ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Οπτική Αναγνώριση και Γραφή με το Ανθρωποειδές Ρομπότ NAO

Δημήτριος Καβρουλάκης

Εξεταστική Επιτροπή
Αναπληρωτής Καθηγητής Μιχαήλ Γ. Λαγουδάκης (ΗΜΜΥ)
Καθηγητής Μιχαήλ Ζερβάκης (ΗΜΜΥ)
Δρ. Νικόλαος Σπανουδάκης (ΜΠΔ)

Χανιά, Δεκέμβριος 2020

TECHNICAL UNIVERSITY OF CRETE, GREECE

SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

Visual Recognition and Writing with the NAO Humanoid Robot

Dimitrios Kavroulakis

Thesis Committee
Associate Professor Michail G. Lagoudakis (ECE)
Professor Michalis Zervakis (ECE)
Dr. Nikolaos I. Spanoudakis (PEM)

Chania, December 2020

Περίληψη

Η τεχνητή νοημοσύνη έχει ως στόχο τη δημιουργία συμπεριφορών, είτε από προγράμματα - πράκτορες, είτε από μηχανήματα-ρομπότ, οι οποίες επιδεικνύουν κάποια βασική ευφυΐα. Σήμερα, ο άνθρωπος μπορεί να κατασκευάζει διαφόρων ειδών ρομπότ με σκοπό να εκτελούν κάποιες ενέργειές του, ωστόσο κάθε τέτοιο ρομπότ χρειάζεται να εκπαιδευτεί κατάλληλα για να τις εκτελεί σωστά και κάποιες φορές ίσως να αποτυγχάνει. Η παρούσα διπλωματική εργασία παρουσιάζει την ανάπτυξη μιας «ανθρώπινης» συμπεριφοράς στο ανθρωποειδές ρομπότ NAO, η οποία αφορά στην οπτική αναγνώριση ενός χειρόγραφου κειμένου (λέξης) από το ρομπότ και στη γραφή αυτού του ίδιου κειμένου σε έναν λευκό πίνακα με τη χρήση ενός μαρκαδόρου που κρατάει στο χέρι του. Πιο συγκεκριμένα, το NAO έχει εκπαιδευτεί με συνελικτικά νευρωνικά δίκτυα (Convolution Neural Networks) και ένα σύνολο δεδομένων από εικόνες με κεφαλαία χειρόγραφα γράμματα του λατινικού αλφαβήτου για να αναγνωρίζει γράμματα. Επίσης, έγινε εκπαίδευση του NAO στην γραφή κεφαλαίων χαρακτήρων του αλφαβήτου μέσω μιας διαδικασίας χειροκίνητης καθοδήγησης των αρθρώσεων του βραχίονά του από τον άνθρωπο-εκπαιδευτή και καταγραφής των χρονισμένων τροχιών των αρθρώσεων μέσα από το εξειδικευμένο λογισμικό Choregraphe. Η ενοποίηση των επιμέρους τμημάτων της επιθυμητής συμπεριφοράς έγινε με χρήση της γλώσσας προγραμματισμού Python. Κατά την εκτέλεση της τελικής συμπεριφοράς, το ρομπότ NAO λαμβάνει κάποια εικόνα από την κάμερά του, την επεξεργάζεται με κατάλληλους αλγορίθμους επεξεργασίας εικόνας από τη βιβλιοθήκη OpenCV, εφαρμόζει το εκπαιδευμένο νευρωνικό δίκτυο και ανιχνεύει ακολουθίες (λέξεις) από χειρόγραφα γράμματα που βλέπει γραμμένα στον πίνακα. Στη συνέχεια γράφει ένα-ένα τα γράμματα που έχει αναγνωρίσει στον λευκό πίνακα, εκτελώντας τις κατάλληλες κινήσεις με τον βραχίονα που κρατάει τον μαρκαδόρο, μετατοπίζοντας τη θέση του κάθε φορά, ώστε να αναπαραχθεί ορθά η ίδια ακολουθία γραμμάτων. Η προσέγγιση αυτή ολοκληρώθηκε με επιτυχία, εφόσον το ρομπότ NAO σημειώνει υψηλά ποσοστά ακρίβειας στην ορθή οπτική αναγνώριση των χειρόγραφων γραμμάτων που του δίνονται, αλλά και στην ορθή, ευανάγνωστη καταγραφή τους με ανεπαίσθητες διαφοροποιήσεις. Η συμπεριφορά αυτή, η οποία προσομοιάζει σε προσφιλείς διαδικασίες μάθησης των παιδιών, μπορεί να αποτελέσει σημείο ενδιαφέροντος ως διαδραστική τεχνολογική επίδειξη σε εκδηλώσεις STEM για παιδιά.

Abstract

Artificial intelligence aims to create behaviors, either by software agents or by robotic machines, which exhibit some basic intelligence. Today, humans can build different types of robots in order to perform some of their actions; however, each such robot needs to be properly trained to perform them accurately and sometimes it may fail. This diploma thesis presents the development of a "human" behavior on the humanoid robot NAO, which involves the visual recognition of a handwritten text (word) by the robot and the writing of the same text on a whiteboard using a marker held in his hand. More specifically, the NAO has been trained with Convolution Neural Networks and a dataset of uppercase handwritten letters of the Latin alphabet to recognize letters. NAO was also trained in the writing of uppercase letters of the Latin alphabet through a process of manual guidance of the arm joints by the man-trainer and recording of the timed trajectories of the joints through the specialized Choregraphe software. The individual components of the target behavior were integrated using the Python programming language. During the execution of the final behavior, the NAO robot receives an image from its camera, processes it with appropriate image processing algorithms from the OpenCV library, applies the trained neural network and detects sequences (words) of handwritten letters that it sees written on the board. Then, the robot writes one by one the letters that have been identified on the whiteboard, performing the appropriate movements with the arm holding the marker, shifting its position each time, so that the same sequence of letters is reproduced correctly. This approach was completed successfully, since the NAO robot achieves high percentages of accuracy in the correct visual recognition of the handwritten letters given to it, but also in their correct, legible writing with subtle differences. This behavior, which resembles children's learning processes, can be a point of interest as an interactive technological demonstration at STEM events for kids.

Ευχαριστίες

Στο σημείο αυτό, με την ολοκλήρωση της παρούσας διπλωματικής εργασίας, θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου για τη συνεχή στήριξή τους όλα αυτά τα χρόνια ενισχύοντάς με, με τα απαραίτητα εφόδια. Επίσης, θα ήθελα να ευχαριστήσω τον Αναπλ. Καθηγητή κύριο Μιχαήλ Γ. Λαγουδάκη για την επίβλεψη αυτής της διπλωματικής εργασίας και για την ευκαιρία, που μου έδωσε να την εκπονήσω, καθώς και τον Καθηγητή Μιχαήλ Ζερβάκη και τον Δρ. Νικόλαο Σπανουδάκη για την συμμετοχή τους στην εξεταστική επιτροπή. Τέλος, να ευχαριστήσω τον φίλο και συνάδελφο Γιώργο Πίτση για την υπομονή και τη βοήθεια, που μου έχει προσφέρει όλα αυτά τα χρόνια.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1

Εισαγωγή

Στην εποχή μας η τεχνητή νοημοσύνη (artificial intelligence - AI) και τα ρομπότ έχουν συνεισφέρει αρκετά στην ανάπτυξη της τεχνολογίας και στη βελτίωση της ζωής του ανθρώπου. Στόχος της τεχνητής νοημοσύνης είναι η δημιουργία συμπεριφορών είτε από προγράμματα - πράκτορες είτε από μηχανήματα - ρομπότ, οι οποίες θα έχουν κάποια βασική ευφυΐα. Η ευφυΐα αυτή μπορεί να αποτελείται από στοιχεία ικανότητας μάθησης, εξαγωγής συμπερασμάτων, επίλυσης προβλημάτων και πολλά άλλα. Προκειμένου να επιτευχθεί αυτό, ο άνθρωπος δημιούργησε ένα από τα σημαντικότερα κομμάτια της τεχνητής νοημοσύνης, τη μηχανική μάθηση (machine learning). Η μηχανική μάθηση είναι ένας από τους ταχύτερα αναπτυσσόμενους και πιο συναρπαστικούς τομείς στη σύγχρονη επιστήμη και τη μηχανική. Στη συνέχεια, δημιουργήθηκε μια ισχυρή κατηγορία αλγορίθμων μηχανικής μάθησης, τα βαθιά νευρωνικά δίκτυα (deep neural networks), που επιτρέπουν στις μηχανές να λαμβάνουν ακριβείς αποφάσεις χωρίς βοήθεια από τον άνθρωπο.

Ο άνθρωπος μπορεί με ευκολία από τα πρώτα κιόλας χρόνια της ζωής του να αναγνωρίζει λέξεις και να τις γράφει. Αντίθετα, για ένα ρομπότ αυτό είναι πολύ δύσκολο, αφού πρέπει να εκπαιδευτεί αρκετά καλά για να το πετύχει αυτό. Η αντιγραφή λέξεων από ένα ρομπότ αποτελεί μια πρόκληση κι ένα πρόβλημα, το οποίο θα ήταν εντυπωσιακό να επιλυθεί.

Με αφορμή τα παραπάνω, η εργασία αυτή επικεντρώνεται στην εκπαίδευση του ρομπότ NAO, ώστε να είναι ικανό να ανιχνεύει και να αναγνωρίζει χειρόγραφες λέξεις με κεφαλαία γράμματα της αγγλικής αλφαβήτου και στη συνέχεια να τις γράφει σε έναν άσπρο πίνακα. Παρακάτω θα αναφερθούμε με περισσότερες λεπτομέρειες στην υλοποίηση του προβλήματος αυτού.

1.1 Συνεισφορά Διπλωματικής Εργασίας

Στο πλαίσιο της παρούσας διπλωματικής εργασίας υλοποιήθηκε μια συμπεριφορά στο ανθρωποειδές ρομπότ NAO. Η συμπεριφορά αυτή αφορά στην αντιγραφή χειρόγραφων λέξεων από το ρομπότ NAO, όπως αναφέρθηκε και παραπάνω. Προκειμένου να υλοποιηθεί η εκπαίδευση για την οπτική αναγνώριση των χειρόγραφων κεφαλαίων γραμμάτων, χρησιμοποιήθηκε ένα dataset από εικόνες με κεφαλαία χειρόγραφα γράμματα της αγγλικής αλφαβήτου. Ακόμη, χρησιμοποιήθηκαν νευρωνικά δίκτυα (Neural Networks) και πιο συγκεκριμένα τα συνελικτικά νευρωνικά δίκτυα (Convolution Neural Networks).

Επιπλέον, για την ανίχνευση των χειρόγραφων γραμμάτων χρησιμοποιήθηκαν αλγόριθμοι επεξεργασίας εικόνας από τη βιβλιοθήκη OpenCV και για την εκπαίδευση του ρομπότ NAO, ώστε να μπορεί να γράψει, χρησιμοποιήθηκε το πρόγραμμα Choregraphe. Τα αποτελέσματα απέδειξαν ότι οι αλγόριθμοι επεξεργασίας εικόνας της OpenCV και τα συνελικτικά δίκτυα (CNN), που χρησιμοποιήσαμε ήταν αρκετά σημαντικά, αφού οι αρχιτεκτονικές, που δοκιμάσαμε πέτυχαν μεγάλη ακρίβεια σε ένα τέστ σετ από εικόνες με κεφαλαία χειρόγραφα γράμματα της αγγλικής αλφαβήτου, τα οποία είχαμε γράψει σε έναν άσπρο πίνακα. Η συμπεριφορά αυτή, η οποία προσομοιάζει σε προσφιλείς διαδικασίες μάθησης των παιδιών, μπορεί να αποτελέσει σημείο ενδιαφέροντος ως διαδραστική τεχνολογική επίδειξη σε εκδηλώσεις STEM (Science, Technology, Engineering, Mathematics) για παιδιά.

1.2 Επισκόπηση Διπλωματικής Εργασίας

Η εργασία αυτή είναι οργανωμένη σε εφτά κεφάλαια και παρακάτω θα παρουσιάσουμε συνοπτικά τη δομή των κεφαλαίων, εκτός του εισαγωγικού Κεφαλαίου 1, ώστε να γίνει πιο εύκολο στον αναγνώστη να ακολουθήσει την ανάπτυξη του θέματος στη συγκεκριμένη εργασία.

Στο Κεφάλαιο 2 παρουσιάζεται το θεωρητικό υπόβαθρο των βασικών τεχνολογιών με ορολογίες και ορισμούς, που σχετίζονται με την παρούσα διπλωματική εργασία. Οι έννοιες, που αναλύονται είναι οι εξής: Ανθρωποειδές Ρομπότ NAO, Choregraphe, η βιβλιοθήκη OpenCV και κάποιοι αλγόριθμοί της, το Tensorflow, Keras, μηχανική μάθηση, νευρωνικά δίκτυα, βαθιά νευρωνικά δίκτυα και συνελικτικά νευρωνικά δίκτυα.

Στο Κεφάλαιο 3 γίνεται η αναλυτική περιγραφή του προβλήματος, καθώς και αναφορές σε σχετικές εργασίες, που έχουν γίνει στο παρελθόν.

Στο Κεφάλαιο 4 περιγράφεται η δική μας προσέγγιση για την οπτική αναγνώριση του χειρόγραφου κειμένου, τη γραφή του χειρόγραφου κειμένου και τη σύνδεσή τους.

Στο Κεφάλαιο 5 παρουσιάζεται η πειραματική διαδικασία με γραφικές παραστάσεις και αποτελέσματα από την εκπαίδευση του συστήματός μας, καθώς και χειρόγραφα κεφαλαία γράμματα, που έχει γράψει το ρομπότ NAO.

Τέλος, το Κεφάλαιο 6 περιλαμβάνει μια ανακεφαλαίωση και τα συμπεράσματα, που προέκυψαν, καθώς και περαιτέρω βελτιώσεις, που θα μπορούσαν να γίνουν στο μέλλον.

Κεφάλαιο 2

Θεωρητικό Υπόβαθρο

Στο κεφάλαιο αυτό περιγράφουμε κάποιες βασικές έννοιες, στις οποίες θα αναφερθούμε στα πλαίσια της διπλωματικής εργασίας μας. Στόχος του συγκεκριμένου κεφαλαίου είναι να κάνει την εργασία μας πιο προσιτή στο ευρύτερο κοινό, βοηθώντας τον αναγνώστη να κατανοήσει τις παρακάτω έννοιες, ώστε να μην έχει ελλείψεις σε θεωρητικό επίπεδο.

2.1 Ανθρωποειδές Ρομπότ NAO

2.1.1 Ιστορική Αναδρομή του Ρομπότ NAO

Ο καρπός ενός μοναδικού συνδυασμού μηχανικής, μηχανολογίας, ηλεκτρονικής και λογισμικού γέννησε το ρομπότ NAO, το οποίο είναι ένα αυτόνομο, προγραμματιζόμενο ανθρωποειδές ρομπότ που αναπτύχθηκε από την γαλλική εταιρεία ρομποτικής, την Aldebaran Robotics με έδρα στο Παρίσι, που μετονομάστηκε το 2015 ως SoftBank Robotics [1]. Η ανάπτυξη του ρομπότ Nao ξεκίνησε το 2004 και στις 15 Αυγούστου του 2007, το Nao αντικατέστησε το ρομπότ της Sony, Aibo, ως το ρομπότ που χρησιμοποιήθηκε στο RoboCup Standard Platform League (SPL) [2], έναν διεθνή διαγωνισμό ποδοσφαίρου ρομπότ, όπως φαίνεται και στο Σχήμα 2.1.

Σχήμα 2.1: RoboCup Standard Platform League (SPL) [3]

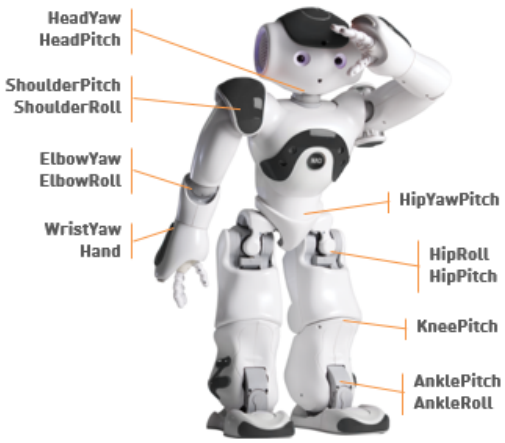
2.1.2 Περιγραφή του NAO και των ικανοτήτων του

Το Nao είναι ένα μικρό ρομπότ και το τελευταίο μοντέλο που έχει κυκλοφορήσει, το 2018, είναι το Nao Power 6 (V6). Το Nao έχει ύψος 58 εκατοστά, μήκος 27.7 εκατοστά, πλάτος 31.1 εκατοστά και το βάρος του είναι 5.5 κιλά. Το ρομπότ αυτό έχει τη δυνατότητα να περπατήσει με προεπιλεγμένη ταχύτητα 0.3 χιλιόμετρα ανά ώρα, να χορέψει, να μιλήσει 20 γλώσσες και να τις αναγνωρίσει ακούγοντας κάποια ομιλία, καθώς και να αναγνωρίσει πρόσωπα και αντικείμενα. Όλα αυτά επιτυγχάνονται με τα παρακάτω χαρακτηριστικά του.

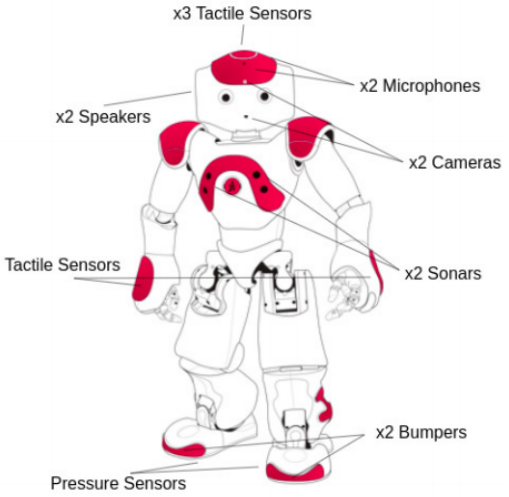
1) **Κίνηση:** Το NAO έχει 26 κινητήρες με 5 διαφορετικούς τύπους (με διαφορές στις στροφές ανά λεπτό, στη ροπή στάσης και στην ονομαστική ροπή) παρέχοντας 25 βαθμούς ελευθερίας (2 στο κεφάλι, 4 σε κάθε ώμο, 2 σε κάθε χέρι, 5 σε κάθε πόδι και 1 στην άρθρωση του κάθε ισχύου) και ανθρωποειδές σχήμα που του επιτρέπει να κινείται στον κόσμο γύρω του. Η τεχνολογία που έχει ενσωματωμένη του επιτρέπει να διατηρεί την ισορροπία του και να γνωρίζει αν στέκεται όρθιο ή ξαπλωμένο. Στο Σχήμα 2.2 φαίνονται οι θέσεις των κινητήρων του NAO.

2) **Αίσθηση:** Οι πολυάριθμοι αισθητήρες στο κεφάλι, τα χέρια και τα πόδια του, καθώς και τα σόναρ του, του επιτρέπουν να αντιλαμβάνεται το περιβάλλον και να αλληλεπιδρά με αυτό. Στο Σχήμα 2.3 φαίνονται οι θέσεις των αισθητήρων του NAO.

3) **Ομιλία και Ακοή:** Διαθέτει τέσσερα μικρόφωνα και μεγάφωνα, κι έτσι μπορεί να αλληλεπιδρά με τους ανθρώπους με έναν φυσικό τρόπο, ακούγοντας και μιλώντας.



Σχήμα 2.2: Οι θέσεις των κινητήρων του NAO [4]



Σχήμα 2.3: Οι θέσεις των αισθητήρων του NAO [5]

4) **Όραση:** Το NAO είναι εξοπλισμένο με δύο κάμερες, που μαγνητοσκοπούν σε υψηλή ανάλυση το περιβάλλον γύρω του, με αποτέλεσμα να μπορεί να αναγνωρίσει αντικείμενα, σχήματα και πρόσωπα.

5) **Σύνδεση:** Το NAO μπορεί να χρησιμοποιήσει μια σειρά από διαφορετικούς τρόπους σύνδεσης (WiFi, Ethernet) για να έχει τη δυνατότητα αυτόνομης πρόσβασης στο διαδίκτυο.

6) **Σκέψη:** Με την Τεχνητή Νοημοσύνη και τη βοήθεια του ανθρώπου, τα ρομπότ NAO είναι σε θέση να αναπαράγουν ανθρώπινη συμπεριφορά.

2.1.3 Τεχνικά Χαρακτηριστικά

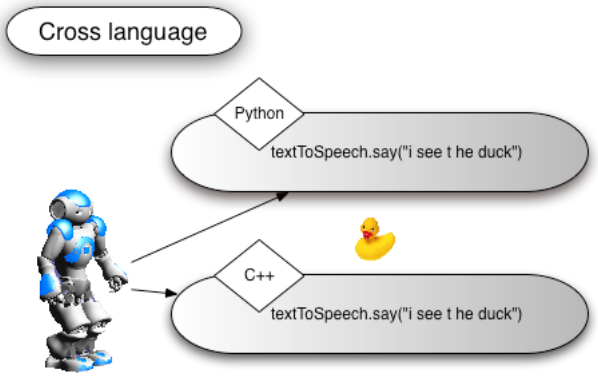
Από την πλευρά των τεχνικών χαρακτηριστικών, το Nao V6 διαθέτει 4 GB DDR3 RAM, 32 GB SSD αποθηκευτικό χώρο, επεξεργαστή Intel Atom E3845 Quad Core στα 1.91 GHz και μπαταρία λιθίου με 62.5 WattHours, που προσφέρει 90 λεπτά λειτουργίας. Επίσης, έχει IEEE 802.11g κάρτα δικτύου, η οποία μπορεί να συνδεθεί ασύρματα ή ενσύρματα με RJ45 Ethernet και χρησιμοποιεί λειτουργικό σύστημα Linux βασισμένο στο Openembedded, δίνοντάς του τη δυνατότητα να εκτελεί απλές υπολογιστικές εργασίες. Τέλος, το NAO διαθέτει δύο πανομοιότυπες κάμερες που παρέχουν ανάλυση έως 2560 * 1920 σε 1 καρέ ανά δευτερόλεπτο (fps) ή 640 * 480 στα 30 fps.

2.1.4 Το λειτουργικό σύστημα NAOqi

Το ρομπότ NAO, όπως αναφέρθηκε και πιο πάνω, έχει λειτουργικό σύστημα Linux και το βασικό λειτουργικό σύστημα του, είναι το NAOqi [6]. Το NAOqi Framework ελέγχει το ρομπότ, αφού χρησιμοποιείται για τον προγραμματισμό του NAO. Είναι αρκετά χρήσιμο σε κοινές ανάγκες ρομποτικής όπως: παραλληλισμός, πόροι, συγχρονισμός και γεγονότα. Επίσης, επιτρέπει την ομοιογενή επικοινωνία μεταξύ διαφορετικών ενοτήτων (κίνηση, ήχος, βίντεο), τον ομοιογενή προγραμματισμό και την ομοιογενή ανταλλαγή πληροφοριών. Τέλος, το NAOqi Framework :

1) είναι **cross-platform**, που σημαίνει ότι είναι δυνατό να αναπτυχθεί με αυτό σε Windows, Linux ή Mac.

2) είναι **cross-language**, με πανομοιότυπο API για C ++ και Python, που σημαίνει ότι αν δημιουργηθεί ένα C ++ ή Python module API functions μπορεί να κληθεί από παντού, όπως φαίνεται και στο Σχήμα 2.4.



Σχήμα 2.4: Cross language [7]

3) παρέχει επίσης **introspection**, που σημαίνει ότι το πλαίσιο γνωρίζει ποιες λειτουργίες είναι διαθέσιμες στις διάφορες ενότητες και πού βρίσκονται.

Κλείνοντας την ενότητα για το ρομπότ NAO θα θέλαμε να προσθέσουμε ότι σήμερα, περισσότερα από 20.000 ρομπότ χρησιμοποιούνται σε πολλούς κλάδους της βιομηχανίας, καθιστώντας το NAO το πιο χρησιμοποιούμενο ανθρωποειδές ρομπότ παγκοσμίως. Ο σχεδιασμός και οι δυνατότητές του το καθιστούν μια ελκυστική και έξυπνη πλατφόρμα που χρησιμοποιείται στην εκπαίδευση, την υγειονομική περίθαλψη, τα αεροδρόμια, σε χώρους εκμάθησης ρομποτικής και το λιανικό εμπόριο.

2.2 Choregraphe

Το Choregraphe [8] είναι μια εφαρμογή για υπολογιστές πολλαπλών πλατφορμών, που σας επιτρέπει να εκτελείτε διάφορες ενέργειες σε ένα προσομοιωμένο ρομπότ ή απευθείας σε ένα πραγματικό ρομπότ NAO, εφόσον έχετε συνδέσει, μέσω αυτής της εφαρμογής, το NAO με τον υπολογιστή σας. Μια από αυτές τις ενέργειες είναι η δημιουργία κινήσεων του NAO όπως περπάτημα, κάποιος χαιρετισμός ή ακόμη και μια αλληλουχία κινήσεων που δημιουργεί ένα χορό. Επίσης, το Choregraphe σας επιτρέπει να δημιουργείτε εφαρμογές που περιέχουν διαλόγους του NAO με τον άνθρωπο, αφού έχει τη δυνατότητα να αναγνωρίζει και να μιλάει 20 διαφορετικές γλώσσες. Ακόμη μια δυνατότα που μπορεί να χρησιμοποιηθεί σε πολλές εφαρμογές είναι η αναγνώριση προσώπων. Σύμφωνα με τα παραπάνω, μπορείτε να δημιουργήσετε ισχυρές συμπεριφορές, ώστε το ρομπότ να αλληλεπιδρά με ανθρώπους, να στέλνει e-mail και όλα αυτά χωρίς να γράψετε ούτε μία γραμμή κώδικα. Τέλος, έχετε τη δυνατότητα να εμπλουτίσετε τις συμπεριφορές του NAO μέσω του Choregraphe με το δικό σας κώδικα στη γλώσσα

προγραμματισμού Python 2.7 και μπορείτε να εξάγετε σε Python module συμπεριφορές - κινήσεις που έχετε δημιουργήσει, συνήθως με τη χρήση της μεθόδου ALProxy, που σας δίνει πρόσβαση σε όλες τις μεθόδους ή την ενότητα, στην οποία πρόκειται να συνδεθείτε.

