



ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Συμπίεση Μοντέλων Χρονοσειρών

του

Δημήτρη Υφαντίδη - 3938

Εκπόνηση πτυχιακής ως μέρος του
Προπτυχιακού Τίτλου Σπουδών

στο

Τμήμα Πληροφορικής,
Σχολή Θετικών Επιστημών

Επιβλέπων Καθηγητής: Δρ. Γεώργιος Κεραμίδας

Ιούλιος 2024

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

Σύνοψη

Σχολή Θετικών Επιστημών

Τμήμα Πληροφορικής

Προπτυχιακός Τίτλος Σπουδών

του Δημήτρη Υφαντίδη - 3938

Τα δεδομένα χρονοσειρών (Time Series Data) αποτελούν αντικείμενο ιδιαίτερου επιστημονικού και εμπορικού ενδιαφέροντος. Σε αντίθεση με τα χωρικά (spatial) ή πινακοειδή (tabular) δεδομένα, οι χρονοσειρές μπορούν να εκφράσουν ποσοτικές αλλαγές στο πέρασμα του χρόνου, συνεισφέροντας έτσι στην απόκτηση γνώσης από το παρελθόν και, κατά συνέπεια, στη δυνατότητα πρόβλεψης του μέλλοντος με σκοπό τη λήψη καλύτερων αποφάσεων. Δεδομένου και της ευρείας χρήσης της Βαθιάς Μάθησης (Deep Learning - DL) στην ανάλυση δεδομένων, η χρήση Βαθιών Νευρωνικών Δικτύων (Deep Neural Networks - DNNs) αποτελεί εξαιρετική επιλογή για την μελέτη τέτοιων ακολουθιακών δεδομένων. Για αυτόν το λόγο, εδώ και αρκετές δεκαετίες, συνεχώς μελετώνται και αναπτύσσονται ειδικές νευρωνικές αρχιτεκτονικές που δίνουν ιδιαίτερη προσοχή στον παράγοντα του χρόνου. Αυτές οι αρχιτεκτονικές κατατάσσονται στην οικογένεια των Αναδρομικών Νευρωνικών Δικτύων (Recurrent Neural Networks - RNNs), όπως τα Μακρά Δίκτυα Βραχείας Μνήμης (Long Short-Term Memory - LSTMs), οι Transformers, κλπ. Επιπλέον, για την ανάλυση χρονοσειρών χρησιμοποιούνται και Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNNs) που ανήκουν στην οικογένεια των Νευρωνικών Δικτύων Εμπρόσθιας Τροφοδότησης (Feedforward Neural Networks - FNNs). Αν και η ανάπτυξη τέτοιων αρχιτεκτονικών έχει μελετηθεί αρκετά, λίγη μέριμνα έχει αποδοθεί από την επιστημονική κοινότητα για τη συμπίεση αυτών. Η εν λόγω συμπίεση αποτελεί ζήτημα ύψιστης σημασίας καθώς αυτή κάνει εφικτή τη χρήση τους σε συσκευές περιορισμένων πόρων μνήμης και επεξεργαστικών δυνατοτήτων, όπως είναι οι επιταχυντές (accelerators), οι μικροελεγκτές (microcontrollers), οι κινητές συσκευές (smartphones), κλπ. Στην παρούσα εργασία εστιάζουμε κυρίως στη συμπίεση μοντέλων με κβαντιστικές μεθόδους (Quantization - QA) παρά σε άλλες μεθόδους συμπίεσης. Αρχικά, θα αναλύσουμε τον αντίκτυπο του κβαντισμού σε απλά LSTM μοντέλα και στη συνέχεια θα προχωρήσουμε σε πιο σύνθετα μοντέλα - κάποια CNNs και κάποια μεικτά νευρωνικά δίκτυα που αποτελούνται και από συνελικτικά αλλά και από LSTM στρώματα.



Aristotle University of Thessaloniki

Compression of Time Series Models

Dimitrios Yfantis - 3938

Thesis as part
of Undergraduate Studies

Faculty of Sciences
Informatics Department

Supervising Professor: Dr. Georgios Keramidas

July 2024

ARISTOTLE UNIVERSITY OF THESSALONIKI

Abstract

Faculty of Sciences

School of Informatics

Undergraduate Studies

Dimitrios Yfantis - 3938

Time Series Data is a subject of particular scientific and commercial interest. Contrary to cross-sectional or spatial/tabular data, time series have the ability to express quantitative fluctuations over time, thus contributing to the acquisition of knowledge from the past and, as a consequence, to the ability to predict the future and achieve better decision making. Given the widespread use of Deep Learning (DL) in data analysis, the use of Deep Neural Networks (DNNs) arises as an excellent choice for the study of such sequential data. For this reason, specific neural architectures that pay special attention to the factor of time have been continuously studied and developed in the past decades. These architectures are classified in the family of Recurrent Neural Networks (RNNs), such as Long Short-Term Memory Networks (LSTMs), Transformers, etc. In addition, Convolutional Neural Networks (CNNs) belonging to the family of Feedforward Neural Networks (FNNs) are also used for time series analysis. Although the development of such architectures has been well studied, little concern has been attributed by the scientific community to their compression. The compression in question is a matter of utmost importance as it enables their use in devices with limited memory resources and processing capabilities, such as accelerators, microcontrollers, mobile devices, etc. In this thesis we mainly focus on model compression using Quantization (QA) rather than other compression methods. First, we will analyse the impact of quantization on simple LSTM models and then move on to more complex models - some of which are CNNs while others are mixed neural networks consisting of both convolutional and LSTM layers.

Eυχαριστίες

Θα ήθελα να ευχαριστήσω την οικογένεια μου, τους φίλους και συναδέλφους μου που με στήριξαν σημαντικά κατά τη διάρκεια της φοίτησής μου. Επιπλέον, θα ήθελα να ευχαριστήσω πολύ τον Επίκουρο Καθηγητή του τμήματός μας, Δρ. Γεώργιο Κεραμίδα και την υποψήφια διδάκτωρα Zahra Kokhazad για την ευκαιρία που μου έδωσαν να ασχοληθώ με το αντικείμενο αυτής της πτυχιακής καθώς και τη συνεργασία και την τεχνογνωσία που μου προσέφεραν κατά την εκπόνησή της.

Περιεχόμενα

Κατάλογος Σχημάτων	VIII
Κατάλογος Πινάκων	XI
Συντομογραφίες	XIII
1 Εισαγωγή	1
2 Βασικές Αρχές και Επισκόπηση Βιβλιογραφίας	5
2.1 Δεδομένα Χρονοσειρών	5
2.2 Τεχνικές Επεξεργασίας Χρονοσειρών	8
2.3 Τεχνητά Νευρωνικά Δίκτυα	8
2.4 Συνελικτικά Νευρωνικά Δίκτυα	11
2.5 Αναδρομικά Νευρωνικά Δίκτυα και Μοντέλα Δίκτυα Βραχείας Μνήμης	12
2.6 Προβλήματα των Νευρωνικών Δικτύων	13
3 Συμπίεση των Νευρωνικών Δικτύων	14
3.1 Μέθοδοι Συμπίεσης και Στόχοι τους	14
3.2 Ενδεικτικές Εφαρμογές Συμπίεσης Μοντέλων	15
3.3 Στρατηγικές Συμπίεσης με Κβαντισμό	16
3.4 Κβαντισμός Μετά την Εκπαίδευση (PTQ)	16
3.5 Εκπαίδευση με Επίγνωση Κβαντισμού (QAT)	19
3.6 Προκλήσεις	21
4 Περιβάλλον Ανάπτυξης και Πειραματικά Μοντέλα	22
4.1 Χαρακτηριστικά Συστήματος	22
4.2 Οι βιβλιοθήκες που χρησιμοποιήθηκαν	23
4.3 Τα Εξεταζόμενα Μοντέλα	23
4.3.1 Απλό LSTM δίκτυο για ανάλυση συναισθήματος (έτοιμο)	24
4.3.2 Απλό LSTM δίκτυο για ανάλυση συναισθήματος (υλοποιήθηκε)	24
4.3.3 Ordonez 2016 Deep Original Model	25
4.3.4 Smart Watch Model	25
4.3.5 BSPC Model	26
4.3.6 BSPC GAP Model	26
4.4 Μετατροπή σε Μοντέλο TensorFlow Lite	30

4.5	Βελτιστοποιήσεις του Μετατροπέα TFLite	30
4.5.1	Απλή Μετατροπή σε TFLite (Default Conversion)	31
4.5.2	Μετατροπή σε TFLite με Συμπίεση DRQ	31
4.5.3	Μετατροπή σε TFLite με Συμπίεση FIQ	31
5	Πειραματικά Δεδομένα και Ανάλυσή τους	34
5.1	Ο Αντίκτυπος του QA στα απλά μοντέλα	34
5.2	Αποτελέσματα του Μοντέλου Ordóñez	36
5.2.1	Περαιτέρω Ανάλυση με τον TFLite Analyzer	38
5.2.2	Περαιτέρω Ανάλυση με τον Quantization Debugger	41
5.2.3	Επιλεκτικός Κβαντισμός Στρωμάτων.	44
5.3	Τα Αποτελέσματα των Συνελικτικών Μοντέλων	45
5.3.1	Μέθοδοι Συμπίεσης που Χρησιμοποιήθηκαν	46
5.3.2	Αποσφαλμάτωση των Συνελικτικών Μοντέλων	51
6	Συμπεράσματα και Προτάσεις για Μελλοντική Έρευνα	64
6.1	Προτάσεις για Μελλοντική Έρευνα	65

Κατάλογος Σχημάτων

1.1	Τιμές μετοχών ως παράδειγμα δεδομένων χρονοσειρών (πηγή: tradingview.com)	1
1.2	(Smartwatch) Μια IoT συσκευή περιορισμένων δυνατοτήτων (πηγή: freepik.com)	2
2.1	Διαφορετικές αναπαραστάσεις μονοδιάστατων χρονοσειρών	6
2.2	(Heatmap) Αναπαράσταση πολυδιάστατων χρονοσειρών	7
2.3	Το μαθηματικό πρότυπο του νευρώνα σύμφωνα με τους McCulloch και Pitts (πηγή: medium.com).	9
2.4	Η αρχιτεκτονική του Πολυστρωματικού Perceptron.	10
2.5	CNN για την εξαγωγή χαρακτηριστικών από εικόνες (πηγή: medium.com) . . .	11
2.6	Οπτικές αναπαραστάσεις ενός RNN νευρώνα (πηγή: wikipedia.org).	12
2.7	Η αρχιτεκτονική ενός κυττάρου LSTM (πηγή: medium.com).	13
3.1	Οπτικοποίηση διαφορετικών μεθόδων συμπίεσης.	15
3.2	Οι βελτιστοποιήσεις που υποστηρίζει η TensorFlow κατά τη μετατροπή μοντέλων σε TFLite μορφή (πηγή: tensorflow.org).	18
3.3	Προδιαγραφές 8-bit συμπίεσης διαφορετικών ειδών τενσόρων της TensorFlow. Από εδώ επιβεβαιώνεται ότι τα βάρη περιορίζονται στο $[-128, 127]$ ενώ οι μεροληψίες παραμένουν στο εύρος των 32 bit. (πηγή: tensorflow.org)	20
4.1	Μετατροπή σε απλό μοντέλο TFLite. Όλοι οι τένσορες/στρώματα του παραγόμενου μοντέλου είναι τύπου float32. Η δεύτερη γραμμή πριν το τέλος είναι προαιρετική.	31
4.2	Μετατροπή σε TFLite μοντέλο με συμπίεση DRQ. Η TF προσπαθεί να μετατρέψει όλους τους τένσορες βαρών του παραγόμενου μοντέλου σε int8, ενώ όλοι οι άλλοι παραμένουν σε float32.	31
4.3	Python3 κώδικας για τη μετατροπή ενός μοντέλου σε μορφή TFLite με συμπίεση FIQ.	33
5.1	Πίνακες Σύγχυσης για τα τέσσερα TFLite μοντέλα Ordóñez.	37
5.2	Ένα μέρος του log file της αρχιτεκτονικής του Default TFLite μοντέλου. Εκτυπώνονται πρώτα τα υπογραφήματα (ένα υπογράφημα στη συγκεκριμένη περίπτωση) και ακολουθούν οι τένσορές τους, μαζί με τις ιδιότητες και τις μεταξύ τους σχέσεις που αυτοί έχουν.	39
5.3	Οι float32 τένσορες του FIQ-compressed TFLite μοντέλου. Το στιγμιότυπο οθόνης δεν αποτελεί απόσπασμα του πραγματικού log file της αρχιτεκτονικής, αλλά ενός αντίστοιχου αρχείου στο οποίο έχουν ταξινομηθεί οι τένσορες με βάση τον τύπο δεδομένων. Το εν λόγω αρχείο μπορεί εύκολα να προκύψει μετά από επεξεργασία του αρχικού log file χρησιμοποιώντας κανονικές εκφράσεις. . .	40

5.4	Στατιστικές μετρικές των διαφορών μεταξύ των FIQ-χβαντισμένων και των αντίστοιχων μη χβαντισμένων στρωμάτων του Ordonez 2016 Deep Original στα σχήματα α' έως δ'. Πέρα από αυτές τις μετρικές, ο Αποσφαλματωτής εξάγει και τους συντελεστές S και Z της εξίσωσης 3.1 (σχήματα ε' και στ').	42
5.5	Τα στρώματα του τελικού FIQ μοντέλου Ordonez, όπως ορίζονται από τον Αποσφαλματωτή. Η περιγραφή της αρχιτεκτονικής συμβαδίζει με την περιγραφή του πίνακα 4.3.	43
5.6	Τα ελαττωματικά στρώματα του τελικού FIQ μοντέλου Ordonez.	43
5.7	Στατιστικές μετρικές των διαφορών μεταξύ των FIQ-χβαντισμένων και των αντίστοιχων μη χβαντισμένων στρωμάτων του μοντέλου Smart Watch (φάσεις 1 &2), μαζί με τα S και Z . ΠΡΑΣΙΝΟ: Pre-Pruning Lean, ΚΟΚΚΙΝΟ: Pre-Pruning Fat, ΜΠΛΕ: Post-Pruning Lean, ΠΟΡΤΟΚΑΛΙ: Post-Pruning Fat.. .	52
5.8	Αντιστοιχίες των δεικτών του οριζόντιου άξονα των γραφημάτων του σχήματος 5.7 στα στρώματα των pre-pruning (φάση 1) και post-pruning (φάση 2) εκδοχών του SmartWatch μοντέλου.	53
5.9	Στατιστικές μετρικές των διαφορών μεταξύ των FIQ-χβαντισμένων και των αντίστοιχων μη χβαντισμένων στρωμάτων του μοντέλου SmartWatch, μαζί με τα S και Z . ΜΠΛΕ: Lean #1 (0-3 bottom-left), ΠΟΡΤΟΚΑΛΙ: Lean #2 (0-5 top-left), ΠΡΑΣΙΝΟ: Lean #3 (4-7 bottom-left), ΚΟΚΚΙΝΟ: Fat #1 (0-0 bottom-left), ΜΩΒ: Fat #2 (7-7 bottom-right).	54
5.10	Αντιστοιχίες των δεικτών του οριζόντιου άξονα των γραφημάτων του σχήματος 5.9 στα στρώματα των Post-LRF (φάση 3) εκδοχών του SmartWatch μοντέλου. .	55
5.11	Στατιστικές μετρικές των διαφορών μεταξύ των FIQ-χβαντισμένων και των αντίστοιχων μη χβαντισμένων στρωμάτων του μοντέλου BSPC (φάσεις 1 &2), μαζί με τα S και Z . ΠΡΑΣΙΝΟ: Pre-Pruning Lean, ΚΟΚΚΙΝΟ: Pre-Pruning Fat, ΜΠΛΕ: Post-Pruning Lean, ΠΟΡΤΟΚΑΛΙ: Post-Pruning Fat.	56
5.12	Αντιστοιχίες των δεικτών του οριζόντιου άξονα των γραφημάτων του σχήματος 5.11 στα στρώματα των pre-pruning (φάση 1) και post-pruning (φάση 2) εκδοχών του BSPC μοντέλου.	57
5.13	Στατιστικές μετρικές των διαφορών μεταξύ των FIQ-χβαντισμένων και των αντίστοιχων μη χβαντισμένων στρωμάτων του μοντέλου BSPC, μαζί με τα S και Z . ΜΠΛΕ: Lean (BSPC_seq_param100K_pruned_90), ΠΟΡΤΟΚΑΛΙ: Fat (BSPC_seq_param4M_pruned_80).	58
5.14	Αντιστοιχίες των δεικτών του οριζόντιου άξονα των γραφημάτων του σχήματος 5.13 στα στρώματα των post-LRF (φάση 3) εκδοχών του BSPC μοντέλου.	59

5.15 Στατιστικές μετρικές των διαφορών μεταξύ των FIQ-κβαντισμένων και των αντίστοιχων μη κβαντισμένων στρωμάτων του μοντέλου BSPC GAP (φάσεις 1 &2), μαζί με τα S και Z . ΠΡΑΣΙΝΟ: Pre-Pruning Lean , ΚΟΚΚΙΝΟ: Pre-Pruning Fat , ΜΠΛΕ: Post-Pruning Lean , ΠΟΡΤΟΚΑΛΙ: Post-Pruning Fat.. .	60
5.16 Αντιστοιχίες των δεικτών του οριζόντιου άξονα των γραφημάτων του σχήματος 5.11 στα στρώματα των pre-pruning (φάση 1) και post-pruning (φάση 2) εκδοχών του BSPC GAP μοντέλου.	61
5.17 Στατιστικές μετρικές των διαφορών μεταξύ των FIQ-κβαντισμένων και των αντίστοιχων μη κβαντισμένων στρωμάτων του μοντέλου BSPC GAP, μαζί με τα S και Z . ΜΠΛΕ: Lean (BSPC_seq_param100K_pruned_90) , ΠΟΡΤΟΚΑΛΙ: Fat (BSPC_seq_param4M_pruned_80).	62
5.18 Αντιστοιχίες των δεικτών του οριζόντιου άξονα των γραφημάτων του σχήματος 5.13 στα στρώματα των post-LRF (φάση 3) εκδοχών του BSPC GAP μοντέλου.	63

Κατάλογος Πινάκων

4.1 Αρχιτεκτονική του έτοιμου LSTM δικτύου. Το Bidirectional στρώμα είναι wrapper type που εφαρμόζεται πάνω σε LSTM στρώματα, προσδιδοντάς τους επιπλέον λειτουργικότητα. Ο αριθμός N είναι το Batch Size ενώ ο αριθμός len_i είναι το μήκος της εκάστοτε χρονοσειράς εισόδου, \mathbf{x}_i	24
4.2 Αρχιτεκτονική του υλοποιημένου LSTM δικτύου. Αποτελείται από δύο απλά LSTM στρώματα χωρίς τα Bidirectional περιτυλίγματα και διαινέτει λιγότερες παραμέτρους.	25
4.3 Αρχιτεκτονική του μοντέλου Ordonez 2016 Deep Original.	26
4.4 Αρχιτεκτονική του μοντέλου SmartWatch. Αποτελείται από λίγες παραμέτρους αλλά έχει μεγάλο βάθος (19 στρώματα). Σκοπός του είναι η επεξεργασία σημάτων στο πεδίο του χρόνου για την αναγνώριση της δραστηριότητας του χρήστη του.	27
4.5 Αρχιτεκτονική του μοντέλου BSPC. Αποτελείται από πολλές παραμέτρους και έχει μεγάλο βάθος (27 στρώματα). Κανονικά, τα BatchNormalization layers δεν έχουν συνάρτηση ενεργοποίησης αλλά κάθε ένα από αυτά ακολουθείται από ένα <code>tf.keras.layers.ReLU</code> στρώμα.	28
4.6 Αρχιτεκτονική του μοντέλου BSPC GAP. Δεν έχει πολλές παραμέτρους αλλά έχει πολύ μεγάλο βάθος (33 στρώματα). Τα στρώματά του είναι σχεδόν πλήρως διαδοχικά με την εξαίρεση των στρωμάτων τύπου Add που συνδέονται με τα δύο προηγούμενά τους, τα οποία και ανθροίζουν.	30
5.1 Τα αποτελέσματα των δύο αρχικών LSTM μοντέλων.	34
5.2 Τα αποτελέσματα των δύο αρχικών LSTM μοντέλων.	37
5.3 Αριθμός τενσόρων ανά TFLite εκδοχή του μοντέλου.	39
5.4 Τα αποτελέσματα του μερικώς FIQ INT8-συμπιεσμένου Ordonez model.	44
5.5 Όλες οι εξεταζόμενες εκδοχές των συνελικτικών μοντέλων. Τα μοντέλα διαχωρίζονται σε τρεις φάσεις: χωρίς συμπίεση, 1 ^η συμπίεση (pruning) και 2 ^η συμπίεση (pruning + LRD). Στις πρώτες δύο φάσεις, κάθε μοντέλο χωρίζεται σε δύο εκδοχές μια <code>lean</code> και μια <code>fat</code> . Στην τρίτη φάση, λόγω του ότι το LRF οδηγεί σε πολλές λύσεις, μπορεί να έχουμε περισσότερα από ένα <code>lean</code> ή <code>fat</code> μοντέλα.	45

5.6 Αρχιτεκτονική μίας εκ των τριών lean εκδοχών του μοντέλου SmartWatch μετά την εφαρμογή parameter pruning και LRF (3 ^η φάση). Αποτελείται από πολύ λιγότερες παραμέτρους παραμέτρους σε σχέση με το αρχικό μοντέλο (πίνακας 4.4). Έχει ήδη επιτευχθεί $\times 23.85$ συμπίεση χωρίς τη χρήση QA που θα εφαρμοστεί σε λίγο.	46
5.7 Η επίδοση των εκδοχών του μοντέλου SmartWatch (phase 1 &2) σε ακρίβεια πρόβλεψης μετά από PTQ. Σε κάθε περίπτωση παρατίθεται το αρχικό μέγεθος του μη κβαντισμένου μοντέλου (σε KB) και, στο εκάστοτε σχέδιο QA, παρατίθεται ο συντελεστής συμπίεσης.	47
5.8 Η επίδοση των εκδοχών του μοντέλου SmartWatch (phase 1 &2) σε ακρίβεια πρόβλεψης μετά από QAT και PTQ (DC = Don't Care).	47
5.9 Η επίδοση των εκδοχών του μοντέλου BSPC (phase 1 &2) σε δυδαδική ακρίβεια πρόβλεψης μετά από PTQ. Σε κάθε περίπτωση παρατίθεται το αρχικό μέγεθος του μη κβαντισμένου μοντέλου (σε KB) και, στο εκάστοτε σχέδιο QA, παρατίθεται ο συντελεστής συμπίεσης.	48
5.10 Η επίδοση των εκδοχών του μοντέλου BSPC (phase 1 &2) σε δυδαδική ακρίβεια πρόβλεψης μετά από QAT και PTQ (DC = Don't Care).	48
5.11 Η επίδοση των εκδοχών του μοντέλου BSPC GAP (phase 1 &2) σε δυδαδική ακρίβεια πρόβλεψης μετά από PTQ. Σε κάθε περίπτωση παρατίθεται το αρχικό μέγεθος του μη κβαντισμένου μοντέλου (σε KB) και, στο εκάστοτε σχέδιο QA, παρατίθεται ο συντελεστής συμπίεσης.	49
5.12 Η επίδοση των εκδοχών του μοντέλου BSPC GAP (phase 1 &2) σε δυδαδική ακρίβεια πρόβλεψης μετά από QAT και PTQ (DC = Don't Care).	49
5.13 Οι επιδόσεις των τελικών εκδοχών του LRF + pruned SmartWatch μοντέλου (3 ^η φάση), σε ακρίβεια πρόβλεψης. Σε κάθε περίπτωση παρατίθεται το αρχικό μέγεθος του μη κβαντισμένου μοντέλου (σε KB) και, στο εκάστοτε σχέδιο QA, παρατίθεται ο συντελεστής συμπίεσης.	50
5.14 Οι επιδόσεις των τελικών εκδοχών του LRF + pruned BSPC μοντέλου (3 ^η φάση), σε δυδική ακρίβεια πρόβλεψης. Σε κάθε περίπτωση παρατίθεται το αρχικό μέγεθος του μη κβαντισμένου μοντέλου (σε KB) και, στο εκάστοτε σχέδιο QA, παρατίθεται ο συντελεστής συμπίεσης.	50
5.15 Οι επιδόσεις των τελικών εκδοχών του LRF + pruned BSPC GAP μοντέλου (3 ^η φάση), σε δυδική ακρίβεια πρόβλεψης. Σε κάθε περίπτωση παρατίθεται το αρχικό μέγεθος του μη κβαντισμένου μοντέλου (σε KB) και, στο εκάστοτε σχέδιο QA, παρατίθεται ο συντελεστής συμπίεσης.	51

Συντομογραφίες

AI	Artificial Intelligence	Τεχνητή Νοημοσύνη
CNN	Convolutional Neural Network	Συνελικτικό Νευρωνικό Δίκτυο
DL	Deep Learning	Βαθιά Μάθηση
DNN	Deep Neural Network	Βαθύ Νευρωνικό Δίκτυο
DRQ	Dynamic Range Quantization	Κβαντισμός Δυναμικού Εύρους
FIQ	Full-Integer Quantization	Κβαντισμός σε Πλήρως Ακέραιες Παραμέτρους
FNN	Feedforward Neural Network	Βαθύ Νευρωνικό Δίκτυο
GPU	Graphics Processing Unit	Μονάδα Επεξεργασίας Γραφικών
HAR	Human Activity Recognition	Αναγνώριση Ανθρώπινης Δραστηριότητας
IoT	Internet of Things	Διαδίκτυο των Πραγμάτων
LSTM	Long Short-Term Memory	Μακρά Δίκτυα Βραχείας Μνήμης
ML	Machine Learning	Μηχανική Μάθηση
PTQ	Post-Training Quantization	Κβαντισμός μετά την Εκπαίδευση
QA	Quantization	Κβαντισμός
QAT	Quantization Aware Training	Εκπαίδευση με Επίγνωση Κβαντισμού
RDS	Representative Dataset	Αντιρποσωπευτικό Σύνολο Δεδομένων
RNN	Recurrent Neural Network	Αναδρομικό Νευρωνικό Δίκτυο
TF	TensorFlow	-
TFLite	TensorFlow Lite	-

Κεφάλαιο 1

Εισαγωγή

Η ανάλυση χρονοσειρών (Time Series Analysis) αποτελεί αντικείμενο μεγάλου ενδιαφέροντος από τους ερευνητές. Η σπουδαιότητά τους οφείλεται στη συνεισφορά τους στην πρόγνωση (forecasting) μελλοντικών φαινομένων, πράγμα που συνεισφέρει σημαντικά σε διάφορους πρακτικούς τομείς όπως στην επιστήμη [1, 2], στα οικονομικά [3], στην επιχειρηματικότητα [4] και στη μηχανική [5]. Η διαδικασία αυτή, δηλαδή της πρόγνωσης φαινομένων μέσω χρονοσειρών (π.χ. των τιμών μετοχών του σχήματος 1.1), μπορεί να πραγματοποιηθεί με διάφορες μεθόδους, συμπεριλαμβανομένου και μεθόδων Βαθιάς Μάθησης (DL) [6].



Σχήμα 1.1: Τιμές μετοχών ως παράδειγμα δεδομένων χρονοσειρών (πηγή: tradingview.com)

Η Βαθιά Μάθηση αποτελεί ένα επιστημονικό πεδίο μελέτης της Τεχνητής Νοημοσύνης (AI) που έχει παρουσιάσει σημαντικές εξελίξεις στα πρόσφατα χρόνια. Οι εφαρμογές της έχουν αποδειχθεί επαναστατικές σε διάφορους τομείς όπως της υγείας [7], της ενέργειας [8], της έρευνας [9], της οικονομίας [10], της αυτόνομης οδήγησης [11] και όχι μόνο. Συγκεκριμένα, τα βαθιά νευρωνικά δίκτυα (DNNs) παρουσιάζουν σημαντικά υπολογιστικά οφέλη [12], προσφέροντας υψηλότερες επιδόσεις σε σχέση με τα κλασικά μοντέλα Μηχανικής Μάθησης (ML) που στερούνται του βάθους.

Την ίδια στιγμή, αυτή η αύξηση των επιδόσεων επωφελείται εις βάρος της αποδοτικότητας σε μνήμη και σε ταχύτητα εκτέλεσης. Σύμφωνα με το [13], τα DNNs αποτελούνται από εκατομμύριες έως δισεκατομμύριες παραμέτρους που χρειάζεται να αποθηκευθούν στη μνήμη, πραγματοποιούν αντίστοιχο αριθμό πράξεων κατά την επεξεργασία ενός δείγματος δεδομένων και η επιτυχία τους στηρίζεται στη μεγάλη διαθεσιμότητα επεξεργαστικής ισχύος που προσφέρεται κυρίως από κάρτες γραφικών (GPUs). Για την ελάττωση αυτών των απαιτήσεων και των περιορισμών που συνεπάγονται, προτείνονται μέθοδοι συμπίεσης των DNNs.

Η διαδικασία της συμπίεσης ενός DNN έχει πολλούς στόχους. Συγκεκριμένα, πέρα από εξουκονόμηση μνήμης, πρέπει να συνοδεύεται και από μεγαλύτερη ταχύτητα εκτέλεσης καθώς αποφεύγονται περιττοί υπολογισμοί. Στην ταχύτητα εκτέλεσης συνεισφέρει ακόμα περισσότερο και η συμπίεση του μοντέλου σε μια αναπαράσταση που αποτελείται αποκλειστικά και μόνο από ακέραιες παραμέτρους καθώς οι αριθμητικές πράξεις ακεραίων είναι γρηγορότερες απ' ότι οι πράξεις αριθμών κινητής υποδιαστολής (Floating Point - FP). Έτσι προκύπτει και ένα ακόμα, πολύ μεγάλο πλεονέκτημα, δηλαδή η συμβατότητα του DNN με υλισμικό (hardware) που δεν υποστηρίζει αναπαραστάσεις πραγματικών αριθμών [14], δηλαδή δε διαθέτουν Μονάδα Κινητής Υποδιαστολής (Floating Point Unit - FPU). Τέλος, μετά από όλες αυτές τις αλλαγές στις παραμέτρους του μοντέλου, πρέπει οι επιδόσεις του να μην επηρεάζονται σημαντικά ή και καθόλου, πράγμα που δεν είναι εύκολο, ιδίως για τα RNNs [15].



