ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΗΣΟΥ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

"Σχεδιασμός και ανάπτυξη ολοκληρωμένης εφαρμογής για ηλεκτρονική μετάδοση παρτίδων σκάκι με χρήση οπτικής αναγνώρισης"

Σπουδαστής:

Δούρος Γεώργιος

Εισηγητής:

Χριστοδούλου Σωτήριος

Περιεχόμενα

Περίληψη

Η παρούσα διπλωματική έχει ως στόχο την υλοποίηση μιας εφαρμογής η οποία αναγνωρίζει οπτικά μια σκακιέρα και τις κινήσεις των πιονιών, με μεγάλη ακρίβεια, χρησιμοποιώντας τεχνολογίες μηχανικής μάθησης και επεξεργασίας εικόνας, αναπαριστώντας την παρτίδα σε μορφή FEN. Η κατάσταση της παρτίδας μεταδίδεται (σε μορφή FEN) σε web εφαρμογή σε πραγματικό χρόνο και απεικονίζεται γραφικά.

Λέξεις-Κλειδιά: OpenCV, TensorFlow, Keras, Μηχανική Μάθηση, Επεξεργασίας Εικόνας

Abstract

This thesis aims to implement an application that visually recognizes a chess board and the movements of the pawns with high accuracy, using machine learning and image processing technologies, representing the game in FEN format. The game's status is transmitted (in FEN format) to a web application in real time and graphically displayed.

Keywords: OpenCV, TensorFlow, Keras, Machine Learning, Image Processing

Εισαγωγή

Η διαδικασία της μετάδοσης είναι η εξής: Η εφαρμογή μέσω μίας κάμερας εντοπίζει την σκακιέρα. Στη συνέχεια μπορεί και αναγνωρίζει τις κινήσεις των πιονιών που γίνονται πάνω σε αυτή. Έπειτα, η κίνηση περνάει από έναν έλεγχο εγκυρότητας για το εάν επιτρέπεται από τους κανόνες ή όχι και τέλος, η κατάσταση της παρτίδας «αναμεταδίδεται» σε ένα εξωτερικό σύστημα.

Η διαδικασία αυτή απαιτεί λοιπόν τα ακόλουθα: Αρχικά έναν τρόπο αναγνώρισης της σκακιέρας. Επιπλέον, έναν τρόπο με τον οποίο θα μπορούν να αναγνωριστούν οι κινήσεις που γίνονται πάνω στη σκακιέρα καθώς επίσης και έναν τρόπο με τον οποίο οι κινήσεις αυτές θα εγκυροποιούνται. Τέλος, απαιτείται ένας τρόπος μετάδοσης της τρέχουσας κατάστασης στο εξωτερικό σύστημα.

Τεχνολογίες και Εργαλεία Υλοποίησης

OpenCV

Για την ανάπτυξη της εφαρμογής, χρησιμοποιήθηκε γλώσσα **Python** καθώς είναι γλώσσα υψηλού επιπέδου και υπάρχει πληθώρα εργαλείων για την αντιμετώπιση των πιο πάνω προβλημάτων. Για την οπτική αναγνώριση χρησιμοποιήθηκε η βιβλιοθήκη OpenCV. Η **OpenCV (Open Source Computer Vision Library)** είναι μια βιβλιοθήκη λογισμικού ανοιχτού κώδικα που χρησιμοποιεί μηχανική μάθηση και «ηλεκτρονική όραση» (Computer Vision).

Η **Ηλεκτρονική Όραση (Computer Vision)** είναι ένας διεπιστημονικός τομέας που ασχολείται με τον τρόπο με τον οποίο μπορούν οι υπολογιστές να αποκτήσουν υψηλού επιπέδου κατανόηση από ψηφιακές εικόνες ή βίντεο. Από την άποψη της μηχανικής, επιδιώκει να αυτοματοποιήσει τα καθήκοντα που εκτελεί το ανθρώπινο οπτικό σύστημα.[1] Η Ηλεκτρονική Όραση ασχολείται με την αυτόματη εξαγωγή, ανάλυση και κατανόηση χρήσιμων πληροφοριών από μία εικόνα ή μια ακολουθία εικόνων που περιλαμβάνει την ανάπτυξη μιας θεωρητικής και αλγοριθμικής βάσης για την επίτευξη αυτόματης οπτικής κατανόησης.

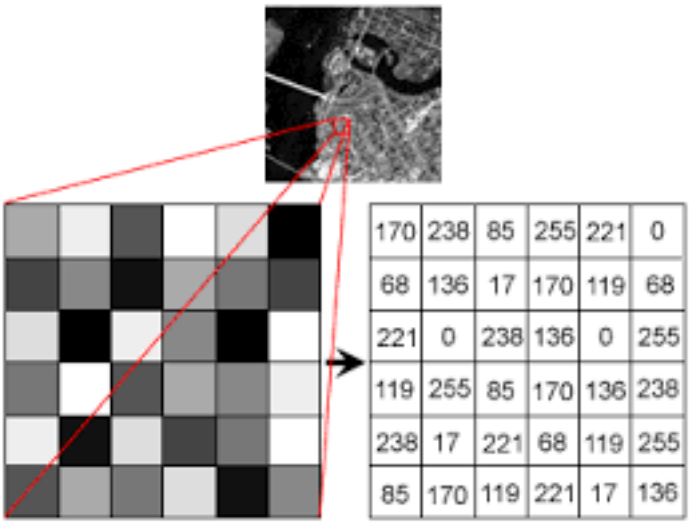
Η OpenCV δημιουργήθηκε λοιπόν για να παρέχει μια κοινή υποδομή για εφαρμογές ηλεκτρονικής όρασης και για να επιταχύνει τη χρήση της «αντίληψης» του μηχανήματος με στόχο τη δημιουργία εμπορικών προϊόντων. Όντας προϊόν με άδεια BSD, το OpenCV διευκολύνει τις επιχειρήσεις να χρησιμοποιούν και να τροποποιούν τον κώδικα.

Η βιβλιοθήκη διαθέτει περισσότερους από 2500 βελτιστοποιημένους αλγόριθμους, οι οποίοι περιλαμβάνουν ένα ολοκληρωμένο σύνολο κλασικών και υπερσύγχρονων αλγορίθμων «όρασης υπολογιστή» 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 κάμερας, την παρακολούθηση κινούμενων αντικειμένων, την εξαγωγή τρισδιάστατων μοντέλων αντικειμένων, την παραγωγή τρισδιάστατων σημείων από στερεοφωνικές κάμερες, τη συρραφή εικόνων για την παραγωγή υψηλής ανάλυσης εικόνας, εύρεση παρόμοιων εικόνων από μια βάση δεδομένων εικόνων, αφαίρεση «κόκκινων ματιών» από εικόνες που λαμβάνονται χρησιμοποιώντας φλας κ.λπ. Ενδεικτικά κάποιες από τις λειτουργίες της βιβλιοθήκης, τις οποίες χρησιμοποιεί και η εφαρμογή:

Ανάγνωση Εικόνων

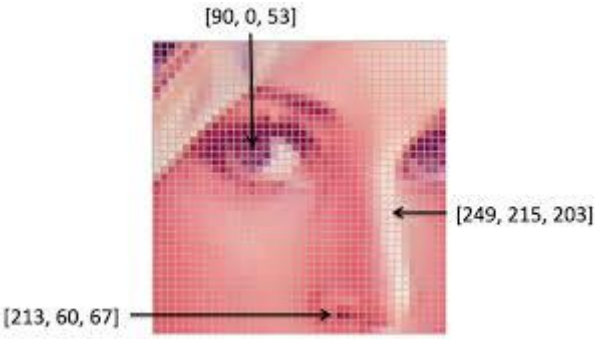
Οι υπολογιστές μπορούν να κατανοήσουν μόνο αριθμούς, αυτό σημαίνει ότι οι εικόνες πρέπει να μετατρέπονται σε ψηφία έτσι ώστε να μπορούν να επεξεργασθούν από την εφαρμογή.

[1] Dana H. Ballard, Christopher M. Brown (1982). “Computer Vision”



Κάθε αριθμός αναπαριστά την απόχρωση κάθε pixel της εικόνα. Στο συγκεκριμένο παράδειγμα, η εικόνα είναι ασπρόμαυρη, επομένως κάθε κελί περιέχει την «ένταση» (intensity) του γκρι, ποιο ανοιχτές αποχρώσεις υποδηλώνονται με μικρότερους αριθμούς και ποιο σκούρες με μεγαλύτερους. Οι μέγιστες τιμές της έντασης κάθε pixel είναι 255 (απολυτό λευκό) και 0 (απόλυτο μαύρο).

Οι έγχρωμες εικόνες αποθηκεύουν μικρά σετ τιμών σε κάθε κελί όπου το κάθε ένα προσδιορίζει τη ποσότητα των χρωμάτων Κόκκινου, Πράσινου και Μπλε (RGB).



Η OpenCV απλοποιεί τις διαδικασίες της μετατροπής εικόνων σε πίνακες αριθμών και επιπλέον κάνει την διαχείριση των εικόνων και των παραγόμενων πινάκων πιο εύκολη.

Αλλαγή Χρώματος

Ωστόσο, η επεξεργασία πινάκων πολλαπλών διαστάσεων οι οποίοι αναπαριστούν έγχρωμες εικόνες μπορεί να αποτελέσουν μία πολύ δύσκολη αλλά και πολύ δαπανηρή σε υπολογιστικούς πόρους διαδικασία. Για τον λόγο αυτό είναι χρήσιμο να όπου επιτρέπεται, να είναι δυνατή η μετατροπή τους σε ασπρόμαυρες ή ακόμα και μονόχρωμες. Έτσι μειώνεται σημαντικά η επεξεργαστική ισχύς που απαιτείται για την διαχείριση τους.

Μετατροπή Μεγέθους

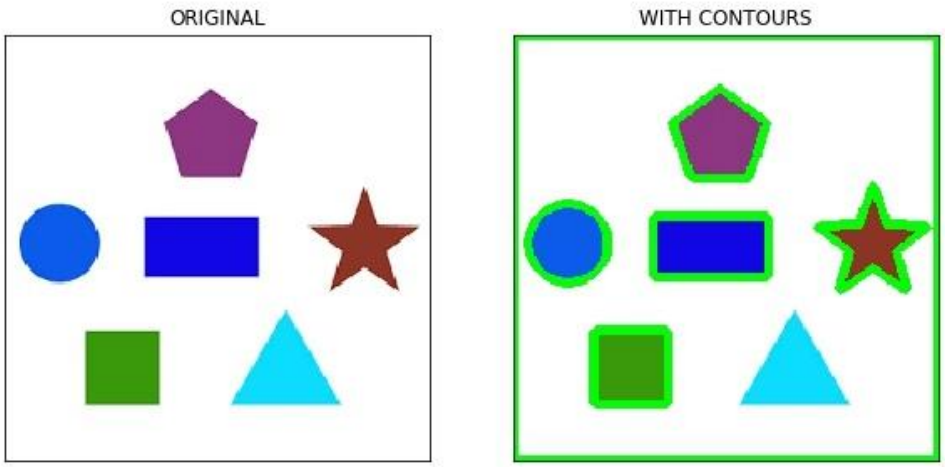
Συνήθως, τα μοντέλα οπτικής αναγνώρισης, όπως και τα μοντέλα μηχανικής μάθησης λειτουργούν δίνοντας τους ένα προκαθορισμένο μέγεθος εικόνας ως είσοδο για την εκπαίδευση τους, επομένως είναι χρήσιμο να μπορεί να πραγματοποιηθεί μεταβολή στο μέγεθος των εικόνων σε πραγματικό χρόνο. Έτσι για παράδειγμα μπορούμε να χρησιμοποιούμε τα καρέ που καταγράφει η κάμερα σε πλήρη ανάλυση και να τα προσαρμόζουμε στις απαιτήσεις εκμάθησης του μοντέλου. Η OpenCV υλοποιεί πολλαπλούς αλγορίθμους μετατροπής μεγέθους εικόνας όπως:

INTER_NEAREST	Nearest neighbor interpolation
INTER_LINEAR	Bilinear interpolation
INTER_AREA	Resampling using pixel area relation
INTER_CUBIC	Bicubic interpolation over 4×4 pixel neighborhood
INTER_LANCZOS4	Lanczos interpolation over 8×8 neighborhood

Περιγράμματα Εικόνας

Το περίγραμμα είναι μια κλειστή καμπύλη σημείων ή τμημάτων γραμμής που αντιπροσωπεύει τα όρια ενός αντικειμένου στην εικόνα. Τα περιγράμματα είναι ουσιαστικά τα σχήματα των αντικειμένων σε μια εικόνα.

Σε αντίθεση με τα άκρα (edges) σε μία εικόνα, τα περιγράμματα δεν αποτελούν μέρος της εικόνας. Αντ' αυτού, είναι μια αφηρημένη **συλλογή σημείων** και τμημάτων γραμμών που αντιστοιχούν στα σχήματα των αντικειμένων που αναπαρίστανται σε αυτήν.



Μπορούμε έτσι, να χρησιμοποιήσουμε περιγράμματα αυτά για να μετρήσουμε τον αριθμό των αντικειμένων σε μια εικόνα, να κατηγοριοποιήσουμε αντικείμενα που απεικονίζονται βάσει των σχημάτων τους ή να επιλέξουμε αντικείμενα συγκεκριμένων σχημάτων από την εικόνα, πράγμα που αποτελεί σημαντικό βήμα για την αρχή της οπτικής αναγνώρισης.

Η OpenCV έχει κοινότητα με περισσότερα από 47.000 χρήστες και ο εκτιμώμενος αριθμός λήψεων της είναι άνω των 18.000.000, ενώ η βιβλιοθήκη χρησιμοποιείται εκτενώς από εταιρείες, ερευνητικές ομάδες και από κυβερνητικούς φορείς.

Η OpenCV είναι εγγενώς γραμμένη σε C++ αλλά διαθέτει διεπαφές (APIs) σε C ++, Python, Java και MATLAB καθώς επίσης υποστηρίζει Windows, Linux, Android και Mac OS.

TensorFlow και Keras

Το επόμενο βήμα είναι να βρούμε έναν τρόπο με τον οποίο θα μπορεί η εφαρμογή να αναγνωρίζει τις κινήσεις των πιονιών πάνω στη σκακιέρα με στόχο να εκτιμάται η κίνηση που κάνει ο εκάστοτε παίκτης με αποτέλεσμα αυτή η κίνηση να περνάει στο επόμενο βήμα της διαδικασίας. Η επίτευξη της φάσης αυτής γίνεται με τη χρήση αλγορίθμων **μηχανικής μάθησης.**

Ο άνθρωπος προσπαθεί να κατανοήσει το περιβάλλον του παρατηρώντας το και δημιουργώντας μια απλοποιημένη (αφαιρετική) εκδοχή του η οποία ονομάζεται μοντέλο (model). Η δημιουργία ενός τέτοιου μοντέλου, ονομάζεται επαγωγική μάθηση (inductive learning). Επιπλέον, ο άνθρωπος έχει τη δυνατότητα να οργανώνει και να συσχετίζει τις εμπειρίες και τις παραστάσεις του δημιουργώντας νέες δομές που ονομάζονται πρότυπα (patterns). Με αυτό το τρόπο, μπορούμε να θέσουμε ως ορισμό της **μηχανικής μάθησης (machine learning)** την δημιουργία τέτοιων μοντέλων και/ή προτύπων από ένα σύνολο δεδομένων, από ένα υπολογιστικό σύστημα.[2]

Θα «εκπαιδεύσουμε» έτσι ένα μοντέλο το οποίο θα χρησιμοποιεί η εφαρμογή έτσι ώστε να μπορεί να κατηγοριοποιεί τα πιόνια πάνω στο ταμπλό και να εντοπίζει τις σκακιστικές κινήσεις που εκτελούν οι παίκτες. Η Google έχει υλοποιήσει μια τέτοια βιβλιοθήκη η οποία ονομάζεται TensorFlow και θα είναι το εργαλείο που θα χρησιμοποιήσουμε για την εκπαίδευση του μοντέλου.

