Διεθνές Πανεπιστήμιο της Ελλάδος

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

Πτυχιακή Εργασία

Εφαρμογή συνελικτικών νευρωνικών δικτύων βαθιάς μηχανικής μάθησης για κατηγοριοποίηση εικόνων: Μελέτη περίπτωσης σε ζωγραφιές παιδιών από νηπιαγωγεία

Ονοματεπώνυμο Φοιτητή: Βαλιάκος Απόστολος

ΑΜ:4482

Υπεύθυνος καθηγητής: Δρ. Τσιμπίρης Αλκιβιάδης

Ημερομηνία: 22/03/2021

Περίληψη

Στην παρούσα εργασία εφαρμόσαμε συνελικτικά νευρωνικά δίκτυα βαθιάς μηχανικής μάθησης για κατηγοριοποίηση εικόνων σε ζωγραφιές παιδιών από νηπιαγωγεία με σκοπό την κατηγοριοποίηση των εικόνων σε 6 κλάσεις. Η πρώτη είναι η χαμηλότερη ικανότητα λεπτής κινητικότητας ενώ 6 η βέλτιστη. Τα διαφορετικά νευρωνικά δίκτυα που δοκιμάστηκαν ακολουθούν διαφορετικές αρχιτεκτονικές Συνελικτικών Νευρωνικών Δικτύων (Convolutional Neural Networks - CNN). Το Σύνολο δεδομένων αυτό αποτελείται από ζωγραφιές παιδιών που αναπαριστούν τη φιγούρα ενός άνδρα και μίας γυναίκας. Σκοπός μας είναι να μελετήσουμε την απόδοση των διαφορετικών μοντέλων κάτω από τις ίδιες συνθήκες στο συγκεκριμένο σύνολο δεδομένων. Θα εξηγήσουμε τους τρόπους με τους οποίους υπολογίζουμε την ακρίβεια του μοντέλου, τι είναι η ακρίβεια και οι άλλες μετρικές. Θα δούμε πως κάναμε την έρευνα προκειμένου να φτάσουμε στην αρχιτεκτονική του μοντέλου μας (MotorSkilllsCNN) έτσι ώστε να έχει την καλύτερη απόδοση για το συγκεκριμένο σύνολο δεδομένων. Μετά τη διαδικασία της εκπαίδευσης είδαμε ότι το πιο αποδοτικό μοντέλο είναι το InceptionV3 της Google με ακρίβεια 100% κατά τη διάρκεια εκπαίδευσης και 50% ακρίβεια κατά τη διάρκεια επαλήθευσης. Στην δεύτερη θέση βρίσκεται το δικής μας κατασκευής, MotorSkillsCNN με 95% ακρίβεια κατά την εκπαίδευση και 40% ακρίβεια κατά την επαλήθευση. Έπειτα ακολουθεί το ResNetV2 με 95% ακρίβεια κατά τη διάρκεια εκπαίδευσης και 30% κατά τη διάρκεια επαλήθευσης και στην τέταρτη θέση έχουμε το MobileNetV2 με 92% ακρίβεια κατά τη διάρκεια εκπαίδευσης και 20% ακρίβεια κατά τη διάρκεια επαλήθευσης .

Abstract

In this thesis we applied convolutional neural networks to classify images of kindergarten kids' drawings in 6 classes depending on their motor skills. The first class is the weaker skill and the sixth is the best. The different architectures of Convolutional Neural Networks (CNN) have different accuracies and results. This dataset represents the drawing figure of a man and a woman from children at a

certain age. Our aim is to study the accuracy of these different models under the same circumstances in the specific dataset. We will explain the ways that we calculated the accuracy and the other metrics and what these metrics are. We will also see how we studied to reach to the architecture of our model (MotorSkilllsCNN) in order to have the best results for this specific dataset. After the training phase we reach to the conclusion that the InceptionV3 model made by Google gave us the best results with 100% of training accuracy and 50% of evaluation accuracy. The following one was the MotorSkillsCNN with 95% of training accuracy and 40% of evaluation accuracy. In the third place we have ResNetV2 with 95% training accuracy and 30% evaluation accuracy and in the fourth place we have MobileNetV2 with 92% training accuracy and 20% of evaluation accuracy.

Περιεχόμενα

Πίνακας Εικόνων

Πίνακας Γραφημάτων

Αναφορές πινάκων

Εισαγωγή και σχετικές Έρευνες

Η λεπτή κινητικότητα είναι η ικανότητα συντονισμού, μυών, νεύρων και κόκκαλων με σκοπό να επιτύχουμε συγκεκριμένες κινήσεις. Υπάρχουν αρκετοί διαφορετικοί τύποι λεπτής κινητικότητας τους οποίους μπορεί να αναπτύξει ένα παιδί, φυσικά υπάρχει εξάρτηση σε αυτό από την δομή που έχουν τα κόκκαλα. Παρόλα αυτά η λεπτή κινητικότητα αναπτύσσεται όσο μεγαλώνουν τα παιδιά. Μαθαίνουν να συντονίζουν χέρια, μάτια κλπ. Αναπτύσσουν μυϊκή μνήμη, και ενδυνάμωση μυών. Μαθαίνουν να προσέχουν το περιβάλλον στο οποίο βρίσκονται και να αναπτύσσουν φυσιολογικές αισθήσεις. [1] Η λεπτή κινητικότητα μπορεί να εμφανιστεί και σε μεγαλύτερες ηλικίες με διαφορετικές μορφές (νόσος Parkinson).[2] Επίσης μπορούμε να διακρίνουμε διαφορές ακόμα και στα φύλα των παιδιών και το ποσοστό λεπτής κινητικότητας που αναπτύσσονται. Πιο συγκεκριμένα τα αγόρια αναπτύσσουν δεξιότητες με χρήση μπάλας νωρίτερα από τα κορίτσια τα οποία αποκτούν επιδεξιότητα στις κινήσεις των χεριών νωρίτερα σε σχέση με τα αγόρια.[3]

Η τεχνολογία και συγκεκριμένα η μηχανική μάθηση έχει ποικίλες εφαρμογές στην μοντελοποίηση των ανθρώπινων κινήσεων αλλά υπάρχει περιορισμένος αριθμός ερευνών στην λεπτή κινητικότητα. Οι συγγραφείς της έρευνας [4] έχουν παρουσιάσει τον τρόπο με τον οποίο αλληλοεπιδρούν διαφορετικές προσεγγίσεις μηχανικής μάθησης. Στην παρούσα πτυχιακή εργασία θα δούμε πως διαφορετικές αρχιτεκτονικές Convolutional Neural Networks (CNN) αλληλοεπιδρούν με το συγκεκριμένο σύνολο δεδομένων. Εκτός αυτών υπάρχει και η έρευνα των Peters, Jan. (2007) [5] στην οποία μελετάται ο τρόπος με τον οποίο μπορούμε να μεταφέρουμε την λεπτή κινητικότητα στη ρομποτική.

Κεφάλαιο 1. Ιστορικό υπόβαθρο τεχνητής νοημοσύνης

Η τεχνητή νοημοσύνη (AI από το Artificial Intelligence) καθιστά τις μηχανές ικανές να μαθαίνουν από την εμπειρία, να προσαρμόζονται σε νέα εισαγόμενα δεδομένα και να εκτελούν ανθρωπομορφικά έργα. Τα περισσότερα παραδείγματα AI τα οποία ακούγονται σήμερα –από τους υπολογιστές που παίζουν σκάκι έως τα αυτο-οδηγούμενα αυτοκίνητα– βασίζονται σε μεγάλο βαθμό στο deep learning και την επεξεργασία φυσικής γλώσσας (ΕΦΓ). Με τη χρήση των τεχνολογιών αυτών, οι

υπολογιστές μπορούν να εκπαιδευτούν ώστε να επιτελούν συγκεκριμένα καθήκοντα με επεξεργασία μεγάλων ποσοτήτων δεδομένων και αναγνώριση μορφών η μοτίβων στα δεδομένα.

1.1 Η ιστορία της Τεχνητής Νοημοσύνης (AI)

Ο όρος της τεχνητής νοημοσύνης (AI) επινοήθηκε το 1956, αλλά το AI έχει γίνει πιο δημοφιλές σήμερα λόγω του αυξημένου όγκου δεδομένων, των προηγμένων αλγορίθμων και των βελτιώσεων στην ισχύ των υπολογιστών και την αποθήκευση των δεδομένων.

Αρχικά, η έρευνα γύρω από το AI επικεντρώθηκε σε θέματα όπως η επίλυση προβλημάτων και οι συμβολικές μέθοδοι. Τη δεκαετία του ’60, το Υπουργείο Άμυνας των ΗΠΑ ενδιαφέρθηκε για αυτόν τον τύπο εργασίας και ξεκίνησε την εκπαίδευση των υπολογιστών στη μίμηση της βασικής ανθρώπινης συλλογιστικής. Για παράδειγμα, η Υπηρεσία Προηγμένων Ερευνητικών Προγραμμάτων Άμυνας (DARPA) ολοκλήρωσε τα προγράμματα χαρτογράφησης δρόμων τη δεκαετία του ’70. Επίσης, η DARPA παρήγαγε ευφυείς προσωπικούς βοηθούς το 2003, πολύ πριν η Siri, Alexa και Cortana γίνουν πασίγνωστες.

Αυτή η πρώτη εργασία, προετοίμασε το έδαφος για την αυτοματοποίηση και την τυπική συλλογιστική που βλέπουμε στους υπολογιστές σήμερα, συμπεριλαμβανομένων των συστημάτων υποστήριξης λήψης αποφάσεων και των έξυπνων συστημάτων αναζήτησης που μπορούν να σχεδιαστούν ώστε να συμπληρώνουν και να βελτιώνουν τις ανθρώπινες ικανότητες.

Ενώ οι ταινίες του Χόλιγουντ και τα μυθιστορήματα επιστημονικής φαντασίας απεικονίζουν το AI ως ανθρωπόμορφα ρομπότ που καταλαμβάνουν τον κόσμο, η τρέχουσα εξέλιξη του AI δεν είναι τόσο τρομακτική – ούτε τόσο έξυπνη. Αντίθετα, το AI έχει εξελιχθεί ώστε να παρέχει συγκεκριμένα οφέλη σε κάθε βιομηχανικό κλάδο.

1.2 Άλαν Μάθισον Τούρινγκ

Εικόνα 1. Άλαν Μάθισον Τούρινγκ πηγή https://el.wikipedia.org/wiki/%CE%86%CE%BB%CE%B1%CE%BD_%CE%A4%CE%BF%CF%8D%CF%81%CE%B9%CE%BD%CE%B3%CE%BA:

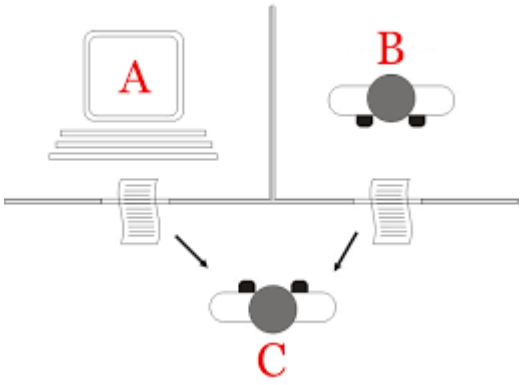
Ο Άλαν Μάθισον Τούρινγκ ή Τιούρινγκ (Alan Matheson Turing, 23 Ιουνίου 1912 – 7 Ιουνίου 1954) ήταν Άγγλος μαθηματικός, καθηγητής της λογικής, κρυπτογράφος και θεωρητικός βιολόγος. Θεωρείται ο «πατέρας της επιστήμης υπολογιστών», χάρη στην πολύ μεγάλη συνεισφορά του στο γνωστικό πεδίο της θεωρίας υπολογισμού κατά τη δεκαετία του 1930, αλλά και της τεχνητής νοημοσύνης, χάρη στο λεγόμενο τεστ Τούρινγκ, την οποία πρότεινε το 1950 έναν τρόπο για να διαπιστωθεί πειραματικά αν μία μηχανή έχει αυθεντικές γνωστικές ικανότητες και μπορεί να σκεφτεί.

Το έργο του από τη δεκαετία του '30 προσέδωσε στην ως τότε άτυπη έννοια του αλγορίθμου μία επίσημη, αυστηρή μαθηματική διατύπωση μέσω της λεγόμενης Μηχανής Τούρινγκ. Ακόμα, ο Τούρινγκ διατύπωσε από κοινού με τον Αλόνζο Τσερτς την περίφημη εικασία του, ευρέως αποδεκτή, σύμφωνα με την οποία οποιοδήποτε μαθηματικό μοντέλο υπολογισμού είναι είτε ισοδύναμο είτε υποδεέστερο της Καθολικής Μηχανής Τούρινγκ, επομένως αυτή περιγράφει τον ευρύτερο δυνατό υπολογιστή γενικού σκοπού: είναι θεωρητικά ικανή να υπολογίσει ό,τι είναι δυνατό να υπολογιστεί αλγοριθμικά.

Οι επιστημονικές συνεισφορές του Τούρινγκ κατά τη διάρκεια του Β' Παγκοσμίου Πολέμου δεν αναγνωρίστηκαν ποτέ δημόσια κατά τη διάρκεια της ζωής του επειδή η εργασία του ήταν απόρρητη. Στο Μπλέτσλεϊ Παρκ (Bletchley Park), κέντρο της Βρετανικής Υπηρεσίας Αντικατασκοπείας, ήταν το κεντρικό πρόσωπο στην αποκρυπτογράφηση των γερμανικών στρατιωτικών κωδικών, όντας ο προϊστάμενος της Ομάδας 8. Η ομάδα αυτή ήταν που επιφορτίστηκε με την αποκωδικοποίηση της γερμανικής κρυπτογραφικής συσκευής Enigma.