Σχήμα 1.2: (Smartwatch) Μια IoT συσκευή περιορισμένων δυνατοτήτων (πηγή: freepik.com)

Με βάση τα παραπάνω, το τελικό DNN μοντέλο θα μπορεί να φορτωθεί και να εκτελεστεί τοπικά σε κάπιε λογής συσκευή όπως το smartwatch του σχήματος 1.2, χωρίς να περιορίζεται σε κέντρα δεδομένων και υπολογιστικά συστήματα υψηλής ισχύος. Σε αυτές τις συσκευές συμπεριλαμβάνονται οι κινητές συσκευές, οι μικροελεγκτές, τα έξυπνα ρολόγια και γενικότερα τα ενσωματωμένα συστήματα (embedded systems) και οι συσκευές του Διαδικτύου των Πραγμάτων (Internet of Things - IoT devices). Επιπλέον, με αυτό τον τρόπο θα μπορούσε να γίνει εφικτή η χρήση του DNN σε εφαρμογές και συστήματα πραγματικού χρόνου που προηγουμένως η χαμηλή ταχύτητα παρεμβολής (inference time) να την καθιστούσε απαγορευτική.

Στα πλαίσια αυτής της πτυχιακής εργασίας θα μελετηθούν τα διάφορα είδη κβαντισμού σε μοντέλα μετά την εκπαίδευσή τους (Post-Training Quantization - PTQ) καθώς και στρατηγικές για Εκπαίδευση με Επίγνωση Κβαντισμού (Quantization Aware Training - QAT) με τη χρήση της βιβλιοθήκης TensorFlow [16].

Αρχικά, θα κατασκευαστούν απλά DNNs που θα περιλαμβάνουν LSTM στρώματα. Τα μοντέλα αυτά θα παραχθούν με τη χρήση της Keras, που αποτελεί Διασύνδεση Προγραμματισμού Εφαρμογών (Application Programming Interface - API) της βιβλιοθήκης TensorFlow (TF). Μετά την κατασκευή και την εκπαίδευσή τους, θα καταγραφούν χαρακτηριστικά τους όπως, μέγεθος και ακρίβεια πρόβλεψης. Στη συνέχεια, τα DNNs θα υποστούν κβαντισμό με τη χρήση της TensorFlow Lite (ή TFLite), μιας βιβλιοθήκης για την τοποθέτηση και εκτέλεση TF μοντέλων σε κινητές και IoT συσκευές. Τα χαρακτηριστικά των TFLite DNNs θα συγκριθούν με αυτά των αρχικών μοντέλων.

Σε δεύτερη φάση, θα μελετηθεί αναλυτικότερα η διαδικασία της μετατροπής των Keras μοντέλων σε TFLite μοντέλα. Συγκεκριμένα, θα προσδιοριστούν τα βήματα της μετατροπής, οι δομικές και λειτουργικές διαφορές ανάμεσα στα μοντέλα TFLite και στα αρχικά μοντέλα Keras, καθώς και ποια είναι η βέλτιστη μεθοδολογία μετατροπής των μοντέλων, αφού όπως προσδιορίστηκε, διαφορετικές συναρτήσεις μετατροπής της TFLite καταλήγουν σε διαφορετικά αποτελέσματα. Για αυτόν τον σκοπό θα χρησιμοποιηθούν εργαλεία όπως το Netron [17] και ο TFLite Analyzer [18].

Ακόμα, θα αναλυθούν περισσότερο τα Σχέδια Κβαντισμού (Quantization Schemes) που υποστηρίζει η TensorFlow Lite. Θα εξεταστούν αναλυτικά οι δύο μεγάλες κατηγορίες κβαντισμού, δηλαδή ο Κβαντισμός Δυναμικού Εύρους (Dynamic Range Quantization - DRQ) και ο Κβαντισμός σε Πλήρως Ακέραιες Παραμέτρους (Full-Integer Quantization - FIQ), καθώς και άλλες, πιο εξειδικευμένες, κατηγορίες κβαντισμού που προσφέρονται από την TFLite.

Τέλος, θα διερευνηθεί ο λόγος που πέφτουν οι επιδόσεις των μοντέλων χρονοσειρών μετά την κβάντισή τους. Έχει παρατηρηθεί ότι ο κβαντισμός ορισμένων αναδρομικών αρχιτεκτονικών προκαλεί σοβαρή πτώση στην απόδοσή τους, οδηγώντας κάποιους ερευνητές στη χρήση ξεχωριστών τεχνικών για την προσέγγιση αυτής της διαδικασίας [15]. Είναι απαραίτητο να καταπολεμηθεί αυτό το πρόβλημα προκειμένου να προσφέρεται η βέλτιστη συμπίεση των μοντέλων χρονοσειρών διατηρώντας, ταυτοχρόνως, τις επιδόσεις τους. Η έρευνα αυτή θα γίνει με τη χρήση των προαναφερθέντων εργαλείων που προσφέρει η TensorFlow σε συνδυασμό με τον Quantization Debugger [19].

Κεφάλαιο 2

Βασικές Αρχές και Επισκόπηση Βιβλιογραφίας

2.1 Δεδομένα Χρονοσειρών

Μαθηματικό Τπόβαθρο:

Μια χρονοσειρά διακριτού χρόνου αποτελεί ένα ακολουθιακό σύνολο δεδομένων, που μετρώνται συνήθως σε διαδοχικές χρονικές στιγμές [20], δηλαδή:

$$\mathbf{x}(t), \quad t = 0, 1, 2, \dots$$

όπου κάθε μεταβλητή $\mathbf{x}(t)$ αντιμετωπίζεται ως μια τυχαία μεταβλητή. Με ελαφρώς διαφορετικό τρόπο, μια χρονοσειρά ορίζεται στο [21] ως το σύνολο διανυσμάτων:

$$\mathbf{X} = \{\mathbf{x}(1), \mathbf{x}(2), \dots, \mathbf{x}(n)\}, \quad \mathbf{x}(t) \in \mathbb{R}^m \quad (2.1)$$

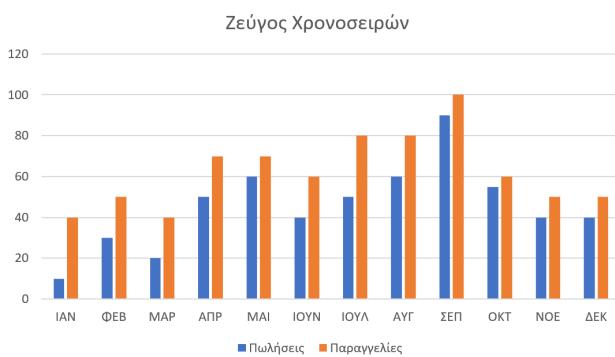
$$\text{δηλ., } \mathbf{x}(t) = (x_1(t), x_2(t), \dots, x_m(t))^T$$

Γενικά, θεωρείται δεδομένο ότι οι μετρήσεις $\mathbf{x}(t-1)$, $\mathbf{x}(t)$ και $\mathbf{x}(t+1)$ απέχουν ίσα χρονικά διαστήματα μεταξύ τους, για κάθε t . Δηλαδή, μια καινούργια μέτρηση γίνεται ανά σταθερό χρονικό διάστημα, Δt . Όταν αυτό δεν ισχύει, η χρονοσειρά θεωρείται άνισα κατανεμημένη (unevenly spaced time series), περίπτωση στην οποία το εκάστοτε πρόβλημα πρέπει να προσεγγιστεί με ιδιαίτερη προσοχή [22].

Κύριος σκοπός της χρονοσειράς \mathbf{X} είναι να χρησιμοποιηθούν όλες οι προηγούμενες μετρήσεις $\mathbf{x}(t)$ ώστε να πραγματοποιηθεί μια εμπειρική προσέγγιση για μια μελλοντική κατάσταση, π.χ. για την επόμενη μέτρηση, $\hat{\mathbf{x}}(n+1)$. Η διαδικασία αυτή ονομάζεται πρόγνωση (forecasting) και είναι διαφορετική από τη διαδικασία της πρόβλεψης (prediction), όπου οι τρέχουσες μετρήσεις της χρονοσειράς $\mathbf{x}(t)$ χρησιμοποιούνται για τον προσεγγιστικό προσδιορισμό μιας μεταβλητής, \hat{y} , που δεν είναι ανάγκη να τοποθετείται σε κάποιο σημείο στο χρόνο. Οι χρονοσειρές μπορούν να χρησιμοποιηθούν και για τους δύο σκοπούς.



(α') Γράφημα γραμμών



(β') Ραβδόγραμμα

Σχήμα 2.1: Διαφορετικές αναπαραστάσεις μονοδιάστατων χρονοσειρών

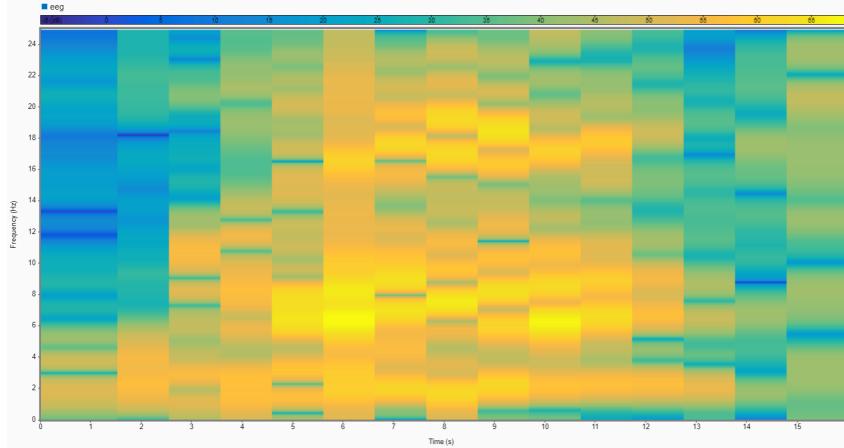
Οπτικοποίηση:

Τα δεδομένα χρονοσειρών μπορούν να αναπαρασταθούν με διάφορους τρόπους, όπως φαίνεται στο σχήμα 2.1. Τις πιο ευρέως χρησιμοποιημένες αναπαραστάσεις για μονομεταβλητές (univariate) μετρήσεις, $x(t) \in \mathbb{R}$, αποτελούν το γράφημα γραμμών (σχήμα 2.1α') και το ραβδόγραμμα (σχήμα 2.1β'), που αντιστοιχίζουν την τιμή της μέτρησης $x(t)$ στην εκάστοτη χρονική στιγμή t . Από την άλλη, για πολυμεταβλητές (multivariate) μετρήσεις, $\mathbf{x}(t) \in \mathbb{R}^m$ με $m \neq 1$, ένας πιο αποδοτικός τρόπος απεικόνισης της χρονοσειράς θα ήταν με ένα θερμικό χάρτη (heatmap), αντιστοιχίζοντας όλες τις μετρήσεις $x_i(t)$ στο εκάστοτε t και την τιμή τους να υποδηλώνεται από την ένταση του χρώματος των κελιών τους. Για παράδειγμα, στο σχήμα 2.2 απεικονίζεται η χρονοσειρά:

$$\mathbf{F} = \{\mathbf{f}(1), \mathbf{f}(2), \dots, \mathbf{f}(t), \dots, \mathbf{f}(16)\}, \text{ με } t \text{ σε sec}$$

όπου κάθε $\mathbf{f}(t)$ είναι το διάνυσμα της ισχύος των συχνοτήτων σε κάθε χρονική στιγμή t . Η

ισχύς της συχνότητας $f_i(t)$ απεικονίζεται ως μπλε αν είναι χαμηλή (-5 dB) και με κίτρινο αν είναι υψηλή ($> 65 \text{ dB}$).



Σχήμα 2.2: (Heatmap) Αναπαράσταση πολυυδιάστατων χρονοσειρών

Εφαρμογές: Οι χρονοσειρές βρίσκουν εφαρμογή σε πληθύρα επιστημονικών και επιχειρηματικών πεδίων. Για παράδειγμα στο [5], οι ερευνητές προτείνουν τη χρήση χρονοσειρών έναντι του γρήγορου μετασχηματισμού Fourier (FFT), για τη μελέτη των ιδιοτήτων κάποιων κατασκευών, όπως μιας γέφυρας, υπό κάποιες συνθήκες. Επίσης, στο [4] αναλύονται χρονολογικά δεδομένα σχετικά με τις αφίξεις τουριστών από κάποιες ασιατικές χώρες στην Αυστραλία το χρονικό διάστημα 1975-1989, αποδυκνείοντας τη συνεισφορά των δεδομένων χρονοσειρών στον τομέα του τουρισμού και του εμορίου. Μια ακόμα αξιοσημείωτη εφαρμογή των χρονοσειρών αποτελεί η Επεξεργασία Φυσικής Γλώσσας (Natural Language Processing - NLP), όπου όλες οι προηγούμενες (κωδικοποιημένες) λέξεις $w(k) \in \mathbb{Z}$ ενός κειμένου εισόδου, \mathbf{W} , θα χρησιμοποιηθούν, αναδρομικά, για τον προσδιορισμό κάθε επόμενης λέξης $w(n+1)$:

$$\mathbf{W} = \{w(1), w(2), \dots, w(n)\}$$

Σε αυτή τη λογική στηρίζονται πολλές από τις σύγχρονες DNN αρχιτεκτονικές για NLP [23]. Το $w(k)$ αναπαριστά την k -οστή λέξη του κειμένου. Αυτό έχει ιδιαίτερη σημασία για την ανάλυση φυσικής γλώσσας καθώς δεν έχουν σημασία μόνο οι λέξεις για την ερμηνεία μιας πρότασης, αλλά και η σειρά με την οποία εμφανίζονται σε αυτήν. Τέλος, ένα παράδειγμα επεξεργασίας χρονοσειρών για πρόβλεψη, αντί πρόγνωσης, αποτελεί η βάση δεδομένων από κριτικές ταινιών που προσφέρει η διεπαφή Keras για ταξινόμηση συναισθήματος (sentiment classification) [24]. Αυτό το σύνολο δεδομένων αποτελείται από ζεύγη (\mathbf{X}, y) , όπου το X είναι μια μονοδιάστατη χρονοσειρά, της οποίας το εκάστοτε $x(k) \in \mathbb{Z}$ αναπαριστά λέξη και το k υποδηλώνει

ότι είναι η k -οστή λέξη της κριτικής. Με αυτόν τον τρόπο μπορεί να πραγματοποιηθεί μια πρόβλεψη για το αν η κριτική είναι αρνητική ($y = 0$) ή θετική ($y = 1$).

2.2 Τεχνικές Επεξεργασίας Χρονοσειρών

Στην επιστημονική βιβλιογραφία όχουν προταθεί διάφορα μοντέλα για την επεξεργασία δεδομένων χρονοσειρών. Ένα μοντέλο χρονοσειρών μπορεί να είναι είτε στοχαστικό όπου τα στοιχεία της χρονοσειράς αναπαριστούν τυχαίες μεταβλητές, είτε υπολογιστικό, όπου τα $\mathbf{x}(t)$ αναπαριστούν διανύσματα [20].

Στοχαστικά μοντέλα αποτελούν το Αυτοπαλινδρομικό Μοντέλο (Autoregressive - AR), το Μοντέλο Κινούμενου Μέσου (Moving Average - MA) και διάφοροι συνδυασμοί αυτών των δύο, όπως τα Autoregressive Moving Average (ARMA), Autoregressive Integrated Moving Average (ARIMA), Seasonal Autoregressive Integrated Moving Average (SARIMA), κλπ [25]. Σε αυτή την περίπτωση, τα στοιχεία των χρονοσειρών είναι τυχαίες μεταβλητές, που ενδεχομένως να επιβάλλεται να ακολουθούν κάποια συγκεκριμένη κατανομή [26], π.χ. την κανονική κατανομή.

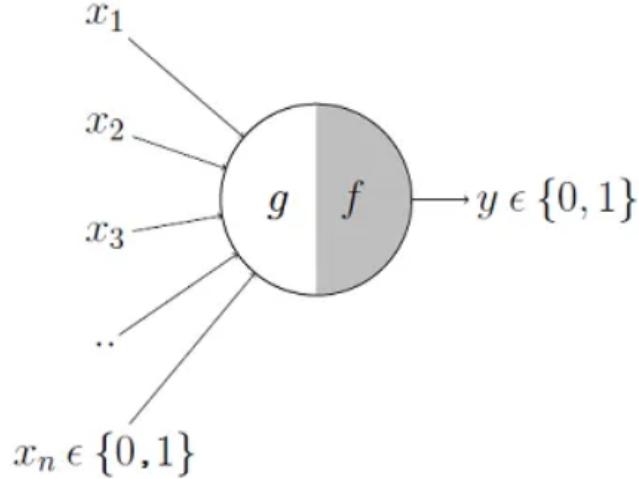
Υπολογιστικά μοντέλα επεξεργασίας χρονοσειρών αποτελούν οι DL αρχιτεκτονικές [6, 20] και οι Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines - SVMs) [3, 20]. Συγκεκριμένα, το υποσύνολο των DL αρχιτεκτονικών που ενδύκνειται για την επεξεργασία χρονοσειρών και θα αποτελέσει κύριο αντικείμενο αυτής της πτυχιακής αποτελούν οι RNN αρχιτεκτονικές, όπως τα Μακρά Δίκτυα Βραχείας Μνήμης, ή LSTMs. Σε αυτή την περίπτωση, τα στοιχεία των χρονοσειρών αποτελούν διανύσματα. Οι υπολογιστικές μέθοδοι παρουσιάζουν σημαντικά πλεονεκτήματα έναντι των στοχαστικών μεθόδων, καθώς τα νευρωνικά δίκτυα καθιδηγούνται από τα δεδομένα (data-driven) και άρα δεν υπάρχει ανάγκη περαιτέρω περίπλοκης μοντελοποίησης. Για παράδειγμα, ο περιορισμός ενός στοχαστικού μοντέλου ότι η εξεταζόμενη χρονοσειρά πρέπει να ακολουθεί μια συγκεκριμένη κατανομή δεν έχει ισχύ στα νευρωνικά δίκτυα, καθώς αυτό είναι κάτι που το DNN θα μάθει από μόνο του μέσω της εκπαίδευσης.

2.3 Τεχνητά Νευρωνικά Δίκτυα

Ιστορικό Τπόβαθρο:

Τα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks - ANNs) είναι υπολογιστικά μοντέλα που χρησιμοποιούνται για την επίλυση προβλημάτων και λήψη αποφάσεων, π.χ. παλινδρόμηση (regression) και ταξινόμηση (classification). Αποτελούν τον κεντρικό πυλώνα της

Βαθιας Μάθησης και είναι έργο εμπνευσμένο από το μαθηματικό μοντέλο της λειτουργίας των βιολογικών νευρώνων που προτάθηκε από τους ερευνητές Warren Sturgis McCulloch και Walter Pitts το 1943 [27].



Σχήμα 2.3: Το μαθηματικό πρότυπο του νευρώνα σύμφωνα με τους McCulloch και Pitts (πηγή: medium.com).

Σύμφωνα με αυτούς, ένας νευρώνας δέχεται n δυαδικά σήματα εισόδου, $\mathbf{x} = (x_1, x_2, \dots, x_n)^T$, όπως φαίνεται και στο σχήμα 2.3. Η έξοδος του νευρώνα προσδιορίζεται από την τιμή κατωφλίου του, θ , δηλαδή:

$$y = f(v) = \begin{cases} 0, & \text{αν } v < \theta \\ 1, & \text{αν } v \geq \theta \end{cases}, \quad v = \sum_{i=1}^n x_i \quad (2.2)$$

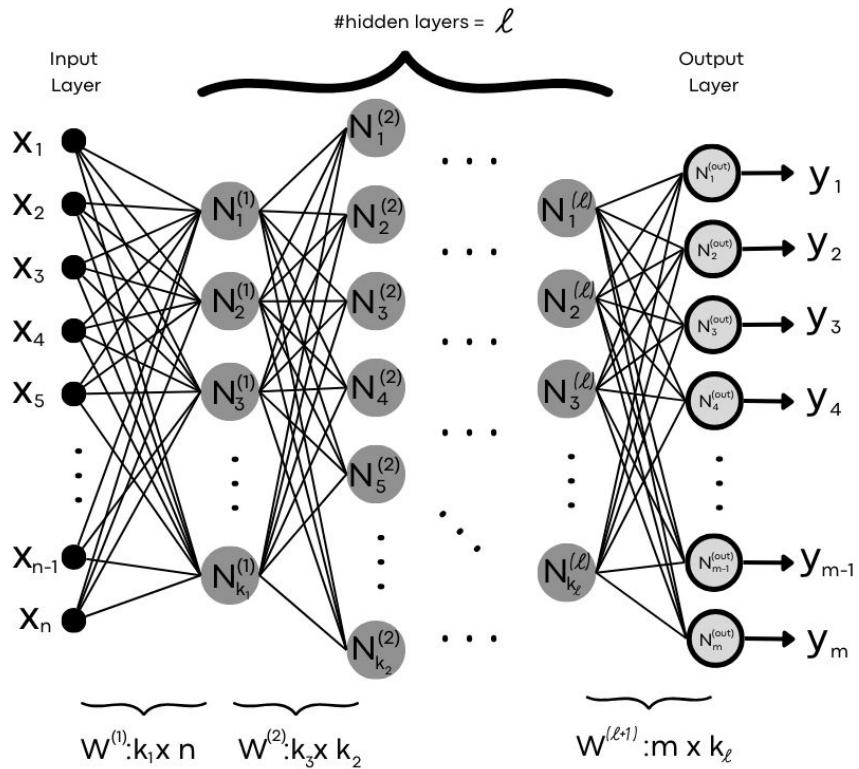
Όπου η $f : \mathbb{R} \rightarrow \mathbb{R}$ αποτελεί τη συνάρτηση ενεργοποίησης του νευρώνα.

Οι Εξελίξεις που Ακολούθησαν:

Εξέλιξη αυτής της ιδέας αποτελεί ο νευρώνας Perceptron [28] όπου τα x_i είναι πραγματικοί αριθμοί και σταθμίζονται από τα «συναπτικά βάρη» (synaptic weights), $\mathbf{w} = (w_1, w_2, \dots, w_n)^T$, με $w_i \in \mathbb{R}$, και στο άθροισμα προστίθεται μία ακόμα παράμετρος, $b \in \mathbb{R}$, που λέγεται «μεροληψία» (bias):

$$y = f(v) = \begin{cases} 0, & \text{αν } v < \theta \\ 1, & \text{αν } v \geq \theta \end{cases}, \quad v = \mathbf{w}^T \mathbf{x} + b \quad (2.3)$$

Πάνω σε αυτή την ιδέα στηρίζεται η DNN αρχιτεκτονική του Πολυστρωματικού Perceptron (Multi-layer Perceptron - MLP) που φαίνεται στο σχήμα 2.4, όπου τα σήματα εισόδου επεξεργάζονται από πολλούς ενδιάμεσους (χρυφούς) νευρώνες (χρυφά στρώματα νευρώνων) μέχρι να παραχθεί η τελική έξοδος. Επίσης, η συνάρτηση ενεργοποίησης είναι συνήθως μη γραμμική και κάθισε νευρώνας του στρώματος l συνδέεται με όλους τους νευρώνες των στρώμάτων $l-1$ και $l+1$.



Σχήμα 2.4: Η αρχιτεκτονική του Πολυστρωματικού Perceptron.

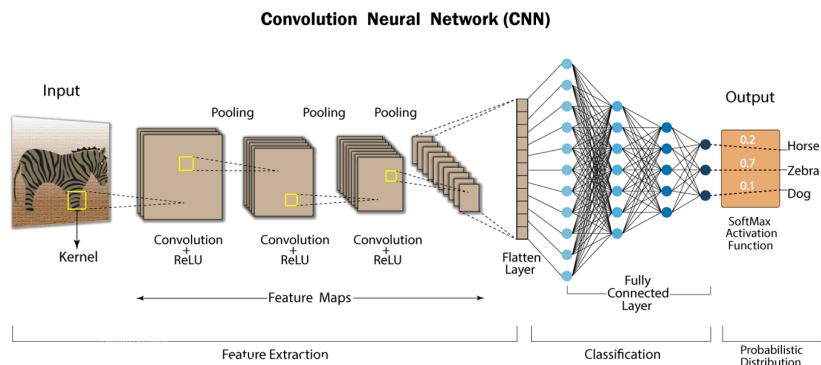
Πρέπει να σημειωθεί ότι από εδώ προέρχεται η επιστημονική ορολογία «Βαθιά Μάθηση». Τα διαδοχικά στρώματα νευρώνων που βρίσκονται μεταξύ της εισόδου και της εξόδου είναι αυτά που δίνουν «βάθος» στην αρχιτεκτονική. Το βάθος είναι αυτό που προσφέρει στα μοντέλα τη δυνατότητα να προσδιορίζουν περίπλοκες, μη γραμμικές σχέσεις μεταξύ των δεδομένων εισόδου. Συνήθως, η αύξηση του βάθους σε συνδυασμό με την αύξηση των χρυφών νευρώνων συνεπάγεται την αύξηση των επιδόσεων και της αξιοπιστίας του μοντέλου.

Εκπαίδευση Νευρωνικών Δικτύων:

Τα ANNs ανήκουν στην κατηγορία της μάθησης με επίβλεψη (supervised learning), κατά την οποία παρέχονται τα δείγματα εισόδου, \mathbf{X} , μαζί με τις αντίστοιχες επιθυμητές εξόδους, \mathbf{y} . Η εκπαίδευση τους πραγματοποιείται με τον αλγόριθμο Οπισθοδιάδοσης (Back Propagation - BP). Σύμφωνα με τον BP, για κάθε δείγμα $\mathbf{x}_i \in \mathbf{X}$, υπολογίζεται μια πρόβλεψη, $\hat{\mathbf{y}}_i$. Έπειτα, μέσω μιας δοσμένης συνάρτησης σφάλματος υπολογίζονται οι αποστάσεις των $\hat{\mathbf{y}}_i$ από τα αντίστοιχα $\mathbf{y}_i \in \mathbf{y}$ και το σφάλμα χρησιμοποιείται για την εκ νέου ανάθεση καινούργιων συναπτικών βαρών που ελαχιστοποιούν όλο και περισσότερο το σφάλμα. Αυτή η διαδικασία επαναλαμβάνεται συνεχώς μέχρι το μοντέλο να επιτύχει επιθυμητές επιδόσεις σε άγνωστα δεδομένα.

2.4 Συνελικτικά Νευρωνικά Δίκτυα

Πέρα από την αρχιτεκτονική MLP, μια άλλη, πολύ δημοφιλή κατηγορία νευρωνικών δικτύων αποτελούν τα Συνελικτικά Νευρωνικά Δίκτυα (Convolutional Neural Networks - CNNs) που προτάθηκαν από τον Yann LeCun [29] και διακρίνονται για τις εκπληκτικές τους ικανότητες στην επεξεργασία εικόνων. Συνήθως, τα CNNs αποτελούνται από συνελικτικά στρώματα (convolutional layers) και στρώματα υποδειγματοληψίας (pooling layers) που εναλλάσσονται.

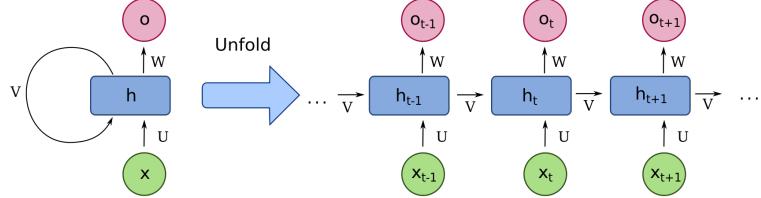


Σχήμα 2.5: CNN για την εξαγωγή χαρακτηριστικών από εικόνες (πηγή: medium.com)

Στο σχήμα 2.5 φαίνεται η αρχιτεκτονική ενός CNN που εξάγει χαρακτηριστικά από εικόνες μέσω μιας συνελικτικής αρχιτεκτονικής και στη συνέχεια τροφοδοτούνται σε ένα δίκτυο αρχιτεκτονικής MLP. Πέρα από τις εφαρμογές στους στην επεξεργασία εικόνων, τα CNNs χρησιμοποιούνται ευρέως και για την επεξεργασία δεδομένων άλλης φύσεως, όπως η ομιλία και οι χρονοσειρές [30].

2.5 Αναδρομικά Νευρωνικά Δίκτυα και Μακρά Δίκτυα Βραχείας Μνήμης

Τα Αναδρομικά Νευρωνικά Δίκτυα (RNNs), αποτελούν μία από τις δύο μεγάλες οικογένειες των τεχνητών νευρωνικών δικτύων μαζί με τα Νευρωνικά Δίκτυα Εμπρόσθιας Τροφοδότησης (FNNs). Το χαρακτηριστικό που διαχωρίζει αυτές τις δύο είναι ότι οι αναδρομικές αρχιτεκτονικές συμπεριλαμβάνουν βρόγχους ανατροφοδότησης (feedback loops). Με άλλα λόγια, διαθέτουν νευρώνες που, μαζί με τις τρέχουσες τιμές με τις οποίες τροφοδοτούνται τη χρονική στιγμή t , τροφοδοτούνται και με τιμές από την προηγούμενη κατάσταση, της χρονικής στιγμής $t - 1$.

