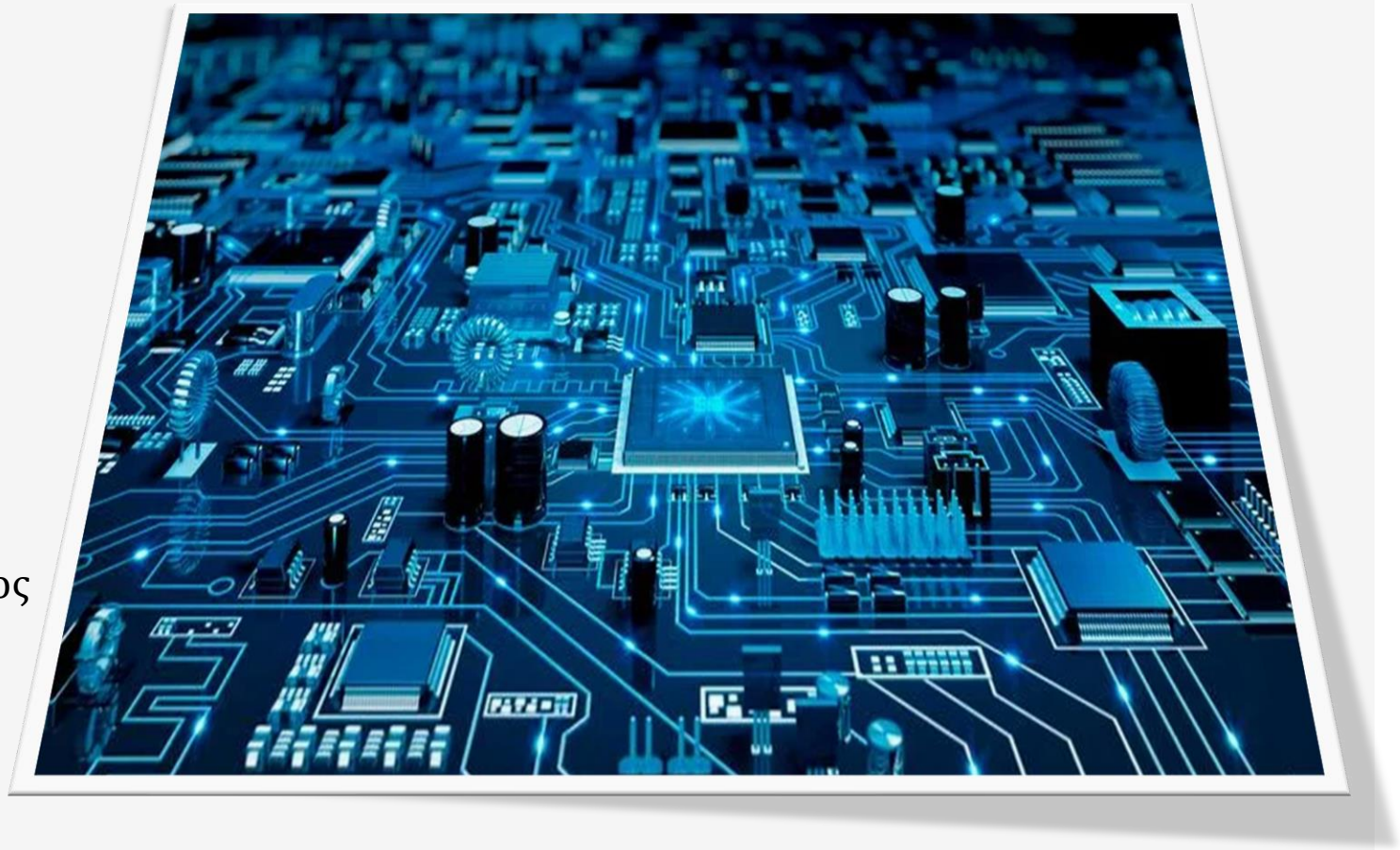
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΠΟΛΟΓΙΣΤΩΝ

