ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Βελτιστοποίηση κατανεμημένης εκπαίδευσης νευρωνικών δικτύων μέσω υβριδικής αρχιτεκτονικής κατανεμημένης εκπαίδευσης σε περιβάλλον νέφους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ιάσων Θ. Χαλά

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Αθήνα, Νοέμβριος 2021

Εθνικο Μετσοβιο Πολυτεχνειο
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΤΕΧΝΟΛΟΓΙΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ
ΕΡΓΑΣΤΗΡΙΟ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

Βελτιστοποίηση κατανεμημένης εκπαίδευσης νευρωνικών δικτύων μέσω υβριδικής αρχιτεκτονικής κατανεμημένης εκπαίδευσης σε περιβάλλον νέφους

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Ιάσων Θ. Χαλά

Επιβλέπων: Νεκτάριος Κοζύρης
Καθηγητής Ε.Μ.Π.

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 'Ημ/νια παρουσίασης':

.....	.....	.....
Νεκτάριος Κοζύρης	Διονύσιος Πνευματικάτος	Ιωάννης Κωνσταντίνου
Καθηγητής	Καθηγητής	Επικ. Καθηγητής
Ε.Μ.Π.	Ε.Μ.Π.	Παν. Θεσσαλίας

Αθήνα, 23 Νοεμβρίου 2021.

.....

Ιάσων Θ. Χαλάς
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών

Περίληψη

Η βαθιά μηχανική μάθηση τα τελευταία χρόνια έχει αποτελέσει έναν πολύ σημαντικό παράγοντα στην δημιουργία μιας πληθώρας εφαρμογών με αποτέλεσμα να έχει συγκεντρώσει μεγάλο ερευνητικό ενδιαφέρον. Η αύξηση των δεδομένων καθώς και η δημιουργία μεγαλύτερων και πιο εκλεπτυσμένης αρχιτεκτονικής νευρωνικών δικτύων έχουν συντελέση στην αναγκαιότητα της εκπαίδευσης των δικτύων αυτών σε κατανεμημένο περιβάλλον. Υπάρχουν διάφορες προκλήσεις τόσο στην αρχιτεκτονική όσο και στον συγχρονισμό της κατανεμημένης εκπαίδευσης παραλληλοποίησης δεδομένων. Το TensorFlow της Google αποτελεί ένα σύστημα που προσφέρει την δυνατότητα κατανεμημένης εκπαίδευσης το οποίο χρησιμοποιήσαμε για να αξιολογήσουμε στρατηγικές κατανεμημένης εκπαίδευσης παραλληλισμού δεδομένων πάνω σε ένα cluster CPU. Στην εργασία αυτή αξιολογήθηκαν η all-reduce αρχιτεκτονική Multi Worker Mirrored Strategy και η Parameter Server Strategy και προτάθηκε μία υβριδική στρατηγική η Strategy Switch. Η Strategy Switch αποδείχθηκε η καλύτερη επιλογή στο trade-off ακρίβεια πρόβλεψης του μοντέλου και χρόνου εκπαίδευσης, προσεγγίζοντας την βέλτιστη απόδοση σε μικρό συγκριτικά χρόνο εκτέλεσης.

Λέξεις Κλειδιά: Βαθιά μηχανική μάθηση, κατανεμημένα συστήματα, κατανεμημένη εκπαίδευση, νευρωνικά δίκτυα, παραλληλοποίηση δεδομένων, TensorFlow

Abstract

Deep learning in recent years has been a very important factor in the creation of a variety of applications and as a result has attracted a great deal of research interest. The growth of data as well as the creation of larger and more sophisticated neural network architecture contribute to the need to train these networks in a distributed environment. There are various challenges in both the architecture and the synchronization of data parallelism distributed training. Google's TensorFlow is a distributed training system that we used to evaluate distributed data parallelism training strategies on a CPU cluster. In this work the all-reduce architecture Multi Worker Mirrored Strategy and the Parameter Server Strategy were evaluated and a hybrid strategy the Strategy Switch was proposed. The Strategy Switch proved to be the best option in the trade-off of model prediction accuracy and training time, approaching optimum accuracy in a relatively short execution time.

Keywords: Deep Learning, distributed systems, distributed deep learning, neural networks, data parallelism, TensorFlow

Ευχαριστίες

Πρωτίστως, θα ήθελα να ευχαριστήσω τον καθηγητή Νεκτάριο Κοζύρη για την ολοκληρωμένη και πολύ εποικοδομητική ερευνητική εμπειρία όπου είχα στο Εργαστήριο Υπολογιστικών Συστημάτων στο οποίο εργάστηκα για την διπλωματική μου εργασία. Θα ήθελα να ευχαριστήσω τον Επίκουρο Καθηγητή Ιωάννη Κωνσταντίνου που μου έδωσε την ευκαιρία να συνεργαστώ μαζί του με ένα τόσο ενδιαφέρον θέμα. Τον υποψήφιο διδάκτορα Νικόδημο Προβατά για την μεθοδικότητα του, τις γνώσεις που μου μεταλαμπάδευσε καθώς και την βοήθεια και τις συμβουλές που μου παρείχε.

Θα ήθελα να ευχαριστήσω όλους τους φίλους μου και την σχέση μου για την στήριξη και την υπομονή τους καθόλη την διάρκεια εκπόνησης αυτής της εργασίας. Χρωστάω ένα μεγάλο ευχαριστώ στον Ανδρέα, τον Άγη, τον Βασίλη, τον Γιώργο και την Νίκη για αυτά τα πέντε αξέχαστα χρόνια που ζήσαμε στην σχολή, τις δύσκολες και ευχάριστες στιγμές, το διάβασμα, τις διαφωνίες, τους στόχους που πετύχαμε και τα όνειρα που κάναμε μαζί.

Τέλος αφιερώνω την διπλωματική μου εργασία στην οικογένεια μου, τους γονείς μου, Τάσο και Μαρία για την στήριξη τους και τις θυσίες που έκαναν όλα αυτά τα χρόνια και την αδελφή μου Ναυσικά για την έμπνευση και κίνητρο που μου δίνει και την ατελείωτη βοήθεια που μου παρέχει.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Κίνητρο της εργασίας

Η βαθιά μηχανική μάθηση τα τελευταία χρόνια έχει αποτελέσει πολύ σημαντικό κομμάτι έρευνας και έχει χρησιμοποιηθεί σε πολλούς και διαφορετικούς τομείς. Για παράδειγμα τα νευρωνικά δίκτυα έχουν χρησιμοποιηθεί για ταξινόμηση εικόνων [1] και αναγνώριση προτύπων [2] για την επίλυση πολύ σημαντικων προβλημάτων στον τομέα της ιατρικής [3]. Άλλοι τομείς με πολύ μεγάλο ερευνητικό ενδιαφέρον στην βαθιά μηχανική μάθηση περιλαμβάνουν την αναγνώριση ομιλίας [4], την ταξινόμηση κειμένου [5] και την αναγνώριση συναισθημάτων [6]. Καθώς η βαθιά μηχανική μάθηση βρίσκει εφαρμογή σε όλους αυτούς τους εντελώς διαφορετικούς τομείς, πολλοί ερευνητές έχουν εργαστεί για τη βελτίωση αυτού του πεδίου.

Ένας από του βασικότερους παράγοντες της επιτυχίας της βαθιάς μηχανικής μάθησης τα τελευταία χρόνια είναι η κλιμάκωση της εκπαίδευσης των δικτύων σε τρεις διαστάσεις. Η πρώτη διάσταση αφορά την κλιμάκωση του μεγέθους και της πολυπλοκότητας των μοντέλων. Αυξάνοντας το βάθος των δικτύων και χρησιμοποιώντας πιο εκλεπτυσμένες αρχιτεκτονικές δικτύων επιτεύχθηκε πολύ μεγάλη ακρίβεια πρόβλεψης των μοντέλων. Η δεύτερη διάσταση κλιμάκωσης είναι το μέγεθος των δεδομένων. Σημαντική βελτίωση στη ακρίβεια πρόβλεψης παρατηρήθηκε με την αύξηση των δεδομένων εκπαίδευσης των μοντέλων. Τέλος η τρίτη διάσταση αφορά την κλιμάκωση των υποδομών και της υπολογιστικής ισχύς που είναι διαθέσιμη και έχει σαν αποτέλεσμα την πολύ σημαντική μείωση χρόνου εκπαίδευσης.

Η εκπαίδευση πολύ μεγάλων μοντέλων με πολύ μεγάλο όγκο δεδομένων σε σύντομο χρονικό διάστημα είναι μία πρόκληση που για να αντιμετωπιστεί είναι αναγκαία η χρήση παράλληλου προγραμματισμού και κατανεμημένων συστημάτων. Οι δύο πιο γνωστές αρχιτεκτονικές κατανεμημένης εκπαίδευσεις παραλληλισμού δεδομένων νευρωνικών δικτύων είναι η all-reduce [7] και η ο parameter server [8]. Η All-reduce στρατηγική χρησιμοποιεί reduce operators ώστε να συνδυαστούν τα gradients από τους διάφορους εργάτες που συμμετέχουν στη διαδικασία εκπαίδευσης. Η parameter server στρατηγική χρησιμοποιεί servers για να αποθηκεύει και ενημερώνει τις παρα-

μέτρους του μοντέλου εκπαίδευσης και όλοι οι εργάτες αλληλεπιδρούν με αυτούς. Επίσης υπάρχουν δύο βασικοί τρόποι συγχρονισμού των παραμέτρων του μοντέλου η σύγχρονη και η ασύγχρονη εκπαίδευση. Στην σύγχρονη όλοι οι εργάτες έχουν ακριβώς το ίδιο μοντέλο κάθε χρονική στιγμή της εκπαίδευσης, ενώ στην ασύγχρονη υπάρχει μία μεγαλύτερη ευελιξία και προσφέρει ταχύτερη εκπαίδευση.

Όλες αυτές οι διαφορετικές αρχιτεκτονικές προσεγγίσεις και η σημασία του τομέα έχουν ανοίξει το δρόμο σε διάφορες εταιρείες τεχνολογίας να σχεδιάσουν και να αναπτύξουν πολλαπλά συστήματα για εκπαίδευση νευρωνικών δικτύων. Η Google αναπτύσσει το TensorFlow [9] από το 2015, ενώ το ερευνητικό εργαστήριο AI του Facebook εργάζεται στο PyTorch [10] από το 2016.

Στην εργασία αυτή στοχεύουμε στην αξιολόγηση των κατανεμημένων στρατηγικών εκπαίδευσης του TensorFlow για clusters από CPU, Multi Worker Mirrored Strategy και Parameter Server Strategy, αλλά και στο να προτείνουμε μία υβριδική στρατηγική την Strategy Switch. Οι βασικές μετρικές αξιολόγησης που λήφθηκαν υπόψη στα πειράματα που πραγματοποιήθηκαν είναι η ακρίβεια του μοντέλου και ο χρόνος εκπαίδευσης. Τα βασικά πορίσματα είναι τα εξής:

- Η Multi Worker Mirrored Strategy έχει την καλύτερη απόδοση και με τον μεγαλύτερο χρόνο εκπαίδευσης.
- Η Parameter Server Strategy είναι η πιο γρήγορα αλλά δεν πετυχαίνει την βέλτιστη ακρίβεια πρόβλεψης.
- Η Strategy Switch πετυχαίνει απόδοση που προσεγγίζει κατα πολύ την Multi Worker Mirrored Strategy ενώ ο χρόνος εκτέλεσης είναι έως και 12% μικρότερος.

1.2 Δομή της εργασίας

- Στο Κεφάλαιο 2 γίνεται μία βιβλιογραφική ανασκόπηση σχετικά με τα νευρωνικά δίκτυα και την εκπαίδευση τους καθώς αναλύονται και δίκτυα που χρησιμοποιήθηκαν στο πειραματικό σκέλος της εργασίας.
- Στο Κεφάλαιο 3 παρουσιάζονται οι βασικές έννοιες της κατανεμημένης εκπαίδευσης νευρωνικών δικτύων καθώς και ανάλυση του TensorFlow.
- Στο Κεφάλαιο 4 αναπτύσσεται η μεθοδολογία του πειραματικού σκέλους της εργασίας, παρουσιάζοντας την πειραματική διάταξη, τα σύνολα δεδομένων και τα πειράματα που πραγματοποιήθηκαν.
- Στο Κεφάλαιο 5 γίνεται μία εκτενής πειραματική αναλυση καθώς παρουσιάζονται και σχολιάζονται όλα τα αποτελέσματα
- Στο Κεφάλαιο 6 παρουσιάζονται τα συγκεντρωτικά αποτελέσματα και δίνονται πιθανές επεκτάσεις αυτή της εργασίας.

Κεφάλαιο 2

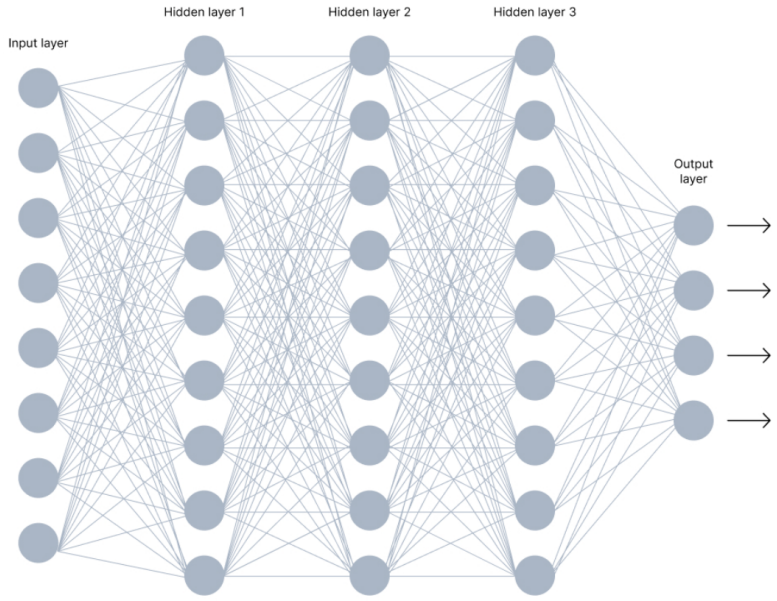
Νευρωνικά Δίκτυα

Στην ενότητα αυτή γίνεται μία βιβλιογραφική αναφορά στα νευρωνικά δίκτυα. Αρχικά θα αναπτυχθούν τα τροφοδοτικά νευρωνικά δίκτυα όπου αποτελούν την βάση των νευρωνικών δικτύων. Στην συνέχεια θα παρουσιαστούν τα συνελικτικά δίκτυα όπου χρησιμοποιήθηκαν για την πειραματική διαδικασία αυτής της εργασίας και πιο συγκεκριμένα δύο γνωστά νευρωνικά δίκτυα το ResNet [11] και το DenseNet [12].Τέλος θα παρουστιστεί ο τρόπος εκπαίδευσης αυτών των νευρωνικών δικτύων καθώς και οι αλγόριθμοι με τους οποίους γίνεται αυτή.

Ένα νευρωνικό δίκτυο είναι ένα δίκτυο διασυνδεδεμένων τεχνητών νευρώνων, οι οποίοι είναι μαθηματικές συναρτήσεις που μετατρέπουν ένα σύνολο σημάτων εισόδου σε ένα σήμα εξόδου. Κατασκευάζοντας επίπεδα νευρώνων και συνδέοντάς τα ξεκινώντας από ένα επίπεδο εισόδου και καταλήγοντας σε ένα επίπεδο εξόδου, δημιουργείται το συνολικό δίκτυο όπου αντιπροσωπεύει μια συνάρτηση f: x $\rightarrow$ y που αντιστοιχεί ένα σήμα εισόδου x που εισέρχεται στο επίπεδο εισόδου (επίπεδο 1) σε ένα σήμα εξόδου y που βγαίνει από το τελευταίο επίπεδο n. Ο στόχος της f είναι να προσεγγίσει μια συνάρτηση στόχου f*, π.χ., έναν ταξινομητή y = f *(x) που αντιστοιχίζει μια είσοδο x σε μια κατηγορία y . Στη διαδικασία εκπαίδευσης, το σύνολο των παραμέτρων Θ, δηλαδή τα βάρη, οι προκαταλήψεις (biases) και τα κατώφλια (thresholds) , σε όλους τους τεχνητούς νευρώνες προσαρμόζονται με τέτοιο τρόπο ώστε η έξοδος της f να προσεγγίζει την έξοδο της συνάρτησης στόχου f* με την καλύτερη δυνατή ακρίβεια. Αυτό επιτυγχάνεται συνήθως με την εφαρμογή του back-propagation [13] με κάποιον gradient descent αλγόριθμο [14].

2.1 Τροφοδοτικά Νευρωνικά Δίκτυα

Τα πιο δημοφιλή νευρωνικά δίκτυα είναι τα πολυεπίπεδα τροφοδοτικά νευρωνικά δίκτυα (multi-layer perceptron) [15] όπου αποτελούνται από μία σειρά πλήρως συνδεδεμένων επιπέδων νευρώνων (εικόνα 2.1).



Εικόνα 2.1: Παράδειγμα Τροφοδοτικού Νευρωνικού Δικτύου.

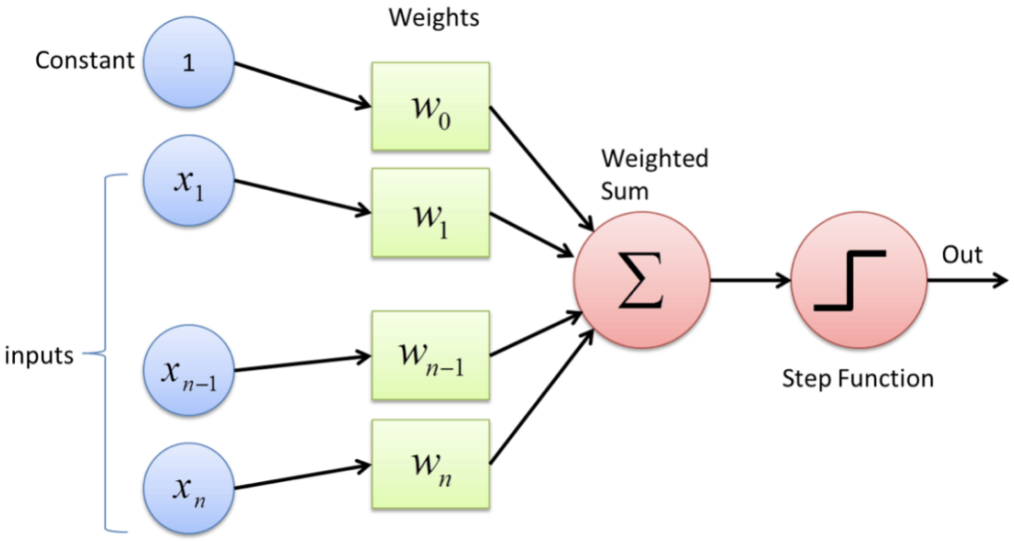
Τα δίκτυα αυτά αποτελούνται από μία σειρά από επίπεδα. Κάθε επίπεδο έχει έναν αριθμό από νευρώνες όπου κάθε νευρώνας συνδέεται με κάθε έναν νευρώνα του επόμενου επιπέδου. Η πληροφορία εισέρχεται από το επίπεδο εισόδου (input layer). Σε αυτό το επίπεδο κάθε νευρώνας αντιστοιχεί σε ένα διαφορετικό χαρακτηριστικό της πληροφορίας εισόδου. Στην συνέχεια είναι τα κρυφά επίπεδα (hidden layers) όπου κάθε ένα από αυτα τα επίπεδα εφαρμόζει ένα μη γραμμικό μετασχηματισμό στην έξοδο του αμέσως προηγούμενου επιπέδου και υπολογίζει μία συγκεκριμένη πιο περίπλοκη πληροφορία που υποδηλώνεται από τον συνδυασμό των χαρακτηριστικών που παίρνει ως είσοδο. Τέλος βρίσκεται το επίπεδο εξόδου (output layer)το οποίο μπορεί να αποτελείται από έναν ή πολλούς νευρώνες. Για παράδειγμα, αν το πρόβλημα είναι δυαδικό, δηλαδή υπάρχουν δύο κλάσεις τότε αρκεί ένας νευρώνας στο επίπεδο εξόδου, αλλά αν υπάρχουν περισσότερες κλάσεις τότε είναι αναγκαίοι και οι αντίστοιχοι νευρώνες.

2.1.1 Μαθηματικό μοντέλο νευρώνα

Στην παρακάτω εικόνα 2.2 βλέπουμε το μαθηματικό μοντέλο ενός νευρώνα. Όπου $x_1, x_2, \ldots, x_n$ αποτελούν τα σήματα εισόδου, $w_1, w_2, \ldots, w_n$ τα βάρη του νευρώνα και w_0 αποτελεί το bias. Η έξοδος του νευρώνα είναι $y = f(w_0 + w_1x_1 + ... + w_nx_n)$ η οποία υπολογίζεται εφαρμόζοντας έναν μετασχηματισμό f πάνω από το σταθμισμένο άθροισμα των σημάτων εισόδου. Ο μετασχηματισμός f ονομάζεται συνάρτηση ενεργοποίησης. Η συνάρτηση ενεργοποίησης επιλέγεται συνήθως ως μη γραμμική. Αυτό επιτρέπει στα νευρωνικά δίκτυα να μάθουν πολύπλοκους μη γραμμικούς μετα-

σχηματισμούς στο σήμα εισόδου. Παρακάτω θα αναλυθούν οι βασικές συναρτήσεις ενεργοποίησης.

Τα βάρη και το bias είναι και οι δύο παράμετροι που εκπαιδεύονται μέσα σε κάθε νευρώνα και κατ΄ επέκταση στο νευρωνικό δίκτυο. Καθώς η εκπαίδευση συνεχίζεται οι δύο παράμετροι προσαρμόζονται στις επιθυμητές τιμές και τη σωστή έξοδο. Η τιμή του bias επιτρέπει την μετατόπιση της καμπύλης της συνάρτησης ενεργοποίησης δεξιά και αριστερά. Ενώ τα βάρη δείχνουν την δύναμη του συγκεκριμένου κόμβου εισόδου.



Εικόνα 2.2: Μαθηματική μοντελοποίηση νευρώνα.

2.1.2 Συνάρτηση Ενεργοποίησης

Μια συνάρτηση ενεργοποίησης σε ένα νευρωνικό δίκτυο ορίζει πώς το σταθμισμένο άθροισμα της εισόδου μετατρέπεται σε έξοδο από έναν κόμβο ή κόμβους σε ένα επίπεδο του δικτύου. Η επιλογή της συνάρτησης ενεργοποίησης έχει μεγάλο αντίκτυπο στην απόδοση του νευρωνικού δικτύου και μπορούν να χρησιμοποιηθούν διαφορετικές συναρτήσεις ενεργοποίησης σε διαφορετικά μέρη του μοντέλου.

Όπως αναφέραμε και προηγουμένως ένα νευρωνικό δίκτυο έχει τρεις τύπους επιπέδων. Τα επίπεδα εισόδου που λαμβάνουν ακατέργαστα τα δεδομένα εισόδου του δικτύου, τα κρυφά επίπεδα που λαμβάνουν είσοδο από άλλο επίπεδο, είτε αυτό είναι κάποιο επίπεδο εισόδου ή κάποιο άλλο κρυφό επίπεδο, και περνούν έξοδο σε άλλο επίπεδο, και τέλος τα επίπεδα εξόδου όπου κάνουν την πρόβλεψη. Όλα τα κρυφά επίπεδα χρησιμοποιούν συνήθως την ίδια συνάρτηση ενεργοποίησης. Το επίπεδο εξόδου θα χρησιμοποιεί συνήθως διαφορετική συνάρτηση ενεργοποίησης από τα κρυφά επίπεδα και εξαρτάται από τον τύπο πρόβλεψης που απαιτείται από το μοντέλο. Οι

συναρτήσεις ενεργοποίησης είναι θεμιτό να είναι διαφοροποιήσιμες, πράγμα που σημαίνει ότι η παράγωγος πρώτης τάξης μπορεί να υπολογιστεί για μια δεδομένη τιμή εισόδου. Αυτό απαιτείται δεδομένου ότι τα νευρωνικά δίκτυα εκπαιδεύονται χρησιμοποιώντας τον αλγόριθμο Gradient Descent (αναλύεται στην συνέχεια) και απαιτεί την παράγωγο του σφάλματος πρόβλεψης προκειμένου να ενημερωθούν τα βάρη του μοντέλου.

Κρυφά επίπεδα

Στα κρυφά επίπεδα του νευρωνικού δικτύου χρησιμοποιείται συνήθως μια διαφοροποιήσιμη μη γραμμική συνάρτηση ενεργοποίησης. Αυτό επιτρέπει στο μοντέλο να μάθει πιο πολύπλοκες συναρτήσεις από ένα δίκτυο που εκπαιδεύεται χρησιμοποιώντας μια γραμμική συνάρτηση ενεργοποίησης.

Οι τρεις πιο δημοφιλής συναρτήσεις που χρησιμοποιούνται είναι η Rectified Linear Activation (ReLu) [16],η Logistic (Sigmoid) [17] και η Hyperbolic Tangent (Tanh) [18].

ReLu

Είναι η πιο χρησιμοποιούμενη συνάρτηση ενεργοποίησης. Καθώς είναι τόσο απλή στην εφαρμογή όσο και αποτελεσματική για την υπέρβαση των περιορισμών άλλων παλαιότερα δημοφιλών συναρτήσεων ενεργοποίησης, όπως η Sigmoid και το Tanh . Συγκεκριμένα, είναι λιγότερο επιρρεπής στο πρόβλημα την εξασθένισης των κλίσεων (Vanishing Gradients Problem) που εμποδίζουν την εκπαίδευσή τους σε βαθιά μοντέλα. Παρόλα αυτά η ReLu μπορει να αντιμετωπίζει άλλα προβλήματα όπως το πρόβλημα των "νεκρών" κόμβων.

Η ReLu περιγράφεται από την εξίσωση 2.1

$$f(x) = \max(0, x) \tag{2.1}$$

Δηλαδή αν η είσοδος είναι αρνητική η ReLu επιστρέφει 0, διαφορετικά επιστρέφει x. Παρακάτω βλέπουμε το διάγραμμα της συνάρτησης αυτής.

Sigmoid

Η συνάρτηση αυτή λαμβάνει οποιαδήποτε πραγματική τιμή ως τιμή εισόδου ενώ η έξοδος είναι στο εύρος 0 έως 1. Όσο μεγαλύτερη είναι η είσοδος (πιο θετική), τόσο πιο κοντά είναι η τιμή εξόδου στο 1, ενώ όσο μικρότερη είναι η είσοδος (πιο αρνητική), τόσο πιο κοντά η έξοδος θα είναι στο 0.