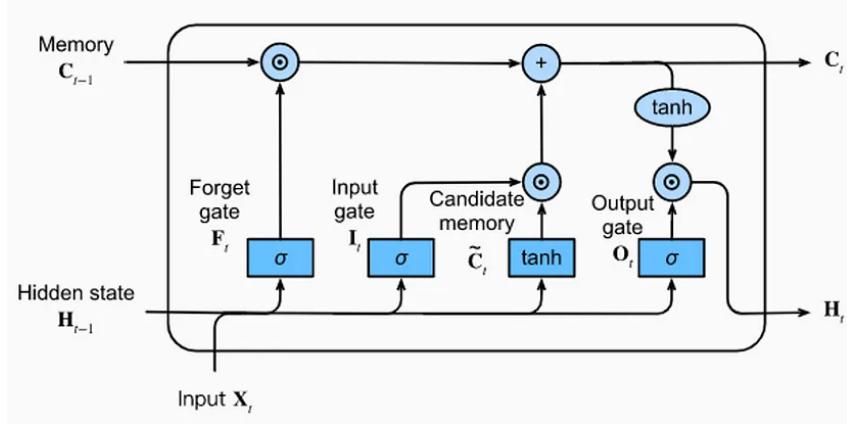


Σχήμα 2.6: Οπτικές αναπαραστάσεις ενός RNN νευρώνα (πηγή: wikipedia.org).

Σχηματικά, ένας αναδρομικός νευρώνας μπορεί να αναπαρασταθεί συμπυκνωμένα με αυτόν τον αναδρομικό βρόγχο, ή να ξεδιπλωθεί στο χρόνο (loop unrolling) όπως φαίνεται στο σχήμα 2.6. Με αυτόν τον τρόπο, το μοντέλο μπορεί να πραγματοποιεί προβλέψεις και να εκπαιδεύεται πάνω σε ακολουθιακά δεδομένα.

Ένα σοβαρό μειονέκτημα που έχουν οι κλασικές RNN αρχιτεκτονικές είναι ότι οι τιμές των κλίσεων των παραμέτρων κατά το βήμα της βελτιστοποίησης μεγεθύνονται ή συρρικνώνονται απαγορευτικά μετά από την επεξεργασία αρκετών διαδοχικών μετρήσεων της χρονοσειράς εισόδου (exploding/vanishing gradients problem). Για αυτόν τον σκοπό τα κλασικά RNNs επεκτάθηκαν και εξελίχθηκαν στην αρχιτεκτονική των Μακρών Δικτύων Βραχείας Μνήμης, ή αλλιώς LSTMs. Τα LSTMs καταπολέμησαν τις προαναφερθέντες δυσκολίες, διατηρώντας ταυτόχρονα το στοιχείο της μνήμης κατά την επεξεργασία χρονοσειρών εισόδου. Στο σχήμα 2.7 παρουσιάζεται σχηματικά η αρχιτεκτονική τους.

Στα κεφάλαια 4 και 5 υπάρχουν αρκετές αρχιτεκτονικές για την επεξεργασία χρονοσειρών, όπως CNN, LSTM, GRU, και άλλες. Θεματικό άξονα της εργασίας αποτελεί η καταπολέμηση



Σχήμα 2.7: Η αρχιτεκτονική ενός κυττάρου LSTM (πηγή: medium.com).

των προβλημάτων που επηρεάζει αυτές, καθώς και όλες τις άλλες ANN αρχιτεκτονικές, που αναλύεται στην ακόλουθη υποενότητα.

2.6 Προβλήματα των Νευρωνικών Δικτύων

Όπως αναφέρθηκε και στο πρώτο κεφάλαιο, τα νευρωνικά δίκτυα που χρησιμοποιούνται σε σύγχρονες εφαρμογές για την επίλυση προβλημάτων και λήψη αποφάσεων, έχουν υπερβολικά μεγάλες απαιτήσεις μνήμης και υπολογιστικής ισχύος. Παρά την εκπληκτική πρόοδο που έχουν επιτύχει οι τεχνολογίες υλισμικού και τις τεράστιες ποσότητες πτητικής και μη πτητικής μνήμης που αυτές μας προσφέρουν σε πολύ χαμηλότερο κόστος απ' ότι παλιότερα, μπορεί αυτές να μην είναι αρκετές για την αποθήκευση και την εκτέλεση μεγάλων νευρωνικών δικτύων. Ενδεικτικά, η μικρότερη εκδοχή (7 δισ. παράμετροι) του Μεγάλου Γλωσσικού Μοντέλου (Large Language Model - LLM) της Meta Inc., γνωστό κι ως LLaMa 2, έχει μεγάλες απαιτήσεις μνήμης ακόμα και για έναν αξιόλογο προσωπικό υπολογιστή [31]. Ακόμα και μικρότερα μοντέλα με απλούστερες αρχιτεκτονικές μπορεί να έχουν απαγορευτικό μέγεθος για την τοποθέτηση και εκτέλεσή τους σε ενσωματωμένα συστήματα και συσκευές αιχμής. Τέλος, ακόμα κι αν ένα μοντέλο έχει αρκετά μικρό μέγεθος δε θα μπορούσε να τοποθετηθεί και να εκτελεστεί τοπικά σε οποιοδήποτε υπολογιστικό σύστημα καθώς πολλές μονάδες επεξεργασίας (processing units), όπως οι ARM-Cortex M0/M0+/M1/M3, δεν υποστηρίζουν πραγματικούς αριθμούς κινητής υποδιαστολής [32], που αποτελούν βασικό στοιχείο της μαθηματικής λογικής των νευρωνικών δικτύων. Στην επίλυση ή στη μετρίαση των προαναφερθέντων προβλημάτων συνεισφέρει η συμπίεση των νευρωνικών δικτύων. Το αντικείμενο της συμπίεσης θα εξεταστεί εις βάθος, θεωρητικά και πειραματικά, σε όλα τα επόμενα κεφάλαια.

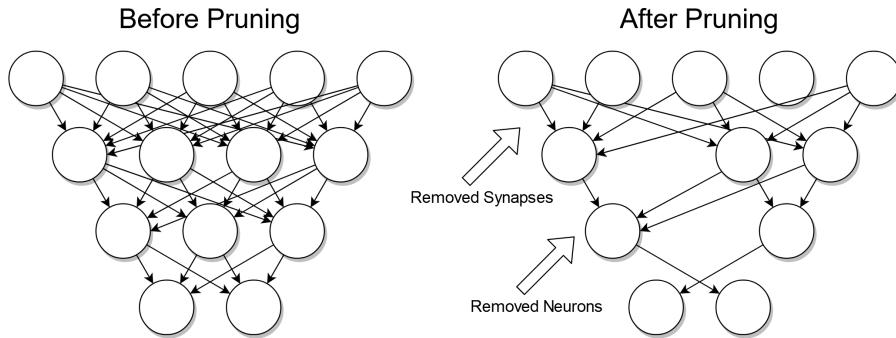
Κεφάλαιο 3

Συμπίεση των Νευρωνικών Δικτύων

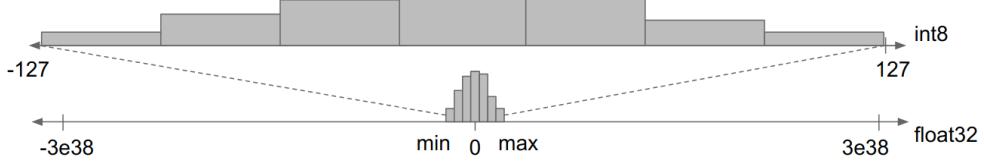
3.1 Μέθοδοι Συμπίεσης και Στόχοι τους

Η διαδικασία της συμπίεσης ενός DNN μοντέλου έχει δύο σκοπούς, που είναι η ελάττωση του μεγέθους του σε μνήμη και η ελάττωση του χρόνου αναμονής κατά την παρεμβολή δεδομένων (inference). Αρχικά, η ανάγκη χωρικής συρρίκνωσης ενός νευρωνικού δικτύου προκύπτει καθώς δεν υπάρχει κάποιος ντετερμινιστικός τρόπος να προσδιοριστεί το ιδανικό μέγεθος ενός μοντέλου. Συνήθως, οι προγραμματιστές λογισμικού προσδιορίζουν το μέγεθος ενός μοντέλου εμπειρικά, αυξάνοντας το μέγεθος νευρώνων και των ενδιάμεσων στρωμάτων μέχρι να παρατηρηθεί ικανοποιητική αύξηση των επιδόσεων, όμως δεν υπάρχει κάποια εγγύηση ότι όλες οι εισαχθείσες παράμετροι είναι απολύτως απαραίτητες. Σε αυτή την περίπτωση θα ήταν πολύ χρήσιμος ένας αλγόριθμος ιχνηλάτησης και αφάρεσης πλεοναστικών συνάψεων ή και πλεοναστικών νευρώνων. Αυτή η μέθοδος συμπίεσης (σχήμα 3.1α') είναι γνωστή ως Κλάδεμα Παραμέτρων (Parameter Pruning). Επιπλέον, σε ένα δοσμένο μοντέλο, είναι αμφισβητούμενο το αν οι τύποι δεδομένων των νευρώνων είναι αποδοτικοί σε μνήμη. Αυτό ακριβώς αντιμετωπίζει η -ευρέως χρησιμοποιούμενη- μέθοδος του Κβατνισμού (Quantization - QA), που μετατρέπει κάποιες παραμέτρους ενός DNN από float32 σε int8 (σχήμα 3.1β'), διατηρώντας όσο το δυνατόν περισσότερο τις αρχικές επιδόσεις.

Οι παραπάνω μέθοδοι έχουν ως συνέπεια την αύξηση της ταχύτητας κατά την παρεμβολή δεδομένων, δηλαδή ελαττώνεται ο χρόνος αναμονής από τη στιγμή που θα λάβει το DNN ένα ερώτημα έως τη στιγμή που θα υπολογίσει την αντίστοιχη απάντηση. Συγκεκριμένα, η ελάττωση του αριθμού των παραμέτρων λιγοστεύει τους περιττούς υπολογισμούς, ενώ η βελτιστοποίηση των παραμέτρων από FP σε ακεραίους ωφελεί το DNN υπολογιστικά καθώς οι πράξεις μεταξύ ακεραίων είναι ταχύτερες από τις πράξεις μεταξύ πραγματικών αριθμών. Κάποιες ακόμα τεχνικές συμπίεσης είναι η Παραγοντοποίηση Χαμηλής Βαθμίδας (Low-Rank Factorization - LRF) και η Μετάγγιση Γνώσης (Knowledge Distillation - KD).



(α') Ένα DNN πριν και μετά το κλάδεμα των παραμέτρων του. (πηγή: wikipedia.org)



(β') Κβαντισμός των παραμέτρων ενός DNN από float32 σε int8. (πηγή: tensorflow.org)

Σχήμα 3.1: Οπτικοποίηση διαφορετικών μεθόδων συμπίεσης.

3.2 Ενδεικτικές Εφαρμογές Συμπίεσης Μοντέλων

Η μέθοδος συμπίεσης επιλέγεται με βάση τις ανάγκες του χρήστη και την πλατφόρμα στην οποία θα τοποθετηθεί το DNN. Για παράδειγμα, σε περίπτωση που οι περιορισμοί σε χώρο δεν είναι οξείς και διατίθεται κάρτα γραφικών ή μια συσκευή Apple, θα μπορούσε να χρησιμοποιηθεί QA από float32 σε float16 (ή bfloat16). Αυτό οφείλεται στην αυξημένη υποστήριξη που παρέχεται για τους πραγματικούς αριθμούς μισής ακρίβειας (half-precision FP) από τις GPU της NVIDIA [33] και από τη Neural Engine της Apple Inc. [34]. Αντιθέτως, αν ένα DNN προορίζεται να τοποθετηθεί σε microcontrollers, microprocessors ή edge devices όπου η διαθέσιμη μνήμη είναι ελάχιστη και μπορεί να μη διαθέτουν FPU, τότε ο Κβαντισμός σε Πλήρως Ακέραιες Παραμέτρους (FIQ) αποτελεί τη βέλτιστη -αν όχι μοναδική- επιλογή. Τέλος, μπορεί να χρησιμοποιηθεί και συνδυασμός μεθόδων συμπίεσης, στρατηγική που χρησιμοποιούν οι

ερευνητές στο [35], όπου συμπιέζουν τα δίκτυα LeNet, AlexNet και VGG χρησιμοποιώντας διαδοχικά τις μεθόδους κλαδέματος, κβαντισμού και κωδικοποίησης Huffman.

3.3 Στρατηγικές Συμπίεσης με Κβαντισμό

Η μέθοδος του Κβαντισμού, που αποτελεί το κύριο αντικείμενο έρευνας αυτής της εργασίας, μπορεί να προσεγγιστεί με δύο διαφορετικές στρατηγικές. Η κύρια εκ των δύο είναι ο Κβαντισμός Μετά την Εκπαίδευση, ή Post-Training Quantization (PTQ) όπως αποκαλείται στη διεθνή επιστημονική βιβλιογραφία. Η στρατηγική αυτή αντιστοιχίζει κάποιες float32 παραμέτρους ενός ήδη εκπαίδευμένου DNN σε ακέραιες παραμέτρους ή σε παραμέτρους σταθερής υποδιαστολής, χωρίς να απαιτείται η εκ νέου εκπαίδευσή του, [36]. Γενικά, το ισχυρό μαθηματικό υπόβαθρο αυτής της στρατηγικής επιτρέπει στο καινούργιο μοντέλο να διατηρεί σε ικανοποιητικό βαθμό τις επιδόσεις του αρχικού μοντέλου. Δυστυχώς, όμως, αυτό μπορεί να μην ισχύει σε κάποιες περιπτώσεις, καθώς μπορεί να παρατηρηθεί δραστική πτώση στην ακρίβεια ενός δοσμένου μοντέλου που έχει υποστεί PTQ. Για την επίλυση αυτού του προβλήματος, εισάγεται η στρατηγική της Εκπαίδευσης με Επίγνωση Κβαντισμού, ή Quantization Aware Training (QAT). Η QAT δεν αποτελεί εναλλακτική στρατηγική της PTQ, αλλά συμπληρωματική της, καθώς η πρώτη αποτελεί μια διαδικασία τύπου fine-tuning που χρησιμοποιείται πριν την εφαρμογή της δεύτερης.

3.4 Κβαντισμός Μετά την Εκπαίδευση (PTQ)

Η στρατηγική PTQ αποτελεί τη βασική ιδέα της μεθόδου συμπίεσης του Κβαντισμού. Ο Κβαντισμός μετά την Εκπαίδευση υλοποιείται ως μια διαδικασία που δέχεται ένα DNN στην είσοδό της και παράγει το συμπιεσμένο, ισοδύναμο δίκτυο στην έξοδό της, χωρίς να χρειαστεί να ληφθούν υπόψη επιπλέον παράγοντες όπως fine-tuning ή δείγματα δεδομένων παρεμβολής. Αυτό τον καθιστά ιδιαίτερα δημοφιλή καθώς προσφέρει μεγάλη ευκολία χρήσης. Δηλαδή ο ποιοσδήποτε μπορεί να συμπιέσει όμεσα οποιοδήποτε DNN διαθέτει χρησιμοποιώντας αυτή τη διαδικασία, η οποία και αποτελεί μαύρο κουτί για πολλούς ενδιαφερόμενους καθώς δεν προϋποθέτει εκτεταμένη γνώση του επιστημονικού υποβάθρου της [36]. Παρ' όλα αυτά είναι σημαντικό να παρουσιαστεί το μαθηματικό υπόβαθρο του Κβαντισμού για λόγους πληρότητας.

Ο Ομοιόμορφος Αφινικός Κβαντισμός (Uniform Affine Quantization) ή απλά Μη Συμμετρικός Κβαντισμός (Assymmetric Quantization) ορίζεται από ένα ζεύγος τριών παραμέτρων. Οι παράμετροι αυτοί είναι ο συντελεστής κλιμάκωσης (scale), S , το μηδενικό σημείο (zero-point), Z και το τελικό πλήθος ψηφίων (bit width), b . Δοσμένων αυτών, ένας τένσορας εισόδου \mathbf{X}_{real} με βαθμωτούς στο \mathbb{R} αντιστοιχίζεται σε έναν τένσορα ακεραίων τιμών \mathbf{X}_{int} με βαθμωτούς στο

διάστημα $[0, 2^b - 1]$ (μη προσημασμένη περίπτωση) ή στο $[-2^{b-1}, 2^{b-1} - 1]$ (προσημασμένη περίπτωση). Η εν λόγω αντιστοίχιση δίνεται από την παρακάτω σχέση [14, 37]:

$$\mathbf{X}_{\text{int}} = \frac{\mathbf{X}_{\text{real}}}{S} + Z \quad (3.1)$$

Στα νευρωνικά δίκτυα, κάθε τένσορας \mathbf{X} (στρώμα) έχει τις δικές του παραμέτρους (S, Z) .

Συντελεστής Κλιμάκωσης: Η κλιμάκωση S έχει τον ίδιο τύπο δεδομένων με τα μηχαντισμένα δεδομένα ($S \in \mathbb{R}$) και προκύπτει από το εύρος του τελικού εύρους τιμών καθώς και τη μέγιστη και ελάχιστη τιμή του τένσορα [38]:

$$S = \frac{2^b - 1}{\max(\mathbf{X}) - \min(\mathbf{X})} \quad (3.2)$$

Στα [14] και [37] εξετάζονται και αναπαραστάσεις σταθερής υποδιαστολής του S για την αποθήκευσή του σε συστήματα που δε διαθέτουν FPU.

Συντελεστής Μηδενικού Σημείου: Το μηδενικό σημείο Z έχει τον ίδιο τύπο δεδομένων με τις κβαντισμένες τιμές και σκοπός του είναι να αναπραστήσει την κβαντισμένη τιμή που αντιστοιχεί στο πραγματικό μηδέν. Δηλαδή μετατοπίζει το αρχικό εύρος τιμών, αφού αυτό έχει πρώτα κλιμακωθεί, ώστε να ταιριάζει το τελικό εύρος τιμών. Προσδιορίζεται από τον τύπο:

$$Z = -\lfloor S \cdot \min(\mathbf{X}) \rfloor - 2^{b-1} \quad (3.3)$$

όπου $\lfloor x \rfloor$ ισούται με το x στρογγυλεμένο στον πλησιέστερο ακέραιο, δηλ. $\lfloor x \rfloor \neq \lceil x \rceil$.

Όταν πραγματοποιηθούν όλοι οι υπολογισμοί στις κβαντισμένες τιμές, οι τιμές του προκύπτοντον τένσορα \mathbf{Y}_{int} μετασχηματίζονται στις αντίστοιχες πραγματικές τιμές. Πράγματι, αν γραφτεί αλλιώς η (3.1) ισχύει ότι:

$$\mathbf{Y}_{\text{real}} = S(\mathbf{Y}_{\text{int}} - Z) \quad (3.4)$$

Με πιο αναλυτικό τρόπο οι παραπάνω σχέσεις γράφονται στο [36] ως:

$$\mathbf{X}_{\text{int}} = \text{clamp}(\lfloor \frac{\mathbf{X}_{\text{real}}}{S} \rfloor + Z; 0, 2^b - 1) \quad (3.5)$$

ή για την προσημασμένη περίπτωση:

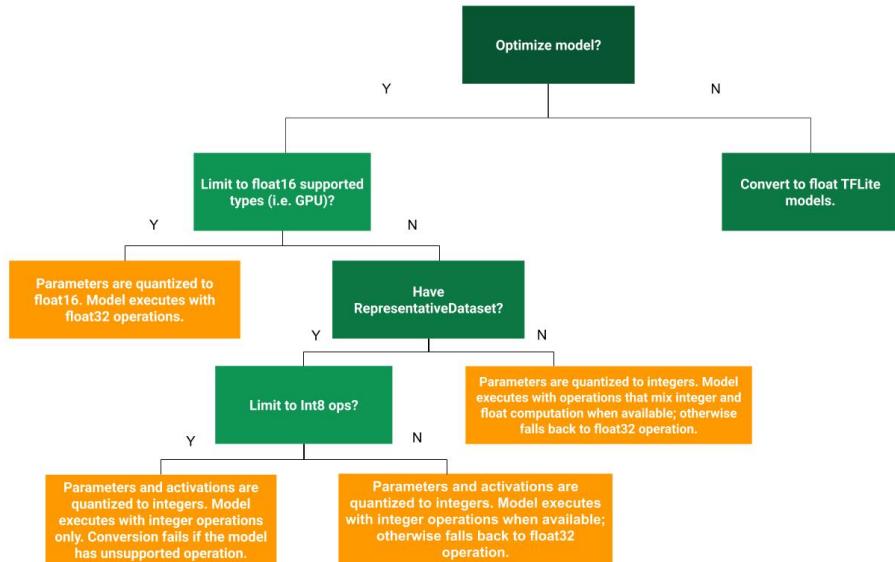
$$\mathbf{X}_{\text{int}} = \text{clamp}(\lfloor \frac{\mathbf{X}_{\text{real}}}{S} \rfloor + Z; -2^{b-1}, 2^{b-1} - 1) \quad (3.6)$$

όπου *clamp* η συνάρτηση αποκοπής:

$$\text{clamp}(x; l, r) = \begin{cases} l, & \text{αν } x < l \\ x, & \text{αν } l \leq x \leq r \\ r, & \text{αν } x > r \end{cases}$$

Πλήθος Ψηφίων: Η τελευταία από τις τρεις παραμέτρους, b , καθορίζει το βαθμό συμπίεσης του νευρωνικού δίκτυου, πέρα από τις τιμές S και Z . Πρακτικά, δε δηλώνεται άμεσα αλλά έμμεσα μέσω του αντίστοιχου τύπου δεδομένων-στόχου, που συνήθως είναι `int8` ή `uint8`. Όσο μικραίνει η τιμή του b , τόσο μεγαλώνει ο βαθμός συμπίεσης και άρα τόσο καλύτερες οι επιδόσεις μνήμης και ισχύος. Ταυτόχρονα, όμως, αυξάνεται όλο και περισσότερο ο κβαντιστικός θόρυβος. Το φαινόμενο αυτό δεν παρατηρείται σε κάθε DNN καθώς σε κάποια από αυτά μπορεί οι συμπίεση των τενσόρων τους να μη συνεπάγεται απώλεια πληροφορίας. Κάποια άλλα DNNs όμως μπορεί να μην είναι τόσο εύρωστα στο θόρυβο της συμπίεσης και να παρατηρείται έντονη πτώση στις επιδόσεις πρόβλεψης του μοντέλου [36].

Οι παραπάνω μαθηματικές αρχές δεν απαντώνται σε κάποια συγκεκριμένη υλοποίηση, αλλά είναι κοινές σε όλες τις πλατφόρμες και προγραμματιστικές βιβλιοθήκες για το σχεδιασμό βαθιών νευρωνικών δίκτυων. Παρ' όλα αυτά, η συγκεκριμένη πτυχιακή εργασία εκπονήθηκε με τη χρήση της βιβλιοθήκης TensorFlow. Στα πλαίσια της εν λόγω πλατφόρμας, η στρατηγική PTQ χωρίζεται σε διάφορες κατηγορίες ανάλογα με το ποιοι τένσορες συμπιέζονται σε ένα νευρωνικό δίκτυο. Παρακάτω παρατίθενται τα διάφορα σχέδια κβαντισμού όπως ορίζονται στο σχετικό παράρτημα [39] του επίσημου οδηγού της TensorFlow.



Σχήμα 3.2: Οι βελτιστοποιήσεις που υποστηρίζει η TensorFlow κατά τη μετατροπή μοντέλων σε TFLite μορφή (πηγή: tensorflow.org).

Κβαντισμός Δυναμικού Εύρους (Dynamic Range Quantization - DRQ): Κατά τον DRQ, ο χρήστης επιλέγει να κβαντιστούν μόνο οι τένσορες των βαρών του συμπλέξου DNN και να εξαρεθούν οι τένσορες εισόδου, εξόδου και οι τένσορες των συναρτήσεων ενεργοποίησης. Η μέθοδος αυτή συμπλέζει άμεσα ένα μεγάλο μέρος του DNN χωρίς να απαιτεί αντιπροσωπευτικό σύνολο δεδομένων (representative dataset). Η συμπίεση με DRQ επιτυγχάνεται ακολουθώντας το μονοπάτι $Y \rightarrow N \rightarrow N$ του σχήματος 3.2.

Κβαντισμός σε Πλήρως Ακέραιες Παραμέτρους (Full Integer Quantization - FIQ): Εδώ ο χρήστης επιλέγει να κβαντιστούν οι τένσορες των βαρών και των ενεργοποίησεων του συμπλέξου DNN. Η χρήση του FIQ, δε συνεπάγεται ότι όλα κβαντιστούν όλοι οι τένσορες, αλλά είναι εγγυημένο ότι όλες οι πραγματικοί παράμετροι όλα αντιστοιχιστούν σε ακέραιες τιμές. Για παράδειγμα, οι μεροληψίες (bias tensors) τύπου float32 δε μικραίνουν σε μέγεθος καθώς δεν αντιστοιχίζονται σε τύπο int8, όμως μετατρέπονται σε τένσορες τύπου int32. Αναλυτικότερα, ο κβαντισμός των μεροληψίων αποφεύγεται καθώς α) οι μεροληψίες είναι διανυσματικής μορφής και όχι πινακοειδής μορφής, καταλαμβάνοντας έτσι ένα μικρό μέρος της μνήμης του DNN και β) κάθε bias vector προστίθενται σε πολλαπλές ενεργοποίησεις, κάνοντας έτσι το νευρωνικό πιο ευπαθές σε σφάλματα λόγω κβαντιστικού θορύβου [14]. Αυτό φαίνεται και από 8-bit Quantization Specification της TensorFlow Lite για διάφορα είδη τενσόρων (σχήμα 3.3). Ακόμα, δε συμπλέζονται ποτέ οι δομικοί (structural) τένσορες που απούνται π.χ. τις τελικές διαστάσεις μετά από ένα Reshape layer. Η μέθοδος αυτή επιτυγχάνει τον υψηλότερο βαθμό συμπίεσης και τη γρηγορότερη παρεμβολή δεδομένων, όμως απαιτεί ένα μικρό αριθμό αντιπροσωπευτικών δειγμάτων (περίπου 100 έως 300 δειγματα) από το σύνολο εκπαίδευσης για να προσδιοριστεί το εύρος των τενσόρων εισόδου και εξόδου. Η συμπίεση με FIQ επιτυγχάνεται ακολουθώντας το μονοπάτι $Y \rightarrow N \rightarrow Y \rightarrow Y$ του σχήματος 3.2.

Άλλα Σχέδια Κβαντισμού: Ο DRQ και ο FIQ αποτελούν τα βασικά σχέδια κβαντισμού αλλά η TensorFlow Lite προσφέρει και άλλες μεθόδους συμπίεσης με QA. Ένα άλλο δημοφιλές σχέδιο είναι η συμπίεση σε float16 που προσφέρει $2 \times$ συμπίεση με ελάχιστη απώλεια ακρίβειας. Αυτό το σχήμα αποτελεί εξαιρετική επιλογή σε περίπτωση που το DNN προορίζεται για να εκτελεστεί σε GPU. Επιτυγχάνεται ακολουθώντας το μονοπάτι $Y \rightarrow Y$ του σχήματος 3.2.

3.5 Εκπαίδευση με Επίγνωση Κβαντισμού (QAT)

Όπως αναφέρθηκε προηγουμένως, η εφαρμογή QA πάνω σε ένα εκπαίδευμένο DNN μπορεί να έχει αρνητικές συνέπειες στις επιδόσεις του όσον αφορά την ικανότητα πρόβλεψης. Η $4 \times$ συμπίεση που προκαλείται από τον 8-bit QA μπορεί και να ζημιώσει έντονα την ακρίβεια του μοντέλου. Ειδικότερα για υψηλούς βαθμούς συμπίεσης όπως ο κβαντισμός σε 4-bit ή, σε ακρα-

```

FULLY_CONNECTED
Input 0:
  data_type : int8
  range    : [-128, 127]
  granularity: per-tensor
Input 1 (Weight):
  data_type : int8
  range    : [-127, 127]
  granularity: per-axis (dim = 0)
  restriction: zero_point = 0
Input 2 (Bias):
  data_type : int32
  range    : [int32_min, int32_max]
  granularity: per-tensor
  restriction: (scale, zero_point) = (input0_scale * input1_scale[...], 0)
Output 0:
  data_type : int8
  range    : [-128, 127]
  granularity: per-tensor

```

(α') Προδιαγραφές των Πλήρως Συνδεδεμένων τενσόρων

```

CONV_2D
Input 0:
  data_type : int8
  range    : [-128, 127]
  granularity: per-tensor
Input 1 (Weight):
  data_type : int8
  range    : [-127, 127]
  granularity: per-axis (dim = 0)
  restriction: zero_point = 0
Input 2 (Bias):
  data_type : int32
  range    : [int32_min, int32_max]
  granularity: per-axis
  restriction: (scale, zero_point) = (input0_scale * input1_scale[...], 0)
Output 0:
  data_type : int8
  range    : [-128, 127]
  granularity: per-tensor

```

(β') Προδιαγραφές των Συνελικτικών τενσόρων

```

DEPTHWISE_CONV_2D
Input 0:
  data_type : int8
  range    : [-128, 127]
  granularity: per-tensor
Input 1 (Weight):
  data_type : int8
  range    : [-127, 127]
  granularity: per-axis (dim = 3)
  restriction: zero_point = 0
Input 2 (Bias):
  data_type : int32
  range    : [int32_min, int32_max]
  granularity: per-axis
  restriction: (scale, zero_point) = (input0_scale * input1_scale[...], 0)
Output 0:
  data_type : int8
  range    : [-128, 127]
  granularity: per-tensor

```

(γ') Προδιαγραφές των Συνελικτικών τενσόρων διαφορετικών πυρήνων (kernel) σε κάθε κανάλι.