HARDWARE ACCELERATION OF AI/ML ALGORITHM FOR FAULT DETECTION IN POWER DISTRIBUTION GRID

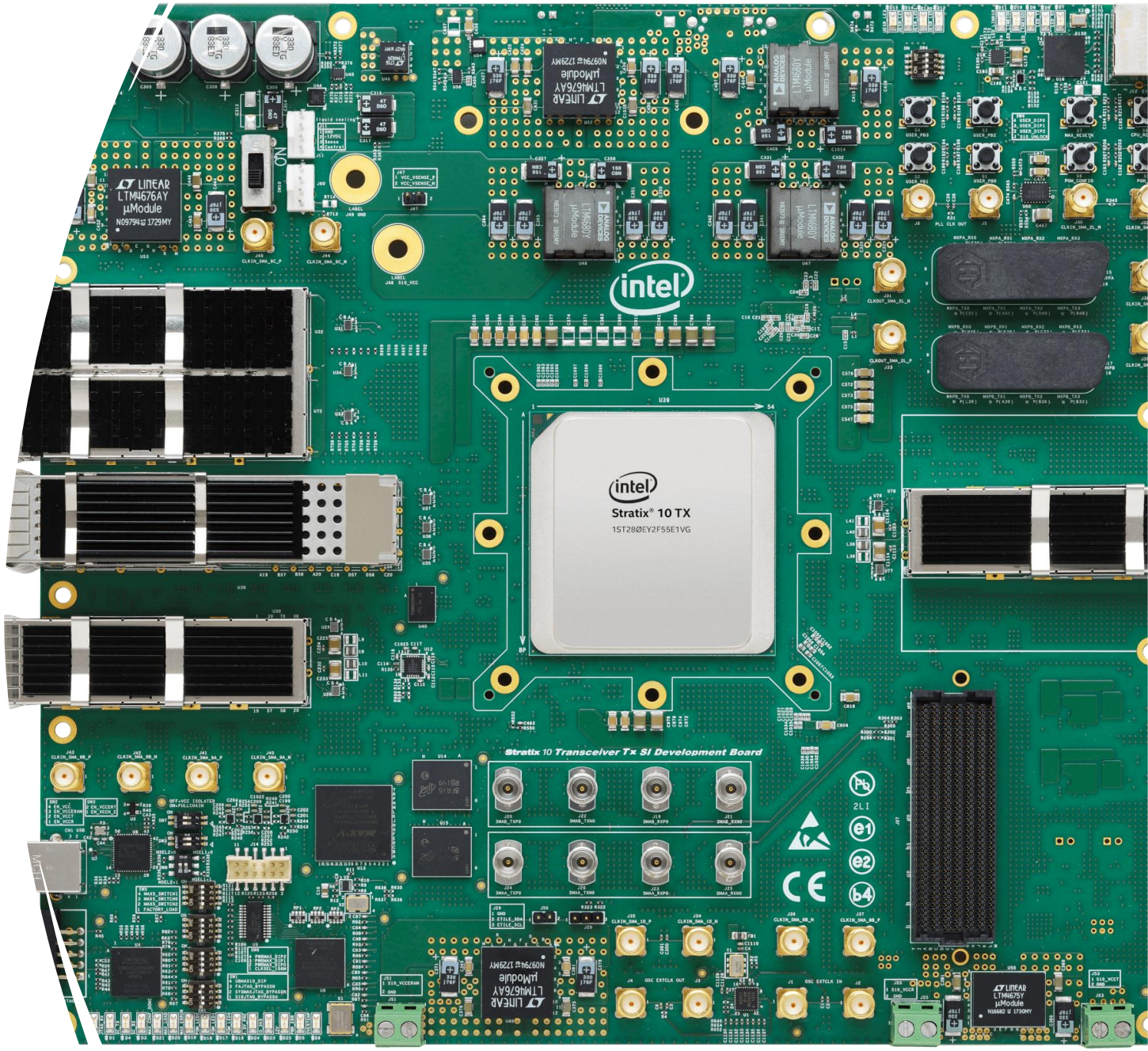


Διπλωματική Εργασία: Κουκουλάς Αλέξανδρος

Επιβλέπων: Μιχαήλ Μπίρμπας

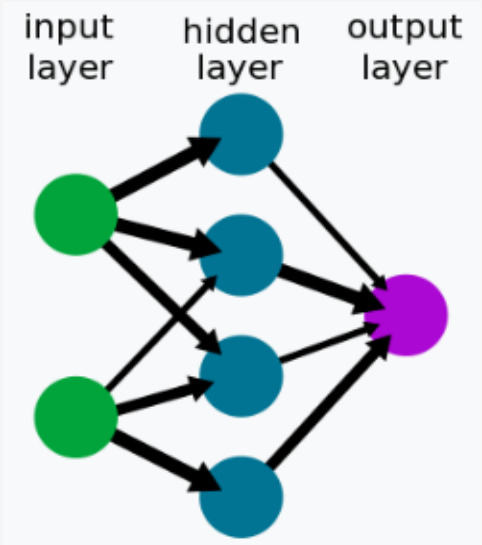
ΕΙΣΑΓΩΓΗ

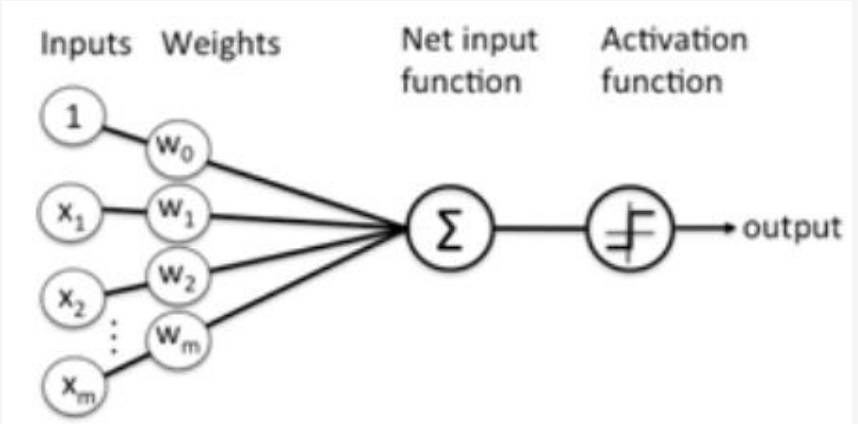
- Ο κύριος σκοπός είναι η υλοποίηση του αλγορίθμου προσδιορισμού θέσης σφάλματος και ταξινόμησης τύπου (FLITC) σε συσκευές υλικού για την αύξηση της ταχύτητας εκτέλεσης, έχοντας, παράλληλα, μικρή απώλεια στην ακρίβεια σωστών προβλέψεων.



Μηχανική Μάθηση – Νευρωνικά Δίκτυα

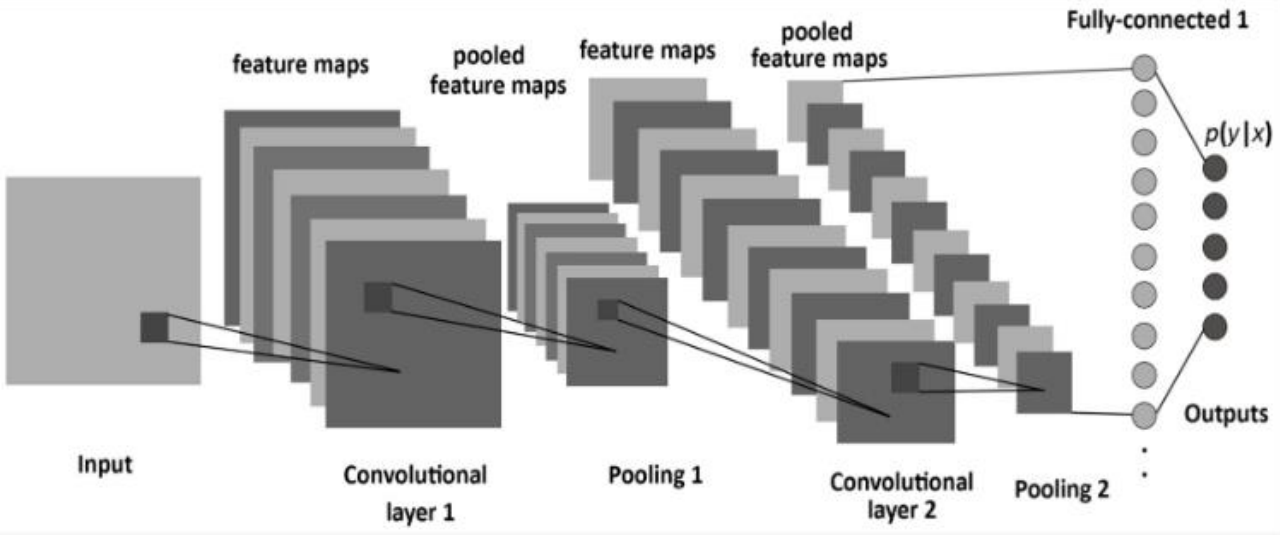
- Η μηχανική μάθηση αποτελεί έναν κλάδο της Τεχνητής Νοημοσύνης. Χαρακτηρίζεται από το γεγονός ότι οι υπολογιστές μαθαίνουν από τα δεδομένα που τους παρέχουμε χωρίς να προγραμματίζονται ρητά.
- Τα νευρωνικά δίκτυα αποτελούν έναν αλγόριθμο ΜΜ εμπνευσμένο από τα βιολογικά νευρωνικά δίκτυα. Κάθε ΝΝ περιέχει κόμβους που δημιουργούν συνδέσεις με άλλους κόμβους.
- Σε κάθε κρυφό νευρώνα πολλαπλασιάζεται κάθε είσοδος με το αντίστοιχο βάρος και υπολογίζεται το συνολικό άθροισμα. Αυτό το άθροισμα αποτελεί την είσοδο για τη συνάρτηση ενεργοποίησης φ, που αποτελεί την έξοδο του Νευρώνα.





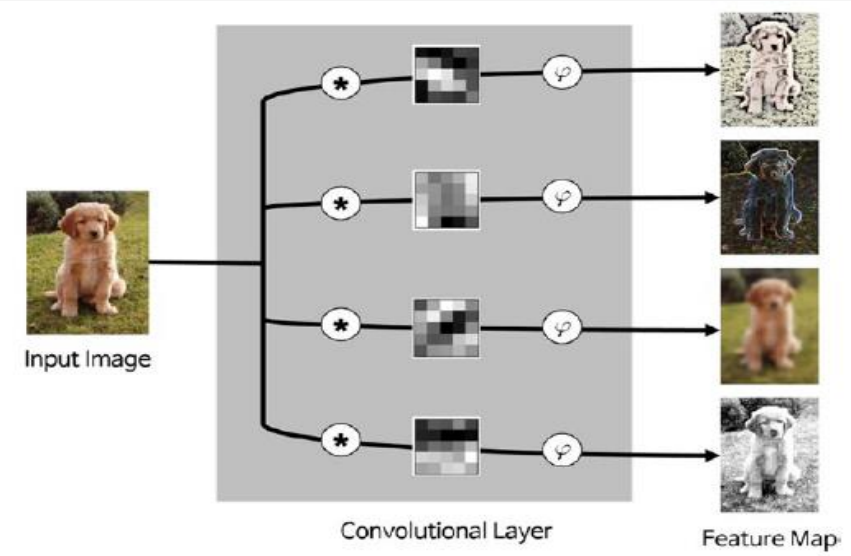
Συνελικτικά Νευρωνικά Δίκτυα (1/2)