Η συνάρτηση Sigmoid περιγράφεται από την εξίσωση 2.2

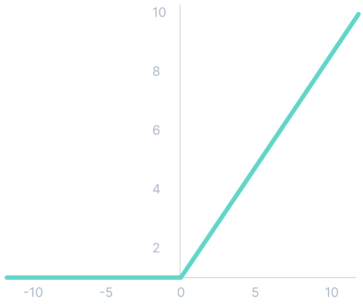
$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

Tanh

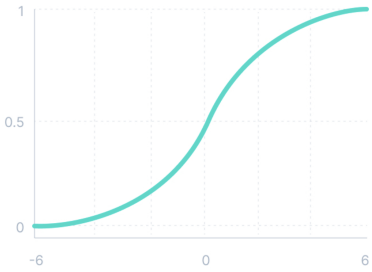
Η Tanh είναι πολύ παρόμοια με τη συνάρτηση ενεργοποίησης Sigmoid και έχει το ίδιο σχήμα S. Η συνάρτηση λαμβάνει οποιαδήποτε πραγματική τιμή ως τιμές εισόδου και η έξοδος είναι στο εύρος τιμών -1 έως 1. Όσο μεγαλύτερη είναι η είσοδος (πιο θετική), τόσο πιο κοντά θα είναι η τιμή εξόδου στο 1, ενώ όσο μικρότερη είναι η είσοδος (πιο αρνητική), τόσο πιο κοντά η έξοδος θα είναι -1.

Η συνάρτηση Tanh περιγράφεται από την εξίσωση 2.3

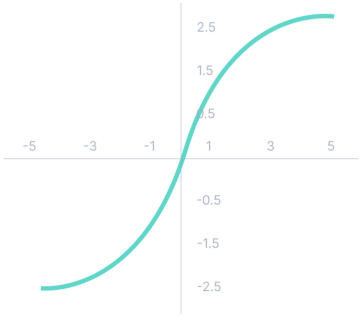
$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.3}$$



(α′) ReLu



(β′) Sigmoid



(γ′) Tanh

Εικόνα 2.3: Συναρτήσεις ενεργοποίησης

Vanishing Gradients Problem

Στη μηχανική μάθηση, το vanishing gradient problem [19] εξαφανίζεται κατά την εκπαίδευση νευρωνικών δικτύων με μεθόδους gradient-based και back-propagation. Σε τέτοιες μεθόδους, κάθε βάρος των νευρωνικών δικτύων λαμβάνει μια ενημέρωση ανάλογη με τη μερική παράγωγο της συνάρτησης σφάλματος σε σχέση με το τρέχον βάρος σε κάθε επανάληψη της εκπαίδευσης. Το πρόβλημα είναι ότι σε ορισμένες περιπτώσεις, αυτή η παράγωγος είναι εξαιρετικά μικρή, εμποδίζοντας αποτελεσματικά το

βάρος να αλλάξει την τιμή του. Οι παραδοσιακές συναρτήσεις ενεργοποίησης , όπως η Tanh, έχουν παραγώγους στο εύρος (0,1] και το back-propagation υπολογίζει παραγώγους με τον κανόνα της αλυσίδας. Αυτό έχει ως αποτέλεσμα τον πολλαπλασιασμό ν μικρών αριθμών για υπολογισμό των παραγώγων των πρώτων επιπέδων σε ένα δίκτυο ν επίπεδα, που σημαίνει ότι η παράγωγος σφάλματος μειώνεται εκθετικά με το ν και συνεπώς τα πρώτα επίπεδα των νευρωνικών δικτύων εκπαιδεύονται πολύ αργά. Έτσι το πρόβλημα αυτό καθυστά δύσκολη ή μέχρι και αδύνατη την εκπαίδευση πολύ βαθιών νευρωνικών δικτύων.

Σχόλια

Τόσο η Sigmoid όσο και η Tanh μπορούν να χρησιμοποιθούν για την εκπαίδευση ένος μοντέλου άλλα όταν αυτό είναι πολύ βαθύ, δηλαδή με πολλά κρυφά επίπεδα αντιμετοπίζουν το πρόβλημα vanishing gradients problem. Η συνάρτηση ενεργοποίησης που χρησιμοποιείται σε κρυμμένα επίπεδα επιλέγεται συνήθως με βάση τον τύπο της αρχιτεκτονικής νευρωνικών δικτύων. Τα τροφοδοτικά νευρωνικά δίκτυα καθώς και τα συνελικτικά τροφοδοτικά δίκτυα χρησιμοποιούν ως συνάρτηση ενεργοποίησης την ReLu. Αντίθερα τα recurrent νευρωνικά δίκτυα χρησιμοποιούν τόσο την Sigmoid όσο και την Tanh σηνάρτηση ενεργοποίησης. Πιο συγκεκριμένα στα LSTM συνήθως χρησιμοποιούνται για τις recurrent συνδέσεις η Sigmoid συνάρτηση ενεργοποίησης ενώ για τις συνδέσεις εξόδου του νευρώνα χρησιμοποιείται η Tanh.

Επίπεδα Εξόδου

Το επίπεδο εξόδου είναι το επίπεδο σε ένα μοντέλο νευρωνικού δικτύου που εξάγει άμεσα την πρόβλεψη. Δύο είναι οι βασικές συναρτήσεις ενεργοποίησης για τα επίπεδα εξόδου στα μοντέλα ταξινόμησης, η Sigmoid που αναλύθηκε προηγουμένως και η Softmax [20].

Softmax

Η συνάρτηση Softmax εξάγει ένα διάνυσμα τιμών που αθροίζεται στο διάστημα [0,1] οι οποίες μπορούν να ερμηνευθούν ως πιθανότητες επιλογής των κλάσεων. Η είσοδος στη συνάρτηση είναι ένα διάνυσμα πραγματικών τιμών και η έξοδος είναι ένα διάνυσμα του ίδιου μήκους με τιμές στο [0,1] σαν πιθανότητες.

Η συνάρτηση Softmax περιγράφεται από την εξίσωση 2.4

$$\sigma(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_y}} \tag{2.4}$$

Όπου $x = [x_1, x_2, \ldots, x_k]$ το διάνυσμα εισόδου.

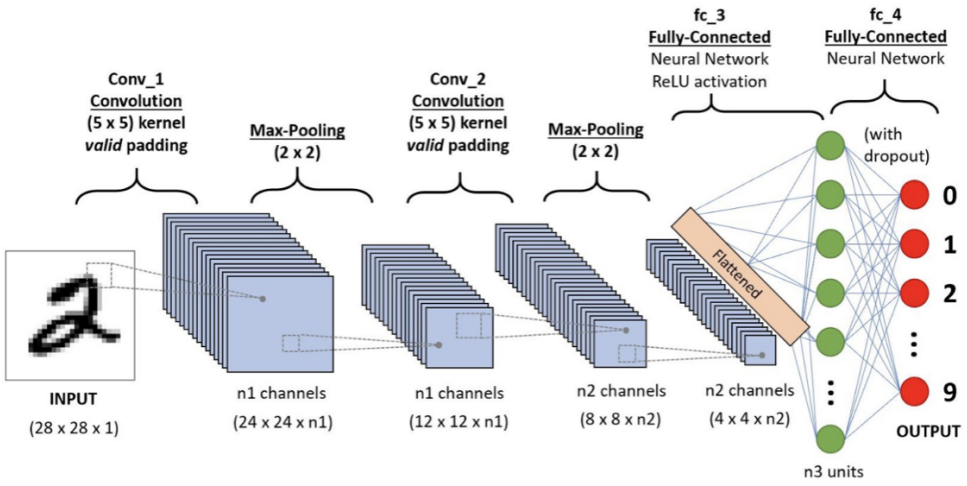
Η συνάρτηση ενεργοποίησης που θα επιλέγει στο επίπεδο εξόδου εξαρτάται άμεσα από τον τύπο της ταξινόμησης του προβλήματος. Εάν υπάρχουν δύο αμοιβαία αποκλειόμενες κλάσεις (δυαδική ταξινόμηση), τότε το επίπεδο εξόδου θα έχει έναν κόμβο

και θα πρέπει να χρησιμοποιηθεί μια συνάρτηση ενεργοποίησης Sigmoid . Εάν υπάρχουν περισσότερες από δύο αμοιβαία αποκλειόμενες κλάσεις (ταξινόμηση πολλαπλών κλάσεων), τότε το επίπεδο εξόδου θα έχει έναν κόμβο ανά κλάση και θα πρέπει να χρησιμοποιείται η συνάρτηση ενεργοποίησης Softmax . Εάν υπάρχουν δύο ή περισσότερες κλάσεις όχι αμοιβαία αποκλειόμενες (ταξινόμηση πολλαπλών ετικετών), τότε το επίπεδο εξόδου θα έχει έναν κόμβο για κάθε κλάση και κάθε κλάση θα χρησιμοποιεί μια συνάρτηση ενεργοποίησης Sigmoid.

2.2 Συνελικτικά Νευρωνικά Δίκτυα

Τα Συνελικτικά Νευρωνικά Δίκτυα Convolutional Neural Networks (CNN) είναι όμοια με τα παραδοσιακά τροφοδοτικά νευρωνικά δίκτυα στο ότι αποτελούνται από νευρώνες που αυτο-βελτιστοποιούνται μέσω της μάθησης. Κάθε νευρώνας συνεχίζει να λαμβάνει μια είσοδο και θα εκτελέσει μία λειτουργία, που θα εφαρμόζει μια μη γραμμική συνάρτηση. Η μόνη αξιοσημείωτη διαφορά μεταξύ των CNN και των παραδοσιακών τροφοδοτικών νευρωνικών δικτύων είναι ότι τα CNN χρησιμοποιούνται κυρίως στον τομέα της αναγνώρισης προτύπων εντός εικόνων. Τα CNN είναι κατασκευασμένα έτσι ώστε να επιτρέπουν την κωδικοποίηση των χαρακτηριστικών της εικόνας στην αρχιτεκτονική τους, καθιστώντας τα δίκτυα καταλληλότερα για προβλήματα εστιασμένα σε εικόνες, μειώνοντας παράλληλα περαιτέρω τις παραμέτρους που απαιτούνται για τη εκπαίδευση του μοντέλου.

Όπως σημειώθηκε νωρίτερα, τα CNN επικεντρώνονται κυρίως στο γεγονός ότι η είσοδος θα αποτελείται από εικόνες. Αυτό εστιάζει την αρχιτεκτονική που πρέπει να δημιουργηθεί έτσι ώστε να ταιριάζει καλύτερα στην ανάγκη αντιμετώπισης του συγκεκριμένου τύπου δεδομένων. Μία από τις βασικές διαφορές είναι ότι τα στρώματα μέσα στο CNN αποτελούνται από νευρώνες οργανωμένους σε τρεις διαστάσεις, τη χωρική διάσταση της εισόδου ύψος, πλάτος και το βάθος. Σε αντίθεση με το τυπικό τροφοδοτικό νευρωνικό δίκτυο, οι νευρώνες σε κάθε δεδομένο επίπεδο συνδέονται μόνο με ένα μικρό σύνολο από τους νευρώνες του προηγούμενου επιπέδου.



Εικόνα 2.4: Παράδειγμα Συνελικτικού Νευρωνικού Δικτύου.

Περιγραφή αρχιτεκτονικής

Τα CNN [21] αποτελούνται από τρεις τύπους επιπέδων. Αυτά είναι **συνελικτικά επίπεδα** (convolutional layers), **επίπεδα συγκέντρωσης** (pooling layers) και **πλήρως συνδεδεμένα επίπεδα** (fully-connected layers). Όταν αυτά τα επίπεδα στοιβάζονται, έχει δημιουργηθεί μια αρχιτεκτονική συνελικτικού νευρωνικού δικτύου. Στην εικόνα 2.4 βλέπουμε ένα Συνελικτικό Νευρωνικό Δίκτυο για την αναγνώριση χειρόγραφου αριθμού. Αποτελείται από δύο συνελικτικά επίπεδα, δύο επίπεδα συγκέντρωσης και δύο πλήρως συνδεδεμένα επίπεδα.

Η βασική λειτουργικότητα του παραδείγματος CNN στη παραπάνω εικόνα 2.4 μπορεί να αναλυθεί σε τέσσερις βασικούς τομείς:

1. Το επίπεδο εισόδου που διατηρεί τις τιμές των pixel της εικόνας

2. Το συνελικτικό επίπεδο που θα καθορίσει την έξοδο των νευρώνων οι οποίοι είναι συνδεδεμένοι με τοπικές περιοχές της εικόνας και υπολογίζουν την έξοδο σύμφωνα με τις τιμές εισόδου και των αντίστοιχων βαρών τους και εφαρμόζοντας μία μη γραμμική συνάρτηση ενεργοποίησης ReLu [16].

3. Το επίπεδο συγκέντρωσης που θα εκτελέσει στη συνέχεια απλή δειγματοληψία κατά μήκος της χωρικής διάστασης της δεδομένης εισόδου, μειώνοντας περαιτέρω τον αριθμό των παραμέτρων.

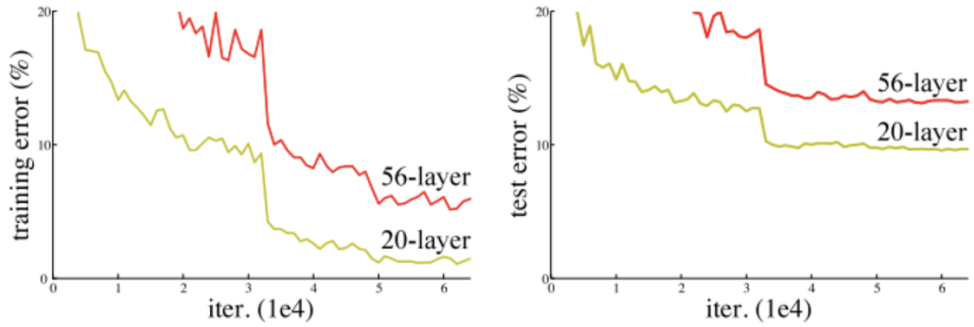
4. Στη συνέχεια, τα πλήρως συνδεδεμένα επίπεδα θα εκτελέσουν τα ίδια καθήκοντα που έχουν στα τυπικά τροφοδοτικά νευρωνικά δίκτυα και προσπαθούν να παράγουν βαθμολογίες για τις κλάσεις από τις ενεργοποιήσεις, που θα χρησιμοποιηθούν για ταξινόμηση.

Μέσω αυτής της απλής μεθόδου μετασχηματισμού, τα CNN είναι σε θέση να μετατρέψουν την αρχική είσοδο(εικόνα) περνώντας την μέσα από τα διαφορετικά επίπεδα χρησιμοποιώντας τεχνικές συνελικτικής και δειγματοληψίας να παράγουν εν τέλη βαθμολογίες για τις κλάσεις για σκοπούς ταξινόμησης και παλινδρόμησης.

2.3 ResNet

Σύμφωνα με το θεώρημα καθολικής προσέγγισης Universal approximation theorem [22], δεδομένης της επαρκούς χωρητικότητας, γνωρίζουμε ότι ένα τροφοδοτικό δίκτυο με ένα μόνο επίπεδο (layer) είναι αρκετό για να αντιπροσωπεύει οποιαδήποτε συνάρτηση. Ωστόσο, το επίπεδο μπορεί να είναι τεράστιο και το δίκτυο είναι επιρρεπές στην υπερπροσαρμογή (overfitting) των δεδομένων. Ως εκ τούτου, υπάρχει μια κοινή τάση στην ερευνητική κοινότητα ότι η αρχιτεκτονική του δικτύου χρειάζεται να έχει όλο και μεγαλύτερο βάθος, περισσότερα κρυφά επίπεδα.

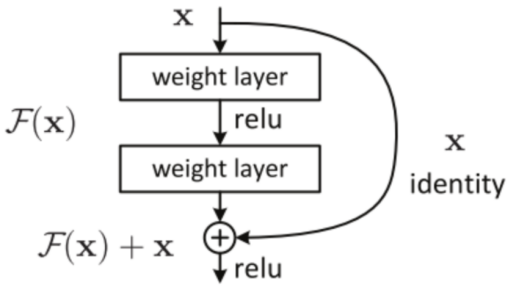
Ωστόσο, η αύξηση του βάθους του δικτύου δεν λειτουργεί απλά στοιβάζοντας επίπεδα μεταξύ τους. Τα βαθιά δίκτυα είναι δύσκολο να εκπαιδευτούν καθώς ο αλγόριθμος του back-propagation δεν μπορεί να μεταφέρει σε πολύ μεγάλο βάθος τις αλλαγές των gradients. Ως αποτέλεσμα, καθώς το δίκτυο γίνεται και βαθύτερο, η απόδοσή του σταθεροποιείται και πολλές φορές αρχίζει να μειώνεται γρήγορα, όπως φαίνεται και στην εικόνα 2.5.



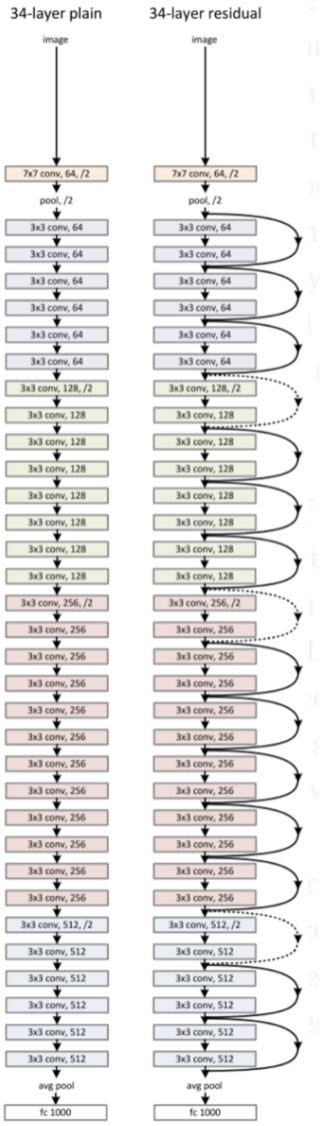
Εικόνα 2.5: Απόδοση απλού Νευρωνικού Δικτύου με 20 και 50 επίπεδα

Το ResNet [11] είναι ένα δίκτυο όπου κατάφερε να ξεπεράσει αυτό το πρόβλημα και να κατασκευάσει πολύ βαθιά δίκτυα με state of the art αποδόσεις. Η βασική ιδέα του ResNet είναι η εισαγωγή μιας λεγόμενης "identity shortcut connection" που παραλείπει ένα ή περισσότερα επίπεδα, όπως φαίνεται στο παρακάτω σχήμα 2.6α΄. Στην εικόνα 2.6β΄ φαίνεται η διαφορά αρχιτεκτονικής ένός απλού Συνελικτικού Νευρωνικού Δικτύου με το ResNet . Με το τρόπο που είναι φτιαγμένο το ResNet τα gradients μπορούν να ρέουν μέσω των συνδέσεων συντόμευσης σε οποιοδήποτε άλλο προηγούμενο επίπεδο ανεμπόδιστα.

Οι συντάκτες του [23] απέδειξαν με πειράματα ότι μπορούν τώρα να εκπαιδεύσουν ένα ResNet 1001 επιπέδων που να ξεπεράσει σε απόδοση τα πιο "ρηχά" δίκτυα και να πετυχαίνει state of the art αποδόσεις. Λόγω αυτών των αποτελεσμάτων του, το ResNet έγινε γρήγορα μια από τις πιο δημοφιλείς αρχιτεκτονικές στον τομέα της όρασης υπολογιστή.



(α′) Residual block



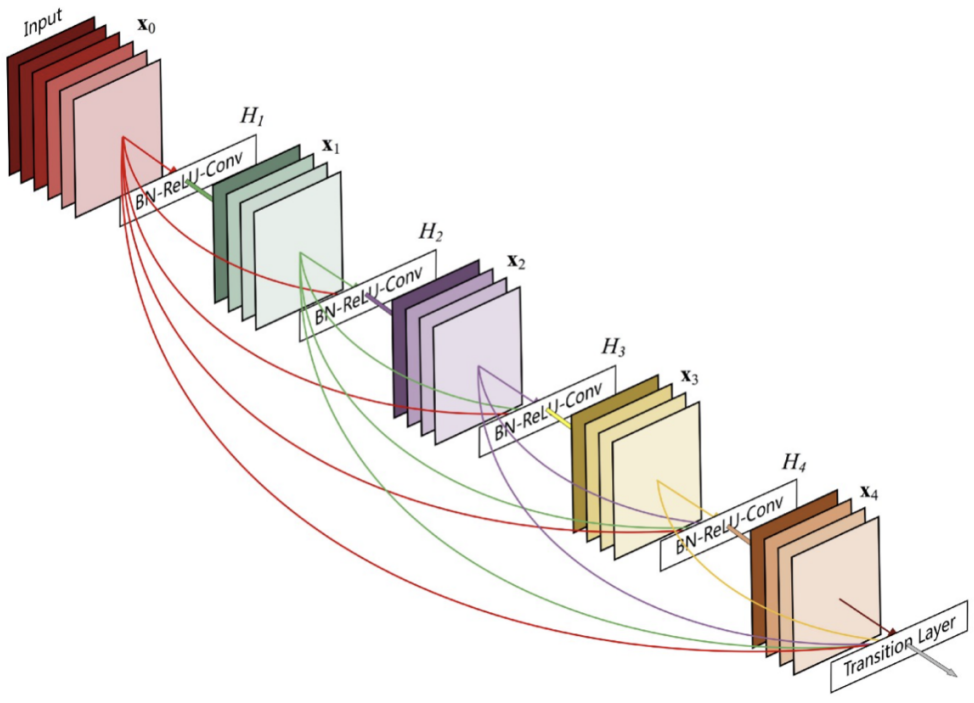
(β′) Σύγκριση απλού Συνελικτικού Νευρωνικού Δικτύου με το ResNet

Εικόνα 2.6: ResNet

2.4 DenseNet

Το DenseNet [12] αποτελεί ένα Συνελικτικό Νευρωνικό Δίκτυο (CNN). Εφευρέθηκε από κοινού από το Πανεπιστήμιο Cornell , το Πανεπιστήμιο Tsinghua και το Facebook AI Research και κέρδισε το καλύτερο άρθρο για το 2017 στο συνέδριο CVPR.Είναι εμπνευσμένο από το ResNet καθώς και σε αυτήν την περίπτωση το δίκτυο προσπαθεί να δημιουργήσει πιο άμεση ροή πληροφοριών μεταξύ αρχικών και τελικών επιπέδων. Με αυτόν τον τρόπο γίνεται εφικτή η εκπαίδευση βαθιών νευρωνικών δικτύων καθώς αντιμετωπίζεται το vanishing gradient problem.

Το DenseNet αποτελείται από dense blocks . Ακολουθώντας τη feed-forward ροή όπου ακολουθούν τα συνελικτικά νευρωνικά δίκτυα, κάθε επίπεδο (layer) σε ένα dense block λαμβάνει feature maps από όλα τα προηγούμενα επίπεδα και περνά την έξοδο του σε όλα τα επόμενα επίπεδα όπως βλέπουμε στην παρακάτω είκόνα 2.7. Τα feature maps που λαμβάνονται από άλλα επίπεδα στο DenseNet συγχωνεύονται μέσω συνένωσης (concatenation) (εικόνα 2.9) και όχι μέσω άθροισης (όπως στο ResNets).

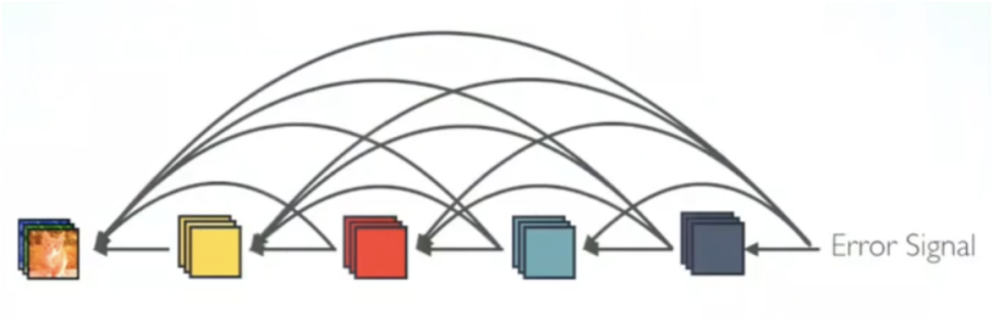


Εικόνα 2.7: Dense Block

Χαρακτιριστικά του DenseNet

Ροή gradient descent

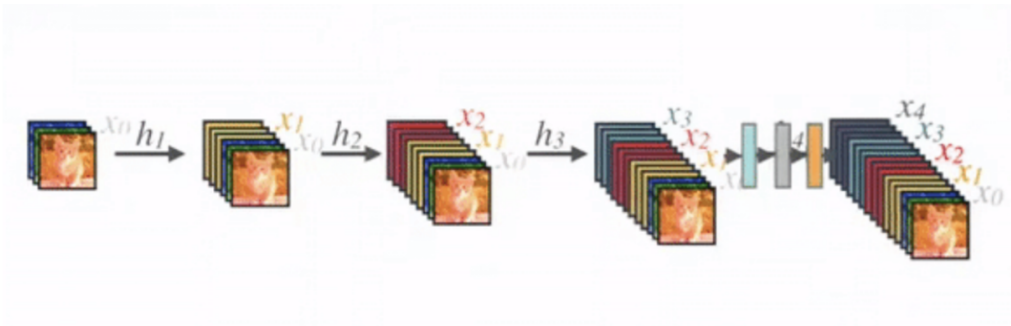
Τα gradients μπορούν εύκολα να μεταδοθούν σε προηγούμενα στρώματα πιο άμεσα κατά την διάρκεια εκτέλεσης του back propagation (εικόνα 2.8). Αυτό είναι ένα είδος έμμεσης βαθιάς εποπτείας καθώς τα προηγούμενα στρώματα μπορούν να λάβουν άμεση εποπτεία από το τελικό επίπεδο ταξινόμησης.



Εικόνα 2.8: Μετάδοση gradients στο back propagation

Παράμετροι

Λόγων των συνενώσεων με τα προηγούμενα επίπεδα όπως φαίνεται στην εικόνα 2.9 δεν απαιτούνται πολλές παράμετροι για την κατασκευή ενός αποδοτικού δικτύου. Έτσι τα DenseNet μπορούν να είναι πολύ βαθιά δίκτυα με σχετικά λίγες παραμέτρους.