Σχήμα 3.3: Προδιαγραφές 8-bit συμπίεσης διαφορετικών ειδών τενσόρων της TensorFlow. Από εδώ επιβεβαιώνεται ότι τα βάρη περιορίζονται στο $[-128, 127]$ ενώ οι μεροληψίες παραμένουν στο εύρος των 32 bit. (πηγή: tensorflow.org)

ίες περιπτώσεις, σε 2-bit ή 1-bit, το DNN επηρεάζεται έντονα από το θόρυβο λόγω απώλειας πληροφοριών. Η στρατηγική της Εκπαίδευσης με Επίγνωση Κβαντισμού (Quantization Aware Training - QAT) στοχεύει στη διατήρηση των επιδόσεων του νευρωνικού δικτύου ακόμα και μετά την εφαρμογή του PTQ. Αυτό επιτυγχάνεται μέσω της μοντελοποίησης του κβαντιστικού θόρυβου που εντοπίζεται κατά την εκπαίδευση [36]. Στην TensorFlow η διαδικασία είναι ως εξής: τα στρώματα του νευρωνικού δικτύου σημειώνονται ως «ψευδοκβαντισμένοι» τένσορες. Έπειτα, το νευρωνικό εκπαιδεύεται για μόνο μία εποχή σε ολόκληρο το σύνολο δεδομένων [40]. Κατά την εκπαίδευση, οι αριθμητικές πράξεις και τα αποτελέσματα παρακολουθούνται και πραγματοποιούνται σαν να ήταν σε κβαντισμένη μορφή (simulated quantization). Έτσι παρακολουθείται και ο κβαντιστικός θόρυβος ώστε να καταπλεμηθεί όταν εφαρμοστεί PTQ στο DNN. Η QAT στρατηγική είναι διαθέσιμη στην TensorFlow μέσω του Python3 πακέτου "tensorflow-model-optimization".

3.6 Προκλήσεις

Σύμφωνα με όσα αναφέρθηκαν στις προηγούμενες υποενότητες, υπάρχουν μοντέλα των οποίων οι επιδόσεις δε ζημιώνονται αφού τους εφαρμοστεί PTQ, ενώ άλλα μπορεί να επηρεαστούν αρνητικά από αυτό. Μια κατηγορία νευρωνικών δικτύων που επηρεάζονται αρνητικά από τη συμπίεση με QA είναι τα μοντέλα που λειτουργούν με δεδομένα χρονοσειρών. Αυτό συνεπάγεται την αδυναμία χρήσης του QA σε πληθύρα εφαρμογών, καθώς δεν είναι σπάνια η χρήση των DNNs σε συσκευές με περιορισμένη μνήμη και ενέργεια για την παρακολούθηση και ανάλυση ακολουθιακών δεδομένων (π.χ. αναγνώριση ανθρώπινης δραστηριότητας με χρήση SmartWatch, άλλες εφαρμογές επεξεργασίας σήματος κλπ). Επομένως, είναι σημαντικό να εξεταστεί πειραματικά ο κβαντισμός διαφόρων μοντέλων χρονοσειρών (χυρίως CNNs και RNNs), πως ανταποκρίνονται αυτά στον PTQ και πως θα μπορούσαν να χρησιμοποιηθούν διάφοροι μέθοδοι, συμπεριλαμβανομένου του QAT για τη βελτίωση της ακρίβειας τους σε περίπτωση που παρουσιάσει πτώση.

Κεφάλαιο 4

Περιβάλλον Ανάπτυξης και Πειραματικά Μοντέλα

4.1 Χαρακτηριστικά Συστήματος

Προχωρώντας στο τεχνικό μέρος της εργασίας, αποτελεί επιτακτική ανάγκη να αναφερθούν οι προδιαγραφές του υλικού και του λογισμικού του συστήματος στο οποίο διεξάχθηκαν όλα τα σχετικά πειράματα ώστε να θεμελιωθεί η αξιοπιστία των αποτελεσμάτων.

- **Κεντρική Μονάδα Επεξεργασίας:** Intel Core i7-8700K, 3.70 GHz, 6 cores, 14MB cache, Coffee Lake 14nm Technology
- **Μνήμη RAM:** 16,0GB Dual-Channel DDR4 @ 1066MHz
- **Μονάδες Αποθήκευσης Χώρου:** 1) 465GB Samsung SSD 870 EVO 500GB, SATA SSD και 2) 1863GB Western Digital WDC WD20EZRZ-00Z5HB0, SATA
- **Κάρτα Γραφικών:** 4095MB NVIDIA GeForce GTX 1070 (ASUSTek Computer Inc), 1506 MHz GPU clock, 4006 MHz shader clock
- **Λειτουργικό Σύστημα:** Windows 10 Home 64-bit, version 22H2
- **Διερμηνέας της Python:** Python 3.10.8 packaged by conda-forge [MSC v.1916 64 bit (AMD64)], conda 24.5.0
- **Περιβάλλον Ανάπτυξης:** Visual Studio Code 1.90.2
- **Python Notebooks:** VSCode Jupyter extension v2024.5.0 by Microsoft
- **TensorFlow:** version 2.15.0
- **TensorFlow Model Optimization:** version 0.8.0

Το μηχάνημα που χρησιμοποιήθηκε είναι σταθερός ηλεκτρονικός υπολογιστής τύπου Desktop και κανένα διάρκεια των πειραμάτων παρέμεινε συνδεδεμένος στο internet μέσω καλωδίου Ethernet.

Χρησιμοποιήθηκε η γλώσσα Python για τη συγγραφή του κώδικα καθώς αυτή διατίθεται για τη χρήση της βιβλιοθήκης TensorFlow καθώς και των συνιστώσων βιβλιοθηκών της όπως Keras, TensorFlow Lite κλπ. Επιπλέον το Visual Studio Code αποτελεί ένα ολοκληρωμένο και προσαρμόσιμο περιβάλλον που είναι αρκετά ελαφρύ και συνεπώς δεν προσδίδει επιπλέον υπολογιστικό φόρτο στο σύστημα, αφήνοντας όσο το δυνατότερο περισσότερους πάρους διαθέσιμους για χρήση από τα εκτελέσιμα scripts. Δεδομένου ακόμα ότι υποστηρίζει και Python notebooks, το καθιστά ιδανικό περιβάλλον για τη συγγραφή του κώδικα και τη διεξαγωγή των αντίστοιχων πειραμάτων.

4.2 Οι βιβλιοθήκες που χρησιμοποιήθηκαν

Σχεδίαση Νευρωνικών Δικτύων: Όλα τα νευρωνικά δίκτυα που εξετάστηκαν σε αυτή την εργασία σχεδιάστηκαν με τη χρήση της προγραμματιστικής βιβλιοθήκης TensorFlow (TF) και των επιμέρους βιβλιοθηκών της που, συλλογικά, αποτελούν έργα ανοιχτού κώδικα. Συγκεκριμένα, η προγραμματιστική διεπαφή Keras που είναι ενσωματωμένη στην TF αποτελεί το κύριο εργαλείο σχεδιασμού των DNNs (sequential models και functional models). Πρωταγωνιστικό ρόλο για την εκπόνηση της εργασίας κατείχε η συνιστώσα βιβλιοθήκη TensorFlow Lite (TFLite). Η TFLite είναι υπεύθυνη για την μετατροπή των DNNs σε κατάλληλη μορφή ώστε να μπορούν να τοποθετηθούν και να εκτελεστούν σε κινητές συσκευές και συσκευές αιχμής. Η συμπίεση των μοντέλων με QA επιτυγχάνεται αποκλειστικά με τη χρήση της TFLite. Τέλος, για την εκτίμηση των επιδόσεων των μοντέλων σε δεδομένα ελέγχου χρησιμοποιήθηκαν συναρτήσεις μετρικών αξιολόγησης όπως ακρίβεια, ανάληση, πίνακας σύγχυσης κλπ. της βιβλιοθήκης Sci-Kit Learn.

Διαχείριση Δεδομένων: Τα δεδομένα εκπαίδευσης, ελέγχου και αποσφαλμάτωσης των μοντέλων αποθηκεύονται και τροφοδοτούνται σε αυτά σε μορφή πινάκων NumPy. Ως προς την ανάλυση και την οπτικοποίηση χρησιμοποιήθηκε η χρήση των βιβλιοθηκών Pandas και Matplotlib.

4.3 Τα Εξεταζόμενα Μοντέλα

Συνολικά εξετάστηκαν 6 διαφορετικά μοντέλα για τη μελέτη του QA στα πλαίσια της TF και τον αντίκτυπό του στη συμπίεση, στην τελική δομή του TFLite μοντέλου και στις επιδόσεις

του μοντέλου. Παρατίθενται με χρονολογική σειρά επεξεργασίας και ανάλυσης τα μοντέλα:

4.3.1 Απλό LSTM δίκτυο για ανάλυση συναίσθημάτος (έτοιμο)

Αποτελεί LSTM δίκτυο που βρέθηκε στην πλατφόρμα Hugging Face (link). Το μοντέλο δεν είναι προ-εκπαιδευμένο άλλα διαθέτει εκ των προτέρων σχεδιασμένη αρχιτεκτονική. Αυτό το DNN αποτελεί ένα toy example κανός η αρχιτεκτονική του είναι πολύ απλή, δεν παρουσιάζει ιδιαίτερες απαιτήσεις μνήμης και το αντίστοιχο σύνολο εκπαίδευσής του διατίθεται εύκολα και άμεσα μέσω του πακέτου keras.datasets.imdb. Αυτό είναι μια συλλογή δεδομένων (\mathbf{x}_i, y_i) όπου τα \mathbf{x}_i είναι κριτικές ταινιών ταινιών σε μορφή κειμένου (είδος χρονοσειράς) που συγκεντρώθηκαν από την πλατφόρμα IMDb που συνοδεύονται από τον αντίστοιχο "thumbs up" ή "thumbs down" χαρακτηρισμό που συνοδεύει την κάθε μία, $y_i \in \{0, 1\}$. Το δοσμένο μοντέλο είναι ένας ταξινομητής που πραγματοποιεί ανάλυση συναίσθημάτος (sentiment analysis). Παρακάτω, παρατίθεται η αρχιτεκτονική του στον Πίνακα 4.1.

Τύπος Στρώματος	Συν. Ενεργοποίησης	Διαστάσεις Εξόδου	Αριθμός Παραμέτρων
InputLayer	-	[(N, len_i)]	0
Embedding	-	(N, len_i, 30)	60,000
Bidirectional	tanh	(N, len_i, 128)	48,640
Bidirectional	tanh	(N, 128)	98,816
Dense	sigmoid	(N, 1)	129
Συνολικός Αριθμός Παραμέτρων:		207,585 (810.88 KB)	
Αριθμός Εκπαίδευσιμων Παραμέτρων:		207,585 (810.88 KB)	
Αριθμός Μη-Εκπαίδευσιμων Παραμέτρων:		0 (0.00 Byte)	

Πίνακας 4.1: Αρχιτεκτονική του έτοιμου LSTM δικτύου. Το Bidirectional στρώμα είναι w-rapper type που εφαρμόζεται πάνω σε LSTM στρώματα, προσδιδόντας τους επιπλέον λειτουργικότητα. Ο αριθμός N είναι το Batch Size ενώ ο αριθμός len_i είναι το μήκος της εκάστοτε χρονοσειράς εισόδου, \mathbf{x}_i .

4.3.2 Απλό LSTM δίκτυο για ανάλυση συναίσθημάτος (υλοποιήθηκε)

LSTM δίκτυο ίδιας λειτουργίας και παρόμοιας αρχιτεκτονικής με το προηγούμενο. Σχεδιάστηκε χειροκίνητα με τη χρήση της κλάσης keras.Sequential. Παρατίθεται η αρχιτεκτονική του στον Πίνακα 4.2.

Τύπος Στρώματος	Συν. Ενεργοποίησης	Διαστάσεις Εξόδου	Αριθμός Παραμέτρων
InputLayer	-	(N, 250)	0
Embedding	-	(N, 250, 64)	128,000
LSTM	tanh	(N, 250, 16)	5,184
LSTM	tanh	(N, 16)	2,112
Dense	sigmoid	(N, 1)	17
Συνολικός Αριθμός Παραμέτρων:		135,313 (528.57 KB)	
Αριθμός Εκπαίδευσιμων Παραμέτρων:		135,313 (528.57 KB)	
Αριθμός Μη-Εκπαίδευσιμων Παραμέτρων:		0 (0.00 Byte)	

Πίνακας 4.2: Αρχιτεκτονική του υλοποιημένου LSTM δικτύου. Αποτελείται από δύο απλά LSTM στρώματα χωρίς τα Bidirectional περιτυλίγματα και διαθέτει λιγότερες παραμέτρους.

4.3.3 Ordonez 2016 Deep Original Model

Το μοντέλο αυτό προέρχεται από το αποθετήριο "Dimension-Adaptive Neural Architecture (DANA)" που είναι διαθέσιμο στην πλατφόρμα GitHub (link). Το έργο αυτό αποσκοπεί στην ανάπτυξη DNNs που εξειδικεύονται στην επεξεργασία σημάτων και είναι εύρωστα στις περιστασιακές ή ολικές αποτυχίες των αισθητήρων. Για να το πετύχουν αυτό, οι ερευνητές προτείνουν αρχιτεκτονικές στις οποίες οι διαστάσεις των τενσόρων/στρωμάτων είναι μεταβλητές. Συγκεχριμένα, σχεδιάζουν διάφορα μοντέλα κάθε ένα από τα οποία έρχεται σε δύο εκδοχές, μία με τένσορες προκαθορισμένων διαστάσεων (κλασική αρχιτεκτονική) και μία με τένσορες δυναμικών διαστάσεων. Το Ordonez 2016 Deep Original Model αποτελεί την κλασική υλοποίηση ενός από αυτά τα μοντέλα. Το σύνολο εκπαίδευσής του είναι το UCI-HAR dataset και η αρχιτεκτονική του αναλύεται στον Πίνακα 4.3.

4.3.4 SmartWatch Model

Το «Μοντέλο Smart Watch» είναι ένας CNN ταξινομητής που προορίζεται προς τοποθέτηση σε ρολόγια χειρός τύπου Smart Watch με σκοπό την αναγνώριση ανθρώπινης δραστηριότητας (Human Activity Recognition - HAR). Σύμφωνα με το [41], οι ερευνητές σχεδίασαν το μοντέλο και το εκπαίδευσαν στο σύνολο δεδομένων WISDM που περιέχει εγγραφές (x_i, y_i) όπου κάθε x_i είναι σήμα από σένσορες επταχυνσιόμετρων και γυροσκόπιων και $y_i \in \{0, 1, 2, , 3 , 4, 5\}$ (walking, typing, drinking, dribbling, writing, clapping) η δραστηριότητα στην οποία αυτό παραπέμπει. Στον Πίνακα 4.4 παρατίθεται η αρχιτεκτονική του μοντέλου Smart Watch.

Τύπος Στρώματος	Συν. Ενεργοποίησης	Διαστάσεις Εξόδου	Αριθμός Παραμέτρων
InputLayer	-	[(N, 128, 6, 1)]	0
Conv2D	ReLU	(N, 124, 6, 64)	384
Conv2D	ReLU	(N, 120, 6, 64)	20,544
Conv2D	ReLU	(N, 116, 6, 64)	20,544
Conv2D	ReLU	(N, 112, 6, 64)	20,544
Reshape	-	(N, 112, 384)	0
LSTM	tanh	(N, 112, 128)	262,656
Dropout	-	(N, 112, 128)	0
LSTM	tanh	(N, 128)	131,584
Dropout	-	(N, 128)	0
Dense	Softmax	(N, 6)	774
Συνολικός Αριθμός Παραμέτρων:		457,030 (1.74 MB)	
Αριθμός Εκπαιδεύσιμων Παραμέτρων:		457,030 (1.74 MB)	
Αριθμός Μη-Εκπαιδεύσιμων Παραμέτρων:		0 (0.00 Byte)	

Πίνακας 4.3: Αρχιτεκτονική του μοντέλου Ordinez 2016 Deep Original.

4.3.5 BSPC Model

Το μοντέλο BSPC, ή όπως αναφέρεται από τους σχεδιαστές του στο [42] ως "ST-CNN-5" (Spatio-Temporal CNN), είναι ένας ταξινομητής πολλαπλών ετικετών (Multi-label Classifier). Το DNN αυτό εκπαιδεύτηκε σε δεδομένα του PTB-XL dataset (link) που συμπεριλαμβάνει εγγραφές (x_i, y_i) όπου x_i σήματα/χρονοσειρές ηλεκτροκαρδιογραφημάτων και $y_i \in \{0, 1\}^5$ το αντίστοιχο δυαδικό διάνυσμα πολλαπλών ετικετών (multilabel binary vector). Οι κλάσεις του συνόλου είναι MI, STTC, HYP, CD και NORM. Η αρχιτεκτονική του παρατίθεται στον πίνακα 4.5.

4.3.6 BSPC GAP Model

Το μοντέλο BSPC GAP, όπως και το BSPC, προτάθηκε στο [42] στο οποίο και αποκαλείται ως "ST-CNN-GAP-5". Σύμφωνα με τους ερευνητές του αποτελεί μια μικρή παραλλαγή του μοντέλου BSPC όπου το τελευταίο Max Pooling στρώμα αντικαθίσταται με ένα Global Average Pooling, μειώνοντας έτσι δραστικά το μέγευθος του μοντέλου. Σύμφωνα με την TensorFlow, μετά από πειραματισμούς, υπάρχει και μια ακόμα μικρή αλλαγή που σχετίζεται με τον στρώματα τύπου Add. Όλες οι παραπάνω αλλαγές φαίνονται στον πίνακα 4.6. Ακόμα, το BSPC GAP εκπαιδεύτηκε, αποσφαλματώθηκε και αξιολογήθηκε στο ίδιο σύνολο δεδομένων με το BSPC.

Τύπος Στρώματος	Συν. Ενεργοποίησης	Διαστάσεις Εξόδου	Αριθμός Παραμέτρων
Conv2D	ReLU	(N, 100, 6, 8)	40
MaxPooling2D	-	(N, 50, 6, 8)	0
Dropout	-	(N, 50, 6, 8)	0
Conv2D	ReLU	(N, 50, 6, 12)	396
MaxPooling2D	-	(N, 25, 6, 12)	0
Conv2D	ReLU	(N, 25, 6, 16)	784
MaxPooling2D	-	(N, 12, 6, 16)	0
Dropout	-	(N, 12, 6, 16)	0
Conv2D	ReLU	(N, 12, 6, 16)	1,040
MaxPooling2D	-	(N, 6, 6, 16)	0
Conv2D	ReLU	(N, 6, 6, 16)	1,040
MaxPooling2D	-	(N, 3, 6, 16)	0
Dropout	-	(N, 3, 6, 16)	0
Conv2D	ReLU	(N, 3, 6, 24)	1,560
MaxPooling2D	-	(N, 2, 6, 24)	0
Flatten	-	(N, 288)	0
Dense	ReLU	(N, 16)	4,624
Dropout	-	(N, 16)	0
Dense	Softmax	(N, 6)	102
Συνολικός Αριθμός Παραμέτρων:		9,586 (37.45 KB)	
Αριθμός Εκπαιδεύσιμων Παραμέτρων:		9,586 (37.45 KB)	
Αριθμός Μη-Εκπαιδεύσιμων Παραμέτρων:		0 (0.00 KB)	

Πίνακας 4.4: Αρχιτεκτονική του μοντέλου Smart Watch. Αποτελείται από λίγες παραμέτρους αλλά έχει μεγάλο βάθος (19 στρώματα). Σκοπός του είναι η επεξεργασία σημάτων στο πεδίο του χρόνου για την αναγνώριση της δραστηριότητας του χρήστη του.

Τύπος Στρώματος	Συν. Ενεργοποίησης	Διαστάσεις Εξόδου	Αριθμός Παραμέτρων
InputLayer	-	[(N, 12, 1000, 1)]	0
Conv2D	Linear	(N, 12, 996, 32)	192
BatchNormalization	ReLU	(N, 12, 996, 32)	128
MaxPooling2D	-	(N, 12, 995, 32)	0
Conv2D	Linear	(N, 12, 991, 32)	5,152
BatchNormalization	ReLU	(N, 12, 991, 32)	128
MaxPooling2D	-	(N, 12, 988, 32)	0
Conv2D	Linear	(N, 12, 984, 64)	10,304
BatchNormalization	ReLU	(N, 12, 984, 64)	256
MaxPooling2D	-	(N, 12, 983, 64)	0
Conv2D	Linear	(N, 12, 981, 64)	12,352
BatchNormalization	ReLU	(N, 12, 981, 64)	256
MaxPooling2D	-	(N, 12, 978, 64)	0
Conv2D	Linear	(N, 12, 976, 64)	12,352
BatchNormalization	ReLU	(N, 12, 976, 64)	256
MaxPooling2D	-	(N, 12, 975, 64)	0
Conv2D	Linear	(N, 1, 975, 64)	49,216
BatchNormalization	ReLU	(N, 1, 975, 64)	256
MaxPooling2D	-	(N, 1, 974, 64)	0
Flatten	-	(N, 62336)	0
Dense	Linear	(N, 128)	7,979,136
BatchNormalization	ReLU	(N, 128)	512
Dropout	-	(N, 64)	0
Dense	Linear	(N, 64)	8,256
BatchNormalization	ReLU	(N, 64)	256
Dropout	-	(N, 64)	0
Dense	Sigmoid	(N, 5)	325
Συνολικός Αριθμός Παραμέτρων:		8,079,333 (30.82 MB)	
Αριθμός Εκπαίδευσιμων Παραμέτρων:		8,079,333 (30.82 MB)	
Αριθμός Μη-Εκπαίδευσιμων Παραμέτρων:		0 (0.00 Byte)	

Πίνακας 4.5: Αρχιτεκτονική του μοντέλου BSPC. Αποτελείται από πολλές παραμέτρους και έχει μεγάλο βάθος (27 στρώματα). Κανονικά, τα BatchNormalization layers δεν έχουν συνάρτηση ενεργοποίησης αλλά κάθε ένα από αυτά ακολουθείται από ένα tf.keras.layers.ReLU στρώμα.

Τύπος Στρώματος	Συν. Ενεργοποίησης	Διαστάσεις Εξόδου	Αριθμός Παραμέτρων
InputLayer	-	[(N, 12, 1000, 1)]	0
Conv2D	Linear	(N, 12, 996, 32)	192
BatchNormalization	ReLU	(N, 12, 996, 32)	128
MaxPooling2D	-	(N, 12, 995, 32)	0
Conv2D	Linear	(N, 12, 991, 32)	5,152
BatchNormalization	ReLU	(N, 12, 991, 32)	128
MaxPooling2D	-	(N, 12, 988, 32)	0
Conv2D	Linear	(N, 12, 989, 64)	14,400
Conv2D	Linear	(N, 12, 984, 64)	10,304
Conv2D	Linear	(N, 12, 984, 64)	24,640
BatchNormalization	ReLU	(N, 12, 984, 64)	256
Add	ReLU	(N, 12, 984, 64)	0
MaxPooling2D	-	(N, 12, 983, 64)	0
Conv2D	Linear	(N, 12, 981, 64)	12,352
BatchNormalization	ReLU	(N, 12, 981, 64)	256
MaxPooling2D	-	(N, 12, 978, 64)	0
Conv2D	Linear	(N, 12, 980, 32)	8,224
Conv2D	Linear	(N, 12, 976, 64)	12,352
Conv2D	Linear	(N, 12, 976, 64)	10,304
BatchNormalization	-	(N, 12, 976, 64)	256
Add	ReLU	(N, 12, 976, 64)	0
MaxPooling2D	-	(N, 12, 975, 64)	0
Conv2D	Linear	(N, 1, 975, 64)	49,216
BatchNormalization	ReLU	(N, 1, 975, 64)	256
GlobalAvgPool2D	-	(N, 64)	0
Flatten	-	(N, 64)	0
Dense	Linear	(N, 128)	8,320
BatchNormalization	ReLU	(N, 128)	512
Dropout	-	(N, 128)	0
Dense	Linear	(N, 64)	8,256
BatchNormalization	ReLU	(N, 64)	256
Dropout	-	(N, 64)	0
Dense	Sigmoid	(N, 5)	325

Συνολικός Αριθμός Παραμέτρων:	166,085 (648.77 KB)
Αριθμός Εκπαιδεύσιμων Παραμέτρων:	166,085 (648.77 KB)
Αριθμός Μη-Εκπαιδεύσιμων Παραμέτρων:	0 (0.00 Byte)

Πίνακας 4.6: Αρχιτεκτονική του μοντέλου BSPC GAP. Δεν έχει πολλές παραμέτρους αλλά έχει πολύ μεγάλο βάθος (33 στρώματα). Τα στρώματα του είναι σχεδόν πλήρως διαδοχικά με την εξαίρεση των στρωμάτων τύπου Add που συνδέονται με τα δύο προηγούμενά τους, τα οποία και αιθροίζουν.

4.4 Μετατροπή σε Μοντέλο TensorFlow Lite

Σε αυτή την παράγραφο θα παρουσιαστεί αναλυτικότερα η συνιστώσα βιβλιοθήκη TFLite και πως αυτή χρησιμοποιήθηκε για την εκπόνηση της εργασίας. Το σημαντικότερο προγραμματιστικό εργαλείο της αποτελεί ο Μετατροπέας TFLite που είναι υλοποιημένος στην κλάση `tf.lite.TFLiteConverter` (CITE). Ο Μετατροπέας ενθυλακώνει τη λογική της μετατροπής ενός keras sequential ή keras functional ή PB μοντέλου στο αντίστοιχο μοντέλο TFLite που παράγεται στην έξοδο του μετατροπέα στη μορφή ενός byte string. Το byte string αυτό μπορεί να αποθηκευτεί για μετέπειτα χρήση σε δυαδικό αρχείο και να εισαχθεί στο δεύτερο σημαντικό εργαλείο της βιβλιοθήκης, τον Διερμηνέα TFLite που υλοποιείται στην κλάση `tf.lite.Interpreter` (CITE). Ο Διερμηνέας χρησιμοποιείται για την παρεμβολή δεδομένων στο δοσμένο TFLite μοντέλο και συνεπώς δίνει τη δυνατότητα στους προγραμματιστές να το χρησιμοποιήσουν και να το αποσφαλματώσουν σε περιβάλλον ανάπτυξης χώδικα.

Πέρα από τον Μετατροπέα και τον Διερμηνέα, η TFLite προσφέρει και άλλα εργαλεία που διευκολύνουν τη σχεδίαση και αποσφαλμάτωση των TFLite μοντέλων. Δύο από αυτά τα εργαλεία που χρησιμοποιήθηκαν ευρέως στην εργασία είναι ο Αναλυτής TFLite (`tf.lite.experimental.Analyzer`) και ο QA Αποσφαλματωτής (`tf.lite.experimental.QuantizationDebugger`). Ο τελευταίος χρησιμοποιείται σε TFLite μοντέλα που έχουν υποστεί χβαντισμό (βλ. επόμενη παράγραφο).

4.5 Βελτιστοποιήσεις του Μετατροπέα TFLite

Όπως προαναφέρθηκε, ο Μετατροπέας αποτελεί την αποκλειστική προγραμματιστική διεπαφή της TensorFlow για τη συμπίεση μοντέλων με μεθόδους QA. Αυτό μπορεί να επιτευχθεί με κατάλληλο χειρισμό των πεδίων της κλάσης `tf.lite.TFLiteConverter`.

4.5.1 Απλή Μετατροπή σε TFLite (Default Conversion)

Είναι πρέπον να εξεταστεί αρχικά η μετατροπή χωρίς βελτιστοποιήσεις. Στο σχήμα 4.1 παρουσιάζεται ο κώδικας για τη μετατροπή ενός μοντέλου από PB μορφή (SavedModel) σε TFLite μορφή, χωρίς να εφαρμόζονται σε αυτό βελτιστοποιήσεις. Η διαδικασία αυτή αντιστοιχεί στο μονοπάτι N του σχήματος 3.2.

```
# Plain TFLite model - All tensors are FLOAT32 (No optimizations)
converter = tf.lite.TFLiteConverter.from_saved_model("./pb_model_dir")
converter.optimizations = []
tflite_model = converter.convert()
```

Σχήμα 4.1: Μετατροπή σε απλό μοντέλο TFLite. Όλοι οι τένσορες/στρώματα του παραγόμενου μοντέλου είναι τύπου float32. Η δεύτερη γραμμή πριν το τέλος είναι προαιρετική.

4.5.2 Μετατροπή σε TFLite με Συμπίεση DRQ

Για τον κβαντισμό του μοντέλου σύμφωνα με το σχέδιο DRQ μπορεί να χρησιμοποιηθεί ο ίδιος κώδικας με πριν, με τη διαφορά ότι ανατίθεται ένα flag, σε μορφή λίστας, στο πεδίο "optimizations" του Μετατροπέα TFLite (σχήμα 4.2). Υπενθυμίζεται ότι το DRQ ανήκει στην PTQ στρατηγική κβαντισμού.

```
# Dynamic Range Quantization - DRQ
converter = tf.lite.TFLiteConverter.from_saved_model("./pb_model_dir")
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()
```

Σχήμα 4.2: Μετατροπή σε TFLite μοντέλο με συμπίεση DRQ. Η TF προσπαθεί να μετατρέψει όλους τους τένσορες βαρών του παραγόμενου μοντέλου σε int8, ενώ όλοι οι άλλοι παραμένουν σε float32.