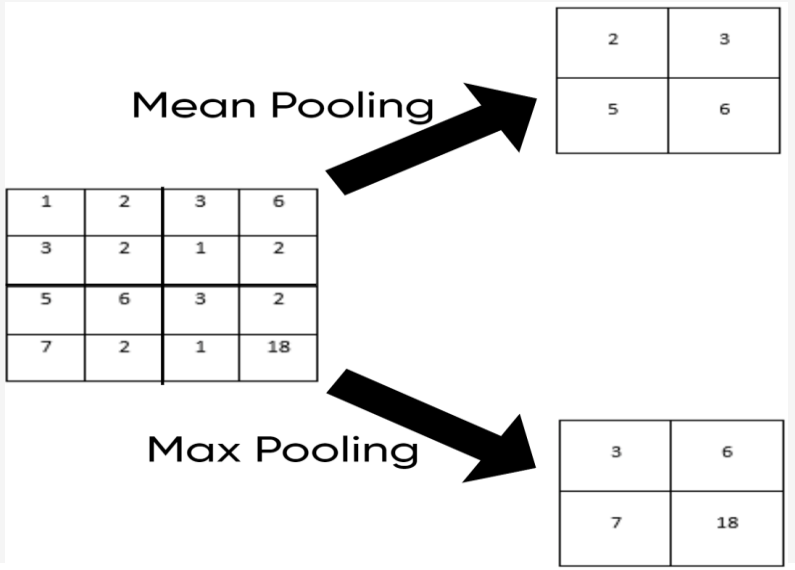
- Στον αλγόριθμο μας παράγονται μοντέλα Συνελικτικών Νευρωνικών Δικτύων (CNNs).
- Τα CNNs ανήκουν στον τομέα της βαθιάς μάθησης (Deep Learning) που επικεντρώνεται στην αναγνώριση μοτίβων. Χρησιμοποιούνται κυρίως στην αναγνώριση εικόνων.
- Η δομή ενός CNN αποτελείται κυρίως από convolutional και pooling layers.



Συνελικτικά Νευρωνικά Δίκτυα (2/2)

- Το επίπεδο συνέλιξης (convolutional) μετασχηματίζει την εικόνα μέσω της διαδικασίας συνέλιξης.
- Το επίπεδο συγκέντρωσης (pooling) συγκεντρώνει τα γειτονικά pixels σε ένα ενιαίο pixel. Ως εκ τούτου, το στρώμα συγκέντρωσης χρησιμεύει στη μείωση της διάστασης της εικόνας.





Αλγόριθμος FLITC

Ο αλγόριθμος FLITC περιλαμβάνει τις ακόλουθες λειτουργίες:

- Τον εντοπισμό δυσλειτουργικών τροφοδοτών με τη χρήση ενός μοντέλου CNN (FFNN).
- Τον εντοπισμό δυσλειτουργικών κλάδων με τη χρήση ενός μοντέλου CNN (FBNN).
- Την ταξινόμηση τύπου βλάβης σε έντεκα κλάσεις με τη χρήση ενός μοντέλου CNN (FCNN). Το είδος του σφάλματος βραχυκυκλώματος στο LVDG μπορεί να προσδιοριστεί από το μοντέλο.
- Τον υπολογισμό της θέσης του ελαττωματικού κλάδου με την εκτίμηση της απόστασης από τον ριζικό κόμβο χρησιμοποιώντας ένα μοντέλο CNN (FDNN).

Εκπαίδευση μοντέλου FFNN (1/2)

- Για την εκπαίδευση του μοντέλου χρησιμοποιούμε ένα κατάλληλο προεπεξεργασμένο σύνολο δεδομένων οπού διαχωρίζεται σε δεδομένα εκπαίδευσης (training) και δεδομένα επικύρωσης (validation).
- Η αριθμητική ακρίβεια του συνόλου δεδομένων είναι στη μορφή 32-bit floating point (FP32).
- Οι διαστάσεις του συνόλου των δεδομένων εκπαίδευσης είναι: Μήκος συνόλου δεδομένων x Ύψος x Πλάτος x Κανάλια και αντιστοιχούν στις παρακάτω τιμές:

```
x_trn shape: (21398, 32, 32, 9)
```

Εκπαίδευση μοντέλου FFNN (2/2)

- Το μοντέλο CNN που δημιουργείται περιέχει στρώματα Conv2D, Dropout, MaxPooling2D, Flatten και Dense.
- Επίσης ορίζονται οι παράμετροι που καθορίζουν την αρχιτεκτονική του μοντέλου και διαχειρίζονται τη διαδικασία εκπαίδευσης. Ονομάζονται υπερπαράμετροι και έχοντας τις βέλτιστες τιμές τους βελτιώνουμε την απόδοση του μοντέλου.
- Οι υπερπαράμετροι είναι: ο αριθμός των layers, Dropout Rate, Units per Layer, Kernel Initializer, batch size, αριθμός epochs, Early Stopping Patience.