2.3 OpenCV

Η OpenCV (Open Source Computer Vision Library) [9] είναι μια βιβλιοθήκη ανοιχτού κώδικα, που ασχολείται με τη μηχανική όραση σε πραγματικό χρόνο και τη μηχανική μάθηση. Η βιβλιοθήκη αυτή δημιουργήθηκε για να παρέχει μια κοινή υποδομή για εφαρμογές τεχνητής όρασης και να επιταχύνει τη χρήση της αντίληψης των μηχανών στα εμπορικά προϊόντα. Διαθέτει περισσότερους από 2500 βελτιστοποιημένους αλγόριθμους, οι οποίοι περιλαμβάνουν ένα ολοκληρωμένο σύνολο κλασικών και υπερσύγχρονων αλγορίθμων υπολογιστικής όρασης και μηχανικής μάθησης. Αυτοί οι αλγόριθμοι μπορούν να χρησιμοποιηθούν για τον εντοπισμό και την αναγνώριση προσώπων, την αναγνώριση αντικειμένων, την ταξινόμηση ανθρώπινων ενεργειών σε βίντεο, την παρακολούθηση κινήσεων της κάμερας, την παρακολούθηση κινούμενων αντικειμένων, την εξαγωγή τρισδιάστατων μοντέλων, εύρεση παρόμοιων εικόνων από μια βάση δεδομένων εικόνων, αφαίρεση κόκκινων ματιών από εικόνες που λαμβάνονται χρησιμοποιώντας φλας, παρακολούθηση κινήσεων των ματιών, αναγνώριση τοπίου και δημιουργία δεικτών για επικάλυψη με επαυξημένη πραγματικότητα και αρκετά άλλα στα οποία δεν θα αναφερθούμε. Υπάρχουν πάνω από 500 αλγόριθμοι και περίπου 10 φορές περισσότερες συναρτήσεις που συνθέτουν ή υποστηρίζουν αυτούς τους αλγορίθμους. Η OpenCV έχει περισσότερους από 47 χιλιάδες χρήστες και εκτιμώμενο αριθμό λήψεων άνω των 18 εκατομμυρίων. Η βιβλιοθήκη χρησιμοποιείται εκτενώς σε εταιρείες, ερευνητικές ομάδες και από κυβερνητικούς φορείς. Τέλος, είναι μια βιβλιοθήκη πολλαπλών πλατφορμών, αφού διαθέτει διεπαφές C ++, Python, Java και MATLAB και υποστηρίζει Windows, Linux, Android και Mac OS.

2.4 Αλγόριθμοι και συναρτήσεις της βιβλιοθήκης OpenCV

2.4.1 Συνάρτηση cvtColor()

Η συνάρτηση cvtColor() [10] μετατρέπει μια εικόνα εισόδου από έναν χρωματικό χώρο σε έναν άλλο. Σε περίπτωση μετασχηματισμού σε χώρο χρώματος από RGB, η σειρά των καναλιών πρέπει να προσδιορίζεται ρητά (RGB ή BGR). Η προεπιλεγμένη μορφή χρώματος στη βιβλιοθήκη OpenCV αναφέρεται συχνά ως RGB αλλά στην πραγματικότητα είναι BGR (τα byte αντιστρέφονται). Έτσι, το πρώτο byte σε μια τυπική έγχρωμη εικόνα (24-bit) θα είναι ένα στοιχείο Blue 8-bit, το δεύτερο byte θα είναι πράσινο και το τρίτο byte θα είναι κόκκινο. Οι συμβατικές περιοχές για τις τιμές καναλιών R, G και B είναι: 0 έως 255 για εικόνες CV_8U, 0 έως 65535 για εικόνες CV_16U και 0 έως 1 για εικόνες CV_32F Σε περίπτωση γραμμικών μετασχηματισμών, το εύρος δεν έχει σημασία, αλλά σε περίπτωση μη γραμμικού μετασχηματισμού, μια εικόνα RGB εισόδου θα πρέπει να ομαλοποιηθεί στο σωστό εύρος τιμών για να πάρει τα

σωστά αποτελέσματα. Τέλος, αν έχετε μια εικόνα 32-bit που μετατρέπεται απευθείας από μια εικόνα 8-bit χωρίς κλιμάκωση, τότε θα έχει το εύρος τιμών 0..255 αντί του 0..1, αυτό λύνεται εύκολα με διαίρεση των εικονοστοιχείων με το 255.

2.4.2 Canny Edge Detection

Το Canny Edge Detection [11] είναι ένας δημοφιλής αλγόριθμος ανίχνευσης άκρων και αναπτύχθηκε από τον John F. Canny. Ο αλγόριθμος αυτός έχει πολλά στάδιο όπως:

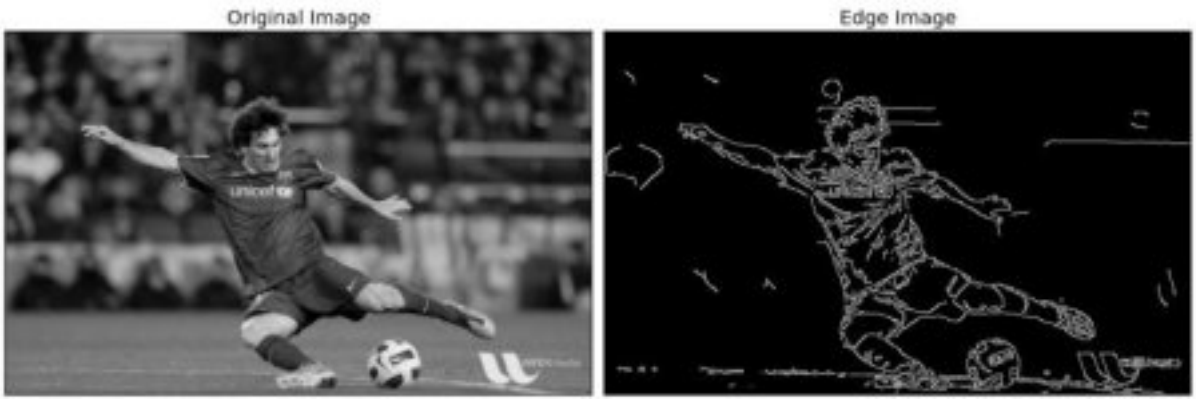
1) Μείωση θορύβου: Δεδομένου ότι η ανίχνευση ακμής είναι ευαίσθητη στον θόρυβο, το πρώτο βήμα είναι να αφαιρέσετε τον θόρυβο στην εικόνα με ένα φίλτρο Gaussian 5x5.

2) Εύρεση κλίσης έντασης της εικόνας: Στη συνέχεια, η εξομαλυμένη εικόνα φιλτράρεται με πυρήνα Sobel σε οριζόντια και κατακόρυφη κατεύθυνση για να πάρει την πρώτη παράγωγο σε οριζόντια κατεύθυνση (Gx) και κάθετη κατεύθυνση (Gy). Από αυτές τις δύο εικόνες, μπορούμε να βρούμε κλίση και κατεύθυνση ακμής για κάθε pixel ως εξής:

$$\text{Edge\_Gradient (G)} = \sqrt{G_x^2 + G_y^2}$$
$$\text{Angle } (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

3) Μη μέγιστη καταστολή: Αφού λάβετε το μέγεθος και την κατεύθυνση της κλίσης, γίνεται πλήρης σάρωση της εικόνας για να αφαιρέσετε τυχόν ανεπιθύμητα εικονοστοιχεία που ενδέχεται να μην αποτελούν το άκρο. Για αυτό, κάθε pixel ελέγχεται εάν είναι ένα τοπικό μέγιστο στη γειτονιά του προς την κατεύθυνση της κλίσης. Εν ολίγοις, το αποτέλεσμα που λαμβάνετε είναι μια δυαδική εικόνα με ¨λεπτές άκρες¨.

4) Κατώφλι υστέρησης: Αυτό το στάδιο αποφασίζει ποια είναι πραγματικά άκρα και ποια όχι και για αυτό, χρειαζόμαστε δύο τιμές κατωφλίου, minVal και maxVal. Οποιαδήποτε άκρη με κλίση έντασης μεγαλύτερη από το maxVal είναι βέβαιο ότι είναι άκρα και αυτά κάτω από το minVal σίγουρα δεν είναι άκρα κι έτσι απορρίπτονται. Όσα βρίσκονται μεταξύ αυτών των δύο ορίων ταξινομούνται άκρα ή μη άκρα με βάση τη συνδεσιμότητά τους. Εάν συνδέονται με εικονοστοιχεία ¨σίγουρης άκρης¨, θεωρούνται μέρος των άκρων, αλλιώς απορρίπτονται. Στο Σχήμα 2.5 παρουσιάζεται ένα παράδειγμα με την Original και την Edge Image με minVal ίσο με 100 και με maxVal ίσο με 200.



Σχήμα 2.5: Original and Edge Image [11]

Τέλος, η OpenCV θέτει όλα τα παραπάνω στη λειτουργία Canny() με πρώτο όρισμα την εικόνα που εισάγει ο χρήστης και δεύτερο και τρίτο τα minVal και maxVal αντίστοιχα.

2.4.3 Συνάρτηση pyrMeanShiftFiltering()

Η συνάρτηση pyrMeanShiftFiltering() [12] υλοποιεί ένα φιλτράρισμα στην εικόνα, που δίνεται ως είσοδος (πρώτο όρισμα). Σε κάθε εικονοστοιχείο (X, Y) της εικόνας εισόδου η συνάρτηση εκτελεί επαναλήψεις μέσης μετατόπισης, δηλαδή η γειτονιά εικονοστοιχείων (X, Y) στο κοινό διάστημα υπερ-χρώματος θεωρείται:

(x,y): $X - \mathtt{sp} \leq x \leq X + \mathtt{sp}$, $Y - \mathtt{sp} \leq y \leq Y + \mathtt{sp}$, $||(R,G,B) - (r,g,b)|| \leq \mathtt{sr}$

όπου (R, G, B) και (r, g, b) είναι τα διανύσματα των χρωματικών συστατικών στα (X, Y) και (x, y), αντίστοιχα (ωστόσο, ο αλγόριθμος δεν εξαρτάται από το χρωματικό χώρο που χρησιμοποιείται, έτσι μπορεί να χρησιμοποιηθεί οποιοσδήποτε χρωματικός χώρος 3 συστατικών). Στη γειτονιά βρίσκονται η μέση χωρική τιμή (X', Y') και ο μέσος χρωματικός φορέας (R', G', B') και λειτουργούν ως το κέντρο γειτονιάς στην επόμενη επανάληψη: (X,Y) (X',Y'), (R,G,B) (R',G',B'). Τέλος, μετά την επανάληψη, τα χρωματικά στοιχεία του αρχικού εικονοστοιχείου (δηλαδή, το εικονοστοιχείο από το οποίο ξεκίνησαν οι επαναλήψεις) ορίζονται στην τελική τιμή (μέσο χρώμα στην τελευταία επανάληψη): $I(X,Y) <- (R*, G*, B*)$.

2.4.4 Συνάρτηση findContours()

Η συνάρτη findContours() [13] ανακτά περιγράμματα από τη δυαδική εικόνα εισόδου και είναι αρκετά χρήσιμο εργαλείο για την ανάλυση σχημάτων, την ανίχνευση και αναγνώριση αντικειμένων. Τρία είναι τα ορίσματα, που χρησιμοποιεί η συνάρτηση αυτή, το πρώτο είναι μια εικόνα εισόδου ενός καναλιού 8-bit, το δεύτερο είναι η λειτουργία ανάκτησης περιγράμματος και το τρίτο είναι η μέθοδος προσέγγισης περιγράμματος. Υπάρχουν 5 λειτουργίες ανάκτησης περιγράμματος [14]:

- **RETR_EXTERNAL:** ανακτά μόνο τα ακραία εξωτερικά περιγράμματα και ορίζει hierarchy[i][2]=hierarchy[i][3]=-1 για όλα τα περιγράμματα.
- **RETR_LIST:** ανακτά όλα τα περιγράμματα χωρίς να δημιουργεί ιεραρχικές σχέσεις.
- **RETR_TREE:** ανακτά όλα τα περιγράμματα και ανακατασκευάζει μια πλήρη ιεραρχία ένθετων περιγραμμάτων.
- **RETR_CCOMP:** ανακτά όλα τα περιγράμματα και τα οργανώνει σε ιεραρχία δύο επιπέδων. Στο ανώτερο επίπεδο, υπάρχουν εξωτερικά όρια των στοιχείων. Στο δεύτερο επίπεδο, υπάρχουν όρια των οπών. Εάν υπάρχει ένα άλλο περίγραμμα μέσα σε μια οπή ενός συνδεδεμένου εξαρτήματος, εξακολουθεί να βρίσκεται στο ανώτερο επίπεδο.
- **RETR_FLOODFILL**

Επίσης, υπάρχουν 4 μέθοδοι προσέγγισης περιγράμματος [15]:

- **CHAIN_APPROX_NONE:** αποθηκεύει όλα τα σημεία περιγράμματος, δηλαδή, οποιαδήποτε 2 επόμενα σημεία (x1, y1) και (x2, y2) του περιγράμματος θα είναι γειτονικά είτε οριζόντια, κάθετα είτε διαγώνια, δηλαδή, max(abs (x1-x2), abs (y2-y1)) == 1.
- **CHAIN_APPROX_SIMPLE:** συμπιέζει τα οριζόντια, κάθετα και διαγώνια τμήματα και αφήνει μόνο τα τελικά σημεία τους, όπως για παράδειγμα, ένα ορθογώνιο περίγραμμα κωδικοποιείται με 4 σημεία.
- **CHAIN_APPROX_TC89_L1:** εφαρμόζει ένα κομμάτι του αλγορίθμου προσέγγισης της αλυσίδας Teh-Chin
- **CHAIN_APPROX_TC89_KCOS:** εφαρμόζει ένα κομμάτι του αλγορίθμου προσέγγισης της αλυσίδας Teh-Chin

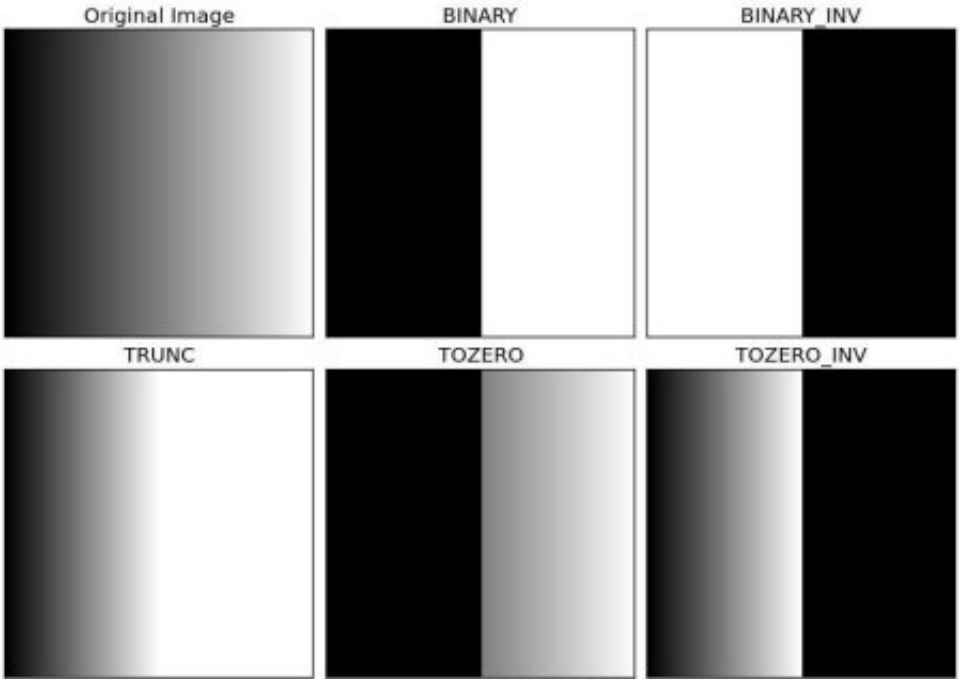
Η συνάρτηση αυτή εξάγει μια τροποποιημένη εικόνα, τα περιγράμματα και την ιεραρχία. Το περίγραμμα είναι μια λίστα Python με όλα τα περιγράμματα στην εικόνα. Κάθε μεμονωμένο περίγραμμα είναι μια συστοιχία Numpy συντεταγμένων (x, y) των οριακών σημείων του αντικειμένου.

2.4.5 Συνάρτηση threshold()

Η συνάρτηση threshold() [16] ορίζει το κάθε εικονοστοιχείο σε τιμή 0, εάν η τιμή του είναι μικρότερη από το κατώφλι που έχει οριστεί, διαφορετικά την ορίζει σε μια μέγιστη τιμή. Το πρώτο όρισμα είναι η εικόνα εισόδου, η οποία θα πρέπει να είναι μια εικόνα σε κλίμακα του γκρι. Το δεύτερο όρισμα είναι η τιμή κατωφλίου που χρησιμοποιείται για την ταξινόμηση των τιμών των εικονοστοιχείων. Το τρίτο όρισμα είναι η μέγιστη τιμή που αποδίδεται στις τιμές των εικονοστοιχείων που υπερβαίνουν το κατώφλι και το τελευταίο όρισμα της συνάρτησης αυτής είναι οι διαφορετικοί τύποι κατωφλίου. Οι τύποι κατωφλίου είναι:

cv.THRESH_BINARY
cv.THRESH_BINARY_INV
cv.THRESH_TRUNC
cv.THRESH_TOZERO
cv.THRESH_TOZERO_INV

και στο Σχήμα 2.6 φαίνονται οι διαφορές των παραπάνω τύπων. Τέλος, η μέθοδος threshold() επιστρέφει δύο εξόδους, η πρώτη είναι το κατώφλι που χρησιμοποιήθηκε και η δεύτερη έξοδος είναι η καινούρια εικόνα.



Σχήμα 2.6: Διαφορετικοί τύποι κατωφλίου [16]

2.5 Tensorflow

Το Tensorflow [17] είναι μαθηματική βιβλιοθήκη ανοιχτού κώδικα, την οποία ανέπτυξε η Google Brain, ομάδα τεχνητής νοημοσύνης της Google, αρχικά για εσωτερική χρήση. Επίσης, συνδυάζει μια σειρά μοντέλων και αλγορίθμων μηχανικής μάθησης (machine learning) και βαθιάς μάθησης (deep learning) και τα καθιστά χρήσιμα μέσω μιας κοινής μεταφοράς. Χρησιμοποιεί τη γλώσσα προγραμματισμού Python για να παρέχει ένα βολικό API front-end για την κατασκευή εφαρμογών με το πλαίσιο, ενώ εκτελεί αυτές τις εφαρμογές σε C ++ υψηλής απόδοσης. Ακόμη, μπορεί να εκπαιδεύσει και να εκτελέσει βαθιά νευρωνικά δίκτυα (Deep Neural Network) για χειρόγραφη ταξινόμηση ψηφίων, αναγνώριση εικόνας, ενσωματώσεις λέξεων, επαναλαμβανόμενα νευρωνικά δίκτυα, μοντέλα ακολουθίας σε σειρά για μηχανική μετάφραση, επεξεργασία φυσικής γλώσσας και προσομοιώσεις βασισμένες σε μερικές διαφορικές εξισώσεις. Οι προγραμματιστές έχουν τη δυνατότητα να δημιουργούν γραφήματα ροής δεδομένων - δομές που περιγράφουν τον τρόπο με τον οποίο τα δεδομένα κινούνται μέσω γραφήματος ή μια σειρά κόμβων επεξεργασίας. Κάθε κόμβος στο γράφημα αντιπροσωπεύει μια μαθηματική λειτουργία και κάθε σύνδεση ή άκρη μεταξύ των κόμβων είναι ένας πολυδιάστατος πίνακας δεδομένων. Τέλος, το Tensorflow 2.0, που κυκλοφόρησε τον Οκτώβριο του 2019, ανανέωσε το πλαίσιο με πολλούς τρόπους με βάση τα σχόλια των χρηστών, για να διευκολύνει την εργασία χρησιμοποιώντας το σχετικά απλό Keras API για εκπαίδευση μοντέλων.

2.6 Keras

Το Keras [18] είναι ένα από τα κορυφαία API νευρωνικών δικτύων υψηλού επιπέδου, είναι γραμμένο σε Python και δημιουργήθηκε για να είναι φιλικό προς το χρήστη, εύκολο να επεκταθεί και να συνεργαστεί με την Python. Το API σχεδιάστηκε για ανθρώπους, όχι για μηχανές και ακολουθεί τις βέλτιστες πρακτικές για τη μείωση του γνωστικού φορτίου. Υπάρχουν δύο βασικοί τύποι μοντέλων στο Keras: το Sequential μοντέλο, που είναι πολύ απλό (μια απλή λίστα επιπέδων), αλλά περιορίζεται σε στοίβες επιπέδων μονής εισόδου, μονής εξόδου και η Model κλάση, που χρησιμοποιείται με το λειτουργικό API, το οποίο χρησιμοποιεί τα ίδια επίπεδα με το Sequential μοντέλο, αλλά παρέχει μεγαλύτερη ευελιξία στο να τα συνδυάζει. Τέλος, τα μοντέλα Keras μπορούν να αναπτυχθούν σε ένα ευρύ φάσμα πλατφορμών, ίσως περισσότερο από οποιοδήποτε άλλο πλαίσιο βαθιάς μάθησης (deep learning).

2.7 Μηχανική Μάθηση (Machine Learning)

Η μηχανική μάθηση [19] αποτελεί ένα από τα σημαντικότερα κομμάτια της τεχνητής νοημοσύνης (AI), παρέχει στα συστήματα τη δυνατότητα αυτόματης μάθησης και βελτίωσης από την εμπειρία χωρίς να προγραμματίζονται ρητά. Η μηχανική μάθηση επικεντρώνεται στην ανάπτυξη προγραμμάτων υπολογιστών, που μπορούν να έχουν πρόσβαση σε δεδομένα και να τα χρησιμοποιούν για να μάθουν μόνοι τους. Η διαδικασία της μάθησης ξεκινά με παρατηρήσεις ή δεδομένα, όπως παραδείγματα, άμεση εμπειρία ή οδηγίες, προκειμένου να αναζητηθούν μοτίβα στα δεδομένα και να ληφθούν καλύτερες αποφάσεις στο μέλλον με βάση τα παραδείγματα που παρέχουμε. Ο βασικός στόχος είναι να επιτρέπεται στους υπολογιστές να μαθαίνουν αυτόματα χωρίς ανθρώπινη παρέμβαση ή βοήθεια και να προσαρμόζουν ανάλογα τις ενέργειες τους. Οι αλγόριθμοι μηχανικής εκμάθησης κατηγοριοποιούνται σε τρεις ευρείες κατηγορίες:

- **Εποπτευόμενοι αλγόριθμοι μηχανικής μάθησης (Supervised machine learning algorithms) [20]:** μπορούν να εφαρμόσουν ότι έχουν μάθει στο παρελθόν σε νέα δεδομένα χρησιμοποιώντας επισημασμένα παραδείγματα για την πρόβλεψη μελλοντικών γεγονότων. Ξεκινώντας από την ανάλυση ενός γνωστού συνόλου δεδομένων κατάρτισης, ο αλγόριθμος εκμάθησης παράγει μια συνάρτηση για να κάνει προβλέψεις σχετικά με τις τιμές εξόδου.

- **Μη επιτηρούμενοι αλγόριθμοι μηχανικής μάθησης (Unsupervised machine learning algorithms) [20]:** μελετούν πώς τα συστήματα μπορούν να συμπεράνουν μια συνάρτηση για να περιγράψουν μια κρυφή δομή από δεδομένα χωρίς ετικέτα. Το σύστημα δεν καταλαβαίνει τη σωστή έξοδο, αλλά διερευνά τα δεδομένα και μπορεί να εξαγάγει συμπεράσματα από σύνολα δεδομένων για να περιγράψει κρυφές δομές από δεδομένα χωρίς ετικέτα.

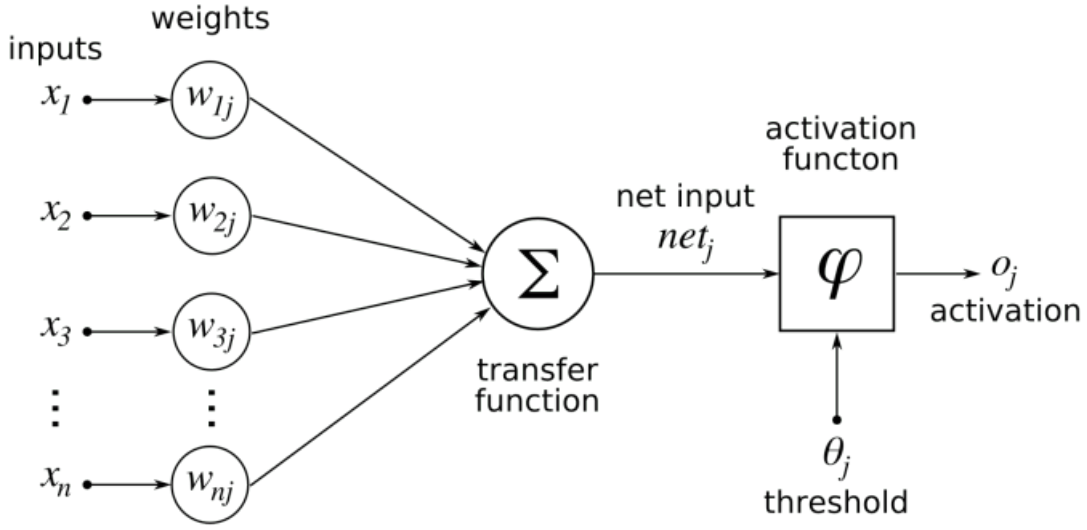
- **Αλγόριθμοι ενισχυτικής μηχανικής μάθησης (Reinforcement machine learning algorithms) [21]:** είναι μια μέθοδος μάθησης που αλληλεπιδρά με το περιβάλλον της παράγοντας ενέργειες και ανακαλύπτει λάθη ή ανταμοιβές. Η αναζήτηση δοκιμών και σφαλμάτων και η καθυστερημένη ανταμοιβή είναι τα πιο σχετικά χαρακτηριστικά της ενισχυτικής μάθησης. Αυτή η μέθοδος

επιτρέπει σε μηχανές και πράκτορες λογισμικού να προσδιορίζουν αυτόματα την ιδανική συμπεριφορά, προκειμένου να μεγιστοποιήσουν την απόδοσή τους. Απαιτείται απλή ανατροφοδότηση ανταμοιβής για να μάθει ο πράκτορας ποια ενέργεια είναι καλύτερη.

Με την εκμάθηση μηχανών, απολαμβάνουμε υπηρεσίες όπως φίλτρα ανεπιθύμητης αλληλογραφίας, αναγνώριση κειμένου και φωνής, αξιόπιστες μηχανές αναζήτησης στο διαδίκτυο και ελπίζουμε σύντομα σε αυτόνομα μέσα μεταφοράς. Συμπερασματικά, η μηχανική μάθηση επιτρέπει την ανάλυση τεράστιων ποσοτήτων δεδομένων και ο συνδυασμός της με την τεχνητή νοημοσύνη μπορεί να την κάνει ακόμη πιο αποτελεσματική στην επεξεργασία μεγάλου όγκου πληροφοριών.