Η **TensorFlow** είναι μια βιβλιοθήκη ανοιχτού κώδικα για αριθμητικούς υπολογισμούς και μηχανική εκμάθηση μεγάλης κλίμακας. Συνδυάζει μια σειρά μοντέλων και αλγορίθμων μηχανικής μάθησης και βαθιάς μάθησης (deep learning) τα οποία καθιστά χρήσιμα μέσω μιας κοινής μεταφοράς. Η TensorFlow χρησιμοποιεί την Python για να παρέχει ένα βολικό API για την ανάπτυξη εφαρμογών, ενώ εκτελεί αυτές τις εφαρμογές σε C ++ υψηλής απόδοσης.

Επιπλέον, η TensorFlow μπορεί να εκπαιδεύσει και να εκτελέσει βαθιά νευρωνικά δίκτυα με σκοπό την αναγνώριση εικόνας, επεξεργασία φυσικής γλώσσας (NLP), προσομοιώσεις βασισμένες σε Διαφορικές Εξισώσεις (partial differential equation, PDE) κ.α.

[2] Βλαχαβάς Ιωάννης, Κεφάλας Πέτρος, Βασιλειάδης Νικόλας, Κόκκορας Φώτης, Σακελλαρίου Ηλίας (2006). "Τεχνητή Νοημοσύνη, Γ' Έκδοση"

Τέλος η TensorFlow μπορεί να προβλέπει το αποτέλεσμα της εκμάθησης του συστήματος χρησιμοποιώντας τα ίδια μοντέλα που παρέχονται στην διαδικασία της εκμάθησης, το οποίο διευκολύνει σε μεγάλο βαθμό τη διαδικασία της εκπαίδευσης.

Η **Keras**, από την άλλη, είναι μια βιβλιοθήκη ανοιχτού κώδικα γραμμένη σε Python, η οποία υλοποιεί λειτουργίες νευρωνικών δικτύων. Το Keras, σχεδιάστηκε από τον μηχανικό της Google, François Chollet, με σκοπό την χρήση της από ευρύ κοινό, το οποίο δεν κατέχει εξειδικευμένες γνώσεις σχεδιασμό νευρωνικών δικτύων. Λόγο αυτού, η βιβλιοθήκη δεν υλοποιεί μαθηματικές συναρτήσεις και λειτουργίες χαμηλού επιπέδου (low-level), αλλά «κουμπώνει» πάνω σε άλλες βιβλιοθήκες που χειρίζονται αυτές τις διαδικασίες, οι οποίες βιβλιοθήκες ονομάζονται **Backend**. Η TensorFlow, αποτελεί μία τέτοια βιβλιοθήκη. Αυτό καταλήγει ουσιαστικά την Keras, να είναι ένας wrapper, υψηλού επιπέδου (high-level) API, για τη χρήση χαμηλού επιπέδου (low-level) API. Διαγραμματικά, οι διαφορές μεταξύ των Keras και TensorFlow:

Keras	TensorFlow
Χρήση υψηλού επιπέδου API	Χρήση χαμηλού επιπέδου API
Χρησιμοποιείται για την υλοποίηση διαδικασιών εκμάθησης και δημιουργίας μοντέλων νευρωνικών δικτύων	Χρησιμοποιείται για την εκτέλεση μαθηματικών συναρτήσεων για τις διαδικασίες υλοποίησης των νευρωνικών δικτύων
Χρησιμοποιείται για ταχεία ανάπτυξη μοντέλων με τυπικά πολλαπλά επίπεδα	Επιτρέπει την δημιουργία υπολογιστικών γράφων και πολύπλοκων μοντέλων μηχανικής μάθησης

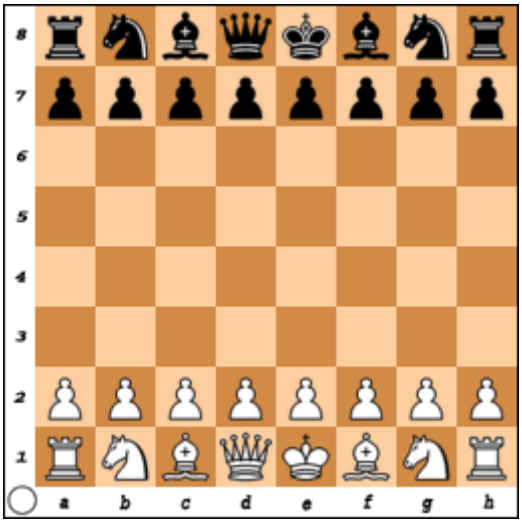
Με τα API της Keras μπορούμε να χειριστούμε τον τρόπο με τον οποίο φτιάχνουμε μοντέλα νευρωνικών δικτύων, να καθορίσουμε τα επίπεδα τους, ή να δημιουργούμε μοντέλα πολλαπλών εισόδων-εξόδων. Επιπλέον, η Keras καταρτίζει το μοντέλο μας με λειτουργίες απώλειας (loss-rate) και βελτιστοποίησης και φυσικά είναι υπεύθυνη για την διαδικασία προπόνησης με λειτουργία του μοντέλου. Εκτός από τα τυπικά νευρωνικά δίκτυα, η Keras υποστηρίζει **συνελικτικά (convolutional)** και **επαναλαμβανόμενα (recurrent)** νευρωνικά δίκτυα καθώς επίσης υποστηρίζει και κοινές λειτουργίες των ενδιάμεσων στρωμάτων των δικτύων, όπως είναι οι pooling, dropout, normalization κ. α.

Μέχρι την έκδοση 2.3 η Keras υποστήριζε πολλαπλά Backend, συμπεριλαμβανομένων των TensorFlow, Microsoft Cognitive Toolkit, Theano και PlaidML. Από την έκδοση 2.4 ωστόσο, υποστηρίζεται μόνο η TensorFlow.

Python Chess

Στη συνέχεια πρέπει να υλοποιηθεί η διαδικασία της εγκυρότητας της κίνησης. Εφόσον η οπτική αναγνώριση και η πρόβλεψη της κίνησης έγινε σε μία δεδομένη στιγμή, είναι απαραίτητο να μπορούμε να ελέγξουμε εάν η κίνηση αυτή είναι επιτρεπτή από τους κανόνες.

Για την απαίτηση αυτή, θα χρησιμοποιηθεί η βιβλιοθήκη Python Chess. Η Python Chess είναι μία ολοκληρωμένη βιβλιοθήκη η οποία υλοποιεί όλες τις απαραίτητες λειτουργίες για τον έλεγχο εγκυρότητας σκακιστικών κινήσεων, καταγραφής της κατάστασης μίας παρτίδας καθώς επίσης και κωδικοποίησης της παρτίδας σε μορφή FEN.



FEN: rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1

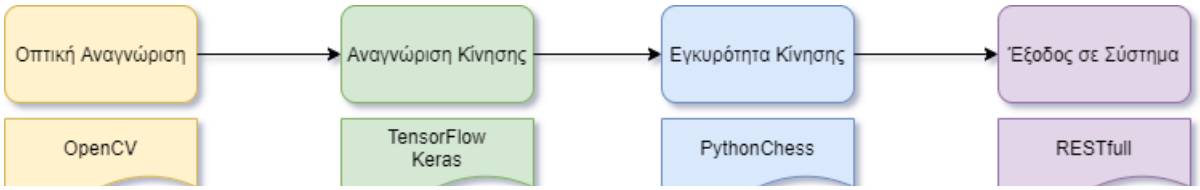
Η βιβλιοθήκη χρησιμοποιείται από πολλές υπηρεσίες και πλατφόρμες για διάφορους σκοπούς όπως ανάλυση παρτίδων, υλοποίηση ηλεκτρονικών αντιπάλων για την εξάσκηση παικτών αλλά και για την δημιουργία βιντεοπαιχνιδιών σκακιού. Χαρακτηριστικό παράδειγμα ενός έργου το οποίο χρησιμοποιεί την βιβλιοθήκη αυτή, αποτελεί το Raspberry Turk. Το Raspberry Turk, είναι ένα ρομπότ εμπνευσμένο από την μηχανή Mechanical Turk του 18^{ου} αιώνα, το οποίο μπορεί και παίζει αυτόνομα παρτίδες σκακιού. Το ρομπότ αποτελείται από μία κάμερα για την αναγνώριση της σκακιέρας, έναν μηχανικό βραχίονα για να εκτελεί τις κινήσεις και ένα raspberry pi για την υπολογιστική επεξεργασία. Η λογική της εγκυρότητας του ρομπότ υλοποιείται με τη χρήση της Python Chess.

Ruby On Rails

Το τελικό βήμα της διαδικασίας είναι η αναμετάδοση της παρτίδας σε ένα εξωτερικό σύστημα. Ένα τέτοιο σύστημα μπορεί να θεωρηθεί είτε ένας server, οποίος καταγράφει τις παρτίδες, είτε μία εξωτερική βάση δεδομένων ή η είσοδος ενός δεύτερου συστήματος για επιπλέον μελέτη των πληροφοριών της παρτίδας. Στα πλαίσια της εργασίας αυτή, ως εξωτερικό σύστημα χρησιμοποιείται ένα web application το οποίο αναμεταδίδει την παρτίδα με τη χρήση ενός γραφικού περιβάλλοντος. Ο τρόπος επικοινωνίας με το σύστημα αυτό γίνεται με τη χρήση REST. Η Python, παρέχει με τη βιβλιοθήκη request όλες τις απαραίτητες συναρτήσεις για την REST-full επικοινωνία μεταξύ των δύο συστημάτων.

Η web εφαρμογή που χρησιμοποιείται ως εξωτερικό σύστημα κατασκευάστηκε με τη χρήση Ruby On Rails. Η Ruby On Rails είναι βιβλιοθήκη ανοιχτού κώδικα η οποία χρησιμοποιείται για την κατασκευή web application με τη χρήση γλώσσας Ruby. Παρέχει απλοϊκό και εύχρηστο API για την δημιουργία και ιεράρχησης των φακέλων και των αρχείων της εφαρμογής. Επιπλέον, είναι σχεδιασμένη πάνω στην δομή του MVC (Model, View, Controller) και ικανοποιεί τον κανόνα σχεδίασης Convention Over Configuration. Αυτό σημαίνει ότι η βιβλιοθήκη αυτοματοποιεί κάποιες από τις διαδικασίες του MVC ακολουθώντας κάποιες «συμβάσεις» (conventions). Έτσι η χρόνος υλοποίησης μειώνεται με στόχο, ο χρόνος αυτός όπου κερδήθηκε να χρησιμοποιηθεί σε σημαντικότερα καθήκοντα.

Ο σχεδιασμός της εφαρμογής έγινε με τρόπο έτσι ώστε όλα τα συστήματα να είναι ανεξάρτητα (de-coupled) μεταξύ τους. Αυτό σημαίνει ότι κάθε τμήμα της διαδικασίας που εκτελεί η εφαρμογή (οπτική αναγνώριση, αναγνώριση κίνησης, εύρεση εγκυρότητας κίνησης και αποστολή σε εξωτερικό σύστημα), είναι αυτόνομο από τα εργαλεία τα οποία χρησιμοποιεί. Με τον τρόπο αυτό, σε μεταγενέστερη ανάπτυξη της εφαρμογής μπορεί να χρησιμοποιηθούν διαφορετικά εργαλεία, χωρίς να χρειάζεται να αλλάξει η «λογική» της διαδικασίας.

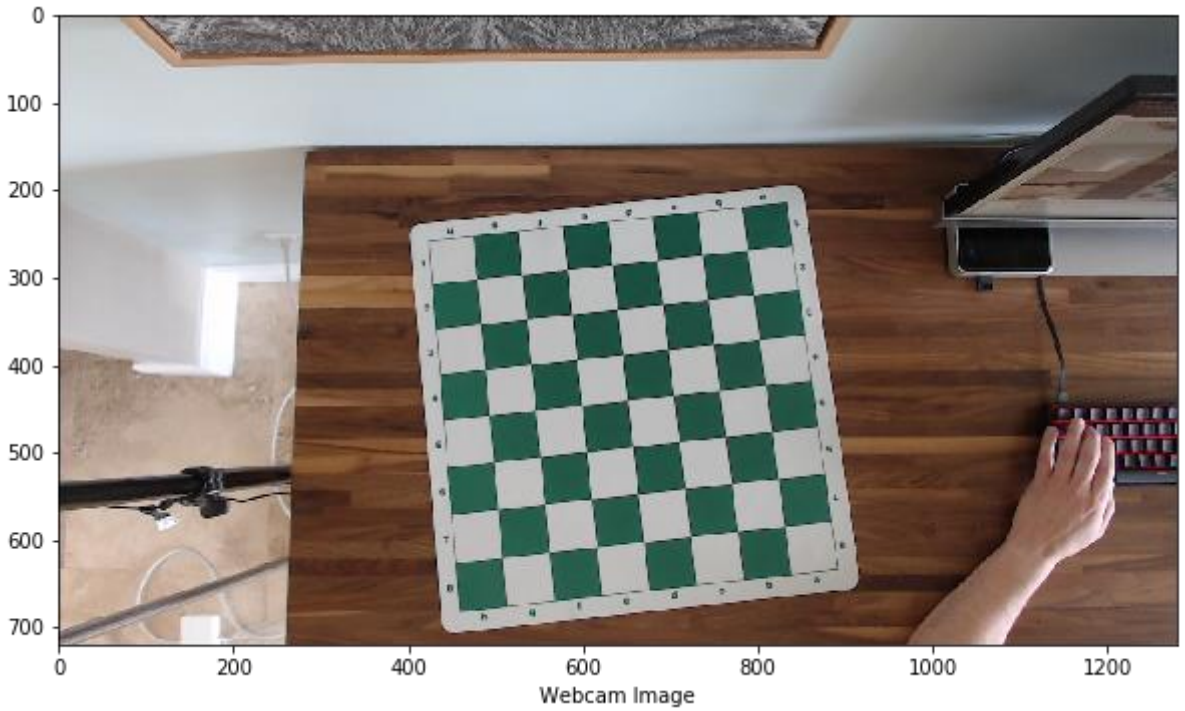


Εικόνα 1: Σχεδιάγραμμα Διαδικασίας

Εκτός από τις βιβλιοθήκες λογισμικού, η εφαρμογή προφανώς, χρησιμοποιεί και μία κάμερα για την οπτική αναγνώριση και την αναγνώριση των κινήσεων. Η κάμερα θα πρέπει για την βέλτιστη λειτουργεία του συστήματος να έχει ανάλυση HD 720p.

Αρχικοποίηση του χώρου εργασίας

Τοποθετούμε την σκακιέρα σε μια επίπεδη επιφάνεια με στημένα τα πιόνια. Έπειτα τοποθετούμε την κάμερα ακριβώς πάνω από τη σκακιέρα σε ορθή γωνία και σε ύψος τέτοιο ώστε τα πιόνια να μην προεξέχουν από το περίγραμμα των κελιών τους (περίπου 1 μέτρο), επιπλέον τοποθετούμε την κάμερα σε τέτοια θέση ώστε εάν κάνουμε λήψη φωτογραφίας τα λευκά πιόνια να απεικονίζονται στο κάτω μέρος της εικόνας. Ιδανικά θα πρέπει η σκακιέρα και η κάμερα να τοποθετηθούν σε σημείο με διάχυτο φωτισμό έτσι ώστε να μην δημιουργούνται σκιές από τα πιόνια πάνω στη σκακιέρα.



Εικόνα 2: Καρέ κάμερας.

Οπτική Αναγνώριση

Η οπτική αναγνώριση είναι το πρώτο βήμα της διαδικασίας της εφαρμογής. Στο βήμα αυτό, η εφαρμογή εντοπίζει τη σκακιέρα, ως εικόνα, με στόχο αργότερα να εκτελεί τους αλγορίθμους αναγνώρισης κίνησης.