Μετά τον Πόλεμο, σχεδίασε έναν από τους πρώτους ηλεκτρονικούς προγραμματίσιμους ψηφιακούς υπολογιστές στο Εθνικό Φυσικό Εργαστήριο, όπως λεγόταν, και κατασκεύασε μια δεύτερη υπολογιστική μηχανή στο Πανεπιστήμιο του Μάντσεστερ. Ο Τούρινγκ πέθανε το 1954, 16 μέρες πριν τα 42α γενέθλιά του από δηλητηρίαση από κυάνιο. Έρευνα προσδιόρισε το θάνατό ως αυτοκτονία, αλλά είναι γνωστό ότι τα στοιχεία επίσης στηρίζουν την πιθανότητα τυχαίας δηλητηρίασης. Το Βραβείο Τούρινγκ, η ύψιστη επιστημονική διάκριση στον χώρο της πληροφορικής από το 1966 κι έπειτα, ονομάστηκε έτσι προς τιμήν του.

1.3 Τεστ Turing



Εικόνα 2. Τεστ Τούρινγκ πηγή: https://en.wikipedia.org/wiki/Turing_test

Το τεστ Turing είναι ένας τρόπος για να ανακαλύψουμε εάν μια μηχανή μπορεί να σκεφτεί. Εφευρέθηκε από τον Alan Turing. Το τεστ διεξάγεται με έναν άντρα, μια γυναίκα και έναν εξεταστή μέσα σε τρία διαφορετικά δωμάτια και κανένας δεν μπορεί να δει τον άλλο. Τα δωμάτια είναι με ηχομόνωση, αλλά κάθε άνθρωπος έχει και ένα βίντεο προκειμένου με αυτό τον τρόπο να μπορούν να επικοινωνήσουν.

10Ο σκοπός της δοκιμασίας αυτής είναι ο εξεταστής να καταφέρει να βρει ποιος άνθρωπος είναι ο άντρας και ποιος η γυναίκα με βάση τις ερωτήσεις προς αυτούς καθώς και τις απαντήσεις τους. Όμως ο άντρας και η γυναίκα δεν είναι υποχρεωμένοι να πουν την αλήθεια. Γνωρίζουν και οι δύο εκ των προτέρων ότι μπορούν να πουν ψέματα. Συγκεκριμένα ο άντρας ενθαρρύνεται να λέει ψέματα συχνά και σε όποια έκταση θέλει. Σκοπός του είναι να παραπλανήσει τον εξεταστή. Προφανώς αυτό κάνει τη δουλειά του εξεταστή δύσκολη και μπορεί μάλιστα να καταλήξει και σε ένα λάθος συμπέρασμα. Αλλά το τεστ δεν ολοκληρώνεται μέχρι να αποφασίσει ο εξεταστής ποιο δωμάτιο περιέχει τον άντρα και ποιο τη γυναίκα. Τι θα συμβεί όμως αν ο άντρας αντικατασταθεί από έναν έξυπνο

υπολογιστή που είναι προγραμματισμένος σαν εκείνον; Εάν η μηχανή είναι «ανόητη», τότε ο εξεταστής θα είναι σωστός ποιο συχνά. Αν όμως η μηχανή είναι «εξυπνότερη» από τον άντρα, τότε ο εξεταστής θα πρέπει να κάνει λάθος περισσότερες φορές(Gibilisco1994: 368,369) Μάλιστα ο Turing είχε προβλέψει πως μέχρι το 2000 θα είχε αναπτυχθεί τεχνητή νοημοσύνη που θα μπορούσε να ξεγελάσει το 30% των ερωτώντων, έπειτα από πέντε λεπτά συζήτησης. Το πρώτο πρόγραμμα που κατάφερε να περάσει το συγκεκριμένο τεστ θεωρείται πως είναι το ELIZA που έφτιαξε το 1976 ο Αμερικανός προγραμματιστής Τζόσεφ Βάιζενμπαουμ και το οποίο κατάφερε να πείσει τη γραμματέα του πως συνομιλούσε με εκείνον. Έκτοτε ακολούθησαν και άλλα προγράμματα τα οποία έδειξαν μεταξύ άλλων πως ο έλεγχος Turing παρόλο που μπορεί να αποδείξει πως οι άνθρωποι μπορούν να ξεγελαστούν από μηχανές δεν απαντά στα σύγχρονα ερωτήματα για την τεχνητή νοημοσύνη όπως τις συνέπειές της αλλά ούτε προβλέπει το πότε θα επιτευχθεί. Εάν και τα προγράμματα που προσπαθούν μέχρι σήμερα να περάσουν το συγκεκριμένο τεστ είναι αναμφίβολα αξιοθαύμαστα δεν επιδεικνύουν την ανθρώπινη ικανότητα της αλληλεπίδρασης με τον εξωτερικό κόσμο ενώ ένας προσεκτικός παρατηρητής με τις κατάλληλες ερωτήσεις θα καταφέρει να διαπιστώσει πως πρόκειται για μηχανές. Ανοικτό πάντως παραμένει το ερώτημα για το εάν και πότε θα καταφέρει η ανθρωπότητα να παραγάγει τεχνητή νοημοσύνη. Μία από τις πιο αληθοφανείς προβλέψεις, αυτή του στελέχους της Google Ρέι Κούρτσβαϊλ τοποθετεί την ανακάλυψη της τεχνητής νοημοσύνης το 2029, βασισμένη στο νόμο του Μουρ για την πρόοδο των ηλεκτρονικών διατάξεων. Αντίθετα ο Φρεντ Μπρουκς, από τους πρωτεργάτες της IBM, εξετάζει το ζήτημα από τη σκοπιά του λογισμικού, υποστηρίζοντας πως δεν είμαστε καν κοντά στο να μπορούμε να προσομοιώσουμε προγραμματιστικά τις 1014 συνδέσεις των νευρώνων του ανθρώπινου εγκεφάλου, υπολογίζοντας πως θα χρειαστούν ακόμη περίπου πέντε αιώνες(Naftemporiki.gr 2014).

1.4 Τύποι Τεχνητής Νοημοσύνης

Υπάρχουν 3 τύποι τεχνητής νοημοσύνης:

Περιορισμένη Τεχνητή Νοημοσύνη (Artificial narrow intelligence) (ANI), η οποία έχει περιορισμένο εύρος δυνατοτήτων

Γενική Τεχνητή Νοημοσύνη (Artificial general intelligence) (AGI), που είναι ισοδύναμη με τις ανθρώπινες δυνατότητες

Τεχνητή Υπερευφυία (Artificial superintelligence) (ASI), που είναι πιο ικανή από έναν άνθρωπο.

1.4.1 Περιορισμένη Τεχνητή Νοημοσύνη (Artificial Narrow Intelligence) (ANI) / Weak AI / Narrow AI

Το weak AI είναι ο μόνος τύπος τεχνητής νοημοσύνης τον οποίο έχουμε καταφέρει να πλησιάσουμε μέχρι τώρα. Το Narrow AI έχει ως σκοπό την επίτευξη μοναδικών στόχων όπως η αναγνώριση προσώπων, αυτόνομη οδήγηση αυτοκίνητων κλπ. Ενώ είναι πολύ αποδοτικό στο να φέρνει εις πέρας τον στόχο που του έχει ανατεθεί.

Ενώ αυτά τα μοντέλα φαίνονται εξαιρετικά έξυπνα, λειτουργούν υπό πολύ μικρό σύνολο περιορισμών. Αυτός είναι και ο λόγος για τον οποίο αναφερόμαστε στον συγκεκριμένο τύπο τεχνητής νοημοσύνης και ως αδύναμος (weak). Το Narrow AI δεν μιμείται την ανθρώπινη νοημοσύνη αλλά προσομοιάζει μερικώς την ανθρώπινη συμπεριφορά βασιζόμενη σε περιορισμένο αριθμό παραμέτρων.

Μερικά ενδεικτικά χαρακτηριστικά Narrow Intelligence είναι:

RankBrain / Google Search [14]

Φίλτρα για αναγνώριση spam Emails

Αναγνώριση εικόνας η προσώπου

Ανίχνευση ασθένειας και εργαλεία πρόληψης

Αυτό-οδηγούμενα αυτοκίνητα

1.4.2 Γενική Τεχνητή Νοημοσύνη (Artificial General Intelligence) (AGI) / Strong AI / Deep AI

Η Artificial general intelligence (AGI), η οποία αναφέρεται επίσης ως Deep AI είναι η ιδέα ενός μοντέλου με γενική νοημοσύνη η οποία μιμείται την ανθρώπινη νοημοσύνη η/και την ανθρώπινη συμπεριφορά με την ικανότητα να μαθαίνει και να

εφαρμόζει την νοημοσύνη για την επίλυση προβλημάτων. Το Deep AI μπορεί να σκεφτεί να καταλάβει και να χρησιμοποιήσει τη νοημοσύνη του για να λύσει ένα πρόβλημα. Επίσης, ο τρόπος με τον οποίο συμπεριφέρεται είναι δυσδιάκριτος από τον τρόπο με τον οποίο συμπεριφέρεται ένας άνθρωπος σε οποιαδήποτε συνθήκη.

Οι επιστήμονες που ασχολούνται με την τεχνητή νοημοσύνη δεν έχουν καταφέρει να πετύχουν κάτι τέτοιο ως σήμερα. Για να το πετύχουν αυτό θα πρέπει να κάνουν τους υπολογιστές να αποκτήσουν συνείδηση προγραμματίζοντας ένα πλήρες σετ γνωστικών ικανοτήτων. Με τον τρόπο αυτό θα οι υπολογιστές θα πάνε την εμπειρική μάθηση στο επόμενο επίπεδο καταφέρνοντας να εφαρμόσουν την αποκτηθείσα γνώση σε ένα μεγαλύτερο εύρος προβλημάτων και όχι απλά να βελτιώνουν την απόδοση τους σε μεμονωμένους στόχους.

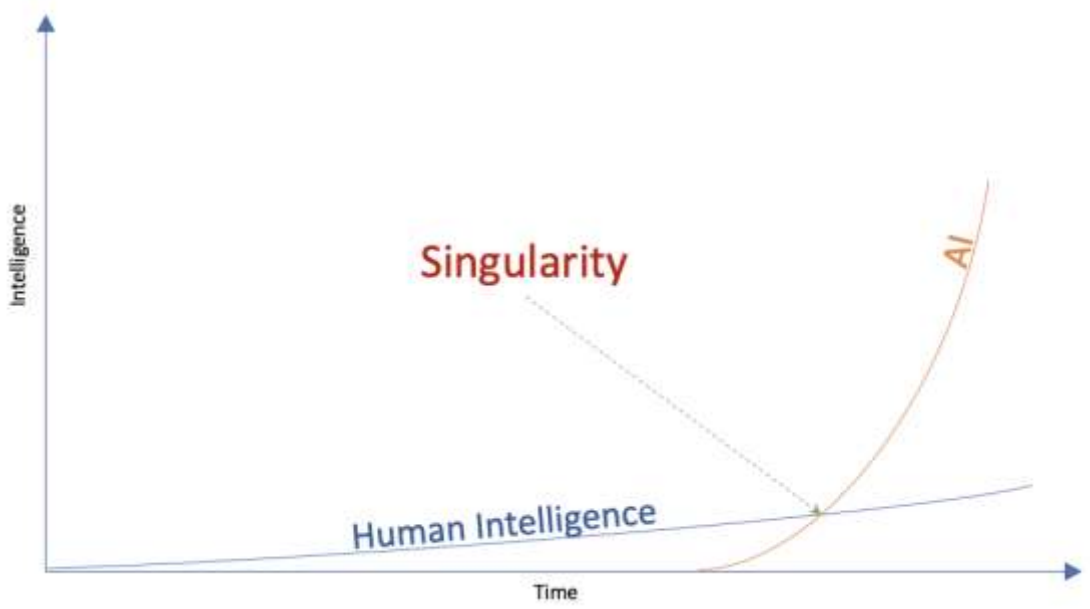
Ο Fujitsu-built K, ένας από τους πιο γρήγορους υπολογιστές του κόσμου είναι μια αξιοσημείωτη προσπάθεια επίτευξης Strong AI αλλά του πήρε 40 λεπτά να διεξάγει μια νευρωνική λειτουργία ενός δευτερολέπτου. Είναι δύσκολο να πούμε αν το Strong AI θα επιτευχθεί στο εγγύς μέλλον. Αλλά όσο βελτιώνεται η αναγνώριση εικόνας/ προσώπου είναι πολύ πιθανό να δούμε σημαντική βελτίωση στον τρόπο με τον οποίο οι μηχανές μαθαίνουν και βλέπουν.

1.4.3 Τεχνητή Υπερ-ευφυία (Artificial Superintelligence) (ASI)

Η τεχνητή υπερ-ευφυία (Artificial Superintelligence) είναι ένα υποθετικό AI το οποίο δεν μιμείται απλά η καταλαβαίνει την ανθρώπινη ευφυία και συμπεριφορά. Η τεχνητή υπερ-ευφυία είναι όταν οι υπολογιστές αποκτήσουν επίγνωση του περιβάλλοντος τους και ξεπεράσουν την ανθρώπινη ευφυία. [15]

1.5 Τεχνολογική Μοναδικότητα (Singularity)

Η τεχνολογική μοναδικότητα (Singualrity) είναι ένα υποθετικό σημείο στον χρόνο (εικόνα 3) στο οποίο η τεχνολογική ανάπτυξη γίνεται ανεξέλεγκτη και μη ανατρέψιμη, έχοντας ως αποτέλεσμα αλλαγές στον ανθρώπινο πολιτισμό που όμοιες τους δεν έχουμε συναντήσει. Σύμφωνα με την πιο διαδεδομένη εκδοχή της υπόθεσης της μοναδικότητας (έκρηξη ευφυίας), ένας αναβαθμίσιμος έξυπνος υπολογιστής ξαφνικά θα εισέλθει σε μία κατάσταση «runaway reaction» από κύκλους αυτό-αναβάθμισης όπου η κάθε μία θα το οδηγεί σε μία νέα γενιά υπολογιστών να εμφανίζεται όλο και πιο γρήγορα, προκαλώντας μια «έκρηξη νοημοσύνης» με αποτέλεσμα αυτή να ξεπεράσει κατά πολύ την ανθρώπινη νοημοσύνη.