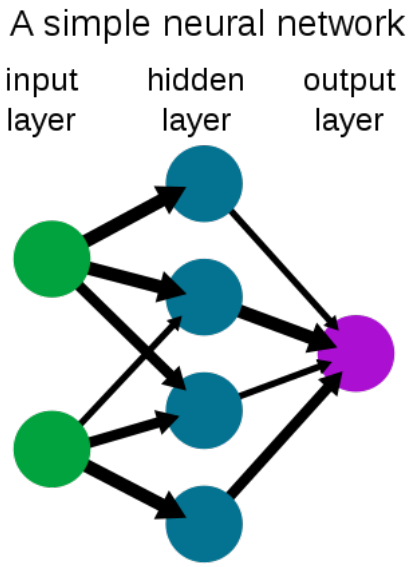
2.8 Νευρωνικά Δίκτυα (Neural Networks)

Νευρωνικό δίκτυο [22] είναι ένα κύκλωμα διασυνδεδεμένων νευρώνων, αποτελούμενο από τεχνητούς νευρώνες ή κόμβους. Τα Νευρωνικά Δίκτυα, χωρίζονται στα Βιολογικά Νευρωνικά Δίκτυα (Biological Neural Networks), που αποτελούνται από πραγματικούς βιολογικούς νευρώνες και στα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks), που κατασκευάστηκαν για την επίλυση προβλημάτων τεχνητής νοημοσύνης (AI). Πιο συγκεκριμένα, ένα νευρωνικό δίκτυο είναι μια σειρά αλγορίθμων, που προσπαθούν να αναγνωρίσουν τις υποκείμενες σχέσεις σε ένα σύνολο δεδομένων μέσω μιας διαδικασίας, που μιμείται τον τρόπο λειτουργίας του ανθρώπινου εγκεφάλου. Ακόμη, τα νευρωνικά δίκτυα αποτελούνται από ένα σύνολο νευρώνων, δηλαδή μιας μονάδας αποθήκευσης και επεξεργασίας πληροφορίας, τα τρία στοιχεία των οποίων είναι ένας αριθμός συνάψεων, ένας αθροιστής και μια συνάρτηση μεταφοράς (ενεργοποίησης) του νευρώνα, όπως παριστάνεται στο Σχήμα 2.7.



Σχήμα 2.7: Η δομή ενός τεχνητού νευρώνα [23]

Ένας νευρώνας μπορεί να έχει ένα μεγάλο αριθμό εισόδων αλλά η έξοδός του θα είναι πάντοτε μια, η οποία είναι δυνατό να αποτελεί είσοδο σε κάποιο άλλο νευρώνα, δημιουργώντας έτσι ένα δίκτυο από νευρώνες. Η σημαντικότητα των συνάψεων (βάρη) διαφέρουν σε κάθε νευρώνα, ενώ η συνάρτηση ενεργοποίησης καθορίζει την έξοδο σχετικά με τις σταθμισμένες εισόδους του νευρώνα. Επίσης, να σημειώσουμε ότι ένα σύνολο από νευρώνες με κοινά χαρακτηριστικά, ονομάζεται επίπεδο νευρώνων και μεταξύ της εισόδου και της εξόδου ενός ΝΔ μπορεί να έχουμε περισσότερα του ενός επίπεδα, τα λεγόμενα "κρυφά επίπεδα" υπολογισμού ή αλλιώς "hidden layers", αλλά και αναδρομικά στοιχεία (με τουλάχιστον ένα βρόχο ανάδρασης). Στο Σχήμα 2.8 εικονίζεται ένα απλό Νευρωνικό Δίκτυο. Τέλος, τα νευρικά δίκτυα και η βαθιά μάθηση (Deep Learning) παρέχουν προς το παρόν τις καλύτερες λύσεις σε πολλά προβλήματα, όπως στην αναγνώριση εικόνας, στην αναγνώριση ομιλίας και στην επεξεργασία φυσικής γλώσσας.

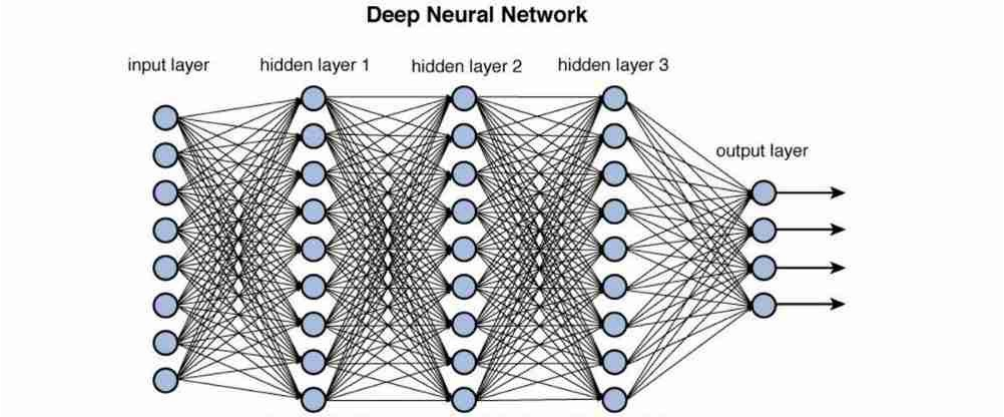


Σχήμα 2.8: Νευρωνικό Δίκτυο [24]

2.9 Βαθιά Νευρωνικά Δίκτυα (Deep Neural Networks)

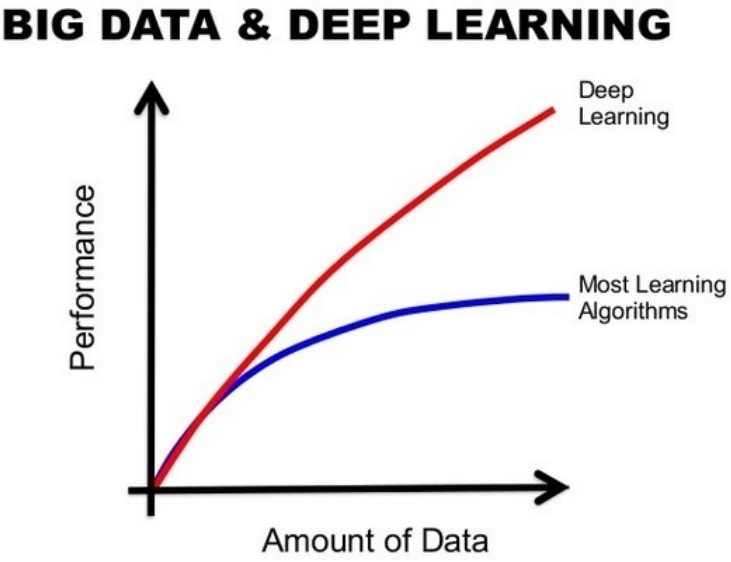
Η Βαθιά Εκμάθηση (Deep Learning) [25] ή τα Βαθιά Νευρωνικά Δίκτυα (Deep Neural Networks) , είναι μια ισχυρή κατηγορία αλγορίθμων μηχανικής μάθησης, η οποία έχει αποκτήσει πολλή έλξη τις τελευταίες δεκαετίες. Πιο συγκεκριμένα, η βαθιά μάθηση θεωρείται η εξέλιξη της μηχανικής μάθησης και χρησιμοποιεί ένα προγραμματιζόμενο νευρωνικό δίκτυο που επιτρέπει στις μηχανές να λαμβάνουν ακριβείς αποφάσεις χωρίς βοήθεια από τον άνθρωπο. Οι εφαρμογές βαθιάς μάθησης χρησιμοποιούν μια πολυεπίπεδη δομή αλγορίθμων, που ονομάζεται τεχνητό νευρωνικό δίκτυο και παριστάνεται στο Σχήμα 2.9. Ο σχεδιασμός ενός τεχνητού νευρωνικού δικτύου εμπνέεται από το βιολογικό νευρωνικό δίκτυο του ανθρώπινου εγκεφάλου, οδηγώντας σε μια διαδικασία

μάθησης, που είναι πολύ πιο ικανή από εκείνη των τυπικών μοντέλων μηχανικής μάθησης.



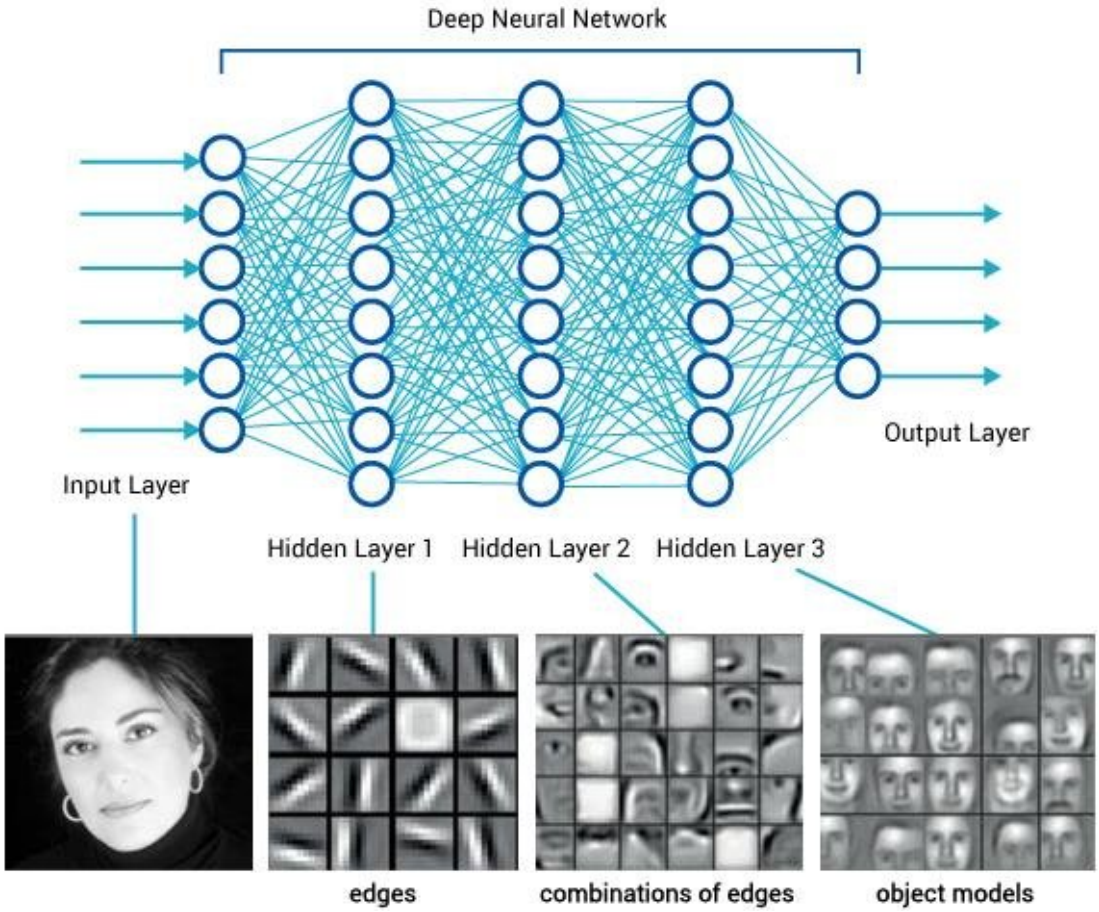
Σχήμα 2.9: Deep Learning [26]

Μέχρι πρόσφατα δεν καταφέρναμε να χειριστούμε τέτοια δίκτυα, ειδικά το τμήμα εκπαίδευσης, αλλά η διαφορά σήμερα είναι ότι έχουμε αναπτύξει αρκετά ισχυρούς υπολογιστές και διαθέτουμε μεγάλα σύνολα δεδομένων, ώστε να μπορούμε να εκμεταλλευτούμε πλήρως τις δυνατότητές τους. Ένα ακόμη σημαντικό σημείο για τα νευρωνικά δίκτυα βαθιάς μάθησης είναι η επεκτασιμότητά τους. Όσο μεγαλύτερα είναι τα δίκτυα, που κατασκευάζουμε και όσο περισσότερα δεδομένα έχουμε για να εκπαιδεύσουμε το σύστημα, τόσο αυξάνεται δραματικά η απόδοσή τους. Γενικά αυτό διαφέρει από άλλες τεχνικές μηχανικής εκμάθησης, που μετά από ένα ορισμένο σημείο φτάνουν στο ανώτατο όριο της απόδοσης τους και στο Σχήμα 2.10 βλέπουμε ένα διάγραμμα, που απεικονίζει αυτήν την έννοια.



Σχήμα 2.10: Απόδοση Deep Learning σε σύγκριση με άλλους αλγορίθμους μάθησης [27]

Ένα άλλο μεγάλο πλεονέκτημα των μοντέλων βαθιάς μάθησης, εκτός από την επεκτασιμότητα, είναι η ικανότητά τους να εξάγουν από μόνα τους χαρακτηριστικά από ανεπεξέργαστα δεδομένα. Αυτό ονομάζεται εκμάθηση χαρακτηριστικών και είναι η διαδικασία με την οποία τα βαθιά δίκτυα συνθέτουν ιεραρχίες χαρακτηριστικών, που τους επιτρέπουν να χτίσουν πολύπλοκα χαρακτηριστικά υψηλού επιπέδου συνδυάζοντας απλούστερα. Σε μια εφαρμογή αναγνώρισης προσώπου, η πρώτη εισαγωγή μπορεί να είναι μια μήτρα εικονοστοιχείων. Το πρώτο αντιπροσωπευτικό στρώμα μπορεί να αφαιρέσει τα pixel και να κωδικοποιήσει τις άκρες. Το δεύτερο στρώμα μπορεί να συνθέτει και να κωδικοποιεί διευθετήσεις άκρων. Το τρίτο στρώμα μπορεί να κωδικοποιεί μια μύτη και τα μάτια και το τέταρτο επίπεδο μπορεί να αναγνωρίσει ότι η εικόνα περιέχει ένα πρόσωπο. Στο Σχήμα 2.11 παριστάνεται η ανίχνευση προσώπου μέσα από ένα σύνολο εικόνων σε ένα βαθύ νευρωνικό δίκτυο. Κλείνοντας, είναι πολύ σημαντικό ότι μια διαδικασία βαθιάς μάθησης μπορεί να μάθει ποια χαρακτηριστικά να τοποθετήσει βέλτιστα σε ποιο επίπεδο από μόνη της.



Σχήμα 2.11: Ανίχνευση προσώπου μέσα από ένα σύνολο εικόνων [28]

2.10 Συνελικτικά Νευρωνικά Δίκτυα (Convolution Neural Networks)

Ένας από τους γνωστούς αλγορίθμους για τη μηχανική μάθηση, πιο συγκεκριμένα, τη βαθιά μάθηση, είναι ένα Convolution Neural Network (CNN ή ConvNet) [29]. Το CNN είναι ένας τύπος τεχνητού νευρωνικού δικτύου (ANN), που ειδικεύεται στην αναγνώριση εικόνων και στις εργασίες μηχανικής όρασης. Στο CNN, το μοντέλο μαθαίνει να εκτελεί εργασίες απευθείας από τα δεδομένα εικόνας, βίντεο, κείμενο ή ήχο, βρίσκει μοτίβα σε εικόνες και αναγνωρίζει αντικείμενα, πρόσωπα και σκηνές. Ακόμη, έχουν δύο κύρια μέρη:

- Τουλάχιστον ένα μηχανισμό convolution/pooling, που χωρίζει την εικόνα σε χαρακτηριστικά και την αναλύει
- Τουλάχιστον ένα fully connected layer, που παίρνει την έξοδο του convolution/pooling και προβλέπει την καλύτερη ετικέτα για την περιγραφή της εικόνας.

Από την άποψη των εργασιών ταξινόμησης εικόνων, τα CNN απαιτούν, σχετικά, λιγότερη προεπεξεργασία δεδομένων, σε σύγκριση με τους κλασικούς αλγόριθμους αναγνώρισης προτύπων, επειδή καταφέρνουν να μάθουν τα φίλτρα, που ταιριάζουν στην εξαγωγή συνόλων χαρακτηριστικών. Αυτό είναι ένα τεράστιο πλεονέκτημα για τους μηχανικούς, καθώς απαιτεί μικρότερη ανθρώπινη προσπάθεια για το σχεδιασμό και την ανάπτυξη πολύπλοκων λύσεων για τέτοιου είδους προβλήματα. Οι εξελίξεις σε GPU και παράλληλους υπολογιστές έχουν κάνει τα CNN πολύ ισχυρά και ικανά να προσφέρουν υψηλή ποιότητα στην αυτοματοποιημένη οδήγηση και αναγνώριση προσώπου, καθώς μαθαίνουν να προσδιορίζουν τις διαφορές μεταξύ ενός σήματος κυκλοφορίας και ενός πεζού.

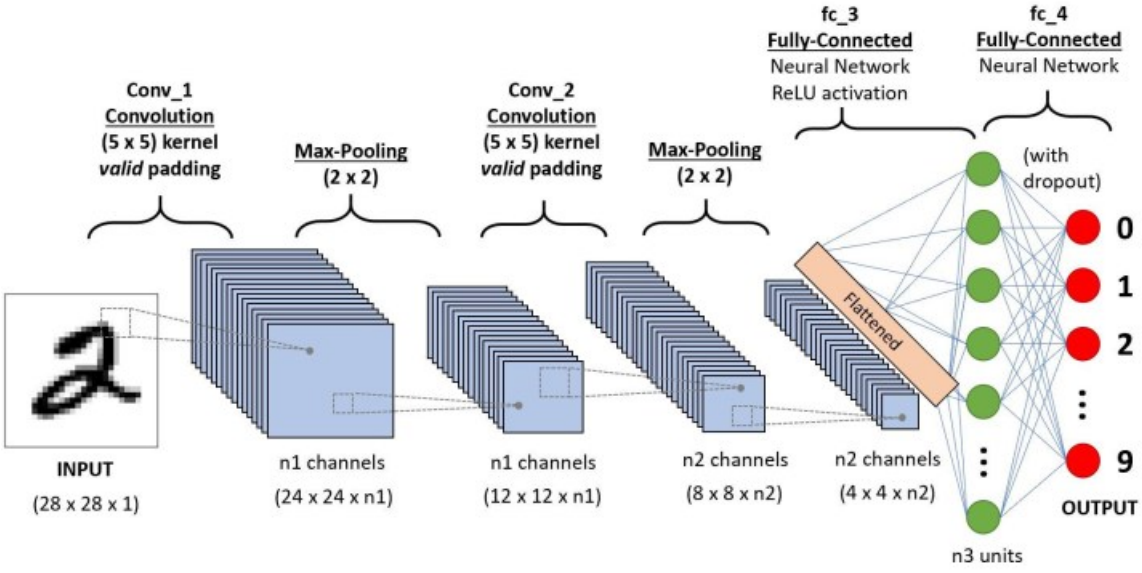
2.10.1 Αρχιτεκτονική CNN: Τύποι επιπέδων

Τα Convolutional Neural Networks έχουν διάφορους τύπους επιπέδων και παρακάτω αναφέρονται κάποιοι απ΄ αυτούς:

- **Convolutional layer:** ένα φίλτρο περνά πάνω από την εικόνα, σαρώνοντας μερικά pixel κάθε φορά και δημιουργώντας έναν χάρτη χαρακτηριστικών, που προβλέπει την κλάση στην οποία ανήκει κάθε χαρακτηριστικό.
- **MaxPooling layer (downsampling):** μειώνει την ποσότητα πληροφοριών σε κάθε χαρακτηριστικό, που λαμβάνεται στο συνελικτικό στρώμα διατηρώντας παράλληλα τις πιο σημαντικές πληροφορίες (συνήθως υπάρχουν αρκετά convolution και pooling).
- **Dropout layer:** ρυθμίζει τυχαία τα εξερχόμενα άκρα των κρυφών μονάδων (νευρώνες που αποτελούν κρυμμένα στρώματα) σε 0 σε κάθε ενημέρωση στη φάση της εκπαίδευσης.
- **Fully connected input layer (flatten):** λαμβάνει την έξοδο των προηγούμενων επιπέδων και τα μετατρέπει σ΄ ένα μόνο διάνυσμα, που μπορεί να αποτελέσει είσοδο για το επόμενο στάδιο.

- **The first fully connected layer:** λαμβάνει τις εισόδους από την ανάλυση χαρακτηριστικών και εφαρμόζει βάρη για να προβλέψει τη σωστή ετικέτα.
- **Fully connected output layer:** δίνει τις τελικές πιθανότητες για κάθε ετικέτα.

Ακολουθεί ένα παράδειγμα ενός συνελικτικού νευρωνικού δικτύου στο Σχήμα 2.12, που δείχνει τα επίπεδα που απαιτούνται για την επεξεργασία μιας εικόνας ενός χειρόγραφου ψηφίου, με τον αριθμό των pixel να υποβάλλονται σε επεξεργασία σε κάθε στάδιο. Πρόκειται για μια πολύ απλή εικόνα και οι πιο περίπλοκες εικόνες θα απαιτούσαν περισσότερα convolutional/pooling layers.



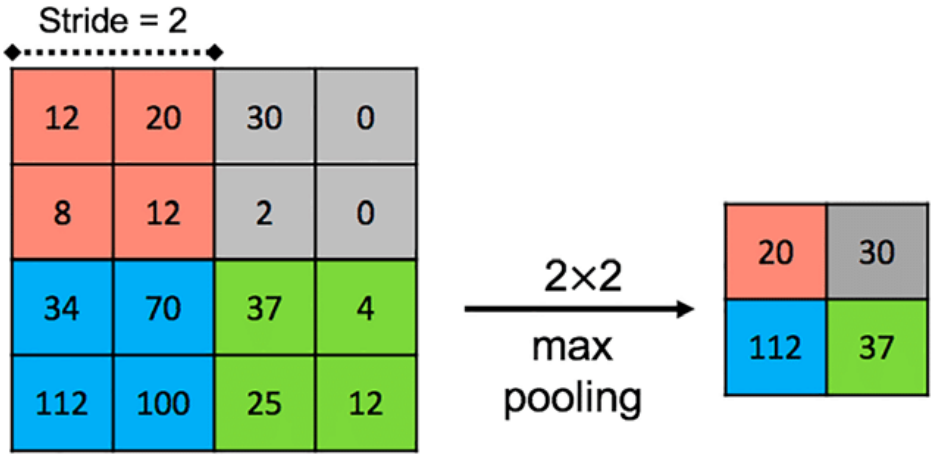
Σχήμα 2.12: Συνελικτικό Νευρωνικό Δίκτυο [29]

Convolutional layer

Ένα Convolutional layer είναι το κύριο μέρος ενός Convolutional Neural Network. Ο σκοπός αυτού του επιπέδου είναι να εφαρμοστεί στην είσοδο με τέτοιο τρόπο, ώστε να φιλτράρει ολόκληρη την είσοδο σε κάθε διάσταση προκειμένου να εξαγάγει χαρακτηριστικά. Η είσοδος σε ένα Convolutional layer είναι μια n*n*r εικόνα, όπου το n είναι το ύψος και το πλάτος της εικόνας, ενώ το r ο αριθμός των καναλιών, πχ για RGB r = 3. Το Convolutional layer έχει k φίλτρα (kernels) μεγέθους m*m*q, όπου m είναι μικρότερο από τη διάσταση της εικόνας και q μπορεί να είναι ίδιου μεγέθους με τα κανάλια r ή μικρότερου και μπορεί να ποικίλει για κάθε kernel (συνήθως 3x3 χρησιμοποιείται στις περισσότερες υλοποιήσεις μοντέλου CNN). Το μέγεθος των φίλτρων προκαλεί τοπικά συνδεδεμένη δομή, όπου καθένα συνελίσσεται με κάθε εικόνα για να παράγουν χάρτες k χαρακτηριστικών (feature maps) μεγέθους n - m + 1.

MaxPooling layer

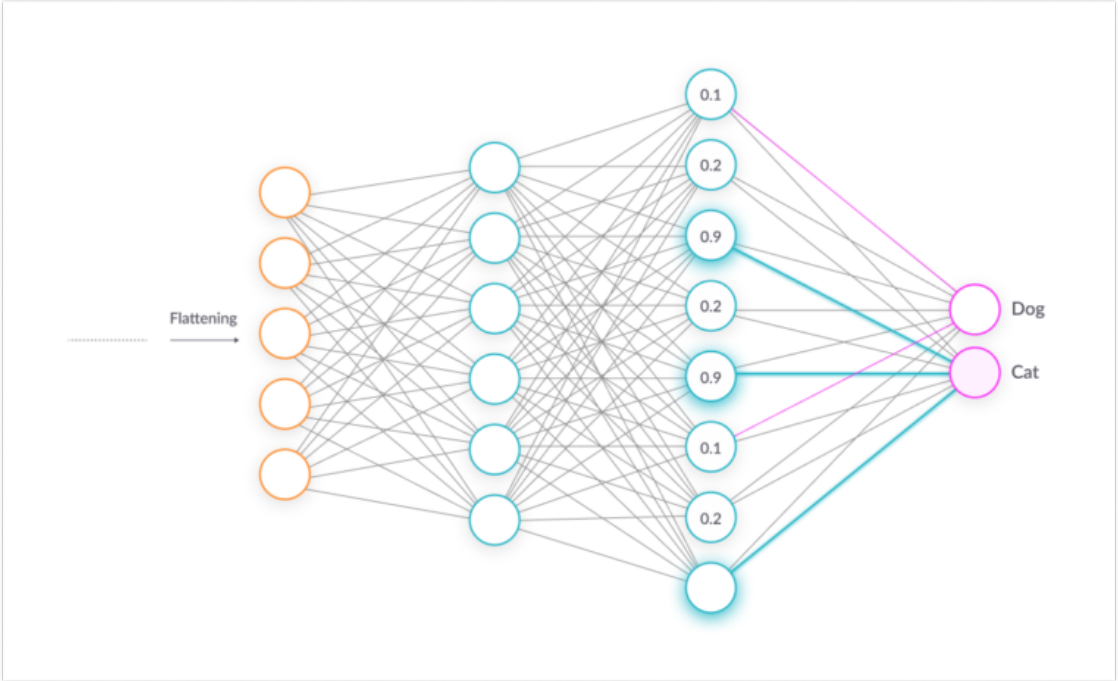
Το MaxPooling layer [30] είναι ένας τυπικός τρόπος δειγματοληψίας των εξόδων ενός επιπέδου σε ένα Νευρωνικό Δίκτυο και χρησιμοποιείται πιο συχνά σε Convolutional Neural Networks μετά από κάθε convolutional layer. Τυπικά, η λειτουργία του είναι να μειώσει σταδιακά το υπολογιστικό κόστος μειώνοντας τον αριθμό των παραμέτρων που πρέπει να μάθουμε. Ο τρόπος με τον οποίο γίνεται το MaxPooling είναι με την εφαρμογή ενός μέγιστου φίλτρου σε (συνήθως) μη επικαλυπτόμενες υποπεριοχές της αρχικής αναπαράστασης, βοηθά επίσης στην αποφυγή του overfitting του μοντέλου μας. Ακολουθεί ένα παράδειγμα στο Σχήμα 2.13 με είσοδο μεγέθους 4x4, που δείχνει ότι το MaxPooling layer θα διατηρήσει μόνο τις υψηλότερες τιμές για κάθε περιοχή και μειώνει το μέγεθος από 4x4 σε 2x2.