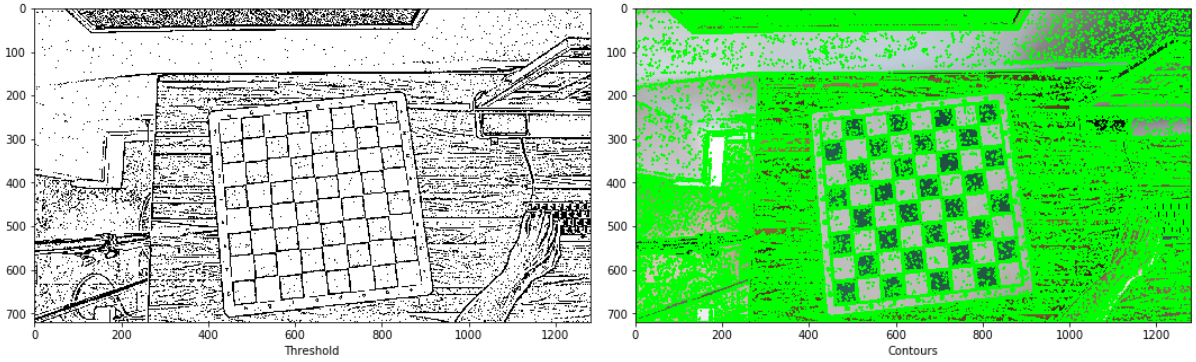
Μέθοδος 1η: Εύρεση Σημείων Περιγράμματος

Ο εντοπισμός πραγματοποιείται με τη χρήση κάμερας που φωτογραφίζει την σκακιέρα. Η αναγνώριση εκτελείτε με τις εξής ενέργειες: Αρχικά, η εφαρμογή λαμβάνει ένα καρέ από την κάμερα και στη συνέχεια με τη χρήση της βιβλιοθήκης OpenCV εντοπίζει τα σημεία περιγράμματος της εικόνας (contour points). Τι είναι τα σημεία αυτά όμως;

Τα **σημεία περιγράμματος (contour points)** μπορούν να εξηγηθούν απλά, ως καμπύλη που ενώνει όλα τα συνεχή σημεία (κατά μήκος του ορίου), με το ίδιο χρώμα ή ένταση. Τα περιγράμματα είναι ένα χρήσιμο εργαλείο για την ανάλυση σχήματος και την ανίχνευση και αναγνώριση αντικειμένων.

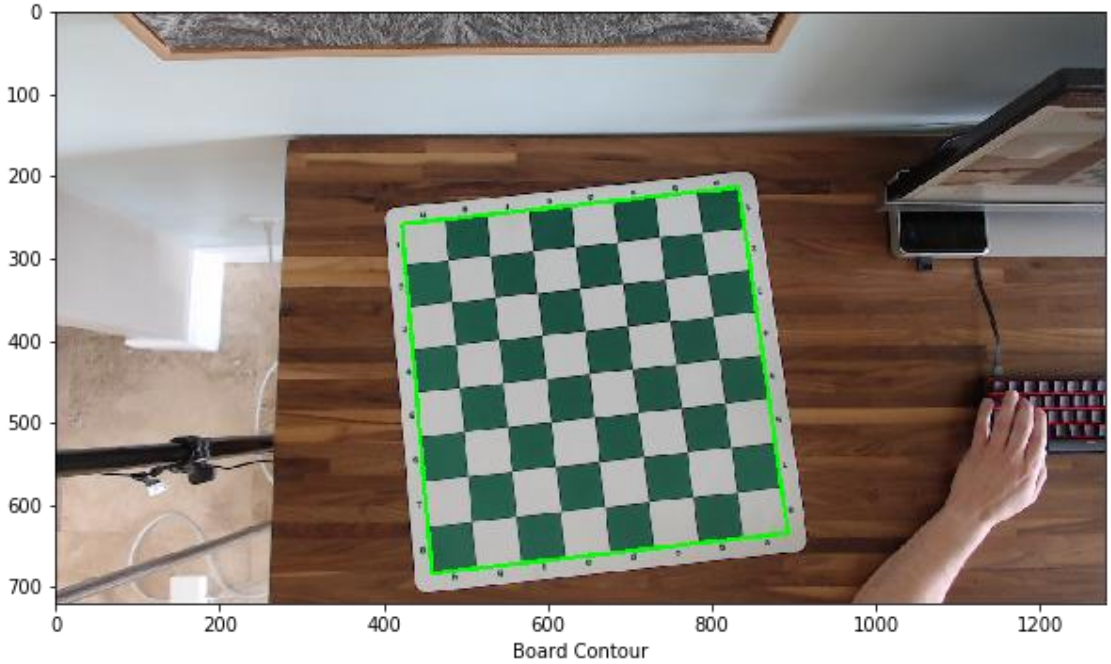
Η εικόνα που φωτογραφίζει η κάμερα περιέχει πολλά άσχετα αντικείμενα. Εμείς θέλουμε να εστιάσουμε στην σκακιέρα. Για να γίνει αυτό πρέπει να χρησιμοποιήσουμε τα σημεία περιγράμματος. Η εφαρμογή σ' αυτό το σημείο λαμβάνει το γενικό καρέ που αποτυπώνει η κάμερα και χρησιμοποιεί εντολές τη βιβλιοθήκης οι οποίες επιστρέφουν ένα σύνολο σημείων

περιγράμματος του καρέ. Στη συνέχεια η εφαρμογή περιορίζει τον αριθμό των σημείων σ' αυτά που αποτελούν τα σημεία της σκακιέρας που έχουν συνέχεια μεταξύ τους.



Εικόνα 3: Σημεία Περιγράμματος (contours) καρέ.

Αυτό επιτυγχάνεται με τον εξής αλγόριθμο: θεωρώντας ότι η ανάλυση της κάμερας είναι 1280x720p η κάμερα τοποθετείται σε τέτοιο ύψος ώστε το πλάτος της στο καρέ να έχει μήκος 400p. Έτσι ο αλγόριθμος αναζητά ένα περίγραμμα τετραγώνου από το σύνολο συνεχών σημείων το οποίο έχει μήκος πλευράς 400p.



Εικόνα 4: Εύρεση της σκακιέρας με την χρήση των σημείων.

Σαν αποτέλεσμα έχουμε την απομόνωση των σημείων της σκακιέρας και κατά συνέπεια την ίδια την σκακιέρα. Έπειτα η εφαρμογή αποκόπτει την εικόνα στα σημεία αυτά του περιγράμματος της σκακιέρας και καταλήγει με μια νέα εικόνα που απεικονίζει μόνο τη σκακιέρα.

Μέθοδος 2η: Χειροκίνητη Μέθοδος Αναγνώρισης

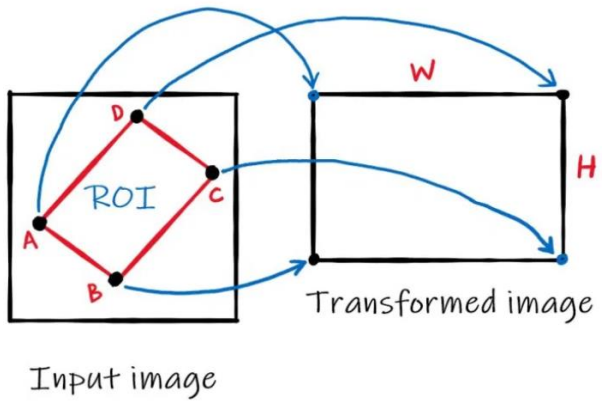
Με τη χρήση της πρώτης μεθόδου, η αναγνώριση της σκακιέρας μπορεί να γίνει αυτόματα ανεξαρτήτως κλίσης και θέσης της σκακιέρας. Ωστόσο, η μέθοδος αυτή μπορεί να δημιουργήσει προβλήματα κατά την αναγνώριση. Υπάρχουν περιπτώσεις όπου ο φωτισμός να δημιουργεί σκιάσεις επάνω στη σκακιέρα με αποτέλεσμα η εύρεση του περιγράμματος της σκακιέρας να υπολογίζεται λάθος. Επιπλέον, αντικείμενα τα οποία μπορεί να βρίσκονται γύρω από τη σκακιέρα αποτελούν επιπλέον πληροφορία η οποία μπορεί να δυσκολέψει την οπτική αναγνώριση εφόσον υπάρχει περίπτωση να δημιουργούν τετράγωνα περιγράμματα 400p τα οποία δεν αποτελούν τη σκακιέρα.

Η χειροκίνητη μέθοδος λύνει τα πιο πάνω προβλήματα λειτουργώντας ως εξής:

Έχοντας τοποθετήσει την κάμερα πάνω από την σκακιέρα, η εφαρμογή τραβάει ένα καρέ από την κάμερα και περιμένει είσοδο από τον χρήστη να ορίσει τις 4 γωνίες της σκακιέρας. Έπειτα, αποκόπτει την σκακιέρα από την υπόλοιπη εικόνα και διορθώνει την προοπτική της σκακιέρας, εφόσον μπορεί να μην βρίσκεται σε απόλυτα σωστή κλίση.

Η διόρθωση προοπτικής πραγματοποιείται με τη χρήση της μεθόδου warpPerspective() της βιβλιοθήκης OpenCV.

```
cv2.warpPerspective(img, M, (int(w), int(h)))
```



Η μέθοδος λαμβάνει ως είσοδο τα 4 άκρα της σκακιέρας (τα οποία παρέχονται από τον χρήστη) καθώς και ένα όρισμα προκαθορισμένων διαστάσεων της τελικής εικόνας.

Η μέθοδος στη συνέχεια αντιστοιχεί τις κορυφές της σκακιέρας με τις κορυφές του προκαθορισμένου ορίσματος και δίνει ως αποτέλεσμα την αρχική εικόνα με διορθωμένη προοπτική.

Αναγνώριση Κίνησης

Η διαδικασία του εντοπισμού της κίνησης αποτελεί το πιο σημαντικό κομμάτι της διαδικασίας, καθώς είναι το κομμάτι το οποίο παράγει την πιο «ζωτική» πληροφορία για το σύστημα. Επομένως η μέθοδος που το σύστημα θα εκτελεί την αναγνώριση της κίνησης θα πρέπει να είναι αποτελεσματική έτσι ώστε και η πληροφορία που θα παράγεται να είναι αξιόπιστη για τελική χρήση.

Η αναγνώριση κίνησης πραγματοποιείται με την ακόλουθη διαδικασία. Η εφαρμογή λαμβάνει το αποκομμένο καρέ της σκακιέρας από το προηγούμενο βήμα (οπτική αναγνώριση) και χωρίζει την εικόνα αυτή σε 64 μικρότερες εικόνες, μία για κάθε κελί. Στη συνέχεια κάθε εικόνα κελιού κατηγοριοποιείται σε μία από τις τρείς κλάσεις οι οποίες είναι βασισμένες στο χρώμα που έχει το πιόνι που βρίσκεται στο κελί (λευκό, μαύρο, κενό) με τη χρήση ενός εκπαιδευμένου μοντέλου.

Το πρώτο μοντέλο που χρησιμοποιήθηκε για την αναγνώριση ήταν ένα μοντέλο το οποίο μπορούσε να κατηγοριοποιήσει κάθε είδος πιονιού (π.χ. λευκός πύργος, μαύρος αξιωματικός, λευκή βασίλισσα κ.λπ.). Αυτό προαπαιτεί τη δημιουργία 13 διαφορετικών κλάσεων, στις οποίες θα μπορούσε να κατηγοριοποιεί την εικόνα που λαμβάνει σαν είσοδο το μοντέλο. Κάθε κλάση είναι ουσιαστικά και κάθε ένα διαφορετικό είδος πιονιού που υπάρχει στο σκάκι, δηλαδή Πύργος, Ιππότης, Αξιωματικός, Βασιλιάς, Βασίλισσα και Στρατιώτης, για κάθε διαφορετικό χρώμα, δηλαδή λευκό και μαύρο, σύνολο 12 συν μία κλάση για τα κενά κελιά.

Αυτό το μοντέλο παρουσίαζε προβλήματα στην επιτυχία της αναγνώρισης, καθώς τα πιόνια είχαν πολύ μικρές διαφορές στα χαρακτηριστικά τους, με αποτέλεσμα το μοντέλο να κατηγοριοποιεί λάθος τα πιόνια.

Γι' αυτό το λόγο το μοντέλο εκπαιδεύτηκε στο να κατηγοριοποιεί τα πιόνια σε τρείς μόνο κλάσεις, όπου η κάθε κλάση δείχνει τί χρώμα πιόνι υπάρχει σε κάθε κελί, λευκό, μαύρο ή κενό, όπως αναφέρθηκε παραπάνω.

Με τον τρόπο αυτό η αποτελεσματικότητα αναγνώρισης του μοντέλου φτάνει σε ποσοστό επιτυχίας 99,8%.

Για να παραχθεί λοιπόν σωστά η πληροφορία χρειάζεται η κάμερα να βρίσκεται σε ορθή κλήση πάνω από τη σκακιέρα και δεν πρέπει τα πιόνια να προεξέχουν από τα κελιά ανεξαρτήτως σε πιά θέση βρίσκονται πάνω στη σκακιέρα.

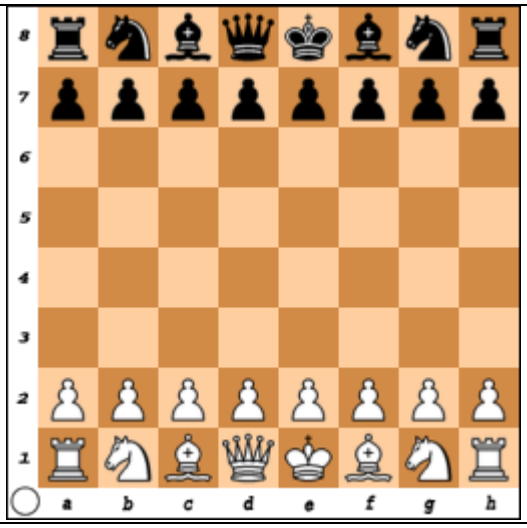
Στη συνέχεια λοιπόν αφού η εφαρμογή κατηγοριοποιήσει τα κελιά σε κάθε κλάση δημιουργεί έναν μονοδιάστατο πίνακα, ο οποίος περιέχει την κάθε κλάση που πρόβλεψε η εφαρμογή για κάθε κελί. Στο τέλος της φάσης αυτής η εφαρμογή στέλνει τον πίνακα αυτόν στο επόμενο βήμα που είναι η εγκυρότητα κίνησης.

Αξίζει να σημειωθεί ότι κατά την αρχικοποίηση του προγράμματος η εφαρμογή δημιουργεί δύο πίνακες. Έναν ο οποίος περιέχει την αρχική κατάσταση της σκακιέρας και ένα αντίγραφό του όπου θα τροποποιείται με βάση τις κινήσεις που γίνονται κατά τη διάρκεια της παρτίδας.

Εύρεση Κίνησης

Ας δούμε πως δουλεύει η μέθοδος αυτή της εύρεσης κίνησης:

Γνωρίζοντας την αρχική θέση των πιονιών κατά την έναρξη της παρτίδας, δημιουργούμε επιπλέον, μία λίστα Python η οποία εμπεριέχει τις κλάσεις των πιονιών που βρίσκονται στα αντίστοιχα τετράγωνα της σκακιέρας. Κάθε θέση της λίστας αποτελεί και ένα τετράγωνο της σκακιέρας.



8 [b, b, b, b, b, b, b, b]
7 [b, b, b, b, b, b, b, b]
6 [_, _, _, _, _, _, _, _]
5 [_, _, _, _, _, _, _, _]
4 [_, _, _, _, _, _, _, _]
3 [_, _, _, _, _, _, _, _]
2 [w, w, w, w, w, w, w, w]
1 [w, w, w, w, w, w, w, w]
a b c d e f g h

Έτσι για παράδειγμα η θέση 0 της λίστας εμπεριέχει την τιμή ‘b’ που αντιπροσωπεύει ένα μαύρο πιόνι, η θέση 16 εμπεριέχει την τιμή ‘_’, δηλαδή ένα κενό κελί και η θέση 48 την τιμή ‘w’, δηλαδή ένα λευκό πιόνι. Επιπλέον εφόσον είναι γνωστή η αρχική θέση των πιονιών και την γνωρίζουμε την ακριβή τους ιδιότητα, δηλαδή όπως αναφέρθηκε για παράδειγμα, η θέση 0 της λίστας εμπεριέχει ένα μαύρο πιόνι και εφόσον είμαστε στην αρχή της παρτίδας το πιόνι αυτό είναι ένας μαύρος Πύργος.

Η πληροφορία αυτή, δηλαδή της ιδιότητας, αποθηκεύεται σε μία μεταβλητή με τη βοήθεια της βιβλιοθήκης Python-chess. Η βιβλιοθήκη[3] συγκεκριμένα αποθηκεύει τις πληροφορίες της παρτίδας σε διαφορετικές οντότητες, οι οποίες είναι κλάσεις αντικειμένων.