Εικόνα 3 Singuarity πηγή: https://towardsdatascience.com/singularity-may-not-require-agi-3fae8378b2

Αν υποθέσουμε ότι μία υπερφυής μηχανή είναι ένας υπολογιστής, σχεδιασμένος από έναν άνθρωπο και ο υπολογιστής αυτός μπορεί να ξεπεράσει όλες τις νοητικές ικανότητες από οποιονδήποτε άνθρωπο έχει υπάρξει, οσοδήποτε έξυπνος είναι ή ήταν αυτός, τότε αυτές οι μηχανές θα μπορούσαν να δημιουργήσουν μία ακόμα πιο έξυπνη μηχανή. Σε αυτή την περίπτωση μιλάμε για μια έκρηξη νοημοσύνης η οποία θα μπορούσε να αφήσει την νοημοσύνη του ανθρώπινου είδους πολύ πίσω. Για αυτόν τον λόγο μία υπερφυής μηχανή είναι η τελευταία μηχανή που θα χρειαστεί ο

άνθρωπος να φτιάξει υπό την προϋπόθεση ότι η μηχανή είναι αρκετά υπάκουη ώστε να μας πει πως να τη χαλιναγωγήσουμε.

Στο καλό σενάριο της έκρηξης νοημοσύνης, όσο οι υπολογιστές αυξάνονται σε νοητική δύναμη, είναι πιθανό οι άνθρωποι να φτιάξουν έναν υπολογιστή πιο έξυπνο από τον άνθρωπο. Αυτός ο υπολογιστής με την υπεράνθρωπη ευφυία, θα κατέχει πολύ καλύτερες ικανότητες ως προς τη λύση προβλημάτων και εφευρετικότητα. Αυτή η μηχανή θα μπορεί να σχεδιάσει μια πολύ πιο έξυπνη από αυτή την ήδη υπεράνθρωπα έξυπνη μηχανή η να βελτιώσει το λογισμικό του. Σε αυτή την περίπτωση οι μηχανές θα βοηθήσουν την ανθρωπότητα στην εξέλιξη προσφέροντας ταχύτητα στην λύση προβλημάτων και μία διαφορετική οπτική του κόσμου μας.[16][17]

Κεφάλαιο 2. Τεχνολογικό υπόβαθρο

2.1 Εκπαίδευση δικτύου

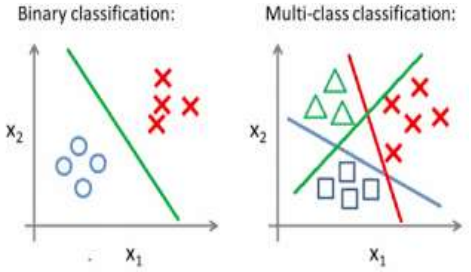
2.1.1 Με επόπτη

Υπάρχουν 3 είδη μηχανικής μάθησης

Επιβλεπόμενη Μάθηση ή εκμάθηση με επιβλέπον (Supervised Learning) κατά αυτή τη διαδικασία της μηχανικής μάθησης ο αλγόριθμος κατασκευάζει μια συνάρτηση που απεικονίζει δεδομένες εισόδους σε γνωστές, επιθυμητές εξόδους (τα δεδομένα εισόδου γνωρίζουμε σε ποια κλάση ανήκουν). Συχνά χρησιμοποιούμενη είναι σε προβλήματα

- Ταξινόμησης (Classification)
- Πρόγνωσης (Prediction)
- Support-vector machines (SVM)

Στον συγκεκριμένο τρόπο εκπαίδευσης χρησιμοποιούνται frameworks στατιστικής εκμάθησης. Τα οποία προτάθηκαν από Vapnik και Chervonenkis (1974) and Vapnik (1982, 1995). Δίνοντας ένα σύνολο δεδομένων προς



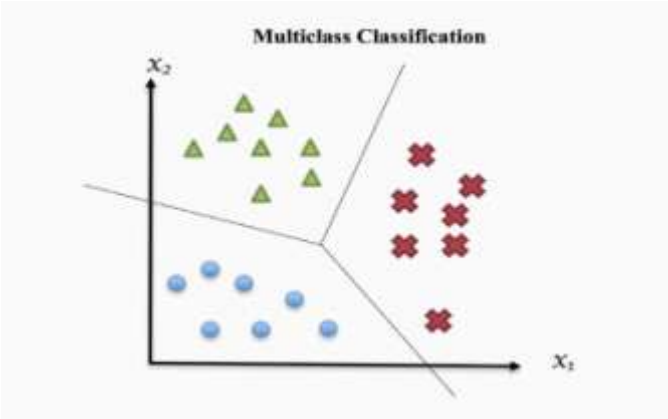
Εικόνα 4 Κατηγοριοποίηση με επόπτη πηγή: https://medium.com/@b.terryjack/tips-and-tricks-for-multi-class-classification-c184ae1c8ffc

εκπαίδευση όπου το καθένα ανήκει σε μια από τις 2 κατηγορίες ένα SVM δημιουργεί ένα μοντέλο το οποίο αναθέτει νέα δείγματα σε μία από τις δύο κατηγορίες. Ένα SVM μετατρέπει τα δείγματα (εισόδους) Σε σημεία στον χώρο έτσι ώστε να μεγιστοποιήσει το κενό ανάμεσα στις δύο κλάσεις

Στην εικόνα 4 φαίνεται η κατηγοριοποίηση σε δυο κλάσεις (αριστερα) και η κατηγοριοποίηση σε πολλαπλές κλάσεις(δεξιά) όπως και στην εικόνα 5. Τα διαφορετικά σχήματα αναπαριστούν διαφορετικές κλάσεις και οι ευθείες αναπαριστούν τον διαχωρισμό τον κλάσεων. Αν το δείγμα είναι πάνω από την πράσινη γραμμή (εικόνα 4 αριστερά) το δείγμα ταξινομείται στην κλάση με τα κόκκινα Χ ενώ αν είναι από κάτω της πράσινης γραμμής ανήκει στην κλάση Με τους κύκλους. Ομοίως λειτουργεί το παράδειγμα με τις πολλαπλές κλάσεις.

Γνωστοί αλγόριθμοι εκπαίδευσης με επόπτη

Τεχνητά νευρωνικά δίκτυα



Εικόνα 5 Κατηγοριοποίηση πολλαπλών κλάσεων πηγή: https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94daed32a87b

Ένας αλγόριθμος εκμάθησης Τεχνητού νευρωνικού δικτύου, που συνήθως ονομάζεται "νευρωνικό δίκτυο" (NN), είναι ένας αλγόριθμος μάθησης, που εμπνέεται από τη δομή και τις λειτουργικές πτυχές των βιολογικών νευρωνικών δικτύων. Η δομή των υπολογισμών βασίζεται σε μια ομάδα εσωτερικά διασυνδεδεμένων τεχνητών νευρώνων, οι οποίοι επεξεργάζονται την πληροφορία και εκτελούν υπολογισμούς επικοινωνώντας μεταξύ τους. Τα σύγχρονα νευρωνικά δίκτυα είναι εργαλεία μη γραμμικής στατιστικής μοντελοποίησης δεδομένων. Συνήθως χρησιμοποιούνται για τη μοντελοποίηση σύνθετων σχέσεων μεταξύ δεδομένων εισόδου και εξόδου, για την ανακάλυψη προτύπων στα δεδομένα, ή για τον εντοπισμό στατιστικής δομής σε μία άγνωστη κοινή κατανομή πιθανότητας μεταξύ των παρατηρούμενων μεταβλητών.

2.1.2 Χωρίς επόπτη

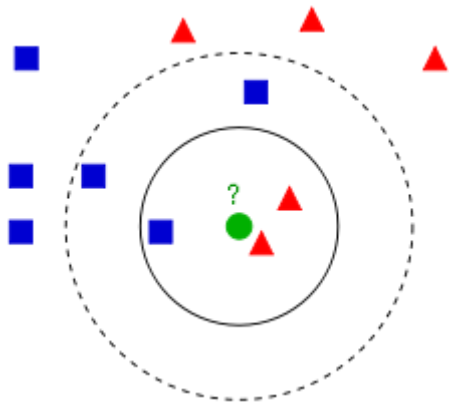
Στην μη επιβλεπόμενη μάθηση (Unsupervised Learning) ο αλγόριθμος κατασκευάζει ένα μοντέλο για κάποιο σύνολο εισόδων με τη μορφή των παρατηρήσεων χωρίς να γνωρίζει την κλάση στην οποία ανήκει το δείγμα (επιθυμητή έξοδος)

- Κατηγοριοποίηση
- Εντοπισμός ανωμαλιών

Γνωστοί αλγόριθμοι εκπαίδευσης χωρίς επόπτη

Αλγόριθμος Κ- κοντινότερων γειτόνων (K-nearest neighbor algorithm)

Ο συγκεκριμένος αλγόριθμος αφού εκπαιδευτεί με ένα σύνολο δεδομένων, όταν κληθεί να θέσει σε μια κατηγορία ένα νέο δείγμα θα τοποθετήσει την είσοδο ως σημείο στον χώρο και θα το κατατάξει στην κατηγορία όπου ανήκουν τα Κ περισσότερα γειτονικά στοιχεία γύρω του όπως φαίνεται στην εικόνα 6. Το μοντέλο καλείται να κατηγοριοποιήσει πράσινο δείγμα (κέντρο) γιαυτό τον λόγο κοιτάει τα Κ κοντινότερα δείγματα του πράσινου δείγματος (δείγματα μέσα σε κύκλο με συνεχόμενη γραμμή) και εφόσον τα περισσότερα δείγματα είναι κόκκινα τρίγωνα τότε το νέο δείγμα (πράσινος κύκλος) θα μπεί στην κλάση των κόκκινων τρίγωνων.



Εικόνα 6 Κατηγοριοποίηση χωρίς επόπτη με τη μέθοδο των Κ κοντινότερων γειτόνων

2.1.3 Ενισχυτική μάθηση (Reinforcement Learning)

Στην συγκεκριμένη διαδικασία ο αλγόριθμος μαθαίνει μια στρατηγική ενεργειών αλληλοεπιδρώντας με το περιβάλλον.

- Έλεγχος κίνησης ρομπότ
- Αυτό οδηγούμενα αυτοκίνητα

Για κάθε πρόβλημα μηχανικής μάθησης υπάρχει και ο κατάλληλος τρόπος μάθησης και για κάθε τύπο μάθησης υπάρχει τουλάχιστον ένας αλγόριθμος ο οποίο μπορεί να υλοποιηθεί.

2.1.4 Εκμάθηση με δέντρο απόφασης

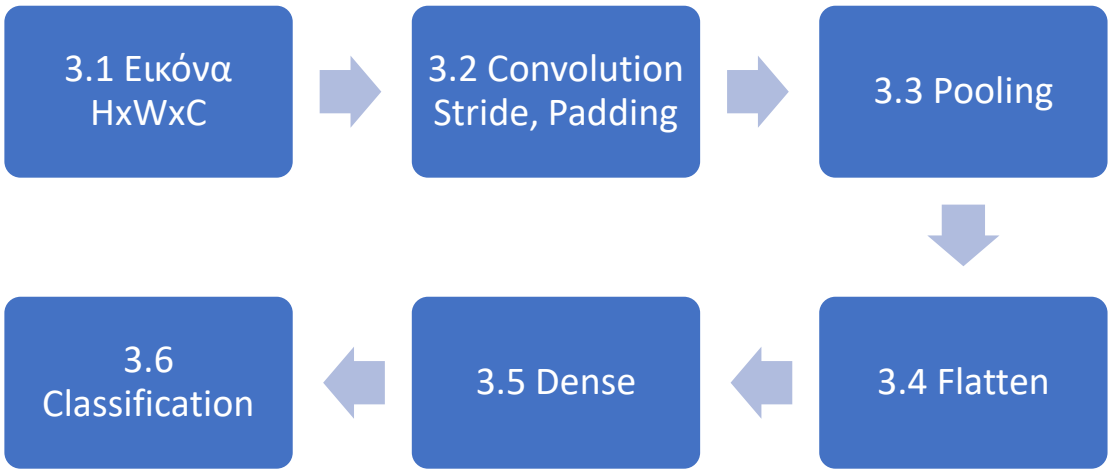
Η εκμάθηση με δέντρο απόφασης χρησιμοποιεί ένα δέντρο απόφασης ως προγνωστικό μοντέλο, το οποίο αντιστοιχίζει παρατηρήσεις σχετικά με ένα στοιχείο σε συμπεράσματα σχετικά με την τιμή στόχο του αντικειμένου.

2.1.5 Εκμάθηση με Κανόνες συσχέτισης

Η εκμάθηση με κανόνες συσχέτισης είναι μια μέθοδος ανακάλυψης ενδιαφερουσών σχέσεων μεταξύ των μεταβλητών σε μεγάλες βάσεις δεδομένων.