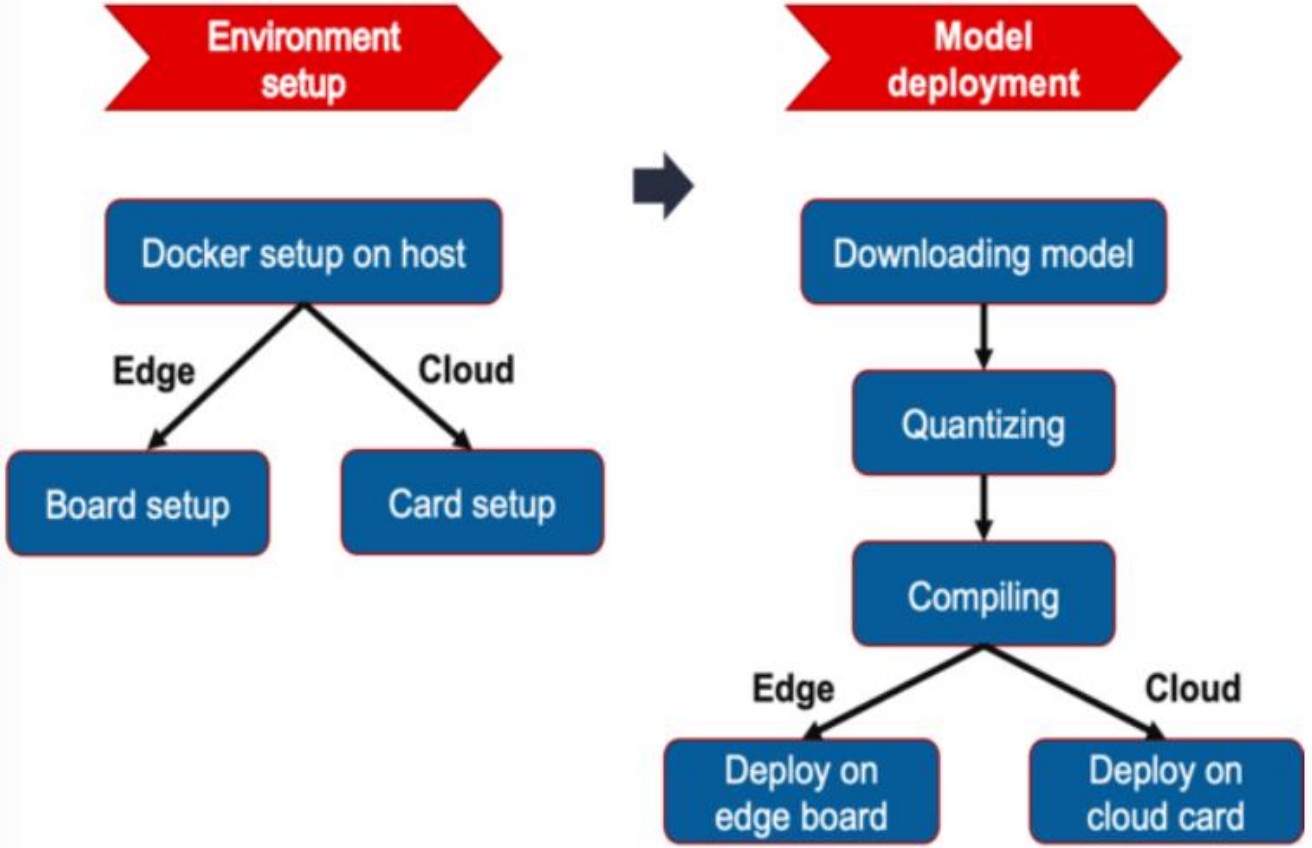
Αξιολόγηση μοντέλου FFNN

- Για την αξιολόγηση το μοντέλο λαμβάνει τα δεδομένα εισόδου (του validation dataset) και κάνει προβλέψεις με βάση αυτά. Η έξοδος υποδεικνύει ποιος τροφοδότης είναι ελαττωματικός.
- Εμείς συγκρίνουμε τις πραγματικές εξόδους (του validation dataset) με τις αντίστοιχες προβλέψεις του μοντέλου.
- Το ποσοστό ακρίβειας των προβλέψεων υπολογίστηκε ότι προσεγγίζει το 89,7%.

```
Accuracy: 0.8971247199402539
```

Διαδικασία Υλοποίησης

- Όλη η διαδικασία που πρέπει να ακολουθήσουμε για να υλοποιήσουμε το μοντέλο σε μια hardware συσκευή φαίνεται στο σχήμα:

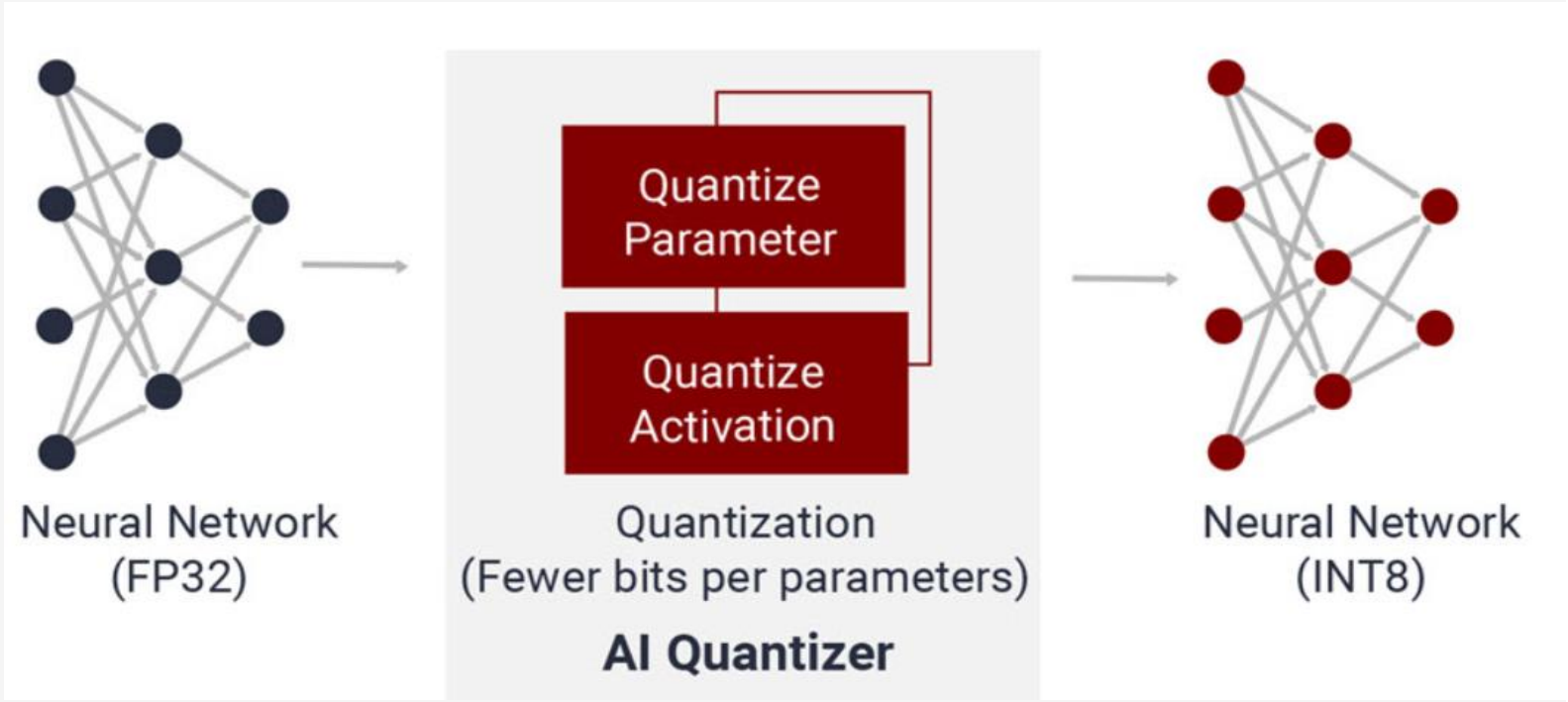


Κβαντοποίηση

- Αφού εκπαιδευτεί το μοντέλο FFNN και εγκατασταθεί το κατάλληλο περιβάλλον (docker, Vitis-AI), το επόμενο βήμα είναι η κβάντιση του.
- Η κβαντοποίηση μετά την εκπαίδευση μπορεί να μειώσει το μέγεθος του μοντέλου με ελάχιστη απώλεια ακρίβειας, ενώ ταυτόχρονα βελτιώνει την καθυστέρηση (latency) της CPU και του επιταχυντή υλικού.
- Ο κβαντιστής συνήθως μειώνει την υπολογιστική πολυπλοκότητα του μοντέλου μετατρέποντας τα βάρη και τις ενεργοποιήσεις κινητής υποδιαστολής από 32 bit (floating point) σε σταθερά σημεία, όπως τον 8 bit ακέραιο.

Τεχνικές Κβαντοποίησης

- TensorFlow-Lite Dynamic Range
- TensorFlow-Lite Full Integer
- Vitis-AI TensorFlow-2



Τεχνικές Κβαντοποίησης – TensorFlow-Lite Dynamic Range (1/2)

Πλεονεκτήματα:

- Η dynamic range κβαντοποίηση προσφέρει ταχύτερο υπολογισμό και μικρότερη χρήση μνήμης, χωρίς να απαιτείται η παροχή ενός αντιπροσωπευτικού συνόλου δεδομένων (calibration dataset) για βαθμονόμηση.
- Το μέγεθος του μοντέλου μειώνεται συνήθως από τέσσερις φορές και πάνω (στη περίπτωση μας από 4.5 MB σε 400 KB). Η επιτάχυνση του μοντέλου μπορεί να φτάσει το 2x-3x ή και περισσότερο.
- Η ακρίβεια του παραγόμενου μοντέλου έχει ελάχιστη μείωση από το αρχικό καθώς φτάνει το 88.6%.