Κλάση	**Πληροφορία**
Square	Η κλάση αυτή εμπεριέχει πληροφορίες για τα τετράγωνα της σκακιέρας.
Piece	Εμπεριέχει τη πληροφορία για τα πιόνια.
Move	Εμπεριέχει πληροφορίες για τις κινήσεις.
Board	Η κλάση αυτή εμπεριέχει συνολικά όλες τις μεθόδους που παρέχει η βιβλιοθήκη και χρησιμοποιεί τις πιο πάνω κλάσεις ως οντότητες.

[3] Επιπλέον πληροφορίες για την βιβλιοθήκη: https://python-chess.readthedocs.io/en/latest/core.html

```
import chess
board = chess.Board()
```

Με την εντολή αυτή αρχικοποιούμε ένα αντικείμενο τύπου Board το οποίο εμπεριέχει την ακριβή πληροφορία για τα πιόνια, τις κινήσεις και την γενικότερη κατάσταση της παρτίδας. Έτσι, μπορούμε να χρησιμοποιήσουμε τις μεθόδους που μας παρέχει η βιβλιοθήκη για να δούμε εάν μία κίνηση είναι έγκυρη. Αυτό επιτυγχάνεται καλώντας την μέθοδο Move, η οποία παίρνει σαν παραμέτρους δύο σκακιστικά τετράγωνα τα οποία είναι αντικείμενα τύπου Square.

```
board.Move(from: chess.Square, to: chess.Square)
```

Η μέθοδος αυτή επιστρέφει λογική τιμή σωστού-λάθους, εάν η κίνηση που δόθηκε για τα συγκεκριμένα τετράγωνα είναι έγκυρη ή όχι αντίστοιχα. Έτσι, το μόνο πού πρέπει να κάνουμε είναι να δώσουμε τα δύο τετράγωνα όπου ανιχνεύτηκε η κίνηση. Στη διαδικασία αυτή όμως υπάρχει ένα πρόβλημα.

Πρόβλημα Αντιστοίχισης Τετραγώνων

Όπως είδαμε πιο πάνω, οι κλάσεις που προβλέπει η εφαρμογή κατά την οπτική αναγνώριση αποθηκεύονται σε έναν μονοδιάστατο πίνακα, με κάθε θέση του πίνακα να είναι ουσιαστικά και ένα τετράγωνο του ταμπλό, ξεκινώντας την αρίθμηση από το πάνω αριστερό τετράγωνο.

Ωστόσο η βιβλιοθήκη της Python-chess αποθηκεύει τις πληροφορίες για τα τετράγωνα ξεκινώντας την αρίθμηση της από το κάτω αριστερά τετράγωνο του ταμπλό (A1), όπως αναγράφεται στη σκακιέρα, έτσι το τετράγωνο A1 έχει την τιμή 0, το B1 την τιμή 1, το Γ1 την τιμή 2 κ.ο.κ.

Python Chess Array	Generic Python Array
[56 57 58 59 60 61 62 63]	[00 01 02 03 04 05 06 07]
[48 49 50 51 52 53 54 55]	[08 09 10 11 12 13 14 15]
[40 41 42 43 44 45 46 47]	[16 17 18 19 20 21 22 23]
[32 33 34 45 46 47 48 49]	[24 25 26 27 28 29 30 31]
[24 25 26 27 28 29 30 31]	[32 33 34 35 36 37 38 39]
[16 17 18 19 20 21 22 23]	[40 41 42 43 44 45 46 47]
[08 09 10 11 12 13 14 15]	[48 49 50 51 52 53 54 55]
[00 01 02 03 04 05 06 07]	[56 57 58 59 60 61 62 63]

Επομένως, είναι απαραίτητη η αντιστοίχιση των τιμών αυτών έτσι ώστε να μπορεί να υπολογίσει η βιβλιοθήκη την εγκυρότητα των κινήσεων ανάμεσα σε δύο τετράγωνα, χρησιμοποιώντας τις θέσεις του μονοδιάστατου πίνακα που εμπεριέχουν την πληροφορία για το πια ήταν η κίνηση που πραγματοποιήθηκε.

Το πρόβλημα αυτό λύνεται υλοποιώντας μία συνάρτηση αντιστοίχισης (mapping) η οποία εκτελεί αυτή τη διαδικασία. Η συνάρτηση αυτή βρίσκεται μέσα στο φάκελο *helpers*, στο αρχείο *mapper.py*. Έτσι δίνοντας τις θέσεις του μονοδιάστατου πίνακα στον mapper, μετατρέπουμε τις θέσεις αυτές σε τετράγωνα τύπου chess.Square, έτσι ώστε να μπορούν να χρησιμοποιηθούν από την βιβλιοθήκη.

Η εύρεση της κίνησης πραγματοποιείται συγκρίνοντας δύο στιγμιότυπα της παρτίδας, την κατάσταση της παρτίδας έως αυτή τη στιγμή και την κατάσταση της αμέσως αφού εκτελεστεί μία κίνηση. Συγκεκριμένα, μπορούμε να «τραβήξουμε» ένα στιγμιότυπο της παρτίδας κατά παραγγελία πιέζοντας το πλήκτρο Space. Αυτό θα δημιουργήσει έναν νέο μονοδιάστατο πίνακα με τη κατάσταση της σκακιέρας κατά τη στιγμή που πατήθηκε το πλήκτρο. Η δημιουργία αυτή του στιγμιότυπου υποδεικνύει και την κίνηση ουσιαστικά του παίκτη στον γύρο. Μόλις η εφαρμογή αποκτήσει τον νέο μονοδιάστατο πίνακα, θα τον συγκρίνει με τον πρώτο, ο οποίος περιέχει την τελευταία κατάσταση της παρτίδας και θα βρει τις διαφορές τους.

Συγκεκριμένα, η σύγκριση πραγματοποιείται με την «λογική» αφαίρεση των δύο πινάκων. Θεωρώντας πως η κίνηση είναι σωστή, οι δύο πίνακες θα πρέπει να διαφέρουν μόνο σε δυο σημεία, όπου τα σημεία αυτά τελικά θα είναι η κίνηση που πραγματοποιήθηκε. Λόγου χάρη, υποθέτουμε πως πραγματοποιείται η πρώτη κίνηση των λευκών στην αρχή της παρτίδας:

```
last_board
[b b b b b b b b]
[b b b b b b b b]
[_ _ _ _ _ _ _ _]
[_ _ _ _ _ _ _ _]
[_ _ _ _ _ _ _ _]
[_ _ _ _ _ _ _ _]
[w w w w w w w w]
[w w w w w w w w]
```

```
current_board
[b b b b b b b b]
[b b b b b b b b]
[_ _ _ _ _ _ _ _]
[_ _ _ _ _ _ _ _]
[_ _ _ w _ _ _ _]
[_ _ _ _ _ _ _ _]
[w w w _ w w w w]
[w w w w w w w w]
```

Η κίνηση αυτή θα δημιουργήσει αντίστοιχα δύο μονοδιάστατους πίνακες με τις αντίστοιχες τιμές όπως στο σχήμα. Στη συνέχεια οι πίνακες αυτοί περνούν σαν είσοδο σε μία συνάρτηση η οποία συγκρίνει τους πίνακες θέση προς θέση και δίνει ως αποτέλεσμα τις δύο θέσεις όπου διαφέρουν.

last_board	current_board
[b_{00} b_{01} b_{02} b_{03} b_{04} b_{05} b_{06} b_{07}]	[b_{00} b_{01} b_{02} b_{03} b_{04} b_{05} b_{06} b_{07}]
[b_{08} b_{09} b_{10} b_{11} b_{12} b_{13} b_{14} b_{15}]	[b_{08} b_{09} b_{10} b_{11} b_{12} b_{13} b_{14} b_{15}]
[_16 _17 _18 _19 _20 _21 _22 _23]	[_16 _17 _18 _19 _20 _21 _22 _23]
[_24 _25 _26 _27 _28 _29 _30 _31]	[_24 _25 _26 _27 _28 _29 _30 _31]
[_32 _33 _34 _35 _36 _37 _38 _39]	[_32 _33 _34 **w_{35}** _36 _37 _38 _39]
[_40 _41 _42 _43 _44 _45 _46 _47]	[_40 _41 _42 _43 _44 _45 _46 _47]
[w_{48} w_{49} w_{50} **w_{51}** w_{52} w_{53} w_{54} w_{55}]	[w_{48} w_{49} w_{50} _51 w_{52} w_{53} w_{54} w_{55}]
[w_{56} w_{57} w_{58} w_{59} w_{60} w_{61} w_{62} w_{63}]	[w_{56} w_{57} w_{58} w_{59} w_{60} w_{61} w_{62} w_{63}]

Στο σχήμα αναγράφονται με τη μικρή γραμματοσειρά ο αριθμός της κάθε θέσης του πίνακα. Έτσι λοιπόν παρατηρούμε ότι η συγκεκριμένη κίνηση αφορά τις θέσεις 51 και 35 στους αντίστοιχους πίνακες. Οι δύο θέσεις αυτές είναι και το αποτέλεσμα της συνάρτησης. Τέλος οι θέσεις αυτές περνούν στη συνάρτηση αντιστοίχισης (mapper) που αναφέρθηκε προηγουμένως και οι θέσεις του μονοδιάστατου πίνακα μετατρέπονται σε θέσεις τύπου chess.Square με σκοπό τον έλεγχο εγκυρότητάς τους.

Μόλις η εγκυρότητα ολοκληρωθεί το τελευταίο στιγμιότυπο της παρτίδας (last_board) αντικαθίσταται με το τρέχον (current_board) έτσι ώστε να είναι έτοιμο να συγκριθεί με την επόμενη κίνηση που θα εκτελέσουν οι παίκτες. Εάν η κίνηση δεν εγκριθεί, το στιγμιότυπο δεν αντικαθίσταται και πρέπει κατά παραγγελία να δημιουργηθεί νέο τρέχον στιγμιότυπο για να πραγματοποιηθεί εκ νέου η σύγκριση με το τελευταίο στιγμιότυπο. Η διαδικασία επαναλαμβάνεται μέχρι η κίνηση να είναι επιτρεπτή.

Εγκυρότητα κίνησης

Η εφαρμογή έχει σχεδιαστεί με τέτοιο τρόπο ώστε η οπτική αναγνώριση κίνησης της σκακιέρας να είναι ανεξάρτητη από την πραγματοποίηση σκακιστικής κίνησης και της κατάστασης της παρτίδας.

Με άλλα λόγια στα πλαίσια της αναγνώρισης κίνησης υπάγεται μόνο η εύρεση της φυσικής κίνησης ενός πιονιού. Η εγκυρότητα και η μετάφραση της φυσικής αυτής κίνησης σε σκακιστική κίνηση είναι δουλειά της εγκυρότητας κίνησης.

Η εύρεση της σκακιστικής κίνησης πραγματοποιείται με την αφαίρεση των δύο μονοδιάστατων πινάκων από το προηγούμενο βήμα. Όπως αναφέρθηκε πιο πάνω ο ένας πίνακας εμπεριέχει την αρχική κατάσταση της σκακιέρας και ο δεύτερος την κατάσταση της σκακιέρας στο τελευταίο καρέ.

Έτσι λοιπόν εάν αφαιρέσουμε τους δύο πίνακες το αποτέλεσμα είναι δύο θέσεις των πινάκων όπου αποτυπώνουν την κίνηση που έγινε στην σκακιέρα.

Η επιλογή της χρήσης μονοδιάστατου πίνακα έγινε με σκοπό την καλύτερη απόδοση της εφαρμογής και τον μαθηματικών πράξεων που χρησιμοποιεί για την εύρεση της κίνησης. Εάν χρησιμοποιούσαμε δυσδιάστατο πίνακα για να αναπαραστήσουμε την σκακιέρα θα χρειαζόμασταν μεγαλύτερη επεξεργαστική ισχύ. Έτσι λοιπόν με τη χρήση μονοδιάστατου πίνακα καταφέρνουμε η εφαρμογή να εκτελεί πιο γρήγορα την εύρεση της κίνησης.

Η μετατροπή της σκακιέρας (που την καθιστά δυσδιάστατο πίνακα) σε μονοδιάστατο γίνεται απλά τοποθετώντας σε διαδοχική σειρά τα κελιά της σκακιέρας ξεκινώντας από το πάνω αριστερό κελί (α8).

Η εφαρμογή χρησιμοποιεί τη βιβλιοθήκη Python Chess για να αποθηκεύσει την κατάσταση της παρτίδας. Η βιβλιοθήκη μπορεί να δημιουργεί προγραμματιστικά αντικείμενα τα οποία αναπαριστούν τη σκακιέρα, τα πιόνια, τις κινήσεις και γενικότερα την κατάσταση της παρτίδας.

Στο σημείο αυτό η εφαρμογή μπορεί να εντοπίσει την σκακιστική κίνηση που ολοκληρώθηκε με βάση τη φυσική κίνηση τροφοδοτώντας τις θέσεις του μονοδιάστατου πίνακα σαν είσοδο σε έναν αλγόριθμο της Python Chess. Επίσης με τη διαδικασία αυτή, η βιβλιοθήκη μπορεί να εντοπίσει εάν η κίνηση είναι έγκυρη βάση των κανόνων και της κατάστασης της παρτίδας. Αφού ενημερωθεί η κατάσταση της παρτίδας, κωδικοποιείται σε μορφή fen και περνάει στο επόμενο στάδιο της διαδικασίας.

Τέλος, στο σημείο αυτό η εφαρμογή αντικαθιστά το περιεχόμενο του πρώτου μονοδιάστατου πίνακα με του δεύτερου, έτσι ώστε στο επόμενο καρέ να συγκριθεί κίνηση που μόλις ολοκληρώθηκε με την μελλοντική κίνηση που θα πραγματοποιηθεί.

Έξοδος σε σύστημα

Όπως αναφέρθηκε και στην αρχή της εργασίας εξωτερικό σύστημα εννοούμε μία Web εφαρμογή η οποία λαμβάνει την κωδικοποιημένη μορφή μιας παρτίδας σκάκι και την μετατρέπει σε εικόνα κατανοητή σε απλούς χρήστες.

Επίσης η εφαρμογή αποθηκεύει το καρέ κάθε έγκυρης κίνησης σε έναν φάκελο για μελλοντική ανάλυση της παρτίδας.

Η εφαρμογή επικοινωνεί με την Web εφαρμογή κάνοντας ένα REST POST request το οποίο περιέχει την κωδικοποιημένη κατάσταση της σκακιέρας σε μορφή fen. Με τη σειρά της η web εφαρμογή λαμβάνει την κωδικοποίηση αυτή και την μετατρέπει σε εικόνα.

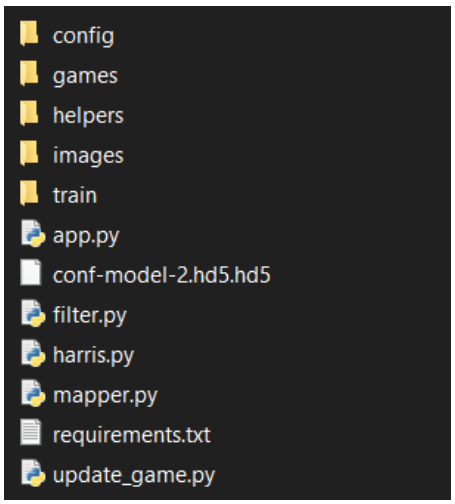
Επιπλέον, η εφαρμογή αποθηκεύει το τελικό PGN της παρτίδας τοπικά στον υπολογιστή καθώς επίσης και τα καρέ της κάθε κίνησης.