2.2 DEEP LEARNING

Η “βαθιά μάθηση” (deep learning) είναι ένα υποσύνολο της μηχανικής μάθησης. Δηλαδή πρόκειται για ένα πεδίο που εξετάζει τους αλγορίθμους υπολογιστών που μαθαίνουν και βελτιώνονται μόνοι τους. Με άλλα λόγια είναι απλά ένας άλλος τρόπος για να περιγράψουμε τα μεγάλα νευρωνικά δίκτυα, μια τεχνολογία που συναντάμε κάθε μέρα όταν κάνουμε περιήγηση στο διαδίκτυο ή ακόμα και όταν χρησιμοποιούμε το κινητό μας τηλέφωνο. Η κατάρτιση ενός μοντέλου βαθιάς μάθησης απαιτεί πολλά δεδομένα. Μάλιστα όσο περισσότερα είναι τα δεδομένα με τα οποία τροφοδοτείται τόσο πιο ακριβές θα είναι και το μοντέλο deep learning . Οι επιστήμονες έχουν καταφέρει να προσεγγίσουν όλο και περισσότερο την κατασκευή μοντέλων βαθιάς μάθησης που έχουν μεγαλύτερη ακρίβεια και που μπορούν να μάθουν χωρίς επίβλεψη. Έτσι το deep learning θα γίνει ταχύτερο και θα απαιτεί λιγότερη δουλειά. Αυτό φυσικά σημαίνει μεγαλύτερα και καλύτερα αποτελέσματα για το μέλλον αυτών των μοντέλων.



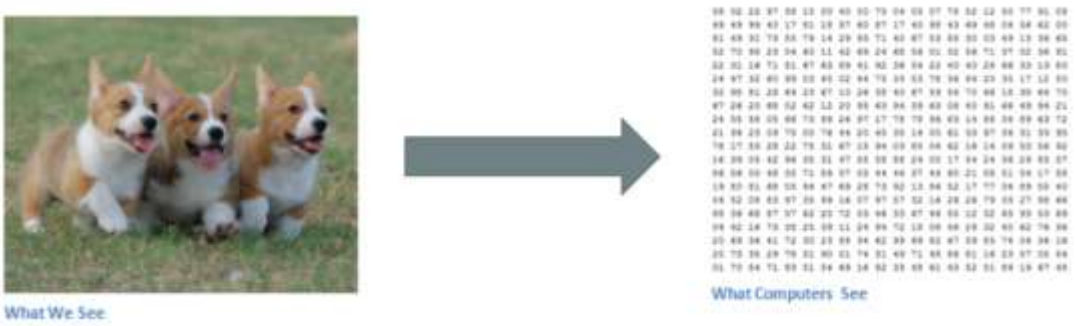
Κεφάλαιο 3. Εισαγωγή στα CNN δίκτυα

Σήμερα, το γενικότερο πρόβλημα της ταυτόχρονης αναγνώρισης και εντοπισμού αντικειμένων σε εικόνες λύνεται με την χρήση Νευρωνικών Δικτύων Συνέλιξης (Convolutional Neural Networks - CNNs). Το γεγονός ότι τα CNNs, σε συνδυασμό με μονάδες GPU, δίνουν την δυνατότητα επίλυσης προβλημάτων αναγνώρισης και εντοπισμού αντικειμένων σε μικρό χρόνο, τα καθιστά ικανά να χρησιμοποιηθούν σε εφαρμογές πραγματικού χρόνου.

Ένα CNN δίκτυο αποτελείται από ένα ή περισσότερα επίπεδα συνέλιξης (convolutional layers) συχνά μαζί με ένα επίπεδο υποδειγματοληψίας (πχ pooling) ακολουθούμενο από ένα ή περισσότερα πλήρως συνδεδεμένο (fully connected) επίπεδα όπως συμβαίνει και σε ένα κλασικό πολύ-επίπεδο νευρωνικό δίκτυο. Η αρχιτεκτονική του CNN σχεδιάζεται έτσι ώστε να εκμεταλλεύεται την 2 διαστάσεων δομή των εικόνων εισόδου ή άλλα δισδιάστατα σήματα όπως σήματα ήχου. Αυτό επιτυγχάνεται με τοπικές συνδέσεις και κατάλληλα βάρη ακολουθούμενα από επίπεδα δειγματοληψίας καταλήγοντας σε ένα πλήρως συνδεδεμένο επίπεδο το οποίο καταλήγει στο επίπεδο εξόδου που τελικά μας εμφανίζει την κλάση στην οποία ανήκει η είσοδος που δώσαμε.

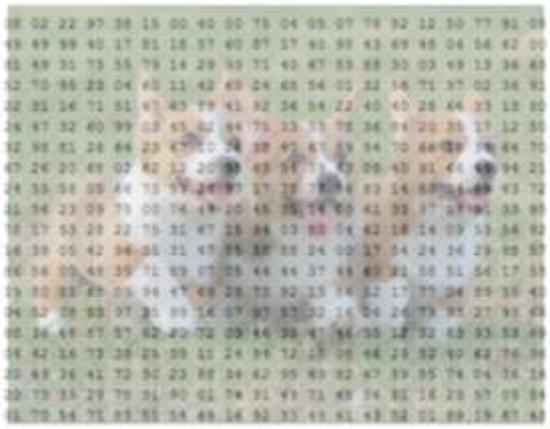
3.1 Εικόνα στον υπολογιστή

Αν υποθέσουμε ότι έχουμε μια φωτογραφία 480x480x3 (Υψος, Πλάτος, Χρωματικά κανάλια) ο τρόπος με τον οποίο ο υπολογιστής αναπαριστά αυτή τη φωτογραφία είναι με τη μορφή ενός πίνακα με τιμές ανάμεσα στο 0 και το 255 το οποίο αναπαριστά την ένταση του pixel στο συγκεκριμένο σημείο.



Εικόνα 7 Πως αντιλαμβάνεται ο υπολογιστής μια εικόνα

Έπειτα αναδιαμορφώνουμε την εικόνα στις επιθυμητές διαστάσεις και το βάζουμε ως είσοδο στο πρώτο επίπεδο (Convolutional layer)



Εικόνα 8 Πως αντιλαμβάνεται ο υπολογιστής την εικόνα 7

3.2 Επίπεδο συνέλιξης (Convolutional layer)

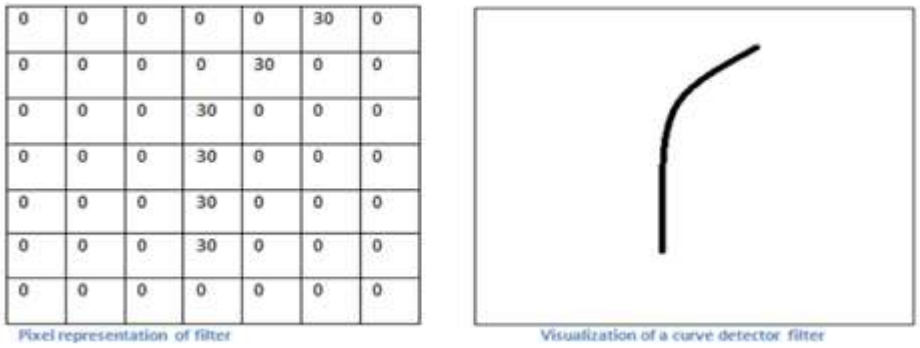
Εκεί υπάρχει ένα φίλτρο ή ένας νευρώνας ή ένα κέλυφος, το οποίο περνά πάνω από τα pixel της εικόνας που έχουμε ως είσοδο (ανάλογα τις διαστάσεις του φίλτρου).

Στην πραγματικότητα το φίλτρο περνάει πάνω από την εικόνα εισόδου και πολλαπλασιάζει τις τιμές του φίλτρου με τις τιμές των pixel της εικόνας. Οι τιμές αυτές προστίθενται και αποθηκεύονται σε έναν νέο πίνακα (Πίνακας χαρακτηριστικών). Η έξοδος αυτή μας λέει πόσο μοιάζει το συγκεκριμένο σημείο της εικόνας με το φίλτρο μας.

Η έξοδος μας, πίνακας χαρακτηριστικών, είναι ένας πίνακας που προκύπτει από τους πολλαπλασιασμούς του φίλτρου με την εικόνα. Τα κανάλια του φίλτρου είναι πάντα αυστηρά ίσα με τα κανάλια της εικόνας που έχουμε ως είσοδο. ΟΙ διαστάσεις της εξόδου εξαρτώνται από τον αριθμό των διαφορετικών φίλτρων των οποίων εφαρμόζουμε στην εικόνα μας.

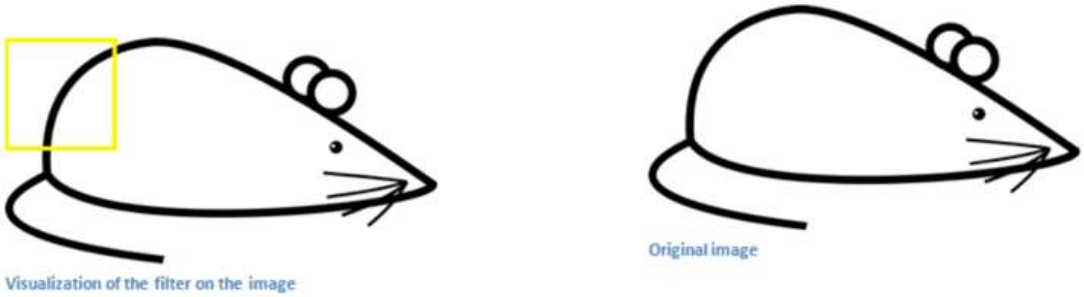
3.5.1 Παράδειγμα

Θα χρησιμοποιήσουμε ένα φίλτρο μεγέθους 7x7x1 και θα το εφαρμόσουμε σε μία εικόνα 32x32x1. Κάθε φορά που θα εφαρμόζουμε το φίλτρο θα το μετακινούμε μία στήλη στα δεξιά και στο τέλος της σειράς θα το μετακινούμε μία σειρά κάτω. Αυτό το φίλτρο θεωρείται να είναι ένα αναγνωριστικό ενός χαρακτηριστικού. Θα θεωρήσουμε ότι το φίλτρο μας είναι μια κυρτή γραμμή.

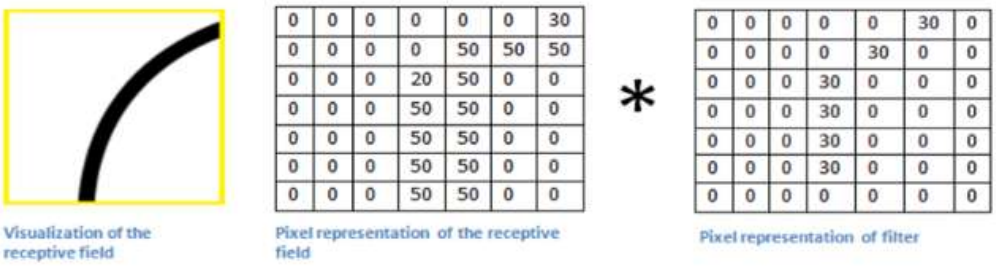


Εικόνα 9 Φίλτρο συνελικτικού νευρωνικού δικτύου

Στη συνέχεια βλέπουμε την αρχική εικόνα και το σημείο στο οποίο θα εφαρμόσουμε το φίλτρο.



Εικόνα 10 Εφαρμογή του φίλτρου



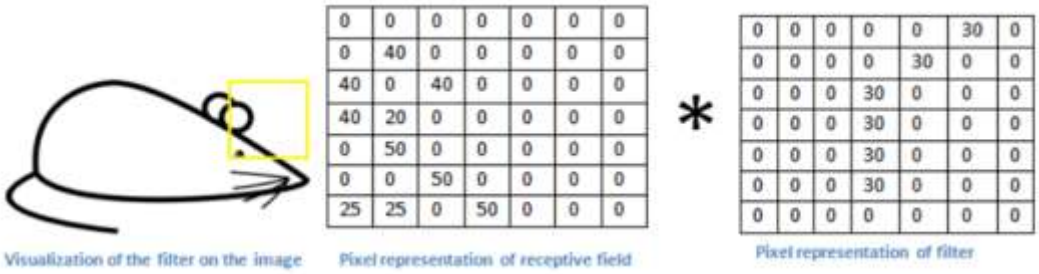
Εικόνα 11 Υπολογισμός γινομένου

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	3000	0	0	0
0	0	0	2100	0	0	0
0	0	0	3000	0	0	0
0	0	0	3000	0	0	0
0	0	0	3000	0	0	0

Πίνακας 1 ενδεικτικό αποτέλεσμα πολλαπλασιασμού πινάκων

Στον πίνακα 1 βλέπουμε το αποτέλεσμα που προκύπτει από τον πολλαπλασιασμό των πινάκων της εικόνας 11 όπου για να προκύψει το πρώτο κελί πολλαπλασιάζουμε την πρώτη γραμμή του αριστερού πίνακα με την πρώτη στήλη του δεξιού πίνακα και προσθέτουμε τα αποτελέσματα.

Τώρα θα μετακινήσουμε το φίλτρο σε ένα άλλο κομμάτι τη εικόνας μας.



Εικόνα 12 Εφαρμογή ίδιου φίλτρου σε διαφορετικό σημείο της εικόνας

0	0	0	0	0	0	0
0	0	0	0	1200	0	0
0	0	0	1200	0	1200	0
0	0	0	0	600	1200	0
0	0	0	0	1500	0	0
0	0	0	1500	0	0	0
0	0	0	1500	750	750	0

Πίνακας 2 αποτέλεσμα εφαρμογής φίλτρου σε διαφορετικό σημείο της εικόνας

Εφαρμόζοντας αυτό το φίλτρο σε όλα τα pixel της εικόνας παίρνουμε μια έξοδο από αυτό το επίπεδο που είναι ένας πίνακας χαρακτηριστικών που μας δείχνει τις περιοχές που είναι πιο πιθανό να είναι κυρτές στην εικόνα μας.

Χρησιμοποιώντας τον τύπο $(W - F + 2P)/S + 1$

Όπου W μέγεθος εικόνας εισόδου (32x32), F το μέγεθος του φίλτρου (7x7), P η επικάλυψη (0) και S μετατόπιση (1) προκύπτει ότι το μέγεθος της εξόδου θα είναι 26x26

Στο προηγμένο παράδειγμα η έξοδος μας θα είναι 1 πίνακας χαρακτηριστικών διαστάσεων 26x26. Η εφαρμογή ενός φίλτρου που αναγνωρίζει κυρτές γραμμές προς τα δεξιά στην πάνω πλευρά, όσο περισσότερα τα φίλτρα τόσο περισσότερο το

βάθος του πίνακα χαρακτηριστικών και τόση περισσότερη πληροφορία έχουμε για την εικόνα εισόδου.