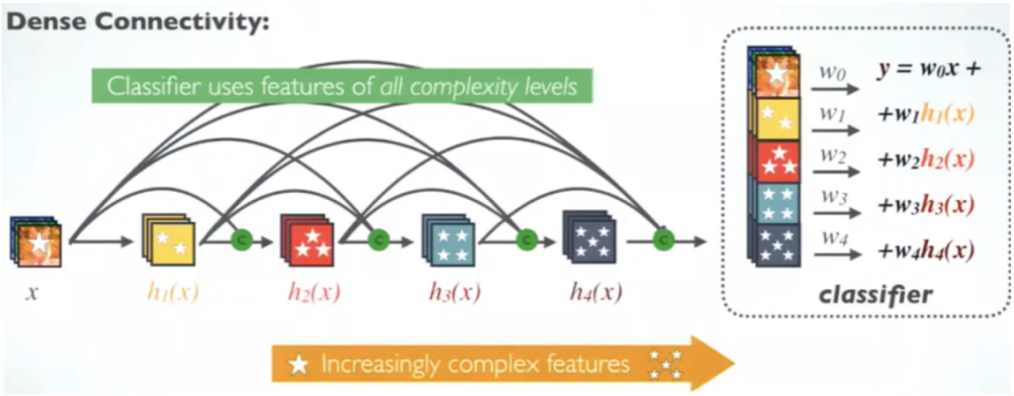
Εικόνα 2.9: Συνένωση (concatenation) των επιπέδων στο DenseNet

Μεγάλη ποικιλομορφία χαρακτηριστικών

Δεδομένου ότι κάθε επίπεδο στο DenseNet λαμβάνει όλα τα προηγούμενα επίπεδα ως είσοδο, δημιουργούνται πιο διαφοροποιημένα χαρακτηριστικά και τείνει να δημιουργεί πλουσιότερα μοτίβα.

Διατηρεί Χαρακτηριστικά Χαμηλής Πολυπλοκότητας

Σε ένα απλό συνελικτικό δίκτυο ο ταξινομητής έχει μεγάλης πολυπλοκότητας χαρακτηριστικά. Αντίθετα στο DenseNet , ο ταξινομητής χρησιμοποιεί χαρακτηριστικά όλων των επιπέδων πολυπλοκότητας όπως αναλύεται στην εικόνα 2.10. Τείνει να δίνει πιο ομαλά όρια αποφάσεων. Εξηγεί επίσης γιατί το DenseNet αποδίδει καλά όταν τα δεδομένα εκπαίδευσης είναι ανεπαρκή.



Εικόνα 2.10: Χαρακτηριστικά όλων των επιπέδων πολυπλοκότητας στο DenseNet

2.5 Εκπαίδευση Νευρωνικών Δικτύων

Για την εκπαίδευση των νευρωνικών δικτύων χρησιμοποιείται ο αλγόριθμος Gradient descent [14] ή κάποια από τις παραλλαγές του. Παρακάτω γίνεται ανάλυση αυτού του αλγορίθμου.

2.5.1 Gradient Descent

Το Gradient Descent είναι ο πιο συνηθισμένος αλγόριθμος βελτιστοποίησης στη βαθιά μηχανική μάθηση. Η βελτιστοποίηση αναφέρεται στη ελαχιστοποίησης της αντικειμενικής συνάρτησης $f(x)$ με παράμετρο το x. Στην βαθιά μηχανική μάθηση ο στόχος της εκπαίδευσης είναι η ελαχιστοποίησης της συνάρτησης κόστους $J(\theta)$ που παραμετρικοποιείται από τις παραμέτρους του μοντέλου(τα βάρη) $\theta \epsilon R^d$. Ο Gradient Descent είναι ένας αλγόριθμος βελτιστοποίησης πρώτης τάξης. Αυτό σημαίνει ότι λαμβάνει υπόψη μόνο την πρώτο παράγωγο κατά την εκτέλεση των ενημερώσεων των παραμέτρων. Επίσης ανήκει στην κατηγορία επαναληπτικών αλγορίθμων βελτιστοποίησης. Σε κάθε επανάληψη, γίνεται ενημέρωση των παραμέτρων προς την αντίθετη

κατεύθυνση της κλίσης της αντικειμενικής συνάρτησης $J(\theta)$. Το μέγεθος του βήματος που γίνεται σε κάθε επανάληψη για να φτάσει στο τοπικό ελάχιστο καθορίζεται από το ρυθμό εκμάθησης (learning rate)α. Επομένως, ακολουθούμε την κατεύθυνση της κλίσης προς τα κάτω μέχρι να φτάσουμε σε ένα τοπικό ελάχιστο. Ο αλγόριθμος αυτός βρίσκει ένα τοπικό ελάχιστο, αν η συνάρτηση είναι κυρτή τότε αυτό είναι και το ολικό ελάχιστο, αν η συνάρτηση δεν είναι κυρτή όπως τις περισσότερες φορές στα νευρωνικά δίκτυα τότε ο αλγόριθμος καταλήγει σε ένα τοπικό ελάχιστο. Σε αυτήν την περίπτωση της μη κυρτής συνάρτησης είναι πολύ σημαντική η αρχικοποίηση των παραμέτρων για την εύρεση ενός καλό τοπικού ελάχιστου.

Υπάρχουν τρεις παραλλαγές του Gradient Descent , οι οποίες διαφέρουν ως προς το πόσα δεδομένα χρησιμοποιούμε για τον υπολογισμό της κλίσης της αντικειμενικής συνάρτησης. Ανάλογα με τον όγκο των δεδομένων, κάνουμε ένα trade-off μεταξύ της ακρίβειας της ενημέρωσης παραμέτρων και του χρόνου που απαιτείται για την εκτέλεση μιας ενημέρωσης.

Batch gradient descent

Ο Batch Gradient Descent υπολογίζει την κλίση της συνάρτησης κόστους ως προς τις παραμέτρους του μοντέλου ϑ για ολόκληρο το σύνολο δεδομένων εκπαίδευσης.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta) \tag{2.5}$$

Καθώς πρέπει να υπολογιστούν οι κλίσεις για ολόκληρο το σύνολο δεδομένων για να εκτελεστεί μόνο μία ενημέρωση, ο Batch Gradient Descent είναι πολύ αργός για μεγάλα σύνολα δεδομένων εκπαίδευσης και δεν μπορεί να λειτουργήσει για σύνολα δεδομένων που δεν χωράνε στην μνήμη.

Ο κώδικας που αντιστοιχεί στο Batch Gradient Descent είναι ο εξής:

```
for i in range(nb_epochs):
  params_grad = evaluate_gradient(loss_function, data, params)
  params = params - learning_rate * params_grad
```

Για κάθε μία εποχή, υπολογίζουμε πρώτα το διάνυσμα κλίσης params grad της συνάρτησης κόστους για ολόκληρο το σύνολο δεδομένων ως προς το διάνυσμα των παραμέτρων του μοντέλου params.Στη συνέχεια, ενημερώνουμε τις παραμέτρους μας προς την αντίθετη κατεύθυνση των κλίσεων με τον ρυθμό εκμάθησης να καθορίζει πόσο μεγάλη ενημέρωση εκτελούμε. Ο Batch Gradient Descent εγγυάται ότι συγκλίνει στο ολικό ελάχιστο για κυρτές συναρτήσεις σφάλματος και στο τοπικό ελάχιστο για μη κυρτές συναρτήσεις.

Stochastic gradient descent

Ο Stochastic Gradient Descent (SGD) σε αντίθεση με την Batch Gradient Descent εκτελεί μια ενημέρωση παραμέτρων για κάθε παράδειγμα εκπαίδευσης $x^{(i)}$ και

ετικέτα $y^{(i)}$.

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{2.6}$$

Ο Batch Gradient Descent εκτελεί περιττούς υπολογισμούς για μεγάλα σύνολα δεδομένων, καθώς υπολογίζει κλίσεις για παρόμοια παραδείγματα πριν από κάθε ενημέρωση παραμέτρων. Ο SGD καταργεί αυτόν τον πλεονασμό εκτελώντας μία ενημέρωση για κάθε έναν υπολογισμό κλίσης που κάνει. Ως εκ τούτου, είναι συνήθως πολύ πιο γρήγορο και μπορεί επίσης να χρησιμοποιηθεί για να μάθει και online δηλαδή με δεδομένα που έρχονται εκείνη στην στιγμή. Ο SGD εκτελεί συχνές ενημερώσεις με μεγάλη διακύμανση που προκαλούν αντίστοιχα μεγάλη διακύμανση της αντικειμενικής συνάρτησης.

Ενώ ο Batch Gradient Descent συγκλίνει στο ελάχιστο της ΄λεκάνης΄ στην οποία τοποθετούνται οι παράμετροι, η διακύμανση του SGD , του επιτρέπει να μεταπηδήσει σε νέα και δυνητικά καλύτερα τοπικά ελάχιστα. Από την άλλη πλευρά, αυτό περιπλέκει τελικά τη σύγκλιση στο ακριβές ελάχιστο, καθώς ο SGD θα συνεχίσει να ταλαντώνεται γύρω από το σημείο αυτό. Ωστόσο, έχει αποδειχθεί ότι όταν μειώνουμε αργά το ρυθμό εκμάθησης, το SGDεμφανίζει την ίδια συμπεριφορά σύγκλισης με τον Batch Gradient Descent , σχεδόν σίγουρα συγκλίνει σε ένα τοπικό ή ολικό ελάχιστο για μη κυρτή και κυρτή συνάρτηση αντίστοιχα.

Το κομμάτι κώδικα προσθέτει απλώς έναν βρόχο που τρέχει πάνω στα δεδομένα εκπαίδευσης. Για κάθε ένα παράδειγμα υπολογίζεται η κλίση και ενημερώνεται το μοντέλο. Επίσης είναι σημαντικό να αναδευτούν τα δεδομένα πριν την εκπαίδευση σε περίπτωση που κοντινά δεδομένα έχουν κοινά χαρακτηριστικά.

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for example in data:
    params_grad = evaluate_gradient(loss_function, example, params)
    params = params - learning_rate * params_grad
```

Mini-batch Gradient Descent

Ο Mini-batch Gradient Descent παίρνει τα καλύτερα χαρακτηριστικά και από τις δύο προηγούμενες περιπτώσεις και πραγματοποιεί μια ενημέρωση για κάθε mini-batchτο οποίο αποτελείται από n παραδείγματα εκπαίδευσης.

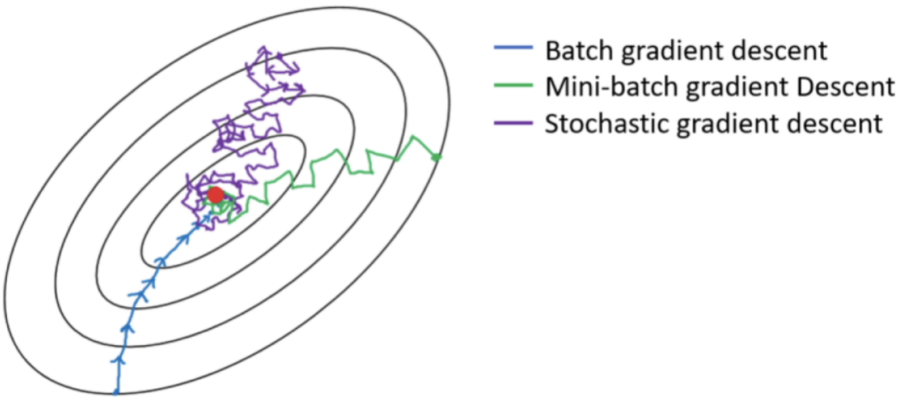
$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \tag{2.7}$$

Με αυτόν τον τρόπο, πρώτον μειώνει τη διακύμανση των ενημερώσεων των παραμέτρων, η οποία μπορεί να οδηγήσει σε πιο σταθερή σύγκλιση και δεύτερον μπορεί να χρησιμοποιήσει βελτιστοποιημένες μεθόδους που είναι κοινές σε state-of-the-art βιβλιοθήκες βαθιάς μηχανικής μάθησης που καθιστά τον υπολογισμό των κλίσεων για mini-batchπάρα πολύ αποδοτικό. Τα κοινά μεγέθη των mini-batch κυμαίνονται

μεταξύ 50 έως 256 παραδείγματα, αλλά μπορεί να διαφέρουν για διαφορετικές εφαρμογές.Ο Mini-batch Gradient Descent είναι συνήθως ο αλγόριθμος επιλογής κατά την εκπαίδευση ενός νευρωνικού δικτύου και ο όρος SGD συνήθως χρησιμοποιείται και όταν χρησιμοποιούνται mini-batches.

Στον κώδικα, αντί ο βρόχος να είναι πάνω στα παραδείγματα, τώρα είναι πάνω στα mini-batches μεγέθους 50

