται είναι να περάσουμε μέσα από το νευρωνικό δίκτυο μας το διάνυσμα *Xnew*.

Αν το αποτέλεσμα είναι 0 τότε θα εμφανιστεί το μήνυμα:

Our Input has a Delay on Body!

Ενώ εάν το αποτέλεσμα είναι 1 θα εμφανιστεί το μήνυμα:

Our Input has a Delay on Tail!

Όπου θεωρούμε *Body & Tail* τα σημεία που βρίσκονται πριν και μετά το σημείο *u*.

ΚΕΦΑΛΑΙΟ 4

ΣΥΜΠΕΡΑΣΜΑΤΑ

4.1 Συμπεράσματα

Τα αποτελέσματα απο τις δοκιμές που κάναμε ήταν στατιστικά σωστά με μικρό σφάλμα <10% . Εάν μπορούσαμε εξάγουμε ένα συμπέρασμα από την εργασία, αυτό θα ήταν ότι ο αλγόριθμος μας είναι σε θέση να υπολογίζει και να διαχωρίζει με σωστό τρόπο τις καθυστερήσεις από ένα νέο κύκλωμα. Αυτό συνεπάγεται ότι είμαστε σε θέση να συλλέξουμε τα κυκλώματα με την μεγαλύτερη καθυστέρηση προκειμένου να χρησιμοποιηθούν σε εργαλία προσομοίωσης κυκλωμάτων για τη δημιουργία *chip*.

Μπορεί, ωστόσο να δεχτεί περαιτέρω βελτιώσεις και επεκτάσεις για ακόμη καλύτερα αποτελέσματα.

Ένα σημείο βελτίωσης θα ήταν να γίνει εκπαίδευση του αλγόριθμου σε μηχάνημα με μεγάλη υπολογιστική ισχύ σε πίνακες, όπως να εμπεριέχει ισχυρές και πολλαπλές *GPUs*. Με αυτόν τον τρόπο θα μπορούσε η αρχιτεκτονική του νευρωνικού να είναι περισσότερο πολύπλοκη και ενδεχομένως να έχουμε αποτελέσματα με ακόμη λιγότερο σφάλμα.

Ακόμα θα μπορούσαμε σε κάποιο άλλο Project να επεκταθεί η εργασία και να συνδιάζει όλες τις επιμέρους αρχιτεκτονικές μοντέλων σε ένα εννιαίο, χωρίς να χρειάζεται η απόλυτη αντιστοιχία για την τροφοδότηση ενός κυκλώματος στο νευρωνικό.

ΚΕΦΑΛΑΙΟ 5

ΒΙΒΛΙΟΓΡΑΦΙΑ

5.1 Βιβλιογραφία

[1] Charalampos Antoniadis, Dimitrios Garyfallou, Nestor Evmorfopoulos, and Georgios Stamoulis, “EVT-based Worst Case Delay Estimation Under Process Variation”, 23 April 2018

[2] https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

[3] https://medium.com/@danqing/a-practical-guide-to-relu-b83ca804f1f7

[4] https://datascienceplus.com/keras-regression-based-neural-networks/

[5] https://en.wikipedia.org/wiki/Recurrent_neural_network

[6] https://en.wikipedia.org/wiki/Self-organizing_map

[7] https://en.wikipedia.org/wiki/Convolutional_neural_network

[8]https://en.wikipedia.org/wiki/Radial_basis_function_network

[9] https://en.wikipedia.org/wiki/Feedforward_neural_network

[10] https://www.tensorflow.org/guide/keras

[11] https://keras.io/