Σχήμα 2.13: MaxPooling [31]

Fully connected τμήμα

Ο στόχος του fully connected layer [29], πιο αναλυτικά, είναι να λάβει τα αποτελέσματα της διαδικασίας convolutional/pooling και να τα χρησιμοποιήσει για να ταξινομήσει την εικόνα σε μια ετικέτα. Η έξοδος του convolutional/pooling μετατρέπεται σε έναν μόνο φορέα τιμών, καθεμία από τις οποίες αντιπροσωπεύει την πιθανότητα ότι ένα συγκεκριμένο χαρακτηριστικό ανήκει σε μια ετικέτα. Για παράδειγμα, εάν η εικόνα είναι γάτας, χαρακτηριστικά, που αντιπροσωπεύουν πράγματα όπως μουστάκια ή γούνα θα πρέπει να έχουν μεγάλες πιθανότητες για την ετικέτα ¨γάτα¨. Στο Σχήμα 2.14 δείχνει πώς οι τιμές εισόδου ρέουν στο πρώτο στρώμα νευρώνων. Πολλαπλασιάζονται με βάρη και περνούν από μια λειτουργία ενεργοποίησης (συνήθως ReLu), όπως ακριβώς σε ένα κλασικό τεχνητό νευρωνικό δίκτυο. Στη συνέχεια περνούν προς τα εμπρός στο επίπεδο εξόδου, στο οποίο κάθε νευρώνας αντιπροσωπεύει μια ετικέτα ταξινόμησης. Το fully connected τμήμα του CNN περνά από τη δική του διαδικασία backpropagation για να προσδιορίσει τα πιο ακριβή βάρη. Κάθε νευρώνας λαμβάνει βάρη που δίνουν προτεραιότητα στην πιο κατάλληλη ετικέτα. Τέλος, οι νευρώνες «ψηφίζουν» σε κάθε μια από τις ετικέτες και ο νικητής αυτής της ψηφοφορίας είναι η απόφαση της ταξινόμησης.



Σχήμα 2.14: Το fully connected τμήμα του CNN [29]

Backpropagation

Το Backpropagation [32] είναι ένας αλγόριθμος, που χρησιμοποιείται συνήθως για την εκπαίδευση νευρωνικών δικτύων. Όταν το νευρικό δίκτυο αρχικοποιείται, ορίζονται βάρη για τα μεμονωμένα στοιχεία του, που ονομάζονται νευρώνες. Οι είσοδοι φορτώνονται, περνούν μέσω του δικτύου νευρώνων και το δίκτυο παρέχει έξοδο για καθένα, δεδομένων των αρχικών βαρών. Το Backpropagation βοηθά στη ρύθμιση των βαρών των νευρώνων, έτσι ώστε το αποτέλεσμα να πλησιάζει συνεχώς στο γνωστό πραγματικό αποτέλεσμα κι έτσι ελαχιστοποιείται το σφάλμα.

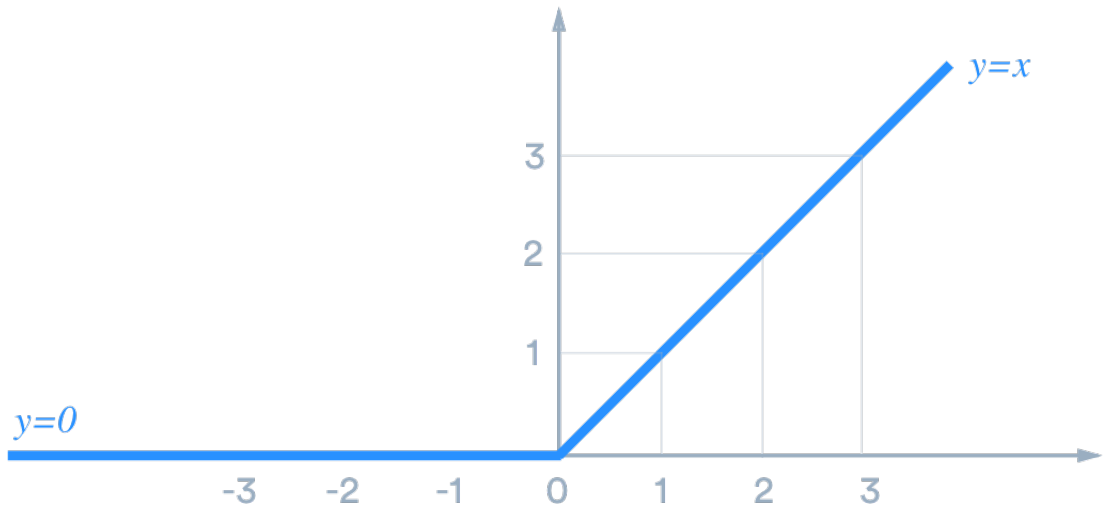
2.10.2 Λειτουργία Ενεργοποίησης Νευρωνικού Δικτύου (Neural Network Activation Function)

Οι συναρτήσεις ενεργοποίησης [33] είναι μαθηματικές εξισώσεις, που καθορίζουν την έξοδο ενός νευρωνικού δικτύου. Η συνάρτηση συνδέεται με κάθε νευρώνα στο δίκτυο και καθορίζει εάν θα πρέπει να ενεργοποιηθεί ή όχι, ανάλογα με το εάν είναι σχετική με την πρόβλεψη του μοντέλου κάθε είσοδος του νευρώνα. Οι λειτουργίες ενεργοποίησης βοηθούν επίσης στην ομαλοποίηση της εξόδου κάθε νευρώνα σε εύρος μεταξύ 0 και 1 ή μεταξύ -1 και 1. Μια επιπρόσθετη πτυχή των λειτουργιών ενεργοποίησης είναι ότι πρέπει να είναι υπολογιστικά αποτελεσματικές επειδή υπολογίζονται σε χιλιάδες ή και εκατομμύρια νευρώνες για κάθε δείγμα δεδομένων. Τα σύγχρονα νευρωνικά δίκτυα χρησιμοποιούν μια τεχνική που ονομάζεται backpropagation για να εκπαιδεύσουν το μοντέλο, το οποίο θέτει αυξημένη υπολογιστική πίεση στη συνάρτηση ενεργοποίησης. Υπάρχουν δύο τύποι λειτουργιών ενεργοποίησης, οι γραμμικοί και οι μη γραμμικοί και περίπου 7 λειτουργίες ενεργοποίησης, αλλά εμείς θα αναφερθούμε σε 2 απ΄ αυτές, στη ReLu (Rectified linear units) και στη Softmax.

ReLu (Rectified linear units)

Η συνάρτηση ReLU [33], που φαίνεται στο Σχήμα 2.15 είναι η πιο κοινή συνάρτηση ενεργοποίησης, δεδομένου ότι χρησιμοποιείται σε σχεδόν όλα τα CNN ή DL. Όταν η είσοδος είναι κάτω από το 0, η κλίση της συνάρτησης είναι 0, ενώ εάν η είσοδος είναι μεγαλύτερη από 0, το αποτέλεσμα ισούται με την είσοδο. Τα πλεονεκτήματα είναι ότι είναι υπολογιστικά αποδοτική, αφού επιτρέπει στο δίκτυο να συγκλίνει πολύ γρήγορα και είναι μη γραμμική παρόλο που μοιάζει με γραμμική συνάρτηση, έχει παράγωγη συνάρτηση και επιτρέπει backpropagation. Ωστόσο, το μειονέκτημα είναι ότι όλες οι αρνητικές τιμές γίνονται αμέσως μηδενικές, γεγονός που μειώνει την ικανότητα του μοντέλου να ταιριάζει ή να εκπαιδεύεται σωστά από τα δεδομένα, αφού οι αρνητικές τιμές δεν αντιστοιχίζονται κατάλληλα. Η μαθηματική έκφραση είναι:

$$f(x) = x^+ = \max(0, x)$$



Σχήμα 2.15: ReLu plot [34]

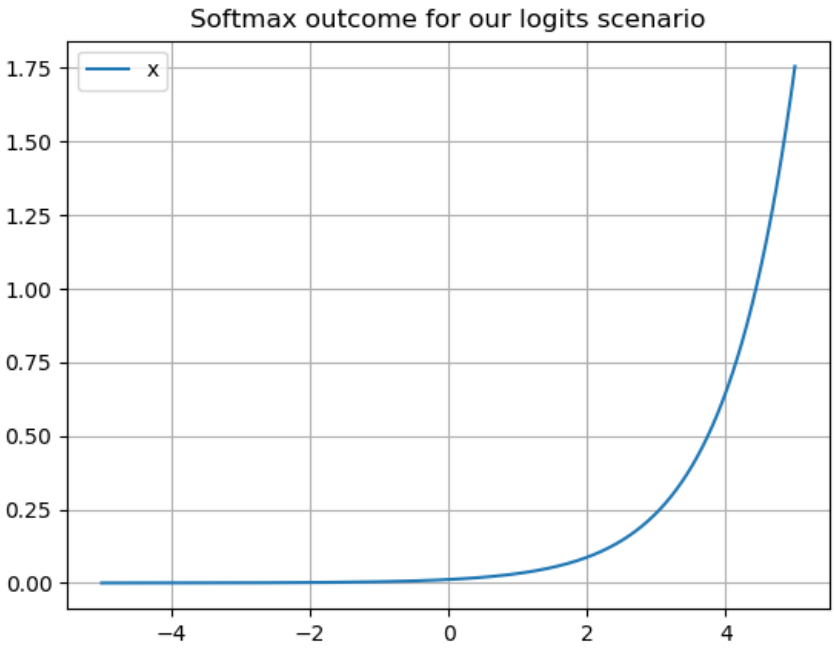
Softmax

Μια ακόμη γνωστή λειτουργία ενεργοποίησης είναι η Softmax [35], η οποία είναι μια συνάρτηση, που παίρνει ως όρισμα ένα διάνυσμα N πραγματικών αριθμών και το μετατρέπει σε κατανομή πιθανότητας, που αποτελείται από N πιθανότητες ανάλογες με τα εκθετικά των αριθμών εισόδου. Οι τιμές εξόδου κυμαίνονται μεταξύ του εύρους [0, 1] και το συνολικό άθροισμα των πιθανοτήτων τους είναι 1. Χρησιμοποιείται στις περισσότερες περιπτώσεις στο τελευταίο επίπεδο των δικτύων. Ακόμη, η συνάρτηση Softmax, που χρησιμοποιείται για το μοντέλο πολλαπλής ταξινόμησης επιστρέφει τις πιθανότητες κάθε κλάσης και η κλάση με την μεγαλύτερη πιθανότητα θα είναι η απόφαση της ταξινόμησης.

Στο Σχήμα 2.16 παριστάνεται η συνάρτηση Softmax.

Η τυπική (μονάδα) λειτουργία softmax $\sigma : \mathbb{R}^K \to \mathbb{R}^K$ ορίζεται από τον τύπο

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, \ldots, K \text{ and } \mathbf{z} = (z_1, \ldots, z_K) \in \mathbb{R}^K$$



Σχήμα 2.16: Softmax plot [36]

Κεφάλαιο 3

Περιγραφή του Προβλήματος

3.1 Αναλυτική περιγραφή του προβλήματος

Ο άνθρωπος στις μέρες μας κατασκευάζει διάφορα είδη ρομπότ και τα προγραμματίζει ώστε να εκτελούν ορισμένες ενέργειες - δραστηριότητες του. Μια από αυτές τις ενέργειες είναι το διάβασμα και η αντιγραφή. Το πρόβλημα με το οποίο έχουμε ασχοληθεί αφορά την οπτική αναγνώριση ενός χειρόγραφου κειμένου (λέξης) από το ανθρωποειδές ρομπότ NAO και την αντιγραφή αυτού του κειμένου σε έναν πίνακα ή κάποιο χαρτί με τη χρήση ενός μαρκαδόρου, τον οποίον κρατάει στο χέρι του το ρομπότ NAO. Παρατηρείτε λοιπόν, ότι το κύριο πρόβλημα μας για να λυθεί χωρίζεται σε δύο υποπροβλήματα:

1) Την Οπτική Αναγνώριση
2) Τη Γραφή

3.1.1 Η Οπτική Αναγνώριση

Το πρώτο υποπρόβλημα, που πρέπει να επιλυθεί είναι η οπτική αναγνώριση του χειρόγραφου κειμένου, το οποίο χωρίζεται σε τρεις κατηγορίες:

1) **Στην αποθήκευση εικόνας,** στην οποία το ρομπότ NAO τραβάει μια φωτογραφία μέσω της κάμερας, που διαθέτει και την αποθηκεύει σε μια συγκεκριμένη τοποθεσία, ώστε να μπορεί να επεξεργαστεί στην επόμενη κατηγορία.

2) **Στην ψηφιακή επεξεργασία εικόνας,** στην οποία το NAO επεξεργάζεται με κατάλληλους αλγορίθμους την αποθηκευμένη φωτογραφία, ανιχνεύει τα χειρόγραφα γράμματα και αποθηκεύει το κάθε γράμμα ως μια εικόνα σε μια συγκεκριμένη τοποθεσία για το δεύτερο υποπρόβλημα.

3) **Στην αναγνώριση των χειρόγραφων γραμμάτων της φωτογραφίας,** στην οποία το ρομπότ έχει εκπαιδευτεί κατάλληλα με τη χρήση των νευρωνικών δικτύων σ΄ ένα dataset με χειρόγραφα γράμματα και μέσω αυτής της εκπαίδευσης το ρομπότ μπορεί να αναγνωρίζει τα διαφορετικά γράμματα της φωτογραφίας.

3.1.2 Η Γραφή

Το δεύτερο υποπρόβλημα, που πρέπει να επιλυθεί είναι η γραφή του χειρόγραφου κειμένου. Σε αυτό το μέρος, το ρομπότ NAO δέχεται ως είσοδο τα γράμματα, που

έχουν εξαχθεί απ΄ το υποπρόβλημα της οπτικής αναγνώρισης. Στη συνέχεια, για να επιτευχθεί η γραφή πρέπει το ρομπότ να εκπαιδευτεί κατάλληλα, ώστε να μπορεί να γράφει όλα τα γράμματα της Αγγλικής αλφαβήτου. Τέλος, το NAO πρέπει να κρατάει σε κατάλληλο σημείο το μαρκαδόρο και ο πίνακας ή το χαρτί, όπου θα γράφει να βρίσκεται σε συγκέκριμένο ύψος, αφού το ρομπότ θα έχει εκπαιδευτεί για να γράφει σε μια συγκεκριμένη στάση.

3.2 Σχετικές Εργασίες

Τα τελευταία χρόνια υπάρχουν κάποιες εφαρμογές - εργασίες παρόμοιες με τη δική μας εργασία, που σχετίζονται κυρίως με το δεύτερο υποπρόβλημα μας, τη γραφή.

3.2.1 Το ρομπότ NAO γράφει ότι ακούει από τον Franck Calzada σε έναν πίνακα

Ο προγραμματιστής ρομπότ Franck Calzada έχει δημιουργήσει ένα βοηθό γραμματέα για τον απλό άνθρωπο, στο Singularity University [37], που το ρομπότ NAO μπορεί να γράψει οποιαδήποτε λέξη ακούσει κι έτσι είναι πραγματικά ικανό να κάνει κάποια δουλειά. Το NAO ζητάει να του δώσουν ένα μαρκαδόρο στο δεξί του χέρι και στη συνέχεια ρωτάει, ποια λέξη να γράψει. Μέσω του μικροφώνου δέχεται τη λέξη που ακούγεται από τον Franck Calzada και στο επόμενο βήμα το ρομπότ τη γράφει στον πίνακα. Τέλος, το ρομπότ αφού έχει γράψει τη λέξη που του ζητήθηκε αφήνει το μαρκαδόρο. Στο Σχήμα 3.1 φαίνεται το ρομπότ NAO να κρατάει το μαρκαδόρο και να γράφει μια λέξη, που έχει ακούσει πάνω σε έναν άσπρο πίνακα.

Σχήμα 3.1: Το ρομπότ NAO γράφει μια λέξη σε πίνακα [37]

3.2.2 Ο Hongyi Lin διδάσκει το ρομπότ NAO να γράφει

Ο προγραμματιστής Hongyi Lin έφτιαξε ένα σύστημα αλληλεπίδρασης ανθρώπου με ρομπότ στο University of Melbourne [38], ώστε ο χρήστης να μπορεί να διδάξει το ρομπότ NAO να γράφει χρησιμοποιώντας το στυλό πάνω στο τάμπλετ όπως φαίνεται και στο Σχήμα 3.2. Ο αλγόριθμος Q-learning χρησιμοποιείται για να βοηθήσει το ρομπότ να μάθει να γράφει. Τέλος, υπάρχουν τρείς τρόποι που μπορούν να χρησιμοποιήσουν οι χρήστες:

1) **Κινηματική:** Μίμηση της τροχιάς της γραφής

2) **Q-learning:** Χρησιμοποιώντας τον αλγόριθμο αυτόν για να μάθει τις ενέργειες γραφής

3) **Έλεγχος πληκτρολογίου:** Χρησιμοποιώντας το πληκτρολόγιο για να ελέγξει το δεξί χέρι του ρομπότ για να γράψει.

Σχήμα 3.2: Το ρομπότ NAO γράφει σε ένα τάμπλετ [38]

Κεφάλαιο 4

Η Προσέγγισή μας

Το πρόβλημα το οποίο έχουμε να υλοποιήσουμε αφορά την οπτική αναγνώριση ενός χειρόγραφου κειμένου (λέξης) από το ανθρωποειδές ρομπότ NAO και τη γραφή αυτού του κειμένου σε έναν πίνακα ή κάποιο χαρτί με τη χρήση ενός μαρκαδόρου, τον οποίον κρατάει στο χέρι του το ρομπότ NAO.

Η υλοποίηση του προβλήματός μας χωρίστηκε σε δύο υποπροβλήματα, όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο. Τα υποπροβλήματα αυτά είναι:

- **Η οπτική αναγνώριση του χειρόγραφου κειμένου από το ρομπότ NAO**
- **Η γραφή του χειρόγραφου κειμένου από το ρομπότ NAO**

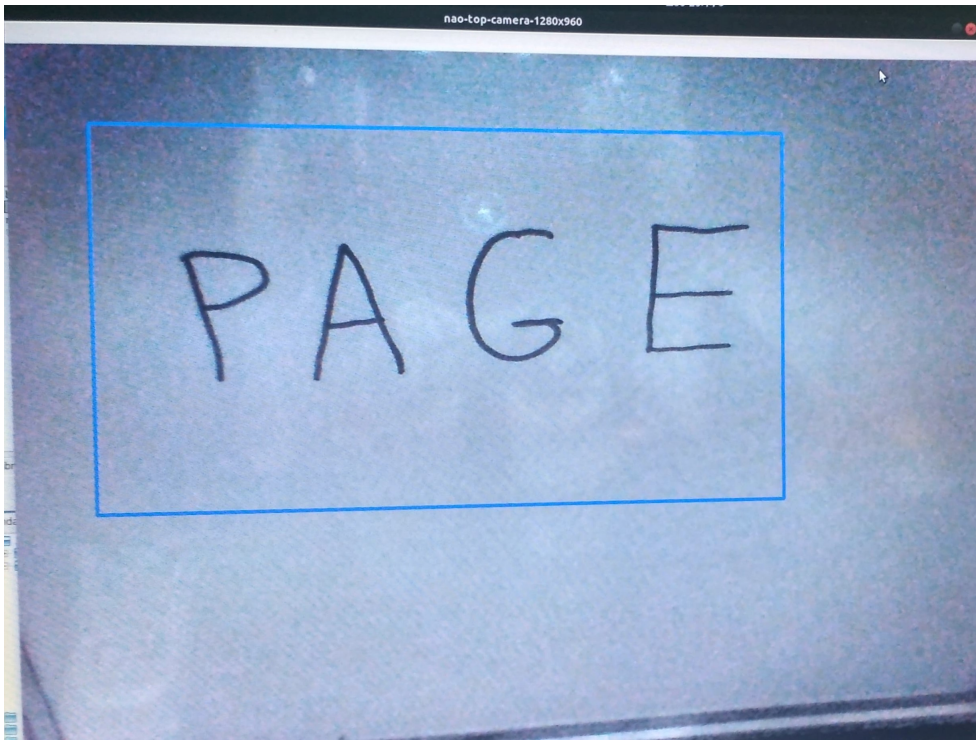
4.1 Οπτική αναγνώριση χειρόγραφου κειμένου

Προκειμένου να πετύχουμε την αποτελεσματική οπτική αναγνώριση του χειρόγραφου κειμένου χωρίσαμε το υποπρόβλημα αυτό στα παρακάτω τρία μέρη:
1) Στην αποθήκευση εικόνας
2) Στην ψηφιακή επεξεργασία εικόνας
3) Στην αναγνώριση των γραμμάτων του κειμένου

4.1.1 Αποθήκευση εικόνας

Στο πρώτο μέρος αυτού του υποπροβλήματος, το ρομπότ NAO πρέπει να τραβήξει μια φωτογραφία μέσω της κάμερας που διαθέτει και να την αποθηκεύσει σε μια συγκεκριμένη τοποθεσία, ώστε να μπορεί να επεξεργαστεί στο επόμενο μέρος. Αρχικά με τον κώδικα που γράψαμε, συνδέσαμε το ρομπότ με τον υπολογιστή μας με τις διευθύνσεις ip τους, χρησιμοποιώντας τη βοήθεια του naoqi και της μεθόδου ALProxy. Στη συνέχεια, με τη χρήση της μεθόδου ALProxy και της subscribeCamera δίνουμε την περιγραφή της κάμερας (με 0 δηλώνεται η πάνω κάμερα του ρομπότ NAO και επιλέξαμε ανάλυση 1280 x 960 pixel). Έπειτα, με τη βοήθεια των παραπάνω μεθόδων και της μεθόδου getImageRemote εμφανίζεται ένα παράθυρο, στο οποίο βλέπουμε σε πραγματικό χρόνο ότι βλέπει το ρομπότ NAO μέσω της πάνω κάμεράς του. Στην εικόνα, που βλέπει το ρομπότ μας, έχουμε προσθέσει ένα μπλε πλαίσιο με τη μέθοδο rectangle της

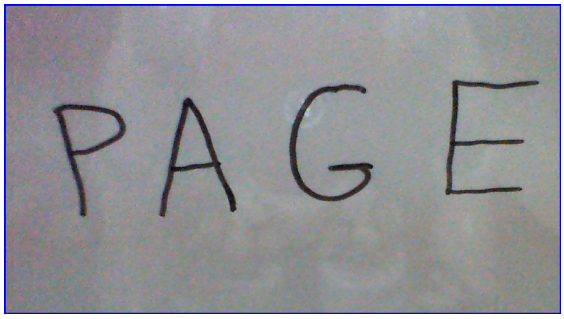
βιβλιοθήκης OpenCV, όπως φαίνεται και στο Σχήμα 4.1, ώστε ο χρήστης να τοποθετεί στο πλαίσιο αυτό, τον πίνακα ή το χαρτί με το χειρόγραφο κείμενο. Όταν τοποθετηθεί το κείμενο στο μπλε πλαίσιο τότε ο χρήστης μπορεί να πατήσει από το πληκτρολόγιο το Esc, ώστε να τραβήξει μια φωτογραφία το χειρόγραφο κείμενο και να την αποθηκεύσει σε μια συγκεκριμένη τοποθεσία με τη μέθοδο imwrite της βιβλιοθήκης OpenCV.



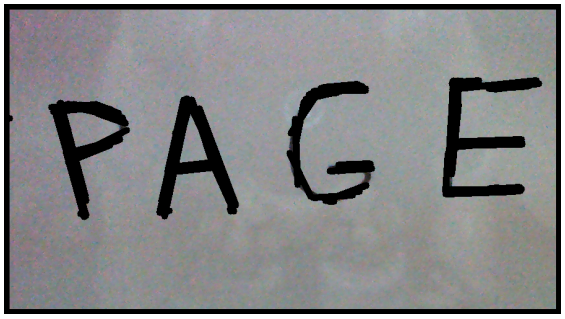
Σχήμα 4.1: Η εικόνα που βλέπει το ρομπότ NAO μέσω της πάνω κάμεράς του

4.1.2 Ψηφιακή επεξεργασία εικόνας

Στο δεύτερο μέρος του υποπροβλήματός μας, το ρομπότ πρέπει να ανιχνεύσει τα γράμματα που υπάρχουν στη φωτογραφία, που έχει αποθηκευτεί απ΄ το προηγούμενο μέρος και στη συνέχεια να αποθηκεύσει το κάθε γράμμα ως εικόνα σε μια συγκεκριμένη τοποθεσία. Προκειμένου να επιλύσουμε αυτό το μέρος, χρησιμοποιήσαμε τη βιβλιοθήκη OpenCV, αφού διαθέτει πολλούς χρήσιμους αλγορίθμους και κάποιους απ΄ αυτούς θα τους αναφέρουμε παρακάτω. Μετατρέψαμε τη φωτογραφία, που τράβηξε το NAO σε μια εικόνα στη κλίμακα του γκρι με τη βοήθεια της συνάρτησης cvtColor. Έπειτα, με το δημοφιλή αλγόριθμο ανίχνευσης άκρων, Canny Edge Detection και τον αλγόριθμο ανίχνευσης σχημάτων - γραμμών, Probabilistic Hough Transform, ανιχνεύσαμε τις γραμμές της φωτογραφίας και ορίσαμε το μέγιστο επιτρεπόμενο κενό μεταξύ σημείων στην ίδια γραμμή για τη σύνδεσή τους ίσο με 30. Στη συνέχεια, με τη χρήση της μεθόδου line κάναμε τις γραμμές, που είχαμε ανιχνεύσει πιο παχές - έντονες, ώστε να αποφύγουμε πιθανά προβλήματα λανθασμένης αναγνώρισης των γραμμάτων λόγω περιορισμένου φωτισμού. Στο Σχήμα 4.2 φαίνεται η αρχική εικόνα που φωτογράφισε το NAO, ενώ στο Σχήμα 4.3 φαίνεται η ίδια εικόνα με έντονα γράμματα.