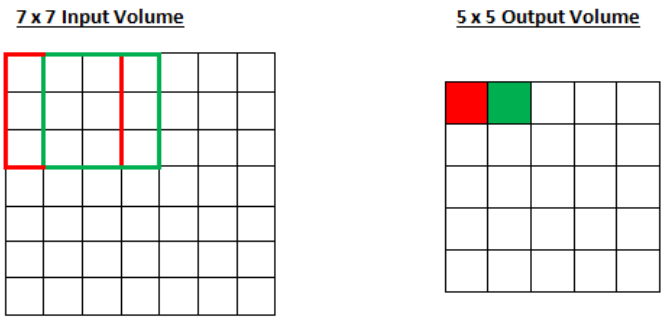
Όσο προσθέτουμε παρόμοια επίπεδα μετά από αυτό, η έξοδος του πρώτου (πίνακες χαρακτηριστικών) γίνεται είσοδος στο δεύτερο κλπ. Επομένως η είσοδος κάθε επιπέδου ουσιαστικά περιγράφει τις περιοχές της αρχικής εικόνας στις οποίες εμφανίζονται συγκεκριμένα χαρακτηριστικά χαμηλού επιπέδου.

Όταν εφαρμόζουμε ένα σύνολο από επιπρόσθετα φίλτρα (Βάζοντας ένα δεύτερο η περισσότερα Convolutional layers) η έξοδος θα είναι χαρακτηριστικά υψηλότερου επιπέδου. Τύποι αυτών των χαρακτηριστικών είναι ημικύκλια, (ο συνδυασμός μιας καμπύλης και μιας ευθείας) ένα τετράγωνο(ο συνδυασμός διάφορων γωνιών) όσο προχωράμε στο δίκτυο και περνάμε από περισσότερα τέτοια επίπεδα παίρνουμε όλο και πιο σύνθετους πίνακες χαρακτηριστικών που αναπαριστούν πιο σύνθετα χαρακτηριστικά.

Μέχρι το τέλος του δικτύου μπορεί ένα έχουμε χαρακτηριστικά που ενεργοποιούν φίλτρα όταν αναγνωρίσουν ένα χειρόγραφο μήνυμα στην εικόνα η ένα χρώμα.

3.2.1 Μετατόπιση (Stride)

Η μετατόπιση είναι είναι μια παράμετρος η οποία χρησιμοποιείται στα συνελικτικά νευρωνικά δίκτυα για να περιγράψει την κίνηση ενός φίλτρου σε μία είσοδο.

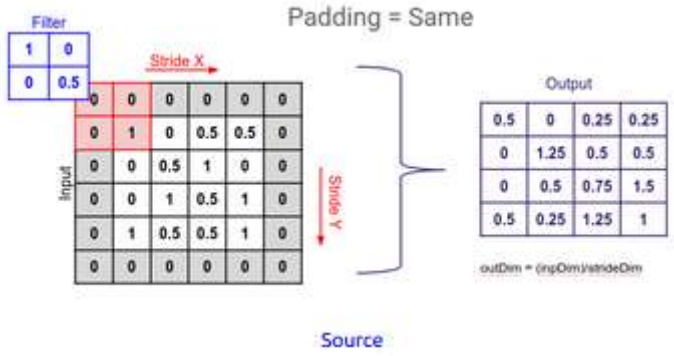


Εικόνα 13. Παράδειγμα εφαρμογής φίλτρου με μετατόπιση = 1 πηγή: https://deepai.org/machine-learning-glossary-and-terms/stride

Στην εικόνα 9 βλέπουμε την εφαρμογή ενός φίλτρου 3x3 με μετατόπιση 1 pixel τη φορά και την έξοδο την οποία παίρνουμε.

3.2.2 Επικάλυψη (Padding)

Η επικάλυψη (padding) είναι μια ακόμα παράμετρος η οποία χρησιμοποιείται στα συνελικτικά νευρωνικά δίκτυα και αναφέρεται στον αριθμό των pixel που προστίθενται σε μία εικόνα όταν εφαρμόζεται σε αυτή ένα φίλτρο. Αν για παράδειγμα θέσουμε την επικάλυψη ίση με ένα τότε στην είσοδο μας θα προστεθεί ένα μηδενικό pixel για να μπορέσουμε να εφαρμόσουμε με μεγαλύτερη ακρίβεια το φίλτρο στην είσοδο μας.



Εικόνα 14 παράδειγμα επικάλυψης = 1 πηγή: https://deepai.org/machine-learning-glossary-and-terms/padding

Στην εικόνα 10 βλέπουμε τα pixel που εχουν προστεθεί (σκούρα γκρί κουτάκια) περιμετρικά της εισόδου (λευκά κουτάκια) και τον τρόπο με τον οποίο τώρα εφαρμόζεται το φίλτρο (μπλε κουτάκια).

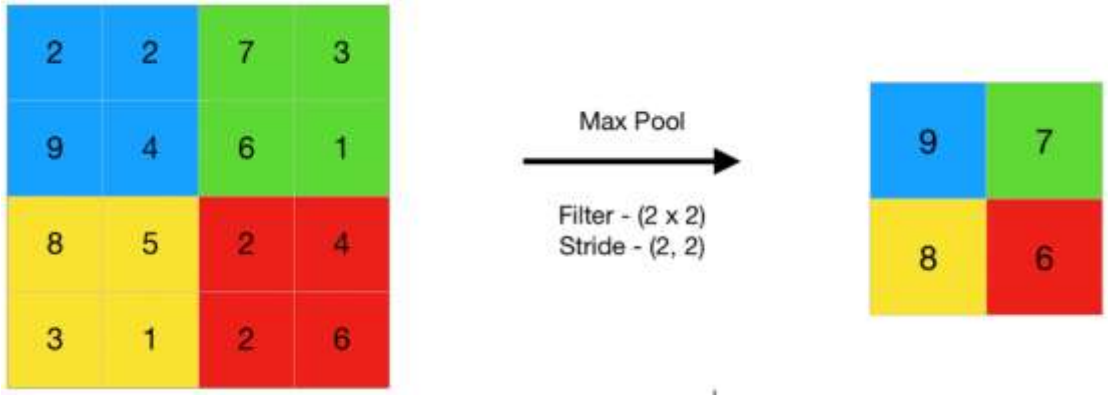
3.3 Επίπεδο Υποδειγματοληψίας (Pooling Layer)

Το επίπεδο υποδειγματλειψίας χρησιμοποιείται για τη μείωση των διαστάσεων. Επίσης ανάλογα τον τύπο της υποδειγματολειψίας που επιλέγουμε (βλέπε συνέχεια), επιλέγουμε τα πιο συχνά εμφανιζόμενα χαρακτηριστικά τα λίγοτερο συχνά εμφανιζόμενα χαρακτηριστικά κλπ. Ο τρόπος με τον οποίο λειτουργεί αυτό το πλήρως συνδεδεμένο δίκτυο είναι με το να παίρνει ως είσοδο την έξοδο του προηγουμένου επιπέδου (Πίνακας Χαρακτηριστικών) και να μειώνει τις διαστάσεις του.

Η διαδικασία αυτή μπορεί να επιτευχθεί με 3 τρόπους, MaxPooling, MinPooling, AveragePooling. Σε κάθε περίπτωση παίρνουμε κομμάτια της εισόδου (στις

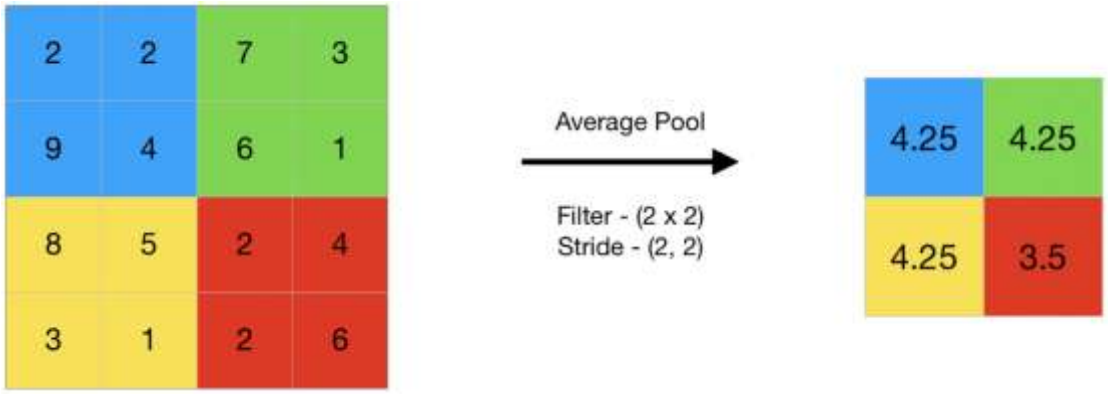
περισσότερες περιπτώσεις 2x2) και βρίσκουμε το χαρακτηριστικό με τη μεγαλύτερη τη μικρότερη η τη μέση τιμή αντίστοιχα.

Παράδειγμα



Εικόνα 15 Εφαρμογή Max Pooling

Στην εικόνα 16 και 17 το φίλτρο μας είναι 2x2 και το stride είναι 2,2 δηλαδή το φίλτρο μετακινείται 2 pixel δεξιά και 2 προς τα κάτω (όταν φτάσει στο τέλος της πρώτης σειράς) δεν υπάρχει δηλαδή επικάλυψη.



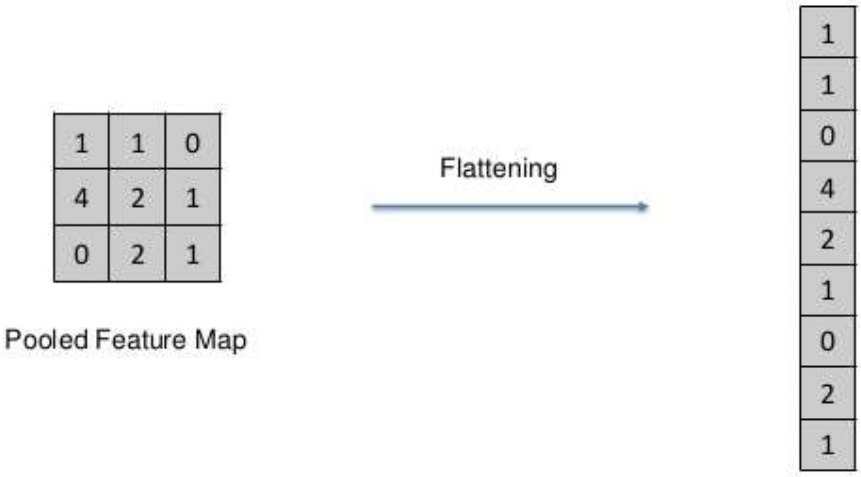
Εικόνα 16 Εφαρμογή Average Pooling

3.4 Flatten

Το Flatten Layer δημιουργεί ένα πλήρως συνδεδεμένο επίπεδο με αριθμό νευρώνων ίσο με τον αριθμό των διαστάσεων του προηγούμενου επιπέδου (Υψος pixel x Πλάτος pixel). Αν έχουμε πχ μια εικόνα διαστάσεων 24*24 και μετά

προσθέσουμε ένα Flatten Layertότε ο αριθμός των νευρώνων που θα έχει είναι 576 νευρώνες.

3.4.1 Παράδειγμα



Εικόνα 17 Οπτικοποίηση Flatten layer πηγή: https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening

3.5 Dense Layer

Το dense layer είναι ένα επίπεδο CNN δικτύων το οποίο είναι πλήρως συνδεδεμένο το οποίο σημαίνει πως κάθε νευρώνας το επιπέδου δέχεται είσοδο από όλους τους νευρώνες του προηγούμενου επιπέδου. Το dense layer είναι ίσως το πιο συχνά χρησιμοποιούμενο επίπεδο στα Convolutional Neural Networks (CNN)

Είναι ένα μη γραμμικό επίπεδο, του οποίο το αποτέλεσμα περνάει από μια συνάρτηση ενεργοποίησης. Αν χρησιμοποιήσουμε συνάρτηση ενεργοποίησης τότε το επίπεδο γίνεται γραμμικό.

Παρασκηνιακά το Dense Layer πραγματοποιεί έναν πολλαπλασιασμό πίνακα με διάνυσμα. Οι τιμές που υπάρχουν στον πίνακα είναι στην πραγματικότητα οι παράμετροι οι οποίοι μπορούν να εκπαιδευτούν και να βελτιωθούν με τη μέθοδο του backpropagation. Η έξοδος είναι συνήθως ένα διάνυσμα μ διαστάσεων. Παρόλα αυτά το dense layer χρησιμοποιείται κατά κόρον για την αλλαγή των διαστάσεων του διανύσματος.