```
for i in range(nb_epochs):
  np.random.shuffle(data)
  for batch in get_batches(data, batch_size=50):
    params_grad = evaluate_gradient(loss_function, batch, params)
    params = params - learning_rate * params_grad
```



Εικόνα 2.11: Σύγκληση των τριών αλγορίθμων Gradient Descent

Στην εικόνα 2.11 παρατηρούμε την σύγκλιση των τριών παραλλαγών του Gradient Descent. Όπως Αναφέρθηκε ο Batch Gradient Descent αφού έχει υπολογίσει τις κλίσεις όλων των παραδειγμάτων πριν την κάθε ενημέρωση των παραμέτρων βλέπουμε ότι κάθε βήμα είναι με κατεύθυνση προς το ελάχιστο. Από την άλλη ο Stochastic Gradient Descent παρατηρούμε ότι κάνει πολύ συχνά ενημερώσεις και όχι πάντα προς την σωστή κατεύθυνση, βλέπουμε μεγάλη διακύμανση στην πορεία προς το ελάχιστο. Τέλος ο Mini batch Gradient Descent είναι η μέση λύση μεταξύ των δύο προηγούμενων ,οδεύει προς το ελάχιστο έχοντας μία όχι τόσο έντονη διακύμανση.

2.5.2 Διαδικασία εκπαίδευσης Νευρωνικών Δικτύων

Η εκπαίδευση των νευρωνικών δικτύων στην μαθιά μηχανική μάθηση γίνεται με την χρήση ενός αλγορίθμου Mini-batch Gradient Descent και η διαδικασία περιγράφεται παρακάτω.

- Αρχικά επιλέγεται τυχαία ένα mini-batch .
- Για κάθε παράδειγμα του mini-batch γίνεται ένα forward πέρασμα, υπολογίζεται το κόστος από την συνάρτηση κόστους που έχει επιλεχθεί.
- Στην συνέχεια υπολογίζονται οι μερικές παράγωγοι της συνάρτησης κόστους ως προς τις παραμέτρους του δικτύου(τα βάρη και τα bias). Αυτό γίνεται με την διαδικασία του back propagation [13], υπολογίζοντας τις παραγώγους του τελευταίου επιπέδου και στην συνέχεια με τον κανόνα της αλυσίδας υπολογίζονται οι παράγωγοι των προηγούμενων επιπέδων μέχρι να φτάσει στο πρώτο επίπεδο.
- Αφού γίνει αυτό για κάθε παράδειγμα του mini-batch αθροίζονται οι παράγωγοι για κάθε μεταβλητή και γίνεται η ενημέρωση όπως είδαμε στον αλγόριθμο του Mini-batch Gradient Descent .
- Μία εποχή ολοκληρώνεται με το πέρασμα όλων των δεδομένων από το δίκτυο.
- Η εκπαίδευση ολοκληρώνεται όταν αυτή η διαδικασία επαναληφθεί για τον αριθμό εποχών όπου έχει οριστεί.

Κεφάλαιο 3

Κατανεμημένη Εκπαίδευση

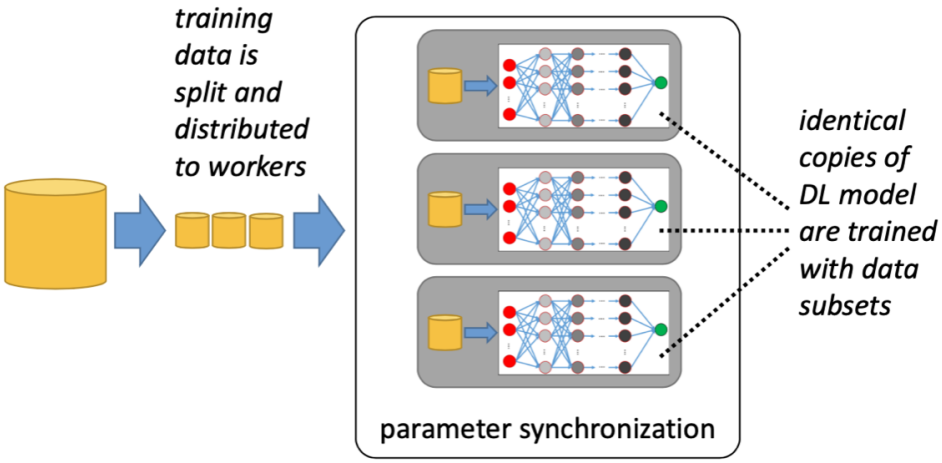
Η εκπαίδευση μεγάλων μοντέλων νευρωνικών δικτύων με τεράστιο όγκο δεδομένων εκπαίδευσης είναι μια δύσκολη εργασία. Συχνά τέτοια δίκτυα τρέχουν σε κατανεμημένες υποδομές πολλαπλών υπολογιστικών κόμβων, καθένας από τους οποίους μπορεί να είναι εξοπλισμένος με πολλαπλές GPU.Αυτό φέρνει μια σειρά προκλήσεων. Μία από τις βασικές προκλήσεις είναι η παραλληλοποίηση των εργασιών. Παρακάτω παρατίθενται οι διαφορετικές μέθοδοι παραλληλοποίησης.

3.1 Μέθοδοι παραλληλοποίησης

Η βαθιά μηχανική μάθηση έρχεται με πολλές δυνατότητες παραλληλισμού. Παρακάτω παρατίθενται τρεις κύριες μέθοδοι παραλληλοποίησης , οι οποίες είναι των δεδομένων (Data Parallelism), του μοντέλο (Model Parallelism) και τον παραλληλισμό των pipeline.

3.1.1 Data Parallelism

Στον παραλληλισμό δεδομένων, ένας αριθμός εργαζομένων (μηχανές ή συσκευές, π.χ. GPU) φορτώνει ένα αντίγραφο του μοντέλου του νευρωνικού δικτύου. Τα δεδομένα εκπαίδευσης χωρίζονται σε μη επικαλυπτόμενα κομμάτια και τροφοδοτούνται στα αντίγραφα των εργατών (workers) για εκπαίδευση, όπως φαίνεται και στην παρακάτω εικόνα. Κάθε εργάτης πραγματοποιεί την εκπαίδευση στο κομμάτι των δεδομένων εκπαίδευσης του, το οποίο στην συνέχεια οδηγεί σε ενημερώσεις των παραμέτρων του μοντέλου του. Ως εκ τούτου, οι παράμετροι των μοντέλων των διαφορετικών εργαζομένων πρέπει να συγχρονιστούν μεταξύ τους. Υπάρχουν πολλές προκλήσεις στο πρόβλημα του συγχρονισμού παραμέτρων. Θα αναλυθούν αυτές οι προκλήσεις και προσεγγίσεις τελευταίας τεχνολογίας για την αντιμετώπισή τους στην Ενότητα 3.2.

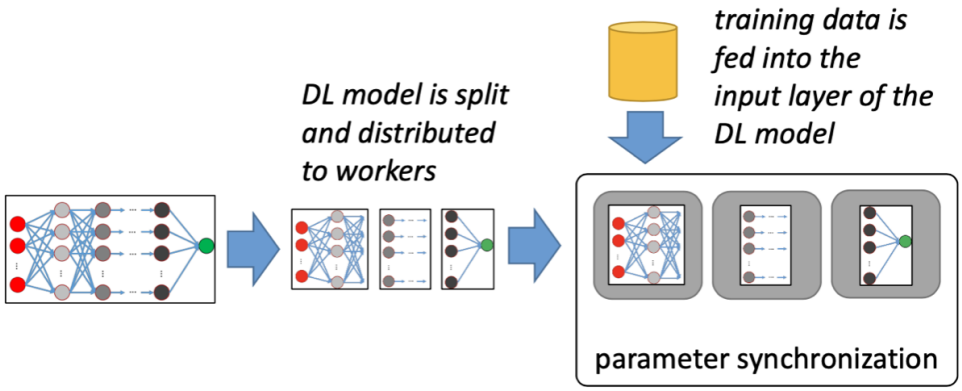


Εικόνα 3.1: Παραλληλισμός Δεδομένων.

Το κύριο πλεονέκτημα του παραλληλισμού δεδομένων είναι ότι μπορεί να εφαρμοστεί σε οποιαδήποτε αρχιτεκτονική μοντέλου νευρωνικού δικτύου χωρίς περαιτέρω γνώση και διαμόρφωση του μοντέλου αυτού. Κλιμακώνεται πολύ καλά για μοντέλα που απαιτούν μεγάλη υπολογιστική ισχύ, αλλά έχουν μόνο λίγες παραμέτρους, όπως τα Συνελικτικά Τροφοδοτικά Νευρωνικά Δίκτυα (CNN). Ωστόσο, ο παραλληλισμός δεδομένων είναι περιορισμένος για μοντέλα που έχουν πολλές παραμέτρους, καθώς ο συγχρονισμός παραμέτρων γίνεται το σημείο συμφόρησης (bottleneck) [24]. Αυτό το πρόβλημα θα μπορούσε να μετριαστεί χρησιμοποιώντας μεγαλύτερα μεγέθη batch size. Ωστόσο, αυτό αυξάνει την παλαιότητα (staleness) των δεδομένων μεταξύ των εργαζομένων και οδηγεί σε χειρότερη σύγκλιση του μοντέλου. Τέλος ένας άλλος περιορισμός της παραλληλοποίησης δεδομένων είναι ότι δεν βοηθά όταν το μέγεθος του μοντέλου είναι πολύ μεγάλο για να χωρέσει σε μία μόνο συσκευή.

3.1.2 Model Parallelism

Στον παραλληλισμό μοντέλων, το μοντέλο χωρίζεται και κάθε εργάτης φορτώνει ένα διαφορετικό μέρος του μοντέλου για εκπαίδευση, όπως φαίνεται στην παρακάτω εικόνα. Ο εργάτης που έχει το επίπεδο εισόδου του μοντέλου τροφοδοτείται με τα δεδομένα εκπαίδευσης. Στο μπροστινό πέρασμα (foreword pass), όπου τα δεδομένα ξεκινούν από την είσοδο και περνάνε από κάθε επίπεδο μέχρι να υπολογιστεί η έξοδος, το σήμα εξόδου υπολογίζεται αφού περάσει από κάθε εργάτη και συνεπώς κάθε επίπεδο του δικτύου. Αντίθετα στο back propagation, οι κλίσεις υπολογίζονται ξεκινώντας από τους εργάτες που συγκρατούν το επίπεδο εξόδου του μοντέλου και διαδίδοντας στους εργάτες που κατέχουν τα επίπεδα εισόδου του μοντέλου.



Εικόνα 3.2: Παραλληλισμός Μοντέλου.

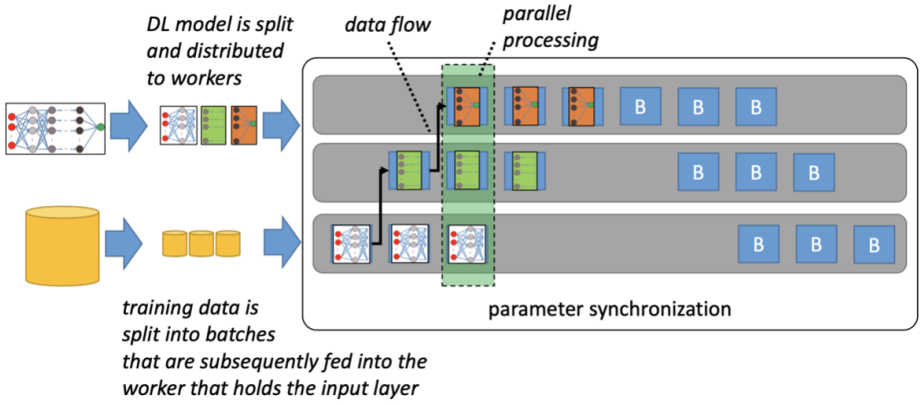
Μια σημαντική πρόκληση του παραλληλισμού μοντέλου είναι πώς να χωριστεί το μοντέλο σε ξεχωριστά κομμάτια που θα μοιραστούν στους παράλληλους εργάτες [25]. Μια γνωστή προσέγγιση για την εύρεση ενός καλού διαχωρισμού μοντέλου είναι η χρήση της ενισχυτικής μάθησης reinforcement learning [26]. Ξεκινώντας από κάποια αρχική διαίρεση, πραγματοποιούνται μεταθέσεις των επιπέδων ματαξύ των εργατών σε αυτήν τη διαίρεση και μετριέται η απόδοση (πχ για μία επανάληψη). Σε περίπτωση βελτίωσης, η μετάθεση διατηρείται και πραγματοποιούνται περαιτέρω μεταθέσεις, μέχρι να συγκλίνει το μοντέλο στην ζητούμενη απόδοση.

Το κύριο πλεονέκτημα του παραλληλισμού μοντέλου είναι η μειωμένη χρήση της μνήμης. Καθώς το μοντέλο χωρίζεται, απαιτείται λιγότερη μνήμη για κάθε εργάτη. Αυτό είναι χρήσιμο όταν το πλήρες μοντέλο είναι πολύ μεγάλο για να χωρέσει σε μία μόνο συσκευή. Τα μειονεκτήματα του παραλληλισμού μοντέλου είναι η βαριά επικοινωνία που απαιτείται μεταξύ των εργατών. Δεδομένου ότι τα μεγάλα νευρωνικά μοντέλα βαθιάς μηχανικής μάθησης είναι δύσκολο να διαχωριστούν αποτελεσματικά, ενδέχεται να υπάρξει μείωση αποδοτικότητας στους εργάτες λόγω επιβαρύνσεων επικοινωνίας και καθυστερήσεων συγχρονισμού. Ως εκ τούτου, η αύξηση του βαθμού παραλληλισμού μοντέλου δεν οδηγεί απαραίτητα σε επιτάχυνση της εκπαίδευσης.

3.1.3 Pipeline Parallelism

Ο παραλληλισμός pipeline συνδυάζει τον παραλληλισμό μοντέλου με τον παραλληλισμό δεδομένων. Σε παραλληλισμό pipeline, το μοντέλο χωρίζεται και κάθε εργάτης φορτώνει ένα διαφορετικό μέρος του μοντέλου για εκπαίδευση, όπως φαίνεται στο σχήμα. Επιπλέον, τα δεδομένα εκπαίδευσης χωρίζονται σε μικρο - παρτίδες. Τώρα, κάθε εργάτης υπολογίζει σήματα εξόδου για ένα σύνολο μικρο-παρτίδων, μεταδίδοντάς τα αμέσως στους επόμενους εργάτες. Με τον ίδιο τρόπο, στο back-propagation, οι εργάτες υπολογίζουν κλίσεις για το κομμάτι του μοντέλου τους για πολλαπλές μικρο-παρτίδες, προωθώντας τις αμέσως σε προηγούμενους εργαζόμενους. Με την

παράλληλη ροή πολλαπλών μικρο-παρτίδων μέσω του foreword και το back propagation περάσματος, η αποδοτικότητα των εργαζομένων μπορεί να αυξηθεί σημαντικά σε σύγκριση με τον καθαρό παραλληλισμό μοντέλων, όπου γίνεται επεξεργασία μόνο μιας παρτίδας κάθε φορά. Ταυτόχρονα, διατηρούνται τα πλεονεκτήματα του παραλληλισμού μοντέλου, καθώς ένας εργαζόμενος δεν χρειάζεται να κρατάει το πλήρες μοντέλο στην μνήμη του.



Εικόνα 3.3: Παραλληλισμός Pipepline

3.2 Ανάλυση Data Parallelism

Όπως αναφέρθηκε και προηγουμένως η μεγαλύτερη πρόκληση στον παραλληλισμό δεδομένων είναι στον συγχρονισμό των παραμέτρων. Ο συγχρονισμός των παραμέτρων για συστήματα παραλληλοποίησης δεδομένων βαθιάς μηχανικής μάθησης έχει δύο προκλήσεις. Η πρώτη πρόκληση σχετίζεται με το πότε θα γίνεται ο συγχρονισμός των παραμέτρων. Ο συγχρονισμός θα γίνεται μετά από κάθε batch ή θα δίνουμε μεγαλύτερη αυτονομία στους εργάτες με κίνδυνο να δουλέψουν με μη ανανεωμένες παραμέτρους· Η δεύτερη αφορά το πως θα γίνει αυτός ο συγχρονισμός. Θα ακολουθηθεί συγκεντρωτική ή αποκεντρωμένη αρχιτεκτονική·

3.2.1 Συγχρονισμός

Το ερώτημα πότε πρέπει να συγχρονιστούν οι παράμετροι μεταξύ των παράλληλων εργατών έχει απασχολήσει αρκετά την επιστημονική κοινότητα. Η κύρια πρόκληση στο συγχρονισμό παραμέτρων είναι να βρεθεί η ισορροπία μεταξύ ποιότητας αποτελεσμάτων και ταχύτητας σύγκλισης. Υπάρχουν δύο κύριες προσεγγίσεις, η σύγχρονη και η ασύγχρονη εκπαίδευση.

Σύγχρονη Εκπαίδευση

Στην σύγχρονη εκπαίδευση μετά από κάθε επανάληψη (επεξεργασία batch) οι εργάτες συγχρονίζουν τις παραμέτρους τους. Το πλεονέκτημα αυτού του αυστηρού συγχρονισμού είναι ότι η απόδειξη σύγκλισής του μοντέλου είναι εύκολη. Ωστόσο, ο αυστηρός συγχρονισμός καθιστά τη διαδικασία επιρρεπή στο πρόβλημα του αργού εργάτη, όπου ο πιο αργός εργάτης επιβραδύνει όλους τους άλλους [27]. Συνεπώς η σύγχρονη εκπαίδευση είναι ιδιαίτερα κατάλληλη για παράλληλη εκπαίδευση σε έναν υπολογιστικό κόμβο πολλαπλών GPU ή σε ένα cluster με ίδια υπολογιστική ισχύ μεταξύ των μηχανημάτων, όπου οι καθυστερήσεις στην επικοινωνία είναι μικρές και το υπολογιστικό φορτίο είναι ισορροπημένο.

Ασύγχρονη Εκπαίδευση

Στην ασύγχρονη εκπαίδευση, οι εργάτες ενημερώνουν το μοντέλο εντελώς ανεξάρτητα ο ένας από τον άλλο. Αλλά δεν υπάρχουν εγγυήσεις για το αν οι εργάτες έχουν κάθε στιγμή το τελευταία ενημερωμένο μοντέλο, δηλαδή, ένας εργάτης μπορεί να κάνει την εκπαίδευση σε ένα μη ενημερωμένο μοντέλο και στην συνέχεια να στείλει τις ενημερωμένες παραμέτρους του. Αυτό καθιστά δύσκολο την μαθηματική απόδειξη για τη σύγκλιση του μοντέλου και έχει επιπτώσεις στην απόδοση του μοντέλου. Ωστόσο, από την άλλη πλευρά, παρέχει στους εργάτες τη μεγαλύτερη δυνατή ευελιξία στη διαδικασία εκπαίδευσης τους, αποφεύγοντας τελείως το πρόβλημα του αργού εργάτη.

Υβριδική Εκπαίδευση

Τα δύο βασικά κατανεμημένα πρωτόκολλα συγχρονισμού που διατυπώθηκαν έχουν κάποιους περιορισμούς, η σύγχρονη εκπαίδευση τείνει να επιβραδύνει την διαδικασία εκπαίδευσης, επειδή οι εργάτες πρέπει να περιμένουν να συγχρονιστούν ενώ η ασύγχρονη εκπαίδευση υποφέρει από μειωμένη ακρίβεια λόγω της εκπαίδευσης σε μη ενημερωμένες παραμέτρους. Με το υβριδικό μοντέλο συγχρονισμού γίνεται προσπάθεια για την δημιουργία ενός τρόπου συγχρονισμού κατανεμημένη εκπαίδευσης όπου θα εκμεταλλεύεται τα θετικά στοιχεία τόσο της σύγχρονης όσο και της ασύγχρονης εκπαίδευσης. Ένα τέτοιο μοντέλο δοκιμάστικε από τον Shijian Li στην εργασία [28]. Στην εργασία αυτή προτείνεται η εναλλαγή του συγχρονισμού κατα την διάρκεια της εκπαίδευσης. Αρχικά το μοντέλο εκπαιδεύεται σύγχρονα και μετά από έναν αριθμό εποχών το μοντέλο αλλάζει τρόπο συγχρονισμού και εκπαιδεύεται ασύγχρονα. Στο υβριδικό μοντέλο συγχρονισμού παρατηρούνται πολύ θετικά αποτελέσματα καθώς βελτιώνεται σε μεγάλο βαθμό ο χρόνος εκπαίδευσης αλλά και η απόδοση του μοντέλου.

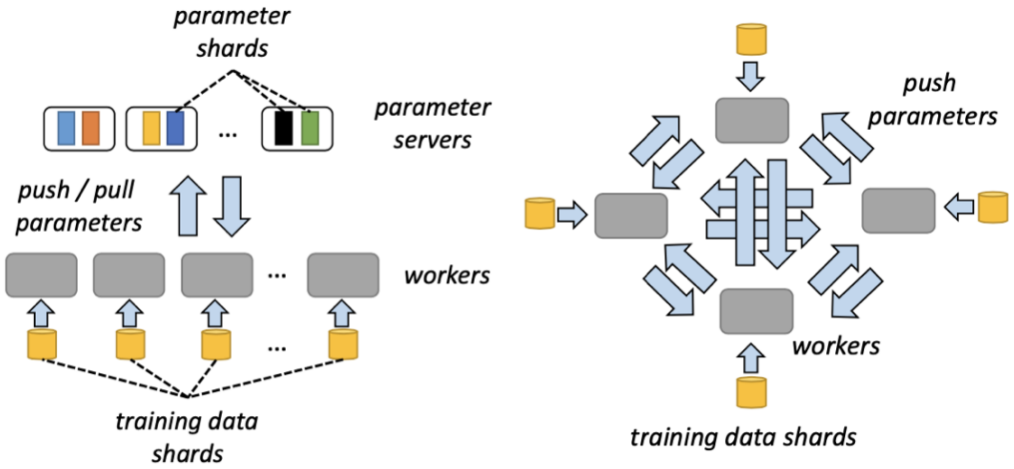
Άλλες στρατηγικές συγχρονισμού

Όπως έχει αναφερθεί το κομμάτι του συγχρονισμού έχει απασχολήσει πολύ και υπάρχουν πολλές και διάφορες άλλες στρατηγικές που έχουν προταθεί. Μία γνωστή στρατηγική είναι η Stale Synchronous Parallel (SSP) [29] όπου στην συγκεκριμένη

οι εργάτες τρέχουν ασύγχρονα χωρίς όμως να επιτρέπεται να ξεπεράσουν ένα όριο. Θέτοντας αυτό το όριο περιορίζεται το πρόβλημα εκπαίδευσης σε μη ενημερωμένα δεδομένα και επιτυγχάνεται καλύτερη σύγκληση. Προέκταση αυτής της στρατηγικής προτάθηκε πρόσφατα μία άλλη στρατηγική η Dynamic Stale Synchronous Parallel (DSSP) [30]. Στην προκειμένη περίπτωση το όριο αλλάζει δυναμικά κατα την διάρκεια της εκπαίδευσης σύμφωνα με το processing time των εργατών και έχει σαν αποτέλεσμα βελτιωμένες επιδόσεις τόσο χρονικά όσο και σε επίπεδο απόδοσης. Μία διαφορετική στρατηγική προτείνεται από τον Victor Campos [31] η mixed strategy όπου δημιουργούνται υποομάδες εργατών όπου δουλεύουν σύγχρονα ενώ ο συγχρονισμός μεταξύ των υποομάδων γίνεται ασύγχρονα. Για την επιλογή αυτή λαμβάνει υπόψη του το hardware, εργάτες που έχουν παρόμοια υπολογιστική ισχύ μπορούν να ενταχθούν στην ίδια υποομάδα.

3.2.2 Κατανεμημένες Αρχιτεκτονικές Εκπαίδευσης

Η αρχιτεκτονική συστήματος περιγράφει το πως οι παράμετροι μεταξύ των διαφορετικών εργατών θα συγχρονίζονται. Ο στόχος είναι να παραχθεί ένα σύστημα αρχιτεκτονικής με δυνατότητα μεγάλης κλιμάκωσης όπου θα διαχειρίζεται μεγάλο αριθμό παράλληλων εργατών που ενημερώνουν τακτικά το μοντέλο DL καθώς και λαμβάνουν ενημερωμένη την εικόνα του μοντέλου για περαιτέρω εκπαίδευση.



(α′) Parameter Server

(β′) All-reduce

Εικόνα 3.4: Κατανεμημένες Αρχιτεκτονικές Εκπαίδευσης

Paramater Server Αρχιτεκτονική

Στην συγκεντρωτική αρχιτεκτονική Parameter Server υπάρχουν ένας ή πολλοί parameter servers όπου έχουν αποθηκευμένο το ενημερωμένο μοντέλο (εικόνα 3.4α′). Ο σχεδιασμός αυτής τις αρχιτεκτονικής είναι εμπνευσμένος από αρχιτεκτονικές σαν

την blackboard architecture [32] και την MapReduce [33]. Η αρχιτεκτονική του PS είναι η πιο δημοφιλής για συστήματα DL παράλληλων δεδομένων. Κοινή τακτική αποτελεί το μοίρασμα και ο τεμαχισμός των παραμέτρων του μοντέλου σε πολλούς PS οι οποίοι ανανεώνουν τις παραμέτρους τους παράλληλα. Οι εργάτες πριν από κάθε βήμα εκπαίδευσης παίρνουν τις ενημερωμένες παραμέτρους από τους PS και όταν ολοκληρώσουν την εκπαίδευση επιστρέφουν πίσω στους PS τις κλίσεις των παραμέτρων για την εκ νέου ενημέρωση τους.

All-reduce Αρχιτεκτονική

Η All-reduce λειτουργεί χωρίς parameter servers. Αντιθέτως οι εργάτες ανταλλάσουν απευθείας μεταξύ τους τις ανανεωμένες παραμέτρους μέσω της διαδικασίας All-reduce (εικόνα 3.4β′). Με αυτόν τον τρόπο λειτουργίας του συστήματος, η τοπολογία των εργατών παίζει σημαντικό ρόλο. Ένα πλήρως συνδεδεμένο δίκτυο, όπου κάθε εργάτης επικοινωνεί με όλους τους άλλους, η επικοινωνία έχει κόστος $O(n^2)$ με n εργαζόμενους, συνεπώς η επικοινωνία μπορεί να αποτελέσει σημείο συμφόρησης (bottleneck) για συστήματα με πολλούς εργάτες. Η πιο κοινή εναλλακτική λύση είναι να χρησιμοποιηθεί μια τοπολογία δακτυλίου (ring all-reduce) η οποία αναλύεται στο τέλος της υποενότητας αυτής. Η ερευνητική ομάδα της Baidu ήταν από τις πρώτες που πρότειναν τη χρήση του ring all-reduce για εκπαίδευση παράλληλων δεδομένων DL [34]. Άλλες τοπολογίες που έχουν προταθεί είναι "Butterfly" [35] και δεντρο [36]. Το κύριο μειονέκτημα των εναλλακτικών τοπολογιών, διάφορων του πλήρως συνδεδεμένου δικτύου, είναι ότι η διάδοση των ενημερώσεων των παραμέτρων σε όλους τους εργαζόμενους χρειάζεται περισσότερο χρόνο, καθώς μπορεί να υπάρχουν πολλοί ενδιάμεσοι κόμβοι μεταξύ ενός ζεύγους εργατών.

Σύγκριση Αρχιτεκτονικών

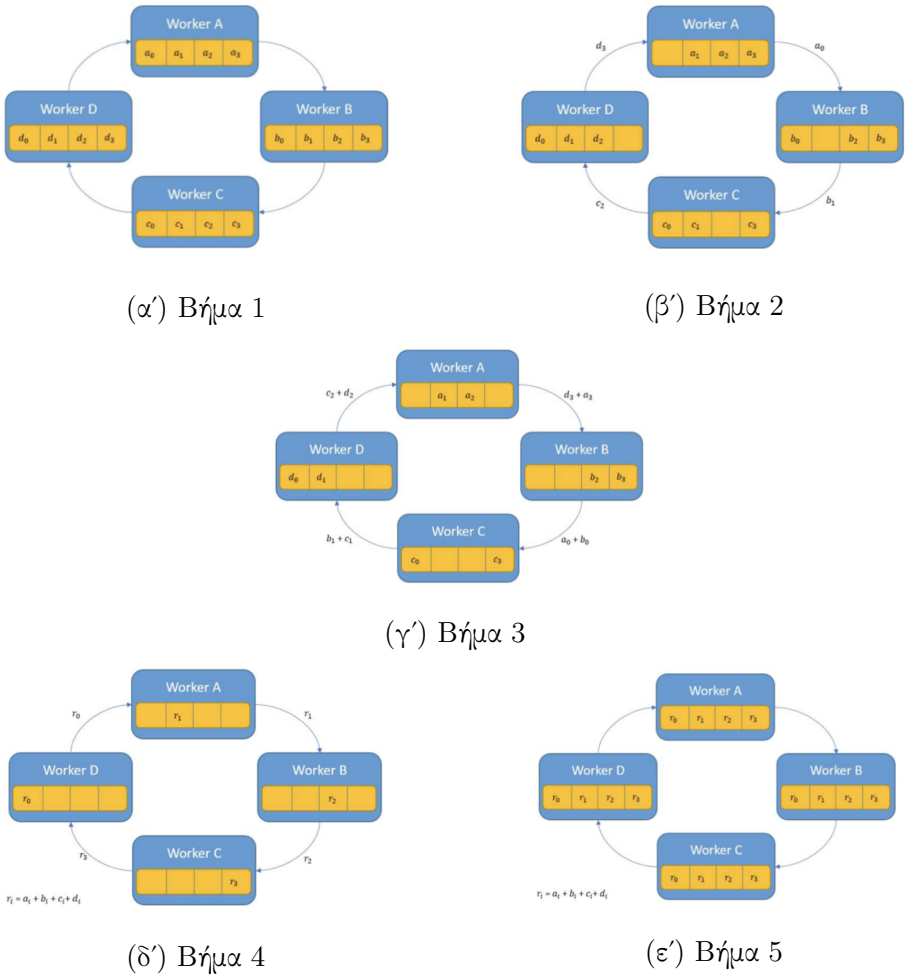
Τα πλεονεκτήματα της αποκεντρωμένης αρχιτεκτονικής All-reduce σε σχέση με την συγκεντρωτική αρχιτεκτονική parameter server συνοψίζονται στα εξής. Χρησιμοποιώντας την αρχιτεκτονική All-reduce , αποφεύγεται η ανάγκη υλοποίησης ενός πιο σύνθετου συστήματος και ο καθορισμός διαμοιρασμού των πόρων μεταξύ των parameter server και των εργατών. Ένα επιπλέον πλεονέκτημα είναι ότι η ανοχή σε σφάλματα μπορεί να επιτευχθεί ευκολότερα, επειδή δεν υπάρχει κόμβος που άμα πάψει να λειτουργεί διακόπτεται η λειτουργία όπως είναι ο parameter server. Όταν ένας κόμβος στην All-reduce αρχιτεκτονική αποτύχει, οι άλλοι κόμβοι μπορούν εύκολα να διαμοιράσουν και να αναλάβουν τον φόρτο εργασίας του και η εκπαίδευση να συνεχιστεί χωρίς διακοπή. Τέλος δεν είναι απαραίτητη η συχνή αποθήκευση μεγάλων σημείων ελέγχου για την επαναφορά του συστήματος σε περίπτωση σφάλματος όπως συμβαίνει με τον PS .

Η αρχιτεκτονική All-reduce έχει επίσης μειονεκτήματα. Πρώτον και κυριότερον, η επικοινωνία στην αρχιτεκτονική All-reduce αυξάνεται τετραγωνικά με τον αριθμό των εργαζομένων, εάν δεν ληφθούν αντίμετρα. Όπως συζητήθηκε παραπάνω, αυτά τα αντίμετρα, όπως η αλλαγή της τοπολογίας, προκαλούν νέες πολυπλοκότητες και

πιθανές καθυστερήσεις στην επικοινωνία. Συνοψίζοντας δεν υπάρχει μία ξεκάθαρα καλύτερη αρχιτεκτονική για τον τρόπο συγχρονισμού παραμέτρων σε συστήματα βαθιάς μηχανικής μάθησης παραλληλοποίησης δεδομένων.

Ring All-reduce

Η εφαρμογή του Ring All-reduce έχει δύο φάσεις. Η πρώτη φάση, είναι η share-reduce και η δεύτερη φάση είναι η share-only. Στη φάση share-reduce, κάθε διαδικασία p στέλνει δεδομένα στη διαδικασία $(p+1) \mod p$. Έτσι, η διαδικασία A θα σταλεί στη διαδικασία B, κ.λ.π. Αυτό δημιουργεί τη σύνδεση που μοιάζει με δακτύλιο. Επιπλέον, ο πίνακας δεδομένων μήκους ν διαιρείται με p, και κάθε ένα από αυτά τα κομμάτια έχει τον δικό του δείκτη i. Στην παρακάτω εικόνα 3.5α΄ βλέπουμε την αρχική κατάσταση.



(α΄) Βήμα 1

(β΄) Βήμα 2

(γ΄) Βήμα 3

(δ΄) Βήμα 4

(ε΄) Βήμα 5

Εικόνα 3.5: Αλγόριθμος Ring All-reduce

Το πρώτο βήμα share-reduce θα είχε τη διεργασία A να στείλει το a_0 στη διεργασία B, τη διεργασία B θα στείλει το b_1 στη διεργασία C (εικόνα 3.5β΄). Στη συνέχεια, όταν κάθε διεργασία λαμβάνει τα δεδομένα από την προηγούμενη διαδικασία, τότε εφαρμόζει τον τελεστή reduce μεταξύ του κομματιού που έλαβε και του αντίστοιχου κομματιού όπου έχει η διεργασία και στη συνέχεια στέλνει το αποτέλεσμα αυτό στην επόμενη διεργασία (εικόνα 3.5γ΄). Είναι επίσης σημαντικό ο τελεστής reduce να είναι προσεταιριστικός και αντιμεταθετικός, διαφορετικά δεν θα γινόταν να μεταφερθεί το ενδιάμεσο αποτέλεσμα από διεργασία σε διεργασία. Η φάση share-reduce τελειώνει όταν κάθε διεργασία διατηρεί τα πλήρη αποτέλεσμα του τεμαχίου i. (εικόνα 3.5δ΄) Το δεύτερο βήμα, το βήμα share-only είναι πολύ απλό, είναι ακριβώς η ίδια διαδικασία κοινής χρήσης δεδομένων με δακτύλιο χωρίς εφαρμογή της λειτουργίας reduce. Έτσι όλες οι διεργασίες έχουν το τελικό αποτέλεσμα όλων των τεμαχίων (εικόνα 3.5ε΄).

3.3 TensorFlow

Το TensorFlow [9] είναι μια διεπαφή για την έκφραση, δημιουργία, υλοποίηση και εκτέλεση αλγορίθμων μηχανικής μάθησης. Ένας αλγόριθμος που εκφράζεται με το TensorFlow μπορεί να εκτελεστεί με μικρή ή καθόλου αλλαγή σε μια μεγάλη ποικιλία ετερογενών συστημάτων, που κυμαίνονται από κινητές συσκευές όπως τηλέφωνα και tablet έως κατανεμημένα συστήματα εκατοντάδων μηχανών μεγάλης κλίμακας που αποτελούνται από χιλιάδες υπολογιστικές συσκευές όπως κάρτες GPU. Το σύστημα είναι ευέλικτο και μπορεί να χρησιμοποιηθεί για να εκφράσει μια μεγάλη ποικιλία αλγορίθμων, συμπεριλαμβανομένων των αλγορίθμων εκπαίδευσης για μοντέλα νευρωνικών δικτύων, και έχει χρησιμοποιηθεί για τη διεξαγωγή έρευνας και για την ανάπτυξη συστημάτων μηχανικής μάθησης για την αγορά εργασίας. Έχει δοκιμαστεί και χρησιμοποιηθεί σε δεκάδες επιστημονικά πεδία όπως η αναγνώριση ομιλίας, η όραση υπολογιστή, η ρομποτική, η ανάκτηση πληροφοριών, η επεξεργασία φυσικής γλώσσας, η εξαγωγή γεωγραφικών πληροφοριών και η υπολογιστική ανακάλυψη φαρμάκων.

Το TensorFlow ξεκίνησε την ανάπτυξη του το 2011 ως έργο της Google Brain για να εξερευνήσει την χρήση πολύ μεγάλης κλίμακας νευρωνικών δικτύων, τόσο για έρευνα όσο και για χρήση στα προϊόντα της Google. Από τον Νοέμβριο του 2015 το API TensorFlow κυκλοφόρησε ως πακέτο ανοιχτού κώδικα υπό την άδεια Apache 2.0.

3.3.1 Μοντέλο προγραμματισμού και βασικές έννοιες

Δομή

Ένας υπολογισμός TensorFlow περιγράφεται από ένα κατευθυνόμενο γράφημα, το οποίο αποτελείται από ένα σύνολο κόμβων και ακμών. Το γράφημα αντιπροσωπεύει έναν υπολογισμό ροής δεδομένων, με επεκτάσεις που επιτρέπουν σε ορισμένα είδη κόμβων να διατηρούν και να ενημερώνουν την κατάσταση τους και ενδέχεται

να έχουν διακλαδώσεις και βρόχους. Το TensorFlow υποστηρίζει στο frontend τις γλώσσες Python και C++ αλλά το backend είναι σε C++ για λόγους απόδοσης. Ένα παράδειγμα για την κατασκευή και στη συνέχεια την εκτέλεση ενός γραφήματος TensorFlow χρησιμοποιώντας την Python φαίνεται στο εικόνα 3.6, και το προκύπτον γράφημα υπολογισμού στην εικόνα 3.7.

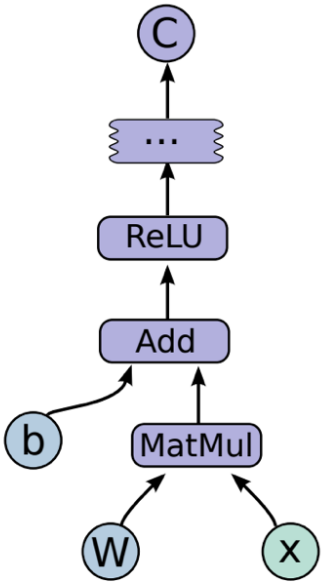
```
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))                    # 100-d vector, init to zeroes
W = tf.Variable(tf.random_uniform([784,100],-1,1)) # 784x100 matrix w/rnd vals
x = tf.placeholder(name="x")                        # Placeholder for input
relu = tf.nn.relu(tf.matmul(W, x) + b)              # Relu(Wx+b)
C = [...]                                           # Cost computed as a function
                                                    # of Relu