4.5.3 Μετατροπή σε TFLite με Συμπίεση FIQ

Ο ίδιος κώδικας για τη συμπίεση του παραγόμενου TFLite DNN με DRQ, μαζί με την προσθήκη μερικών επιπλέον γραμμών, χρησιμοποιείται και για τη συμπίεση με τη μέθοδο FIQ. Οι προσθήκες στον κώδικα αφορούν κάποια flags σχετικά με τις προδιαγραφές στόχου και ένα αντιπροσωπευτικό σύνολο δεδομένων (Representative Dataset - RDS). Στο σχήμα 4.3α' φαίνεται η υλοποίηση ενός RDS ενώ στα σχήματα 4.3β', 4.3γ' και 4.3δ' παρουσιάζονται διάφορα

κομμάτια κώδικα για συμπίεση με FIQ. Να σημειωθεί ότι η διαφορά των τριών αποσπασμάτων κώδικα για FIQ συμπίεση είναι το αν και πως θα κβαντιστούν οι τένσορες εισόδου εξόδου (FIQ-FLOAT32, FIQ-INT8 και FIQ-UINT8 αντίστοιχα).

```
# Constructs a representative dataset generator with the specified dataset and sample size
def representative_dataset(dataset: np.ndarray, n_samples: int):
    def _representative_data_gen():
        for input_value in tf.data.Dataset.from_tensor_slices(dataset).batch(1).take(n_samples):
            # Model has only one input so each data point has one element.
            yield [input_value]
    return _representative_data_gen
```

(α') Υλοποίηση του αντιπροσωπευτικού συνόλου δεδομένων. Δεν αποτελεί πραγματικό σύνολο δεδομένων (π.χ. NumPy πίνακας) αλλά μια συνάρτηση-γεννήτρια που επιστρέφει δειγματοληπτικά κάποιες από τις εγγραφές εκπαίδευσης.

```
# Full Integer Quantization - INT8 weights, INT8 activations, INT32 biases & FLOAT32 I/O
converter = tf.lite.TFLiteConverter.from_saved_model("./pb_model_dir")
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.representative_dataset = representative_dataset(dataset=X_train, n_samples=300)
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
converter.target_spec.supported_types = [tf.int8]
tflite_model = converter.convert()
```

(β') Μετατροπή σε TFLite μοντέλο με συμπίεση FIQ. Όλοι οι τένσορες του παραγόμενου μοντέλου μετατρέπονται σε int8 εκτός δύο, τους τένσορες εισόδου (input layer) και εξόδου (final output layer), οι οποίοι παραμένουν σε float32 για λόγους συμβατότητας (σχέδιο FIQ-FLOAT32).

```
# Full Integer Quantization - INT8 weights, INT8 activations, INT32 biases & INT8 I/O
converter = tf.lite.TFLiteConverter.from_saved_model("./pb_model_dir")
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.representative_dataset = representative_dataset(dataset=X_train, n_samples=300)
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
converter.target_spec.supported_types = [tf.int8]
converter.inference_input_type = tf.int8
converter.inference_output_type = tf.int8
tflite_model = converter.convert()
```

(γ') Μετατροπή σε TFLite μοντέλο με συμπίεση FIQ. Όλοι οι τένσορες του παραγόμενου μοντέλου μετατρέπονται σε int8 και οι τένσορες εισόδου/εξόδου συμπλέζονται σε int8 αφού αυτό δηλωθεί ρητά στον κώδικα (σχέδιο FIQ-INT8).

```

# Full Integer Quantization - INT8 weights, INT8 activations, INT32 biases & UINT8 I/O
converter = tf.lite.TFLiteConverter.from_saved_model("./pb_model_dir")
converter.optimizations = [tf.lite.Optimize.DEFAULT]
converter.representative_dataset = representative_dataset(dataset=X_train, n_samples=300)
converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
converter.target_spec.supported_types = [tf.int8]
converter.inference_input_type = tf.uint8
converter.inference_output_type = tf.uint8
tflite_model = converter.convert()

```

(δ') Μετατροπή σε TFLite μοντέλο με συμπίεση FIQ. Όλοι οι τένσορες του παραγόμενου μοντέλου μετατρέπονται σε int8 και οι τένσορες εισόδου/εξόδου συμπιέζονται σε uint8 αφού αυτό δηλωθεί ρητά στον κώδικα (σχέδιο FIQ-UINT8).

Σχήμα 4.3: Python3 κώδικας για τη μετατροπή ενός μοντέλου σε μορφή TFLite με συμπίεση FIQ.

Κεφάλαιο 5

Πειραματικά Δεδομένα και Ανάλυσή τους

5.1 Ο Αντίκτυπος του QA στα απλά μοντέλα

Σε αυτό το κεφάλαιο θα συζητηθούν σε βάθος τα πειραματικά αποτελέσματα που προέκυψαν από τις χβαντιστικές στρατηγικές και μεθόδους που αναφέρθηκαν στο προηγούμενο κεφάλαιο, όπου αυτές εφαρμόστηκαν. Ξεκινώντας από τα απλά LSTM μοντέλα που αναφέρθηκαν στις παραγράφους 4.3.1 και 4.3.2. Εξετάστηκε αρχικά η μετατροπή των εν λόγω μοντέλων σε TFLite με, και χωρίς βελτιστοποιήσεις. Τα αποτελέσματα αναγράφονται στον πίνακα 5.1.

Εξεταζόμενη Ποσότητα	Έτοιμο μοντέλο	Υλοποιημένο μοντέλο
Μέγεθος Αρχικού (Keras) Μοντέλου:	810.88KB	528.57KB
Μέγεθος Τελικού: (TFLite) Μοντέλου:	843.9KB	546.1KB
Μέγεθος TFLite Μοντέλου με DRQ (PTQ):	239.7KB	150.6KB
Παραγόντας Συμπίεσης:	×3.52	×3.6
Ακρίβεια Πρόβλεψης Αρχικού Μοντέλου:	88.3%	87.3%
Ακρίβεια Πρόβλεψης TFLite Μοντέλου μετά από DRQ:	-	49.7%
Ακρίβεια Πρόβλεψης του DRQ TFLite με QAT fine-tuning:	N/A	62.84%

Πίνακας 5.1: Τα αποτελέσματα των δύο αρχικών LSTM μοντέλων.

Με βάση τα παραπάνω αποτελέσματα γίνονται οι κατάλληλες παρατηρήσεις και εξάγονται τα ακόλουθα συμπεράσματα:

1. Η μετατροπή ενός μοντέλου σε μοντέλο TFLite ελαφρώς αυξάνει το μέγεθος του μοντέλου. Αυτή η μικρή αύξηση στην ποσότητα μνήμης προφανώς οφείλεται στην υλοποίηση της TensorFlow.
2. Η συμπίεση με DRQ δεν μικράνει το μέγεθος του μοντέλου κατά ακριβώς $\times 4$, αλλά κατά περίπου $\times 3.5$ έως $\times 3.6$. Από όλες δοκιμαστικές εκτελέσεις, για μεγαλύτερο πλήθος νευρώνων στο ίδιο υλοποιημένο μοντέλο, παρατηρήθηκε ότι ο παράγοντας συμπίεσης αυξάνεται σε $\times 3.8$ και παραπάνω, χωρίς να φτάνει ή να υπερβαίνει την τιμή $\times 4$. Από αυτό είναι ασφαλές να συμπεράνουμε ότι όσο πιο μεγάλο το μοντέλο, τόσο πιο πολύ ο παράγοντας τείνει στο 4 που αποτελεί και το μέγιστο όριο. Από την όλη, όσο πιο μικρό το αρχικό μέγεθος του μοντέλου, τόσο πιο μικρός ο παράγοντας (φαινομενικά τείνει στο 3), χωρίς να υπάρχει κάποιο εξακριβωμένο κάτω όριο. Αυτά τα συμπεράσματα συμβαδίζουν με τις θεωρητικές αρχές του DRQ που αναλύθηκαν σε προηγούμενα κεφάλαια, καιώς οι πίνακες των βαρών αυξάνονται τετραγωνικά σε σχέση με τα γραμμικά αυξανόμενα διανύσματα μεροληψιών και ενεργοποιήσεων που δεν συμπλέζονται.
3. Η ακρίβεια πρόβλεψης της PTQ εκδοχής του έτοιμου μοντέλου δεν μπορεί, προς το παρόν, να εκτιμηθεί καθώς το στρώμα εισόδου του έχει τη μορφή (None,) για να δέχεται διανύσματα εισόδου απροσδιόριστου μεγέθους. Αυτό επηρεάζει αρνητικά τη μετατροπή του μοντέλου σε TFLite καθώς η μετατροπή της εισόδου αντιστοιχίζεται σε σχήμα εισόδου ίσο με (1,). Επιπλέον δεν μπορεί να εφαρμοστεί QAT σε αυτό καθώς δεν υποστηρίζονται τα Embedding, LSTM και Bidirectional στρώματα από τις συναρτήσεις "quantize_apply" και "quantize_annotation_model" που είναι απαραίτητες για το QAT.
4. Το προαναφερθέν πρόβλημα δεν παρουσιάστηκε στο υλοποιημένο μοντέλο καθώς η έισοδός του ορίστηκε ως (maxlen,) αντί για (None,) προκειμένου να αντιμετωπιστεί αυτό το πρόβλημα. Όμως, παρά το ότι η παρεμβολή δεδομένων μπόρεσε να πραγματοποιηθεί, η ακρίβεια πρόβλεψης έπεσε από 87.3% σε 49.7%. Γενικά, είναι γνωστό ότι το PTQ προκαλεί μια πτώση στην ακρίβεια του μοντέλου όμως η παρατηρούμενη πτώση είναι καταστροφική καθώς το PTQ μοντέλο φτάνει στην ελάχιστη δυνατή απόδοσή του, σαν να μην εκπαιδεύτηκε ποτέ (δυαδικός κατηγοριοποιητής με απόδοση 50% είναι σαν

να κάνει τυχαίες προβλέψεις).

5. Το πρόβλημα της μεγάλης πτώσης στην ακρίβεια παρουσιάστηκε στο μοντέλο και κατά το PTQ του υλοποιημένου μοντέλου. Η μία εποχή εκπαίδευσης μετά τη συμπίεση ξεκίνησε από 50% ακρίβεια και τερμάτισε σε 63%, ενώ θα έπρεπε να είχε ξεκινήσει από το 87% του κανονικού μοντέλου και να συνέχιζε από εκεί.

Τα παραπάνω μοντέλα παρουσιάζουν σοβαρά θέματα ως προς το DRQ για τις δύο στρατηγικές συμπίεσης. Ένας σοβαρός λόγος που παρατηρούνται τα παραπάνω είναι ότι τα δοσμένα μοντέλα αποτελούνται από Embedding layers. Τα στρώματα αυτά παιζουν καθοριστικό ρόλο στην επεξεργασία φυσικής γλώσσας (NLP). Συγκεκριμένα, κάθε λέξη w_i αντιστοιχίζεται σε ένα διάνυσμα συγκεκριμένης διάστασης, $w_i \in \mathbb{R}^d$ (π.χ. $d = 32$). Αυτό αποφεύγει την αναπαράσταση των λέξεων σε One-Hot encoded μορφή που θα αποτελούσε μεγάλη σπατάλη μνήμης χωρίς, ταυτόχρονα, να καταφεύγει στη χρήση ακεραίων αριθμών που, λανθασμένα, θα συσχέτιζε τις λέξεις που αντιστοιχίζονται σε αριθμητικά κοντινές τιμές. Ουσιαστικά, το Embedding layer είναι ένας πίνακας διάστασης $N_w \times d$, όπου N_w ο αριθμός των λέξεων στο λεξιλόγιο. Η i -οστή σειρά αντιστοιχεί σε ένα word embedding για τη λέξη w_i .

Προφανώς τα word embeddings, δεν προορίζονται για κβαντισμό όπως φαίνεται από τις καταστροφικές επιδόσεις του μοντέλου μετά τον κβαντισμό του embedding matrix σε int8. Αυτό επιβεβαιώνεται αν συμπιέσουμε άλλα απλά, τον example μοντέλα παρόμοιας αρχιτεκτονικής που συμπεριλαμβάνουν LSTM στρώματα αλλά δεν περιλαμβάνουν Embedding στρώματα. Σε αυτή την περίπτωση, π.χ. για ένα μικρό DNN που εκπαιδεύτηκε στο CIFAR-10 dataset, παρατηρήθηκε ότι η ακρίβεια πρόβλεψης διατηρήθηκε σταθερά υψηλή.

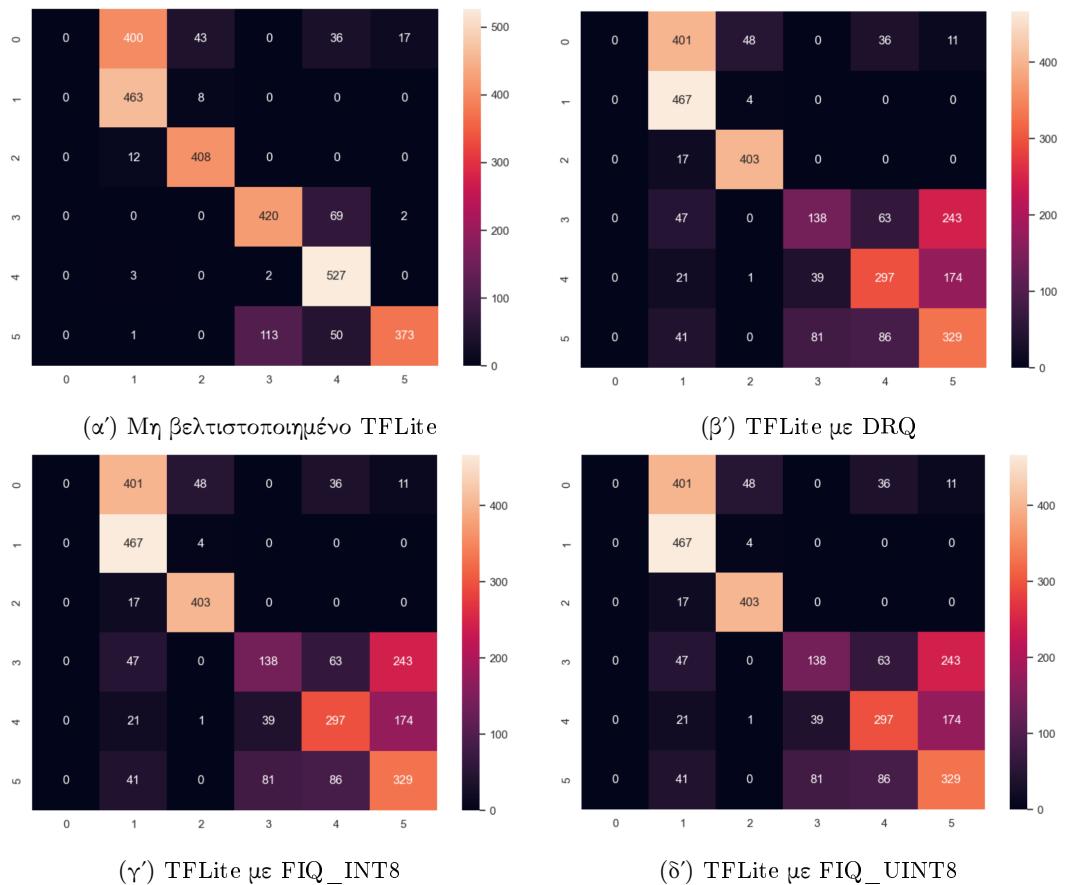
Η προβληματική φύση του Embedding layer δεν αποτελεί εμπόδιο για τη συνέχιση της εργασίας. Τα υπόλοιπα τέσσερα DNNs δεν ασχολούνται με επεξεργασία φυσικής γλώσσας και συνεπώς δε συμπεριλαμβάνουν αυτόν τον τύπο στρώματος.

5.2 Αποτελέσματα του Μοντέλου Ordonez

Το μοντέλο Ordonez 2016 Deep Original συμπιέστηκε με τη στρατηγική PTQ. Χρησιμοποιήθηκαν τρεις διαφορετικές μέθοδοι QA οι οποίες είναι οι DRQ, FIQ INT8 και FIQ UINT8. Τα αποτελέσματά των διαφόρων εκδοχών του DNN συγχρίνονται μεταξύ τους καθώς και με τη μη-βελτιστοποιημένη TFLite εκδοχή στον πίνακα 5.2. Ακόμα, στο σχήμα 5.1 παρέχονται οι πίνακες σύγχυσης των διαφορετικών εκδοχών του μοντέλου που παράχθηκαν με τη χρήση του πακέτου Seaborn.

Μετρική	Default TFLite	DRQ	FIQ INT8	FIQ UINT8
Accuracy	74.3%	74.5%	55.4%	55.4%
Precision	65.4%	65.6%	49.0%	49.0%
Recall	74.9%	75.1%	56.7%	56.7%
F1 Score	68.4%	68.5%	50.3%	50.3%
Balanced Accuracy	74.9%	75.1%	56.7%	56.7%
Cohen's Kappa	69.2%	69.4%	46.5%	46.5%
Μέγεθος σε Μνήμη	1.75 MB	465.21 KB	466.48 KB	466.48 KB

Πίνακας 5.2: Τα αποτελέσματα των δύο αρχικών LSTM μοντέλων.



Σχήμα 5.1: Πίνακες Σύγχυσης για τα τέσσερα TFLite μοντέλα Ordóñez.

Από τα παραπάνω γίνεται φανερό ότι, φαίνεται ότι το DRQ διατηρεί ικανοποιητικά τις επιδόσεις του μοντέλου και μάλιστα παρουσιάζει περίπου +0.2% αύξηση στις περισσότερες μετρικές.

Ταυτοχρόνως, ο παράγοντας συμπίεσης είναι $\times 3.85$. Από την άλλη, δεν μπορεί να ειπωθεί το ίδιο για τα πειραματικά αποτελέσματα του Κβαντισμού σε Πλήρως Ακέραιες Παραμέτρους, καθώς αυτά είναι κάθε άλλο παρά επαινετικά. Οι FIQ μέθοδοι με int8 και uint8 I/O τένσορες έχουν ακριβώς τις ίδιες επιδόσεις μεταξύ τους, αλλά πολύ χειρότερες από τη μέθοδο DRQ. Επιπλέον, η συμπίεση που πετυχαίνουν είναι ίδια με της μεθόδου DRQ.

5.2.1 Περαιτέρω Ανάλυση με τον TFLite Analyzer

Στη συγκεκριμένη περίπτωση η συμπίεση με τη χρήση DRQ αποδείχτηκε επαρκής και συνεπώς δε χρειάζεται να προσφύγει κάποιος σε FIQ, εκτός αν το επεξεργαστικό σύστημα-στόχος του DNN δεν υποστηρίζει FPU. Σε κάθε περίπτωση, όμως, είναι σημαντικό να γίνει μια βαθύτερη ανάλυση του τι συνέβη στην εκάστοτε μέθοδο συμπίεσης και γιατί συνέβη ότι συνέβη. Αυτό μπορεί να διαπιστωθεί σε πρώτη φάση με την επιθεώρηση των τελικών αρχιτεκτονικών των TFLite μοντέλων Ordenez, πράγμα που γίνεται δυνατό με τη χρήση του Αναλυτή TFLite που αναφέρθηκε νωρίτερα.

Ο Αναλυτής TFLite εκτυπώνει τον DNN γράφο με σαφή και ακριβή τρόπο σε ένα ρεύμα εξόδου. Κάθε TFLite μοντέλο είναι, ουσιαστικά, ένας γράφος (TF Graph) που αποτελείται από επιμέρους γράφους (Subgraph):

$$G = \{S_1, S_2, \dots, S_N\} \quad (5.1)$$

Κάθε subgraph αποτελείται από ένα σύνολο κόμβων ή τελεστών (Operators), O , και ένα σύνολο τενσόρων, T :

$$S_i = (\{O_1^{(i)}, O_2^{(i)}, \dots, O_{n_i}^{(i)}\}, T^{(i)}) \quad (5.2)$$

Στις περισσότερες περιπτώσεις, ένα TFLite μοντέλο αποτελείται από ένα subgraph, $G = \{S_1\}$. Κάθε τελεστής είναι μια συνάρτηση που αντιστοιχίζει τους τένσορες σε τένσορες:

$$O_j^{(i)} : T_{in}^{(i,j)} \rightarrow T_{out}^{(i,j)} \quad (5.3)$$

με

$$(T_{in}^{(i,j)}, T_{out}^{(i,j)} \subset T^{(i)}) \wedge (T_{in}^{(i,j)} \cap T_{out}^{(i,j)} = \emptyset) \quad (5.4)$$

Ένας τελεστής μπορεί, χωρίς αυτό να είναι απαραίτητο, να αντιστοιχεί σε ένα στρώμα του αρχικού μοντέλου Keras.

```

*** ./TFLiteAPI_data/TFLiteOriginal/Ordonez2016OepOriginal.tflite ***

Your TFLite model has '1' subgraph(s). In the subgraph description below,
T# represents the Tensor numbers. For example, in Subgraph#0, the CONV_2D op takes
tensor #0 and tensor #5 and tensor #1 as input and produces tensor #40 as output.

Subgraph#0 main[T#0] -> [T#52]
Op#0 CONV_2D(T#0, T#5, T#1) -> [T#40]
Op#1 CONV_2D(T#40, T#6, T#2) -> [T#41]
Op#2 CONV_2D(T#41, T#7, T#3) -> [T#42]
Op#3 CONV_2D(T#42, T#8, T#0) -> [T#43]
Op#4 RESHAPE([T#43, T#9[1, 112, 384]]) -> [T#44]
Op#5 UNIDIRECTIONAL_SEQUENCE_LSTM([T#44, T#20, T#21, T#22, T#23, T#19, T#18, T#15, T#1, T#1, T#1, T#1, T#16, T#17, T#18, T#19, T#1, T#1, T#10, T#45, T#1, T#1, T#1, T#1] -> [T#46]
Op#6 UNIDIRECTIONAL_SEQUENCE_LSTM([T#46, T#22, T#3, T#4, T#35, T#27, T#1, T#25, T#26, T#27, T#28, T#29, T#30, T#31, T#1, T#1, T#47, T#48, T#1, T#1, T#1, T#1] -> [T#49]
Op#7 STRIDED_SLICE(T#49, T#37[0, -1, 0], T#38[0, 1, 1]) -> [T#50]
Op#8 FULLY_CONNECTED([T#50, T#36, T#11]) -> [T#51]
Op#9 SOFTMAX(T#51) -> [T#52]

Tensors in Subgraph#0
T#0{erving_default_X0} shape:[1, 128, 6, 1] type:FLOAT32
T#1{erving_default_B0} shape:[1, 128, 6, 1] type:FLOAT32
T#2{Ordonez2016OepOriginal/conv2d/1/BiasAdd/ReadVariableOp} shape:[64] type:FLOAT32 R0 256 bytes, buffer: 2, data:[-0.00803411, -0.124549, -0.196648, -0.077513, -0.0199805, ...]
T#3{Ordonez2016OepOriginal/conv2d/2/BiasAdd/ReadVariableOp} shape:[64] type:FLOAT32 R0 256 bytes, buffer: 3, data:[-0.00803034, -0.134824, -0.170547, -0.0362999, -0.0271092, ...]
T#4{Ordonez2016OepOriginal/conv2d/3/BiasAdd/ReadVariableOp} shape:[64] type:FLOAT32 R0 256 bytes, buffer: 4, data:[-0.117748, -0.159945, 0.034555, -0.0836584, 0.00597235, ...]
T#44{Ordonez2016OepOriginal/conv2d/Conv2D} shape:[64, 5, 1, 1] type:FLOAT32 R0 128 bytes, buffer: 6, data:[0.459056, 0.684819, 0.441297, 0.610416, 0.60319, ...]
T#46{Ordonez2016OepOriginal/conv2d/1/Conv2D} shape:[64, 5, 1, 64] type:FLOAT32 R0 256 bytes, buffer: 7, data:[0.0687966, -0.0553872, 0.0257135, -0.0888524, -0.156818, ...]
T#47{Ordonez2016OepOriginal/conv2d/2/Conv2D} shape:[64, 5, 1, 64] type:FLOAT32 R0 256 bytes, buffer: 8, data:[0.0704327, 0.0110409, -0.144053, 0.0391149, 0.0397749, ...]
T#48{Ordonez2016OepOriginal/conv2d/3/Conv2D} shape:[64, 5, 1, 64] type:FLOAT32 R0 256 bytes, buffer: 9, data:[-0.141425, -0.0744295, -0.195969, -0.280924, 0.186446, ...]
T#49{Ordonez2016OepOriginal/Reshape/Reshape/shape} shape:[3] type:INT32 R0 12 bytes, buffer: 10, data:[1, 112, 384]
T#50{Ordonez2016OepOriginal/1/lstm_1/zero} shape:[1, 128] type:FLOAT32
T#51{Ordonez2016OepOriginal/act_smax/BiasAdd/ReadVariableOp} shape:[6] type:FLOAT32 R0 24 bytes, buffer: 12, data:[-0.48629, -0.44409, -0.348137, 0.00764835, 0.0708629, ...]
T#52{arith.constant} shape:[128, 128] type:FLOAT32 R0 65536 bytes, buffer: 13, data:[0.258995, 0.284971, -0.041245, -0.304958, 0.154378, ...]
T#53{arith.constant1} shape:[128, 128] type:FLOAT32 R0 65536 bytes, buffer: 14, data:[0.0570565, 0.158553, 0.149234, -0.141468, 0.0865517, ...]
T#54{arith.constant2} shape:[128, 128] type:FLOAT32 R0 65536 bytes, buffer: 15, data:[-0.0498687, -0.243882, 0.130425, -0.22345, 0.256805, ...]
T#55{arith.constant3} shape:[128, 128] type:FLOAT32 R0 65536 bytes, buffer: 16, data:[0.0589119, 0.0681213, -0.246118, 0.259315, 0.0740927, ...]
T#56{arith.constant4} shape:[128, 128] type:FLOAT32 R0 65536 bytes, buffer: 17, data:[-0.0197305, -0.0601358, 0.134541, -1.07561, 0.9968284, ...]
T#57{arith.constants} shape:[128] type:FLOAT32 R0 512 bytes, buffer: 18, data:[0.216647, 1.1598, 0.950456, 0.860289, 0.049331, ...]
T#58{arith.constants1} shape:[128] type:FLOAT32 R0 512 bytes, buffer: 19, data:[0.8382216, 0.111155, 0.0447223, 0.0743328, 0.104592, ...]
T#59{arith.constants2} shape:[128] type:FLOAT32 R0 512 bytes, buffer: 20, data:[-0.0795939, 0.146259, -0.0195804, -0.0547612, -0.0135013, ...]
T#60{arith.constants3} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 21, data:[-0.148892, -0.0625826, -0.0682294, -0.0246636, -0.0705258, ...]
T#61{arith.constants4} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 22, data:[-0.0768944, 0.0311715, -0.16822, -0.162803, 0.073488, ...]
T#62{arith.constants5} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 23, data:[-0.0473734, -0.0328612, 0.0742154, -0.00471381, -0.0198856, ...]
T#63{arith.constants6} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 24, data:[-0.0473734, -0.0328612, 0.0742154, -0.00471381, -0.0198856, ...]
T#64{arith.constants7} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 25, data:[-0.006961, -0.0877379, -0.236617, 0.032346, -0.158717, ...]
T#65{arith.constants8} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 26, data:[0.207344, 0.17718, 0.21069, 0.106098, 0.153624, -0.0046765, ...]
T#66{arith.constants9} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 27, data:[-0.171958, 0.0421638, 0.0421638, 0.0416285, -0.099408, ...]
T#67{arith.constants10} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 28, data:[0.0669021, -0.104021, 0.0948391, 0.0416285, -0.099408, ...]
T#68{arith.constants11} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 29, data:[-0.295921, -0.0676814, 0.017416, -0.132088, 0.133546, ...]
T#69{arith.constants12} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 30, data:[1.36582, 1.15577, 1.17069, 1.1153, 0.04372, ...]
T#70{arith.constants13} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 31, data:[0.146795, 0.0489693, 0.212538, 0.049873, 0.0100789, ...]
T#71{arith.constants14} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 32, data:[0.146795, 0.0489693, 0.212538, 0.049873, 0.0100789, ...]
T#72{arith.constants15} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 33, data:[0.122659, 0.198247, -0.0280887, 0.331538, -0.123318, ...]
T#73{arith.constants16} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 34, data:[0.044302, 0.187792, -0.00811242, 0.154878, -0.0803052, ...]
T#74{arith.constants17} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 35, data:[0.0548626, -0.187334, -0.071959, -0.123805, 0.246647, ...]
T#75{arith.constants18} shape:[128, 384] type:FLOAT32 R0 196608 bytes, buffer: 36, data:[0.0661808, 0.224546, -0.0572519, -0.0302795, -0.347681, ...]
T#76{Ordonez2016OepOriginal/act_smax/MatMul} shape:[6, 128] type:FLOAT32 R0 3072 bytes, buffer: 37, data:[-0.374556, -1.00484, -1.02492, 0.09668, -0.143924, ...]
T#77{arith.constants19} shape:[3] type:INT32 R0 12 bytes, buffer: 38, data:[0, -1, 0]
T#78{arith.constants20} shape:[3] type:INT32 R0 12 bytes, buffer: 39, data:[0, 0, 0]
T#79{arith.constants26} shape:[3] type:INT32 R0 12 bytes, buffer: 40, data:[1, 1, 1]
...

```

Σχήμα 5.2: Ένα μέρος του log file της αρχιτεκτονικής του Default TFLite μοντέλου. Εκτυπώνονται πρώτα τα υπογραφήματα (ένα υπογράφημα στη συγκεκριμένη περίπτωση) και ακολουθούν οι τένσορές τους, μαζί με τις ιδιότητες και τις μεταξύ τους σχέσεις που αυτοί έχουν.

Όπως φαίνεται στο σχήμα 5.2, ο Αναλυτής καταγράφει αναλυτικά όλα τα υπογραφήματα, όλους τους τελεστές και όλους τους τένσορες μαζί με τις μεταξύ τους σχέσεις, σύμφωνα με όσα ορίστηκαν στις σχέσεις 5.1, 5.2, 5.3 και 5.4. Επιπλέον παρατίθενται χρήσιμες πληροφορίες όπως ο τύπος δεδομένων και το μέγεθος του κάθε τένσορα.

Αριθμός Τενσόρων	Default TFLite	DRQ	FIQ
int8 τένσορες	0	19	38
int16 τένσορες	0	0	2
int32 τένσορες	4	4	17
float32 τένσορες	49	30	8
Σύνολο	53	53	65

Πίνακας 5.3: Αριθμός τενσόρων ανά TFLite εκδοχή του μοντέλου.