Τεχνικές Κβαντοποίησης – TensorFlow-Lite Dynamic Range (2/2)

Μειονεκτήματα

- Η επιτάχυνση της κβαντοποίησης αυτής δεν είναι τόσο μεγάλη όσο εκείνη ενός full fixed-point υπολογισμού, επειδή οι έξοδοι διατηρούνται σε κινητής υποδιαστολής (floating-point).
- Το μοντέλο που παράγεται μπορεί να υλοποιηθεί μόνο πάνω σε CPU, πράγμα το οποίο είναι αρκετά περιοριστικό.

Technique	Benefits	Hardware
Dynamic range quantization	4x smaller, 2x-3x speedup	CPU

Τεχνικές Κβαντοποίησης – TensorFlow-Lite Full Integer (1/2)

Πλεονεκτήματα

- Τα βάρη 32-bit κινητής υποδιαστολής (floating point) και οι έξοδοι ενεργοποίησης μετατρέπονται στον πλησιέστερο αριθμό σταθερής υποδιαστολής 8-bit.
- Το μοντέλο μειώνεται συνήθως από τέσσερις φορές και πάνω (στη περίπτωση μας από 4,5MB σε 400KB) και επιταχύνεται πάνω από τρεις φορές (3x+).
- Το μοντέλο μπορεί να χρησιμοποιηθεί σε συσκευές χαμηλής κατανάλωσης ενέργειας όπως οι μικροελεγκτές (ενσωματωμένα συστήματα) και σε επιταχυντές που υποστηρίζουν μόνο ακέραιους αριθμούς, όπως η Edge TPU, αλλά και σε CPU.

Τεχνικές Κβαντοποίησης – TensorFlow-Lite Full Integer (2/2)

Μειονεκτήματα

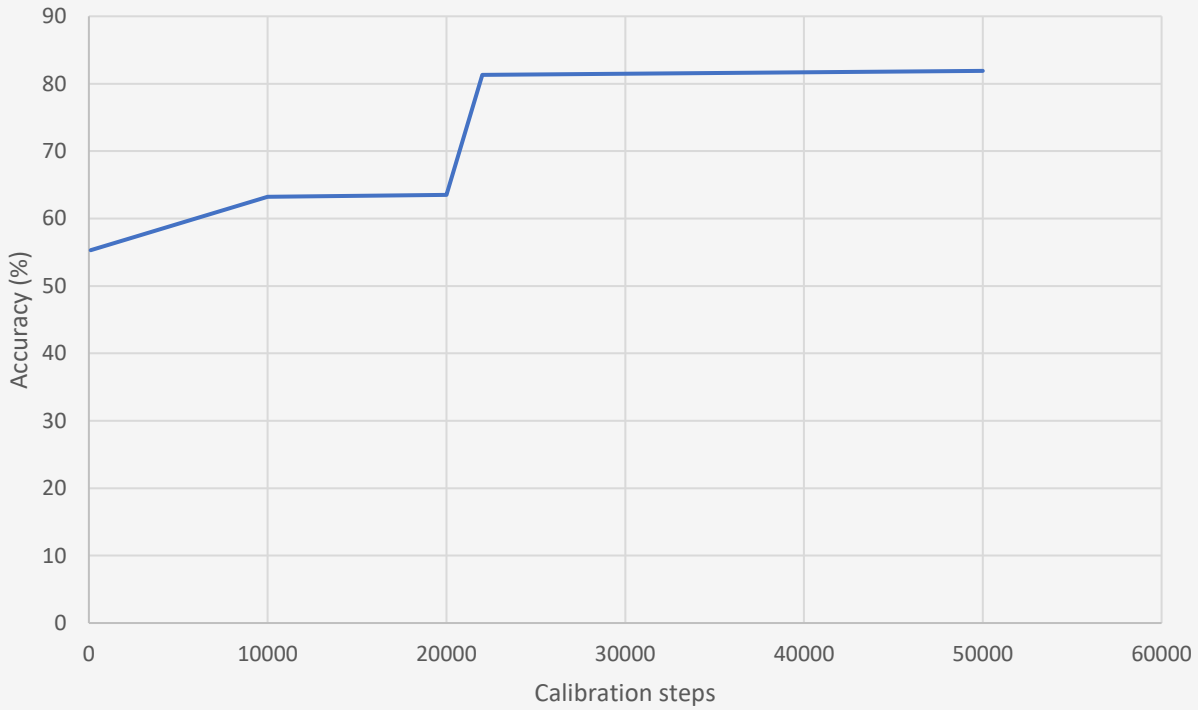
- Παρατηρούμε ότι η ακρίβεια του μοντέλου μειώνεται σημαντικά καθώς φτάνει το 57,5%. Η απόκλιση είναι αρκετά μεγάλη και κρίνεται απαγορευτική για την υλοποίηση του μοντέλου μας.

Technique	Benefits	Hardware
Dynamic range quantization	4x smaller, 2x-3x speedup	CPU
Full integer quantization	4x smaller, 3x+ speedup	CPU, Edge TPU, Microcontrollers

Τεχνικές Κβαντοποίησης – Vitis-AI TensorFlow-2 (1/3)

- Είναι μία διαδικασία κατά την οποία τα μοντέλα με αριθμητική ακρίβεια 32-bit floating point (FP32) μετατρέπονται σε μοντέλα με χαμηλότερη αριθμητική ακρίβεια των 8 bit ακεραίων (INT8).
- Η μέθοδος αυτή χρησιμοποιεί δύο μεταβλητές για την κβαντοποίηση του μοντέλου. Είναι το σύνολο δεδομένων βαθμονόμησης (calib_dataset) και τα βήματα βαθμονόμησης (calib_steps).
- Ο στόχος είναι να προσδιοριστεί ο βέλτιστος αριθμός βημάτων βαθμονόμησης σε σχέση με τον αριθμό των δεδομένων βαθμονόμησης που απαιτείται για τη μεγιστοποίηση της ακρίβειας του μοντέλου μετά τον κβαντισμό.

Τεχνικές Κβαντοποίησης – Vitis-AI TensorFlow-2 (2/3)



- Το calibration dataset που χρησιμοποιούμε περιλαμβάνει τα δεδομένα εκπαίδευσης που αποτελούνται από περίπου 22000 images. Για να βρούμε τη βέλτιστη τιμή των calibration steps κάνουμε δοκιμές κβαντοποιώντας το μοντέλο και έπειτα μετράμε την ακρίβεια του.
- Σύμφωνα με το παρακάτω σχήμα η ακρίβεια του μοντέλου αυξάνεται και σταθεροποιείται όταν ο αριθμός των calib_step προσεγγίζει τον αριθμό των δειγμάτων.
- Το μοναδικό μειονέκτημα είναι ότι με την αύξηση των βημάτων κβαντοποίησης αυξάνεται και ο χρόνος εκτέλεσης της διαδικασίας.