s = tf.Session()
for step in xrange(0, 10):
  input = ...construct 100-D input array ...        # Create 100-d vector for input
  result = s.run(C, feed_dict={x: input})           # Fetch cost, feeding x=input
  print step, result
```

Εικόνα 3.6: Παράδειγμα κώδικα που παράγει γράφο.

Σε ένα γράφημα TensorFlow, κάθε κόμβος έχει μηδενικές ή περισσότερες εισόδους και μηδέν ή περισσότερες εξόδους, και αντιπροσωπεύει μία λειτουργία. Οι τιμές που ρέουν κατά μήκος των φυσιολογικών άκρων στο γράφημα (από τις εξόδους στις εισόδους) είναι τανυστές, πίνακες αυθαίρετων διαστάσεων όπου ο τύπος τους καθορίζεται ή συνάγεται στον χρόνο κατασκευής του γραφήματος. Ειδικές ακμές, που ονομάζονται εξαρτήσεις ελέγχου (control dependencies), μπορούν επίσης να υπάρχουν στο γράφημα: δεν ρέουν δεδομένα κατά μήκος αυτών των άκρων, αλλά υποδεικνύουν ότι ο κόμβος προέλευσης για την ακμή εξάρτησης ελέγχου πρέπει να ολοκληρώσει την εκτέλεση πριν ξεκινήσει η εκτέλεση του κόμβου προορισμού της ακμής αυτής. Αυτές οι ακμές εξάρτησης ελέγχου είναι προσβάσιμες και στους χρήστες (clients) σε περίπτωση που θέλουν να δημιουργήσουν χρονικές εξαρτήσεις μεταξύ δύο λειτουργιών. Τέλος αυτές οι ακμές μπορεί να χρησιμοποιηθούν από το TensorFlow και για λόγους αποδοτικότητας για παράδειγμα, για τον έλεγχο της μέγιστης χρήσης της μνήμης.



Εικόνα 3.7: Παράδειγμα γράφου

Sessions

Τα προγράμματα των χρηστών αλληλεπιδρούν με το σύστημα TensorFlow δημιουργώντας ένα Sessions. Για την δημιουργία ενός γραφήματος υπολογισμού, η διεπαφή του Sessions υποστηρίζει μια μέθοδο επέκτασης για να αυξήσετε το τρέχον γράφημα που διαχειρίζεται το Sessions και να προστεθούν επιπλέον κόμβοι και ακμές (το αρχικό γράφημα όταν δημιουργείται ένα Sessions είναι κενό). Η κύρια λειτουργία που υποστηρίζεται από τη διεπαφή Sessions είναι η Run , η οποία λαμβάνει ένα σύνολο ονομάτων εξόδου που πρέπει να υπολογιστούν, καθώς και ένα προαιρετικό σύνολο τανυστών για να εισαχθούν στο γράφημα σαν είσοδοι σε κάποιον κόμβο. Χρησιμοποιώντας τους τανυστές ορίσματα της Run , η εφαρμογή TensorFlow μπορεί πλέον να υπολογίσει το μεταβατικό κλείσιμο όλων των κόμβων που πρέπει να εκτελεστούν για τον υπολογισμό των εξόδων που ζητήθηκαν και στη συνέχεια να κανονίσει την εκτέλεση των κατάλληλων κόμβων με μια σειρά που σέβεται τις εξαρτήσεις τους. Στις περισσότερες περιπτώσης χρήσης του TensorFlow δημιουργείται ένας γράφος τον οποίο τον αναθέτουμε σε ένα Sessions και στη συνέχεια, εκτελείται ολόκληρος ο γράφος ή μερικοί υπογράφοι χιλιάδες ή εκατομμύρια φορές μέσω κλήσεων της εντολής Run .

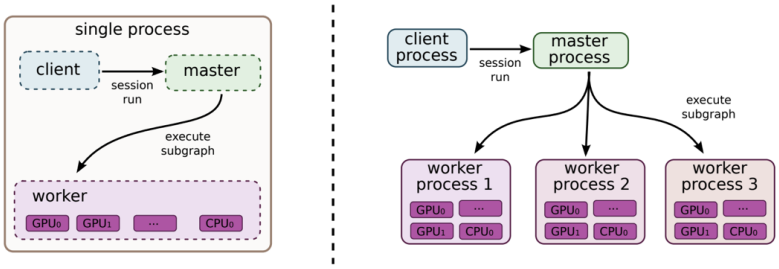
Variables

Τις περισσότερες φορές όπως αναφέραμε και προηγούμενος ένα γράφημα εκτελείται πολλές φορές. Οι περισσότεροι τανυστές δεν επιβιώνουν μετά από μία εκτέλεση

του γραφήματος. Υπάρχει όμως μια λειτουργία ειδικού τύπου, που ονομάζεται μεταβλητή (Variables), η οποία επιστρέφει μία αναφορά σε έναν τανυστή του οποίου η τιμή μπορεί να διατηρηθεί αλλά και να ανανεωθεί μεταξύ των εκτελέσεων του γράφου. Στην εφαρμογές μηχανικής μάθησης στο TensorFlow οι μεταβλητές χρησιμοποιούνται για να αποθηκεύονται η παράμετροι των μοντέλων.

Εκτέλεση

Τα κύρια στοιχεία σε ένα σύστημα TensorFlow είναι ο χρήστης (client), ο οποίος χρησιμοποιεί τη διεπαφή Sessions για επικοινωνία με τον master και μία ή περισσότερες διαδικασίες workers , με κάθε μια υπεύθυνη για την πρόσβαση σε μία ή περισσότερες υπολογιστικές συσκευές (CPU ή GPU). Ο master δίνει τις οδηγίες και οι workers υπολογίζουν τους κόμβους του γράφου που τους έχουν ανατεθεί. Υπάρχουν τόσο τοπικές όσο και κατανεμημένες υλοποιήσεις της διεπαφής TensorFlow . Η τοπική εφαρμογή χρησιμοποιείται όταν ο χρήστης (client) , ο master και ο worker λειτουργούν σε ένα μόνο μηχάνημα στο πλαίσιο ενός λειτουργικού συστήματος. Η κατανεμημένη υλοποίηση μοιράζεται το μεγαλύτερο μέρος του κώδικα με την τοπική υλοποίηση, αλλά τον επεκτείνει αποτέλεσμα να υποστηρίζει ένα περιβάλλον όπου ο χρήστης, ο master και οι workers μπορούν να βρίσκονται σε διαφορετικές διεργασίες σε διαφορετικά μηχανήματα.



Εικόνα 3.8: Παράδειγμα απλής και κανεμημένης εκτέλεσης TensorFlow

Υπολογισμός Κλίσης

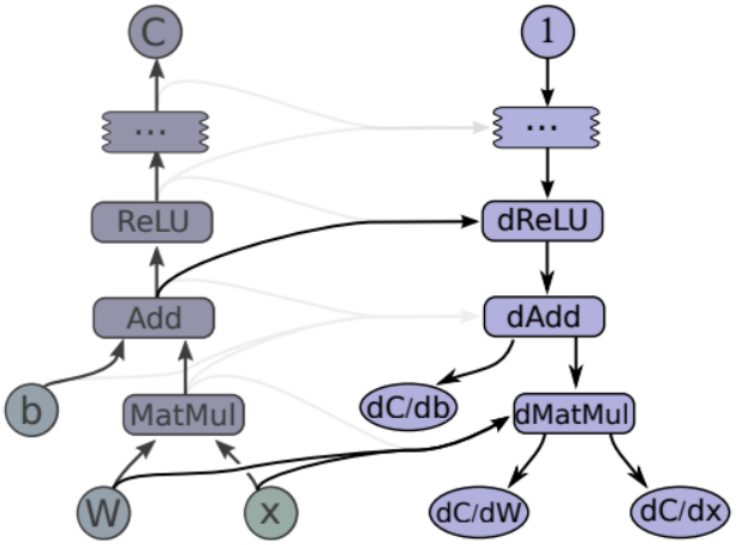
Στους περισσότερους αλγορίθμους μηχανικής μάθησης η εκπαίδευση γίνεται με κάποιον αλγόριθμο στοχαστικής επαναληπτικής βελτιστοποίησης όπως ο Stochastic Giadient Descent (SGD), οι οποίοι υπολογίζουν την παράγωγο της συνάρτησης κόστους ως προς τις εισόδους. Επειδή αυτό αποτελεί μία τόσο κοινή ανάγκη στην μηχανική μάθηση, το TensorFlow διαθέτει ενσωματωμένη υποστήριξη για αυτόματο υπολογισμό παραγώγου. Εάν ένας τανυστής C σε ένα γράφημα TensorFlow εξαρτάται, ίσως μέσω ενός σύνθετου υπογράφου λειτουργιών, από ένα σύνολο τανυστών X_k, τότε υπάρχει μια ενσωματωμένη συνάρτηση στο TensorFlow που θα επιστρέψει

τους τανυστές $\frac{dC}{dX_k}$. Οι τανυστές των παραγώγων αυτών υπολογίζονται όπως όλοι οι άλλοι τανυστές επεκτείνοντας το γράφημα TensorFlow , χρησιμοποιώντας την ακόλουθη διαδικασία. Όταν το TensorFlow χρειάζεται να υπολογίσει τη κλίση ενός τανυστή C σε σχέση με κάποιον τανυστή I από τον οποίο εξαρτάται το C , βρίσκει πρώτα τη διαδρομή στο γράφημα υπολογισμού από I έως C . Στη συνέχεια κάνει την αντίστροφη διαδρομή από το C στο I και για κάθε λειτουργία στην διαδρομή προσθέτει έναν κόμβο στο γράφημα TensorFlow , συνθέτοντας τις μερικές παραγώγους κατά μήκος της διαδρομής προς τα πίσω χρησιμοποιώντας τον κανόνα της αλυσίδας. Ο κόμβος που προστέθηκε υπολογίζει τη παράγωγο για την αντίστοιχη λειτουργία στην εμπρόσθια διαδρομή.

Στο παρακάτω εικόνα 3.10 φαίνεται η επέκταση του γράφου όταν ζητείται να υπολογιστούν οι κλίσεις των συναντήσεων εικόνα 3.9 .