3.6 Συνάρτηση Ενεργοποίησης (Activation Function)

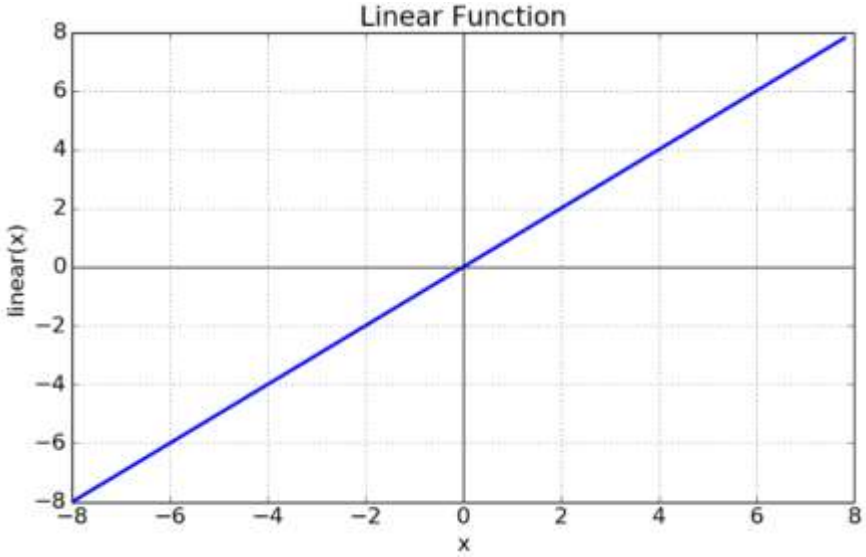
Στα Τεχνητά Νευρωνικά Δίκτυα η συνάρτηση ενεργοποίησης ενός κόμβου προσδιορίζει την έξοδο αυτού του κόμβου αφού δώσουμε μια είσοδο η ενός συνόλου εισόδων.

Οι πιο συχνά χρησιμοποιούμενες συναρτήσεις ενεργοποίησης μπορούν να διαιρεθούν σε 3 κατηγορίες.

- Ridge functions[10]
- Radial functions[11]
- Fold functions

Ridge functions είναι συναρτήσεις πολλών μεταβλητών οι οποίες δρουν σε έναν γραμμικό συνδυασμό με τις μεταβλητές εισόδου. Συχνά χρησιμοποιούμενες συναρτήσεις είναι:

Γραμμική συνάρτηση ενεργοποίησης (Linear activation): $f(x) = x$



Γράφημα 1 Γραφική παράσταση της Linear activation πηγή: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Από τον τύπο και την γραφική της παράστασης βλέπουμε πως η έξοδος x θα πάρει την πραγματική τιμή που έχεις την είσοδο της συνάρτησης f(x)

ReLU activation: $$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases}$$

$$f(u) =$$

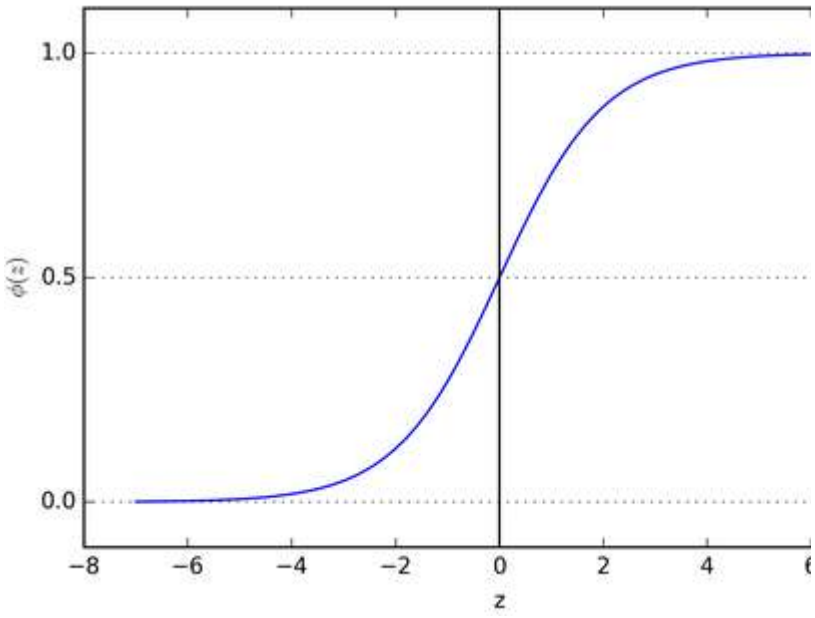
$$1$$

$$0$$

Γράφημα 2 Γραφική παράσταση της ReLU πηγή: https://www.researchgate.net/figure/ReLU-activation-function_fig3_319235847

Από τον τύπο και την γραφική παράσταση της εξίσωσης αυτής βλέπουμε ότι όλες οι τιμές που είναι μικρότερες του 0 γίνονται ίσες με 0, ενώ οι τιμές που είναι μεγαλύτερες η ίσες με το 0 παίρνουν την πραγματική τους τιμή.

Σιγμοειδής συνάρτηση ενεργοποίησης (Sigmoid activation): $$\phi(z) = \frac{1}{1 + e^{-z}}$$



Γράφημα 3 Γραφική παράσταση της Sigmoid Function πηγή: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Τέλος η σιγμοειδής συνάρτηση εκτείνεται από το 0 έως το 1 (υπάρχουν παραλλαγές της από -1 έως 1) υπολογίζοντας από τον τύπο όπου –z είναι η τιμή που παίρνει η συνάρτηση ως είσοδο.

3.6.1 Ακτινικές συναρτήσεις ενεργοποίησης (Radial activation functions)

Μια ειδική κατηγορία συναρτήσεων ενεργοποίησης είναι οι Radial functions. Οι συναρτήσεις αυτές είναι εξαιρετικά αποδοτικές σε Radial basis function networks.

3.6.2 Folding activation functions

Folding activation functions χρησιμοποιούνται εκτενώς στα pooling layers στα CNN, και στα επίπεδα εξόδου σε προβλήματα κατηγοριοποίησης πολλών κλάσεων. Αυτές οι συναρτήσεις παίρνουν το σύνολο της εισόδου όπως πχ το να παίρνουν τον μέσο όρο την μεγαλύτερη η την μικρότερη τιμή. Σε κατηγοριοποίηση πολλών κλάσεων η συνάρτηση softmax χρησιμοποιείται πιο συχνά.

3.6.3 SoftMax

Η συνάρτηση ενεργοποίησης Softmax είναι μία γενική μορφή της λογιστικής συνάρτησης (logistics Function) σε πολλαπλές διαστάσεις. Στα CNN δίκτυα συνήθως χρησιμοποιείται στο τελευταίο επίπεδο ως κατηγοριοποιητής η συνάρτηση δηλαδή που θα βγάλει την κλάση στην οποία ανήκει το δείγμα που της βάλαμε ως είσοδο.

Περιγράφεται από τον τύπο της εικόνας 9

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

Εικόνα 18. Συνάρτηση Ενεργοποίησης Softmax πηγή:
https://en.wikipedia.org/wiki/Softmax_function#Neural_networks

Στην δική μας έρευνα χρησιμοποιούμε την softmax επειδή μας δίνει μια κατανομή της πιθανότητας για το δείγμα που βάζουμε ως είσοδο. Αυτό σημαίνει πως μας λέει ποια είναι η πιθανότητα το δείγμα να ανήκει στην κάθε κλάση. Τελικά ανήκει στην κλάση με την μεγαλύτερη πιθανότητα. Πιο αναλυτικά στον τύπο, το s(z)i είναι η κατανομή της πιθανότητας που προκύπτει από το κλάσμα με αριθμητή το διάνυσμα εξόδου για κάθε κλάση και παρονομαστή το άθροισμα της πιθανότητας.

Στην εικόνα το i είναι ο αριθμός των κλάσεων (σε μας 1 έως 6) και z η πιθανότητα να ανήκει η είσοδος σε μία κλάση επομένως z1 η πιθανότητα να ανήκει το δείγμα στην κλάση 1 κλπ.

3.7 Βελτιστοποιητές (Optimizers)

Οι βελτιστοποιητές είναι αλγόριθμοι ή μέθοδοι που χρησιμοποιούνται για να αλλάξουν τα γνωρίσματα του νευρωνικού δικτύου όπως είναι τα βάρη ή το ρυθμό εκμάθησης (το πόσο αλλάζουν τα βάρη στην κάθε διόρθωση).

3.7.1 Adam

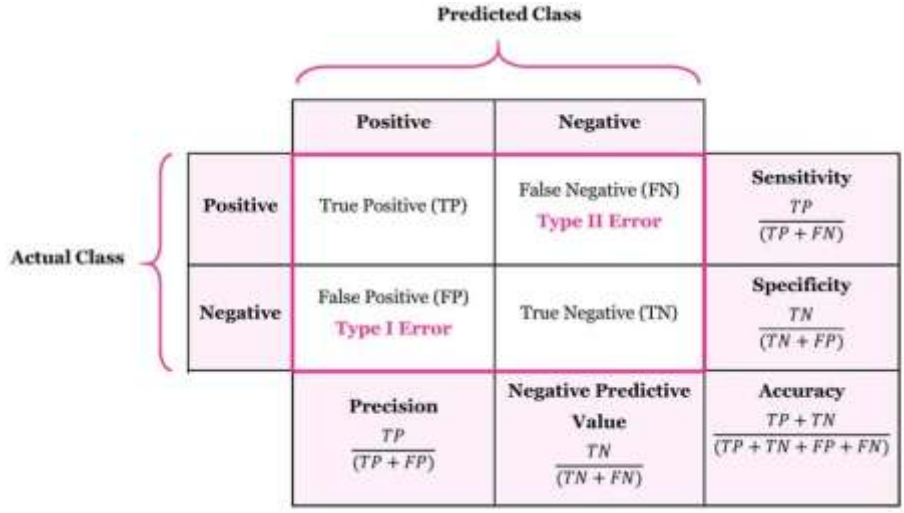
Ο βελτιστοποιητής adam είναι μία συνάρτηση που προσαρμόζει τον ρυθμό εκμάθησης του μοντέλου. Χρησιμοποιεί την απλή γραμμική παλινδρόμηση για να τροποποιήσει τον ρυθμό εκμάθησης χρησιμοποιώντας διάφορες παραμέτρους. Το όνομα του (adam) προκύπτει από το adaptive moment estimation (προσαρμοστική στιγμιαία εκτίμηση) ο λόγος για τον οποίο λέγεται έτσι είναι επειδή χρησιμοποιεί εκτιμήσεις από την πρώτη και τη δεύτερη «στιγμή» της γραμμικής παλινδρόμησης. Με την εκτίμηση αυτή αλλάζει τον ρυθμό εκμάθησης για κάθε βάρος του μοντέλου.

Ως ν-ιοστή στιγμή μιας τυχαίας μεταβλητής ορίζεται η αναμενόμενη τιμή της μεταβλητής στην δύναμη ν. Πιο συγκεκριμένα $m_n = E[x^n]$ [20]

Κεφάλαιο 4. Αποτίμηση μοντέλου

4.1 πίνακας σύγχυσης (Confusion Matrix)

Στον τομέα της μηχανικής μάθησης και συγκεκριμένα το πρόβλημα της στατιστικής ταξινόμησης, ένας πίνακας σύγχυσης, επίσης γνωστός ως πίνακας σφάλματος, είναι μια συγκεκριμένη διάταξη πίνακα που επιτρέπει την οπτικοποίηση της απόδοσης ενός αλγορίθμου.



Εικόνα 19 Επεξήγηση του πίνακα σύγχυσης πηγή: https://manisha-sirsat.blogspot.com/2019/04/confusion-matrix.html

TP: True Positive: οι τιμές οι οποίες κατηγοριοποιήθηκαν από το μοντέλο ως θετικές ανήκουν όντως στα θετικά

FP: False Positive οι τιμές οι οποίες εσφαλμένα κατηγοριοποιήθηκαν στις θετικές.

FN: False Negative: θετικές τιμές οι οποίες κατηγοριοποιήθηκαν ως αρνητικές

TN: True Negative: οι αρνητικές τιμές που κατηγοριοποιήθηκαν όντως ως αρνητικές

Έχοντας τα δεδομένα αυτά μπορούμε να υπολογίσουμε την ακρίβεια του μοντέλου με τον παρακάτω τύπο.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Εικόνα 20 Τρόπος υπολογισμού της ακρίβειας

Metric	Formula
True positive rate, recall	$\frac{TP}{TP+FN}$
False positive rate	$\frac{FP}{FP+TN}$
Precision	$\frac{TP}{TP+FP}$
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$
F-measure	$\frac{2 \cdot precision \cdot recall}{precision + recall}$

Εικόνα 21 Σημαντικότερες μετρικές και οι τρόποι υπολογισμού αυτών

4.2 True Positive Rate και False Positive Rate

Διαφορετικά αποκαλούνται και ως ευαισθησία και ειδίκευση (sensitivity and specificity) . Οι 2 αυτές έννοιες είναι τρόποι μέτρησης της απόδοσης στατιστικά ενός θέματος κατηγοριοποίησης 2 κλάσεων.