Από την έξοδο του Αναλυτή TFLite, εξήχθησαν τα αποτελέσματα του πίνακα 5.3. Το Default TFLite μοντέλο αποτελείται εξ ολοκλήρου από float32 τένσορες, εξαιρώντας τους δομικούς τένσορες που αποθηκεύονται ως int32. Επιπλέον, όπως ήταν αναμενόμενο, παρατηρείται μια αύξηση στους int8 τένσορες και μια ελάττωση στους float32 τένσορες στο DRQ μοντέλο και ακόμα περισσότερο στο FIQ μοντέλο. Ταυτοχρόνως, τα αποτελέσματα θίγουν και μια ενδιαφέρουσα ανωμαλία. Συγκεκριμένα, το TFLite μοντέλο που βελτιστοποιήθηκε με FIQ έχει 8 float32 τένσορες που κανονικά απαγορεύεται από αυτή τη κβαντιστική μέθοδο. Διερευνώντας περισσότερο τη δομή του αντίστοιχου TFLite στο αρχείο log file του Αναλυτή, παρατηρείται το εξής:

FLOAT32 tensors:

```
> T#47(input_to_input_intermediate) shape:[0], type:FLOAT32
> T#48(input_to_forget_intermediate) shape:[0], type:FLOAT32
> T#49(input_to_cell_intermediate) shape:[0], type:FLOAT32
> T#50(input_to_output_intermediate) shape:[0], type:FLOAT32
> T#55(input_to_input_intermediate1) shape:[0], type:FLOAT32
> T#56(input_to_forget_intermediate1) shape:[0], type:FLOAT32
> T#57(input_to_cell_intermediate1) shape:[0], type:FLOAT32
> T#58(input_to_output_intermediate1) shape:[0], type:FLOAT32
```

Σχήμα 5.3: Οι float32 τένσορες του FIQ-compressed TFLite μοντέλου. Το στιγμιότυπο ουθόνης δεν αποτελεί απόσπασμα του πραγματικού log file της αρχιτεκτονικής, αλλά ενός αντίστοιχου αρχείου στο οποίο έχουν ταξινομηθεί οι τένσορες με βάση τον τύπο δεδομένων. Το εν λόγω αρχείο μπορεί εύκολα να προκύψει μετά από επεξεργασία του αρχικού log file χρησιμοποιώντας κανονικές εκφράσεις.

Οι float32 τένσορες που κανονικά δε θα έπρεπε να υπάρχουν, είναι βαθμίδας 1 και διάστασης 0 (σχήμα 5.3). Άρα δεν έχουν καθόλου δεδομένα και μπορεί λοιπόν αυτή η κατάσταση να θεωρηθεί παράγωγη ενός bug. Ενδεχομένως αυτοί να είναι ενδιάμεσοι τένσορες που χρησιμοποιήθηκαν κατά τη διαδικασία μετατροπής σε TFLite, αλλά δε διαγράφτηκαν ποτέ. Πάντως, ο Αναλυτής TFLite αποτελεί πειραματικό εργαλείο στην TF v2.15.0 και τα αποτελέσματά του πρέπει να χρησιμοποιούνται με επιφύλαξη.

Μια τελευταία παρατήρηση πάνω στα δεδομένα του πίνακα 5.3 είναι ότι, ακόμα κι αν αφαιρούνται αυτοί οι κενοί, ελαττωματικοί float32 τένσορες, ο συνολικός αριθμός των τενσόρων θα ήταν πάλι μεγαλύτερος (57 αντί για 53). Δε βρέθηκε κάποια σχετική αναφορά στην αύξηση του αριθμού των τενσόρων λόγω της χρήσης της βελτιστοποίησης FIQ κατά τη μετατροπή.

5.2.2 Περαιτέρω Ανάλυση με τον Quantization Debugger

Ένα δεύτερο εργαλείο που μπορεί να χρησιμοποιηθεί για ακόμα περισσότερη εμβάθυνση πάνω στα συμπιεσμένα μοντέλα είναι ο QA Αποσφαλματωτής ή Quantization Debugger ή, απλά, Αποσφαλματωτής. Ο Αποσφαλματωτής αποτελεί πειραματικό εργαλείο που χρησιμοποιείται αποκλειστικά για την επιλεώρηση της συμπεριφοράς των FIQ-compressed μοντέλων TFLite με int8 ενεργοποιήσεις [19]. Δεν μπορεί να χρησιμοποιηθεί για την εξέταση των DRQ μοντέλων, πράγμα που επιβεβαιώθηκε πειραματικά.

Ο Quantization Debugger υπολογίζει τις αριθμητικές διαφορές ανάμεσα στα στρώματα του κβαντισμένου μοντέλου και στα αντίστοιχα στρώματα του Default TFLite μοντέλου. Αυτές οι αριθμητικές διαφορές μπορούν να εκφραστούν ως στατιστικές μετρικές όπως τυπική απόκλιση, μέσο σφάλμα, μέγιστο απόλυτο σφάλμα κλπ.

$$d_i = \text{out}(l'_i) - \text{out}(l_i) \quad (5.5)$$

όπου l_i και l'_i το i -οστό στρώμα του default και του κβαντισμένου TFLite μοντέλου αντίστοιχα. Η σχέση 5.5 περιγράφει τη διαδικασία κατά την παρεμβολή ενός δείγματος. Για την εξαγωγή των προαναφερθέντων στατιστικών μετρικών χρειάζονται πολλά δείγματα και άρα ο Αποσφαλματωτής στηρίζεται στην ύπαρξη ενός RDS. Για κάθε \mathbf{x}_j , $j = 1\dots N$ παράγονται τα αντίστοιχα $d_i^{(j)}$ που χρησιμοποιούνται για να εξαγχθούν οι στατιστικές μετρικές, π.χ.

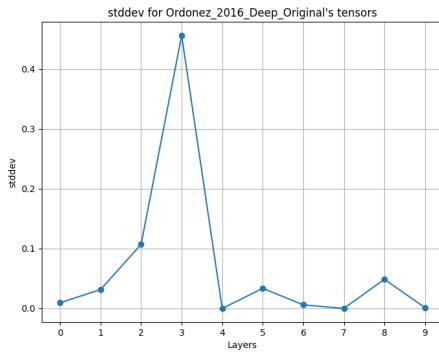
$$\sigma_i = \sqrt{\frac{1}{N} \sum_{j=1}^N (d_i^{(j)} - \mu_i)^2} \quad (5.6)$$

με

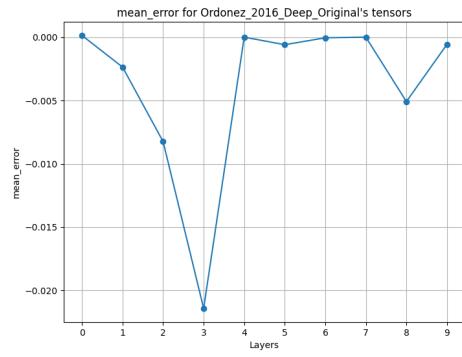
$$\mu_i = \frac{1}{N} \sum_{j=1}^N d_i^{(j)} \quad (5.7)$$

Με πιο απλά λόγια:

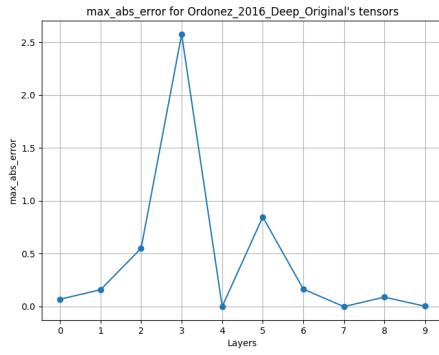
- Χρησιμοποιείται ένα δείγμα, \mathbf{x}_j , του RDS για την εξαγωγή του $d_i^{(j)}$ στο i -οστό στρώμα του TFLite νευρωνικού.
- Επαναλαμβάνουμε για κάθε δείγμα \mathbf{x}_j .
- Επαναλαμβάνουμε για κάθε στρώμα.
- Επαναλαμβάνουμε για κάθε μετρική.



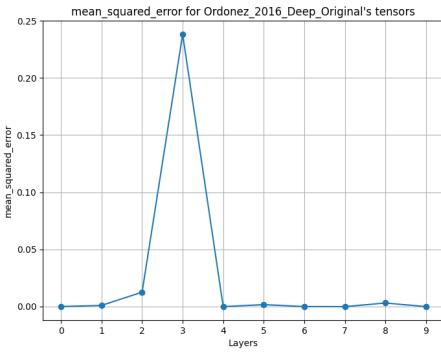
(α') Τυπική Απόκλιση



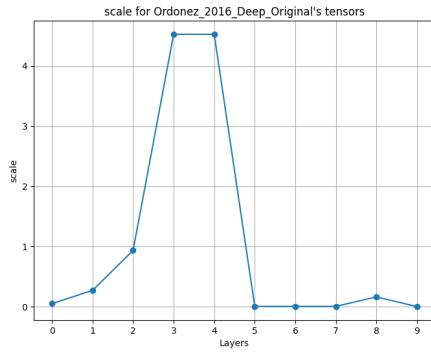
(β') Μέσο Σφάλμα



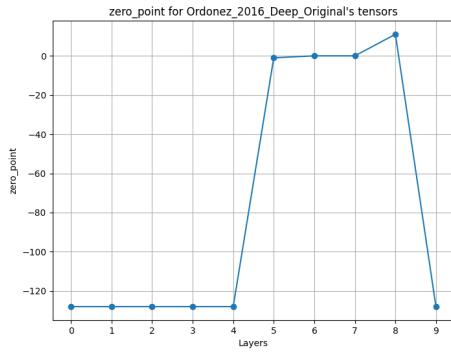
(γ') Μέγιστο Απόλυτο Σφάλμα



(δ') Μέσο Τετραγωνικό Σφάλμα



(ε') Συντελεστής Κλιμάκωσης, S



(φ') Συντελεστής Μηδενικού Σημείου, Z

Σχήμα 5.4: Στατιστικές μετρικές των διαφορών μεταξύ των FIQ-κβαντισμένων και των αντίστοιχων μη κβαντισμένων στρωμάτων του Ordonez 2016 Deep Original στα σχήματα α' έως δ'. Πέρα από αυτές τις μετρικές, ο Αποσφαλματωτής εξάγει και τους συντελεστές S και Z της εξίσωσης 3.1 (σχήματα ε' και στ').

Στο παρακάτω σχήμα (5.5) παρατίθεται η αντιστοίχιση των στρωμάτων του DNN και της δεικτού διότησης του οριζόντιου άξονα των γραφημάτων του σχήματος 5.4. Πέρα από αυτό όμως, αναγράφονται και δύο ακόμα μετρικές. Η πρώτη είναι η εμβέλεια, $range = 255 \cdot S$, που δεν έχει ιδιαίτερη σημασία. Από την άλλη, η δεύτερη είναι η $\frac{\sqrt{MSE}}{S}$ (ρίζες των τιμών του σχήματος 5.4δ' προς τις αντίστοιχες τιμές του σχήματος 5.4ε') και είναι ζωτικής σημασίας στον εντοπισμό κακώς κβαντισμένων στρωμάτων.

	op_name	range	rmse/scale
0	CONV_2D	13.603479	0.175395
1	CONV_2D	69.710686	0.116620
2	CONV_2D	238.833000	0.120061
3	CONV_2D	1153.436068	0.107895
4	RESHAPE	1153.436068	0.000000
5	UNIDIRECTIONAL_SEQUENCE_LSTM	2.000000	5.256445
6	UNIDIRECTIONAL_SEQUENCE_LSTM	1.999971	0.911836
7	STRIDED_SLICE	1.999971	0.000000
8	FULLY_CONNECTED	41.883747	0.340047
9	SOFTMAX	0.996094	0.383085

Σχήμα 5.5: Τα στρώματα του τελικού FIQ μοντέλου Ordnez, όπως ορίζονται από τον Αποσφαλματωτή. Η περιγραφή της αρχιτεκτονικής συμβαδίζει με την περιγραφή του πίνακα 4.3.

Σύμφωνα με το [19], το $\frac{\sqrt{MSE}}{S}$ έχει τιμή κοντά στο $1/\sqrt{12}$ (≈ 0.289) όταν η κβαντισμένη κατανομή είναι παρόμοια με την αρχική (float32) κατανομή, υποδεικνύοντας ένα καλώς κβαντισμένο μοντέλο. Όσο αυξάνεται η τιμή του $\frac{\sqrt{MSE_i}}{S_i}$, τόσο πιο πιθανό είναι να μην έχει κβαντιστεί σωστά το i -οστό στρώμα. Απομονώνοντας τα στρώματα με υψηλή τιμή $\frac{\sqrt{MSE_i}}{S_i}$ εξάγονται τα ακόλουθα αποτελέσματα:

layer_stats[layer_stats['rmse/scale'] > 0.7][['op_name', 'range', 'rmse/scale', 'tensor_name']]			
	op_name	range	rmse/scale
5	UNIDIRECTIONAL_SEQUENCE_LSTM	2.000000	5.256445
6	UNIDIRECTIONAL_SEQUENCE_LSTM	1.999971	0.911836

Σχήμα 5.6: Τα ελαττωματικά στρώματα του τελικού FIQ μοντέλου Ordnez.

Από το σχήμα 5.6 φαίνεται ότι τα αναδρομικά στρώματα δεν κβαντίστηκαν σωστά, πράγμα που συνέβαλε στη δραστική πτώση των επιδόσεων του μοντέλου.

5.2.3 Επιλεκτικός Κβαντισμός Στρωμάτων.

Ο QA Αποσφαλματωτής επιτρέπει τον επιλεκτικό κβαντισμό στρωμάτων. Με αυτόν τον τρόπο, τα επιλεγμένα στρώματα παραμένουν σε float32 ενώ τα υπόλοιπα συμπιέζονται κανονικά με τη μέθοδο FIQ INT8. Ταυτοχρόνως, ο online οδηγός της TFLite στο [19] συμβουλεύει να εξαιρέσουμε και κάποια αρχικά στρώματα πέρα από τα προβληματικά. Άρα μπορούμε να συμπιέσουμε εκ νέου το μοντέλο Ordnez με FIQ INT8 εξαιρώντας α) τα δύο προβληματικά LSTM στρώματα και β) τα δύο αρχικά στρώματα του μοντέλου και, έπειτα, να υπολογίσουμε τις επιδόσεις του μερικώς κβαντισμένου μοντέλου. Τα αποτελέσματα αυτής της διαδικασίας φαίνονται στον πίνακα 5.4:

Accuracy	Precision	Recall	F1 Score	Balanced Acc.	Cohen's Kappa
69.6%	61.9%	70.6%	63.5%	70.6%	63.6%

Πίνακας 5.4: Τα αποτελέσματα του μερικώς FIQ INT8-συμπιεσμένου Ordnez model.

Από τα αποτελέσματα είναι φανερό ότι παρουσιάζεται σημαντική βελτίωση στις μετρικές των επιδόσεων σε σχέση με τα δεδομένα του πίνακα 5.2. Αναλυτικότερα, η ακρίβεια πρόβλεψης αυξάνεται στο 69.6% από το αρχικό 55.4%. Η ευστοχία (precision) ανεβαίνει στο 61.9% από 49%, η ανάχληση (recall) ανεβαίνει στο 70.6% από 56.7% και αυτή η σημαντική αύξηση παρατηρείται και σε όλες τις υπόλοιπες μετρικές.

Δυστυχώς, παρά τις βελτιώσεις των επιδόσεων, η προσέγγιση αυτή δεν φαίνεται να αποτελεί καλύτερη επιλογή από τη μέθοδο DRQ λόγω των ακόλουθων μειονεκτημάτων:

1. Παρατηρείται σημαντική μείωση στο συντελεστή συμπίεσης του παραγόμενου TFLite μοντέλου, καθώς το DNN συμπιέζεται στα 1.63 MB 1.75 MB. Την ίδια στιγμή, όταν η μέθοδος FIQ εφαρμόζεται σε όλο το νευρωνικό, συμπιέζεται στα 466.48 KB.
2. Το μερικώς FIQ compressed μοντέλο έχει το ίδιο πρόβλημα με το πλήρες DRQ μοντέλο, δηλαδή υποστηρίζεται από επεξεργαστικές συσκευές που δε διαθέτουν FPU, καθώς αποτελείται από float32 τένσορες. Αυτή η κατακερματισμένη δομή του μοντέλου, έχει ως αποτέλεσμα πιο αργό λανθάνον χρόνο παρεμβολής που προκαλείται κυρίως από το κόστος μεταφοράς δεδομένων μεταξύ της ΠΥ και των εν λόγω συσκευών [19].
3. Τέλος, δεν πετυχαίνουμε καλύτερες επιδόσεις από τη μέθοδο DRQ έτσι κι αλλιώς.

Συνοψίζοντας, η μέθοδος DRQ αποδείχθηκε πολύ πιο ισχυρή και αποδοτική, σε κάθε περίπτωση, από τη μέθοδο FIQ. Επομένως, η τελική επιλογή για συμπίεση ενός DNN με παρόμοια αρχιτεκτονική (Conv layers + LSTM layers) είναι η DRQ.

5.3 Τα Αποτελέσματα των Συνελικτικών Μοντέλων

Σε αυτήν την υποενότητα θα εξεταστούν τα μοντέλα SmartWatch, BSPC και BSPC GAP. Αρχικά όμως, είναι σημαντικό να αναφερθεί ότι το κάθε μοντέλο έχει διαφορετικές εκδοχές και όρα τα εξεταζόμενα μοντέλα δεν είναι τρία, αλλά περισσότερα. Στον παρακάτω πίνακα (5.5) φαίνονται αναλυτικά όλα τα μοντέλα και οι περιγραφές τους.

Οι 3 Φάσεις των Μοντέλων			
Μοντέλο	Αρχική Εκδοχή: Χωρίς pruning ή LRF	1 ^η συμπίεση: Εφαρμογή pruning, χωρίς LRF)	2 ^η συμπίεση: Εφαρμογή LRF πάνω από το pruning
SmartWatch	1 Lean & 1 Fat	1 Lean & 1 Fat	3 Lean & 2 Fat
BSPC	1 Lean & 1 Fat	1 Lean & 1 Fat	1 Lean & 1 Fat
BSPC GAP	1 Lean & 1 Fat	1 Lean & 1 Fat	1 Lean & 1 Fat

Πίνακας 5.5: Όλες οι εξεταζόμενες εκδοχές των συνελικτικών μοντέλων. Τα μοντέλα διαχωρίζονται σε τρεις φάσεις: χωρίς συμπίεση, 1^η συμπίεση (pruning) και 2^η συμπίεση (pruning + LRD). Στις πρώτες δύο φάσεις, κάθε μοντέλο χωρίζεται σε δύο εκδοχές μια lean και μια fat. Στην τρίτη φάση, λόγω του ότι το LRF οδηγεί σε πολλές λύσεις, μπορεί να έχουμε περισσότερα από ένα lean ή fat μοντέλα.

Οι χαρακτηρισμοί "lean" και "fat" σχετίζονται με την αρχική αρχιτεκτονική του μοντέλου. Για παράδειγμα, το lean SmartWatch είναι το SmartWatch μοντέλο αλλά με λιγότερες παραμέτρους (νευρώνες ή/και στρώματα) από ότι η αυθεντική του εκδοχή που περιγράφεται στον πίνακα 4.4. Συνεπώς το lean SmartWatch είναι μικρότερο σε μέγεθος από ότι αυτό του πίνακα 4.4. Αντιστοίχως το fat SmartWatch είναι ένα μοντέλο παρόμοιας αρχιτεκτονικής αλλά με περισσότερες παραμέτρους, δηλαδή ένα πιο «χοντρό» SmartWatch μεγαλύτερου μεγέθους. Παρομοίως, προκύπτουν και τα lean/fat BSPC και BSPC GAP μοντέλα που είναι πιο ελαφριές/βαριές αρχιτεκτονικές από αυτές που φαίνονται στους πίνακες 4.5 και 4.6.

5.3.1 Μέθοδοι Συμπίεσης που Χρησιμοποιήθηκαν

Για τον κβαντισμό όλων των μοντέλων του της 1^{ης} και 2^{ης} φάσης του πίνακα 5.5 χρησιμοποιήθηκαν και οι δύο στρατηγικές κβαντισμού, PTQ και QAT. Όμως, για τα τελικά μοντέλα (3^η φάση) χρησιμοποιήθηκε μόνο PTQ καθώς δεν μπορεί να εφαρμοστεί QAT στους τένσορες KerasDense που αυτά διαθέτουν (βλ. πίνακα 5.6), οι οποίοι και προέχουν από τη συμπίεση με LRF. Το γιατί προέχουν οι KerasDense, ποια είναι η λειτουργία τους και γιατί δεν μπορούν να γίνουν fine-tuned είναι ερωτήματα που δε θα απαντηθούν καθώς είναι εκτός της εμβέλειας αυτής της εργασίας.

Τύπος Στρώματος	Συν. Ενεργοποίησης	Διαστάσεις Εξόδου	Αριθμός Παραμέτρων
InputLayer	-	[(N, 100, 6, 1)]	0
Conv2D	ReLU	(N, 100, 6, 4)	20
MaxPooling2D	-	(N, 50, 6, 4)	0
Dropout	-	(N, 50, 6, 4)	0
Conv2D	ReLU	(N, 50, 6, 6)	102
MaxPooling2D	-	(N, 25, 6, 6)	0
Conv2D	ReLU	(N, 25, 6, 3)	75
MaxPooling2D	-	(N, 13, 6, 3)	0
Dropout	-	(N, 13, 6, 3)	0
Conv2D	ReLU	(N, 13, 6, 4))	52
MaxPooling2D	-	(N, 7, 6, 4)	0
Flatten	-	(N, 168)	0
KerasDense	Linear	(N, 8)	100
Dropout	-	(N, 8)	100
Dense	Softmax	(N, 6)	54
Συνολικός Αριθμός Παραμέτρων:		403 (1.57 KB)	
Αριθμός Εκπαιδεύσιμων Παραμέτρων:		403 (1.57 KB)	
Αριθμός Μη-Εκπαιδεύσιμων Παραμέτρων:		0 (0.00 Byte)	

Πίνακας 5.6: Αρχιτεκτονική μίας εκ των τριών lean εκδοχών του μοντέλου SmartWatch μετά την εφαρμογή parameter pruning και LRF (3^η φάση). Αποτελείται από πολύ λιγότερες παραμέτρους παραμέτρους σε σχέση με το αρχικό μοντέλο (πίνακας 4.4). Έχει ήδη επιτευχθεί ×23.85 συμπίεση χωρίς τη χρήση QA που θα εφαρμοστεί σε λίγο.

Χρησιμοποιήθηκαν τέσσερεις διαφορετικές PTQ μέθοδοι. Αυτές είναι οι DRQ, FIQ FLOAT32, FIQ INT8 και FIQ UINT8, όπου:

- **FIQ FLOAT32:** Εφαρμογή της default FIQ μεθόδου όπου ο πρώτος και ο τελευταίος τένσορας του DNN παραμένουν σε float32 για λόγους συμβατότητας.
- **FIQ INT8:** Εφαρμογή της FIQ μεθόδου όπου ο πρώτος και ο τελευταίος τένσορας συμπιέζονται με QA σε int8.
- **FIQ UINT8:** Εφαρμογή της FIQ μεθόδου όπου ο πρώτος και ο τελευταίος τένσορας συμπιέζονται με QA σε uint8.

Παρακάτω, στους πίνακες 5.7, 5.8, 5.9, 5.10, 5.11, 5.12, παρουσιάζεται η ακρίβεια πρόβλεψης των μοντέλων κατά τις φάσεις 1 και 2. Τα χρωματιστά βελάκια στους πίνακες των αποτελεσμάτων του QAT υποδεικνύουν αν το QAT fine-tuning βοήθησε ή χειροτέρεψε την κατάσταση σε σχέση με το αντίστοιχο PTQ σενάριο.

Smart Watch (PTQ)	Pre-Pruning		Post-Pruning		
	Σχέδιο QA	Lean	Fat	Lean	Fat
Mη Κβαντισμένο	93.4%: 23.36	94.9%: 337.28	92.3%: 10.61	94.0%: 128.42	
DRQ	93.4%: $\times 2.01$	94.8%: $\times 3.59$	92.3%: $\times 1.57$	94.0%: $\times 3.23$	
FIQ FLOAT32	88.8%: $\times 2.10$	91.2%: $\times 3.46$	87.2%: $\times 1.45$	92.7%: $\times 3.06$	
FIQ INT8	80.9%: $\times 2.16$	82.1%: $\times 3.47$	77.6%: $\times 1.51$	83.0%: $\times 3.08$	
FIQ UINT8	80.0%: $\times 2.09$	76.6%: $\times 3.46$	75.5%: $\times 1.45$	80.3%: $\times 3.05$	

Πίνακας 5.7: Η επίδοση των εκδοχών του μοντέλου SmartWatch (phase 1 & 2) σε ακρίβεια πρόβλεψης μετά από PTQ. Σε κάθε περίπτωση παρατίθεται το αρχικό μέγεθος του μη κβαντισμένου μοντέλου (σε KB) και, στο εκάστοτε σχέδιο QA, παρατίθεται ο συντελεστής συμπίεσης.

Smart Watch (QAT)	Pre-Pruning		Post-Pruning		
	Σχέδιο QA	Lean	Fat	Lean	Fat
Mη Κβαντισμένο	DC	DC	DC	DC	DC
DRQ	90.4% ↓	91.8% ↓	90.4% ↓	91.8% ↓	
FIQ FLOAT32	91.5% ↑	91.5% ↑	90.5% ↑	93.3% ↑	
FIQ INT8	79.3% ↓	75.0% ↓	62.4% ↓	70.0% ↓	
FIQ UINT8	68.7% ↓	71.9% ↓	68.9% ↓	73.4% ↓	

Πίνακας 5.8: Η επίδοση των εκδοχών του μοντέλου SmartWatch (phase 1 & 2) σε ακρίβεια πρόβλεψης μετά από QAT και PTQ (DC = Don't Care).

BSPC (PTQ)	Pre-Pruning		Post-Pruning		
	Σχέδιο QA	Lean	Fat	Lean	Fat
Mη Κβαντισμένο	86.2%: 423.87	87.8%: 15959.73	85.1%: 114.06	86.3%: 3273.36	
DRQ	86.1%: $\times 3.58$	87.8%: $\times 3.99$	85.1%: $\times 2.65$	86.4%: $\times 3.94$	
FIQ FLOAT32	85.0%: $\times 3.47$	87.7%: $\times 3.98$	84.9%: $\times 2.77$	86.5%: $\times 3.94$	
FIQ INT8	84.3%: $\times 3.48$	87.4%: $\times 3.98$	84.6%: $\times 2.79$	85.8%: $\times 3.94$	
FIQ UINT8	85.1%: $\times 3.47$	87.6%: $\times 3.98$	84.3%: $\times 2.77$	86.5%: $\times 3.94$	

Πίνακας 5.9: Η επίδοση των εκδοχών του μοντέλου BSPC (phase 1 & 2) σε δυδαδική ακρίβεια πρόβλεψης μετά από PTQ. Σε κάθε περίπτωση παρατίθεται το αρχικό μέγεθος του μη κβαντισμένου μοντέλου (σε KB) και, στο εκάστοτε σχέδιο QA, παρατίθεται ο συντελεστής συμπίεσης.

BSPC (QAT)	Pre-Pruning		Post-Pruning		
	Σχέδιο QA	Lean	Fat	Lean	Fat
Mη Κβαντισμένο	DC	DC	DC	DC	DC
DRQ	87.3%	87.1%	84.5%	85.8%	
FIQ FLOAT32	87.7%	87.1%	85.0%	85.9%	
FIQ INT8	84.4%	85.5%	82.6%	84.1%	
FIQ UINT8	85.1%	86.5%	82.9%	84.2%	

Πίνακας 5.10: Η επίδοση των εκδοχών του μοντέλου BSPC (phase 1 & 2) σε δυδαδική ακρίβεια πρόβλεψης μετά από QAT και PTQ (DC = Don't Care).

BSPC GAP (PTQ)	Pre-Pruning		Post-Pruning		
	Σχέδιο QA	Lean	Fat	Lean	Fat
Mη Κβαντισμένο	85.9%: 1561.68	87.5%: 16186.70	84.0%: 399.56	85.1%: 2697.38	
DRQ	86.1%: $\times 3.82$	87.6%: $\times 3.98$	83.8%: $\times 3.42$	85.1%: $\times 3.91$	
FIQ FLOAT32	84.4%: $\times 3.77$	86.7%: $\times 3.98$	82.7%: $\times 3.37$	85.1%: $\times 3.89$	
FIQ INT8	84.1%: $\times 3.77$	86.6%: $\times 3.98$	81.6%: $\times 3.38$	85.0%: $\times 3.89$	
FIQ UINT8	84.6%: $\times 3.77$	86.3%: $\times 3.98$	82.9%: $\times 3.37$	85.0%: $\times 3.89$	

Πίνακας 5.11: Η επίδοση των εκδοχών του μοντέλου BSPC GAP (phase 1 & 2) σε δυδαδική ακρίβεια πρόβλεψης μετά από PTQ. Σε κάθε περίπτωση παρατίθεται το αρχικό μέγεθος του μη κβαντισμένου μοντέλου (σε KB) και, στο εκάστοτε σχέδιο QA, παρατίθεται ο συντελεστής συμπίεσης.

BSPC GAP (QAT)	Pre-Pruning		Post-Pruning		
	Σχέδιο QA	Lean	Fat	Lean	Fat
Mη Κβαντισμένο	DC	DC	DC	DC	DC
DRQ	86.4% \uparrow	87.4% \downarrow	84.1% \uparrow	85.6% \uparrow	
FIQ FLOAT32	87.3% \uparrow	86.0% \downarrow	83.8% \uparrow	85.1% \sim	
FIQ INT8	84.0% \downarrow	83.8% \downarrow	82.4% \uparrow	85.0% \sim	
FIQ UINT8	81.7% \uparrow	86.2% \downarrow	83.7% \uparrow	84.0% \downarrow	

Πίνακας 5.12: Η επίδοση των εκδοχών του μοντέλου BSPC GAP (phase 1 & 2) σε δυδαδική ακρίβεια πρόβλεψης μετά από QAT και PTQ (DC = Don't Care).