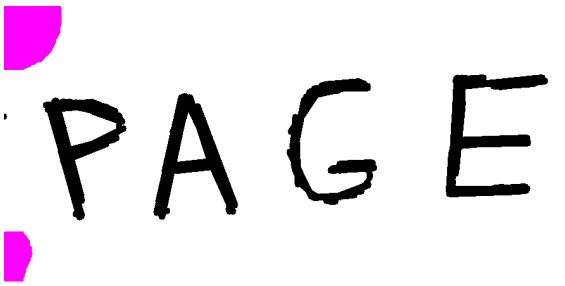
Σχήμα 4.2: Η αρχική εικόνα που φωτογράφισε το NAO



Σχήμα 4.3: Η αρχική εικόνα με έντονα γράμματα

Έπειτα, για να έχουμε πιο αποτελεσματική αναγνώριση των γραμμάτων χρειαζόταν να επεξεργαστούμε ξανά την εικόνα μας, ώστε να έχει άσπρο φόντο και μαύρα γράμματα. Σε αυτήν την επεξεργασία χρησιμοποιήσαμε τη συνάρτηση threshold της OpenCV, στην οποία σε κάθε εικονοστοιχείο εφαρμόζεται η ίδια τιμή κατωφλίου, εάν η τιμή των εικονοστοιχείων είναι μικρότερη από το όριο 127, ορίζεται σε 0 (μαύρο), αλλιώς έχει οριστεί σε μια μέγιστη τιμή 255 (άσπρο). Με την παραπάνω διαδικασία πετυχαίνουμε το άσπρο φόντο με τα μαύρα γράμματα, αλλά δημιουργείται θόρυβος, που εικονίζεται στο Σχήμα 4.4, με χρωματιστά στίγματα. Ο θόρυβος, που δημιουργήθηκε για να περιοριστεί χρησιμοποιήσαμε πριν από τη συνάρτηση threshold, το φίλτρο pyrMeanShiftFiltering της βιβλιοθήκης OpenCV, το οποίο αφαιρεί απ΄ την εικόνα μας μεγάλο μέρος απ΄ τα χρωματιστά στίγματα. Επιπλέον, για να μπορούμε να ανιχνεύσουμε τα γράμματα, κάναμε μια μικρή περικοπή της εικόνας, ώστε να αφαιρέσουμε το μαύρο περίγραμμα, όπως φαίνεται στο Σχήμα 4.5.

Σχήμα 4.4: Η εικόνα με μαύρα γράμματα και άσπρο φόντο χωρίς το φίλτρο pyrMeanShiftFiltering

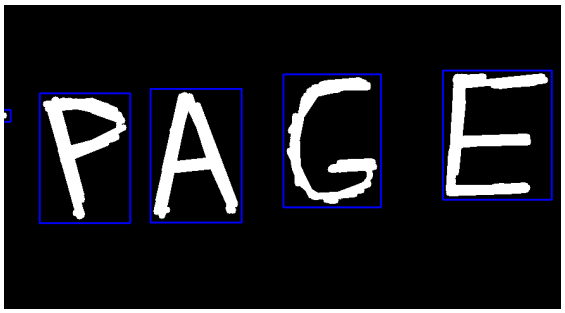


Σχήμα 4.5: Η εικόνα με μαύρα γράμματα και άσπρο φόντο με το φίλτρο pyrMeanShiftFiltering, χωρίς το μαύρο περίγραμμα

Προκειμένου να ανιχνεύσουμε τα γράμματα της φωτογραφίας, χρησιμοποιήσαμε τη μέθοδο findContours της OpenCV, η οποία ανιχνεύει την καμπύλη που ενώνει όλα τα συνεχή σημεία (κατά μήκος του ορίου), με το ίδιο χρώμα ή ένταση κι έτσι έχει τη δυνατότητα να ανιχνεύσει το κάθε γράμμα. Η μέθοδος αυτή για να λειτουργήσει σωστά απαιτεί να υπάρχει μαύρο φόντο και το αντικείμενο που θα ανιχνεύσει να είναι λευκό. Οι προϋποθέσεις αυτές μας ωθούν σε ακόμη μια επεξεργασία της εικόνας μας, στην οποία

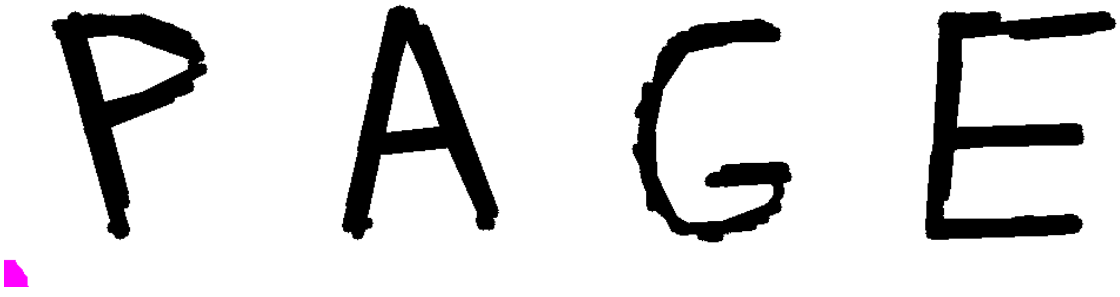
το λευκό φόντο θα γίνει μαύρο και τα μαύρα γράμματα θα γίνουν λευκά. Η επεξεργασία αυτή υλοποιήθηκε εύκολα μετατρέποντας τα μαύρα pixels σε λευκά και όλα τα υπόλοιπα σε μαύρα, όπως φαίνεται στο Σχήμα 4.6. Στη συνέχεια, μετατρέπουμε την εικόνα σε κλίμακα του γκρι και για κάθε contour, που έχει εντοπίσει ο αλγόριθμος findContours, βρίσκουμε τα ελάχιστα και τα μέγιστα (x,y) σε σχήμα ορθογωνίου παραλληλογράμμου στο δισδιάστατο χώρο με τη βοήθεια της μεθόδου boundingRect. Έπειτα, χρησιμοποιώντας τα ελάχιστα και τα μέγιστα (x,y) και με τη μέθοδο rectangle σχεδιάσαμε με μπλε χρώμα ένα ορθογώνιο περίγραμμα για την περιοχή, όπου ανήκει το κάθε contour πάνω στο Σχήμα 4.6, το οποίο παρουσιάζεται στο Σχήμα 4.7. Επιπροσθέτως, για κάθε

Σχήμα 4.6: Η εικόνα με μαύρο φόντο και λευκά γράμματα



Σχήμα 4.7: Οι σχεδιασμένες περιοχές, όπου ανήκει το κάθε γράμμα

contour περικόψαμε την αντίστοιχη σχεδιασμένη περιοχή, όπου ανήκει το κάθε γράμμα και με τη μέθοδο imwrite αποθηκεύσαμε σε συγκεκριμένη τοποθεσία τα γράμματα του Σχήματος 4.5, τα οποία απεικονίζονται στα Σχήματα 4.8, 4.9, 4.10 και 4.11. Τέλος, το όνομα που δώσαμε στις παραπάνω αποθηκευμένες εικόνες ήταν το μέγιστο x που αντιστοιχεί στο κάθε γράμμα, ώστε να αποθηκεύονται τα γράμματα με τη σωστή σειρά στο φάκελο και δεν αποθηκεύσαμε contours, που ανιχνεύονται στα άκρα της εικόνας ή με διαφορά ελάχιστου x από το μέγιστο x μικρότερη του 50, αφού σε αυτή την περίπτωση δεν υπάρχει κάποιο γράμμα, αλλά θόρυβος.



Σχήμα 4.8: Το πρώτο γράμμα που έχει αποθηκευτεί

Σχήμα 4.9: Το δεύτερο γράμμα που έχει αποθηκευτεί

Σχήμα 4.10: Το τρίτο γράμμα που έχει αποθηκευτεί

Σχήμα 4.11: Το τελευταίο γράμμα που έχει αποθηκευτεί

4.1.3 Αναγνώριση των γραμμάτων του κειμένου

Στο τρίτο μέρος αυτού του υποπροβλήματος, το NAO πρέπει να αναγνωρίσει τα γράμματα της φωτογραφίας, που έχουν αποθηκευτεί σε συγκεκριμένη τοποθεσία απ' το προηγούμενο μέρος. Προκειμένου να αναγνωρίσει αποτελεσματικά τα διαφορετικά γραμμάτα χρειάστηκε να εκπαιδεύσουμε το σύστημά μας με ένα καλο dataset με τη χρήση των νευρωνικών δικτύων και πιο συγκεκριμένα χρησιμοποιώντας Deep Neural Network.

Περιγραφή του Dataset

Αρχικά, αναζητήσαμε στο διαδίκτυο ένα Ελληνικό dataset, το οποίο θα περιείχε πολλαπλές εικόνες με κεφαλαία χειρόγραφα γράμματα του ελληνικού αλφαβήτου. Δυστυχώς, δεν βρήκαμε κάποιο dataset με τις παραπάνω προδιαγραφές, οπότε αναζητήσαμε ένα ίδιο dataset με το παραπάνω, με τη διαφορά ότι θα είναι στην Αγγλική γλώσσα. Πιο συγκεκριμένα, το dataset [39], που βρήκαμε περιείχε αρκετές εικόνες με χειρόγραφα κεφαλαία, πεζά γράμματα και αριθμούς, που είχαν γραφτεί από 3600 συγγραφείς. Το σύστημα όμως, που θα εκπαιδεύσουμε θέλουμε να περιέχει μόνο κεφαλαία χειρόγραφα γράμματα, οπότε θα πρέπει να δημιουργήσουμε ένα καινούριο dataset. Μελετώντας το dataset που βρήκαμε, παρατηρήσαμε ότι κάποιες από τις εικόνες με τα χειρόγραφα κεφαλαία γράμματα δεν ήταν ευδιάκριτες με αποτέλεσμα να μη μπορεί να εκπαιδευτεί σωστά το σύστημά μας. Έτσι, δημιουργήσαμε το καινούριο dataset επιλέγοντας τις πιο ευδιάκριτες εικόνες, περίπου 410 - 500 από κάθε κεφαλαίο χειρόγραφο γράμμα του αγγλικού αλφαβήτου, από τις οποίες κάποιες τις χρησιμοποιήσαμε για την εκπαίδευση του συστήματός μας και κάποιες άλλες για να ελέγχουμε πόσο καλό είναι το σύστημα, που έχουμε εκπαιδεύσει. Η εκπαίδευση πραγματοποιήθηκε με ένα σύνολο 11275 εικόνων, όπου το 80% χρησιμοποιήθηκε για το training set, το 10% για το test set και το 10% για το validation set.

Εκπαίδευση συστήματος για αναγνώριση κεφαλαίων χειρόγραφων γραμμάτων

Η ορθή αναγνώριση των κεφαλαίων χειρόγραφων γραμμάτων του αγγλικού αλφαβήτου από το ρομπότ NAO απαιτεί την κατάλληλη εκπαίδευση του συστήματός μας και για να το πετύχουμε αυτό χρησιμοποιήσαμε Fully connected layers σε συνελεκτικά νευρωνικά δίκτυα (CNN) κι ένα αγγλικό dataset, όπως αναφέρθηκε και πιο πάνω. Αρχικά, η πρώτη μας ενέργεια ήταν να δημιουργήσουμε ένα φάκελο στον οποίο θα υπάρχουν όλα τα κεφαλαία χειρόγραφα γράμματα του αγγλικού dataset κι έπειτα αυξήσαμε λίγο την ανάλυση των εικόνων σε 130x130 pixels, ώστε να έχουμε μεγαλύτερη ακρίβεια στην αναγνώριση των γραμμάτων κι αυτό συμβαίνει επειδή έχουμε περισσότερη πληροφορία σε pixels. Στη συνέχεια, φτιάξαμε 26 φακέλους, όπου ανήκει η κάθε κατηγορία χειρόγραφων γραμμάτων κι έτσι ορίσαμε την κλάση, στην οποία ανήκει το κάθε γράμμα. Έπειτα, με τη λειτουργία sklearn.utils.shuffle ανακατέψαμε τυχαία τις εικόνες με τις αντίστοιχες κλάσεις τους και με τη συνάρτηση train_test_split χωρίσαμε τυχαία όλες τις εικόνες του dataset στο train set και στο test set (10% του dataset). Ακόμη, κανονικοποίησαμε τις εικόνες διαιρώντας τις με το 255 κι έτσι ολοκληρώθηκε η προετοιμασία των δεδομένων, ώστε να προχωρήσουμε στην εκπαίδευση του συστήματος.

Προκειμένου να ολοκληρώσουμε το κομμάτι της εκπαίδευσης έπρεπε να εκτελέσουμε αρκετά πειράματα με διαφορετικές παραμέτρους σε διαφορετικές αρχιτεκτονικές νευρωνικών δικτύων χρησιμοποιώντας τη βοήθεια της μαθηματικής βιβλιοθήκης Tensorflow και του Keras. Οι διαφορετικοί παράμετροι, που συνθέτουν ένα μοντέλο είναι:

1) το kernel_size των Convolutional layer: 2 ακέραιοι αριθμοί, που καθορίζουν το ύψος και το πλάτος του 2D παραθύρου συνέλιξης.

2) το batch_size: ο αριθμός των δειγμάτων απ΄ το σύνολο δεδομένων εκπαίδευσης, που εκπαιδεύουν το δίκτυο μας

3) το epoch: αναφέρεται σε έναν κύκλο μέσω του πλήρους συνόλου δεδομένων εκπαίδευσης

4) ο αριθμός των νευρώνων των Convolutional layer

5) ο αριθμός των νευρώνων των Dense layer.

Η αξιολόγηση των μοντέλων, που προέκυψαν έγινε με ένα νέο τεστ σετ απ΄ τον πραγματικό κόσμο, που περιείχε 200 εικόνες με κεφαλαία χειρόγραφα γράμματα της αγγλικής αλφαβήτου, που είχαμε γράψει σ΄ έναν άσπρο πίνακα, τα φωτογράφισε και τα αποθήκευσε ως εικόνες το ρομπότ NAO.

Αρχιτεκτονικές συνελικτικών νευρωνικών δικτύων

Η πρώτη αρχιτεκτονική, που εφαρμόσαμε είχε 2 Convolutional layer, όπου μετά από το καθένα προσθέσαμε ένα Activation layer ReLU (Rectified Linear Unit) κι ένα MaxPooling layer 2x2. Έπειτα, προσθέσαμε 1 Flatten layer και 3 Dense layer, όπου μετά απ΄ το καθένα, στα πρώτα δύο, προσθέσαμε ένα Activation layer ReLU, και μετά το τρίτο ένα Activation layer Softmax. Η αξιολόγηση των μοντέλων έγινε με το νέο τεστ σετ και το ποσοστό ακρίβειας του καλύτερου μοντέλου αυτής της αρχιτεκτονικής είναι 79%. Στη συνέχεια, για να βελτιώσουμε την ακρίβεια της παραπάνω αρχιτεκτονικής διπλασιάσαμε τον αριθμό των Epochs από 10 σε 20, προσθέσαμε μετά απ΄ το δεύτερο Convolutional layer ένα Dropout layer και μετά απ΄ το καθένα απ΄ τα πρώτα δύο Dense layer ένα Dropout layer, τα οποία αύξησαν τη μέγιστη ακρίβεια των προηγούμενων μοντέλων κατα 8%.

Η δεύτερη αρχιτεκτονική, που εφαρμόσαμε είχε 4 Convolutional layer, όπου μετά απ΄το καθένα προσθέσαμε ένα Activation layer ReLU, ένα MaxPooling layer 2x2 και στα τελευταία 3 Convolutional layer ένα Dropout layer. Ακόμη, προσθέσαμε 1 Flatten layer και 3 Dense layer, όπου μετά απ΄ το καθένα, στα πρώτα δύο, προσθέσαμε ένα Activation layer ReLU κι ένα Dropout layer και μετά το τρίτο ένα Activation layer Softmax. Η αξιολόγηση των μοντέλων έγινε με το νέο τεστ σετ και το ποσοστό ακρίβειας του καλύτερου μοντέλου αυτής της αρχιτεκτονικής είναι 89%.

Η τρίτη αρχιτεκτονική, που εφαρμόσαμε είχε 6 Convolutional layer, όπου μετά απ΄το καθένα προσθέσαμε ένα Activation layer ReLU, ένα MaxPooling layer 2x2 στα πρώτα 4 Convolutional layer κι ένα Dropout layer στα τελευταία 5. Ακόμη, προσθέσαμε 1 Flatten layer και 4 Dense layer, όπου μετά απ΄ το καθένα, στα πρώτα τρία, προσθέσαμε ένα Activation layer ReLU κι ένα Dropout layer και μετά το τελευταίο ένα Activation layer Softmax. Η αξιολόγηση των μοντέλων έγινε με το νέο τεστ σετ και το ποσοστό ακρίβειας του καλύτερου μοντέλου αυτής της αρχιτεκτονικής είναι 93%.

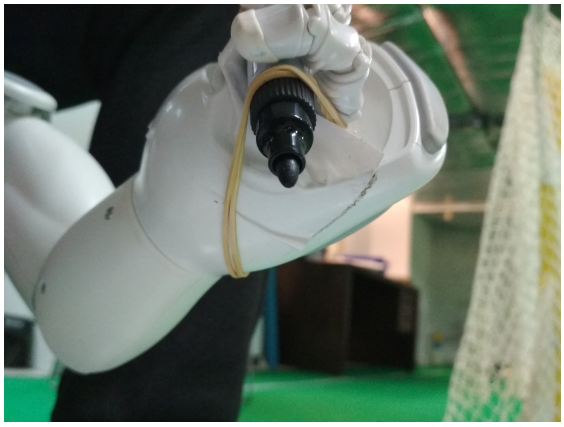
Η τελευταία αρχιτεκτονική, που εφαρμόσαμε ήταν παρόμοια με την τρίτη αρχιτεκτονική με τη μόνη διαφορά ότι αφαιρέσαμε το τρίτο Dense, Activation και Dropout layer. Η αξιολόγηση των μοντέλων έγινε με το νέο τεστ σετ και το ποσοστό ακρίβειας του καλύτερου μοντέλου αυτής της αρχιτεκτονικής είναι 96%.

Ακόμη, η κλάση Sequential μας βοήθησε να υλοποιήσουμε την εκπαίδευση του συστήματός μας χρησιμοποιώντας κάποιες συναρτήσεις της, όπως η συνάρτηση add για να προσθέτουμε το κάθε layer, η συνάρτηση compile με ορίσματα: loss function= 'categorical_crossentropy', optimizer= 'adam', list of metrics= ['accuracy'], η συνάρτηση fit με ορίσματα: X_train (οι εικόνες για εκπαίδευση), Y_train (οι κλάσεις, που αντιστοιχούν σε κάθε εικόνα εκπαίδευσης), batch_size= 16, epochs= 30, verbose=1, validation_split=0.1, callbacks=[tensorboard] και η συνάρτηση save, που αποθηκεύει το μοντέλο.

Τέλος, να επισημάνουμε ότι σε κάθε αρχιτεκτονική τα Dropout layer είχαν ως παράμετρο τυχαιότητας το 0.2 και το τελευταίο Dense layer είχε 26 νευρώνες, που συμβολίζουν τον αριθμό των κλάσεων για το κάθε χειρόγραφο κεφαλαίο γράμμα της αγγλικής αλφαβήτου. Οι γραφικές παραστάσεις, που δείχνουν πόσο καλά έχει εκπαιδευτεί το σύστημα και τα αποτελέσματα των διαφορετικών μοντέλων των παραπάνω αρχιτεκτονικών παρουσιάζονται στο κεφάλαιο Αποτελέσματα.

4.2 Γραφή χειρόγραφου κειμένου

Σκοπός αυτού του υποπροβλήματος είναι η γραφή των κεφαλαίων χειρόγραφων γραμμάτων σ΄ έναν άσπρο πίνακα, που βρίσκεται σε συγκεκριμένο ύψος χρησιμοποιώντας ένα μαρκαδόρο, που έχει τοποθετηθεί σε κατάλληλο σημείο στο χέρι του ρομπότ. Προκειμένου να επιλύσουμε το παραπάνω υποπρόβλημα εργαστήκαμε αποκλειστικά με το ρομπότ NAO και το πρόγραμμα Choregraphe, το οποίο μας επιτρέπει να συνδέσουμε το NAO με τον υπολογιστή μας, ώστε να μπορούμε να το χειριστούμε απ΄ αυτόν. Αρχικά, το πρώτο βήμα που έπρεπε να ακολουθήσουμε ήταν να τοποθετήσουμε το μαρκαδόρο σε κατάλληλο σημείο στο αριστερό χέρι του NAO και για ασφάλεια χρησιμοποιήσαμε δύο λαστιχάκια, ώστε να μην μπορεί να μετακινηθεί ο μαρκαδόρος, όπως φαίνεται από τρεις διαφορετικές οπτικές γωνίες στα Σχήματα 4.12, 4.13, 4.14. Στη συνέχεια, τοποθετήσαμε τον άσπρο πίνακα πάνω σε 5 προστατευτηκά, όπως παρουσιάζεται στο Σχήμα 4.15, για να είναι σε κατάλληλο ύψος, ώστε να μπορούμε να εκπαιδεύσουμε το NAO στη γραφή των γραμμάτων της αγγλικής αλφαβήτου.



Σχήμα 4.12: Το κάτω μέρος του αριστερού χεριού

Σχήμα 4.13: Το πλαϊνό μέρος του αριστερού χεριού

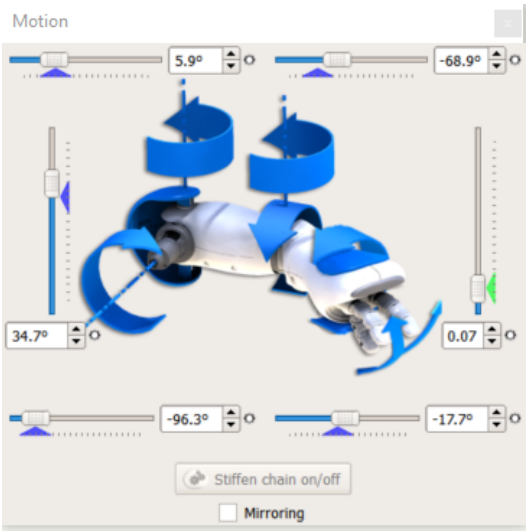
Σχήμα 4.14: Το πάνω μέρος του αριστερού χεριού

Σχήμα 4.15: Το ύψος που βρίσκεται ο πίνακας χρησιμοποιώντας τα 5 προστατευτικά

Εκπαίδευση συστήματος για γραφή κεφαλαίων γραμμάτων

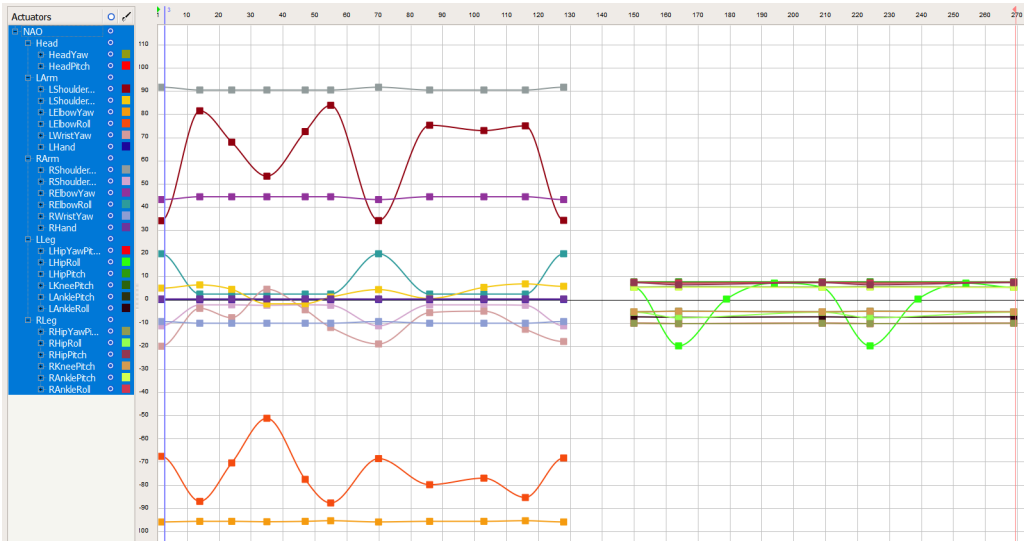
Η γραφή των κεφαλαίων γραμμάτων του λατινικού αλφαβήτου απαιτεί την κατάλληλη εκπαίδευση του συστήματός μας και για να το πετύχουμε αυτό χρησιμοποιήσαμε το πρόγραμμα Choregraphe. Με το Choregraphe έχουμε τη δυνατότητα να δημιουργήσουμε ένα Timeline module, στο οποίο αποθηκεύουμε διάφορες πόζες, οι οποίες συνθέτουν μια κίνηση. Οι μοίρες στις οποίες βρίσκονται οι αρθρώσεις του NAO σε μια συγκεκριμένη χρονική στιγμή αποτελούν μια πόζα και μια κίνηση μπορεί να θεωρηθεί ως ο σχεδιασμός ενός κεφαλαίου χειρόγραφου γράμματος. Προκειμένου να εκπαιδεύσουμε το ρομπότ μας να είναι ικανό να γράψει, δεν χρησιμοποιήσαμε την αντίστροφη κινηματική αλλά ένα διαφορετικό τρόπο μάθησης πιο οικείο σε παιδιά. Πιο

συγκεκριμένα, καθοδηγήσαμε χειροκίνητα τις αρθρώσεις του βραχίονα του NAO και καταγράψαμε τις χρονισμένες τροχιές των αρθρώσεών του. Ουσιαστικά, είχαμε το NAO σε όρθια στάση και με το κουμπί Stiffen chain on/off του Σχήματος 4.16 χαλαρώνουμε τις αρθρώσεις στο αριστερό βραχίονά του, ώστε να μπορούμε να αλλάξουμε τις μοίρες στις αρθρώσεις αυτές και να σχηματίσουμε μια πόζα, που θα μας οδηγήσει στο σχεδιασμό του πρώτου κεφαλαίου γράμματος. Έπειτα, αφού έχουμε τοποθετήσει το βραχίονα στις μοίρες που χρειάζεται για την πρώτη πόζα ξανά χρησιμοποιούμε το κουμπί Stiffen chain on/off για να κλειδώσουμε τις αρθρώσεις και αποθηκεύουμε την πόζα αυτή επιλέγοντας το Store joints in keyframe $->$ Arms. Όμως, ο τρόπος αυτός δεν ήταν πάντα ικανοποιητικός, αφού σε ορισμένες περιπτώσεις είχαμε απότομες μεταβολές κι έτσι δεν είχαμε τα επιθυμητά αποτελέσματα. Προκειμένου να αποφύγουμε τις απότομες μεταβολές, προσθέσαμε κάποιες ενδιάμεσες πόζες ή μεγαλώναμε τη διάρκεια του χρόνου από τη μια πόζα στην επόμενη, ώστε να έχουμε πιο ομαλή μεταβολή. Με την ίδια διαδικασία αποθηκεύοντας αρκετές πόζες, συνθέτουμε την κίνηση για τη γραφή του πρώτου κεφαλαίου γράμματος και την αποθηκεύουμε ως ένα Timeline module. Στη συνέχεια, εκτελέσαμε ξεχωριστές διαδικασίες για να φτιάξουμε όλα τα κεφαλαία γράμματα της αγγλικής αλφαβήτου και παρατηρήσαμε ότι το ρομπότ θα γράφει συνέχεια στο ίδιο σημείο. Έτσι, σκεφτήκαμε να υλοποιήσουμε και μια κίνηση προς τα δεξιά, μετατοπίζοντας τη θέση του NAO κάθε φορά που θα γράφει ένα γράμμα. Η υλοποίση αυτή δεν ήταν εύκολη, αφού όταν προσπαθούμε να προσθέσουμε κάποια κίνηση στα πόδια, το ρομπότ μπορεί να χάσει την ισορροπία του και να πέσει λόγω της βαρύτητας. Με αρκετές προσπάθειες καταφέραμε να δημιουργήσουμε μια κίνηση δύο βημάτων προς τα δεξιά και την προσθέσαμε στο Timeline module του κάθε γράμματος. Στο Timeline module του κάθε γράμματος επιλέξαμε το Export motion to clipboard $->$ Python $->$ bezier κι έτσι εξάγουμε για το κάθε γράμμα ένα Python module, το οποίο περιέχει σε κώδικα τις πόζες που έχουμε αποθηκεύσει μέσω του Choregraphe. Τέλος, τα γράμματα που παίρνουν πιο πολύ χρόνο να γραφούν και μας δυσκόλεψαν πιο πολύ στην εκπαίδευση είναι αυτά που αποτελούνται από περισσότερες πόζες, έχουν καμπύλες και για να γραφούν πρέπει να σηκώσουμε τον μαρκαδόρο.