Τεχνική Περιγραφή και Εγκατάσταση της εφαρμογής

Εγκατάσταση

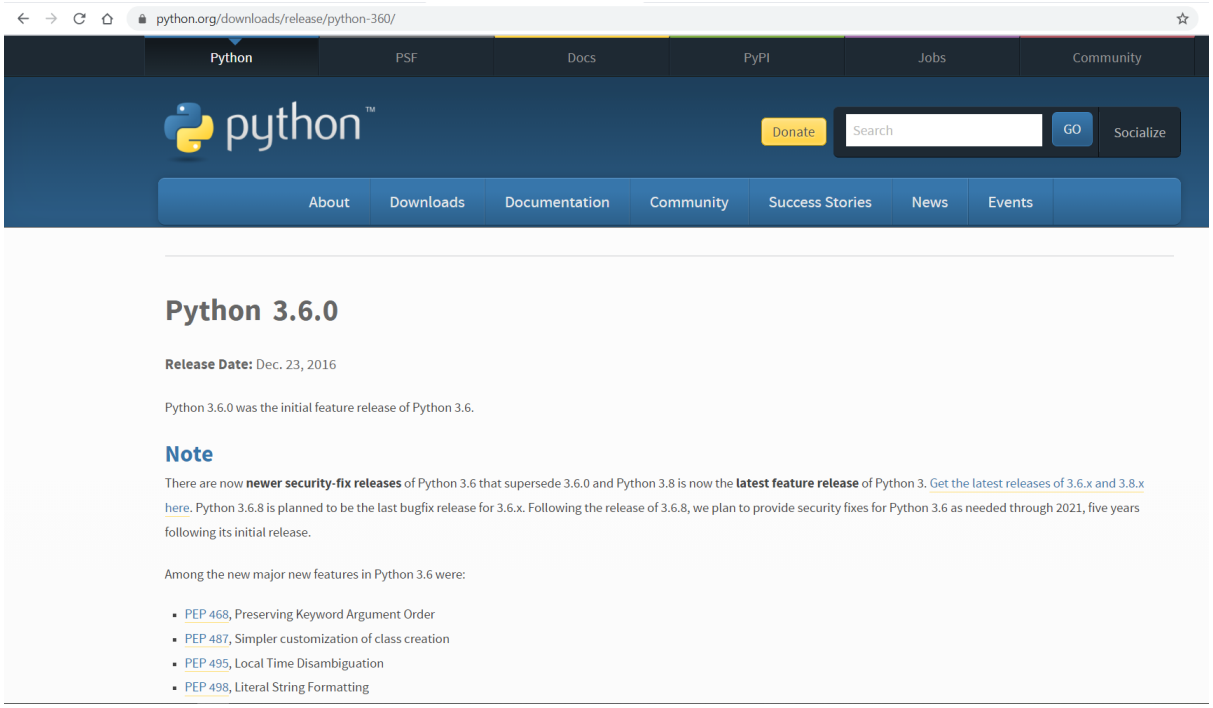
Για να λειτουργήσει η εφαρμογή θα πρέπει αρχικά να εγκαταστήσουμε Python καθώς επίσης και τις απαραίτητες βιβλιοθήκες της. Οι βιβλιοθήκες που πρέπει να εγκαταστήσουμε βρίσκονται στο αρχείο **requirements.txt**.

Για καλύτερη οργάνωση θα εγκαταστήσουμε το πρόγραμμα και όλες τις βιβλιοθήκες Python σε ένα ξεχωριστό «περιβάλλον» με τη χρήση της βιβλιοθήκης **virtualevn**.

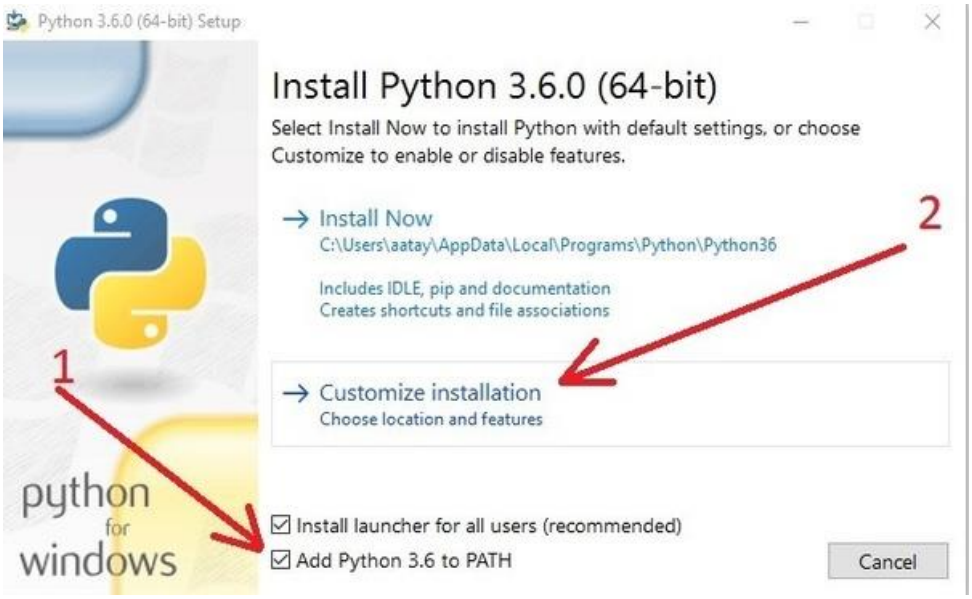


Στον φάκελο του πηγαίου κώδικα υπάρχουν **5 .py αρχεία** τα οποία αποτελούν την ίδια την εφαρμογή, **1 αρχείο το οποίο περιέχει τις απαραίτητες βιβλιοθήκες** που πρέπει να εγκαταστήσουμε και **1 αρχείο .pdf**, στο οποίο βρίσκονται οι οδηγίες εγκατάστασης. Επιπλέον, υπάρχει **ένα αρχείο .hd5**, το οποίο αποτελεί το εκπαιδευμένο μοντέλο το οποίο χρησιμοποιεί η εφαρμογή για την οπτική αναγνώριση.

Λόγο συγκεκριμένων εξαρτήσεων που έχουν μερικές από τις βιβλιοθήκες, είναι απαραίτητο να εγκαταστήσουμε την έκδοση 3.6.0 της Python, την οποία μπορούμε να την βρούμε στο site της.



Επιπλέον, θα πρέπει κατά την εγκατάσταση να συμπεριλάβουμε την θέση εγκατάστασης της γλώσσας στο path. Αυτό μπορεί να γίνει κατά την εγκατάσταση της γλώσσας, με σκοπό να μπορούμε να εκτελούμε εντολές της Python μέσο του Power Shell και του Windows Terminal.



Στη συνέχεια, ανοίγουμε ένα Terminal ως διαχειριστής στον φάκελο όπου θέλουμε να εγκαταστήσουμε το project. Αφού μπούμε στον φάκελο μέσο του terminal, εκεί εκτελούμε την εντολή:

```
pip install virtualenv
```

Μόλις η εγκατάσταση ολοκληρωθεί, μέσα στον φάκελο τρέχουμε:

```
virtualenv deltachess-env
```

Με τη διαδικασία αυτή δημιουργούμε ένα εικονικό περιβάλλον Python, εκεί θα εγκαταστήσουμε όλες τις βιβλιοθήκες που χρειάζεται το πρόγραμμα. Μετά, μπαίνουμε (από τον explorer) στον νέο φάκελο που δημιουργήσαμε και κάνουμε επικόλληση όλα τα αρχεία της εφαρμογής.

ΣΗΜΕΙΩΣΗ

Σε περιβάλλον Windows ενδέχεται οι εκτέλεση εντολών και προγραμμάτων εντός του τερματικού να είναι απενεργοποιημένη. Για την επίλυση του προβλήματος αυτού αρκεί να εκτελέσουμε την εντολή στο τερματικό:

```
Set-ExecutionPolicy RemoteSigned
```

Στη συνέχεια εάν θέλουμε να επιστρέψουμε το σύστημα στην αρχική του κατάσταση, εκτελούμε την εντολή:

```
Set-ExecutionPolicy Restricted
```

Σε κάθε περίπτωση, εάν χρησιμοποιήσουμε κάποια από τις παραπάνω εντολές είναι απαραίτητο να κλείσουμε τα τερματικά και να γίνει επανεκκίνηση του υπολογιστή για να εφαρμοστούν οι αλλαγές.

Έπειτα, εφόσον το περιβάλλον Python έχει δημιουργηθεί με επιτυχία, στο τερματικό τρέχουμε:

```
cd .\deltachess-env\Scripts\
.\activate
```

Με τις εντολές αυτές, ενεργοποιούμε το εικονικό περιβάλλον που δημιουργήσαμε πιο πριν και έτσι, ότι βιβλιοθήκες Python εγκαταστήσουμε, θα εγκατασταθούν μόνο για το περιβάλλον αυτό και θα τρέχουν μόνο εντός αυτού. Έτσι, κρατάμε καθαρό το καθολικό περιβάλλον της Python που κάναμε εγκατάσταση και περιορίζουμε όλες τις απαιτήσεις (dependencies) της εφαρμογής σε ένα περιβάλλον, έτσι η εφαρμογή μπορεί να είναι μεταφέρσιμη και να δουλεύει παντού.

Το τελικό βήμα είναι να εγκαταστήσουμε τις βιβλιοθήκες στο περιβάλλον αυτό, επομένως μετακινούμαστε στον φάκελο όπου βρίσκεται το αρχείο requirements.txt και εκτελούμε:

```
pip install -r requirements.txt
```

Μόλις τελειώσει η εγκατάσταση, η εφαρμογή είναι έτοιμη και μπορούμε να εκτελέσουμε το αρχείο app.py.

```
python .\app.py
```

Στο φάκελο της εφαρμογής υπάρχουν διάφορα άλλα αρχεία τα οποία θα αναλύσουμε ενδεικτικά πιο κάτω:

- app.py
  Αποτελεί την αφετηρία της εφαρμογής. Είναι το αρχείο που πρέπει να εκτελεστεί για να ξεκινήσει η εφαρμογή.

- filter.py
  Χρησιμοποιείται από το κεντρικό αρχείο (app.py) και εμπεριέχει την κλάση Filter() η οποία είναι υπεύθυνη για τον χειρισμό και την επεξεργασία των στιγμιότυπων που παράγει η κάμερα για την εφαρμογή.

- harris.py
  Χρησιμοποιείται επίσης από το κεντρικό αρχείο και εμπεριέχει την κλάση Cropper(). Η κλάση αυτή είναι υπεύθυνη για όλες τις ενέργειες οπτικής αναγνώρισης εικόνας. Είναι ο διάμεσος εφαρμογής και κάμερας και παράγει στιγμιότυπα τα οποία στη συνέχεια χρησιμοποιούνται ως είσοδοι στην κλάση Filter().

- mapper.py
  Είναι το αρχείο το οποίο εμπεριέχει την βοηθητική κλάση Mapper() η οποία χρησιμοποιείται από την κεντρική εφαρμογή για την επίλυση του *«Προβλήματος Αντιστοίχισης Τετραγώνων»*.

Στον φάκελο της εφαρμογής εμπεριέχονται και φάκελοι οι οποίοι χρησιμοποιούνται για την καλύτερη οργάνωση αρχείων που απαιτεί η εφαρμογή, εκτός των εκτελέσιμων .py αρχείων.

- config
  Εμπεριέχει ένα αρχείο config.json, το οποίο περιέχει διάφορες παραμέτρους για την εφαρμογή και την εκπαίδευση του μοντέλου. Περισσότερα θα αναφερθούν στην επόμενη παράγραφο.

- images
  Εδώ εμπεριέχονται οι φωτογραφίες της κατάστασης της παρτίδας που παράγει η εφαρμογή σε κάθε παιχνίδι.

- games
  Στον φάκελο αυτό αποθηκεύονται τα τελικά PGN αρχεία της παρτίδας κατά τη λήξη του παιχνιδιού.

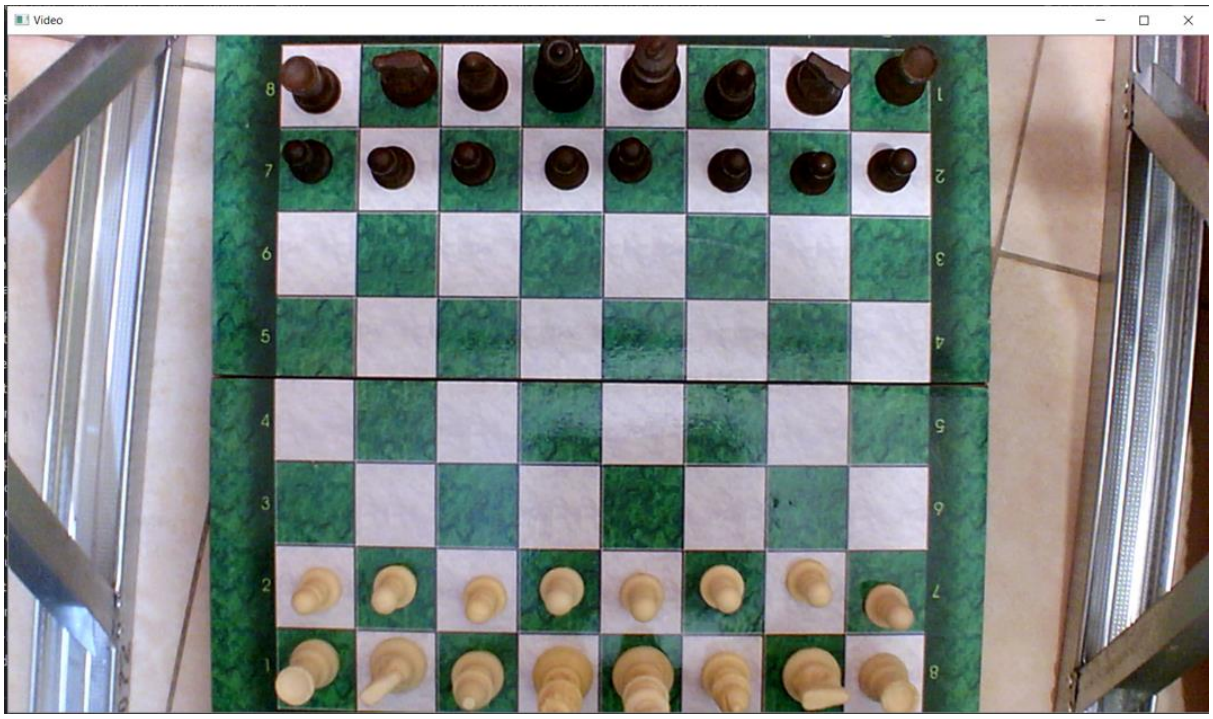
- train
  Στον φάκελο train, εμπεριέχονται όλα τα απαραίτητα αρχεία για την εκπαίδευση του μοντέλου. Περισσότερες πληροφορίες στο κεφάλαιο *"Εκμάθηση Μοντέλου και Datasets"*.

Εκτέλεση

Κατά την εκτέλεση της εφαρμογής θα μας ζητηθεί να εισάγουμε στο τερματικό μερικά στοιχεία για την παρτίδα, τα οποία θα αποθηκευτούν κατά την λήξη της εκτέλεσης της εφαρμογής σε στο αρχείο PGN, τα οποία θα περιγράφουν την παρτίδα.

```
Welcome! Please enter information about the game.
White Player Name: Player A
Black Player Name: Player B
Event Title: Friendly Chess Match
Initializing Camera...
```

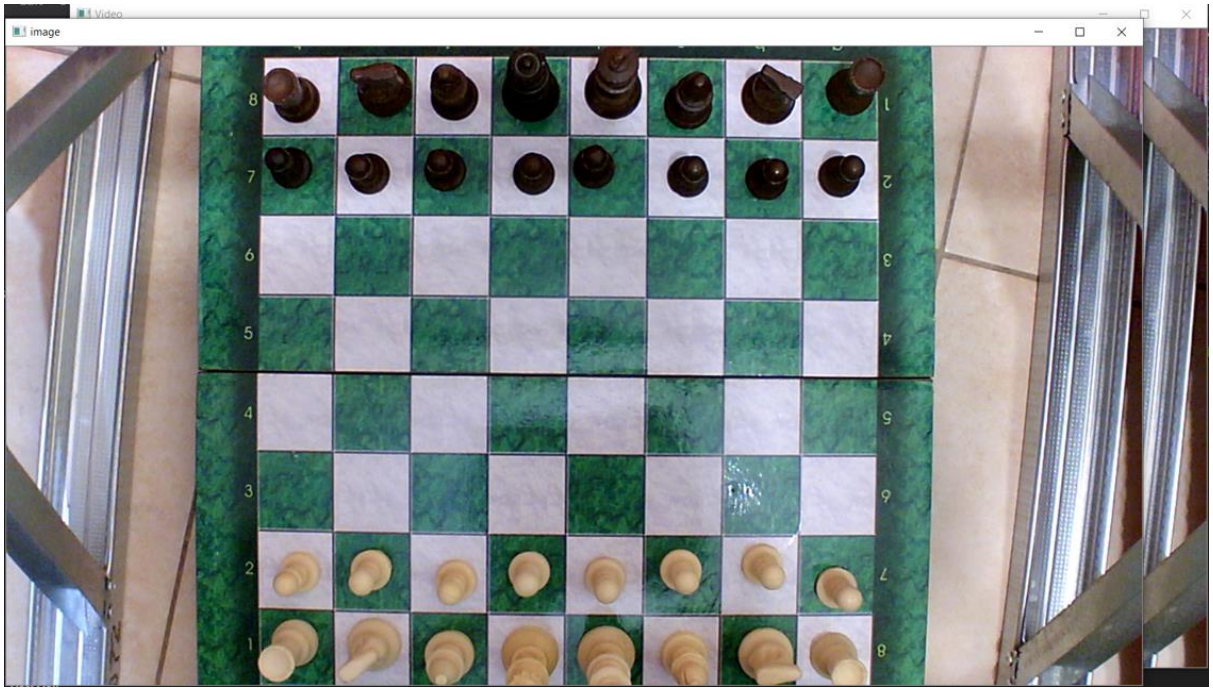
Στην συνέχεια, η κάμερα θα ενεργοποιηθεί και στο σημείο αυτό μπορούμε να κάνουμε τοποθετήσουμε την κάμερα στη σωστή θέση, έχοντας καθαρή λήψη της σκακιέρας από πάνω.