Από τα αποτελέσματα φαίνεται ότι τα BSPC και BSPC GAP μοντέλα παρουσιάζουν μεγαλύτερη ευστάθεια, καθώς ο QA δεν επηρεάζει τόσο έντονα το accuracy τους. Κατά τα άλλα, για όλα τα μοντέλα η ακρίβεια είναι υψηλότερη για τα μη κβαντισμένα και τα DRQ compressed μοντέλα, ενώ παρατηρείται μια πτώση στα FIQ FLOAT32 compressed μοντέλα και μια ακόμα μεγαλύτερη πτώση για τα FIQ INT8 και FIQ UINT8 compressed μοντέλα. Επίσης, στις περισσότερες περιπτώσεις τα fat μοντέλα έχουν ελαφρώς καλύτερες επιδόσεις από τα lean μοντέλα. Τέλος, η χρήση της στρατηγικής QAT άλλες φορές μπορεί να ευνοήσει το μοντέλο και άλλες όχι τόσο. Η πιο ωφέλιμη χρήση της, όμως, φαίνεται να είναι στη FIQ FLOAT32 συμπίεση που είναι και η default FIQ. Από τους πίνακες 5.8, 5.10 και 5.12 μπορεί να διαπιστωθεί ότι τα FIQ FLOAT32 μοντέλα φτάνουν ή, ακόμα σε κάποιες περιπτώσεις, ξεπερνάνε τις επιδόσεις της μεθόδου DRQ στο σενάριο της QAT.

Post-LRF SmartWatch					
Σχέδιο QA	Lean #1 (0-3 bottom-left)	Lean #2 (0-5 top-left)	Lean #3 (4-7 bottom-left)	Fat #1 (0-0 bottom-left)	Fat #2 (7-7 bottom-right)
Mη κβαντισμένο	91.6%: 7.86	92.5%: 7.94	92.8%: 8.72	94.2%: 75.86	95.2%: 107.08
DRQ	91.6%: $\times 1.00$	92.5%: $\times 1.00$	92.8%: $\times 1.00$	94.2%: $\times 2.58$	95.1%: $\times 1.77$
FIQ FLOAT32	86.1%: $\times 0.92$	89.1%: $\times 0.92$	89.3%: $\times 0.99$	92.4%: $\times 2.46$	93.8%: $\times 2.81$
FIQ INT8	77.0%: $\times 0.95$	78.0%: $\times 0.96$	80.9%: $\times 1.03$	79.4%: $\times 2.49$	81.6%: $\times 2.83$
FIQ UINT8	76.9%: $\times 0.91$	78.4%: $\times 0.92$	80.0%: $\times 0.99$	80.7%: $\times 2.46$	83.6%: $\times 2.81$

Πίνακας 5.13: Οι επιδόσεις των τελικών εκδοχών του LRF + pruned SmartWatch μοντέλου (3^η φάση), σε ακρίβεια πρόβλεψης. Σε κάθε περίπτωση παρατίθεται το αρχικό μέγευθος του μη κβαντισμένου μοντέλου (σε KB) και, στο εκάστοτε σχέδιο QA, παρατίθεται ο συντελεστής συμπίεσης.

Post-LRF BSPC		
Σχέδιο QA	Lean	Fat
Mη κβαντισμένο	86.9%: 84.80	88.7%: 113.96
DRQ	86.9%: $\times 2.25$	88.7%: $\times 2.52$
FIQ FLOAT32	84.6%: $\times 2.28$	88.6%: $\times 2.65$
FIQ INT8	82.7%: $\times 2.30$	88.2%: $\times 2.67$
FIQ UINT8	85.2%: $\times 2.28$	88.6%: $\times 2.65$

Πίνακας 5.14: Οι επιδόσεις των τελικών εκδοχών του LRF + pruned BSPC μοντέλου (3^η φάση), σε δυαδική ακρίβεια πρόβλεψης. Σε κάθε περίπτωση παρατίθεται το αρχικό μέγευθος του μη κβαντισμένου μοντέλου (σε KB) και, στο εκάστοτε σχέδιο QA, παρατίθεται ο συντελεστής συμπίεσης.

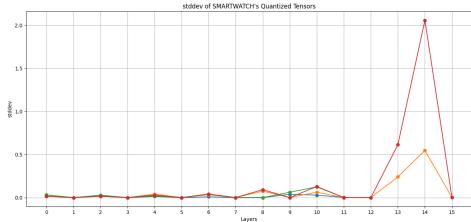
Post-LRF BSPC GAP		
Σχέδιο QA	Lean	Fat
Mη κβαντισμένο	86.5%: 252.72	88.3%: 270.54
DRQ	86.4%: $\times 3.01$	88.2%: $\times 3.01$
FIQ FLOAT32	85.5%: $\times 3.03$	88.3%: $\times 3.09$
FIQ INT8	85.5%: $\times 3.05$	88.3%: $\times 3.10$
FIQ UINT8	85.4%: $\times 3.03$	88.0%: $\times 3.09$

Πίνακας 5.15: Οι επιδόσεις των τελικών εκδοχών του LRF + pruned BSPC GAP μοντέλου (3^η φάση), σε δυαδική ακρίβεια πρόβλεψης. Σε κάθε περίπτωση παρατίθεται το αρχικό μέγεθος του μη κβαντισμένου μοντέλου (σε KB) και, στο εκάστοτε σχέδιο QA, παρατίθεται ο συντελεστής συμπίεσης.

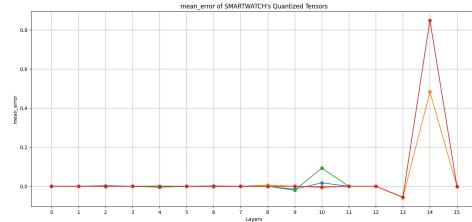
Στους πίνακες 5.13, 5.14 και 5.15 αναγράφονται αναλυτικά τα αποτελέσματα όλων των εκδοχών των τριών μοντέλων μετά από κάθε μέθοδο PTQ.

5.3.2 Αποσφαλμάτωση των Συνελικτικών Μοντέλων

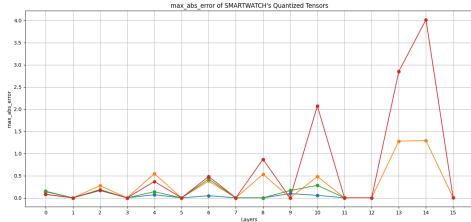
Στην τελευταία ενότητα αυτού του κεφαλαίου θα παρουσιαστούν τα αποτελέσματα του Quantization Debugger για τα τρία CNNs. Ο Αποσφαλματωτής διαθέτει μέθοδο για την εξαγωγή των μετρικών ενός μοντέλου σε αρχείο CSV. Για κάθε μοντέλο, τα CSV files των εκδοχών του ενοποιήθηκαν και οπτικοποιήθηκαν σε μορφή γραφήματος γραμμών με τη χρήση της βιβλιοθήκης Matplotlib. Ο κατακόρυφος άξονας του εκάστοτε γραφήματος αναφέρεται στην αντίστοιχη μετρική που παρήγαγε ο Αποσφαλματωτής για τα στρώματα, ενώ ο οριζόντιος άξονας αποτελεί δεικτοδότηση των στρωμάτων (δείκτης 0 → 1° στρώμα, δείκτης 1 → 2° στρώμα, κ.ο.κ.). Στις ακόλουθες σελίδες παρατίθενται αναλυτικά όλα τα σχετικά αποτελέσματα.



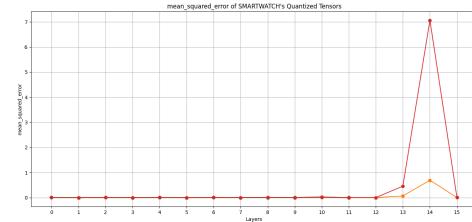
(α') Τυπική Απόκλιση



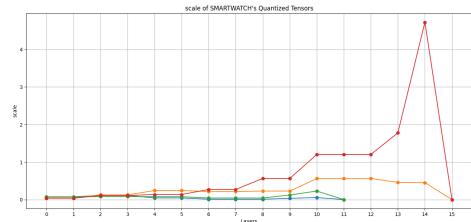
(β') Μέσο Σφάλμα



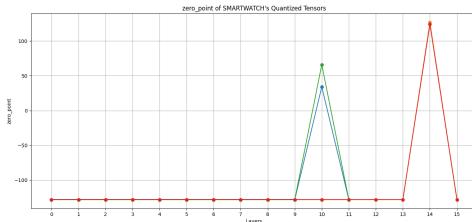
(γ') Μέγιστο Απόλυτο Σφάλμα



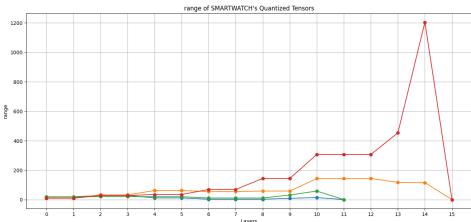
(δ') Μέσο Τετραγωνικό Σφάλμα (ΜΤΣ)



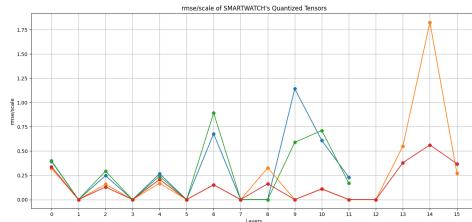
(ε') Συντελεστής Κλιμάκωσης, S



(ζ') Συντελεστής Μηδενικού Σημείου, Z



(ζ') Εμβέλεια, ίση με $255 \cdot S$



(η') Ρίζα του ΜΤΣ προς την κλιμάκωση

Σχήμα 5.7: Σπαστικές μετρικές των διαφορών μεταξύ των FIQ-χβαντισμένων και των αντίστοιχων μη χβαντισμένων στρωμάτων του μοντέλου SmartWatch (φάσεις 1 & 2), μαζί με τα S και Z . **ΠΡΑΣΙΝΟ:** Pre-Pruning Lean , **ΚΟΚΚΙΝΟ:** Pre-Pruning Fat , **ΜΠΛΕ:** Post-Pruning Lean , **ΠΟΡΤΟΚΑΛΙ:** Post-Pruning Fat.

op_name	
0	CONV_2D
1	MAX_POOL_2D
2	CONV_2D
3	MAX_POOL_2D
4	CONV_2D
5	MAX_POOL_2D
6	CONV_2D
7	MAX_POOL_2D
8	CONV_2D
9	MAX_POOL_2D
10	CONV_2D
11	MAX_POOL_2D
12	RESHAPE
13	FULLY_CONNECTED
14	FULLY_CONNECTED
15	SOFTMAX

op_name	
0	CONV_2D
1	MAX_POOL_2D
2	CONV_2D
3	MAX_POOL_2D
4	CONV_2D
5	MAX_POOL_2D
6	CONV_2D
7	MAX_POOL_2D
8	CONV_2D
9	MAX_POOL_2D
10	CONV_2D
11	MAX_POOL_2D
12	RESHAPE
13	FULLY_CONNECTED
14	FULLY_CONNECTED
15	SOFTMAX

(α') Pre-Pruning Lean

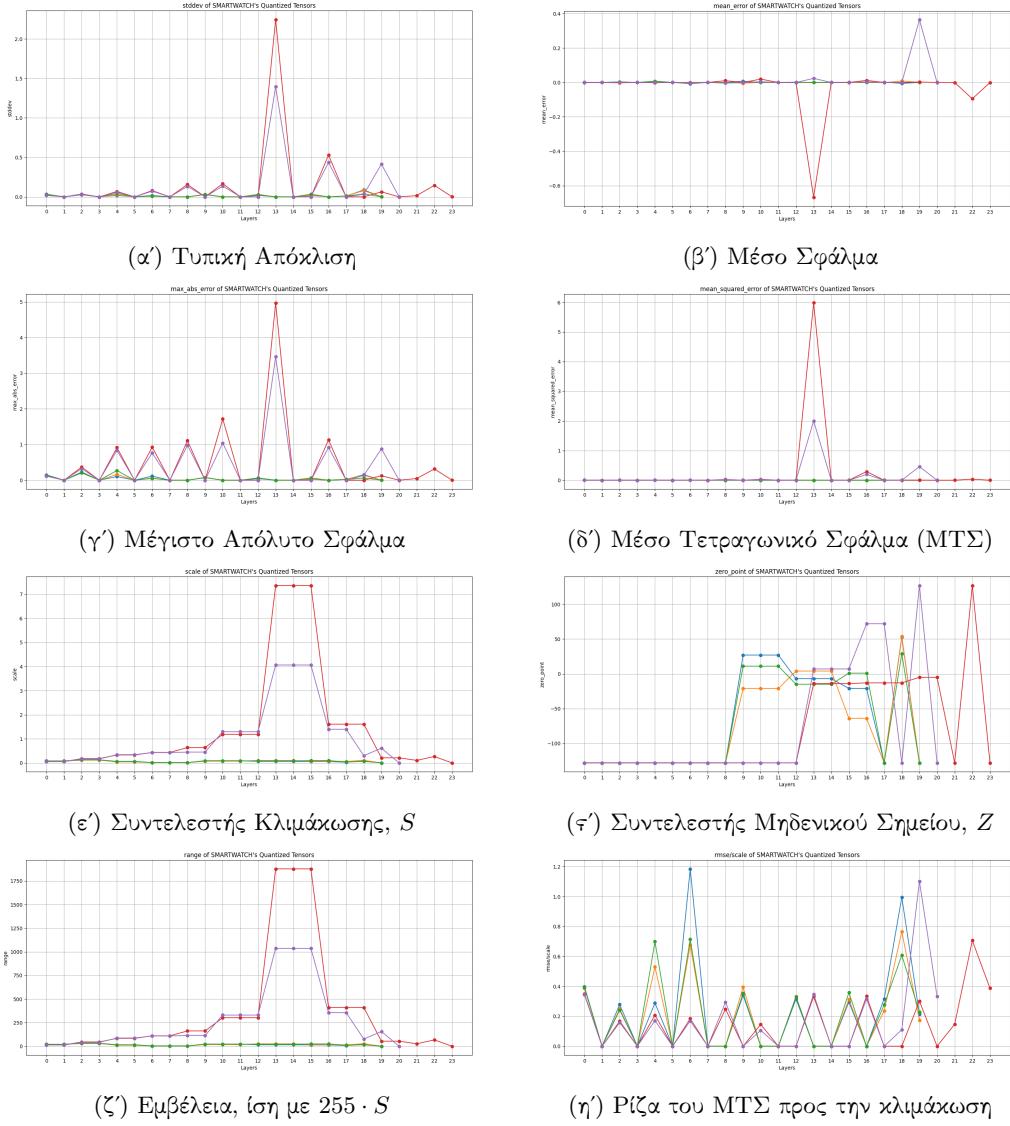
(β') Pre-Pruning Fat

(γ') Post-Pruning Lean

(δ') Post-Pruning Fat

Σχήμα 5.8: Αντιστοιχίες των δεικτών του οριζόντιου άξονα των γραφημάτων του σχήματος 5.7 στα στρώματα των pre-pruning (φάση 1) και post-pruning (φάση 2) εκδοχών του SmartWatch μοντέλου.

Από τα γραφήματα του σχήματος 5.7, φαίνεται ότι μοντέλο SmartWatch αποκλίνει περισσότερο προς τα τελευταία στρώματα του σε όλες τις pre-pruning και post-pruning εκδοχές του. Ιδιαίτερα, μεγαλύτερη απόκλιση παρατηρείται στο προτελευταίο στρώμα, δηλαδή στον πλήρως συνδεδεμένο τένσορα ακριβώς πριν την τελική συνάρτηση ενεργοποίησης, softmax. Εκεί η τυπική απόκλιση φτάνει στην τιμή 2, το μέσο σφάλμα 0.8, το μέγιστο απόλυτο σφάλμα 4 και το μέγιστο τετραγωνικό σφάλμα 7. Η αντιστοιχίη των στρωμάτων με τις τιμές του οριζόντιου άξονα των γραφημάτων μπορεί να γίνει με τη βοήθεια του σχήματος 5.8.



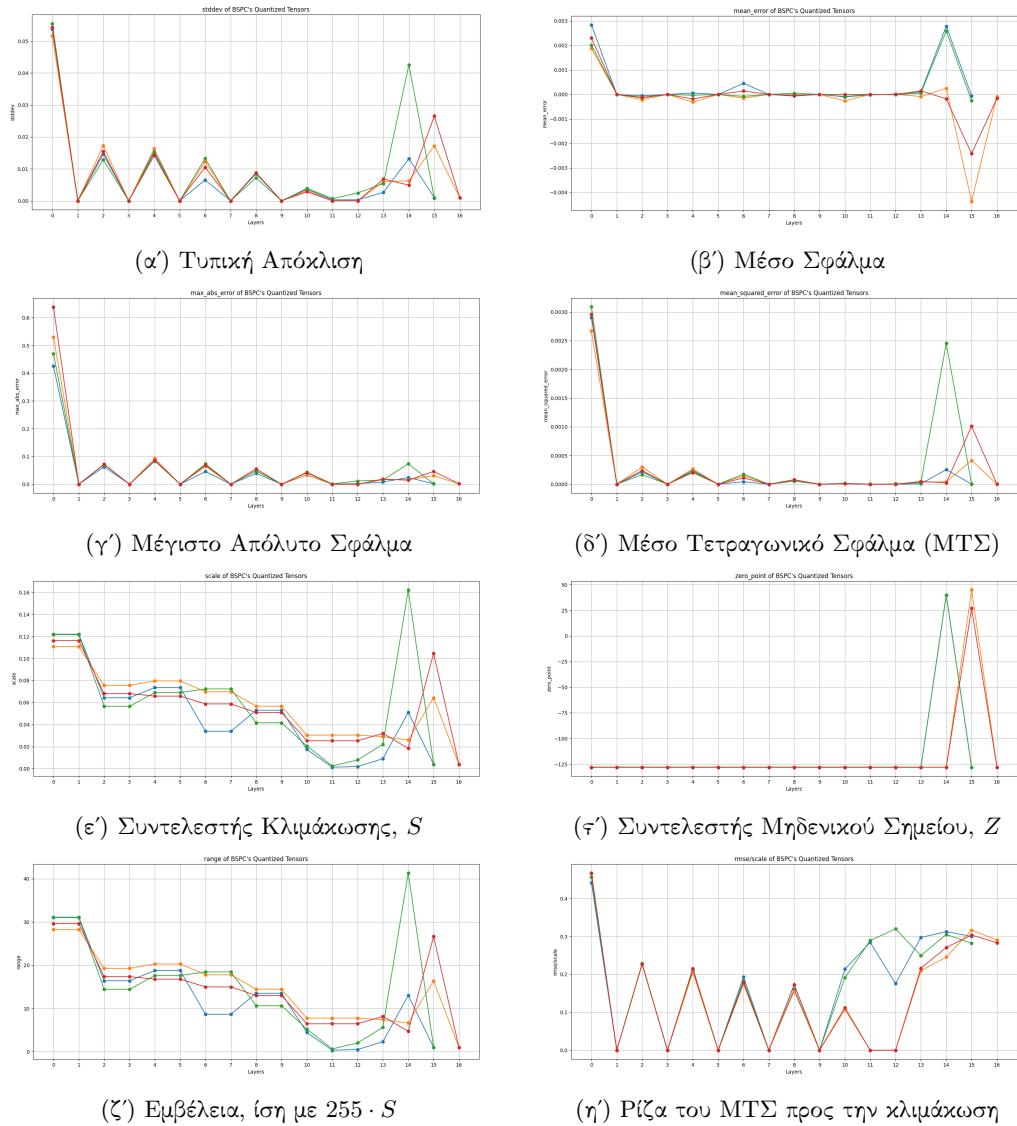
Σχήμα 5.9: Στατιστικές μετρικές των διαφορών μεταξύ των FIQ-κβαντισμένων και των αντίστοιχων μη κβαντισμένων στρωμάτων του μοντέλου SmartWatch, μαζί με τα S και Z .
ΜΠΑΛΕ: Lean #1 (0-3 bottom-left), **ΠΟΡΤΟΚΑΛΙ:** Lean #2 (0-5 top-left), **ΗΡΑΣΙΝΟ:** Lean #3 (4-7 bottom-left), **KOKKINO:** Fat #1 (0-0 bottom-left), **ΜΩΒ:** Fat #2 (7-7 bottom-right).

op_name	op_name	op_name	op_name	op_name
0 CONV_2D				
1 MAX_POOL_2D				
2 CONV_2D				
3 MAX_POOL_2D				
4 CONV_2D				
5 MAX_POOL_2D				
6 CONV_2D				
7 MAX_POOL_2D				
8 RESHAPE				
9 BATCH_MATMUL				
10 TRANSPOSE				
11 RESHAPE				
12 BATCH_MATMUL				
13 TRANSPOSE				
14 RESHAPE				
15 BATCH_MATMUL				
16 RESHAPE				
17 ADD				
18 FULLY_CONNECTED				
19 SOFTMAX				

(α') Final Lean#1 (β') Final Lean#2 (γ') Final Lean#3 (δ') Final Fat#1 (ε') Final Fat#2

Σχήμα 5.10: Αντιστοιχίες των δεικτών του οριζόντιου άξονα των γραφημάτων του σχήματος 5.9 στα στρώματα των Post-LRF (φάση 3) εκδοχών του SmartWatch μοντέλου.

Από τα γραφήματα του σχήματος 5.9, φαίνεται ότι οι τελικές fat εκδοχές του μοντέλου SmartWatch αποκλίνουν περισσότερο στα μεσαία στρώματα, ενώ οι lean είναι πολύ πιο ευσταθείς. Ιδιαίτερα, μεγαλύτερη απόκλιση παρατηρείται στο 13° στρώμα, δηλαδή στο στρώμα πολλαπλασιασμού δέσμης πινόχων ακριβώς μετά το τέλος της συνελικτικής αρχιτεκτονικής του μοντέλου. Εκεί η τυπική απόκλιση φτάνει στην τιμή ~ 2.5 , το μέσο σφάλμα 0.4, το μέγιστο απόλυτο σφάλμα 5 και το μέγιστο τετραγωνικό σφάλμα 6. Η αντιστοίχιση των στρώματων με τις τιμές του οριζόντιου άξονα των γραφημάτων μπορεί να γίνει με τη βοήθεια του σχήματος 5.10.



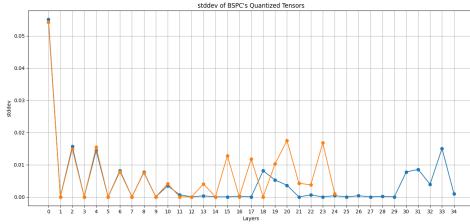
Σχήμα 5.11: Στατιστικές μετρικές των διαφορών μεταξύ των FIQ-κβαντισμένων και των αντίστοιχων μη κβαντισμένων στρωμάτων του μοντέλου BSPC (φάσεις 1 & 2), μαζί με τα S και Z . **ΠΡΑΣΙΝΟ**: Pre-Pruning Lean, **ΚΟΚΚΙΝΟ**: Pre-Pruning Fat, **ΜΠΑΕ**: Post-Pruning Lean, **ΠΟΡΤΟΚΑΛΙ**: Post-Pruning Fat.

op_name		op_name		op_name	
	0 CONV_2D		0 CONV_2D		0 CONV_2D
0	CONV_2D	1	MAX_POOL_2D	0	CONV_2D
1	MAX_POOL_2D	2	CONV_2D	1	MAX_POOL_2D
2	CONV_2D	3	MAX_POOL_2D	2	CONV_2D
3	MAX_POOL_2D	4	CONV_2D	3	MAX_POOL_2D
4	CONV_2D	5	MAX_POOL_2D	4	CONV_2D
5	MAX_POOL_2D	6	CONV_2D	5	MAX_POOL_2D
6	CONV_2D	7	MAX_POOL_2D	6	CONV_2D
7	MAX_POOL_2D	8	CONV_2D	7	MAX_POOL_2D
8	CONV_2D	9	MAX_POOL_2D	8	CONV_2D
9	MAX_POOL_2D	10	CONV_2D	9	MAX_POOL_2D
10	CONV_2D	11	MAX_POOL_2D	10	CONV_2D
11	MEAN	12	RESHAPE	11	MEAN
12	FULLY_CONNECTED	13	FULLY_CONNECTED	12	FULLY_CONNECTED
13	FULLY_CONNECTED	14	FULLY_CONNECTED	13	FULLY_CONNECTED
14	FULLY_CONNECTED	15	FULLY_CONNECTED	14	FULLY_CONNECTED
15	LOGISTIC	16	LOGISTIC	15	LOGISTIC
16				16	LOGISTIC

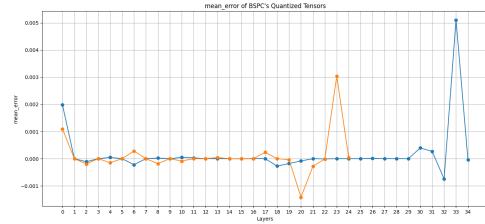
(α') Pre-Pruning Lean (β') Pre-Pruning Fat (γ') Post-Pruning Lean (δ') Post-Pruning Fat

Σχήμα 5.12: Αντιστοιχίες των δεικτών του οριζόντιου άξονα των γραφημάτων του σχήματος 5.11 στα στρώματα των pre-pruning (φάση 1) και post-pruning (φάση 2) εκδοχών του BSPC μοντέλου.

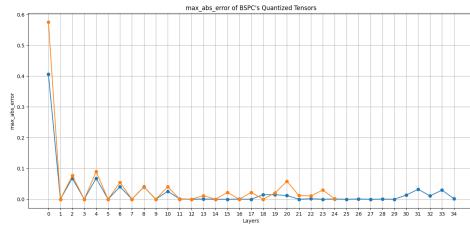
Σε αντίθεση με το μοντέλο SmartWatch, από τα γραφήματα του σχήματος 5.11, φαίνεται ότι οι τιμές των γραφημάτων του BSPC είναι τάξης μεγέθους μικρότερες από αυτών του SmartWatch, πράγμα που εξηγεί και την ευστάθεια που παρουσιάζει το πρώτο μοντέλο ως προς τις επιδόσεις σε σχέση με το δεύτερο όταν σε αυτά εφαρμόσουμε QA. Ιδιαίτερα, μεγαλύτερη απόκλιση παρατηρείται στο 13° στρώμα, δηλαδή στο στρώμα πολλαπλασιασμού δέσμης πινάκων ακριβώς μετά το τέλος της συνελικτικής αρχιτεκτονικής του μοντέλου. Η μέγιστη τυπική απόκλιση είναι 0.05 έως 0.055, το μέσο σφάλμα 0.02 με 0.03, το μέγιστο απόλυτο σφάλμα 0.4 με 0.6 και το μέγιστο τετραγωνικό σφάλμα 0.0025 με 0.003. Βέβαια, το μεγαλύτερο μέσο σφάλμα (κατά απόλυτη τιμή) του post-pruning fat BSPC εμφανίζεται στο 15° στρώμα του. Η αντιστοίχιση των στρωμάτων με τις τιμές του οριζόντιου άξονα των γραφημάτων μπορεί να γίνει με τη βοήθεια του σχήματος 5.12.



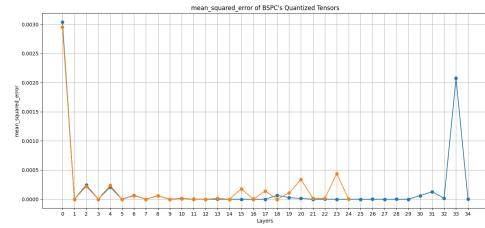
(α') Τυπική Απόκλιση



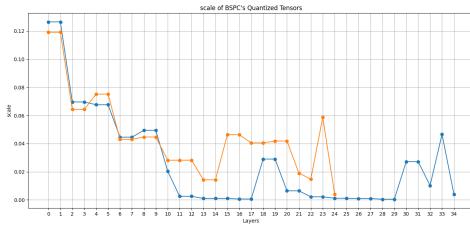
(β') Μέσο Σφάλμα



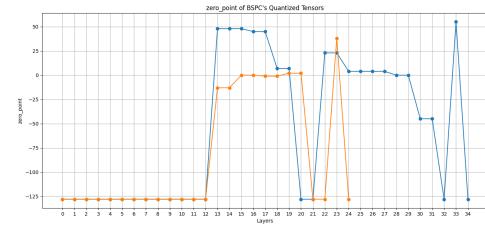
(γ') Μέγιστο Απόλυτο Σφάλμα



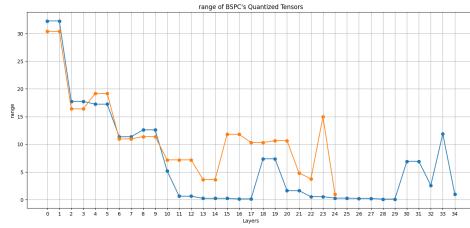
(δ') Μέσο Τετραγωνικό Σφάλμα (ΜΤΣ)



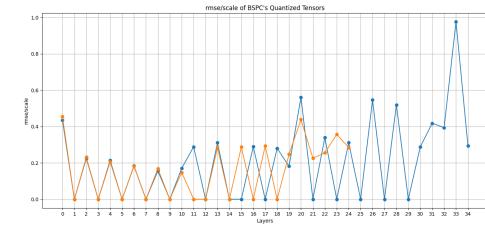
(ε') Συντελεστής Κλιμάκωσης, S



(τ') Συντελεστής Μηδενικού Σημείου, Z



(ζ') Εμβέλεια, ίση με $255 \cdot S$



(η') Ρίζα του ΜΤΣ προς την κλιμάκωση

Σχήμα 5.13: Στατιστικές μετρικές των διαφορών μεταξύ των FIQ-χβαντισμένων και των αντίστοιχων μη χβαντισμένων στρωμάτων του μοντέλου BSPC, μαζί με τα S και Z . **ΜΠΛΕ:** Lean (BSPC_seq_param100K_pruned_90) . **ΠΟΡΤΟΚΑΛΙ:** Fat (BSPC_seq_param4M_pruned_80).