Τεχνικές Κβαντοποίησης – Vitis-AI TensorFlow-2 (3/3)

- Η ακρίβεια του κβαντισμένου μοντέλου προσεγγίζει το ποσοστό 82%, επιτυγχάνοντας μικρή μείωση από το αρχικό μοντέλο.
- Τα κβαντισμένα μοντέλα προορίζονται για επιταχυντές που υποστηρίζουν υπολογιστική πολυπλοκότητα ακεραίων αριθμών, όπως είναι οι συσκευές Edge, η σειρά Xilinx Versal AI Core, οι κάρτες Xilinx Alveo, οι Xilinx Zynq UltraScale+ MPSoC και οι Xilinx Zynq-7000 SoCs.
- Ο κβαντιστής του Vitis-AI μπορεί να παρέχει πάνω από 3 φορές (3x+) επιτάχυνση στην εκτέλεση του μοντέλου, ενώ ταυτόχρονα μειώνει το μέγεθος του αρχείου κατά περίπου τέσσερις φορές (στη περίπτωση μας από 4.5 MB σε 1.6MB).

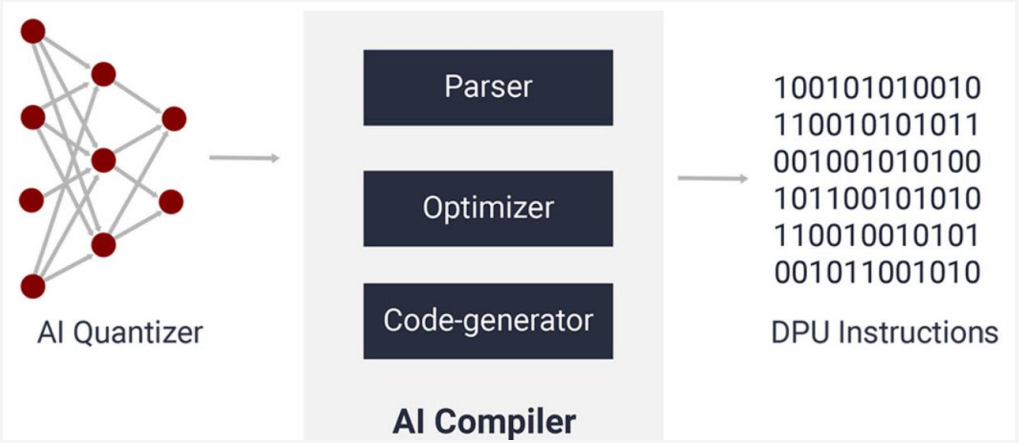
Σύγκριση τεχνικών κβαντοποίησης

Techniques	Vai_q (tf2)	Dynamic Range (tflite)	Full-Integer (tflite)
Accuracy reduction	-8,7%	-1,2%	-35,9%
Hardware Devices	Edge, Versal AI, Alveo, ZynQ	CPU	CPU, Edge, Microcontrollers
Speedup	3x+	(2x-3x) +	3x+
Size Reduction	4x	4x+	4x+

Λαμβάνοντας υπόψη τις απαιτήσεις για την υλοποίηση ενός μοντέλου που παρέχει τη βέλτιστη απόδοση και επιτάχυνση και μπορεί να χρησιμοποιηθεί σε συσκευές υλικού που υποστηρίζουν αριθμητική ακρίβεια ακεραίου, καταλήγουμε στο συμπέρασμα ότι η καλύτερη τεχνική για το μοντέλο FFNN είναι η vai_q_tensorflow-2.

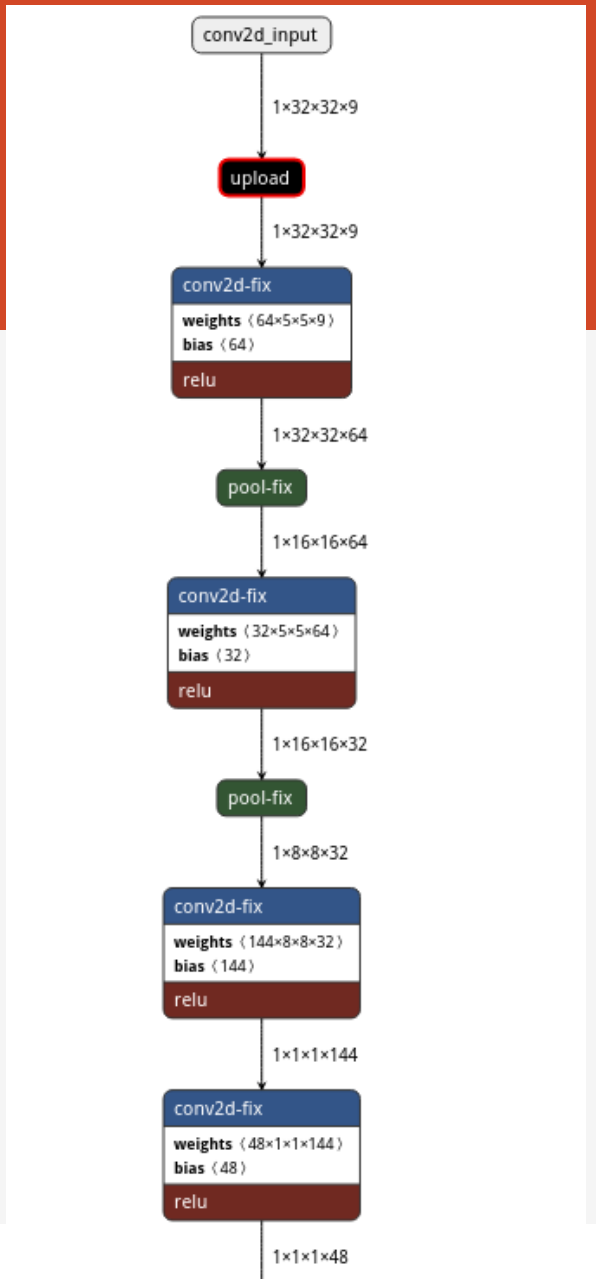
Compiling του μοντέλου

- Μετά την κβαντοποίηση είναι η σειρά της διαδικασίας της μεταγλώττισης. Το μοντέλο αντιστοιχίζεται σε ένα εξαιρετικά αποδοτικό σύνολο εντολών και μοντέλο ροής δεδομένων από τον μεταγλωττιστή του Vitis-AI. Στην προκειμένη περίπτωση, αντιστοιχίζει το μοντέλο σε σύνολο εντολών DPU.
- Η μονάδα επεξεργαστή Deep Learning (DPU) είναι ένας ρυθμιζόμενος μηχανισμός που έχει σχεδιαστεί ειδικά για βαθιά νευρωνικά δίκτυα. Πρόκειται για μια συλλογή από παραμετροποιήσιμους IP cores υλοποιημένους σε hardware, οι οποίοι δεν χρειάζονται θέση ή διαδρομή.
- Η DPU μπορεί να προσαρμοστεί ώστε να ταιριάζει σε διαφορετικές Edge devices. Στην παρούσα εργασία, χρησιμοποιούμε την DPUCZDX8G πάνω στην Xilinx Zynq UltraScale+ ZCU104 device. Αυτή η πλατφόρμα ανήκει στην κατηγορία των Field-Programmable-Gate-Array (FPGA).



Υλοποίηση στο Hardware (1/3)

- Μετά το compilation, φορτώνουμε το μοντέλο. Αναζητούμε το DPU subgraph και αφού βρεθεί δημιουργούμε έναν runner DPU.
- Προεπεξεργαζόμαστε τα δεδομένα εισόδου σε μορφή int-8 για να τα καταστήσουμε συμβατά με τα αποδεκτά δεδομένα εισόδου της DPU.
- Το subgraph εκτελείται στην DPU με τα προεπεξεργασμένα δεδομένα ως είσοδο και επιστρέφονται τα αποτελέσματα.