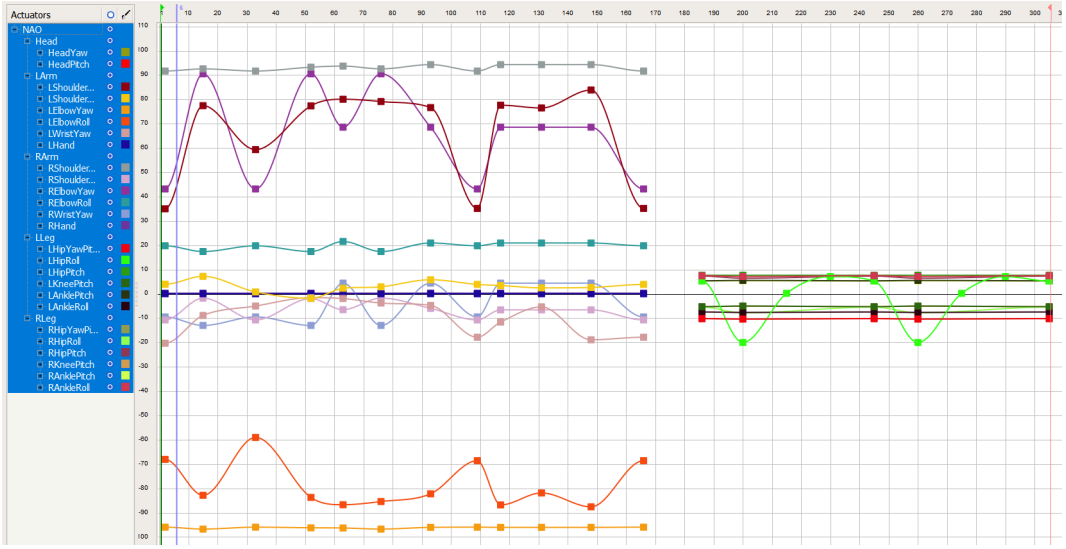


Σχήμα 4.16: Το κουμπί Stiffen chain on/off χαλαρώνει ή σκληραίνει τις αρθρώσεις του ρομπότ NAO

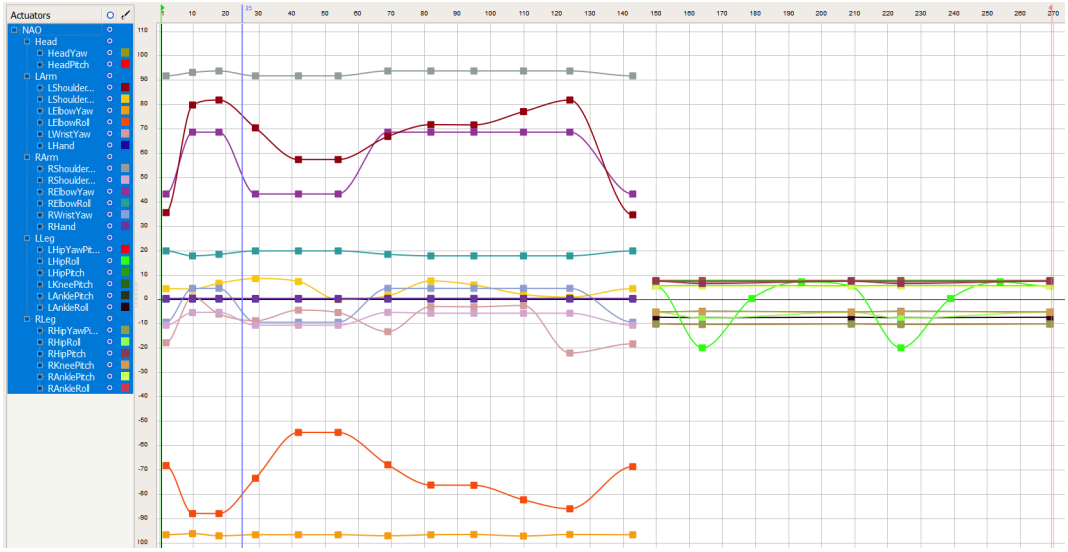
Ακολουθούν οι κυματομορφές με τις αλλαγές που γίνονται στις γωνίες των αρθρώσεων ως προς τον χρόνο, καθώς μεταβαίνει από πόζα σε πόζα και εικονίζονται στο Σχήμα 4.17 για το A, στο Σχήμα 4.18 για το Q, στο Σχήμα 4.19 για το R, και στο Σχήμα 4.20 για το S.



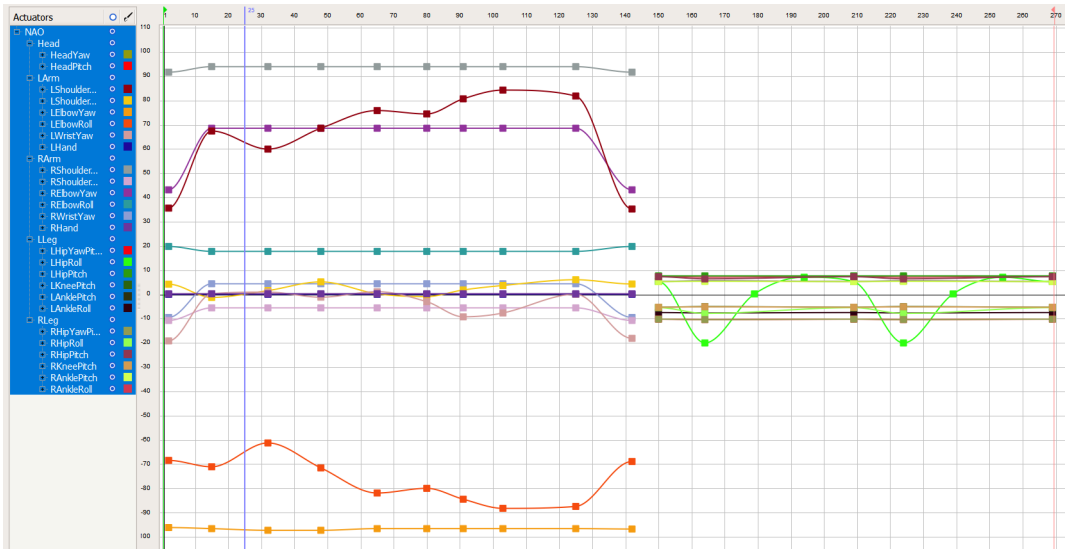
Σχήμα 4.17: Κυματομορφή του A



Σχήμα 4.18: Κυματομορφή του Q



Σχήμα 4.19: Κυματομορφή του R



Σχήμα 4.20: Κυματομορφή του S

4.3 Σύνδεση των υποπροβλημάτων

Η σύνδεση των υποπροβλημάτων της οπτικής αναγνώρισης και της γραφής του χειρόγραφου κειμένου είναι το τελευταίο κομμάτι της προσέγγισής μας. Ουσιαστικά, η σύνδεση των υποπροβλημάτων πραγματοποιήθηκε σε ένα Python module, στο οποίο καλούμε στην αρχή τα Python module takephoto (αποθήκευση εικόνας) και segm_code (ψηφιακή επεξεργασία εικόνας) και συμπεριλάβαμε το μοντέλο που είχαμε εκπαιδεύσει στο τρίτο μέρος της οπτικής αναγνώρισης. Έπειτα, προσπελαύνουμε το φάκελο με τις εικόνες με τα κεφαλαία χειρόγραφα γράμματα, που είχαν αποθηκευτεί σε συγκεκριμένη τοποθεσία από το δεύτερο μέρος της οπτικής αναγνώρισης, τα ταξινομούμε σε αύξουσα σειρά, ώστε να τα διαβάζουμε με τη σωστή σειρά και χρησιμοποιώντας το μοντέλο μας αναγνωρίζουμε το κάθε γράμμα και τα αποθηκεύουμε σε μια λίστα. Τέλος, για κάθε γράμμα που έχουμε προσθέσει στη λίστα μας καλούμε το αντίστοιχο Python module

κι έτσι το ρομπότ γράφει στον άσπρο πίνακα τη λέξη, που είχαμε γράψει στην αρχή.

4.4 Η Υλοποίησή μας

Προκειμένου να υλοποιήσουμε το πρόβλημά μας εργαστήκαμε στην πιο δημοφιλή διανομή των Linux τα Ubuntu 18.04 στο περιβάλλον του Visual Studio Code με τη γλώσσα προγραμματισμού Python 2.7, αφού αυτή είναι η τελευταία έκδοση με την οποία είναι συμβατή το ρομπότ NAO και το πρόγραμμα Choregraphe. Η εκπαίδευση του συστήματος έγινε στα Windows 10 χρησιμοποιώντας τη γλώσσα προγραμματισμού Python 3.5.2, αφού με αυτήν την έκδοση είναι συμβατή η μαθηματική βιβλιοθήκη Tensorflow 2.1.0.

Κεφάλαιο 5

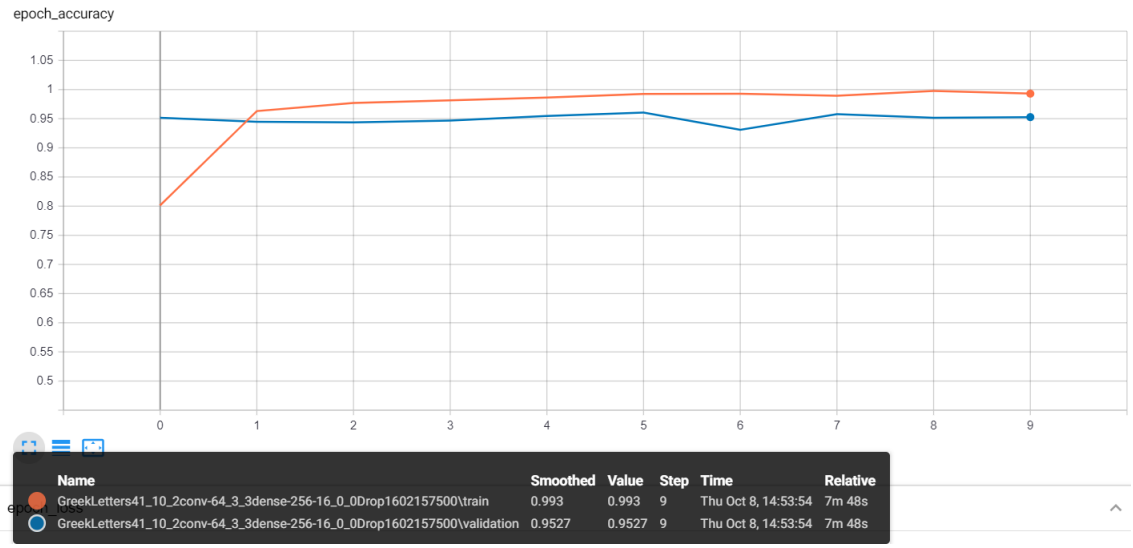
Αποτελέσματα

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τις γραφικές παραστάσεις από την εκπαίδευση του συστήματός μας για την αναγνώριση των γραμμάτων και τα αποτελέσματα διαφορετικών μοντέλων τεσσάρων αρχιτεκτονικών, καθώς και εικόνες με χειρόγραφα κεφαλαία γράμματα, που έχει γράψει το ρομπότ NAO.

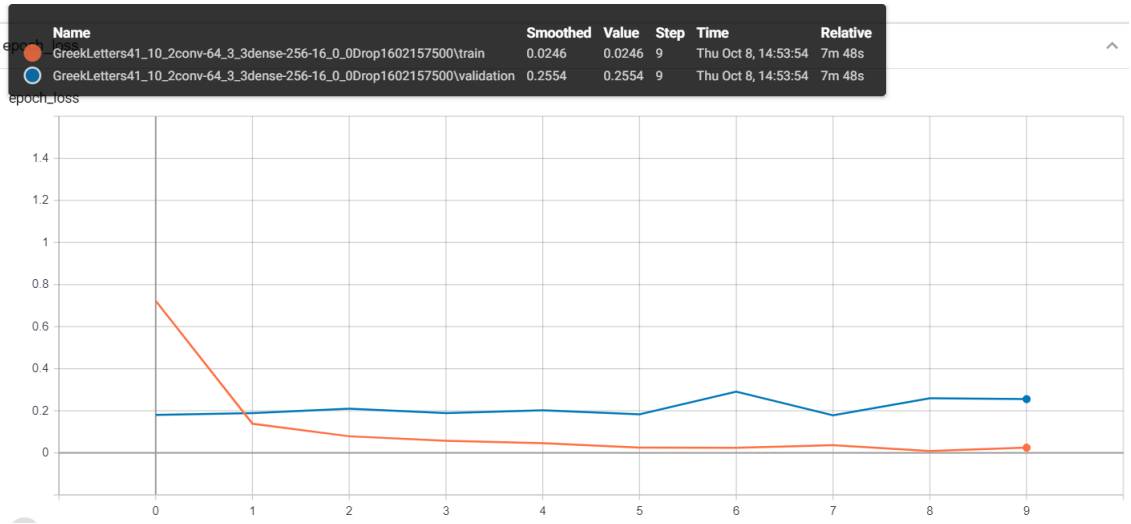
5.1 Γραφικές παραστάσεις και αποτελέσματα εκπαίδευσης

Οι γραφικές παραστάσεις δείχνουν την ακρίβεια και την απώλεια κατά την εκπαίδευση του συστήματος, στη διάρκεια των Epochs, αφορούν το καλύτερο μοντέλο από κάθε αρχιτεκτονική και προκύπτουν χρησιμοποιώντας το TensorBoard, ένα εργαλείο οπτικοποίησης, που παρέχεται με το TensorFlow.

Αρχικά θα παρουσιάσουμε τις γραφικές παραστάσεις της ακρίβειας και της απώλειας στη διάρκεια των Epochs της πρώτης αρχιτεκτονικής στο Σχήμα 5.1 και 5.2 αντίστοιχα.



Σχήμα 5.1: Γραφική παράσταση ακρίβειας της πρώτης αρχιτεκτονικής



Σχήμα 5.2: Γραφική παράσταση απώλειας της πρώτης αρχιτεκτονικής

Οι παραπάνω γραφικές παραστάσεις μας δίνουν και τις τελικές τιμές, που πέτυχαν το train και το validation στο τελαιταίο epoch της ακρίβειας και της απώλειας, οι οποίες είναι 99.3%, 0.9527 και 2.46%, 25.54% αντίστοιχα.

Οι διαφορετικοί παράμετροι και η ακρίβεια, που πετυχαίνει το κάθε μοντέλο της πρώτης αρχιτεκτονικής στο νέο τεστ σετ εικονίζονται στο Σχήμα 5.3. Παρατηρούμε ότι διπλασιάζοντας τον αριθμό των νευρώνων στα Convolutional layer του δεύτερου μοντέλου πετυχαίνουμε μεγαλύτερη ακρίβεια κατά 7% στο πρώτο μοντέλο και η ακρίβεια ελαττώθηκε κατά 20% όταν μεταβήκαμε από το τεστ σετ του dataset στο νέο τέστ με τις εικόνες απ΄ τον πραγματικό κόσμο.

Model	Accuracy
kernel_size= (3,3) batch_size= 16 epochs= 10 conv_neurons= 64 dense_neurons= 256	79%
kernel_size= (3,3) batch_size= 16 epochs= 10 conv_neurons= 32 dense_neurons= 256	72%
kernel_size= (3,3) batch_size= 64 epochs= 10 conv_neurons= 32 dense_neurons= 256	71%

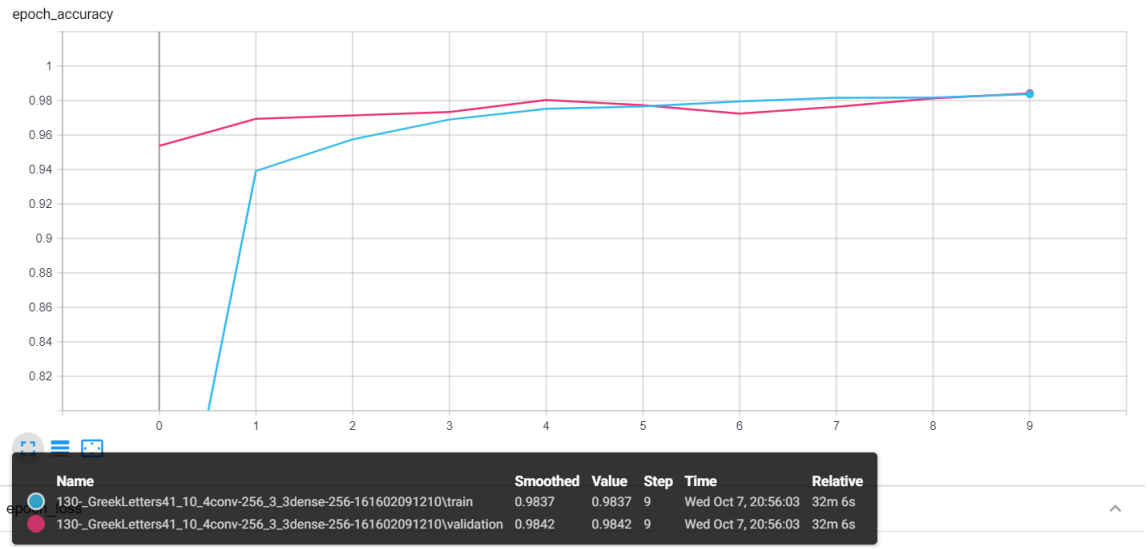
Σχήμα 5.3: Οι διαφορετικοί παράμετροι και η ακρίβεια των μοντέλων της πρώτης αρχιτεκτονικής

Στη συνέχεια, διπλασιάζοντας τον αριθμό των Epochs από 10 σε 20 και προσθέτοντας κάποια Dropout layer αυξήσαμε κι άλλο την ακρίβεια της παραπάνω αρχιτεκτονικής, καταλήγοντας σε ένα ποσοστό ακριβείας 87%, όπως φαίνεται στο Σχήμα 5.4. Με αυτήν την αύξηση στην ακρίβεια, που πετύχε το μοντέλο μετα τον διπλασιασμό των Epochs και την προσθήκη Dropout layer, κατανοήσαμε πόσο σημαντικά είναι τα Dropout layers, αλλά και ο αριθμός των Epochs.

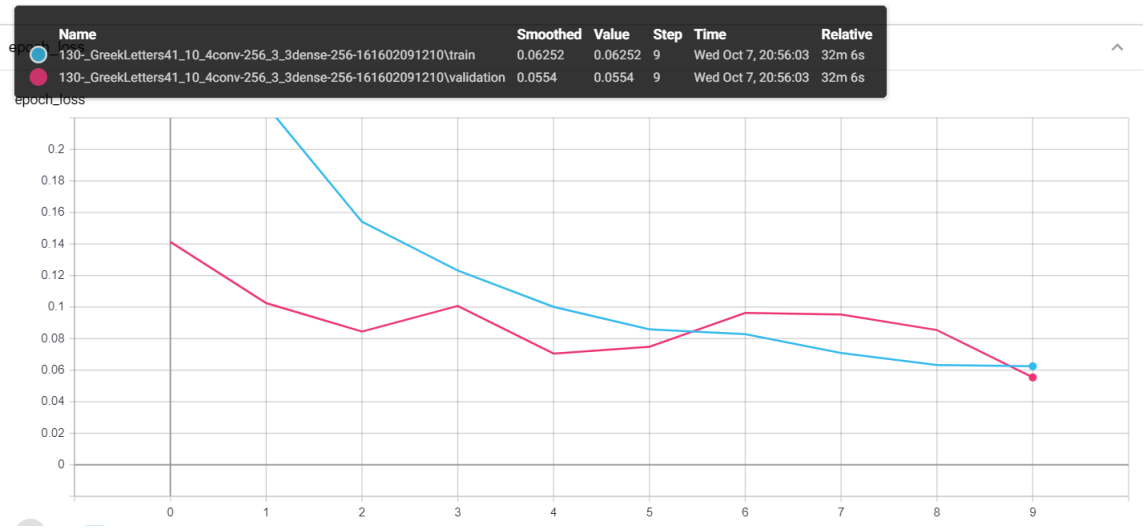
Model	Accuracy
kernel_size= (3,3) batch_size= 16 epochs= 20 conv_neurons= 64 dense_neurons= 256	87%

Σχήμα 5.4: Οι διαφορετικοί παράμετροι και η ακρίβεια του βελτιωμένου μοντέλου

Οι γραφικές παραστάσεις της ακρίβειας και της απώλειας στη διάρκεια των Epochs της δεύτερης αρχιτεκτονικής εικονίζονται στο Σχήμα 5.5 και 5.6 αντίστοιχα.



Σχήμα 5.5: Γραφική παράσταση ακρίβειας της δεύτερης αρχιτεκτονικής



Σχήμα 5.6: Γραφική παράσταση απώλειας της δεύτερης αρχιτεκτονικής

Οι παραπάνω γραφικές παραστάσεις μας δίνουν και τις τελικές τιμές, που πέτυχαν το train και το validation στο τελαιταίο epoch της ακρίβειας και της απώλειας, οι οποίες είναι 98.37%, 98.42% και 6.252%, 5.54% αντίστοιχα.

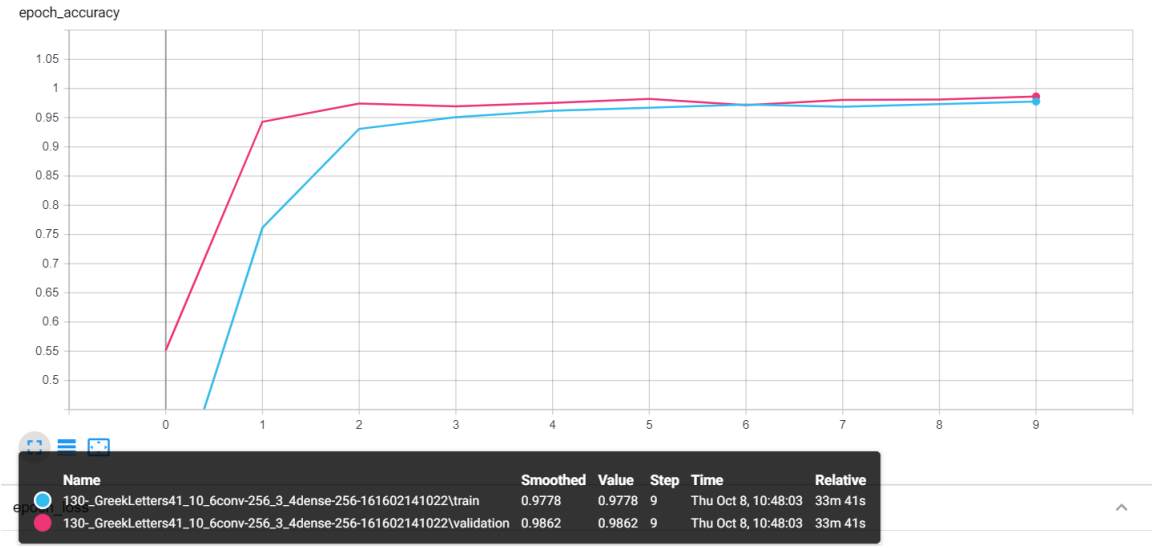
Στη δεύτερη αρχιτεκτονική, οι διαφορετικοί παράμετροι και η ακρίβεια, που πετυχαίνει το κάθε μοντέλο της εικονίζονται στο Σχήμα 5.7. Παρατηρούμε ότι η μέγιστη ακρίβεια, που σημείωσαν και τα τρία μοντέλα ήταν καλύτερη κατά 10% σε σύγκριση με το καλύτερο μοντέλο της πρώτης αρχιτεκτονικής και κατά 2% σε σύγκριση με το βελτιωμένο μοντέλο της. Αυτή η αύξηση, που σημειώθηκε ανάμεσα στο καλύτερο μοντέλο της πρώτης αρχιτεκτονικής και στο πρώτο μοντέλο της δεύτερης αρχιτεκτονικής,

οφείλεται στη μεταβολή του αριθμού των φίλτρων - νευρώνων των Convolutional layer από 64 σε 256 και στην προσθήκη 2 Convolutional, Activation και Dropout layers. Ακόμη, παρατηρούμε ότι η ακρίβεια ελαττώθηκε κατά 10% περίπου, όταν μεταβήκαμε από το τεστ σετ του dataset στο νέο τέστ με τις εικόνες απ΄ τον πραγματικό κόσμο.