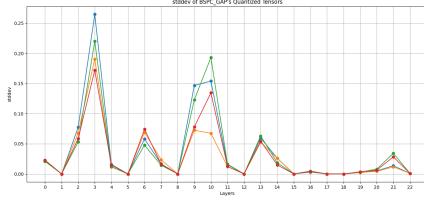
op_name	18	ADD	op_name	13	BATCH_MATMUL
0	CONV_2D	19	MUL	0	CONV_2D
1	MAX_POOL_2D	20	ADD	1	MAX_POOL_2D
2	CONV_2D	21	RESHAPE	2	CONV_2D
3	MAX_POOL_2D	22	BATCH_MATMUL	3	MAX_POOL_2D
4	CONV_2D	23	RESHAPE	4	CONV_2D
5	MAX_POOL_2D	24	BATCH_MATMUL	5	MAX_POOL_2D
6	CONV_2D	25	RESHAPE	6	CONV_2D
7	MAX_POOL_2D	26	BATCH_MATMUL	7	MAX_POOL_2D
8	CONV_2D	27	RESHAPE	8	CONV_2D
9	MAX_POOL_2D	28	BATCH_MATMUL	9	MAX_POOL_2D
10	CONV_2D	29	RESHAPE	10	CONV_2D
11	MEAN	30	ADD	11	MAX_POOL_2D
12	RESHAPE	31	MUL	12	CONV_2D
13	BATCH_MATMUL	32	ADD	13	MAX_POOL_2D
14	TRANSPOSE	33	FULLY_CONNECTED	14	CONV_2D
15	RESHAPE	34	LOGISTIC	15	MAX_POOL_2D
16	BATCH_MATMUL			16	RESHAPE
17	RESHAPE			17	BATCH_MATMUL

(α') Post-LRF Lean

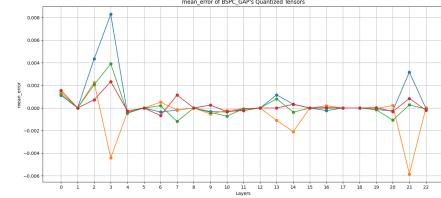
(β') Post-LRF Fat

Σχήμα 5.14: Αντιστοιχίες των δεικτών του οριζόντιου άξονα των γραφημάτων του σχήματος 5.13 στα στρώματα των post-LRF (φάση 3) εκδοχών του BSPC μοντέλου.

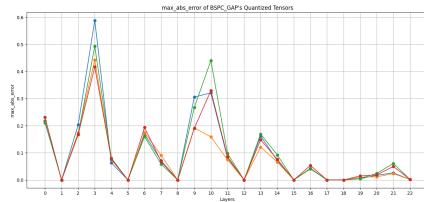
Οι τελικές εκδοχές του BSPC μοντέλου εμφανίζουν την ίδια ευστάθεια σε σχέση με το μοντέλο SmartWatch. Οι μέγιστες τιμές των στατιστικών μετρικών σφάλματος των γραφημάτων του σχήματος 5.13 συνεχίζουν να είναι μικρότερες από τις αντιστοιχίες των γραφημάτων του SmartWatch. Η αντιστοίχιση των στρωμάτων με τις τιμές του οριζόντιου άξονα των γραφημάτων μπορεί να γίνει με τη βοήθεια του σχήματος 5.14.



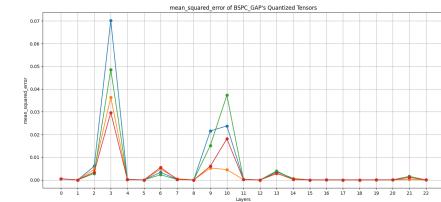
(α') Τυπική Απόκλιση



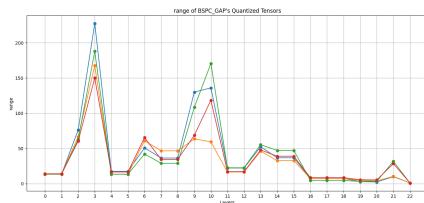
(β') Μέσο Σφάλμα



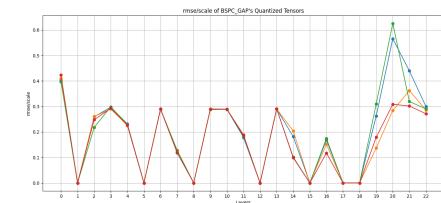
(γ') Μέγιστο Απόλυτο Σφάλμα



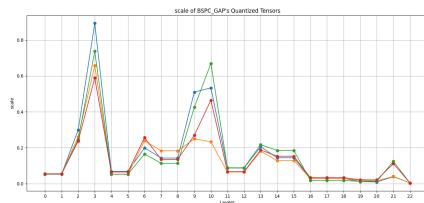
(δ') Μέσο Τετραγωνικό Σφάλμα (ΜΤΣ)



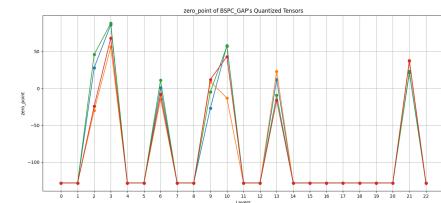
(ε') Εμβέλεια, ίση με $255 \cdot S$



(τ') Ρίζα του ΜΤΣ προς την κλιμάκωση



(ζ') Συντελεστής Κλιμάκωσης, S



(η') Συντελεστής Μηδενικού Σημείου, Z

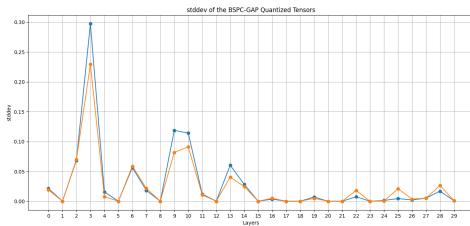
Σχήμα 5.15: Στατιστικές μετρικές των διαφορών μεταξύ των FIQ-κβαντισμένων και των αντίστοιχων μη κβαντισμένων στρωμάτων του μοντέλου BSPC GAP (φάσεις 1 & 2), μαζί με τα S και Z . **ΠΙΡΑΣΙΝΟ:** Pre-Pruning Lean , **ΚΟΚΚΙΝΟ:** Pre-Pruning Fat , **ΜΠΑΕ:** Post-Pruning Lean , **ΠΟΡΤΟΚΑΛΙ:** Post-Pruning Fat.

op_name	op_name	op_name	op_name
0 CONV_2D	0 CONV_2D	0 CONV_2D	0 CONV_2D
1 MAX_POOL_2D	1 MAX_POOL_2D	1 MAX_POOL_2D	1 MAX_POOL_2D
2 CONV_2D	2 CONV_2D	2 CONV_2D	2 CONV_2D
3 CONV_2D	3 CONV_2D	3 CONV_2D	3 CONV_2D
4 CONV_2D	4 CONV_2D	4 CONV_2D	4 CONV_2D
5 MAX_POOL_2D	5 MAX_POOL_2D	5 MAX_POOL_2D	5 MAX_POOL_2D
6 CONV_2D	6 CONV_2D	6 CONV_2D	6 CONV_2D
7 ADD	7 ADD	7 ADD	7 ADD
8 MAX_POOL_2D	8 MAX_POOL_2D	8 MAX_POOL_2D	8 MAX_POOL_2D
9 CONV_2D	9 CONV_2D	9 CONV_2D	9 CONV_2D
10 CONV_2D	10 CONV_2D	10 CONV_2D	10 CONV_2D
11 CONV_2D	11 CONV_2D	11 CONV_2D	11 CONV_2D
12 MAX_POOL_2D	12 MAX_POOL_2D	12 MAX_POOL_2D	12 MAX_POOL_2D
13 CONV_2D	13 CONV_2D	13 CONV_2D	13 CONV_2D
14 ADD	14 ADD	14 ADD	14 ADD
15 MAX_POOL_2D	15 MAX_POOL_2D	15 MAX_POOL_2D	15 MAX_POOL_2D
16 CONV_2D	16 CONV_2D	16 CONV_2D	16 CONV_2D
17 MAX_POOL_2D	17 MAX_POOL_2D	17 MAX_POOL_2D	17 MAX_POOL_2D
18 RESHAPE	18 RESHAPE	18 RESHAPE	18 RESHAPE
19 FULLY_CONNECTED	19 FULLY_CONNECTED	19 FULLY_CONNECTED	19 FULLY_CONNECTED
20 FULLY_CONNECTED	20 FULLY_CONNECTED	20 FULLY_CONNECTED	20 FULLY_CONNECTED
21 FULLY_CONNECTED	21 FULLY_CONNECTED	21 FULLY_CONNECTED	21 FULLY_CONNECTED
22 LOGISTIC	22 LOGISTIC	22 LOGISTIC	22 LOGISTIC

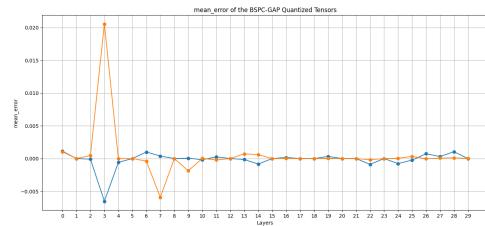
(α') Pre-Pruning Lean (β') Pre-Pruning Fat (γ') Post-Pruning Lean (δ') Post-Pruning Fat

Σχήμα 5.16: Αντιστοιχίες των δεικτών του οριζόντιου άξονα των γραφημάτων του σχήματος 5.11 στα στρώματα των pre-pruning (φάση 1) και post-pruning (φάση 2) εκδοχών του BSPC GAP μοντέλου.

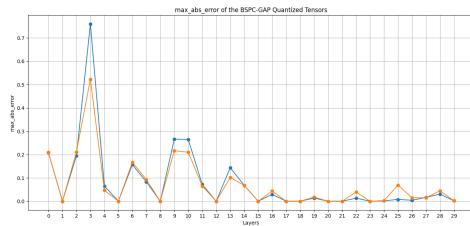
Οι pre-pruning και post-pruning εκδοχές του BSPC GAP μοντέλου εμφανίζουν την ίδια ευστάθεια σε σχέση με το μοντέλο SmartWatch. Οι μέγιστες τιμές των στατιστικών μετρικών σφάλματος των γραφημάτων του σχήματος 5.15 συνεχίζουν να είναι μικρότερες από τις αντίστοιχες των γραφημάτων του SmartWatch. Όμως, οι μετρικές ταλαντώνονται πολύ περισσότερο κατά μήκος των στρωμάτων απ' ότι του BSPC GAP και άρα θα ήταν παραπλανητικό να θίξουμε τα στρώματα που παρουσιάζουν τη μέγιστη απόκλιση. Η αντιστοίχιση των στρωμάτων με τις τιμές του οριζόντιου άξονα των γραφημάτων μπορεί να γίνει με τη βοήθεια του σχήματος 5.16.



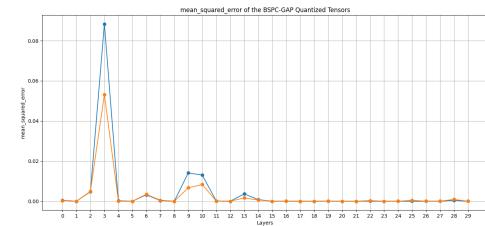
(α') Τυπική Απόκλιση



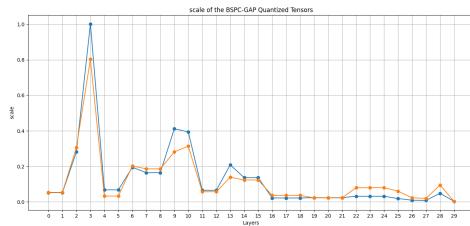
(β') Μέσο Σφάλμα



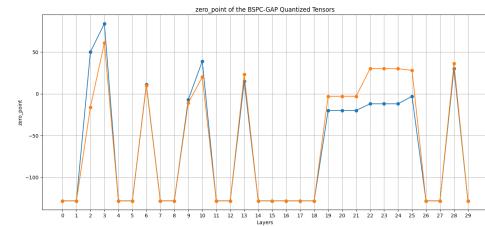
(γ') Μέγιστο Απόλυτο Σφάλμα



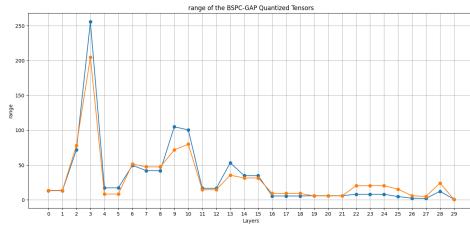
(δ') Μέσο Τετραγωνικό Σφάλμα (ΜΤΣ)



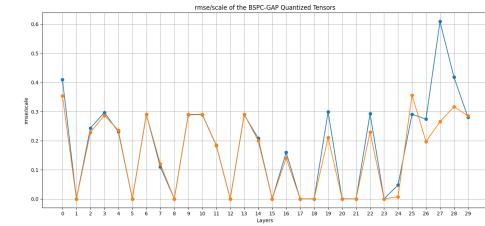
(ε') Συντελεστής Κλιμάκωσης, S



(ζ') Συντελεστής Μηδενικού Σημείου, Z



(ζ') Εμβέλεια, ίση με $255 \cdot S$



(η') Ρίζα του ΜΤΣ προς την κλιμάκωση

Σχήμα 5.17: Στατιστικές μετρικές των διαφορών μεταξύ των FIQ-χβαντισμένων και των αντίστοιχων μη χβαντισμένων στρωμάτων του μοντέλου BSPC GAP, μαζί με τα S και Z . **ΜΠΑΙΕ:** Lean (BSPC_seq_param100K_pruned_90), **ΠΟΡΤΟΚΑΛΙ:** Fat (BSPC_seq_param4M_pruned_80).

op_name	15	MAX_POOL_2D	op_name	15	MAX_POOL_2D
0 CONV_2D	16	CONV_2D	0 CONV_2D	16	CONV_2D
1 MAX_POOL_2D	17	MAX_POOL_2D	1 MAX_POOL_2D	17	MAX_POOL_2D
2 CONV_2D	18	RESHAPE	2 CONV_2D	18	RESHAPE
3 CONV_2D	19	BATCH_MATMUL	3 CONV_2D	19	BATCH_MATMUL
4 CONV_2D	20	TRANSPOSE	4 CONV_2D	20	TRANSPOSE
5 MAX_POOL_2D	21	RESHAPE	5 MAX_POOL_2D	21	RESHAPE
6 CONV_2D	22	BATCH_MATMUL	6 CONV_2D	22	BATCH_MATMUL
7 ADD	23	RESHAPE	7 ADD	23	RESHAPE
8 MAX_POOL_2D	24	ADD	8 MAX_POOL_2D	24	ADD
9 CONV_2D	25	MUL	9 CONV_2D	25	MUL
10 CONV_2D	26	ADD	10 CONV_2D	26	ADD
11 CONV_2D	27	FULLY_CONNECTED	11 CONV_2D	27	FULLY_CONNECTED
12 MAX_POOL_2D	28	FULLY_CONNECTED	12 MAX_POOL_2D	28	FULLY_CONNECTED
13 CONV_2D	29	LOGISTIC	13 CONV_2D	29	LOGISTIC
14 ADD			14 ADD		

(α') Post-LRF Lean

(β') Post-LRF Fat

Σχήμα 5.18: Αντιστοιχίες των δεικτών του οριζόντιου άξονα των γραφημάτων του σχήματος 5.13 στα στρώματα των post-LRF (φάση 3) εκδοχών του BSPC GAP μοντέλου.

Οι τελικές εκδοχές του BSPC GAP μοντέλου είναι επίσης ευσταθείς όπως φαίνεται από τα γραφήματα του σχήματος 5.17 με τον ίδιο τρόπο που ήταν και τα προηγούμενα BSPC και BSPC GAP μοντέλα. Η αντιστοίχιση των στρωμάτων με τις τιμές του οριζόντιου άξονα των γραφημάτων μπορεί να γίνει με τη βοήθεια του σχήματος 5.18.

Κεφάλαιο 6

Συμπεράσματα και Προτάσεις για Μελλοντική Έρευνα

Στην παρούσα εργασία παρουσιάστηκε μια διεξοδική ανάλυση όλων των στρατηγικών και μεθόδων QA και πραγματοποιήθηκαν σχετικά, εξονυχιστικά πειράματα για την εκτίμηση της αποδοτικότητας αυτών σε διάφορα μοντέλα ανάλυσης χρονοσειρών. Αρχικά, μελετήθηκε η επίδραση των PTQ και QAT στις επιδόσεις και στη συμπίεση απλών μοντέλων LSTM για την ανάλυση φυσικής γλώσσας. Η συμπίεση των εν λόγω DNNs με QA αποδείχτηκε ιδιαίτερα προβληματική και δύσκολη λόγω της έλλειψης υποστήριξης QAT για LSTM στρώματα και Embedding στρώματα και, κυρίως, λόγω της δυσκολίας εφαρμογής QA στα Embedding στρώματα.

Το επόμενο βήμα ήταν η διεξαγωγή πειραμάτων στο έτοιμο συνελικτικό μοντέλο Ordenez 2016 Deep Original που χρησιμοποιείται για την επεξεργασία HAR δεδομένων. Σε αυτή τη φάση, μελετήθηκε ο αντίκτυπος πληθώρας μεθόδων PTQ όπως DRQ, FIQ FLOAT32, FIQ INT8 και FIQ UINT8 σε πολύ περισσότερες μετρικές, καθώς και στο μέγεθος του τελικού TFLite μοντέλου. Κατά την ανάλυση του Ordenez model έγινε και μια εισαγωγή σε δύο πολύ σημαντικά πειραματικά εργαλεία, στον Αναλυτή TFLite (`tf.lite.experimental.Analyzer`) και στον Αποσφαλματωτή Κβαντισμού (`tf.lite.experimental.QuantizationDebugger`). Ο Αναλυτής συνείσφερε σημαντικά στον αναλυτικό προσδιορισμό των τενσόρων του τελικού TFLite μοντέλου (π.χ. μέγεθος, τύπος δεδομένων), πράγμα που συνείσφερε και στην εμβάθυνση των εσωτερικών μηχανισμών της TensorFlow και πως αυτή μετατρέπει τα Keras μοντέλα σε TFLite, καθώς και στη λεπτομερή οπτικοποίηση του δομικού αντίκτυπου του κβαντισμού στην αρχιτεκτονική αυτών. Ταυτοχρόνως, ο Αποσφαλματωτής χρησιμοποιήθηκε για την αναλυτική επιμερηση της λειτουργικότητας των τελικών, συμπιεσμένων μοντέλων σε σύγχριση με τα αρχικά μοντέλα. Έγινε μια επίδειξη της χρήσης του για τον εντοπισμό την απομόνωση των προβληματικών στρωμάτων και πως αυτό επηρεάζει την εκ νέου συμπίεση και τις εκ νέου επιδόσεις των μερικώς κβαντισμένων μοντέλων. Για το μοντέλο Ordenez, που είναι συνελικτικό και έπειτα αναδρομικό, αποδείχτηκε ότι η καλύτερη μέθοδος συμπίεσης είναι η DRQ,

ακολουθούμενη από την FIQ FLOAT32, ενώ ο μερικός κβαντισμός του μοντέλου για το FIQ INT8 σενάριο αποδείχτηκε ότι δεν αξίζει για τις δεδομένες συνθήκες.

Στο τελευταίο μέρος της εργασίας, μελετήθηκαν τα τρία CNNs που χρησιμοποιούνται, επίσης, σε εφαρμογές HAR, τα οποία κατείχαν τον πρωταγωνιστικό ρόλο σε αυτή την εργασία. Εξετάστηκαν τα μοντέλα σε τρεις φάσεις, δηλαδή κατά την αρχική τους κατάσταση, κατά τη φάση που συμπιέστηκαν με κλάδεμα παραμέτρων και κατά τη φάση που συμπιέστηκαν με κλάδεμα παραμέτρων + LRF. Κάθε φάση αποτελούνταν από lean και fat μοντέλα. Μετά τη συμπίεση τους με τη χρήση DRQ, FIQ FLOAT32, FIQ INT8 και FIQ UINT8 εκτιμήθηκαν οι επιδόσεις τους και ο βαθύς συμπίεσης τους. Παρατηρήθηκε ότι όλα τα μοντέλα ανταποκρίνονται πολύ θετικά στη μέθοδο DRQ, όμως τα BSPC και BSPC GAP παρουσιάζουν μικρότερη μεταβολή στις επιδόσεις τους μετά την εφαρμογή μεθόδων FIQ σε αυτά. Μια άλλη σημαντική παρατήρηση είναι ότι η χρήση της στρατηγικής QAT δε συνοδεύεται πάντα με αύξηση της απόδοσης στα κβαντισμένα μοντέλα, αλλά ήταν αφέλιμη σε συγκεκριμένα σενάρια όπως σε όλες τις εκδηλώσεις του FIQ FLOAT32 συμπιεσμένου SmartWatch ή σε όλα τα σχέδια κβαντισμού QA του pre-pruned lean BSPC μοντέλου. Τέλος, ο Αποσφαλματωτής αποδείχτηκε ισχυρό εργαλείο για την ερμηνεία των αποτελεσμάτων μέσω της οπτικοποίησης των μετρικών σφάλματος ανά στρώμα καθώς και στην ανάγνωση των κβαντιστικών τους παραμέτρων S και Z .

6.1 Προτάσεις για Μελλοντική Έρευνα

Πρόταση για μελλοντική έρευνα είναι η υλοποίηση ενός προγραμματιστικού πλαισίου για τον κβαντισμό περισσότερων και πιο εξελιγμένων δικτύων με βαθύτερη έμφαση στα δίκτυα αναδρομικής αρχιτεκτονικής. Συγκεκριμένα, τα RNNs που εξετάστηκαν στην εργασία είναι τα αρχικά γλωσσικά μοντέλα και το μοντέλο Ordonez. Δεδομένου των δυσκολιών που αντιμετωπίστηκαν ως προς το QAT, θα ήταν πολύ χρήσιμο να μελετηθεί περισσότερο η διεπαφή QuantizeConfig για τη δημιουργία κατάλληλων custom quantizers για LSTM και Embedded στρώματα καθώς αυτά δεν υποστηρίζονται από την TensorFlow 2.15.0. Ιδιαίτερα, τα Embedded στρώματα παρουσιάζουν προβλήματα ακόμα και στο PTQ. Δεδομένου ότι α) αποτελούν έναν τύπο στρώματος ζωτικής σημασίας για την επεξεργασία φυσικής γλώσσας και β) τα μεγάλα γλωσσικά μοντέλα έχουν κάνει θραύση στην αγορά αλλά και σε εφαρμογές ανοιχτού λογισμικού, είναι σημαντικό να μελετηθούν ακόμα περισσότερο η συμπίεση με QA των Embedded στρώματων. Τέλος, η οπτικοποίηση των αποτελεσμάτων του Αποσφαλματωτή χρησιμοποιήθηκεν για την εξαγωγή συμπερασμάτων με ευριστικούς τρόπους. Επομένως θα ήταν σωστό να μελετηθούν ακόμα περισσότερο οι μετρικές που εξάγονται από τον Quantization Debugger και η θεμελίωση ενός μαθηματικού μοντέλου για την εξαγωγή βέλτιστων και ορθολογικών συμπερασμάτων για τις συνέπειες του QA στα DNNs και στα μοντέλα χρονοσειρών.

Βιβλιογραφία

- [1] Zahra Karevan and Johan AK Suykens. Transductive lstm for time-series prediction: An application to weather forecasting. *Neural Networks*, 125:1–9, 2020.
- [2] Janet M Box-Steffensmeier, John R Freeman, Matthew P Hitt, and Jon CW Pevehouse. *Time series analysis for the social sciences*. Cambridge University Press, 2014.
- [3] Francis EH Tay and Lijuan Cao. Application of support vector machines in financial time series forecasting. *omega*, 29(4):309–317, 2001.
- [4] Christine Lim and Michael McAleer. Time series forecasts of international travel demand for australia. *Tourism management*, 23(4):389–396, 2002.
- [5] JS Owen, BJ Eccles, BS Choo, and MA Woodings. The application of auto-regressive time series modelling for the time-frequency analysis of civil engineering structures. *Engineering Structures*, 23(5):521–536, 2001.
- [6] John Cristian Borges Gamboa. Deep learning for time-series analysis. *arXiv preprint arXiv:1701.01887*, 2017.
- [7] Hasan Hejbari Zargar, Saha Hejbari Zargar, and Raziye Mehri. Review of deep learning in healthcare. *arXiv preprint arXiv:2310.00727*, 2023.
- [8] Faisal Mohammad, Ki Boem Lee, and Young-Chon Kim. Short term load forecasting using deep neural networks. *arXiv preprint arXiv:1811.03242*, 2018.
- [9] Jessica Vamathevan, Dominic Clark, Paul Czodrowski, Ian Dunham, Edgardo Ferran, George Lee, Bin Li, Anant Madabhushi, Parantu Shah, Michaela Spitzer, et al. Applications of machine learning in drug discovery and development. *Nature reviews Drug discovery*, 18(6):463–477, 2019.
- [10] Marko Kolanovic and Rajesh Krishnamachari. Big data and ai strategies: Machine learning and alternative data approach to investing. *JP Morgan Global Quantitative & Derivatives Strategy Report*, 25, 2017.
- [11] Teng Liu, Xingyu Mu, Xiaolin Tang, Bing Huang, Hong Wang, and Dongpu Cao. Dueling deep q network for highway decision making in autonomous vehicles: A case study. *arXiv preprint arXiv:2007.08343*, 2020.
- [12] Matus Telgarsky. Benefits of depth in neural networks. In *Conference on learning theory*, pages 1517–1539. PMLR, 2016.

- [13] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.
- [14] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713, 2018.
- [15] Qinyao He, He Wen, Shuchang Zhou, Yuxin Wu, Cong Yao, Xinyu Zhou, and Yuheng Zou. Effective quantization methods for recurrent neural networks. *arXiv preprint arXiv:1611.10176*, 2016.
- [16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: A system for Large-Scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, Savannah, GA, November 2016. USENIX Association.
- [17] Lutz Roeder. Netron app. <https://netron.app/>. Accessed: 2024-05-06.
- [18] Official TensorFlow Guide. Tensorflow lite model analyzer. https://www.tensorflow.org/lite/guide/model_analyzer. Accessed: 2024-05-06.
- [19] Official TensorFlow Guide. Inspecting quantization errors with quantization debugger. https://www.tensorflow.org/lite/performance/quantization_debugger. Accessed: 2024-05-06.
- [20] Ratnadip Adhikari and Ramesh K Agrawal. An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*, 2013.
- [21] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, Puneet Agarwal, et al. Long short term memory networks for anomaly detection in time series. In *Esann*, volume 2015, page 89, 2015.
- [22] Andreas Eckner. A framework for the analysis of unevenly spaced time series data. *Preprint. Available at: http://www.eckner.com/papers/unevenly_spaced_time_series_analysis*, page 93, 2012.
- [23] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al.

- Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.
- [24] Keras 3 API documentation. IMDB movie review sentiment classification dataset. <https://keras.io/api/datasets/imdb/>. Accessed: 2024-05-08.
 - [25] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.
 - [26] John H Cochrane. Time series for macroeconomics and finance, 1997.
 - [27] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943.
 - [28] Andrej Krenker, Janez Bešter, and Andrej Kos. Introduction to the artificial neural networks. *Artificial Neural Networks: Methodological Advances and Biomedical Applications*. InTech, pages 1–18, 2011.
 - [29] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
 - [30] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.
 - [31] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasamine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
 - [32] Wikipedia. ARM Cortex-M — Wikipedia, the free encyclopedia. <http://en.wikipedia.org/w/index.php?title=ARM%20Cortex-M&oldid=1217668369>, 2024. [Online; accessed 11-May-2024].
 - [33] Greg Henry, Ping Tak Peter Tang, and Alexander Heinecke. Leveraging the bfloat16 artificial intelligence datatype for higher-precision computations. In *2019 IEEE 26th Symposium on Computer Arithmetic (ARITH)*, pages 69–76. IEEE, 2019.
 - [34] Apple Machine Learning Research. Deploying transformers on the apple neural engine. <https://machinelearning.apple.com/research/neural-engine-transformers>. Accessed: 2024-05-12.

- [35] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
- [36] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.
- [37] C. Foucart. Understanding the tflite quantization approach (part i). *CLIKa Compressed Blog*, January 6 2024. Accessed: 2024-06-27 from <https://blog.clika.io/understanding-the-tensorflow-lite-quantization-approach-part-i/>.
- [38] Luis Antonio Vasquez. Zero-point quantization : How do we get those formulas? *Medium*, Jan 2024. Accessed: 2024-06-27 from <https://medium.com/@luis.vasquez.work.log/zero-point-quantization-how-do-we-get-those-formulas-4155b51a60d6>.
- [39] TensorFlow. Post-training quantization: for Mobile & Edge. *Official TensorFlow Guide*. Accessed: 2024-06-28 from https://www.tensorflow.org/lite/performance/post_training_quantization.
- [40] TensorFlow. Quantization aware training: Tensorflow Model Optimization. *Official TensorFlow Guide*. Accessed: 2024-06-28 from https://www.tensorflow.org/model_optimization/guide/quantization/training.
- [41] Dimitrios Gkountelos, Milad Kokhazadeh, Charalampos Bournas, Georgios Keramidas, and Vasilios Kelefouras. Towards highly compressed cnn models for human activity recognition in wearable devices. In *2023 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pages 83–88. IEEE, 2023.
- [42] Atul Anand, Tushar Kadian, Manu Kumar Shetty, and Anubha Gupta. Explainable ai decision model for ecg data of cardiac disorders. *Biomedical Signal Processing and Control*, 75:103584, 2022.