```
[db,dW,dx] = tf.gradients(C, [b,W,x])
```

Εικόνα 3.9: Παράδειγμα κώδικα υπολογισμού των κλίσεων στο TensorFlow



Εικόνα 3.10: Παράδειγμα γράφου υπολογισμού των κλίσεων στο TensorFlow

3.3.2 TensorFlow 2

Το TensorFlow 2 επικεντρώνεται στην απλότητα και την ευχρηστία με ενημερώσεις όπως το eager execution, ενσωμάτωση υψηλού επιπέδου TensorFlow Keras API και ευέλικτη δημιουργία μοντέλων σε οποιαδήποτε πλατφόρμα.

Eager execution

Το eager execution του TensorFlow είναι ένα προστακτικό περιβάλλον προγραμματισμού που υπολογίζει τις εντολές και λειτουργίες αμέσως, χωρίς να δημιουργεί γραφήματα: οι λειτουργίες επιστρέφουν συγκεκριμένες τιμές αντί να κατασκευάζουν ένα υπολογιστικό γράφημα για να εκτελεστεί αργότερα. Αυτό καθιστά εύκολο την δημιουργία νέων μοντέλων νευρωνικών δικτύων στο TensorFlow καθώς και στον εντοπισμό σφαλμάτων.

Το eager execution είναι μια ευέλικτη πλατφόρμα μηχανικής μάθησης για έρευνα και πειραματισμό, παρέχοντας:

- Εύκολο στην χρήση interface: Δομώντας τον κώδικα σου όπως σε μία προστακτική γλώσσα χρησιμοποιώντας τις δομές δεδομένων της Python. Προσφέροντας γρήγορες δοκιμές σε μικρά μοντέλα με λίγα δεδομένα.
- Ευκολότερο εντοπισμό σφαλμάτων: Δίνει την δυνατότητα απευθείας κλίσης των λειτουργιών για έλεγχο των μοντέλων την στιγμή της κατασκευής και έλεγχο για δοκιμαστικές αλλαγές.
- Φυσική ροή ελέγχου - Χρησιμοποιεί την ροή ελέγχου της Python αντί για την ροή ελέγχου γραφήματος, απλοποιώντας έτσι τις προδιαγραφές δυναμικών μοντέλων.

tf.function

Στο TensorFlow 2, το eager execution είναι σαν προεπιλογή ενεργοποιημένο. Η διεπαφή χρήστη είναι εύχρηστη και ευέλικτη (η εκτέλεση των λειτουργιών είναι πολύ πιο εύκολη και ταχύτερη), αλλά αυτό μπορεί να έρθει σε βάρος της απόδοσης και της ανάπτυξης. Γι' αυτό κατασκευάστηκε η tf.function η οποία δημιουργεί γραφήματα από τον κώδικα όπου έχει γραφτεί. Είναι ένα εργαλείο μετασχηματισμού που δημιουργεί γραφήματα ροής δεδομένων από τον κώδικα Python.

TensorFlow Keras API

Το Keras είναι ένα API ειδικά σχεδιασμένο για να προσφέρει ευχρηστία και παραγωγικότητα. Ακολουθεί τις βέλτιστες πρακτικές για τη μείωση του γνωστικού φορτίου: προσφέρει συνεπή και απλά API , ελαχιστοποιεί τον αριθμό των ενεργειών του χρήστη που απαιτούνται για την δημιουργία των πιο κοινών περιπτώσεων χρήσης και παρέχει σαφή και ενεργή μηνύματα σφάλματος. Διαθέτει επίσης εκτενή τεκμηρίωση

και οδηγούς. Το Keras είναι το πιο χρησιμοποιημένο framework βαθιάς μάθησης γιατί διευκολύνει την εκτέλεση νέων πειραμάτων, δίνει τη δυνατότητα δοκιμής περισσότερων ιδεών πιο γρήγορα και εύκολα. Τέλος είναι στενά συνδεδεμένο με το οικοσύστημα του TensorFlow 2, που καλύπτει κάθε βήμα της ροής εργασίας της μηχανικής μάθησης, από τη διαχείριση δεδομένων έως την εκπαίδευση υπερπαραμέτρων.

Tensor Processing Unit (TPU)

Η TPU είναι ένα ολοκληρωμένο κύκλωμα, επιταχυντής τεχνητής νοημοσύνης, application-specific integrated circuit (ASIC) που αναπτύχθηκε από την Google ειδικά για μηχανική μάθηση νευρωνικών δικτύων, χρησιμοποιώντας το λογισμικό TensorFlow της Google. Η Google άρχισε να χρησιμοποιεί τις TPU το 2015 και το 2018 τις έκανε διαθέσιμες για χρήση από τρίτους, τόσο ως μέρος της υποδομής cloud όσο και προσφέροντας μια μικρότερη έκδοση του τσιπ προς πώληση. Το TPU είναι μια ειδικά διαμορφωμένη μονάδα επεξεργασίας για εκπαίδευση νευρωνικών δικτύων. Με την χρήση του TPU υπάρχει μεγάλη βελτίωση στον χρόνο εκπαίδευσης και ειδικότερα σε συνελικτικά νευρωνικά δίκτυα.

Remote Procedure Call (RPC)

Στα κατανεμημένα συστήματα, ένα remote procedure call (RPC) είναι όταν ένα πρόγραμμα υπολογιστή προκαλεί την εκτέλεση μιας διεργασίας σε διαφορετικό χώρο διευθύνσεων (συνήθως σε άλλο υπολογιστή σε κοινόχρηστο δίκτυο), η οποία καλείται σαν να ήταν μια κανονική (τοπική) διεργασία, χωρίς ο προγραμματιστής να ασχοληθεί και να προγραμματίζει ρητά τις λεπτομέρειες για την απομακρυσμένη αλληλεπίδραση. Δηλαδή, ο προγραμματιστής γράφει ουσιαστικά τον ίδιο κώδικα είτε η υπορουτίνα είναι τοπική στο πρόγραμμα εκτέλεσης είτε είναι απομακρυσμένη. Αυτή είναι μια μορφή αλληλεπίδρασης client-server (ο καλών είναι ο client, ο εκτελεστής είναι server), και συνήθως εφαρμόζεται μέσω ενός συστήματος μηνυμάτων request–response.

3.3.3 Κατανεμημένη Εκπαίδευση στο TensorFlow

Το tf.distribute.Strategy είναι ένα TensorFlow API για κατανεμημένη εκπαίδευση σε πολλά μηχανήματα CPU, GPU αλλά και TPU. Χρησιμοποιώντας αυτό το API, είναι εφικτή η κατανεμημένη εκπαίδευση των μοντέλων νευρωνικών δικτύων με ελάχιστες αλλαγές κώδικα.

Το API αυτό έχει σχεδιαστεί με βάση αυτούς τους 3 στόχους:

- Να είναι εύκολο στην χρήση και να μπορεί να υποστηρίξει πολλές ομάδες χρηστών, όπως ερευνητές, μηχανικούς μηχανικής μάθησης κλπ.
- Να παρέχει καλή απόδοση παρέχοντας έτοιμα υλοποιημένα κομμάτια κώδικα
- Να είναι εύκολη η εναλλαγή μεταξύ στρατηγικών κατανεμημένης εκπαίδευσης

Η χρήση του tf.distribute.Strategy γίνεται με πολύ λίγες αλλαγές στον κώδικα, επειδή τα υποκείμενα στοιχεία του TensorFlow, όπως οι μεταβλητές, τα επιπεδα, τα μοντέλα κλπ, έχουν αλλάξει ώστε να αντιλαμβάνονται για ποια στρατηγική τρέχουν και να δρουν αναλόγως.

Στο TensorFlow 2 είναι δυνατή η εκτέλεση τόσο με την μέθοδο του eager execution όσο και σε ένα γράφημα χρησιμοποιώντας την tf.function. Η tf.distribute.Strategy σκοπεύει να υποστηρίξει και τους δύο αυτούς τρόπους εκτέλεσης, αλλά λειτουργεί καλύτερα με τη tf.function.

Τύποι Στρατηγικών

Για να καλύψει όλα τα ενδεχόμενα το TensorFlow έχει 5 διαφορετικές στρατηγικές κατανεμημένης εκπαίδευσης οι οποίες παρατίθενται παρακάτω.

MirroredStrategy

Το tf.distribute.MirroredStrategy υποστηρίζει σύγχρονη κατανεμημένη εκπαίδευση σε πολλαπλές GPU σε ένα μηχάνημα. Δημιουργεί ένα αντίγραφο ανά συσκευή GPU. Κάθε μεταβλητή στο μοντέλο αντικατοπτρίζεται σε όλα τα αντίγραφα. Μαζί, αυτές οι μεταβλητές σχηματίζουν μια ενιαία εννοιολογική μεταβλητή που ονομάζεται MirroredVariable. Αυτές οι μεταβλητές διατηρούνται σε συγχρονισμό μεταξύ τους εφαρμόζοντας τις ίδιες ενημερώσεις. Για την σύγχρονη ενημέρωση των παραμέτρων χρησιμοποιούνται αποδοτικοί αλγόριθμοι all-reduce. Υπάρχουν πολλοί αλγόριθμοι all-reduce, ανάλογα με τον τύπο της διαθέσιμης επικοινωνίας μεταξύ συσκευών. Ως προεπιλογή, χρησιμοποιείται η NCCL από την NVIDIA Collective Communication Library αλλά μπορούν να επιλεχθούν και άλλες στρατηγικές all-reduce ή ακόμα και να γράψει κάποιος την δικιά του.

TPUStrategy

Η tf.distribute.TPUStrategy επιτρέπει την εκπαίδευση μοντέλων νευρωνικών δικτύων του TensorFlow στις μονάδες επεξεργασίας Tensor Processing Units (TPU). Όσον αφορά την αρχιτεκτονική της κατανεμημένης εκπαίδευσης, η TPUStrategy είναι ίδια την MirroredStrategy και εφαρμόζει και αυτή σύγχρονη κατανεμημένη εκπαίδευση. Οι TPUs παρέχουν την δικιά τους αποδοτική υλοποίηση all-reduce και αλλων κατανεμημένων λειτουργιών για εκπαίδευση σε πολλούς πυρήνες TPU, οι οποίες χρησιμοποιούνται στην στρατηγική TPUStrategy.

MultiWorkerMirroredStrategy

Η tf.distribute.MultiWorkerMirroredStrategy μοιάζει πολύ με την MirroredStrategy. Εφαρμόζει σύγχρονη κατανεμημένη εκπαίδευση σε πολλούς εργάτες, ο καθένας με δυνητικά πολλαπλές GPU. Η διαφορά σε σχέση με την MirroredStrategy είναι ότι μπορούν να χρησιμοποιηθεί για την εκπαίδευση ένα cluster όχι απλά μία συσκευή με πολλαπλές GPU. Παρόμοια με την tf.distribute.MirroredStrategy, δημιουργεί

αντίγραφα όλων των μεταβλητών του μοντέλου σε κάθε συσκευή σε όλους τους εργάτες.

Το MultiWorkerMirroredStrategy διαθέτει δύο εφαρμογές για επικοινωνίες μεταξύ συσκευών. Το CommunicationImplementation.RING βασίζεται σε RPC και υποστηρίζει τόσο CPU όσο και GPU και το CommunicationImplementation.NCCL που χρησιμοποιεί NCCL και παρέχει state of the art επιδόσεις σε GPU αλλά δεν υποστηρίζει CPU.

Μια από τις βασικές διαφορές για να ξεκινήσει η εκπαίδευση πολλών εργατών, σε σύγκριση με την εκπαίδευση πολλαπλών GPU, είναι η ρύθμιση του cluster. Στο TensorFlow οι μεταβλητές περιβάλλοντος είναι ο τυπικός τρόπος για να καθοριστεί η διαμόρφωση του cluster σε κάθε εργαζόμενο που είναι μέρος του.

ParameterServerStrategy

Η εκπαίδευση με Parameter server είναι μία κλασική data-parallel μέθοδος για εκπαίδευση νευρωνικών δικτύων χρησιμοποιώντας πολλά μηχανήματα. Σε αυτήν την περίπτωση το cluster αποτελείται από εργάτες και parameter servers. Οι μεταβλητές δημιουργούνται στους parameter servers και διαβάζονται και ενημερώνονται από τους εργάτες σε κάθε βήμα. Στο TensorFlow 2, η εκπαίδευση με parameter server χρησιμοποιεί μια κεντρική αρχιτεκτονική που βασίζεται σε έναν coordinator μέσω της κλάσης tf.distribute.experimental.coordinator.ClusterCoordinator.

Σε αυτήν την στρατηγική , οι εργάτες και οι parameter server τρέχουν σε Servers οι οποίοι εκτελούν τις λειτουργίες που τους ορίζει ο coordinator. Ο συντονιστής είναι υπεύθυνος για την δημιουργία πόρων, ανάθεση εργασιών εκπαίδευσης στους εργάτες, να αποθηκεύει τα checkpoints και αντιμετωπίζει αποτυχίες.

CentralStorageStrategy

Η tf.distribute.experimental.CentralStorageStrategy κάνει επίσης σύγχρονη εκπαίδευση. Οι μεταβλητές δεν αντιγράφονται σε κάθε εργάτη, αλλά τοποθετούνται στην CPU και οι λειτουργίες αναπαράγονται σε όλες τις τοπικές GPU. Εάν υπάρχει μόνο μία GPU, όλες οι μεταβλητές και οι λειτουργίες θα τοποθετηθούν σε αυτήν τη GPU.

Κεφάλαιο 4

Μεθοδολογία

4.1 Πειραματική Διάταξη

Στην εργασία αυτή το πειραματικό σκέλος έγινε πάνω σε ένα cluster μηχανημάτων. Το cluster αυτό αποτελείται από πανομοιότυπα 5 vms όπου τα χαρακτηριστικά τους φαίνονται στον παρακάτω πίνακα 4.1.

Table 4.1: Πειραματική Διάταξη

Χαρακτηριστικό	Τιμή
Επεξεργαστής	Intel Core Processor (Skylake) 2.2GHz
Αριθμός πυρήνων	8
Μνήμη RAM	16GB
Έκδοση λειτουργικού	Ubuntu 16.04.3 LTS

4.2 Συστήματα

Παρακάτω παρατίθενται δύο συστήματα τα οποία χρησιμοποιήθηκαν στην παρούσα εργασία το πρώτο είναι το Ganglia ένα εργαλείο για παρακολούθηση του συστήματος και δεύτερο είναι το Apache Hadoop ένα εργαλείο για κατανεμημένα συστήματα.

4.2.1 Ganglia

Για την παρακολούθηση των παραμέτρων του συστήματος χρησιμοποιήθηκε το Ganglia Monitoring System [37]. ο Ganglia είναι ένα κλιμακούμενο κατανεμημένο

σύστημα παρακολούθησης για υπολογιστικά συστήματα υψηλής απόδοσης όπως clusters και Grids. Το Ganglia είναι ένα έργο ανοιχτού κώδικα με άδεια BSD που αναπτύχθηκε από το Πανεπιστήμιο της Καλιφόρνιας, Berkeley. Δίνει την δυνατότητα παρακολούθησης πληροφοριών για την χρησιμοποίηση του επεξεργαστή, το φόρτωμα διεργασιών, στην κατανάλωση της μνήμης RAM καθώς και την χρήση του δικτύου.

4.2.2 Apache Hadoop

Για την διαχείριση δεδομένων στο κατανεμημένο σύστημα που χρησιμοποιήθηκε για τα πειράματα επιλέχθηκε το Apache Hadoop [38]. Το Apache Hadoop αποτελεί λογισμικό ανοιχτού κώδικα για αξιόπιστο, κλιμακώσιμο, κατανεμημένο υπολογισμό. Η βιβλιοθήκη λογισμικού Apache Hadoop είναι ένα framework που επιτρέπει την κατανεμημένη επεξεργασία μεγάλων συνόλων δεδομένων σε ομάδες υπολογιστών χρησιμοποιώντας απλά μοντέλα προγραμματισμού. Έχει σχεδιαστεί για να κλιμακώνει συστήματα που αποτελούνται από μεμονωμένους servers σε χιλιάδες μηχανές, καθένας από τους οποίους προσφέρει τοπικό υπολογισμό και αποθήκευση.

4.3 Βιβλιοθήκες

Για την ανάπτυξη του κώδικα χρησιμοποιήθηκε το framework TensorFlow [9] στην γλώσσα προγραμματισμού Python. Χρησιμοποιήθηκαν διάφορες επιμέρους βιβλιοθήκες του TensorFlow από τις οποίες οι πιο βασικές είναι η TensorFlow.Keras και η Tensorflow.data. Η TensorFlow.Keras επιλέχθηκε καθώς καθιστά πολύ εύκολη και απλή την κατασκευή και εκπαίδευση πολύπλοκων νευρωνικών δικτύων για βαθιά μηχανική μάθηση. Η TensorFlow.data χρησιμοποιήθηκε για την προεπεξεργασία των δεδομένων που αποτελεί ένα πολύ σημαντικό κομμάτι για την υψηλή απόδοση στην βαθιά μηχανική μάθηση. Για την κατανεμημένη εκπαίδευση χρησιμοποιήθηκε η βιβλιοθήκη tf.distribute.Strategy η οποία αναπτύχθηκε σε προηγούμενη υποενότητα.

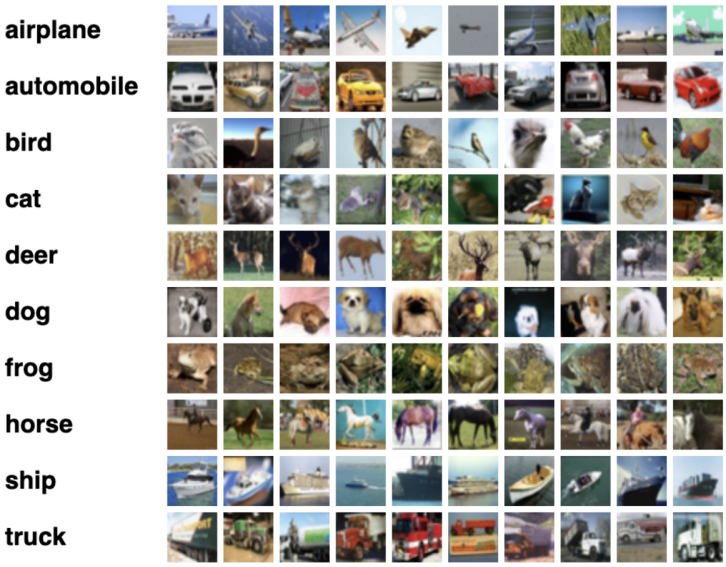
4.4 Σύνολα Δεδομένων

Στην εργασία αυτή επιλέχθηκαν σύνολα δεδομένων κατάλληλα για ταξινόμηση εικόνων. Η ταξινόμηση εικόνων είναι μια θεμελιώδης εργασία που επιχειρεί να κατανοήσει μια εικόνα στο σύνολό της. Ο στόχος είναι να ταξινομηθεί η εικόνα, αναθέτοντάς την σε μια συγκεκριμένη κλάση. Συνήθως, η ταξινόμηση εικόνων αναφέρεται σε εικόνες στις οποίες εμφανίζεται και αναλύεται μόνο ένα αντικείμενο. Η επιλογή τέτοιων συνόλων δεδομένων έγινε καθώς η εκπαίδευση νευρωνικών δικτύων με εικόνες αποτελεί μία πολύ απαιτητική υπολογιστικά εργασία με αποτέλεσμα πολύ μεγάλο χρόνο εκπαίδευσης. Τέτοια σύνολα δεδομένων είναι κατάλληλα για την αξιολόγηση εκπαίδευσης βαθιών νευρωνικών δικτύων σε κατανεμημένο περιβάλλον. Τα πειράματα στην

εργασία αυτή έγιναν με δύο γνωστά σύνολα δεδομένων το CIFAR-10 [39] και το Fashion MNIST [40].

4.4.1 CIFAR-10

Τα CIFAR-10 αποτελεί ένα σύνολο εικόνων με ετικέτες που συλλέχθηκαν από τους Alex Krizhevsky, Vinod Nair και Geoffrey Hinton. Το σύνολο δεδομένων CIFAR-10 αποτελείται από 60000 έγχρωμες εικόνες 32x32 χωρισμένες σε 10 κλάσεις, με 6000 εικόνες ανά κλάση. Το σύνολο δεδομένων χωρίζεται σε 50000 δεδομένα εκπαίδευσης και 10000 δεδομένα ελέγχου. Το σύνολο δεδομένων χωρίζεται σε πέντε batches εκπαίδευσης και ένα batch ελέγχου, κάθε ένα από τα οποία αποτελείται από 10000 εικόνες. Το batch ελέγχου περιέχει ακριβώς 1000 τυχαία επιλεγμένες εικόνες από κάθε κλάση. Τα baches εκπαίδευσης περιέχουν τις υπόλοιπες εικόνες σε τυχαία σειρά, αλλά μερικά batches εκπαίδευσης μπορεί να περιέχουν περισσότερες εικόνες από τη μία κλάση. Συνολικά τα baches εκπαίδευσης περιέχουν ακριβώς 5000 εικόνες από κάθε τάξη. Οι κλάσεις στις οποίες χωρίζονται τα δεδομένα είναι οι εξής: αεροπλάνο, αυτοκίνητο, πουλί, γάτα, ελάφι, σκύλος, βάτραχος, άλογο, πλοίο και φορτηγό. Στην εικόνα 4.1 βλέπουμε παραδείγματα εικόνων από κάθε κλάση.

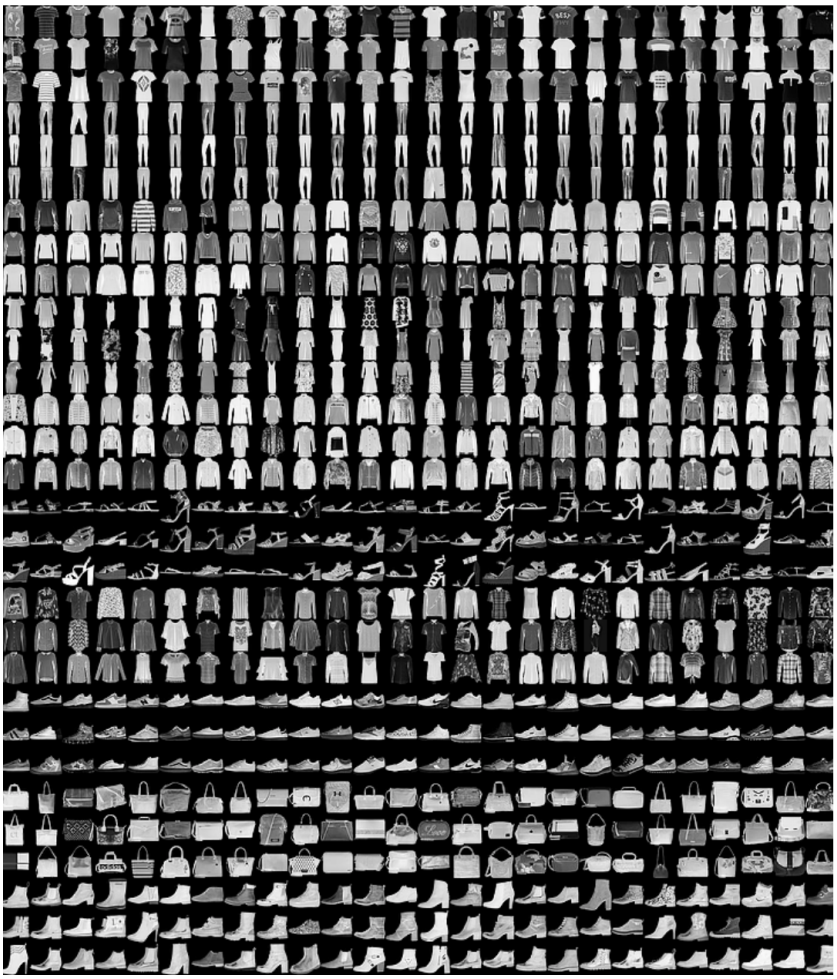


Εικόνα 4.1: Παραδείγματα εικόνων του CIFAR-10 για κάθε κατηγορία.

4.4.2 Fashion MNIST

Το Fashion-MNIST είναι ένα σύνολο δεδομένων από εικόνες με ρούχα και αποτελείται από ένα εκπαιδευτικό σύνολο 60.000 παραδειγμάτων και ένα σύνολο ελέγχου 10.000 παραδειγμάτων. Σκοπός του Fashion-MNIST είναι η αντικατάσταση του αρχι-

κού συνόλου δεδομένων MNIST για συγκριτική αξιολόγηση αλγορίθμων μηχανικής μάθησης.



Εικόνα 4.2: Παραδείγματα εικόνων του Fashion MNIST για κάθε κατηγορία.

Μοιράζονται το ίδιο μέγεθος εικόνας και ίδιο αριθμό δεδομένων εκπαίδευσης και ελέγχου. Το original σύνολο δεδομένων MNIST περιέχει πολλά χειρόγραφα ψηφία. Τα μέλη της κοινότητας της μηχανικής μάθησης το χρησιμοποιούν ως σημείο αναφοράς για την επικύρωση των αλγορίθμων τους. Στην πραγματικότητα, το MNIST είναι συχνά το πρώτο σύνολο δεδομένων που προσπαθούν οι ερευνητές. Το Fashion-MNIST φτιάχτηκε για να το αντικαταστήσει καθώς το MNIST είναι πολύ εύκολο να πετύχει κάνεις με ένα απλό συνελικτικό δίκτυο απόδοση της τάξης του 99,7%. Το σύνολο δεδομένων Fashion-MNIST χωρίζεται σε 10 κλάσεις, οι οποίες είναι οι εξής: T-shirt, παντελόνι, πουλόβερ, φόρεμα, παλτό, σανδάλι, πουκάμισο, αθλητικό παπούτσι, τσάντα και μποτάκια. Κάθε εικόνα έχει ύψος 28 pixels και πλάτος 28 pixels, συνολικά 784 pixels συνολικά. Κάθε pixels έχει μια τιμή υποδεικνύοντας τη

φωτεινότητα αυτού του pixels, όσο πιο μεγάλος είναι ο αριθμός σε τόσο πιο σκούρο χρώμα pixels. Αυτή η τιμή pixels είναι ένας ακέραιος αριθμός μεταξύ 0 και 255. Στην παρακάτω εικόνα βλέπουμε κάποια παραδείγματα από το σύνολο δεδομένων Fashion-MNIST όπου είναι εμφανής και οι δέκα κατηγορίες που αναφέρθηκαν.

4.4.3 Προεπεξεργασία δεδομένων

Η προεπεξεργασία δεδομένων [41] είναι ένα αναπόσπαστο βήμα στη βαθιά μηχανική μάθηση καθώς η ποιότητα των δεδομένων και οι χρήσιμες πληροφορίες που μπορούν να προκύψουν επηρεάζουν άμεσα την ικανότητα εκμάθησης του νευρωνικού δικτύου. Ως εκ τούτου, είναι εξαιρετικά σημαντική η επεξεργασία εκ των προτέρων των δεδομένων πριν χρησιμοποιηθούν για την εκπαίδευση του μοντέλου.

Στα σύνολα δεδομένων που χρησιμοποιήθηκαν στην συγκεκριμένη εργασία έγιναν 3 βασικές μέθοδοι προεπεξεργασίας. Η πρώτη και πιο κλασική μέθοδος είναι η κανονικοποίηση των δεδομένων. Οι επόμενες δύο αφορούν την αύξηση δεδομένων (data augmentation).

4.4.4 Κανονικοποίηση

Η κανονικοποίηση των δεδομένων είναι ένα σημαντικό βήμα που διασφαλίζει ότι κάθε παράμετρος εισόδου (pixel, στην περίπτωση αυτή) έχει παρόμοια κατανομή δεδομένων. Αυτό καθιστά τη σύγκλιση ταχύτερη κατά την εκπαίδευση του δικτύου. Τα δεδομένα των εικόνων αναπαριστώνται σε pixel. Στο σύνολο δεδομένων CIFAR-10 για κάθε pixel έχουμε 3 τιμές, τα τρία χρώματα RGB (κόκκινο, πράσινο, μπλε) όπου κάθε χρώμα πέρνει μία τιμή από το 0 μέχρι το 255. Αντίθετα στο σύνολο δεδομένων Fashion MNIST οι εικόνες έχουν μόνο αποχρώσεις του γκρι συνεπώς μία μόνο τιμή αναπαριστά κάθε pixel στο εύρος [0,255]. Η κανονικοποίηση των δεδομένων γίνεται αφαιρώντας το μέσο όρο από κάθε pixel και στη συνέχεια διαιρώντας με την τυπική απόκλιση. Η κατανομή αυτών των δεδομένων θα μοιάζει με μια καμπύλη Gauss με επίκεντρο το μηδέν. Για τις εισόδους εικόνας οι τιμές των pixel πρέπει να είναι θετικοί, οπότε επιλέγεται να κλιμακωθουν τα κανονικοποιημένα δεδομένα στο [0,1]. Άρα η διαδικασία είναι, για κάθε χρώμα pixel στην περίπτωση του CIFAR-10 και για κάθε pixel στο Fashion MNIST, η αφαίρεση της μέσης τιμής και στην συνέχεια η διαίρεση με το 255.

4.4.5 Data Augmentation

Για να εκπαιδευτεί ένα μοντέλο και να έχει μεγάλη ακρίβεια πρόβλεψης, πρέπει να έχει μεγάλο αριθμό παραμέτρων, όπως το ResNet και το DenseNet. Έτσι μαθαίνει σχεδόν όλα τα χαρακτηριστικά των δεδομένων και έχει μεγαλύτερη απόδοση. Για να εκπαιδευτούν όλες αυτές οι παράμετροι, πρέπει να υπάρχει ένας ικανός όγκος δεδομένων. Τα μοντέλα βαθιάς μάθησης συχνά απαιτούν περισσότερα δεδομένα τα οποία δεν είναι πάντα διαθέσιμα γι'αυτό χρησιμοποιούνται διάφορες τεχνικές του Data

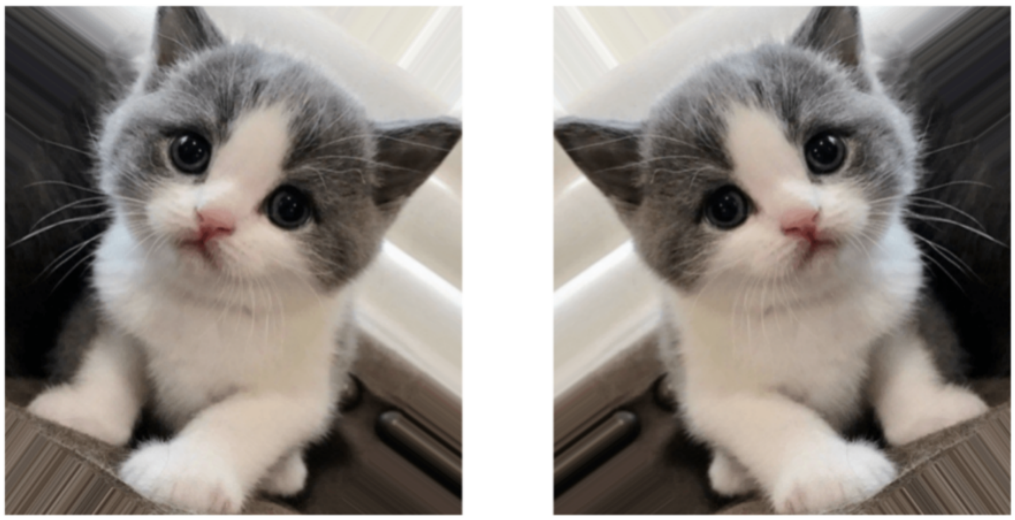
Augmentation [42].

Το Data Augmentation είναι η πρακτική της χρήσης δεδομένων που έχουμε ήδη για τη δημιουργία νέων παραδειγμάτων εκπαίδευσης για να βοηθήσουμε τα μοντέλα μηχανικής μάθησης να γενικεύουν καλύτερα την γνώση. Με άλλα λόγια, αυξάνουμε τεχνητά το μέγεθος του συνόλου δεδομένων δημιουργώντας διαφορετικές εκδόσεις των υπαρχόντων δεδομένων από το σύνολο δεδομένων μας. Ο κύριος λόγος για αυτό, είναι ότι τα δεδομένα του πραγματικού κόσμου μπορεί να μην είναι πάντα στη σωστή μορφή. Για παράδειγμα, ένα αυτοκίνητο σε μια εικόνα, μπορεί να μην είναι στο κέντρο σε όλες τις περιπτώσεις, μερικές φορές μπορεί να είναι στην αριστερή πλευρά της εικόνας ή στη δεξιά. Μπορεί η εικόνα να τραβηχτεί σε μια φωτεινή ηλιόλουστη μέρα ή σε μια συννεφιασμένη μέρα. Στην εικόνα μπορεί να φαίνεται η αριστερή όψη του αυτοκινήτου ή η δεξιά όψη. Όλοι αυτοί οι παράγοντες επηρεάζουν το μοντέλο κατά την αξιολόγηση μιας εικόνας. Το μοντέλο πρέπει να εκπαιδευτεί με τέτοιο τρόπο ώστε να μπορεί να ανιχνεύσει το αντικείμενο με ακρίβεια ανεξάρτητα από τους παραπάνω παράγοντες. Τέλος τεχνική του Data Augmentation εφαρμόζεται μόνο στο σύνολο δεδομένων εκπαίδευσης. Κατα την διάρκεια μέτρησης της απόδοσης του μοντέλου με το σύνολο δεδομένων ελέγχου(test dataset) οι εικόνες περνάνε κατευθείαν στο μοντέλο χωρίς κανέναν μετασχηματισμό.

Υπάρχουν διάφορες τεχνικές Data Augmentation , στην συγκεκριμένη εργασία χρησιμοποιήθηκαν δύο. Η πρώτη αφορά το Random Flip και η δεύτερη το Random Crop. Επίσης στην παρούσα εργασία οι μέθοδοι που εφαρμόστηκαν για το Data Augmentationήταν από την βιβλιοθήκη tensorflow.image.

4.4.6 Random Flip