Μόλις είμαστε ικανοποιημένοι με την λήψη, επιβεβαιώνουμε τη θέση πιέζοντας οποιοδήποτε πλήκτρο. Έπειτα, ένα νέο παράθυρο θα εμφανιστή με το καρέ της σκακιέρας, στο σημείο αυτό πρέπει να ορίσουμε τις γωνίες της σκακιέρας κάνοντας κλικ στις γωνίες της, πάνω στο καρέ.

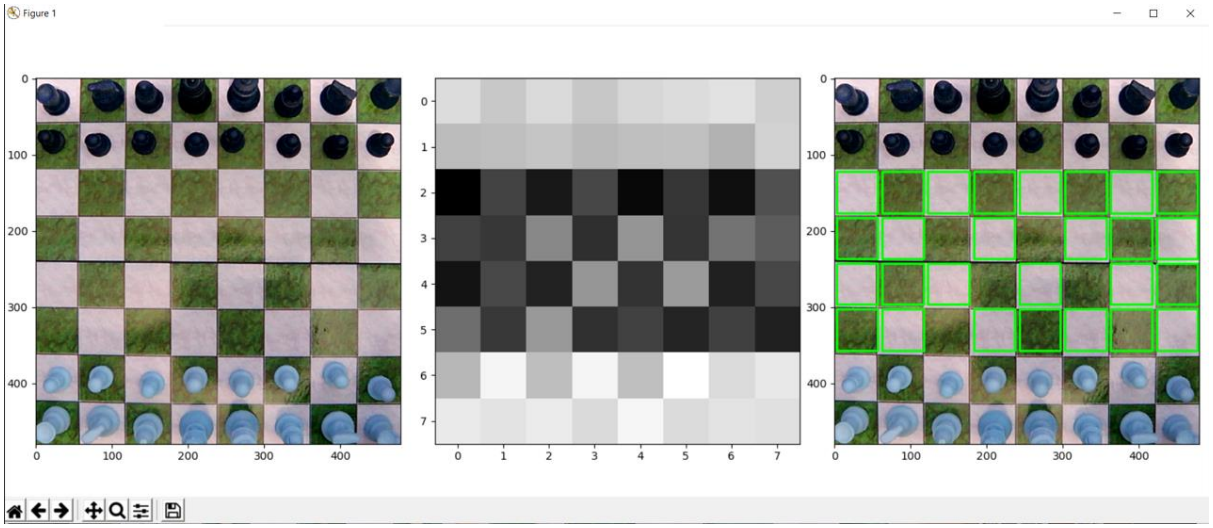
ΣΗΜΕΙΩΣΗ

Οι γωνίες πρέπει να επιλεχθούν με την εξής σειρά:
ΑΝΩ ΑΡΙΣΤΕΡΑ – ΑΝΩ ΔΕΞΙΑ – ΚΑΤΩ ΑΡΙΣΤΕΡΑ – ΚΑΤΩ ΔΕΞΙΑ



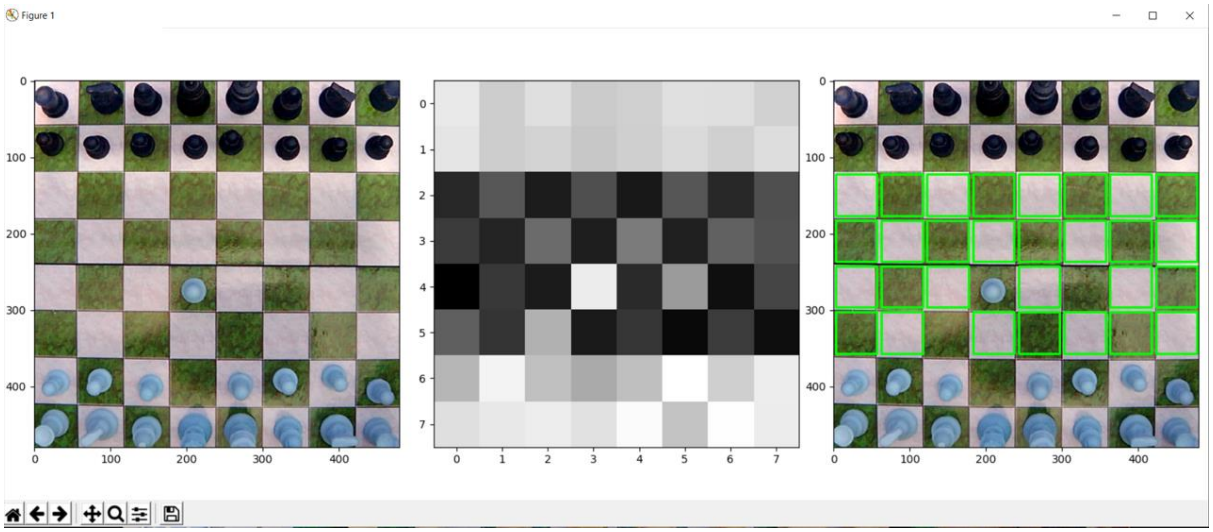
Αμέσως μετά το 4^{ο} κλικ, η εφαρμογή θα κρατήσει μόνο την σκακιέρα, θα διορθώσει την προοπτική και θα εμφανίσει ένα παράθυρο με το τελικό αποτέλεσμα. Στο σημείο αυτό επιβεβαιώνουμε το αποτέλεσμα πιέζοντας οποιοδήποτε πλήκτρο.

Εάν έχουμε ενεργοποιημένη τη ρύθμιση `debug: true` στο config.json, τότε θα αναδυθεί ένα νέο παράθυρο το οποίο θα απεικονίζει τη σκακιέρα και ενδιάμεσα στάδια επεξεργασίας του καρέ.



Στη συνέχεια κλείνουμε όλα τα αναδυόμενα παράθυρα (εκτός του πρώτου) και μπορούμε να εκτελέσουμε την πρώτη κίνηση. Το πρώτο παράθυρο χρησιμοποιείται για έλεγχο των καρέ και μπορούμε να βλέπουμε σε πραγματικό χρόνο τη λήψη της κάμερας.

Μόλις ολοκληρώσουμε την κίνηση, επιβεβαιώνουμε το καρέ πατώντας οποιοδήποτε πλήκτρο.



Ομοίως, όπως και στο προηγούμενο βήμα, εάν έχουμε ενεργή την ρύθμιση `debug`, θα δούμε το βοηθητικό παράθυρο με τα στάδια επεξεργασίας του καρέ. Σε κάθε περίπτωση όμως, εάν η κίνηση είναι έγκυρη, θα εμφανιστεί στη κονσόλα η νέα κατάσταση της παρτίδας και ένα μήνυμα που θα επισημαίνει τη σειρά του επόμενου παίκτη.

```
r n b q k b n r
P P P P P P P P
. . . . . . . .
. . . . . . . .
. . . P . . . .
. . . . . . . .
P P P . P P P P
R N B Q K B N R

Turn: BLACK
```

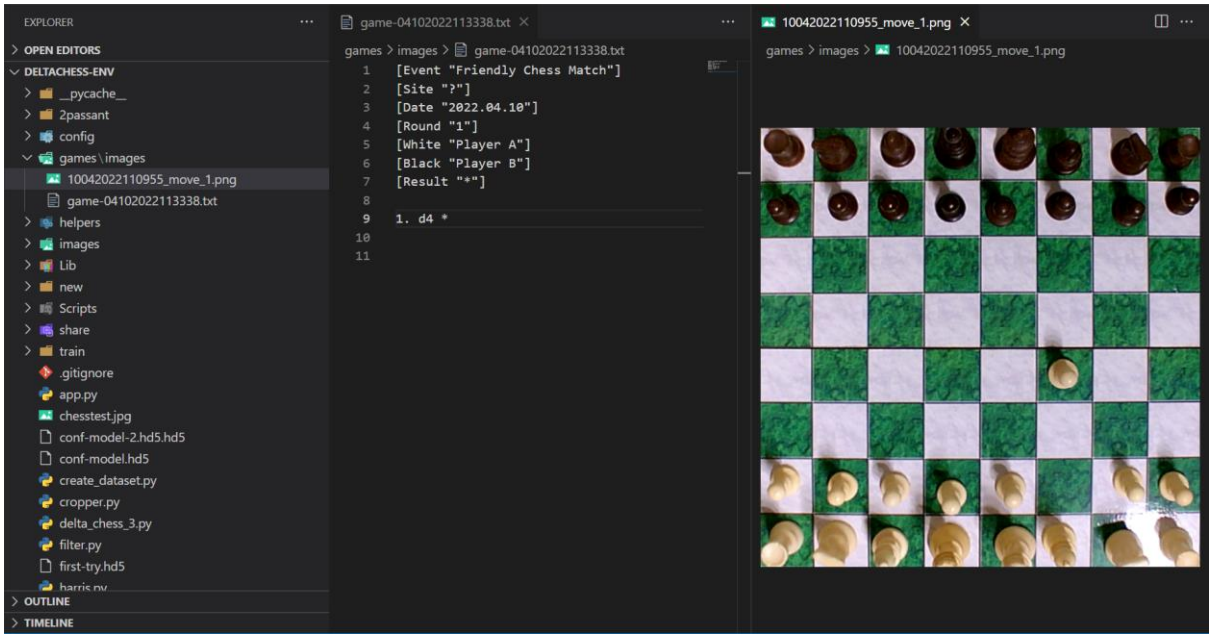
ΣΗΜΕΙΩΣΗ

Εάν για κάποιο λόγο, η κίνηση δεν αναγνωριστεί από το σύστημα, επιστρέφουμε το πιόνι στην αρχική του θέση, εκτελούμε την διαδικασία επιβεβαίωσης του καρέ και επαναλαμβάνουμε την κίνηση.

Επιπλέον, στο σημείο αυτό η εφαρμογή επικοινωνεί με το εξωτερικό σύστημα, στέλνοντας την κατάσταση της παρτίδας σε μορφή FEN.

Η διαδικασία αυτή επαναλαμβάνεται για κάθε παίκτη εναλλάξ μέχρις ότου κλείσουμε σταματήσουμε την εφαρμογή, το οποίο επιτυγχάνεται κλείνοντας το αρχικό παράθυρο ή πιέζοντας το πλήκτρο Esc.

Κατά τη λήξη της εκτέλεσης, όλα τα καρέ των έγκυρων κινήσεων αποθηκεύονται μέσα στον φάκελο images, καθώς επίσης και το PGN της παρτίδας με όλα τα στοιχεία της.



Εκμάθηση μοντέλου και Datasets

Εκμάθηση μοντέλου

Όπως αναφέρθηκε και στη περιγραφή της αναγνώρισης κίνησης, η εφαρμογή χρησιμοποιεί ένα εκπαιδευμένο μοντέλο για να μπορέσει να κατηγοριοποιήσει τα πιόνια με βάση το χρώμα τους.

Η βασική ιδέα στην κατηγοριοποίηση των πιονιών με βάση το χρώμα τους μόνο και όχι ανάλογα με το αξίωμα τους είναι πρώτα, η ευκολότερη εκπαίδευση του μοντέλου αλλά και η βέλτιστη απόδοση. Είναι δύσκολο ένα πιόνι να κατηγοριοποιηθεί μόνο με το αξίωμα του διότι είναι δύσκολο να δημιουργηθούν κατάλληλα σημεία περιγράμματος έτσι ώστε να αναγνωρίζει το μοντέλο τι πιόνι είναι διότι από την οπτική όπου η κάμερα τα βιντεοσκοπεί, όλα μοιάζουν μεταξύ τους. Αυτό προκαλεί σημαντικές αποκλίσεις στα αποτελέσματα της οπτικής αναγνώρισης και η συνολική απόδοση της εφαρμογής δεν ήταν ικανοποιητική.

Επιπλέον για την υλοποίηση αυτής της εκπαίδευσης αλλά και για τη βελτίωση της χωρίς να αλλάξουν οι κλάσεις κατηγοριοποίησης, η διαδικασία δημιουργίας δειγμάτων ήταν χρονοβόρα και περίπλοκη, καθώς χρειαζόταν τεράστιος όγκος φωτογραφικών δειγμάτων. Συγκεκριμένα, ένα πλήθος 10.000 φωτογραφιών μπορούσε βοηθήσει το μοντέλο να εκπαιδευτεί στο να κατηγοριοποιεί τα πιόνια με βάση το χρώμα τους έχοντας ποσοστό επιτυχής αναγνώρισης σχεδόν 100%, ενώ αντίθετα, το ίδιο πλήθος μπορούσε να εκπαιδεύσει το μοντέλο να κατηγοριοποιεί τα πιόνια βάση αξιώματός τους, έχοντας επιτυχία αναγνώρισης κάτω από 90%.

Η λογική της κατηγοριοποίησης όπου τελικά χρησιμοποιήθηκε δουλεύει σε συνδυασμό με τις βιβλιοθήκες εγκυρότητας κίνησης δίνοντας πολύ ικανοποιητικά αποτελέσματα χωρίς να αναγκάζεται η εφαρμογή να βασίζεται εξ ολοκλήρου στη εκπαίδευση του μοντέλου. Με άλλα λόγια, η αρχική προσέγγιση θα έδινε τη δυνατότητα υλοποίησης μίας εφαρμογής με τόσο καλά αποτελέσματα όσο και η εκπαίδευση του μοντέλου. Αντίθετα «παντρεύοντας», την εκπαίδευση ενός πιο απλού μοντέλου και τις βιβλιοθήκες της Python για τις κινήσεις το αποτέλεσμα είναι πολύ πιο ικανοποιητικό και η υλοποίηση της εφαρμογής μπορεί να κάνει καλύτερη κλιμάκωση (scale-up).

Αξίζει να σημειωθεί πως η επιλογή των εικόνων για την εκμάθηση του μοντέλου αποτελεί πολύ σημαντικό παράγοντα για την αποτελεσματικότητά του. Για τη συγκεκριμένη έπρεπε να φωτογραφηθούν οι κατόψεις όλων των πιονιών και σε λευκά και σε μαύρα τετράγωνα, καθώς επίσης και κενά κελιά. Έτσι θα μπορεί η εφαρμογή να εντοπίζει τι χρώματος πιόνι βρίσκεται σε κάθε κελί. Όπως αναφέρθηκε και στην παράγραφο της Οπτικής Αναγνώρισης, ο εντοπισμός της σκακιέρας γίνεται με τον υπολογισμό των pixel που καταλαμβάνει η σκακιέρα στο στιγμιότυπο που δημιουργεί η κάμερα, όπου είναι 400x400 pixels. Αυτό σημαίνει ότι κάθε κελί της σκακιέρας καταλαμβάνει 50x50 pixel (400 pixel / 8 κελιά) στο στιγμιότυπο, επομένως για την αποτελεσματικότερη εκμάθηση του μοντέλου θα πρέπει να χρησιμοποιηθούν εικόνες τέτοιων διαστάσεων.