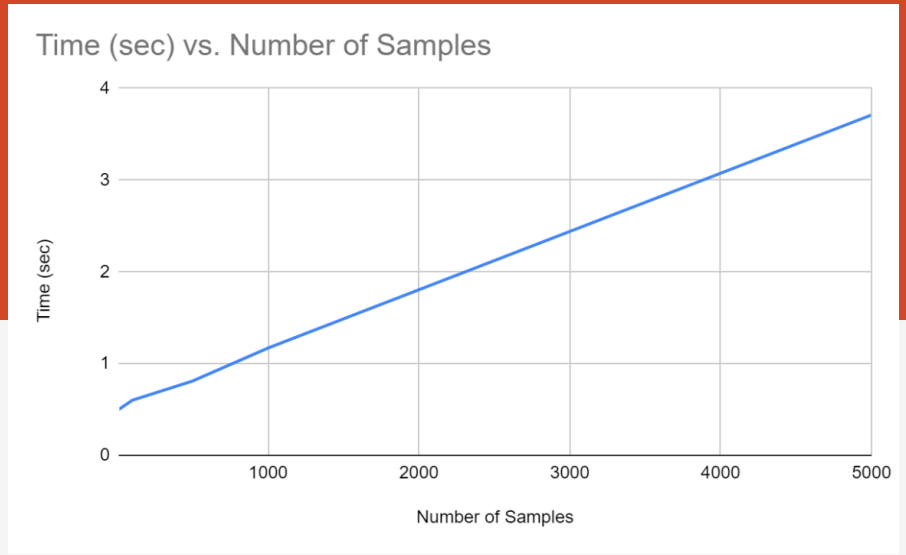
Υλοποίηση στο Hardware (2/3)

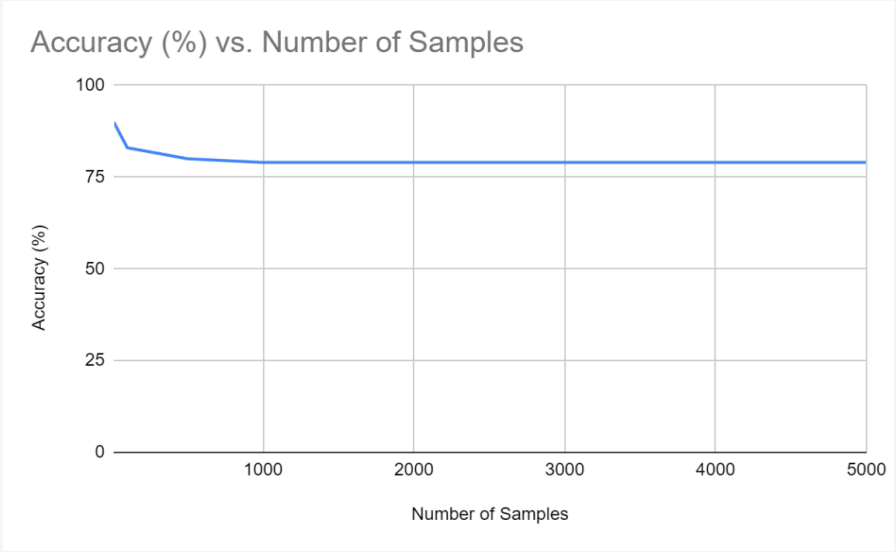
- Ο παρακάτω πίνακας απεικονίζει τις επιδόσεις και τα αποτελέσματα του χρόνου εκτέλεσης για 10, 100, 500, 1000, 3000 και 5000 δείγματα εισόδου:

Number of Samples	10	100	500	1000	3000	5000
Time (sec)	0,5	0,6	0,81	1,17	2,44	3,71
Accuracy (%)	90	83	80	79	79	79

Υλοποίηση στο Hardware (3/3)

- Η αύξηση μεγέθους του δείγματος συνοδεύεται από ανάλογη αύξηση της χρονικής διάρκειας εκτέλεσης, διατηρώντας όμως το χρόνο εκτέλεσης αρκετά μικρό.
- Το μοντέλο σταθεροποιείται σε ποσοστό ακρίβειας 79% στις σωστές προβλέψεις του. Επομένως, αφού το ποσοστό ακρίβειας του μοντέλου μετά την κβάντιση μετρήθηκε στο 81-82%, αυτό σημαίνει ότι η απόκλιση είναι μικρή.





Σύγκριση μοντέλων - Xmodel vs Original (1/2)

- Σύμφωνα με τους πίνακες, υπάρχει πολύ ταχύτερη εκτέλεση του “xmodel” σε σύγκριση με το αρχικό μοντέλο και υπάρχει μεγάλη μείωση του μεγέθους του μοντέλου (από 4,5 MB σε 500 KB) με μικρή μείωση της ακρίβειας.
- Παρατηρείται ότι, με την αύξηση των δειγμάτων, ο χρόνος εκτέλεσης ανά δείγμα στο αρχικό μοντέλο σταθεροποιείται στα 0,062 δευτερόλεπτα. Από την άλλη, ο μέσος χρόνος ανά δείγμα στη DPU μειώνεται συνεχώς. Αυτό αποδεικνύει την επιτάχυνση που προσφέρει η DPU.

	Accuracy	Size	Data Type
“h5”	89%	4.5 MB	Float32
“xmodel”	79-82%	500 KB	Int8

Number of Samples	10	100	500	1000	3000	5000
“h5” (time/sample sec) (CPU)	0,073	0,06	0,062	0,062	0,062	0,062
“xmodel” (time/sample sec) (DPU)	0,05	0,006	0,00162	0,00117	0,00081	0,00074

Σύγκριση μοντέλων - Xmodel vs Original (2/2)

- Αν θέλουμε να αξιοποιήσουμε τη CPU για καλύτερες επιδόσεις, πρέπει να χρησιμοποιήσουμε batches για την εκτέλεση του μοντέλου με πολλές εισόδους ταυτόχρονα. Μπορούμε να παρατηρήσουμε ότι τα batches μειώνουν σημαντικά τον χρόνο εκτέλεσης ακόμα και από τη DPU (2x) λόγω των πόρων του server του Πανεπιστημίου.

Number of Samples	10	100	500	1000	3000	5000
"h5" with no batches on CPU (sec)	0,73	6	31	62	186	310
"h5" with batches x128 on CPU (sec)	0,26	0,3	0,39	0,52	1,06	1,6
"xmodel" with no batches on DPU (sec)	0,5	0,6	0,81	1,17	2,44	3,71

Σύγκριση μοντέλων – Xmodel vs Tflite (1/2)

- Από τον πίνακα επιβεβαιώνεται ότι η DPU είναι ειδικά σχεδιασμένη για να εκτελεί νευρωνικά δίκτυα με υψηλές ταχύτητες και αποδοτικότητα, ιδίως μοντέλα int8. Έτσι, η εκτέλεση του μοντέλου “xmodel” στο FPGA αποτελεί τον ταχύτερο τρόπο, έχοντας παράλληλα μικρή μείωση στην ακρίβεια.