Η εικόνα μπορεί να αναστραφεί είτε οριζόντια είτε κάθετα με βάση το αντικείμενο της εικόνας. Για παράδειγμα, η εικόνα ενός αυτοκινήτου δεν μπορεί να αναστραφεί κατακόρυφα, καθώς θα έχει ως αποτέλεσμα το αυτοκίνητο ανάποδα πράγμα που δεν αποτελεί χρήσιμο δεδομένο για την εκπαίδευση. Ωστόσο, μπορεί να αναστραφεί οριζόντια δημιουργώντας δύο εικόνες όπου στην μία απεικονίζεται η αριστερή όψη και μία όπου εμφανίζεται δεξιά όψη ενός αυτοκινήτου. Στα σύνολα δεδομένων όπου χρησιμοποιήσαμε το CIFAR-10 και το Fashion-MNIST εφαρμόσαμε μόνο οριζόντια αναστροφή εικόνας καθώς τα αντικείμενα που απεικονίζονται δεν θα εμφανιζόντουσαν σε καμία περίπτωση ανάποδα στην πραγματικότητα. Παρακάτω βλέπουμε ένα παράδειγμα μίας εικόνας με μία γάτα που έχει δεχθεί οριζόντια αναστροφή (εικόνα 4.3). Για την εφαρμογή του random flip στον οριζόντιο άξονα χρησιμοποιήθηκε η συνάρτηση random_flip_left_right του tensorflow.image.



Εικόνα 4.3: Παράδειγμα αναστροφής εικόνας

4.4.7 Random Crop

Το Random crop είναι μια τεχνική αύξησης δεδομένων όπου επιλέγει ένα τυχαίο υποσύνολο μιας αρχικής εικόνας. Αυτό βοηθά το μοντέλο να γενικεύει καλύτερα επειδή τα αντικείμενα ενδιαφέροντος που επιδιώκει να μάθει το μοντέλο δεν είναι πάντα εντελώς ορατά στην εικόνα και πάντα στο κέντρο της. Για την εφαρμογή του random crop χρησιμοποιήθηκαν δύο συναρτήσεις καθώς ήταν θεμιτό όλες οι φωτογραφίες να έχουν τον ίδιο μέγεθος. Αρχικά χρησιμοποιήθηκε η tf.image.pad_to_bounding_box με την οποία αυξάνεται το μέγεθος της εικόνας προς όλες τις διαστάσεις. Στην συνέχεια γίνεται ένα τυχαίο crop πάνω στο επαυξημένο μέγεθος με την χρήση της συνάρτησης tf.image.random_crop.

4.5 Μοντέλα

Για την εργασία αυτή χρησιμοποιήθηκαν δύο μοντέλα συνελικτικών νευρωνικών δικτύων, το ResNet [11] και το DenseNet [12]. Για την σύγκριση των πειραμάτων και μεταξύ των διαφορετικών στρατηγικών οι παράμετροι των μοντέλων παρέμειναν ίδιοι. Στους παρακάτω πίνακες απεικονίζονται οι παράμετροι των μοντέλων. Το ResNet υλοποιήθηκε για το σύνολο δεδομένων CIFAR-10 [39] ενώ το DenseNet [40] για το σύνολο δεδομένων Fashion-MNIST. Στους παρακάτω πίνακες 4.2 και 4.3 βλέπουμε τις παραμέτρους των μοντέλων που χρησιμοποιήθηκαν για τα πειράματα.

Πίνακας 4.2: ResNet

Χαρακτηριστικό	Τιμή
Depth (Βάθος)	50
Αριθμός residual block	8 x 3 = 24
Αριθμός παραμέτρων	0.85 εκατομμύρια
Αριθμός κλάσεων	10
Input Shape	(32,32,3)

4.6 Στρατηγικές

Στην εργασία αυτή αναπτύχθηκαν 3 διαφορετικές στρατηγικές για κατανεμημένη εκπαίδευση νευρωνικών δικτύων. Οι στρατηγικές έχουν διαφορές τόσο στην αρχιτεκτονική όσο και στον συγχρονισμό. Χρησιμοποιήθηκαν 2 στρατηγικές που παρέχει το TensorFlow [9]. Η πρώτη είναι η Multi Worker Mirrored Strategy και η δεύτερη είναι η Parameter Server Strategy. Τέλος αναπτύχθηκε μία υβριδική στρατηγική με σκοπό την εκμετάλλευση των θετικών στοιχείων και των δύο στρατηγικών.

4.6.1 Multi Worker Mirrored Strategy

Για αυτήν την στρατηγική χρησιμοποιήθηκε αποκεντρωμένη αρχιτεκτονική all-reduce, όπου δεν υπάρχει κάποιος κεντρικός κόμβος και οι εργάτες επικοινωνούν απευθείας μεταξύ τους. Για την επικοινωνία αυτή χρησιμοποιήθηκε ο αλγόριθμος ring all-reduce όπου αναπτύχθηκε σε προηγούμενη ενότητα. Όσον αφορά τον συγχρονισμό του συστήματος χρησιμοποιήθηκε σύγχρονη ενημέρωση των παραμέτρων. Ο αλγόριθμος με τον οποίο κατασκευάστηκε η στρατηγική multi worker είναι ο εξής:

- Βήμα 1: Κατασκευή του μοντέλου και δημιουργία αντιγράφων των παραμέτρων σε κάθε εργάτη.
- Βήμα 2: Προεπεξεργασία και μοίρασμα των δεδομένων στους εργάτες, διαχωρισμός αυτών ώστε κάθε εργάτης να χρησιμοποιήσει διαφορετικό κομμάτι των δεδομένων.
- Βήμα 3: Κάθε εργάτης εκπαιδεύει τις παραμέτρους του με ένα batch δεδομένων.
- Βήμα 4: Συγχρονισμός των παραμέτρων, οι εργάτες ανταλλάσσουν τις αλλαγές στις παραμέτρους τους, τις ενημερώνουν και όλοι έχουν το ίδιο ενημερωμένο μοντέλο.

Πίνακας 4.3: DenseNet

Χαρακτηριστικό	Τιμή
Depth (Βάθος)	40
Growth rate	12
Αριθμός dense block	3
Reduction	0.5
Dropout rate	0.1
Bottleneck	TRUE
Αριθμός κλάσεων	10
Input Shape	(28,28,1)

- Βήμα 5: Το Βήμα 3 και 4 επαναλαμβάνονται για τον αριθμό των βημάτων μιας εποχής.
- Βήμα 6: Το Βήμα 5 επαναλαμβάνεται για τον αριθμό των εποχών.

4.6.2 Parameter Server Strategy

Στην στρατηγική αυτή χρησιμοποιείται συγκεντρωτική αρχιτεκτονική, υπάρχει ένας ή πολλοί parameter servers όπου έχουν αποθηκευμένες και ενημερωμένες τις παραμέτρους του μοντέλου. Οι εργάτες όταν ξεκινάνε την εκπαίδευση σε ένα καινούργιο σύνολο δεδομένων ζητάνε από τους ps τις παραμέτρους και όταν τελειώσουν την εκπαίδευση που κάνουν, στέλνουν πίσω τις αλλαγές των παραμέτρων ώστε οι ps να κάνουν την ενημέρωση στο μοντέλο. Επίσης χρησιμοποιείται ασύγχρονη εκπαίδευση δηλαδή κάθε εργάτης δουλεύει ανεξάρτητα και οι ενημερώσεις στις παραμέτρους των ps γίνονται ασύγχρονα. Ο αλγόριθμος με τον οποίο κατασκευάστηκε η στρατηγική του parameter server είναι η εξής:

- Βήμα 1: Κατασκευή του μοντέλου στον parameter server.
- Βήμα 2: Προεπεξεργασία και μοίρασμα του συνόλου δεδομένων στους εργάτες.

  Τα παρακάτω τρία βήματα τα κάνει ο κάθε εργάτης ανεξάρτητα.
- Βήμα 3: Κάθε εργάτης ζητάει το ενημερωμένο μοντέλο από τον parameter server.

- Βήμα 4: Κάθε εργάτης εκπαιδεύει τις παραμέτρους που έχει λάβει με ένα batch εκπαίδευσης των δεδομένων που του αναλογούν.
- Βήμα 5: Όταν τελειώνει την εκπαίδευση κάθε εργάτης στέλνει τις ενημερώσεις των παραμέτρων του στον ps. Ο ps ενημερώνει ασύγχρονα τις παραμέτρους του κάθε φορά που λαμβάνει ενημερώσεις από κάποιον εργάτη.
- Βήμα 6: Το Βήμα 3, 4 και 5 επαναλαμβάνονται για τον αριθμό των βημάτων μιας εποχής.
- Βήμα 7: Το Βήμα 6 επαναλαμβάνεται για τον αριθμό των εποχών.

4.6.3 Στρατηγική Strategy Switch

Παρατηρήθηκε στην βιβλιογραφία (ενότητα 3.2.1) ότι μία υβριδική χρήση συγχρονισμού μπορεί να επιφέρει θετικά αποτελέσματα στην κατανεμημένη εκπαίδευση μεγάλων νευρωνικών δικτύων. Συνεπώς κατασκευάσαμε και μελετήσαμε την στρατηγική Strategy Switch. Η Strategy Switch αποτελεί μία υβριδική στρατηγική μεταξύ των στρατηγικών του Multi Worker Mirrored Strategy και του Parameter Server Strategy. Με την κατασκευή αυτής της στρατηγικής γίνεται προσπάθεια να αξιοποιηθούν τα πλεονεκτήματα και των δύο στρατηγικών. Για την στρατηγική αυτή αρχικά κατασκευάζεται το μοντέλο και γίνεται η εκπαίδευση σύμφωνα με τον αλγόριθμο που διατυπώθηκε για την Multi Worker Mirrored Strategy. Στην συνέχεια κάποια στιγμή μετά από έναν συγκεκριμένο αριθμό εποχών αλλάζει η στρατηγική και ο συγχρονισμός καθώς για την εκπαίδευση ακολουθείται ο αλγόριθμος του Parameter Server Strategy και η σύγχρονη στρατηγική. Για την υλοποίηση αυτής της στρατηγικής χρησιμοποιήθηκε η μέθοδος του Transfer Learning [43]. Με την μέθοδο αυτή μεταφέρεται η γνώση που αποκτά το μοντέλο που τρέχει με την Multi Worker Mirrored Strategy στο μοντέλο που τρέχει με την στρατηγική του Parameter Server Strategy.

4.7 Πειράματα

Στην εργασία αυτή πραγματοποιήθηκαν πειράματα για την αξιολόγηση των στρατηγικών εκπαίδευσης Multi Worker Mirrored Strategy , Parameter Server Strategy και Στρατηγική Strategy Switch. Βασικό μέτρο αξιολόγησης είναι τόσο η ακρίβεια των μοντέλων όσο και ο χρόνος εκτέλεσης που συχνά αποτελούν δύο αντικρουόμενες μετρικές, για την βελτιστοποίηση της ακρίβειας απαιτείται μεγαλύτερος χρόνος εκπαίδευσης και αντίστροφα για την βελτιστοποίηση του χρόνου εκπαίδευσης παρατηρείται πτώση στην ακρίβεια.

Για τα πειράματα χρησιμοποιήθηκαν δύο μεγάλα και γνωστά συνελικτικά νευρωνικά δίκτυα που επιφέρουν εξαιρετικά αποτελέσματα στην ταξινόμηση εικόνων το ResNet [11] και το DenseNet [12]. Τα μοντέλα αυτά δοκιμάστηκαν σε δύο σύνολα δεδομένων ταξινόμησης εικόνων το CIFAR-10 [39] και το Fashion-MNIST [40].

Στους πίνακες 4.2 και 4.3 αναφέρονται αναλυτικά οι παράμετροι του μοντέλου ResNet και DenseNet αντίστοιχα που χρησιμοποιήθηκαν για τα πειράματα. Για όλα τα πειράματα χρησιμοποιήθηκε το ίδιο μοντέλο ώστε να είναι αντικειμενική η σύγκριση. Στους παρακάτω πίνακες 4.4 4.5 αναφέρονται οι υπερπαράμετροι των πειραμάτων εκπαίδευσης.

Table 4.4: Πειράματα ResNet

Χαρακτηριστικό	Τιμή
Μοντέλο	ResNet50
Σύνολο Δεδομένων	CIFAR-10
Αριθμός εργατών	5
Εποχές	180
Global batch size	128
Per worker batch size	26
Learning rate	0.1 0.01 0.001 (80 και 130 εποχές)
Αλγόριθμος	Stochastic Gradient Descent
Momentum	0.9
Data Augmentation	True

Οι υπερπαράμετροι αυτοί επιλέχθηκαν σύμφωνα με τις υπερπαραμέτρους του πειράματος σε έναν κόμβο και ισχύουν για όλα τα πειράματα που πραγματοποιήθηκαν με μόνη διαφορά στο Learning rate του Parameter Server Strategy, καθώς έχει παρατηρηθεί ότι βγάζει καλύτερα αποτελέσματα στην κατανεμημένη εκπαίδευση όταν προσαρμόζεται με τον τύπο 4.1. Αυτό συμβαίνει επειδή οι ενημερώσεις των παραμέτρων είναι ασύγχρονες και γίνονται από κάθε εργάτη ξεχωριστά με αποτέλεσμα να είναι πολύ περισσότερες και ο βαθμός της ενημέρωσης να πρέπει να είναι αντιστρόφως ανάλογος του αριθμού των εργατών.

$$Parameter\ server\ Learning\ rate = \frac{Learning\ rate}{number\ of\ workers} \tag{4.1}$$

Για κάθε δίκτυο πραγματοποιήθηκαν πέντε διαφορετικοί τύποι πειραμάτων. Ένα πείραμα για τις στρατηγικές Multi Worker Mirrored Strategy και Parameter Server Strategy και τρία για την στρατηγική Strategy Switch. Τα πειράματα των Multi

Table 4.5: Πειράματα DenseNet

Χαρακτηριστικό	Τιμή
Μοντέλο	DenseNet
Σύνολο Δεδομένων	Fashion-MNIST
Αριθμός εργατών	5
Εποχές	40
Global batch size	128
Per worker batch size	26
Learning rate	0.1 0.01 0.001 (20 και 30 εποχές)
Αλγόριθμος	Stochastic Gradient Descent
Momentum	0.9
Data Augmentation	False

Worker Mirrored Strategy και Parameter Server Strategy έγιναν σύμφωνα με τις παραμέτρους που έχουν αναφερθεί. Στα πειράματα της στρατηγικής Strategy Switch υπάρχει άλλη μία παράμετρος η οποία αφορά την στιγμή στην οποία θα γίνει η αλλαγή της στρατηγικής. Αρχικά η εκπαίδευση ξεκίναγε με την στρατηγική Multi Worker Mirrored Strategy και την στιγμή που έχει επιλεχθεί γινόταν η αλλαγή στρατηγικής σε Parameter Server Strategy. Δοκιμάστηκαν τρεις διαφορετικές στιγμές για την αλλαγή η πρώτη είναι στο 25% η δεύτερη στο 50% και η τρίτη στο 75% των εποχών της εκπαίδευσης.

Κεφάλαιο 5

Αποτελέσματα Πειραμάτων

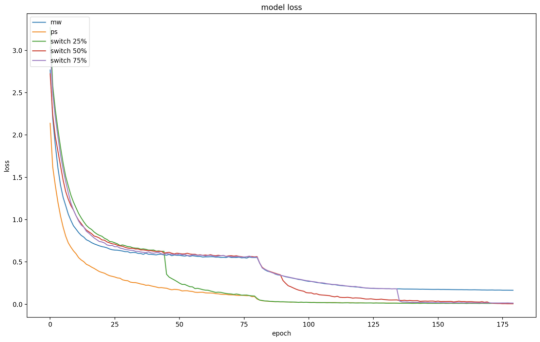
5.1 Αποτελέσματα ResNet

Παρακάτω παρουσιάζονται τα αποτελέσματα των πειραμάτων του ResNet με το σύνολο δεδομένων CIFAR-10. Γίνανε συνολικά πέντε τύποι πειραμάτων, ένα πείραμα για τις στρατηγικές Multi Worker Mirrored Strategy και Parameter Server Strategy και τρία για την στρατηγική Strategy Switch.

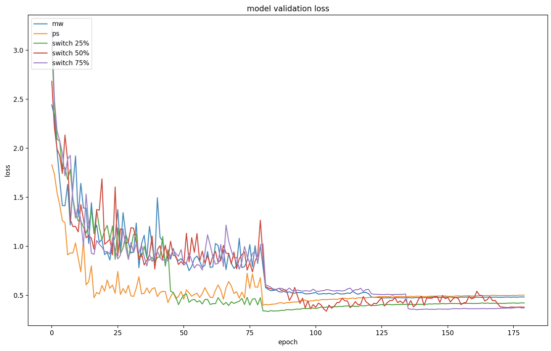
Table 5.1: Αποτελέσματα ResNet

Χαρακτηριστικό	MW	PS	SS 25%	SS 50%	SS 75%
Training Accuracy	99.80%	99.64%	99.68%	99.83%	99.61%
Validation Accuracy	92.71%	90.81%	91.70%	92.54%	92.29%
Training Loss	0.1597	0.0118	0.0104	0.0062	0.014
Validation Loss	0.4789	0.5008	0.4210	0.3779	0.3706
Elapsed time	7:38:25	6:36:24	6:42:44	6:57:05	7:19:01

5.1.1 Διαγράμματα ResNet



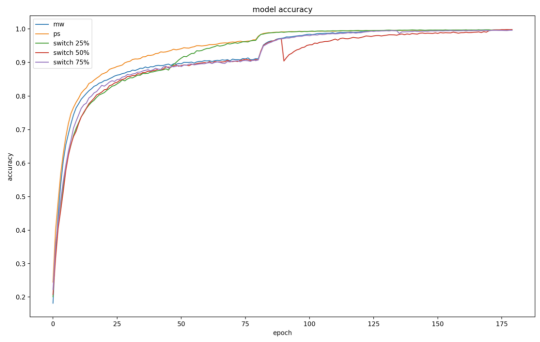
(a) ResNet Training Loss



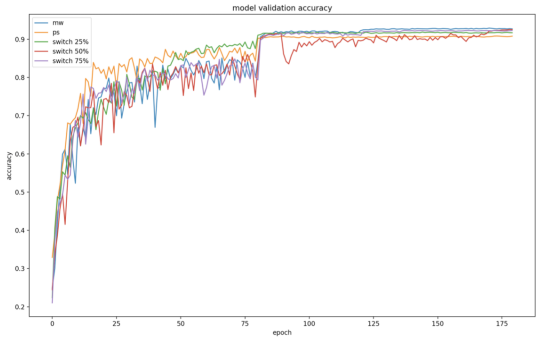
(b) ResNet Validation Loss

Figure 5.1: ResNet Loss

Στην εικόνα 5.1 βλέπουμε το Training και Validation Loss σε κάθε εποχή κατα την διάρκεια της εκπαίδευσης του μοντέλου ResNet. Αρχικά στην εικόνα 5.1α παρατηρούμε ότι η στρατηγική Parameter Server Strategy έχει χαμηλότερο loss από την στρατηγική του Multi Worker Mirrored Strategy η οποία όμως δεν ακολουθείται από την ίδια συμπεριφορά στοValidation Loss όπου εκεί οι δύο στρατηγικές έχουν πολύ παρόμοιες τιμές. Συνεπώς η στρατηγική Parameter Server Strategy κάνει λίγο περισσότερο overfitting. Όπως βλέπουμε οι στρατηγικές Strategy Switch ακολουθούν την συμπεριφορά του Parameter Server Strategy όσον αφορά το Training Loss αλλά παρατηρείται ότι έχουν και χαμηλότερες τιμές στο Validation Loss. Ιδιαίτερα βελτιωμένες φαίνεται να είναι οι τιμές του Validation Loss στις Strategy Switch 50% και 75%, αυτό μπορεί να οφείλεται σε καλύτερη γενίκευση και σε περισσότερα δεδομένα να βλέπαμε μεγαλύτερες τιμές Accuracy.



(α′) ResNet Training Accuracy



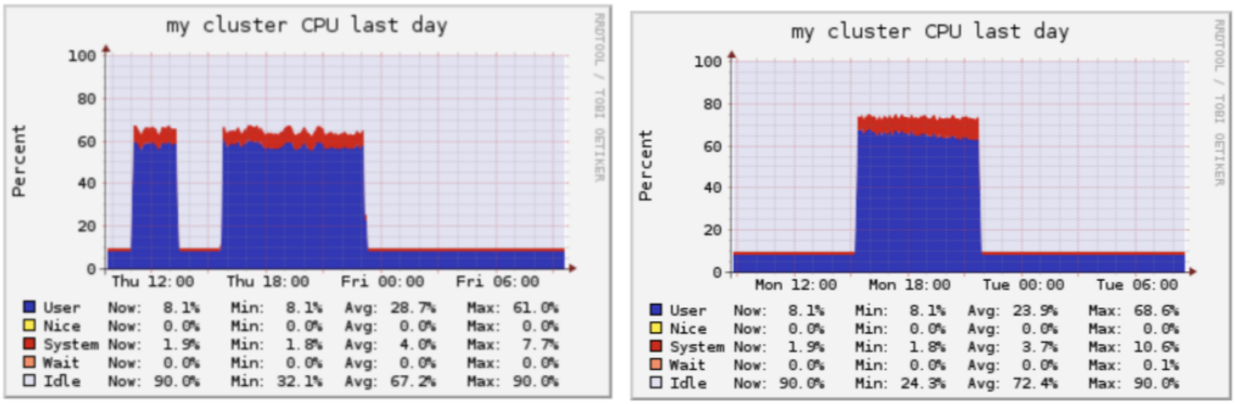
(β′) ResNet Validation Accuracy

Εικόνα 5.2: ResNet Accuracy

Στην εικόνα 5.2 βλέπουμε το Training και Validation Accuracy σε κάθε εποχή κατα την διάρκεια της εκπαίδευσης. Όσον αφορά το Training Accuracy όλες οι στρατηγικές έχουν την ίδια συμπεριφορά με μόνη διαφορά λίγο γρηγορότερη σύγκληση στην περίπτωση του Parameter Server Strategy. Στο Validation Accuracy που βλέπουμε στην εικόνα 5.2β΄ παρατηρούμε διαφορές στις τιμές σύγκλισης. Η Multi Worker Mirrored Strategy και η Strategy Switch 75% αλλά και 50% έχουν παρόμοια και πολύ υψηλή τιμή. Αντίθετα ο Parameter Server Strategy βλέπουμε να συγκλίνει χαμηλότερα.

5.1.2 Ganglia ResNet

CPU ResNet

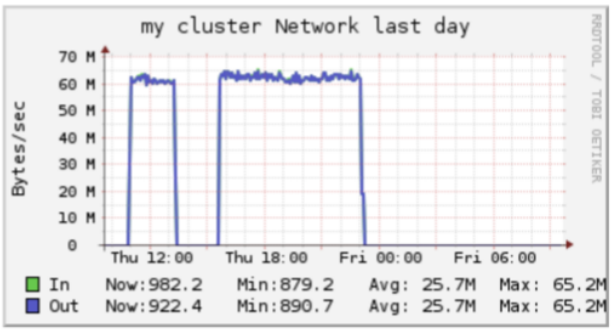


(α΄) CPU Multi Worker Mirrored Strategy (β΄) CPU Parameter Server Strategy

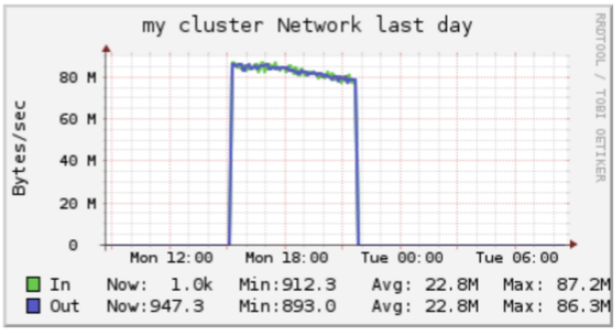
Εικόνα 5.3: CPU ResNet

Στις εικόνες 5.3α΄ και 5.3β΄ βλέπουμε την χρησιμοποίηση της CPU του cluster από τις δύο στρατηγικές Multi Worker Mirrored Strategy και Parameter Server Strategy. Κατα την εκπαίδευση με την Multi Worker Mirrored Strategy παρατηρούμε ότι η χρησιμοποίηση της CPU κυμαίνεται στο **68%**. Πιο ανεβασμένες τιμές χρησιμοποίησης της CPU παρατηρούνται για την Parameter Server Strategy, περίπου στο **76%**. Η μικρότερη απόδοση της Multi Worker Mirrored Strategy οφείλεται στο κόστος συγχρονισμού. Οι εργάτες που τελειώνουν την εκπαίδευση τους πρέπει να περιμένουν να τελειώσει και ο πιο αργός εργάτης το οποίο μειώνει την απόδοση αλλά και η διαδικασία του συγχρονισμού καθυστερεί την εκπαίδευση. Από την άλλη στην Parameter Server Strategy όταν ένας εργάτης τελειώνει την εκπαίδευση ενός batch επικοινωνεί με τον ps ανταλλάσουν τις πληροφορίες και συνεχίζει την εκπαίδευση στο επόμενο batch χωρίς να καθυστερεί και να μένει αδρανής.

Δίκτυο ResNet



(α′) Δίκτυο Multi Worker Mirrored Strategy

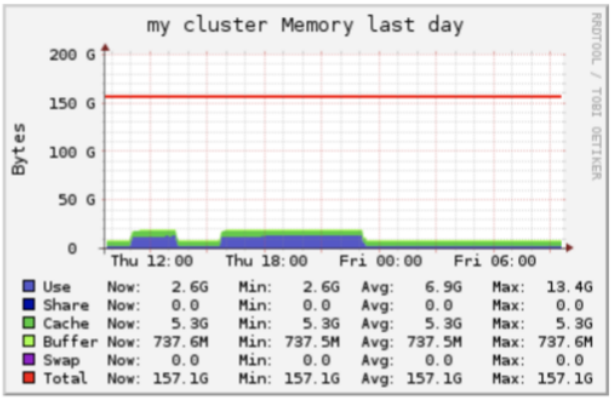


(β′) Δίκτυο Parameter Server Strategy

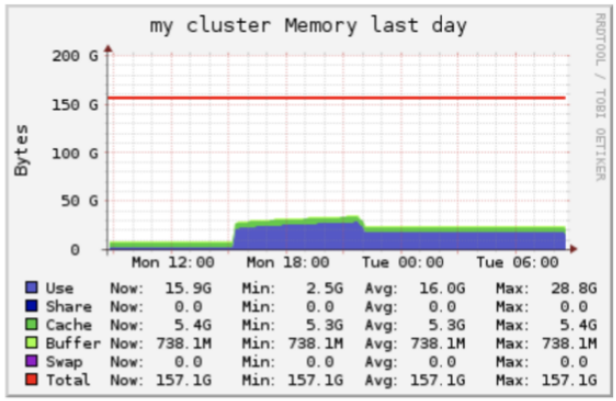
Εικόνα 5.4: Δίκτυο ResNet

Στις εικόνες 5.4α′ και 5.4β′ βλέπουμε την επιβάρυνση του δικτύου του cluster κατα την διάρκεια της εκπαίδευσης από τις δύο στρατηγικές Multi Worker Mirrored Strategy και Parameter Server Strategy. Οι τιμές που παρατηρούνται είναι περίπου **65M** Bytes/sec και **87M** Bytes/sec για Multi Worker Mirrored Strategy και Parameter Server Strategy αντίστοιχα. Η Multi Worker Mirrored Strategy έχει μικρότερη κατανάλωση δικτύου και αυτό οφείλεται στο γεγονός ότι χρησιμοποιεί τον αλγόριθμος Ring All-reduce σε αντίθεση με την στρατηγική Parameter Server Strategy όπου όλες οι πληροφορίες στέλνονται στον parameter server. Επίσης η Parameter Server Strategy είναι πιο γρήγορη συνεπώς και το δίκτυο θα είναι ελαφρός πιο επιβαρυμένο.

Μνήμη ResNet



(α′) Μνήμη Multi Worker Mirrored Strategy



(β′) Μνήμη Parameter Server Strategy

Εικόνα 5.5: Μνήμη ResNet

Στις εικόνες 5.5α΄ και 5.5β΄ βλέπουμε την κατανάλωση μνήμης RAM του cluster κατα την διάρκεια της εκπαίδευσης από τις δύο στρατηγικές Multi Worker Mirrored Strategy και Parameter Server Strategy. Οι τιμές που παρατηρούνται είναι **13.4GB** για την Multi Worker Mirrored Strategy και **28.8GB** για την Parameter Server Strategy. Ένα βασικός λόγος είναι ότι στην περίπτωση του Parameter Server Strategy υπάρχουν περισσότερες διεργασίας που έχουν στην μνήμη τους ανα πάσα στιγμή το μοντέλο για τον ίδιο αριθμό εργατών, όπως ο coordinator, οι parameter servers και ο evaluator.

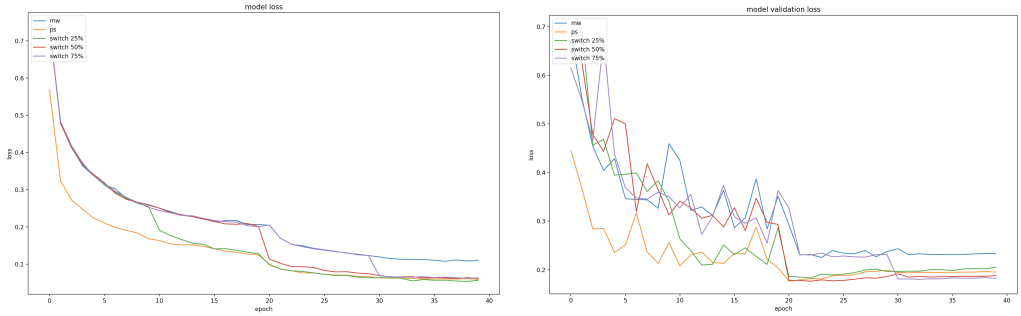
5.2 Αποτελέσματα DenseNet

Παρακάτω παρουσιάζονται τα αποτελέσματα των πειραμάτων του DenseNet με το σύνολο δεδομένων Fashion-MNIST Γίνανε συνολικά πέντε τύποι πειραμάτων, ένα πείραμα για τις στρατηγικές Multi Worker Mirrored Strategy και Parameter Server Strategy και τρία για την στρατηγική Strategy Switch.

Table 5.2: Αποτελέσματα DenseNet

Χαρακτηριστικό	MW	PS	SS 25%	SS 50%	SS 75%
Training Accuracy	97.89%	97.92%	98.03%	97.90%	97.82%
Validation Accuracy	94.51%	93.99%	94.26%	94.48%	94.50%
Training Loss	0.1093	0.059	0.0567	0.0615	0.0629
Validation Loss	0.2331	0.1952	0.2052	0.1880	0.1823
Elapsed time	1:59:30	1:33:28	1:42:13	1:44:28	1:53:03

5.2.1 Διαγράμματα DenseNet

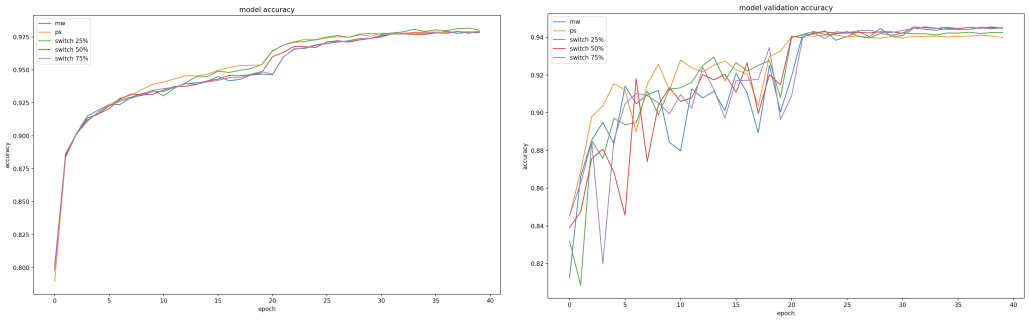


(a) DenseNet Training Loss

(b) DenseNet Validation Loss

Figure 5.6: ResNet Loss

Στην εικόνα 5.6 βλέπουμε το Training και Validation Loss σε κάθε εποχή κατα την διάρκεια της εκπαίδευσης του μοντέλου DenseNet. Η συμπεριφορά των στρατηγικών και τα διαγράμματα είναι πολύ παρόμοια με αυτά του RenNet. Βλέπουμε ότι υπάρχει μία μεγαλύτερη διακύμανση στις τιμές του Validation Losstο οποίο οφείλεται στο ότι το μοντέλο εκπαιδεύεται σε λιγότερες εποχές.



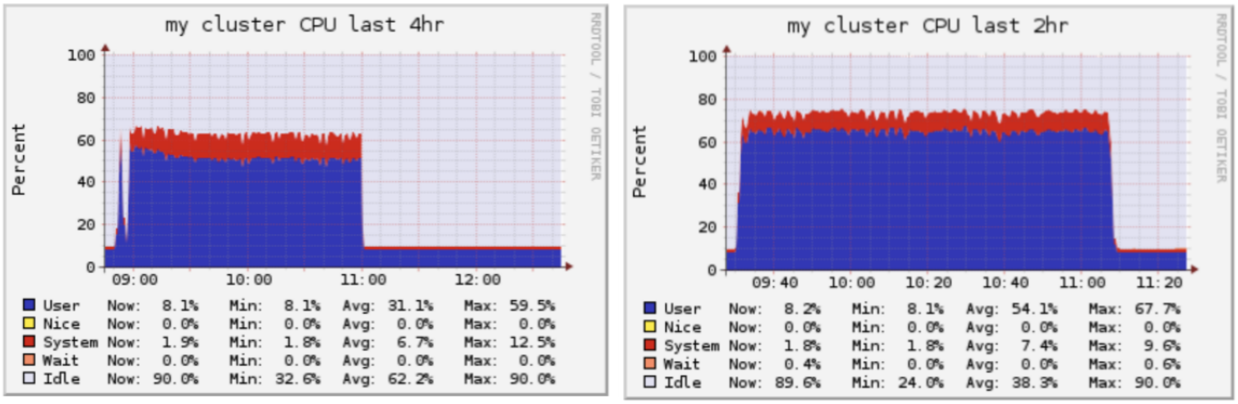
(α′) DenseNet Training Accuracy (β′) DenseNet Validation Accuracy

Εικόνα 5.7: DenseNet Accuracy

Στην εικόνα 5.7 βλέπουμε το Training και Validation Accuracy σε κάθε εποχή κατα την διάρκεια της εκπαίδευσης. Οι τιμές στα δύο διαγράμματα είναι παρόμοιες με αυτές του ResNet.

5.2.2 Ganglia DenseNet

CPU DenseNet

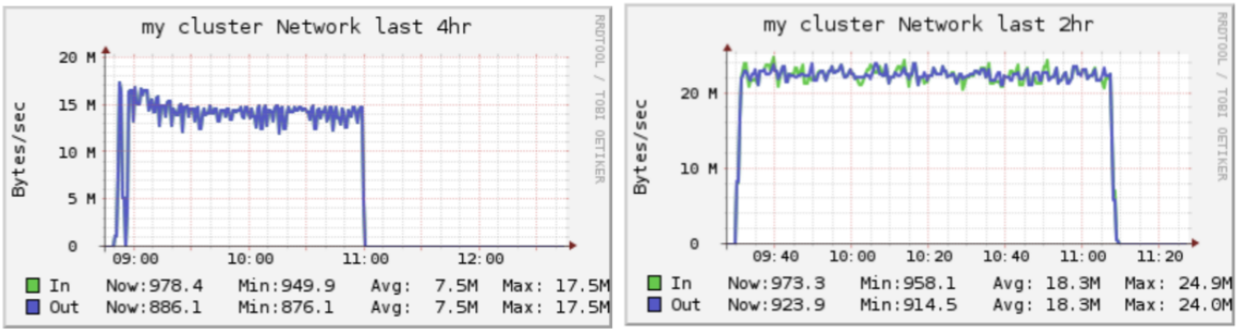


(α′) CPU Multi Worker Mirrored Strategy (β′) CPU Parameter Server Strategy

Εικόνα 5.8: CPU DenseNet

Στις εικόνες 5.8α΄ και 5.8β΄ βλέπουμε την χρησιμοποίηση της CPU του cluster από τις δύο στρατηγικές Multi Worker Mirrored Strategy και Parameter Server Strategy. Κατα την εκπαίδευση με την Multi Worker Mirrored Strategy παρατηρούμε ότι η χρησιμοποίηση της CPU κυμαίνεται στο **68%**. Πιο ανεβασμένες τιμές χρησιμοποίησης της CPU παρατηρούνται για την Parameter Server Strategy, περίπου στο **76%**. Οι τιμές είναι όμοιες με αυτές του ResNet.

Δίκτυο DenseNet

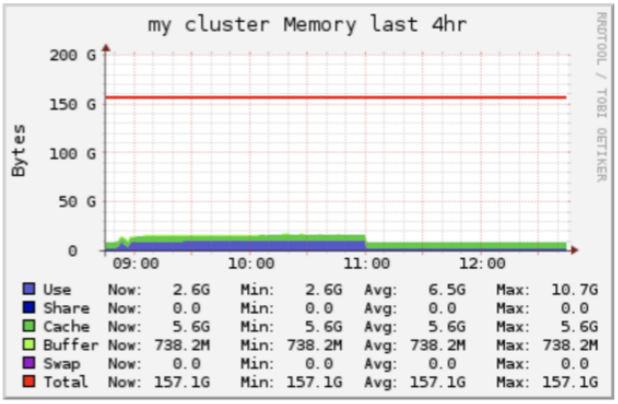


(α΄) Δίκτυο Multi Worker Mirrored Strategy (β΄) Δίκτυο Parameter Server Strategy

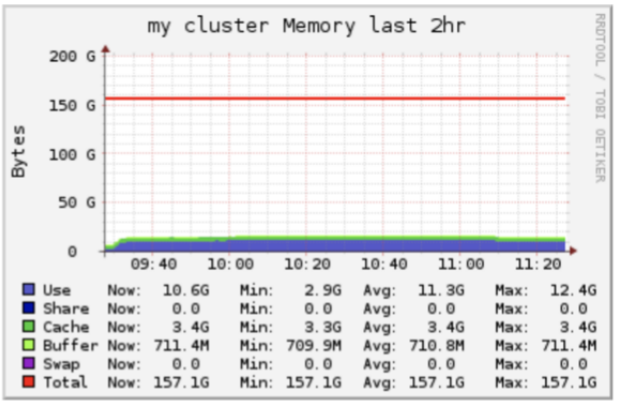
Εικόνα 5.9: Δίκτυο DenseNet

Στις εικόνες 5.4α΄ και 5.4β΄ βλέπουμε την επιβάρυνση του δικτύου του cluster κατα την διάρκεια της εκπαίδευσης από τις δύο στρατηγικές Multi Worker Mirrored Strategy και Parameter Server Strategy. Οι τιμές που παρατηρούνται είναι περίπου **17M** Bytes/sec και **24M** Bytes/sec για Multi Worker Mirrored Strategy και Parameter Server Strategy αντίστοιχα. Οι τιμές εδώ είναι αρκετά κατεβασμένες σε σχέση με του ResNet καθώς το Densenet που έχουμε χρησιμοποιήσει σε αυτήν την περίπτωση είναι μικρότερο και έχει πολύ λιγότερες παραμέτρους από το ResNet και συνεπώς το δίκτυο δεν φορτώνεται τόσο πολύ. Παρόλα αυτά οι διαφορές στις τιμές μεταξύ Multi Worker Mirrored Strategy και Parameter Server Strategy είναι όμοιες με αυτές του ResNet.

Μνήμη DenseNet



(α′) Μνήμη Multi Worker Mirrored Strategy



(β′) Μνήμη Parameter Server Strategy

Εικόνα 5.10: Μνήμη DenseNet

Στις εικόνες 5.10α′ και 5.10β′ βλέπουμε την κατανάλωση μνήμης RAM του cluster κατα την διάρκεια της εκπαίδευσης από τις δύο στρατηγικές Multi Worker Mirrored Strategy και Parameter Server Strategy. Οι τιμές που παρατηρούνται είναι **10.7GB** για την Multi Worker Mirrored Strategy και **12.4GB** για την Parameter Server Strategy. Όπως αναφέρθηκε η μνήμη που χρησιμοποιείται είναι πολύ λιγότερη από το ResNet λόγο μεγέθους του μοντέλου.

5.3 Σχολιασμός

Παρακάτω σχολιάζονται τα αποτελέσματα από τα πειράματα που πραγματοποιήθηκαν για τα δύο δίκτυα το ResNet με σύνολο δεδομένων το CIFAR-10 και το DenseNet με το σύνολο δεδομένων Fashion-MNIST. Τα αποτελέσματα και στις δύο περιπτώσεις έχουν παρόμοιο μοτίβο και βγάζουν τα ίδια συμπεράσματα.

5.3.1 Multi Worker Mirrored Strategy