Model	Accuracy
kernel_size= (3,3) batch_size= 16 epochs= 10 conv_neurons= 256 dense_neurons= 256	89%
kernel_size= (3,3) batch_size= 16 epochs= 30 conv_neurons= 64 dense_neurons= 256	89%
kernel_size= (3,3) batch_size= 16 epochs= 30 conv_neurons= 256 dense_neurons= 256	89%

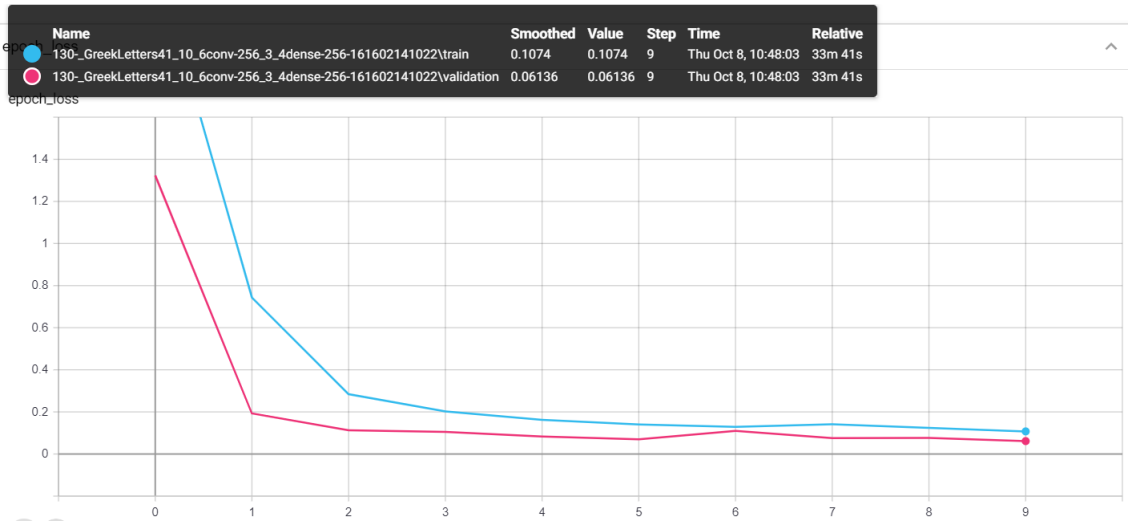
Σχήμα 5.7: Οι διαφορετικοί παράμετροι και η ακρίβεια των μοντέλων της δεύτερης αρχιτεκτονικής

Στην τρίτη αρχιτεκτονική, οι γραφικές παραστάσεις της ακρίβειας και της απώλειας στη διάρκεια των Epochs παριστάνονται στο Σχήμα 5.8 και 5.9 αντίστοιχα.



Σχήμα 5.8: Γραφική παράσταση ακρίβειας της τρίτης αρχιτεκτονικής

Οι γραφικές παραστάσεις της τρίτης αρχιτεκτονικής μας δίνουν και τις τελικές τιμές, που πέτυχαν το train και το validation στο τελαιταίο epoch της ακρίβειας και της απώλειας, οι οποίες είναι 97.78%, 98.62% και 10.74%, 6.136% αντίστοιχα.



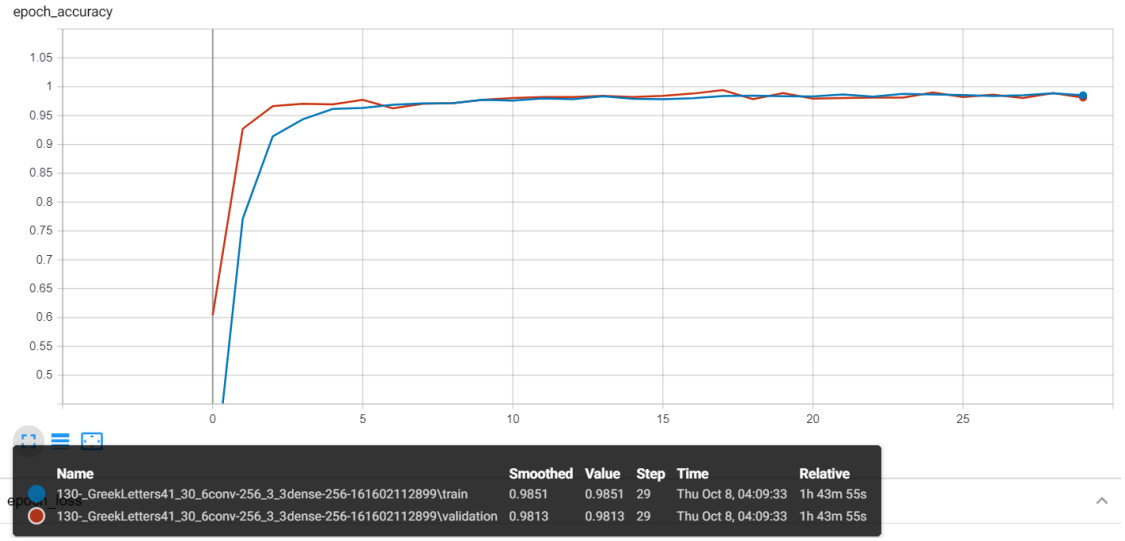
Σχήμα 5.9: Γραφική παράσταση απώλειας της τρίτης αρχιτεκτονικής

Στην τρίτη αρχιτεκτονική, οι διαφορετικοί παράμετροι και η ακρίβεια, που πετυχαίνει το κάθε μοντέλο της εικονίζονται στο Σχήμα 5.10. Συγκρίνοντας με τη δεύτερη αρχιτεκτονική, παρατηρούμε μια αύξηση κατά 4% μεταξύ των μοντέλων τους κι αυτό οφείλεται στην προσθήκη ακόμα 2 Convolutional, Activation και Dropout layers. Επίσης, παρατηρούμε ότι η ακρίβεια ελαττώθηκε κατά 4% όταν μεταβήκαμε από το τεστ σετ του dataset στο νέο τέστ με τις εικόνες απ΄ τον πραγματικό κόσμο.

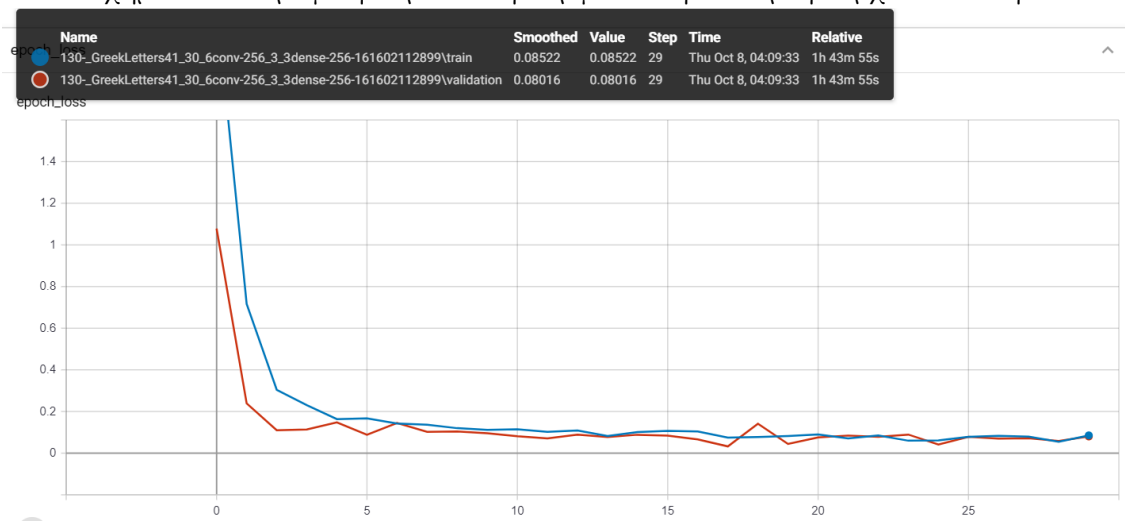
Model	Accuracy
kernel_size= (3,3) batch_size= 16 epochs= 10 conv_neurons= 256 dense_neurons= 256	93%
kernel_size= (3,3) batch_size= 16 epochs= 20 conv_neurons= 64 dense_neurons= 256	91%
kernel_size= (3,3) batch_size= 16 epochs= 30 conv_neurons= 256 dense_neurons= 256	88%

Σχήμα 5.10: Οι διαφορετικοί παράμετροι και η ακρίβεια των μοντέλων της τρίτης αρχιτεκτονικής

Στην τέταρτη αρχιτεκτονική, οι γραφικές παραστάσεις της ακρίβειας και της απώλειας στη διάρκεια των Epochs παριστάνονται στο Σχήμα 5.11 και 5.12 αντίστοιχα. Οι γραφικές παραστάσεις της τέταρτης αρχιτεκτονικής μας δίνουν και τις τελικές τιμές, που πέτυχαν το train και το validation στο τελαιταίο epoch της ακρίβειας και της απώλειας, οι οποίες είναι 98.51%, 98.13% και 8.522%, 8.016% αντίστοιχα.



Σχήμα 5.11: Γραφική παράσταση ακρίβειας της τέταρτης αρχιτεκτονικής



Σχήμα 5.12: Γραφική παράσταση απώλειας της τέταρτης αρχιτεκτονικής

Στην τέταρτη αρχιτεκτονική, οι διαφορετικοί παράμετροι και η ακρίβεια, που πετυχαίνει το κάθε μοντέλο αυτής της αρχιτεκτονικής φαίνονται στο Σχήμα 5.13. Συγκρίνοντας με την τρίτη αρχιτεκτονική παρατηρούμε μια αύξηση κατά 3% ανάμεσα στα καλύτερα μοντέλα τους κι αυτό οφείλεται στον τριπλασιασμό των Epochs από 10 σε 30 και στην αφαίρεση ενός Dense, Activation και Dropout layer. Επιπλέον, παρατηρούμε ότι η ακρίβεια ελαττώθηκε κατά 2% όταν μεταβήκαμε από το τεστ σετ του dataset στο νέο τέστ με τις εικόνες απ΄ τον πραγματικό κόσμο.

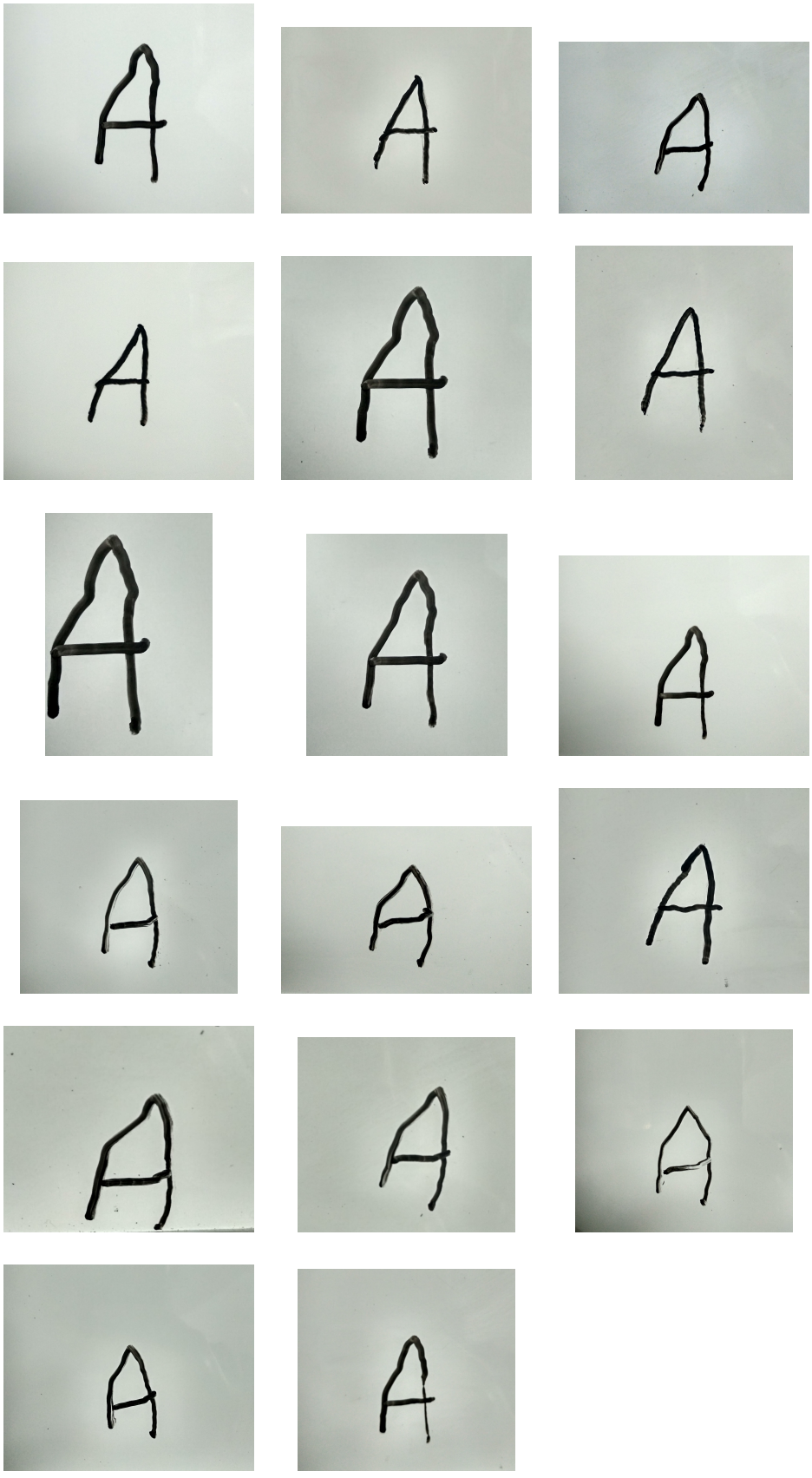
Model	Accuracy
kernel_size= (3,3) batch_size= 16 epochs= 30 conv_neurons= 256 dense_neurons= 256	96%
kernel_size= (3,3) batch_size= 16 epochs= 30 conv_neurons= 64 dense_neurons= 256	94%
kernel_size= (3,3) batch_size= 16 epochs= 10 conv_neurons= 64 dense_neurons= 256	94%
kernel_size= (3,3) batch_size= 64 epochs= 10 conv_neurons= 64 dense_neurons= 256	93%
kernel_size= (3,3) batch_size= 16 epochs= 10 conv_neurons= 64 dense_neurons= 256	93%
kernel_size= (3,3) batch_size= 16 epochs= 100 conv_neurons= 64 dense_neurons= 256	93%
kernel_size= (3,3) batch_size= 16 epochs= 10 conv_neurons= 256 dense_neurons= 256	92%
kernel_size= (3,3) batch_size= 16 epochs= 50 conv_neurons= 64 dense_neurons= 256	92%
kernel_size= (3,3) batch_size= 32 epochs= 30 conv_neurons= 128 dense_neurons= 256	87%

Σχήμα 5.13: Οι διαφορετικοί παράμετροι και η ακρίβεια των μοντέλων της τελευταίας αρχιτεκτονικής

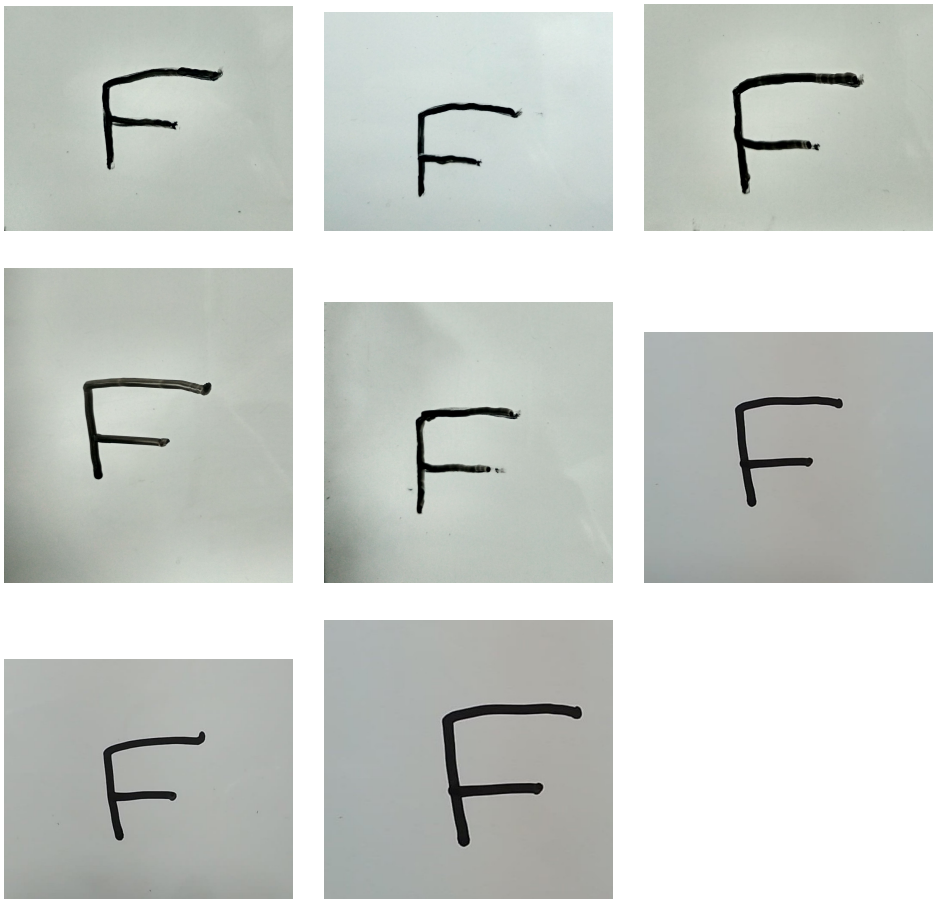
Σύμφωνα με τα παραπάνω αποτελέσματα των διαφορετικών μοντέλων στις τέσσερις αρχιτεκτονικές, συμπεραίνουμε ότι μπορούμε να αυξήσουμε την ακρίβεια των μοντέλων, αυξάνοντας τον αριθμό των Convolutional και Dropout layers, τον αριθμό των Epochs και τον αριθμό των φίλτρων - νευρώνων των Convolutional layer. Δυστυχώς, δεν μπορούμε να αυξάνουμε συνεχώς τα παραπάνω και να αυξάνεται και η ακρίβεια, που πετυχαίνουν τα μοντέλα μας, διότι μετά από ένα σημείο θα έχουμε αντίθετα αποτελέσματα, με την ακρίβεια να μειώνεται.

5.2 Χειρόγραφα κεφαλαία γράμματα από το ρομπότ NAO

Σε αυτή την ενότητα θα παρουσιάσουμε χειρόγραφα κεφαλαία γράμματα, που έχει γράψει το ρομπότ NAO με διαφορετικούς τρόπους και πιο συγκεκριμένα τα γράμματα A (Σχήμα 5.14), F (Σχήμα 5.15), G(Σχήμα 5.16), J(Σχήμα 5.17), Q(Σχήμα 5.18), U(Σχήμα 5.19), W(Σχήμα 5.20), X(Σχήμα 5.21). Παρατηρούμε ότι ορισμένα απ΄ αυτά μοιάζουν αρκετά μεταξύ τους κι άλλα όχι τόσο κι αυτές οι μικρές διαφορές οφείλονται στην αβεβαιότητα στην κίνηση στον πραγματικό κόσμο και πιο συγκεκριμένα στις διαφορετικές θέσεις, που έχουμε τοποθετήσει το ρομπότ για να γράψει το κάθε γράμμα καθώς και σε μικρά σφάλματα και αστοχίες του υλικού. Παρ΄ όλα αυτά τα αποτελέσματα είναι ευανάγνωστα και οι μικρές αποκλίσεις είναι όχι μόνο απειροελάχιστες, αλλά προσδίδουν κι έναν όμορφο χαρακτήρα μη τυποποιημένης και πιο ανθρώπινης συμπεριφοράς.