Models	xmodel	TFlite (Dynamic Range)	TFlite (Full Integer)
Runtime (for 5000 samples)	3,7 sec	6,2 sec	23 sec
Accuracy reduction	-11%	-1,2%	-36%
File Size	500 KB	400 KB	400 KB
Data Type	Int8	Float32	Int8
Device	DPU/FPGA	CPU	Raspberry Pi
Device Cost	$1678	$64-594	$15-100

Σύγκριση μοντέλων – Xmodel vs Tflite (2/2)

- Ο χρόνος εκτέλεσης του μοντέλου TFlite στην CPU είναι σημαντικά μικρότερος σε σύγκριση με το μοντέλο “h5”, ωστόσο είναι πιο αργός σε σύγκριση με το DPU. Επίσης, υπάρχει ελάχιστη μείωση στην ακρίβεια.
- Το Raspberry Pi έχει περιορισμένους πόρους σε σύγκριση με έναν προσωπικό υπολογιστή και συνεπώς αυτό επηρεάζει την ταχύτητα εκτέλεσης. Η υπολογιστική του ισχύς είναι συγκριτικά περιορισμένη, γεγονός που εξηγεί τον χρόνο εκτέλεσης των 23 δευτερολέπτων για το μοντέλο TFLite. Η ακρίβεια του μοντέλου επίσης είναι αρκετά μειωμένη.
- Μπορεί η DPU να είναι η πιο γρήγορη μέθοδος όμως είναι και η πιο κοστοβόρα. Η αγορά ενός FPGA Zynq UltraScale+ MPSoC ZCU104 φτάνει τα 1678$. Η CPU αποτελεί συνήθως μία πιο φθηνή επιλογή ωστόσο εξαρτάται από το μοντέλο. Επίσης είναι μία πιο εύκολη λύση καθώς CPU διαθέτει κάθε προσωπικός υπολογιστής. Το Raspberry Pi είναι ακόμα πιο φθηνή επιλογή γιατί κυμαίνεται από 15-100$ στην αγορά.

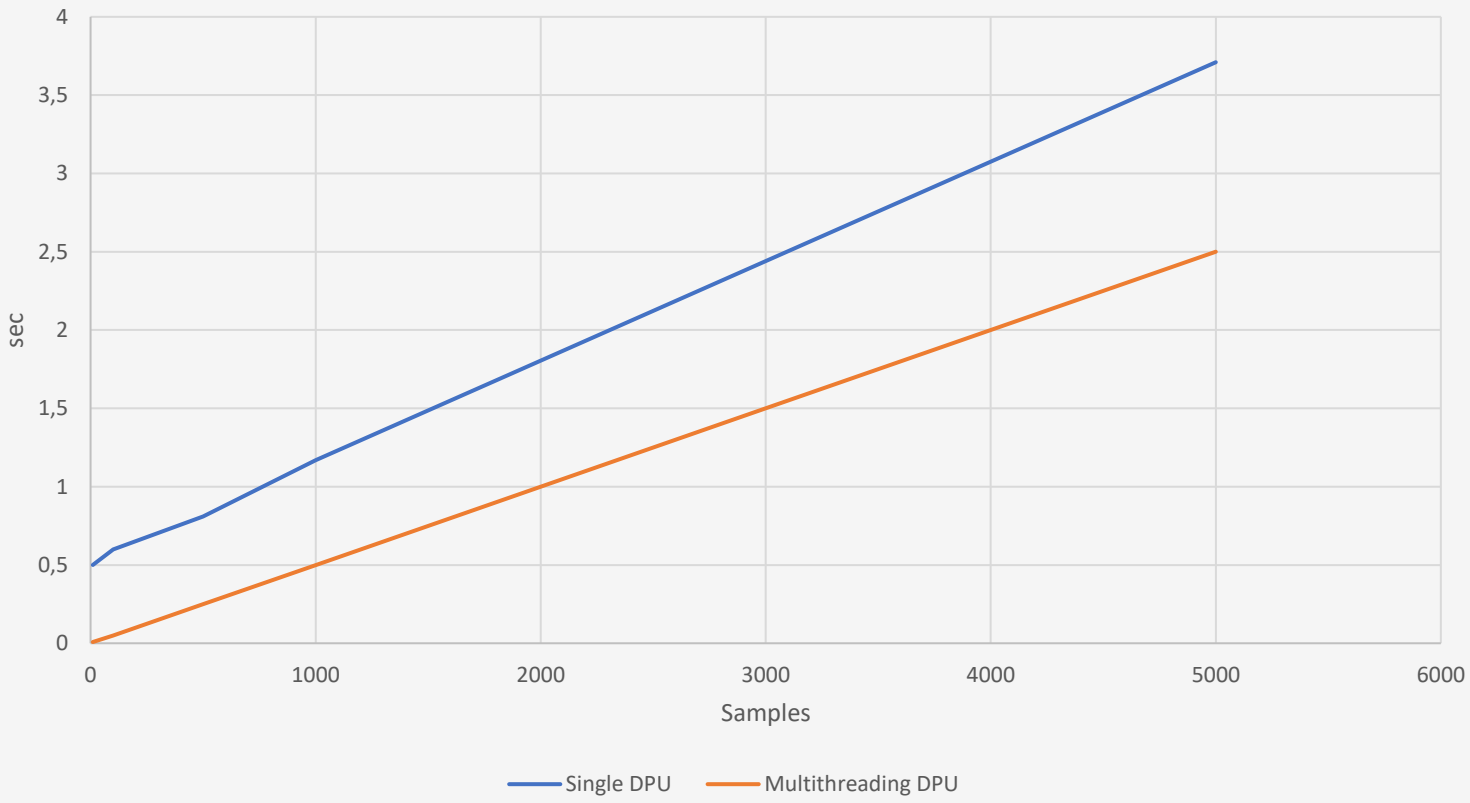
Σύγκριση μοντέλων – Single vs Multithreading DPU (1/2)

- Αν και η DPU είναι ο ταχύτερος τρόπος εκτέλεσης ενός μοντέλου, μπορούμε να μειώσουμε περαιτέρω τον χρόνο εκτέλεσης χρησιμοποιώντας και τις δύο DPU που διαθέτει το σύστημά μας. Για να γίνει αυτό προσθέσαμε threads στον κώδικα και με αυτόν τον τρόπο μπορούμε να επεξεργαζόμαστε τα δεδομένα παράλληλα.

Number of Samples	10	100	500	1000	3000	5000
Single DPU runtime	0,5 sec	0,6 sec	0,81 sec	1,17 sec	2,44 sec	3,71 sec
Multithreading DPU runtime	0,008 sec	0,05 sec	0,25 sec	0,5 sec	1,5 sec	2,5 sec

Σύγκριση μοντέλων – Single vs Multithreading DPU (2/2)

- Καθώς τα δείγματα αυξάνονται, καταλήγουμε στο συμπέρασμα ότι το multithreading προσφέρει επιτάχυνση κατά περίπου μιάμιση φορά (1,5x) σε σχέση με μία DPU.
- Έτσι επιτυγχάνουμε, μία εξαιρετικά γρήγορη εκτέλεση του μοντέλου έχοντας μικρή απόκλιση στην ακρίβεια των προβλέψεων.



Σύνοψη

- Εν κατακλείδι, η παρούσα εργασία αποτελεί μια καλή απεικόνιση του τρόπου με τον οποίο τα μοντέλα μηχανικής μάθησης εκτελούμενα σε συσκευές υλικού μπορούν να βελτιώσουν την ταχύτητα τους. Η μελλοντική υλοποίηση και των υπόλοιπων σταδίων του αλγορίθμου FLITC με τη διαδικασία που αναλύσαμε θα συμβάλει στη μείωση της καθυστέρησης στην ανίχνευση σφαλμάτων στα δίκτυα διανομής ηλεκτρικής ενέργειας, οδηγώντας ενδεχομένως σε πολύ καλύτερη διαχείριση του δικτύου.

Ακρίβεια μοντέλων

Model	FFNN	FBNN-1	FBNN-2	FBNN-3	FDNN	FCNN
Original (“h5”) accuracy	89,7%	84,5%	91,2%	96,33%		
Quantized (“xmodel”) accuracy	82%	83,5%	90,7%	96,29%		