Η στρατηγική Multi Worker Mirrored Strategy παρατηρείται από τους πίνακες 5.1 και 5.2 ότι πετυχαίνει την μεγαλύτερη ακρίβεια. Αυτό συμβαίνει καθώς η στρατηγική αυτή όπως έχει αναφερθεί είναι σύγχρονη, δηλαδή μετά την εκπαίδευση σε ένα batch δεδομένων οι εργάτες, ανταλλάζουν τα gredients που υπολόγισαν και τέλος ενημερώνουν τις παραμέτρους τους. Συνεπώς είναι ένα λογικό και αναμενόμενο αποτέλεσμα αφού ο αλγόριθμος σύγκλησης παραμένει ίδιος με την περίπτωση ενός single node.

Λόγω του συγχρονισμού όμως οι εργάτες όταν τελειώσουν τον υπολογισμού των gradients πρέπει να περιμένουν τα τελειώσει και ο πιο αργός εργάτης για να ανταλλάξουν τις τιμές, να ενημερώσουν το μοντέλο τους και να συνεχίσουν την εκπαίδευση με το επόμενο batch. Αυτό έχει σαν αποτέλεσμα σε συνδιασμό με την καθυστέρηση λόγο συγχονισμού η χρησιμοποίηση της υπολογιστικής μονάδας, στην περίπτωση μας των CPU να μένει σχετικά χαμηλά και ο χρόνος εκπαίδευσης να είναι μεγάλος, όπως φαίνεται και από τις εικόνες 5.3α΄ και 5.8α΄ Επίσης από τους πίνακες 5.1 και 5.2 παρατηρείται ότι έχουν τον μεγαλύτερο χρόνο εκπαίδευσης.

5.3.2 Parameter Server Strategy

Η στρατηγική Parameter Server Strategy όπως παρατηρείται από τους πίνακες 5.1 και 5.2 πετυχαίνει τον καλύτερο χρόνο εκπαίδευσης. Αυτό συμβαίνει καθώς η στρατηγική αυτή είναι ασύγχρονη. Κάθε εργάτης λειτουργεί ανεξάρτητα από τους υπόλοιπους. Οι εργάτες λαμβάνουν το ενημερωμένο μοντέλο από τον parameter server και ξεκινάνε την εκπαίδευση σε ένα batch δεδομένων. Όταν τελειώσουν την εκπαίδευση τους στέλνουν τα gredients στον parameter server αυτός ενημερώνει επι τόπου τις παραμέτρους του μοντέλου, τις στέλνει στον εργάτη και αυτός είναι έτοιμος να συνεχίσει την εκπαίδευση με το επόμενο batch δεδομένων. Με αυτόν τον τρόπο δεν καθυστερεί περιμένοντας να τελειώσουν και οι υπόλοιποι εργάτες την εκπαίδευση τους και έχει μεγαλύτερη χρησιμοποίηση της υπολογιστικής μονάδας CPU όπως φαίνεται και στις εικόνες 5.3β΄ και 5.8β΄.

Από την άλλη λόγω της ασύγχρονης εκπαίδευσης ο αλγόριθμος σύγκλισης αλλάζει. Ένας εργάτης όταν ξεκινάει την εκπαίδευση σε ένα batch λαμβάνει κάθε φορά τις ενημερωμένες παραμέτρους από τον parameter server αλλά όταν τελειώσει την εκπαίδευση και στείλει πίσω στον parameter server τα gradients οι παράμετροι του μοντέλου έχουν αλλάξει από ενημερώσεις άλλων εργατών και συνεπώς η ενημέρωση που στέλνει ο εργάτης μπορεί να μην είναι πάντα η βέλτιστη και προς την σωστή κατεύθυνση. Αυτό έχει σαν αποτέλεσμα η σύγκλιση του μοντέλου με την στρατηγική Parameter Server Strategy να μην είναι τόσο ακριβείς. Μπορεί να παρατηρηθεί από τους πίνακες 5.1 και 5.2 ότι η στρατηγική Parameter Server Strategy έχει την χαμηλότερη ακρίβεια πρόβλεψης.

5.3.3 Σύγκριση Στρατηγικών

Η στρατηγική Multi Worker Mirrored Strategy σε σχέση με την στρατηγική Parameter Server Strategy παρέχει μεγαλύτερη ακρίβεια πρόβλεψης αλλά έχει πιο μεγάλο χρόνο εκπαίδευσης. Όπως βλέπουμε από στον πίνακα 5.1 στην περίπτωση του ResNet η απόδοση του μοντέλου είναι 92.71% και 90.81% αντίστοιχα για Multi Worker Mirrored Strategy και Parameter Server Strategy. Αντίθετα ο χρόνος εκτέλεσης είναι 7 ώρες και 38 λεπτά για την Multi Worker Mirrored Strategy και 6 ώρες και 36 λεπτά για την Parameter Server Strategy. Όμοιο μοτίβο βλέπουμε και στα πειράματα του DenseNet όπου η απόδοση του μοντέλου και χρόνος εκπαίδευσης

είναι 94.51% με χρόνο 2 ώρες και 93.99% με χρόνο 1 ώρα και 33 λεπτά αντίστοιχα για Multi Worker Mirrored Strategy και Parameter Server Strategy.

Μία σημαντική παρατήρηση για τον χρόνο εκτέλεσης αποτελεί ότι το cluster στο οποίο έγιναν τα πειράματα αποτελείται από όμοιους υπολογιστές και συνεπώς υπάρχει πολύ μικρή απόκλιση στην υπολογιστική ισχύ. Έτσι το πρόβλημα του αργού εργάτη που αντιμετωπίζει η στρατηγική Multi Worker Mirrored Strategy δεν είναι τόσο χρονοβόρο όσο θα ήταν σε ένα cluster όπου υπάρχει μία ετερογένεια. Αντίθετα σε ένα cluster με μεγάλη ετερογένεια η στρατηγική Parameter Server Strategy δεν θα αντιμετώπιζε κανένα πρόβλημα και θα είχε παρόμοια αποτελέσματα.

5.3.4 Strategy Switch

Με την στρατηγική Strategy Switch θελήσαμε με κάποιον τρόπο να εκμεταλλευτούμε τα πλεονεκτήματα και των δύο άλλων στρατηγικών, δηλαδή την πολύ υψηλή ακρίβεια πρόβλεψης της στρατηγικής Multi Worker Mirrored Strategy και τον μικρό χρόνο εκπαίδευσης της στρατηγικής Parameter Server Strategy. Στην Strategy Switch στρατηγική η εκπαίδευση του μοντέλου ξεκινάει με την Multi Worker Mirrored Strategy στρατηγική και τελειώνει με την Parameter Server Strategy. Μία υπερπαράμετρος που δημιουργείται είναι το πότε θα γίνει η αλλαγή από την μία στρατηγική στην άλλη. Στα πειράματα που πραγματοποιήθηκαν δοκιμάστηκαν τρία διαφορετικά σημεία αλλαγής, στο 25%, στο 50% και στο 75% των εποχών.

Αρχικά να σημειωθεί ότι και στις τρεις περιπτώσεις υπήρχαν βελτιώσεις και στις δύο μετρικές, η ακρίβεια πρόβλεψης που πετύχανε είναι μεγαλύτερη από αυτήν της στρατηγικής Parameter Server Strategy και ο χρόνος εκπαίδευσης είναι μικρότερος από τον αντίστοιχο της στρατηγικής Multi Worker Mirrored Strategy.

Ακρίβεια πρόβλεψης

Όσον αφορα την ακρίβεια πρόβλεψης τα καλύτερα αποτελέσματα επιτεύχθηκαν απο την Strategy Switch 50% και 75%. Σύμφωνα με τον πίνακα 5.1 και τα αποτελέσματα του ResNet η στρατηγική Strategy Switch 50% πέτυχε ακρίβεια 92.54% και η στρατηγική Strategy Switch 75% 94.29%. Και στις δύο περιπτώσεις η ακρίβεια πρόβλεψης είναι πολύ υψηλή και πολύ κοντα στην ακρίβεια πρόβλεψης της στρατηγικής Multi Worker Mirrored Strategy.

Αντίστοιχα συμπεράσματα βγάζουμε και απο τα πειράματα για το DenseNet πίνακας 5.2. Η στρατηγική Strategy Switch 50% πέτυχε ακρίβεια 94.48% και η στρατηγική Strategy Switch 75% 94.50%. Εδώ η ακρίβεια πρόβλεψης αυτών των στρατηγικών ταυτίζεται με αυτή της Multi Worker Mirrored Strategy. Η στρατηγική Strategy Switch 25% τόσο στο ResNet όσο και στο DenseNet πέτυχε χαμηλότερα αποτελέσματα. Στο ResNet η ακρίβεια πρόβλεψης ήταν 91.70% και στο DenseNet 94.26%.

Συμπεραίνουμε ότι η εκπαίδευση του μοντέλου με την στρατηγική Multi Worker Mirrored Strategy μόνο με το 25% των εποχών δεν αρκεί για να πλησιάσει ή να φτάσει την βέλτιστη ακρίβεια πρόβλεψης. Αντίθετα το 50% παρατηρείται ότι είναι αρκετό για να φτάσει το μοντέλο σε πολύ υψηλά επίπεδα ακρίβειας.

Χρόνος εκτέλεσης

Είναι λογικό όσο μεγαλύτερο είναι το ποσοστό των εποχών όπου το μοντέλο εκπαιδεύεται με την στρατηγική Multi Worker Mirrored Strategy τόσο πιο αργή θα είναι και η διαδικασία της εκπαίδευσης. Αυτό επιβεβαιώθηκε και από τα πειράματα, αφού τόσο για το ResNet όσο και για το DenseNet οι χρόνοι εκπαίδευσεις ανέβαιναν αναλογικά με την αύξηση του ποσοστού εκπαίδευσης με την στρατηγική Multi Worker Mirrored Strategy. Οι τιμές φαίνονται στους πίνακες 5.1 και 5.2.

Άλλα σχόλια

Παρατηρήθηκε ότι για την στρατηγική Strategy Switch το validation loss μικραίνει αρκετά και είναι μικρότερο από τις άλλες δύο στρατηγικές. Αυτό ενδέχεται να σημαίνει ότι τα μοντέλα αυτά παρόλο που έχουν ίδια ή παρόμοια ακρίβεια με την στρατηγική Multi Worker Mirrored Strategy μπορεί να γενικεύουν καλύτερα και σε ένα μεγαλύτερο test dataset να είχαν καλύτερα αποτελέσματα.

Κεφάλαιο 6

Συμπεράσματα και Επεκτάσεις

Στην εργασία αυτή αρχικά μελετήθηκαν οι βασικές στρατηγικές data parallelism σε κατανεμημένο περιβάλλον. Η πρώτη αφορά σύγχρονη στρατηγική με μία αποκεντρωμένη αρχιτεκτονική την All-reduce και η δεύτερη αφορά ασύγχρονη εκπαίδευση με μία συγκεντρωτική αρχιτεκτονική την Parameter Server. Για τις δύο αυτές στρατηγικές χρησιμοποιήθηκαν οι βιβλιοθήκες του TensorFlow, για την πρώτη η Multi Worker Mirrored Strategy και για την δεύτερη η Parameter Server Strategy. Γίνανε πειράματα με δύο μεγάλα και γνωστά μοντέλα νευρωνικών δικτύων που αφορούν την όραση υπολογιστών, το ResNet και το DenseNet. Για τα πειράματα αυτά χρησιμοποιήθηκαν δύο κλασικά σύνολα δεδομένων ταξινόμησης εικόνων, το CIFAR-10 και το Fashion-MNIST.

Αποτελέσματα από αυτά τα πειράματα δείξανε ότι η σύγχρονη στρατηγική Multi Worker Mirrored Strategy πετυχαίνει μεγαλύτερη ακρίβεια πρόβλεψης από την στρατηγική Parameter Server Strategy αλλά ο χρόνος εκπαίδευσης της είναι αρκετά μεγαλύτερος.

Στην συνέχεια προσπαθήσαμε να υλοποιήσουμε μία υβριδική στρατηγική όπου εκμεταλλεύεται τα πλεονεκτήματα των άλλων δύο στρατηγικών. Την στρατηγική Strategy Switch η οποία ξεκινάει την εκπαίδευση του μοντέλου με την στρατηγική Multi Worker Mirrored Strategy για ένα ποσοστό των εποχών και στην συνέχεια αλλάζει στρατηγική και το μοντέλο εκπαιδεύεται με την Parameter Server Strategy. Δοκιμάστηκαν τρία διαφορετικά σημεία αλλαγής, στο 25%, στο 50% και στο 75% των εποχών εκπαίδευσης.

Τα καλύτερα αποτελέσματα παρατηρούνται όταν η αλλαγή μεταξύ των δύο στρατηγικών γίνεται στο 50% των εποχών. Σε αυτή τη περίπτωση η ακρίβεια πρόβλεψης που πετυχαίνει το μοντέλο είναι πολύ κοντά αν όχι ίση με την βέλτιστη ακρίβεια πρόβλεψης και ο χρόνος σε σχέση με την Multi Worker Mirrored Strategy είναι πολύ βελτιωμένος. Στο ResNet όπου είναι πιο εμφανής οι χρονικές διαφορές λόγω μεγέθους η Multi Worker Mirrored Strategy έκανε 7 ώρες και 38 λεπτά με ακρίβεια 92.71% ενώ στην Strategy Switch 50% έκανε 6 ώρες και 57 λεπτά με ακρίβεια 92.54%.

Συνεπώς η Strategy Switch 50% μπορεί να αποτελέσει την καλύτερη επιλογή για εκπαίδευση ενός μεγάλου νευρωνικού δικτύου σε κατανεμημένο περιβάλλον, όταν

απαιτείται μικρός χρόνος εκπαίδευσης με υψηλή ακρίβεια πρόβλεψης.

Η παρούσα διπλωματική εργασία θα μπορούσε να επεκταθεί στους ακόλουθους άξονες:

- Γενίκευση των αποτελεσμάτων πραγματοποιώντας πειράματα σε διαφορετικούς τομείς των νευρωνικών δικτύων που απαιτούν κατανεμημένη εκπαίδευση όπως τα Natural Language Processing (NLP)
- Εύρεση σημείου αλλαγής στρατηγικής με πιο στοχαστικό τρόπο για κάθε πρόβλημα, όπως με δυαδική αναζήτηση.
- Πειραματική ανάλυση της στρατηγικής σε περισσότερα και πιο ισχυρά υπολογιστικά μηχανήματα όπως ένα μεγάλο cluster από GPU.
- Δοκιμή της αντίστροφης στρατηγικής, πρώτα εκπαίδευση με την στρατηγική Parameter Server Strategy και στην συνέχεια με την Multi Worker Mirrored Strategy

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. «ImageNet Classification with Deep Convolutional Neural Networks». In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, 1097–1105.

[2] T. Yu, J. Meng, and J. Yuan. «Multi-view Harmonized Bilinear Network for 3D Object Recognition». In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, 2018, pp. 186–194. DOI: 10.1109/CVPR.2018.00027. URL: https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00027.

[3] Geert Litjens et al. «A Survey on Deep Learning in Medical Image Analysis». In: *CoRR* abs/1702.05747 (2017). arXiv: 1702.05747. URL: http://arxiv.org/abs/1702.05747.

[4] Dario Amodei et al. «Deep Speech 2 : End-to-End Speech Recognition in English and Mandarin». In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, 2016, pp. 173–182. URL: https://proceedings.mlr.press/v48/amodei16.html.

[5] Wei-Cheng Chang et al. «Taming Pretrained Transformers for Extreme Multi-Label Text Classification». In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '20. Virtual Event, CA, USA: Association for Computing Machinery, 2020, 3163–3171. ISBN: 9781450379984. DOI: 10.1145/3394486.3403368. URL: https://doi.org/10.1145/3394486.3403368.

[6] Ahmed Shehab Khan et al. «Group-Level Emotion Recognition Using Deep Models with A Four-Stream Hybrid Network». In: *Proceedings of the 20th ACM International Conference on Multimodal Interaction*.

ICMI '18. Boulder, CO, USA: Association for Computing Machinery, 2018, 623–629. ISBN: 9781450356923. DOI: 10.1145/3242969.3264987. URL: https://doi.org/10.1145/3242969.3264987.

[7] Pitch Patarasuk and Xin Yuan. «Bandwidth optimal all-reduce algorithms for clusters of workstations». In: *Journal of Parallel and Distributed Computing* 69.2 (2009), pp. 117–124. ISSN: 0743-7315. DOI: https://doi.org/10.1016/j.jpdc.2008.09.002. URL: https://www.sciencedirect.com/science/article/pii/S0743731508001767.

[8] Mu Li. «Scaling Distributed Machine Learning with the Parameter Server». In: *Proceedings of the 2014 International Conference on Big Data Science and Computing*. BigDataScience '14. Beijing, China: Association for Computing Machinery, 2014. ISBN: 9781450328913. DOI: 10.1145/2640087.2644155. URL: https://doi.org/10.1145/2640087.2644155.

[9] Martín Abadi et al. «TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems». In: *CoRR* abs/1603.04467 (2016). arXiv: 1603.04467. URL: http://arxiv.org/abs/1603.04467.

[10] Adam Paszke et al. «PyTorch: An Imperative Style, High-Performance Deep Learning Library». In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[11] Kaiming He et al. «Deep Residual Learning for Image Recognition». In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[12] Gao Huang et al. «Densely Connected Convolutional Networks». In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243.

[13] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. «Learning representations by back-propagating errors». In: *Nature* 323.6088 (1986), pp. 533–536. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0.

[14] Sebastian Ruder. «An overview of gradient descent optimization algorithms». In: *CoRR* abs/1609.04747 (2016). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[15] Xavier Glorot and Y. Bengio. «Understanding the difficulty of training deep feedforward neural networks». In: *Journal of Machine Learning Research - Proceedings Track* 9 (Jan. 2010), pp. 249–256.

[16] Vinod Nair and Geoffrey E. Hinton. «Rectified Linear Units Improve Restricted Boltzmann Machines». In: *Proceedings of the 27th International Conference on International Conference on Machine Learning*. ICML'10. Haifa, Israel: Omnipress, 2010, 807–814. ISBN: 9781605589077.

[17] Jun Han and Claudio Moraga. «The influence of the sigmoid function parameters on the speed of backpropagation learning». In: *From Natural to Artificial Neural Computation*. Ed. by José Mira and Francisco Sandoval. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 195–201. ISBN: 978-3-540-49288-7.

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. «Deep learning». In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539.

[19] Boris Hanin. *Which Neural Net Architectures Give Rise To Exploding and Vanishing Gradients?* 2018. arXiv: 1801.03744 [stat.ML].

[20] Siddharth Sharma, Simone Sharma, and Anidhya Athaiya. «ACTIVATION FUNCTIONS IN NEURAL NETWORKS». In: *International Journal of Engineering Applied Sciences and Technology* 04 (May 2020), pp. 310–316. DOI: 10.33564/IJEAST.2020.v04i12.054.

[21] Keiron O'Shea and Ryan Nash. «An Introduction to Convolutional Neural Networks». In: *CoRR* abs/1511.08458 (2015). arXiv: 1511.08458. URL: http://arxiv.org/abs/1511.08458.

[22] G. Cybenko. «Approximation by superpositions of a sigmoidal function». In: *Mathematics of Control, Signals and Systems* 2.4 (1989), pp. 303–314. DOI: 10.1007/BF02551274. URL: https://doi.org/10.1007/BF02551274.

[23] Kaiming He et al. «Identity Mappings in Deep Residual Networks». In: *CoRR* abs/1603.05027 (2016). arXiv: 1603.05027. URL: http://arxiv.org/abs/1603.05027.

[24] Zhihao Jia, Matei Zaharia, and Alex Aiken. «Beyond Data and Model Parallelism for Deep Neural Networks». In: *CoRR* abs/1807.05358 (2018). arXiv: 1807.05358. URL: http://arxiv.org/abs/1807.05358.

[25] Ruben Mayer, Christian Mayer, and Larissa Laich. «The TensorFlow Partitioning and Scheduling Problem: It's the Critical Path!» In: *CoRR* abs/1711.01912 (2017). arXiv: 1711.01912. URL: http://arxiv.org/abs/1711.01912.

[26] Azalia Mirhoseini et al. «Device Placement Optimization with Reinforcement Learning». In: *CoRR* abs/1706.04972 (2017). arXiv: 1706.04972. URL: http://arxiv.org/abs/1706.04972.

[27] James Cipar et al. «Solving the Straggler Problem with Bounded Staleness». In: *14th Workshop on Hot Topics in Operating Systems (HotOS XIV)*. Santa Ana Pueblo, NM: USENIX Association, May 2013. URL: https://www.usenix.org/conference/hotos13/session/cipar.

[28] Shijian Li et al. «Sync-Switch: Hybrid Parameter Synchronization for Distributed Deep Learning». In: *CoRR* abs/2104.08364 (2021). arXiv: 2104.08364. URL: https://arxiv.org/abs/2104.08364.

[29] Qirong Ho et al. «More Effective Distributed ML via a Stale Synchronous Parallel Parameter Server». In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, 1223–1231.

[30] Xing Zhao et al. «Dynamic Stale Synchronous Parallel Distributed Training for Deep Learning». In: *CoRR* abs/1908.11848 (2019). arXiv: 1908.11848. URL: http://arxiv.org/abs/1908.11848.

[31] Víctor Campos et al. «Distributed training strategies for a computer vision deep learning algorithm on a distributed GPU cluster». In: *Procedia Computer Science* 108 (2017). International Conference on Computational Science, ICCS 2017, 12-14 June 2017, Zurich, Switzerland, pp. 315–324. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.05.074. URL: https://www.sciencedirect.com/science/article/pii/S1877050917306129.

[32] Alexander Smola and Shravan Narayanamurthy. «An Architecture for Parallel Topic Models». In: *Proc. VLDB Endow.* 3.1–2 (Sept. 2010), 703–710. ISSN: 2150-8097. DOI: 10.14778/1920841.1920931. URL: https://doi.org/10.14778/1920841.1920931.

[33] Jeffrey Dean and Sanjay Ghemawat. «MapReduce: Simplified Data Processing on Large Clusters». In: *Commun. ACM* 51.1 (Jan. 2008), 107–113. ISSN: 0001-0782. DOI: 10.1145/1327452.1327492. URL: https://doi.org/10.1145/1327452.1327492.

[34] Andrew Gibiansky. «Bringing HPC techniques to deep learning.(2017)». In: *URL http://research. baidu. com/bringing-hpc-techniquesdeep-learning* (2017).

[35] Huasha Zhao and John Canny. «Butterfly Mixing: Accelerating Incremental-Update Algorithms on Clusters». In: *Proceedings of the 2013 SIAM International Conference on Data Mining (SDM)*, pp. 785–793. DOI: 10.1137/1.9781611972832.87. eprint: https://epubs.siam.org/

doi/pdf/10.1137/1.9781611972832.87. URL: https://epubs.siam.org/doi/abs/10.1137/1.9781611972832.87.

[36] Alekh Agarwal et al. «A Reliable Effective Terascale Linear Learning System». In: *CoRR* abs/1110.4198 (2011). arXiv: 1110.4198. URL: http://arxiv.org/abs/1110.4198.

[37] Matthew L Massie, Brent N Chun, and David E Culler. «The ganglia distributed monitoring system: design, implementation, and experience». In: *Parallel Computing* 30.7 (2004), pp. 817–840. ISSN: 0167-8191. DOI: https://doi.org/10.1016/j.parco.2004.04.001. URL: https://www.sciencedirect.com/science/article/pii/S0167819104000535.

[38] Apache Software Foundation. *Hadoop*. Version 0.20.2. Feb. 19, 2010. URL: https://hadoop.apache.org.

[39] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. «CIFAR-10 (Canadian Institute for Advanced Research)». In: (). URL: http://www.cs.toronto.edu/~kriz/cifar.html.

[40] Han Xiao, Kashif Rasul, and Roland Vollgraf. «Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms». In: *CoRR* abs/1708.07747 (2017). arXiv: 1708.07747. URL: http://arxiv.org/abs/1708.07747.

[41] Sotiris Kotsiantis, Dimitris Kanellopoulos, and P. Pintelas. «Data Preprocessing for Supervised Learning». In: *International Journal of Computer Science* 1 (Jan. 2006), pp. 111–117.

[42] Connor Shorten and Taghi M. Khoshgoftaar. «A survey on Image Data Augmentation for Deep Learning». In: *Journal of Big Data* 6.1 (2019), p. 60. DOI: 10.1186/s40537-019-0197-0. URL: https://doi.org/10.1186/s40537-019-0197-0.

[43] Sinno Jialin Pan and Qiang Yang. «A Survey on Transfer Learning». In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359. DOI: 10.1109/TKDE.2009.191.