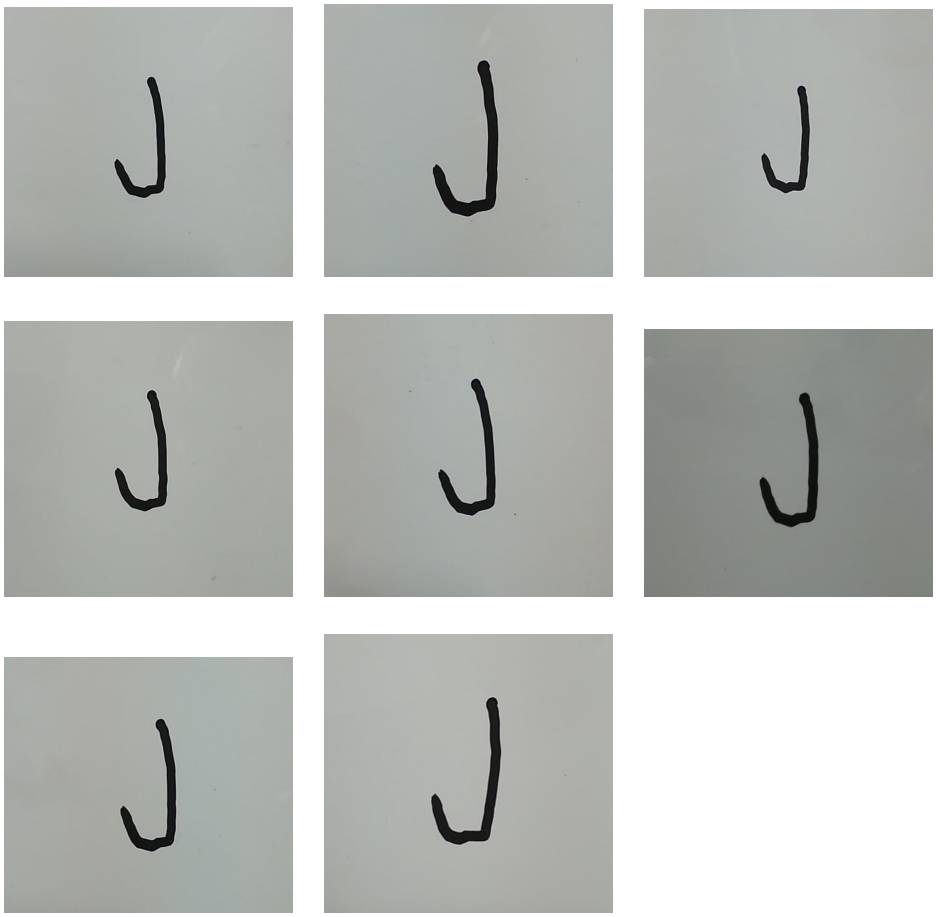


Σχήμα 5.14: Τα Α που έγραψε το NAO

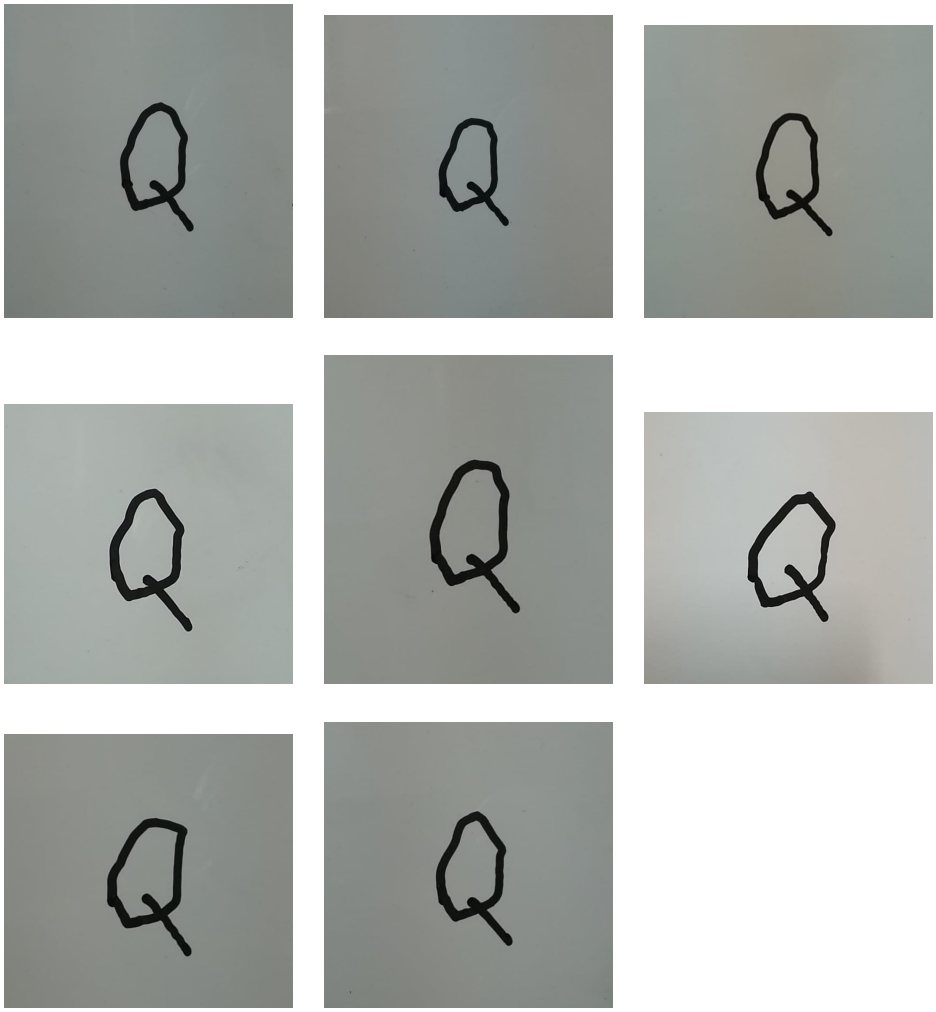


Σχήμα 5.15: Τα F που έγραψε το NAO

Σχήμα 5.16: Τα G που έγραψε το NAO

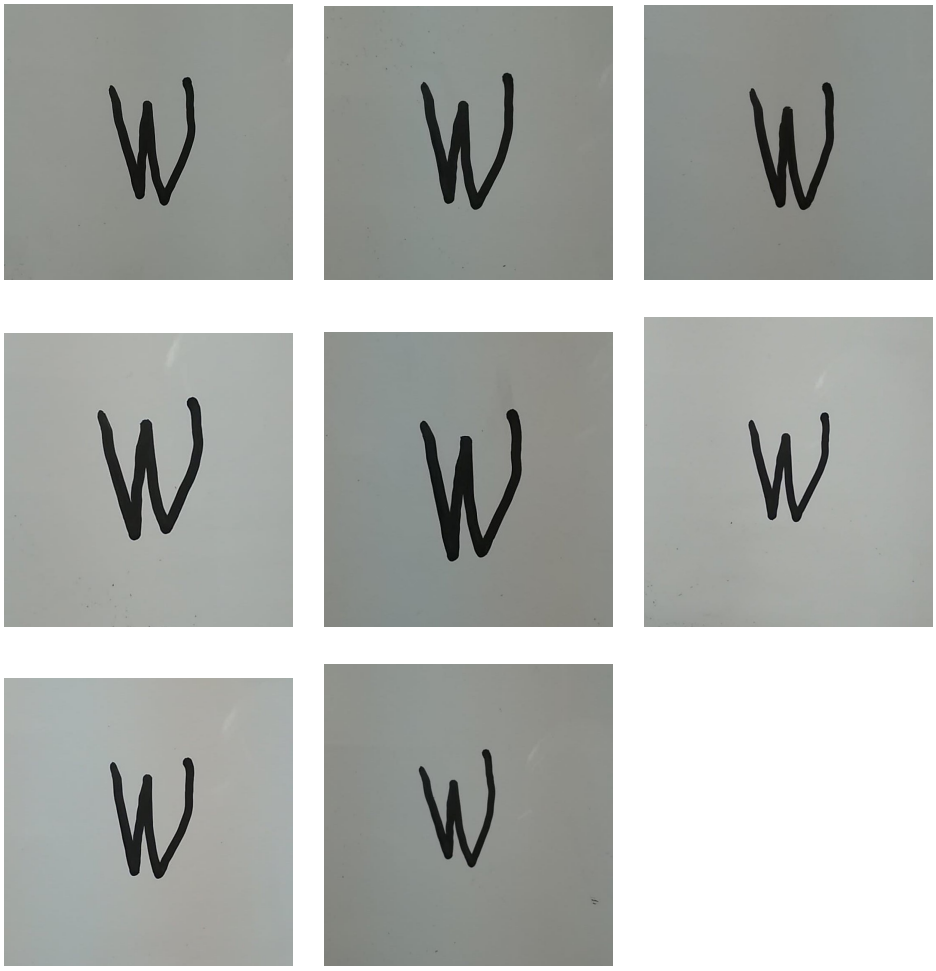


Σχήμα 5.17: Τα J που έγραψε το NAO

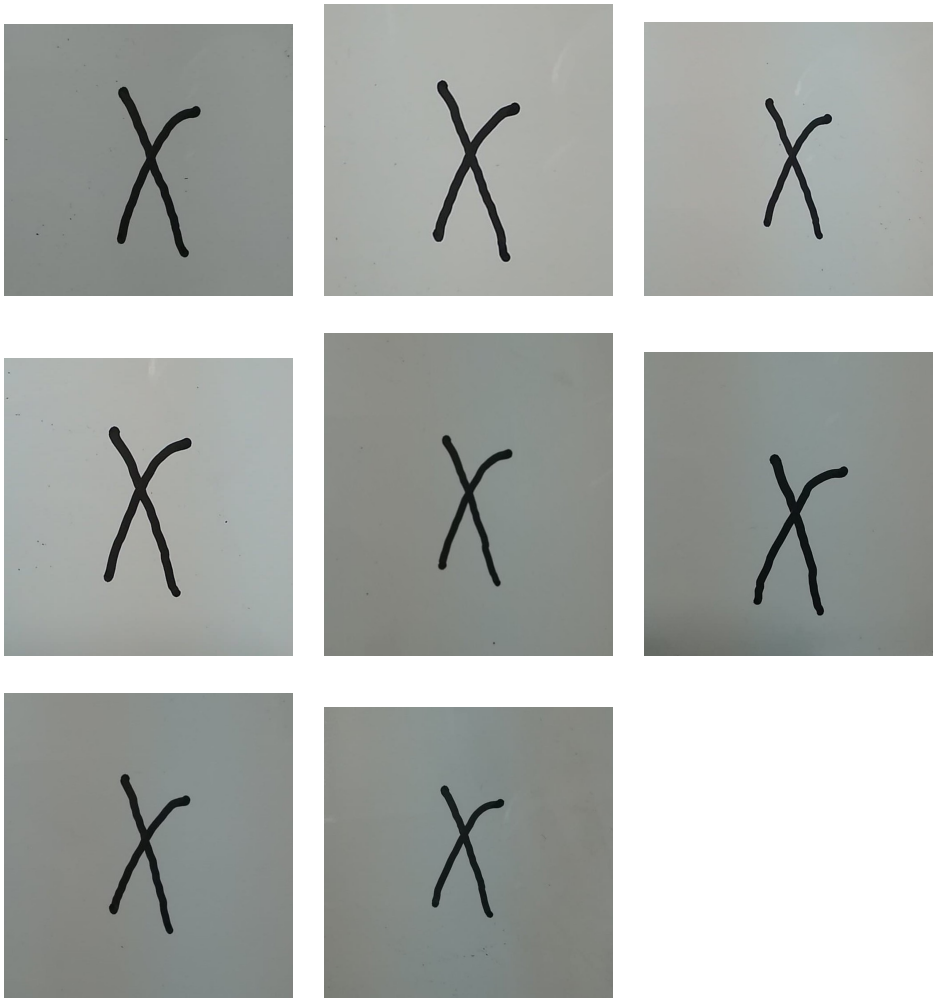


Σχήμα 5.18: Τα Q που έγραψε το NAO

Σχήμα 5.19: Τα U που έγραψε το NAO



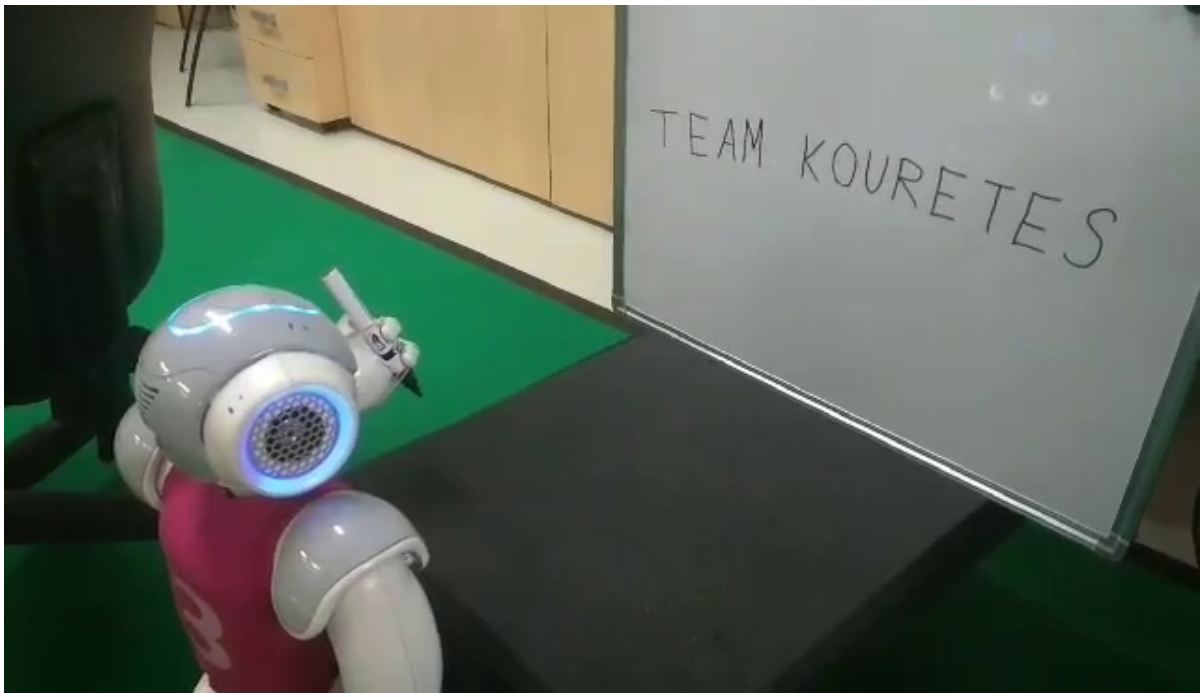
Σχήμα 5.20: Τα W που έγραψε το NAO



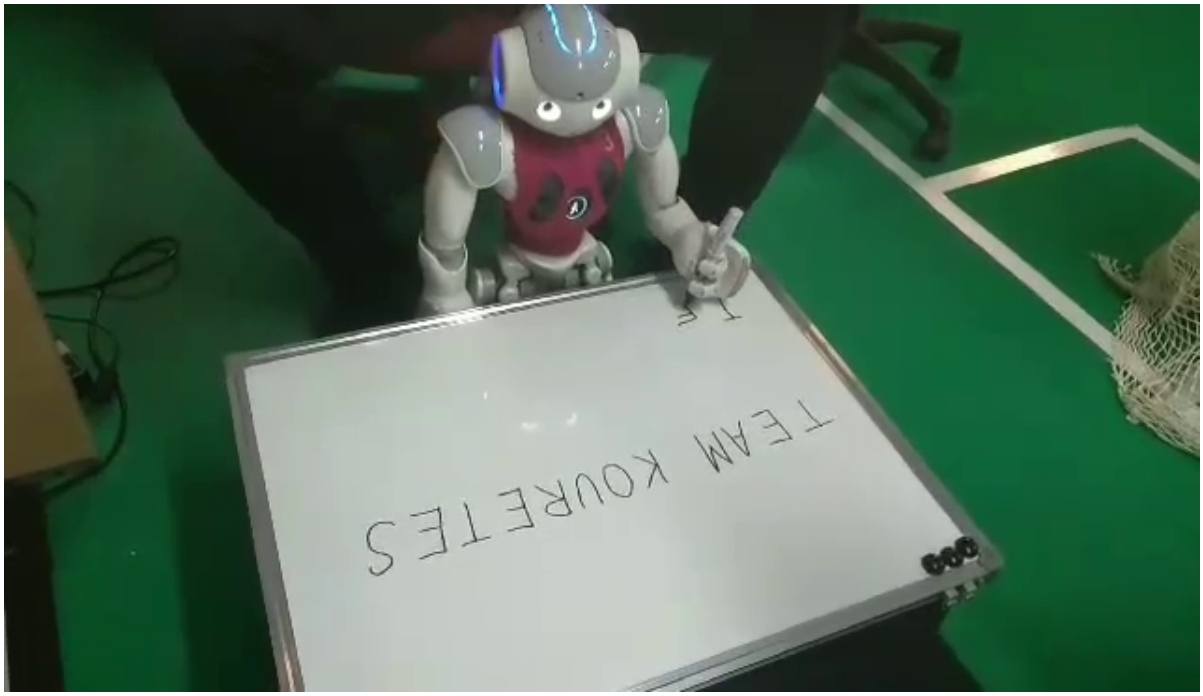
Σχήμα 5.21: Τα Χ που έγραψε το NAO

5.3 Στιγμιότυπα ολοκληρωμένης συμπεριφοράς

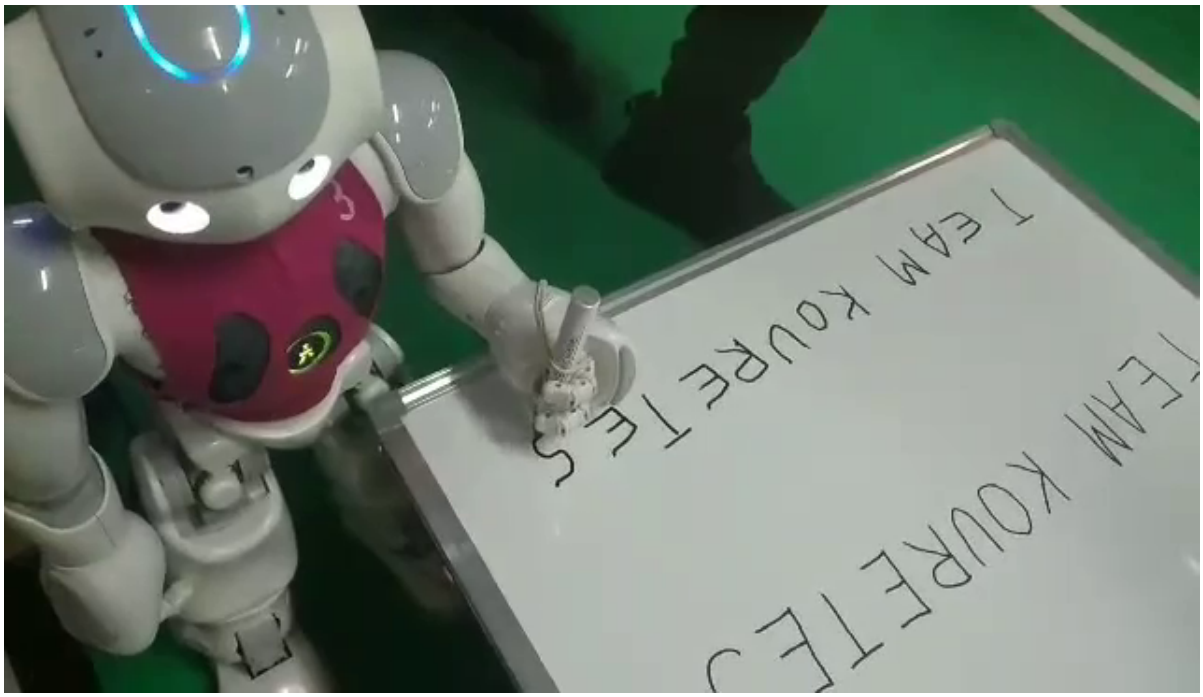
Σε αυτήν την ενότητα παρουσιάζουμε την ολοκληρωμένη συμπεριφορά που υλοποιήσαμε στο ρομπότ NAO, με κάποια στιγμιότυπα από την οπτική αναγνώριση (Σχήμα 5.22) και τη γραφή του χειρόγραφου κειμένου (Σχήματα 5.23 και 5.24). Πιο συγκεκριμένα το ρομπότ NAO αναγνωρίζει με επιτυχία τις δύο λέξεις που έχουμε γράψει στον πίνακα, στη συνέχεια τις γράφει στον ίδιο πίνακα και έχουμε ένα ευανάγνωστο αποτέλεσμα, όπως φαίνεται στο Σχήμα 5.25.



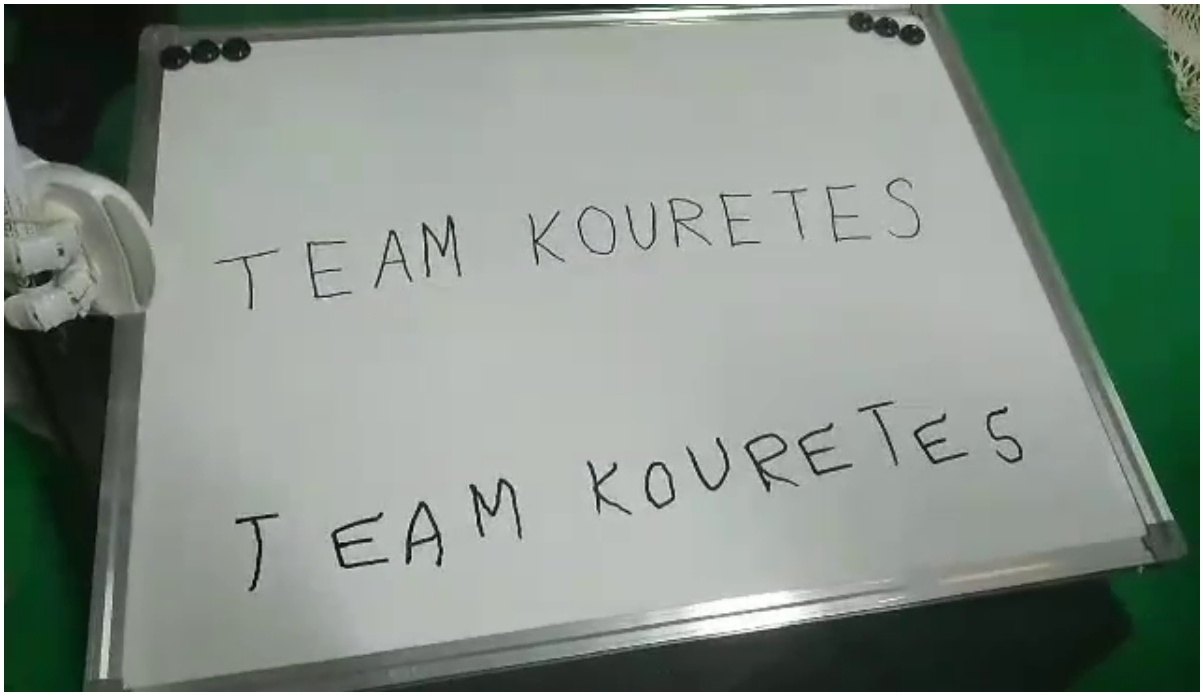
Σχήμα 5.22: Οπτική αναγνώριση χειρόγραφου κειμένου



Σχήμα 5.23: Αρχή της γραφής των δύο λέξεων



Σχήμα 5.24: Ολοκλήρωση της γραφής των δύο λέξεων



Σχήμα 5.25: Τελικό αποτέλεσμα της γραφής του χειρόγραφου κειμένου

Κεφάλαιο 6

Συμπεράσματα

6.1 Συζήτηση

Η εκπόνηση αυτής της διπλωματικής εργασίας ολοκληρώθηκε με επιτυχία σε σχέση με τον αρχικό στόχο, που ήταν η υλοποίηση μιας συμπεριφοράς στο ρομπότ NAO και πιο συγκεκριμένα η οπτική αναγνώριση και γραφή μιας λέξης, που έχει γράψει ο χρήστης. Προκειμένου να αναγνωρίσει το ρομπότ τη λέξη, που του έχει δοθεί, χρησιμοποιήσαμε αλγορίθμους επεξεργασίας εικόνας και συνελικτικά νευρωνικά δίκτυα. Τα αποτελέσματα από την οπτική αναγνώριση είχαν πολύ υψηλό ποσοστό ακρίβειας και για να μην επηρεαστεί αυτό, ο χρήστης πρέπει να γράφει τα γράμματα της κάθε λέξης με ένα κενό ανάμεσά τους, ώστε να είναι πιο ευδιάκριτα και να γίνει πιο σωστά η ανίχνευση και η αναγνώρισή τους. Επιπλέον, για να μην αντιμετωπίσουμε προβλήματα μεγάλης πολυπλοκότητας χρησιμοποιήθηκαν μόνο κεφαλαία χειρόγραφα γράμματα της αγγλικής αλφαβήτου για την εκπαίδευση του συστήματος. Από αυτή τη διπλωματική εργασία συμπεραίνουμε ότι στα νευρωνικά δίκτυα τίποτα δεν είναι τετριμμένο και όταν εργαζόμαστε σε ένα ρομπότ είναι δύσκολη η σχεδίαση κινήσεων του, λόγω ορισμένων παραγόντων, όπως είναι η βαρύτητα και οι περιορισμοί, που έχει στην ελευθερία των κινήσεων του. Τέλος, η συμπεριφορά που δημιουργήσαμε, η οποία προσομοιάζει σε προσφιλείς διαδικασίες μάθησης των παιδιών, μπορεί να αποτελέσει σημείο ενδιαφέροντος ως διαδραστική τεχνολογική επίδειξη σε εκδηλώσεις STEM (Science, Technology, Engineering, Mathematics) για παιδιά, αφού μπορούν να γράφουν όποια λέξη ή φράση θέλουν σε έναν πίνακα, στη συνέχεια τον δείχνουν στο ρομπότ και το ρομπότ τη γράφει σε έναν άλλο πίνακα.

6.2 Μελλοντική Δουλειά

Η εργασία αυτή έχει περιθώρια βελτίωσης τόσο στο κομμάτι της εκπαίδευσης, που αφορά την οπτική αναγνώριση, όσο και στο κομμάτι της γραφής και της συμπεριφοράς του ρομπότ NAO.

6.2.1 Βελτιώσεις οπτικής αναγνώρισης

Αρχικά, μια βελτίωση, που θα μπορούσε να πραγματοποιηθεί, αφορά το κομμάτι της οπτικής αναγνώρισης των χειρόγραφων γραμμάτων, όπου η εκπαίδευση θα γίνει με ένα

διαφορετικό dataset, που θα αποτελείται από εικόνες με κεφαλαία και πεζά γράμματα της ελληνικής και της αγγλικής αλφαβήτου. Στη συνέχεια, για να επεκτείνουμε περαιτέρω την παραπάνω βελτίωση, θα μπορούσαμε στο dataset να προσθέσουμε εικόνες με αριθμούς και αριθμητικούς τελεστές, ώστε το ρομπότ NAO να είναι ικανό, αφού αναγνωρίσει μια μαθηματική πράξη και έχει λάβει την κατάλληλη εκπαίδευση να γράψει το αποτέλεσμά της. Ακόμη, μια διαφορετική προσέγγιση στο θέμα της εκπαίδευσης θα μπορούσε να χρησιμοποιεί ένα dataset με χειρόγραφες λέξεις και Convolutional, Recurrent layers για την αναγνώριση ολόκληρων λέξεων και προτάσεων. Με αυτήν την προσέγγιση παραλείπεται το κομμάτι της ανίχνευσης των γραμμάτων της λέξης και ο χρήστης μπορεί να γράφει αρκετές λέξεις. Η κάθε λέξη μπορεί να γραφτεί από το χρήστη χωρίς κενό ανάμεσα στα γράμματά της, διότι αυτή η προσέγγιση αναγνωρίζει απευθείας τη λέξη χωρίς να χρειάζεται τμηματοποίηση στα γράμματά της.

6.2.2 Βελτιώσεις γραφής και συμπεριφοράς του NAO

Προκειμένου να πραγματοποιηθούν ορισμένες από τις παραπάνω βελτιώσεις θα πρέπει το ρομπότ NAO να εκπαιδευτεί κατάλληλα, ώστε να είναι ικανό να γράφει αριθμούς, κεφαλαία και πεζά γράμματα της ελληνικής και της αγγλικής αλφαβήτου, καθώς και να εκτελεί παραπάνω βήματα προς τα δεξιά για να γράφει τη νέα λέξη της πρότασης, που θα του έχει δοθεί. Επιπλέον, μια δύσκολη και εντυπωσιακή βελτίωση είναι το ρομπότ να έχει εκπαιδευτεί, ώστε να ανιχνεύει, που βρίσκεται ο πίνακας να πηγαίνει προς αυτόν, να προσαρμόζει τη θέση του και να μπορεί να γράφει σε διαφορετικές γωνίες, που θα βρίσκεται ο πίνακας. Τέλος, μια σημαντική βελτίωση θα ήταν η αναβάθμιση του εξοπλισμού με ένα καινούριο ανθρωποειδές ρομπότ για να έχουμε καλύτερα αποτελέσματα σε όλους τους τομείς.

6.3 Μαθήματα

Η διπλωματική εργασία, μας βοήθησε να εξοικειωθούμε με τα νευρωνικά δίκτυα, με αλγορίθμους επεξεργασίας εικόνας και με το ρομπότ NAO. Με αφορμή την εργασία αυτή, είναι κοινή διαπίστωση όλων μας ότι κάποια στιγμή στο άμεσο μέλλον τα ρομπότ θα αντικαταστήσουν τους ανθρώπους, που εργάζονται στη γραμματειακή υποστήριξη.

Βιβλιογραφία

[1] SoftBank Robotics.
https://www.softbankrobotics.com/

[2] RoboCup Standard Platform League.
https://spl.robocup.org/

[3] RoboCup Standard Platform League image.
https://www.robotlab.com/blog/soccer-and-robots-an-educative-winning-combination

[4] NAO motors.
http://doc.aldebaran.com/2-8/family/nao_technical/motors_naov6.html

[5] *Nektarios Sfyris, "Hybrid Visual Simultaneous Localization and Mapping (SLAM) on the Nao Robot using ROS", 2019. NAO sensors image.*

[6] NAOqi software.
http://doc.aldebaran.com/2-5/index_dev_guide.html

[7] Cross language image.
http://doc.aldebaran.com/1-14/dev/naoqi/index.html

[8] Choregraphe, a multi-platform desktop application.
http://doc.aldebaran.com/2-4/software/choregraphe/choregraphe_overview.html

[9] M. Naveenkumar and A. Vadivel, "OpenCV for Computer Vision Applications". 2015.
https://www.researchgate.net/publication/301590571_OpenCV_for_Computer_Vision_Applications

[10] Function cvtColor.
https://docs.opencv.org/master/d8/d01/group__imgproc__color_
_conversions.html#ga397ae87e1288a81d2363b61574eb8cab

[11] Function Canny Edge Detection.
https://docs.opencv.org/master/da/d22/tutorial_py_canny.html

[12] Function pyrMeanShiftFiltering.
https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#
ga9fabdce9543bd602445f5db3827e4cc0

[13] Function findContours.
https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#
ga17ed9f5d79ae97bd4c7cf18403e1689a

[14] RetrievalModes.
https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#
ga819779b9857cc2f8601e6526a3a5bc71

[15] ContourApproximationModes.
https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#
ga4303f45752694956374734a03c54d5ff

[16] Function threshold.
https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.
html

[17] Tensorflow Library.
https://www.infoworld.com/article/3278008/what-is-tensorflow-the-
machine-learning-library-explained.html

[18] Keras.
https://www.infoworld.com/article/3336192/what-is-keras-the-deep-
neural-network-api-explained.html

[19] Machine Learning.
https://expertsystem.com/machine-learning-definition/

[20] *Konstantinos Koutroumbas and Sergios Theodoridis, "Pattern Recognition". Broken Hill Publishers LTD, 4 ed., 2012. Supervised and unsupervised learning 9-11.*

[21] *Stuart Russel and Peter Norvig, "Artificial Intelligence: A Modern Approach". Prentice Hall, 2 ed., 2003. Reinforcement learning 852-875.*

[22] *Stuart Russel and Peter Norvig, "Artificial Intelligence: A Modern Approach". Prentice Hall, 2 ed., 2003. Neural Networks 824-836.*

[23] Artificial neuron.
https://thenextweb.com/neural-basics/2020/05/27/everything-you-need-to-know-about-artificial-neural-networks/

[24] Neural network image.
https://en.wikipedia.org/wiki/Neural_network

[25] Deep Learning.
https://en.wikipedia.org/wiki/Deep_learning

[26] Deep Learning image.
https://towardsdatascience.com/training-deep-neural-networks-9fdb1964b964

[27] Deep Learning vs Most Learning Algorithms image.
https://www.quora.com/Is-there-a-rigorous-justification-of-why-deep-learning-algorithms-need-a-lot-of-data

[28] Deep Learning Face Recognition image.
https://www.researchgate.net/figure/Layers-and-their-abstraction-in-deep-learning-Image-recognition-as-measured-by-ImageNet_fig17_326531654

[29] Convolution Neural Networks.
https://missinglink.ai/guides/convolutional-neural-networks/fully-connected-layers-convolutional-neural-networks-complete-guide/

[30] MaxPooling layer.
https://deepai.org/machine-learning-glossary-and-terms/max-pooling

[31] MaxPooling layer image.
https://www.researchgate.net/figure/The-sub-sampled-output-of-a-max-pooling-operation-with-a-stride-of-2-applied-on-an-input_fig3_329789435

[32] *Konstantinos Koutroumbas and Sergios Theodoridis, "Pattern Recognition". Broken Hill Publishers LTD, 4 ed., 2012. Backpropagation algorithm 174-183.*

[33] Activation Functions in Neural Networks - ReLu.
https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

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

[35] Activation Function Softmax.
https://en.wikipedia.org/wiki/Softmax_function

[36] Activation Function Softmax plot.
https://www.machinecurve.com/index.php/2020/01/08/how-does-the-softmax-activation-function-work/

[37] Singularity University.
https://singularityhub.com/2013/05/10/nao-robot-has-learned-to-write/

[38] University of Melbourne.
https://github.com/Hongyil1/NAO-Robot-Handwriting

[39] NIST Handprinted Characters Database.
https://www.nist.gov/srd/nist-special-database-19