4.3 Ευαισθησία (Sensitivity)

Υπολογίζει τον αριθμό των δειγμάτων που έχουν κατηγοριοποιηθεί στη σωστή κλάση σε σχέση με το σύνολο των δειγμάτων που έπρεπε να κατηγοριοποιηθούν σε αυτή όπως φαίνεται στην εικόνα 21.

$$\text{sensitivity} = \frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}}$$

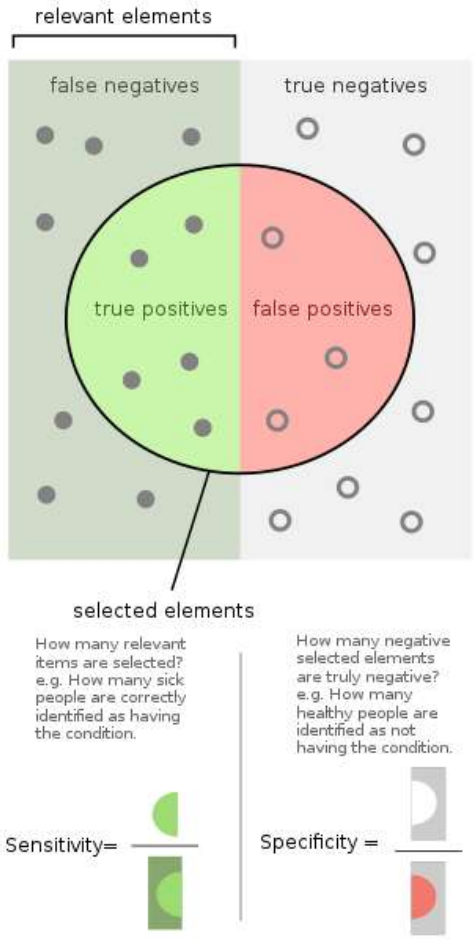
Εικόνα 22. Τύπος Ευαισθησίας

4.4 Ειδίκευση (Specificity)

Υπολογίζει τον αριθμό των δειγμάτων τα οποία κατηγοριοποιήθηκαν στη $2^{η}$ κλάση σε σχέση με το σύνολο των δειγμάτων που έπρεπε να κατηγοριοποιηθούν σε αυτή όπως φαίνεται στην εικόνα 22.[12]

$$\text{specificity} = \frac{\text{number of true negatives}}{\text{number of true negatives} + \text{number of false positives}}$$

Εικόνα 23. Τύπος ειδίκευσης



Εικόνα 24 Ειδίκευση και ευαισθησία πηγή: https://en.wikipedia.org/wiki/Sensitivity_and_specificity

Στην εικόνα 23 έχουμε οπτικοποιήσει τις εννοιες Ειδίκευση και ευαισθησία

4.5 Ακρίβεια (Precision)

Στην μηχανική μάθηση και στην αναγνώριση προτύπων ο όρος ακρίβεια (Precision) είναι το κλάσμα των δειγμάτων μιας κλάσης ως προς το σύνολο των δεδομένων που το μοντέλο αναγνώρισε πως ανήκουν σε αυτή την κλάση όπως βλέπουμε στον τύπο της εικόνας 22.[13]

$$\text{precision} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{retrieved documents}\}|}$$

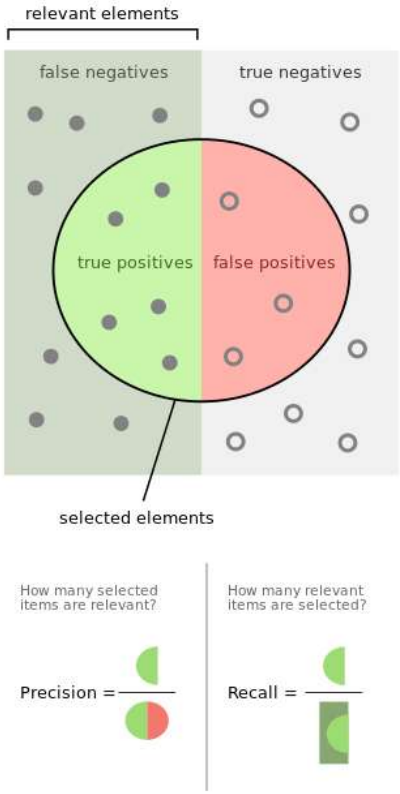
Εικόνα 25. Τύπος ακρίβειας

4.6 Επισκόπηση (Recall)

Ο όρος επισκόπηση είναι το κλάσμα των δειγμάτων που κατηγοριοποιήθηκαν σε μια κλάση και όντως ανήκουν σε αυτή προς τον συνολικό αριθμό δειγμάτων που κατηγοριοποιήθηκαν σε αυτή την κλάση είτε σωστά είτε εσφαλμένα όπως βλέπουμε στην εικόνα 22.

$$\text{recall} = \frac{|\{\text{relevant documents}\} \cap \{\text{retrieved documents}\}|}{|\{\text{relevant documents}\}|}$$

Εικόνα 26. Τύπος Επισκόπισης



Εικόνα 27 Ακρίβεια και επισκόπηση πηγή: https://en.wikipedia.org/wiki/Precision_and_recall

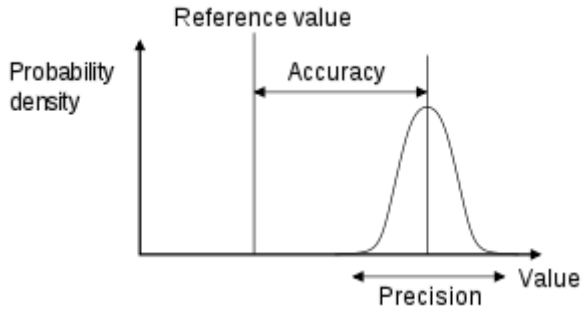
Στην εικόνα 26 έχουμε οπτικοποιήσει τις εννοιες της ακρίβειας και της επισκόπισης.

4.7 Πιστότητα (Accuracy)

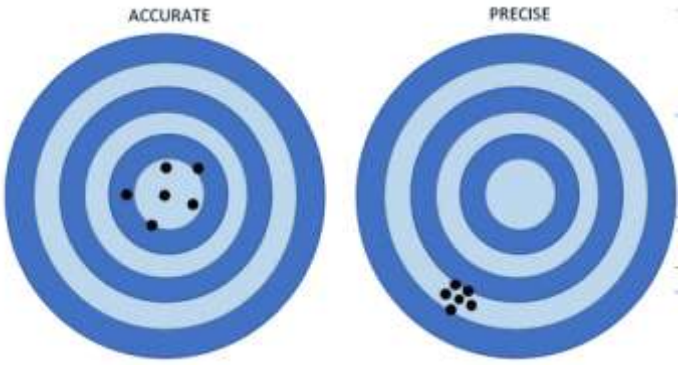
Η πιστότητα (accuracy) είναι ένας τρόπος αποτίμησης ενός μοντέλου κατηγοριοποίησης. Η ακρίβεια είναι το κλάσμα του αριθμού των δειγμάτων που σωστά κατηγοριοποιήθηκαν στις εκάστοτε κλάσεις προς το σύνολο των δειγμάτων. Ο τρόπος υπολογισμού της φαίνεται στην εικόνα 19

4.7.1 Διαφορά Accuracy και Precision

Η βασική διαφορά ανάμεσα στις 2 αυτές μετρικές είναι ότι το Accuracy υπολογίζει τιμές τις οποίες είναι οι σωστά κατηγοριοποιημένες δηλαδή οι τιμή που υπολογίστηκε από το μοντέλο είναι κοντά στην πραγματική ενώ το Precision είναι το πόσο συνεπή είναι τα αποτελέσματα όταν οι υπολογισμοί επαναλαμβάνονται. Οι τιμές του precise διαφέρουν μεταξύ τους λόγω του τυχαίου λάθους.



Εικόνα 28 Διαφορά Accuracy και precision πηγή: https://www.researchgate.net/figure/Figure-7-The-difference-between-accuracy-and-precision_fig2_280297590



Εικόνα 29. Διαφορά πιστότητας και ακρίβειας πηγή: https://www.precisa.co.uk/difference-between-accuracy-and-precision-measurements/

Στην παρούσα εργασία χρησιμοποιήθηκε η πιστότητα (accuracy) επειδή μας επιστρέφει συγκεντρωτικά αποτελέσματα για τον μοντέλο μας.

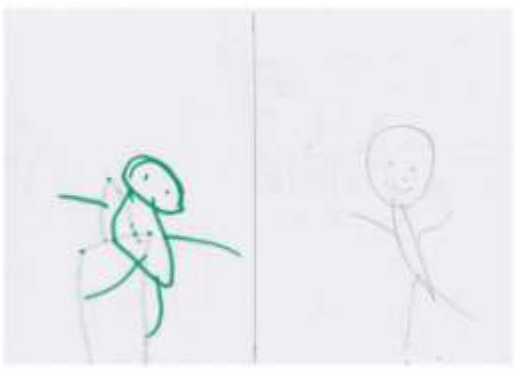
Κεφάλαιο 5. Δεδομένα

Τα δεδομένα τα οποία χρησιμοποιήθηκαν σε αυτή την εργασία είναι φωτογραφίες. Οι φωτογραφίες αυτές απεικονίζουν ζωγραφιές παιδιών συγκεκριμένου εύρους ηλικίας. Οι ζωγραφιές αυτές απεικονίζουν τη φιγούρα ενός άνδρα και μίας γυναίκας. Σκοπός μας είναι να αναλύσουμε τα δείγματα και να δούμε την διαφορετική απόδοση που έχουν διαφορετικές αρχιτεκτονικές CNN δικτύων. Και το πόσο καλά μπορούν αυτές οι ζωγραφιές να κατηγοριοποιηθούν σε 6 κλάσεις ανάλογα την λεπτή κινητικότητα του παιδιού. Όσο πιο υψηλή είναι η κλάση τόσο καλύτερα χαρακτηρίζεται η λεπτή κινητικότητα του παιδιού (Καλύτερη η κλάση 6 χειρότερη η κλάση 1)

Εικόνα 30δείγμα φωτογραφίας από την κλάση 1

Εικόνα 31 Δείγμα φωτογραφίας από την κλάση 2

Εικόνα 32 Δείγμα φωτογραφίας από την κλάση 3



Εικόνα 33δείγμα φωτογραφίας από την κλάση 4

Εικόνα 34 Δείγμα φωτογραφίας από την κλάση 5

Εικόνα 35 Δείγμα φωτογραφίας από την κλάση 6

5.1 Σύνολο Δεδομένων Εκπαίδευσης (Training Set)

Από το αρχικό σύνολο δεδομένων (64 εικόνες) επιλέξαμε τυχαία να κρατήσουμε 54 για το σύνολο εικόνων το οποίο θα χρησιμοποιούσαμε κατά την εκπαίδευση του μοντέλου μας. Οι 54 αυτές εικόνες κατανέμονται όμοια στις 6 κλάσεις (9 εικόνες ανά κλάση) ο λόγος για τον οποίο επιλέξαμε να έχουμε όμοιο αριθμό εικόνων σε κάθε κλάση είναι επειδή θέλαμε κατά την εκπαίδευση του μοντέλου μας να μην υπάρχει υπερτροφοδότηση μιας κλάσης και υποτροφοδότηση των υπόλοιπων.

5.2 Σύνολο Δεδομένων Επαλήθευσης (Test Set)

Το σύνολο δεομένων που χρησιμοποιήθηκε κατά την επαλήθευση του μοντέλου, αποτελούνταν από 10 εικόνες. Οι 10 αυτές εικόνες είναι δείγματα απ' όλες τις κλάσεις. Κάποιες κλάσεις έχουν από 1 εικόνα στο σύνολο επαλήθευσης (κλάση 1, 2) και οι υπόλοιπες κλάσεις από 2 εικόνες. Ο λόγος για αυτή την ανόμοια κατανομή είναι ο περιορισμένος αριθμός των εικόνων στο αρχικό σύνολο δεδομένων και η επιθυμία μας για ομοιόμορφη κατανομή στο σύνολο εκπαίδευσης

5.3 Στάδιο προεπεξεργασίας

Για να μπορέσουμε να περάσουμε τις φωτογραφίες στα δίκτυα θα πρέπει πρώτα να τις περάσουμε από ένα στάδιο προ-επεξεργασίας. Στο στάδιο αυτό γράφτηκε ένα script το οποίο μετατρέπει όλες τις φωτογραφίες σε συγκεκριμένες διαστάσεις. Οι διαστάσεις αυτές είναι 224x224x3 (Μήκος εικόνας x Πλάτος εικόνας x Χρωματικά Κανάλια). Ο λόγος για τον οποίο επιλέξαμε τις διαστάσεις αυτές είναι επειδή τα μοντέλα τα οποία χρησιμοποιήσαμε έχουν ως προαπαιτούμενο, η είσοδος τους να είναι αυτών των συγκεκριμένων διαστάσεων.

Επίσης λόγω του περιορισμένου αριθμού δειγμάτων που είχαμε χωρίσαμε την κάθε φωτογραφία στη μέση και προέκυψαν από την κάθε φωτογραφία 2 νέες. Η μία περιλαμβάνει τη φιγούρα του άνδρα ενώ η άλλη της γυναίκας. Για τη διαδικασία αυτή χρειάστηκε να γράψουμε ένα script σε γλώσσα python το οποίο χώριζε την εικόνα σε 2 νέες. Η πρώτη από το pixel 0 έως το pixel στο μέσο της εικόνας (συνολικό μήκος/2), και η δεύτερη από τη μέση ως το τέλος της εικόνας.

Εικόνα 36 Φιγούρα άνδρα της εικόνας 1

Εικόνα 37 Φιγούρα γυναίκας της εικόνας 1

Κεφάλαιο 6. Μοντέλα τα οποία χρησιμοποιήθηκαν

Τα μοντέλα τα οποία χρησιμοποιήθηκαν στην παρούσα εργασία είναι 4. Τα 3 από αυτά είναι μοντέλα προ εκπαιδευμένα. Αυτό σημαίνει ότι χρησιμοποιήσαμε το μοντέλο χρησιμοποιώντας τα βάρη τα οποία είχαν δημιουργηθεί από την εκπαίδευση του μοντέλου στο ImageNet. Το image-net είναι μια βάση δεδομένων με εικόνες οργανωμένη σύμφωνα με κάποια ιεραρχία. Η ιεραρχία αυτή αποτελείται από κόμβους, ο κάθε κόμβος έχει κάποιο κόμβο παιδί και κάποιο γονέα. Ο κάθε κόμβος της ιεραρχίας αποτελείται από εκατοντάδες χιλιάδες εικόνες. Το image-net διοργανώνει διάφορες προκλήσεις για δημιουργούς δικτύων τεχνητής νοημοσύνης με σκοπό να δοκιμαστούν σε συγκεκριμένο σύνολο εικόνων το οποίο τους παρέχει. Στο website του image-net [22] μπορούμε να βρούμε περισσοτερες πληροφορίες για την ιεραρχία (WordNet [23]) τους διαγωνισμούς καθώς και για λήψη δεδομένων που μπορούν να χρησιμοποιηθούν για εκπαίδευση δικτύων Μηχανικής Μάθησης.