Ένας άλλος σημαντικός παράγοντας είναι και η ποιότητα των εικόνων. Αν και οι διαστάσεις των κελιών δεν επιτρέπουν την χρήση πολύ «καθαρών» εικόνων (μιας και οι διαστάσεις είναι μόνο 50x50), η ανάλυση της κάμερας έπαιξε καθοριστικό ρόλο.

480p WebCam	720p WebCam (HD)
	

Στον πιο πάνω πίνακα παρατηρούμε τη διαφορά της ποιότητας των δειγμάτων με δυο τη χρήση δυο διαφορετικών καμερών. Όπως παρατηρείται, είναι πρακτικά αδύνατο ακόμα και για το ανθρώπινο μάτι να καταλάβει εάν η κάτω αριστερή εικόνα είναι ένα μαύρο κελί ή ένα μαύρο πιόνι σε μαύρο κελί.

Δημιουργία Dataset

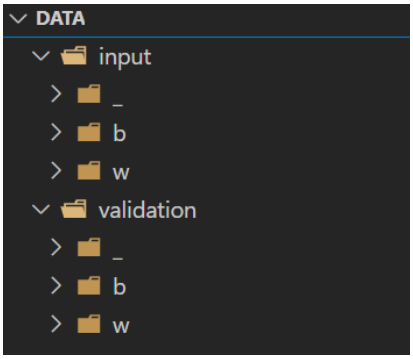
Για την δημιουργία του dataset υλοποιήθηκε ένα εργαλείο το οποίο φωτογραφίζει τη σκακιέρα και κόβει τα κελιά της, αποθηκεύοντας ξεχωριστά κάθε πιόνι.

Η διαδικασία έχει ως εξής:

1. Τα πιόνια τοποθετούνται στην αρχική θέση της παρτίδας
2. Παραμετροποιούμε το εργαλείο ορίζοντας εάν το dataset που θα παραχθεί, είναι dataset εκπαίδευσης ή επαλήθευσης. Για την σωστή εκμάθηση, πρέπει να δημιουργηθούν και τα δύο, με το dataset της εκπαίδευσης να είναι 2-3 φορές μεγαλύτερο από εκείνο της επαλήθευσης.
3. Το εργαλείο εκτελεί την διαδικασία οπτικής αναγνώρισης και χωρίζει τη σκακιέρα σε 64 κομμάτια τα οποία τα τοποθετεί στους αντίστοιχους φακέλους με τα ονόματα των κλάσεων όπου ανήκουν (λευκά, μαύρα, κενά).

Αξίζει να σημειωθεί πως κατά την δημιουργία των dataset θα ήταν καλό τα πιόνια να μετακινούνται λίγα χιλιοστά εντός των κελιών τους, έτσι ώστε να δημιουργούνται διαφορετικές εικόνες, με διαφορετικές σκιάσεις και γωνίες πιονιών. Αυτό θα έχει σαν αποτέλεσμα την πιο αποτελεσματική εκμάθηση του μοντέλου.

Μόλις ολοκληρωθεί η διαδικασία, οι εικόνες αποθηκεύονται στους αντίστοιχους φακέλους που αποτελούν τις κλάσεις του μοντέλου. Επιπλέον, οι εικόνες αποθηκεύονται μέσα σε έναν φάκελο ο οποίος φανερώνει εάν το σύνολο των εικόνων χρησιμοποιείται ως είσοδος στον αλγόριθμο (input) ή σαν επαλήθευση (validation) κατά τη διαδικασία της εκμάθησης.



Διαδικασία Εκμάθησης

Εφόσον δημιουργήσουμε τις εικόνες του dataset, στη συνέχεια θα πρέπει να οργανωθούν στις κλάσεις που θέλουμε να κατηγοριοποιηθούν. Για τον λόγο αυτό πρέπει να δημιουργηθούν 3 φάκελοι με ονόματα w, b και _. Κάθε φάκελος αντιπροσωπεύει την κάθε κλάση (w: λευκά, b: μαύρα και _: κενά). Έπειτα, τοποθετούμε την κάθε εικόνα στον αντίστοιχο φάκελο, βάση του χρώματος των πιονιών.

Κάθε εικόνα του dataset εμπεριέχει σαν πληροφορία τις διαστάσεις της καθώς και τη τιμή του χρώματος κάθε pixel που την απαρτίζει. Έτσι καταλήγουμε η κάθε εικόνα να είναι ουσιαστικά ένας τρισδιάστατος πίνακας 50x50x3. Αυτή η πληροφορία είναι πολύ περίπλοκη και δαπανηρή, με αποτέλεσμα να επιβαρύνεται σημαντικά η διαδικασία της δημιουργίας του μοντέλου εάν διαβάζουμε την κάθε εικόνα ατόφια από την μνήμη.

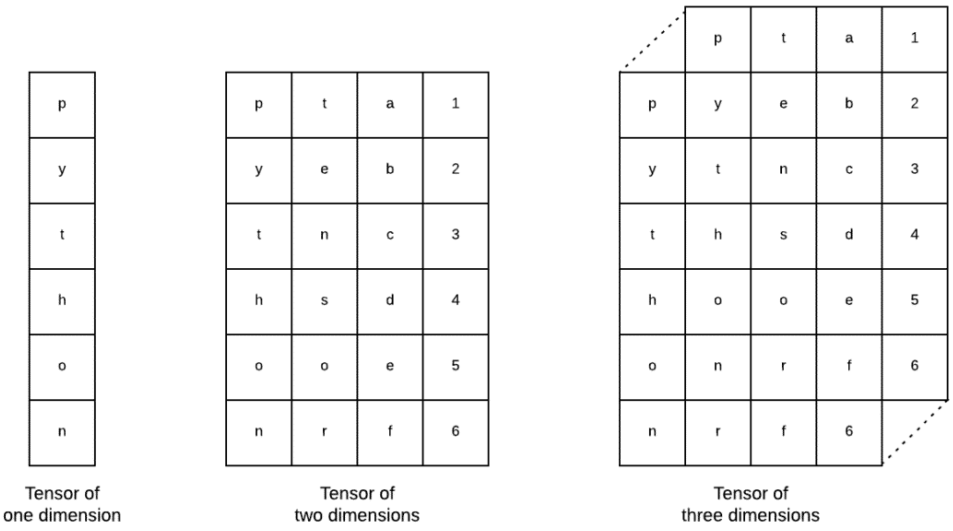
Για τον λόγο αυτό οι εικόνες κατά την εκπαίδευση περνούν από ένα φίλτρο το οποίο τις μετατρέπει σε ασπρόμαυρες. Έτσι η κάθε εικόνα πλέον ένας πίνακας 50x50 με την τιμή της «έντασης» του γκρι για κάθε pixel. Επιπλέον για ακόμα μεγαλύτερη απόδοση, μετατρέπουμε τον δισδιάστατο πίνακα σε μονοδιάστατο, 2.500 θέσεων (50x50=2.500), με κάθε θέση να εμπεριέχει την απόχρωση κάθε pixel και μεταφέρουμε όλη αυτή τη πληροφορία σε ένα csv, προσθέτοντας και την κλάση όπου ανήκει η κάθε εικόνα.

Έτσι καταλήγουμε με έναν πίνακα, όπου κάθε γραμμή αντιπροσωπεύει μία εικόνα, ενώ οι στήλες, τις αποχρώσεις του γκρι σε κάθε pixel, με εξαίρεση την πρώτη θέση των στηλών να δείχνει την κλάση που ανήκει το πιόνι που απεικονίζεται.

	0	1	2	3	…	2497	2498	2499	2500
0	1	0	1	0		194	196	192	180
1	1	0	1	0		196	193	191	185
2	1	0	2	0		204	203	201	198
3	1	1	1	0		80	80	83	98
4	1	117	116	112		190	190	186	157

Στο σημείο αυτό μπορούμε να τροφοδοτήσουμε τα δεδομένα σε ένα μοντέλο για εκπαίδευση. Για τη συγκεκριμένη περίπτωση θα εκπαιδεύσουμε ένα Διαδοχικό Μοντέλο (Sequential Model). Ένα διαδοχικό μοντέλο αποτελεί ουσιαστικά, μία στοίβα από επίπεδα (layers) όπου το κάθε επίπεδο έχει έναν τανυστή (tensor) ως είσοδο και έναν ως έξοδο. Το σύνολο των επιπέδων αυτών αποτελεί στο τέλος και το νευρωνικό δίκτυο. Επιπλέον κάθε επίπεδο του μοντέλου ευθύνεται για μια συγκεκριμένη διαδικασία, όπου το αποτέλεσμα της περνάει ως είσοδος στο επόμενο επίπεδο.

Ένας τανυστής αποτελεί ουσιαστικά ένα γενικευμένο διάνυσμα τιμών[4]. Ένας τανυστής μπορεί να απεικονιστεί σαν μία πολυδιάστατη διάταξη αριθμητικών τιμών, με την τάξη (ή βαθμό) ενός τανυστή είναι οι διαστάσεις της διάταξης που χρειάζεται για να τον απεικονίσει ή ισοδύναμα, ο αριθμός των δεικτών που χρειάζονται για να ονοματιστεί και να διαχωριστεί ένα στοιχείο αυτής της διάταξης. Για παράδειγμα, ένας γραμμικός μετασχηματισμός μπορεί να απεικονιστεί από ένα πίνακα (matrix) δηλαδή μία δισδιάστατη διάταξη και επομένως είναι τανυστής 2ης τάξης. Ένα διάνυσμα μπορεί να απεικονιστεί σαν μία μονοδιάστατη διάταξη (μονοδιάστατος πίνακας) και είναι τανυστής 1ης τάξης. Αντίστοιχα, οι τανυστές μηδενικής τάξης απεικονίζουν κανονικούς αριθμούς.

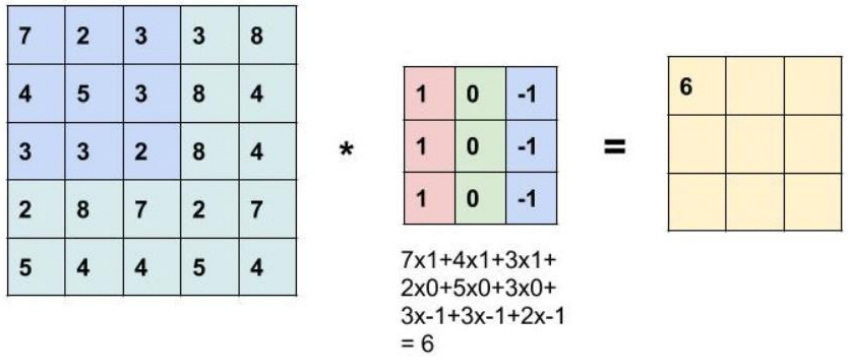


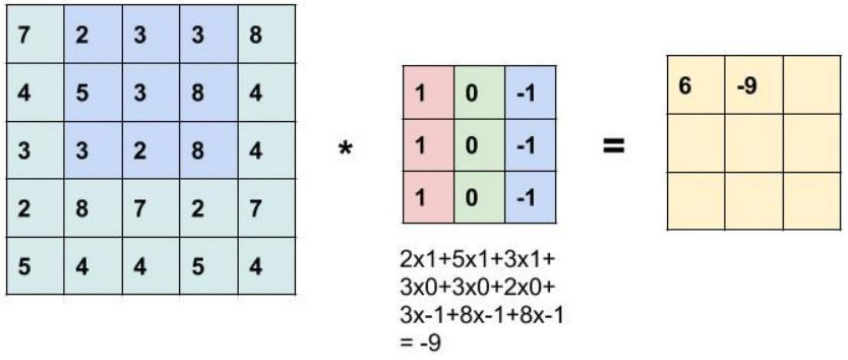
Προγραμματιστικά μπορούμε να διαχειριστούμε τους τανυστές ως πολυδιάστατους πίνακες. Αυτό στην Python επιτυγχάνεται με τη χρήση της βιβλιοθήκης NumPy η οποία εμπεριέχει σύνηθες συναρτήσεις και τελεστές πολυδιάστατων πινάκων και διανυσμάτων. Επιπλέον σε συνδυασμό με τις συναρτήσεις της TensorFlow και της Keras, μπορούμε να τροφοδοτήσουμε τα επίπεδα του μοντέλου με δεδομένα αλλά και να τα επεξεργαστούμε προκειμένου να αυξηθεί η απόδοση της εκπαίδευσης του. Συγκεκριμένα πρέπει, πρώτού εκτελέσουμε τη διαδικασία εκγύμνασης, να τροποποιήσουμε τα δεδομένα των εικόνων που εισάγουμε στο μοντέλο. Κάθε επίπεδο, όπως αναφέρθηκε, θα είναι υπεύθυνο για μία λειτουργεία τροποποίησης των δεδομένων και θα εισάγει το αποτέλεσμα στο επόμενο επίπεδο. Στο τελευταίο στρώμα, θα εκτελείται η εκπαίδευση του μοντέλου.

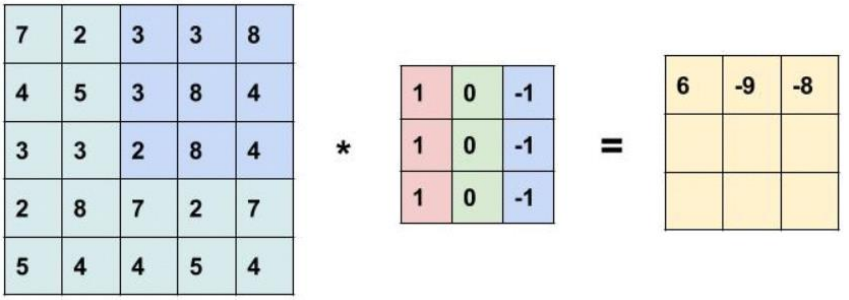
[4] Morris Kline (1972). “Mathematical Thought From Ancient To Modern Times”

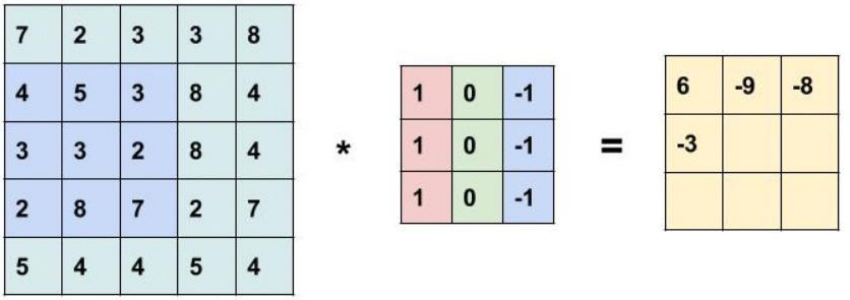
Δισδιάστατη Συνέλιξη

Το πρώτο στρώμα υλοποιεί την διαδικασία της δισδιάστατης συνέλιξης (2D Convolution). Συνοπτικά, ένα επίπεδο συνέλιξης «σαρώνει» μία εικόνα με τη χρήση ενός «φίλτρου», ο οποίος κατά το Keras ονομάζεται πυρήνας (kernel) με στόχο την εξόρυξη «στοιχείων» (features) της εικόνας που ίσως φανούν σημαντικά κατά την κατηγοριοποίηση. Ο πυρήνας μπορεί να διαθέτει επίσης και βάρη, (weights) τα οποία μπορούν να ρυθμιστούν για να επιτευχθεί καλύτερη πρόβλεψη από το μοντέλο.









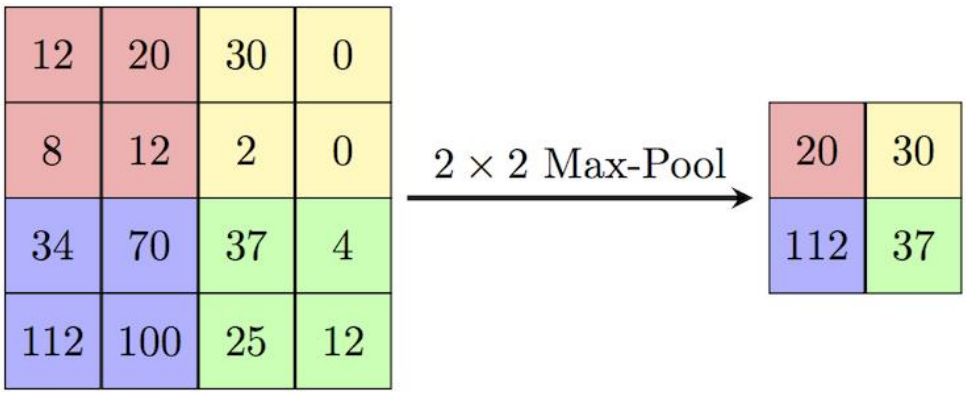
Στη συγκεκριμένη περίπτωση, ως είσοδο έχουμε μία εικόνα μεγέθους 50x50 που αποτελεί το πιόνι και χρησιμοποιείται ένας πυρήνας 3x3. Ο πυρήνας αυτός δημιουργείται λαμβάνοντας δείγματα από μια περικομμένη κανονική κατανομή με επίκεντρο το 0 και διασπορά: $\sqrt{\frac{2}{n}}$, όπου n είναι ο αριθμός μονάδων εισόδου στον τανυστή. Ο πυρήνας με αυτή την ιδιότητα ονομάζεται «κανονικός» (normal kernel) και κατασκευάζεται από την Keras κατά τον ορισμό του επιπέδου. Τέλος, ως συνάρτηση ενεργοποίησης του επιπέδου ορίζεται η γραμμική συνάρτηση (rectified linear-activation function, re.l.u.).

```
Conv2D(64, kernel_size=(3, 3),
                activation='relu',
                kernel_initializer='he_normal',
                input_shape=input_shape)
```

Max Pooling

Η «συγκέντρωση» (Pooling), είναι ένα σύνηθες χαρακτηριστικό των αρχιτεκτονικών των συνελικτικών νευρωνικών δικτύων (Convolutional Neural Network, CNN). Η βασική ιδέα πίσω από ένα στρώμα συγκέντρωσης είναι να «συσσωρεύονται» τα δεδομένα που δημιουργούνται από ένα επίπεδο συνέλιξης. Τυπικά, η λειτουργία του είναι να μειώσει σταδιακά το μέγεθος της εικόνας που εφαρμόζεται η συνέλιξη, για να μειώσει την ποσότητα παραμέτρων και υπολογισμού που χρειάζεται για την εκγύμναση του δικτύου. Η πιο κοινή μορφή συγκέντρωσης είναι η μέγιστη συγκέντρωση (max pooling).

Στο δεύτερο επίπεδο λοιπόν, «υποβαθμίζουμε» (downgrade) ή συγκεντρώνουμε, τα αποτελέσματα της εξόδου του πρώτου στρώματος, κρατώντας μόνο τη μέγιστη συγκέντρωση τιμών από το προηγούμενο βήμα.



Εφόσον στο πρώτο επίπεδο συγκλίναμε τα δεδομένα με την Conv2D, ορισμένες τιμές μεγιστοποιήθηκαν και άλλες ελαχιστοποιήθηκαν. Η μέγιστη συγκέντρωση στο σημείο αυτό, συμπυκνώνει τα νέα δεδομένα έπειτα από την συνέλιξη, κάνοντας έτσι τα δυνατά «χαρακτηριστικά» της εικόνας να εμφανιστούν, το οποίο βοηθά στην καλύτερη

κατηγοριοποίηση των εικόνων. Επιπλέον, το επίπεδο συγκέντρωσης, όπως και το Dropout, βοηθούν στο να αποτραπεί η υπερ-εκπαίδευση (over-fitting) του δικτύου.

Η διαδικασία της συγκέντρωσης υλοποιείται εφαρμόζοντας ένα φίλτρο το οποίο χωρίζει τα εισερχόμενα δεδομένα σε ίσου μεγέθους περιοχές, με την έξοδο του φίλτρου να είναι είτε η μέση τιμή των τιμών των δεδομένων της εκάστοτε περιοχής (average pooling), είτε η μέγιστη τιμή τους (max pooling).

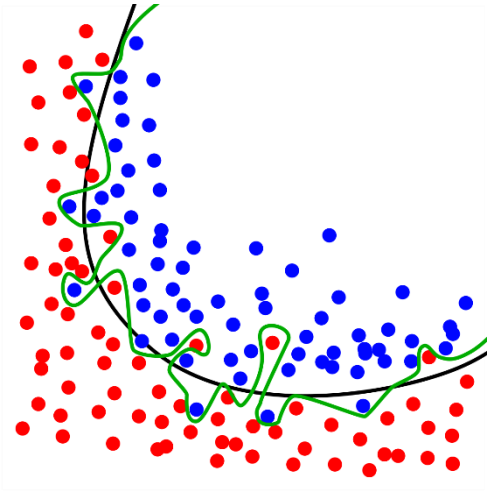
```
model.add(MaxPooling2D(pool_size))
```

Η Keras χρειάζεται μία παράμετρο pool_size, η οποία ορίζει τις διαστάσεις την περιοχής στην οποία θα υπολογιστή η μέγιστη τιμή.

Dropout

Στο επίπεδο αυτό, τυχαία επιλέγουμε να «κλείσουμε» κάποιους κόμβους του δικτύου, κάνοντας τους να παράξουν μηδενική έξοδο. Ο λόγος που γίνεται αυτό είναι για να μην περάσει το μοντέλο σε στάδιο over-fitting.

Η υπερ-εκπαίδευση (over-fitting) ενός μοντέλου είναι η δημιουργία ενός μοντέλου του οποίου η στατιστική ανάλυση αντιστοιχεί πολύ στενά ή και ακριβώς, σε ένα συγκεκριμένο σύνολο δεδομένων, με αποτέλεσμα το μοντέλο να μην μπορεί να εντάξει στην εκπαίδευση νέα δεδομένα ή διαφοροποιημένα από το αρχικό σύνολο, το οποίο έχει με τη σειρά του σαν συνέπεια το μοντέλο να μην είναι σε θέση να δώσει αξιόπιστες προβλέψεις.



Όπως φαίνεται και στην εικόνα, η πράσινη γραμμή αντιπροσωπεύει ένα υπερβολικό μοντέλο και η μαύρη γραμμή αντιπροσωπεύει ένα «κανονικό» μοντέλο.

Παρόλο που η πράσινη γραμμή ακολουθεί καλύτερα τα δεδομένα εκπαίδευσης, είναι άμεσα εξαρτημένη με από αυτά τα δεδομένα και είναι πιθανό να έχει υψηλότερο ποσοστό σφάλματος (loss-rate) σε νέα δεδομένα που δεν εμφανίζονται.

Ένας επιπλέον τρόπος αποφυγής του over-fitting είναι η χρήση μεγάλης ποικιλίας δεδομένων για την εκπαίδευση. Στην προκειμένη περίπτωση, το μοντέλο εκπαιδεύτηκε αναλύοντας εικόνες οι οποίες ήταν είχαν ληφθεί σε διαφορετικές συνθήκες φωτός και τα πιόνια στην κάθε φωτογραφία ήταν λίγο διαφορετικά τοποθετημένα. Έτσι το μοντέλο δεν «κλειδώνει» σε ένα συγκεκριμένο pattern της εισόδου για να το κάνει να εξαρτάται από τα δεδομένα που εισάγονται.

Η μέθοδος για τη δημιουργία ενός στρώματος Dropout στην Keras απαιτεί τουλάχιστον μία παράμετρο (rate) η οποία δείχνει τη συχνότητα με την οποία η Keras θα μηδενίζει την έξοδο τους όταν περνάει στο στρώμα αυτό.

```
model.add(Dropout(rate=0.4))
```

Flatten

Όπως αναφέρθηκε και προηγουμένως, μία εικόνα αναπαρίσταται ως ένας πολυδιάστατος πίνακας το οποίο περιπλέκει την ανάλυση και διαχείριση των δεδομένων και για τον λόγο αυτό μετατρέπουμε τις εικόνες σε μονοδιάστατους πίνακες. Για παράδειγμα μία μαύρη, μονόχρωμη εικόνα 4x4 pixel, χωρίς καμία μετατροπή αναπαρίσταται ως πίνακας της μορφής [[0,0,0,0], [0,0,0,0], [0,0,0,0], [0,0,0,0]]. Μετά την μετατροπή αναπαρίσταται ως μονοδιάστατος πίνακας της μορφής: [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]. Η μετατροπή αυτή ονομάζεται ισοπέδωση (flatten).

Η διαδικασία αυτή εκτελείται σε αυτό το επίπεδο του νευρωνικού δικτύου και με τη χρήση της Keras υλοποιείται τόσο εύκολα όσο το να γραφτεί η εντολή:

```
model.add(Flatten())
```

Dense

Στο τελικό επίπεδο του δικτύου απλά εφαρμόζουμε τανυστές εξόδου. Αυτό είναι απλώς ένα τυπικό επίπεδο όπου κάθε κόμβος στην είσοδο συνδέεται με κάθε κόμβο στην έξοδο. Ο λόγος για τον οποίο επιτυγχάνουμε ισοπέδωση είναι επειδή αυτό είναι το πραγματικό στρώμα ταξινόμησης που αναμένει ένα μονοδιάστατο πίνακα δεδομένων και επιπλέον στο ορίζει και τη συνάρτηση ενεργοποίησης των κόμβων.

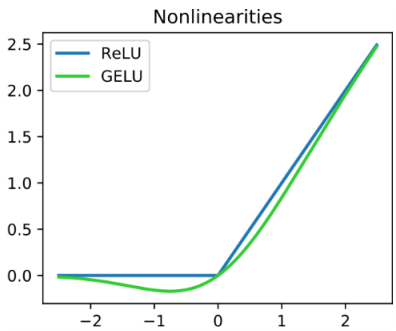
Η Keras παρέχει διάφορες συναρτήσεις ενεργοποίησης, δύο εκ των οποίων είναι η softmax και relu (γραμμική συνάρτηση) οι οποίες χρησιμοποιήθηκαν και για την εκπαίδευση του μοντέλου της εφαρμογής.

Η γραμμική συνάρτηση είναι η πιο σύνηθες χρησιμοποιημένη συνάρτηση ενεργοποίησης νευρώνων και αναπαρίσταται από τη συνάρτηση:

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

$f(x) = 0$ εάν $x \leq 0$

$f(x) = x$ εάν $x > 0$



Η συνάρτηση έχει διάφορες μαθηματικές παραλλαγές οι οποίες εφαρμόζονται σε διαφορετικές περιπτώσεις, αναλόγως το αποτέλεσμα που θέλουμε να επιτευχθεί.

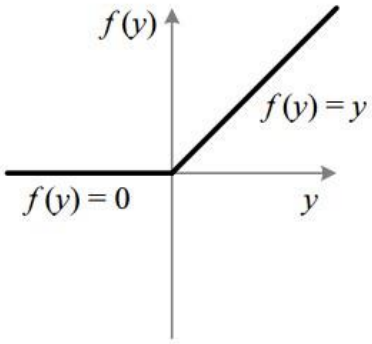
Με τη σειρά της η softmax, αποτελεί μια συνάρτηση κανονικοποίησης (normalization) η οποία χρησιμοποιείται από την Keras στο τελευταίο επίπεδο του δικτύου. Με άλλα λόγια, η συνάρτηση λαμβάνει ως είσοδο ένα διάνυσμα z πραγματικών αριθμών K και το ομαλοποιεί σε κατανομή πιθανότητας που αποτελείται από K πιθανότητες, ανάλογες με τα εκθετικά των αριθμών εισόδου.

Μελλοντικές Ενέργειες

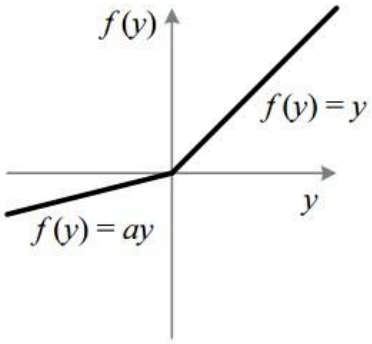
Dying ReLu

Υπάρχουν περιπτώσεις στις οποίες οι νευρώνες οι οποίοι χρησιμοποιούν την Γραμμική Συνάρτηση ενεργοποίησης, μερικές φορές να ωθούνται σε καταστάσεις, στις οποίες καθίστανται ανενεργοί για ουσιαστικά όλες τις εισόδους. Σε αυτή την κατάσταση, καμία κλίση δεν ρέει προς τα πίσω μέσω του νευρώνα, και έτσι ο νευρώνας κολλάει σε μια διαρκώς ανενεργή κατάσταση και «πεθαίνει».

Αυτή είναι μια μορφή του προβλήματος ονομάζεται Νεκρή Γραμμική Συνάρτηση (Dying ReLu) και σε ορισμένες περιπτώσεις, μεγάλος αριθμός νευρώνων σε ένα δίκτυο μπορεί να κολλήσει σε νεκρές καταστάσεις, μειώνοντας ουσιαστικά την χωρητικότητα του μοντέλου. Αυτό το πρόβλημα προκύπτει συνήθως όταν ο ρυθμός εκμάθησης είναι πολύ υψηλός. Το πρόβλημα αυτό μπορεί να μετριαστεί εκχωρώντας μια μικρή θετική κλίση για τιμές $x < 0$ στην είσοδο του νευρώνα.



Απλή Γραμμική Συνάρτηση (ReLu)



Γραμμική Συνάρτηση με μικρή κλίση (“Leaky” ReLu)

Πρόβλημα Σκιών

Όπως αναφέρθηκε και σε προηγούμενη παράγραφο, ένα σημαντικό πρόβλημα στην αναγνώριση της σκακιέρας είναι οι σκιές που δημιουργούνται από τα πιόνια αλλά και από τα αντικείμενα στο χώρο γύρω από τη σκακιέρα. Η πτώση σκιών επάνω στην σκακιέρα μπορεί να αλλοιώσουν τα χρώματα των καρέ που χρησιμοποιεί η εφαρμογή για την πρόβλεψη του χρώματος του κάθε πιονιού, με αποτέλεσμα να καταλήξει στο να κατηγοριοποιήσει λάθος το πιόνι.

Ένας τρόπος που θα μπορούσε να αποτραπεί αυτό, στα πλαίσια βελτίωσης της εφαρμογής είναι στο να ελέγχεται το ποσοστό πρόβλεψης του πιονιού και με βάση την προηγούμενη κατάσταση της παρτίδες, στο προηγούμενο καρέ, να επιβεβαιώνει την αντίστοιχη κλάση ώστε να κατηγοριοποιηθεί το πιόνι.

Για παράδειγμα:

Έστω ότι ελέγχουμε το κελί A5 για το καρέ F, τα ποσοστά πρόβλεψης για κάθε κλάση, δεδομένου του καρέ είναι:

Λευκό (W)	Μαύρο (B)	Κενό (_)
44%	42%	16%

Θεωρούμε ωστόσο, ότι στο καρέ F-1, η κατάσταση της σκακιέρας (μέσω της βιβλιοθήκης PythonChess, δείχνει ότι στο κελί A5 βρισκόταν ένα μαύρο πιόνι και η θέση του δεν απειλείται από κανένα άλλο πιόνι. Έτσι λοιπόν, χρησιμοποιώντας τις πληροφορίες αυτές, θα μπορούσε η εφαρμογή να επαληθεύσει το ποσοστό πρόβλεψης και να επιλέξει την κλάση (B) ως σωστή πρόβλεψη συνδυάζοντας τα δεδομένα και όχι απλά χρησιμοποιώντας το εκπαιδευμένο μοντέλο ως μοναδική πηγή αλήθειας για την πρόβλεψη.

Η μέθοδος αυτή μπορεί να μειώσει τα προβλήματα που παρουσιάζονται από την επικάλυψη της σκακιέρας από σκιές καθώς επίσης και προβλήματα που εμφανίζονται από την μέτρια εκπαίδευση του μοντέλου, εφόσον, η τελική επιλογή για την πρόβλεψη δεν αποδίδεται εξ' ολοκλήρου από το μοντέλο αλλά και από την κατάσταση της παρτίδας.

Βιβλιογραφία

Dana H. Ballard, Christopher M. Brown (1982). "Computer Vision"

Βλαχαβάς Ιωάννης, Κεφάλας Πέτρος, Βασιλειάδης Νικόλας, Κόκκορας Φώτης, Σακελλαρίου Ηλίας (2006). "Τεχνητή Νοημοσύνη, Γ' Έκδοση"

Morris Kline (1972). "Mathematical Thought From Ancient To Modern Times"