Τα μοντέλα τα οποία πρόκειται να χρησιμοποιήσουμε είναι τα εξής:

GoogleNet InceptionV3 model [6] το οποίο είναι ένα μοντέλο το οποίο δημιουργήθηκε από την Google.

ResNetV2 το οποίο είναι ένα δίκτυο των Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun με βάθος 152 επιπέδων κάνοντας το 8 φορές πιο βαθύ από τα VGG δίκτυα. [7]

Mobile Net [8] το οποίο είναι ένα δίκτυο που μπορεί να χρησιμοποιηθεί σε συσκευές με περιορισμένους πόρους μνήμης.

MotorSkillsCNN αυτό το δίκτυο κατασκευάστηκε από εμάς κατά τη διάρκεια αυτής της εργασίας.

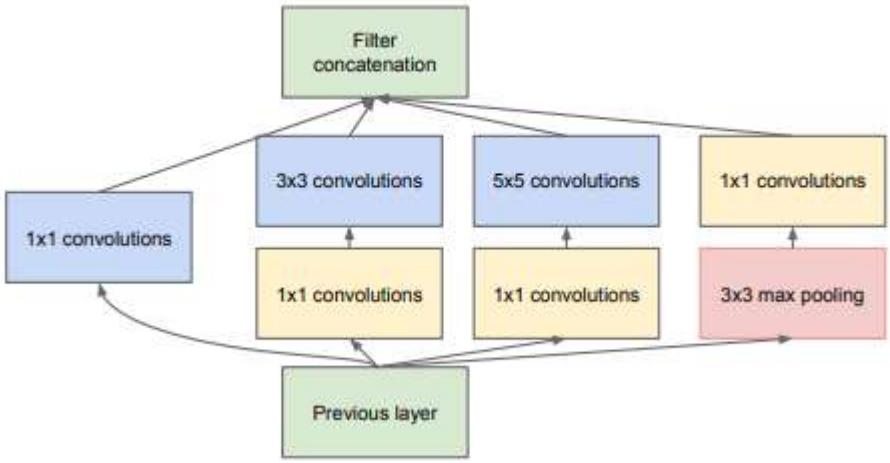
6.1 InceptionV3

Το InceptionV3 είναι ένα ευρέως γνωστό μοντέλο για την αναγνώριση εικόνων το οποίο έχει δείξει να αποσπά ακρίβεια μεγαλύτερη του 78,1% στο σύνολο δεδομένων του image-net. Το μοντέλο είναι ένα συνονθύλευμα πολλών ιδεών αναπτυγμένων από πολλούς ερευνητές ανά τα χρόνια.[6]

Το μοντέλο αυτό ανήκει στην οικογένεια των Inception δικτύων με αρκετές βελτιώσεις.

6.1.1 Inception Module

Το συγκεκριμένο δίκτυο αποτελείται από επαναλαμβανόμενα inception modules που βοηθούν στην μείωση των υπολογισμών και την αύξηση της ακρίβειας.



Εικόνα 38 Inception Module πηγή: https://www.researchgate.net/figure/nception-module-of-GoogLeNet-This-figure-is-from-the-original-paper-10_fig3_312515254

Η είσοδος του κάθε επιπέδου, είναι η έξοδος του προηγούμενου επίπεδου (κάτω μέρος της εικόνας) περνάει από ένα φίλτρο Convolution 1x1 (αριστερό μέρος της εικόνας), επίσης από ένα Convolution 1x1 και στη συνέχεια από ένα Convolution 3x3 (δεύτερη στήλη από αριστερά), επίσης από ένα Convolution 1x1 και στη

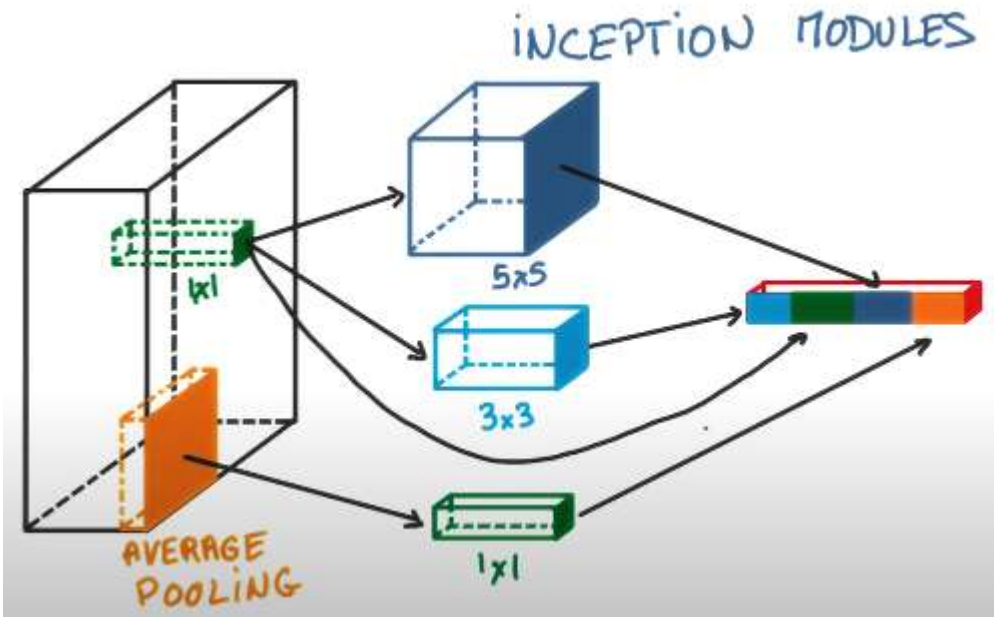
συνέχεια από ένα Convolution 5x5 (Τρίτη στήλη από αριστερά) και επίσης σε κάποια επίπεδα υπάρχει και ένα maxpooling επίπεδο 3x3 και στη συνέχεια από ένα Convolution 1x1 (δεξιά στήλη). Όλα αυτά τα αποτελέσματα αποθηκεύονται το ένα μετά το άλλο στο τελευταίο block (πάνω μέρος της εικόνας).

Ένα τέτοιο μπλοκ ονομάζεται Inception Module και το μοντέλο GoogLeNet αποτελείται από 22 επίπεδα, όπου τα περισσότερα είναι Inception Modules, Υπάρχουν φυσικά και επίπεδα Dropout Επίπεδα για την αποφυγή του Overfitting

Η τεχνική αυτή είναι πολύ πιο «φθηνή» όσο αφορά το υπολογιστικό κόστος που απαιτεί το μοντέλο καθώς επίσης και 12 φορές λιγότερες παραμέτρους από τον νικητή του ILSVRC 2013[19] Την υλοποίηση ενός δικτύου από τον Krizhevsky [9][6]

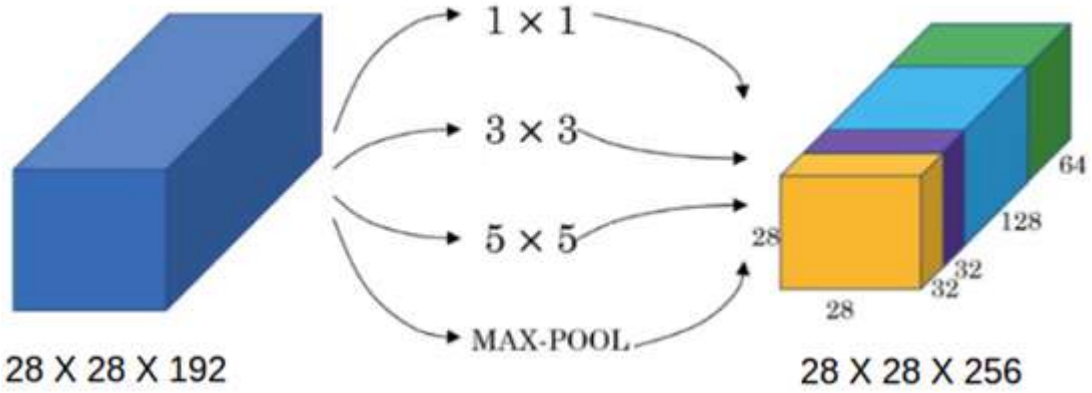
Στην εικόνα 40 βλέπουμε τον τρόπο με τον οποίο λειτουργεί το συγκεκριμένο δίκτυο.

Παίρνουμε μέσω ενός 1x1 φίλτρου τα χαρακτηριστικά της εικόνας (πράσινος κύβος αριστερά στην εικόνα) και το περνάμε από διάφορα άλλα φίλτρα (μέσω της εικόνας) ενώ τελικά παραθέτουμε τα αποτελέσματα όλων αυτών των φίλτρων στο τελικό επίπεδο (δεξί μέρος της εικόνας)



Εικόνα 39 Οπτικοποίηση του module και των διαφορετικών φίλτρων

Γιατί και πως λειτουργεί το Inception module

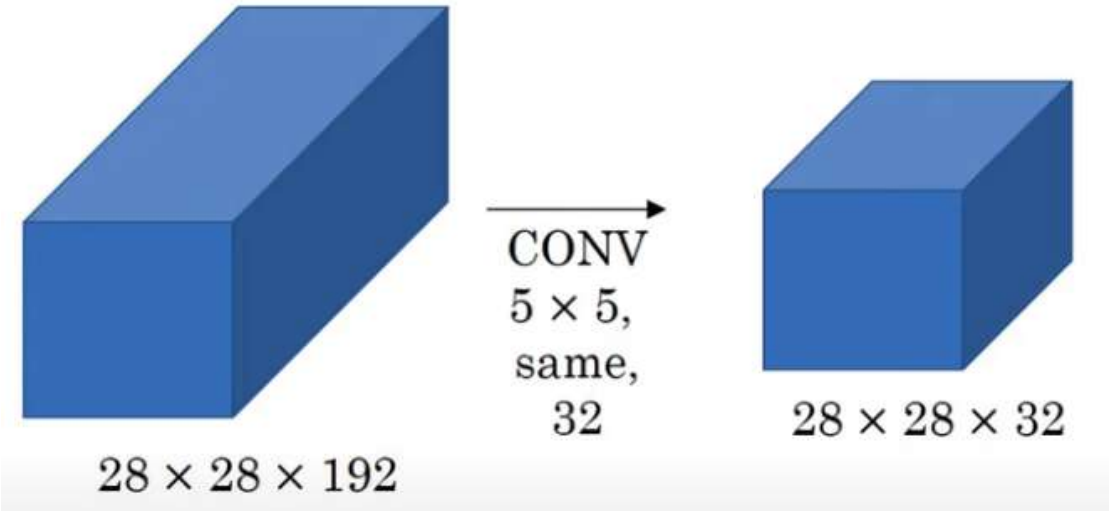


Εικόνα 40 Εφαρμογή διαφορετικών φίλτρων σε μία είσοδο

Στην εικόνα 41 έχουμε μια είσοδο 28x28x192 στην οποία εφαρμόζουμε φίλτρα (convolution layers) 64 φίλτρα διαστάσεων 1x1 , 128 φίλτρα διαστάσεων 3x3, 32 φίλτρα διαστάσεων 5x5 και ένα maxPooling χωρίς όμως να μειώσουμε τις διαστάσεις, μήκος, πλάτος. Σε αυτή την περίπτωση δεν εφαρμόζουμε το convolution 1x1 στην είσοδο μας. Παραθέτοντας τα αποτελέσματα το ένα δίπλα στο άλλο προκύπτει έξοδος 28x28x256.

Θα μελετήσουμε τώρα τον αριθμό των παραμέτρων μόνο για το 5x5 convolution επίπεδο.

6.1.2 Υπολογισμός μεταβλητών χωρίς το 1x1 convolution επίπεδο



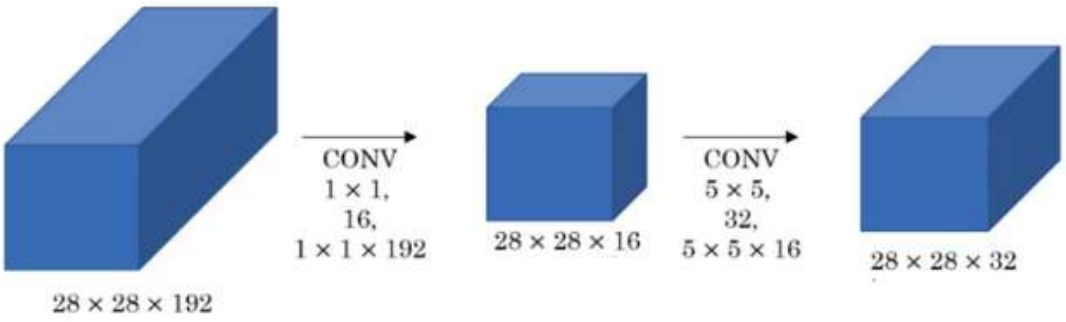
Εικόνα 41 Υπολογισμός μεταβλητών για τα 32 φίλτρα διαστάσεων 5x5

Εφόσον η τελική έξοδος θα είναι 28x28x32 τόσες θα είναι και οι παράμετροι αλλά επίσης πρέπει να λάβουμε υπόψιν και τους πολλαπλασιασμούς με τα φίλτρα οι οποίοι είναι 5x5x192

Άρα έχουμε 32 φίλτρα διαστάσεων 5x5x192 αφού τα κανάλια των φίλτρων πρέπει να είναι ίσα με αυτά της εικόνας εισόδου. Επίσης εφαρμόζουμε 32 φίλτρα (γι αυτον τον λόγο η έξοδος έχει 32 κανάλια) και κρατάμε ίσες τις διαστάσεις. Άρα 28x28x32x5x5x192 = 120.422.400 παράμετροι.

6.1.3 Υπολογισμός μεταβλητών με το 1x1 convolution επίπεδο

Εφαρμόζοντας πριν από το κάθε convolutional layer όμως ένα convolution επίπεδο 1x1 16 φίλτρων πετυχαίνουμε το εξής.



Εικόνα 42 Υπολογισμός μέσω του Inception Module

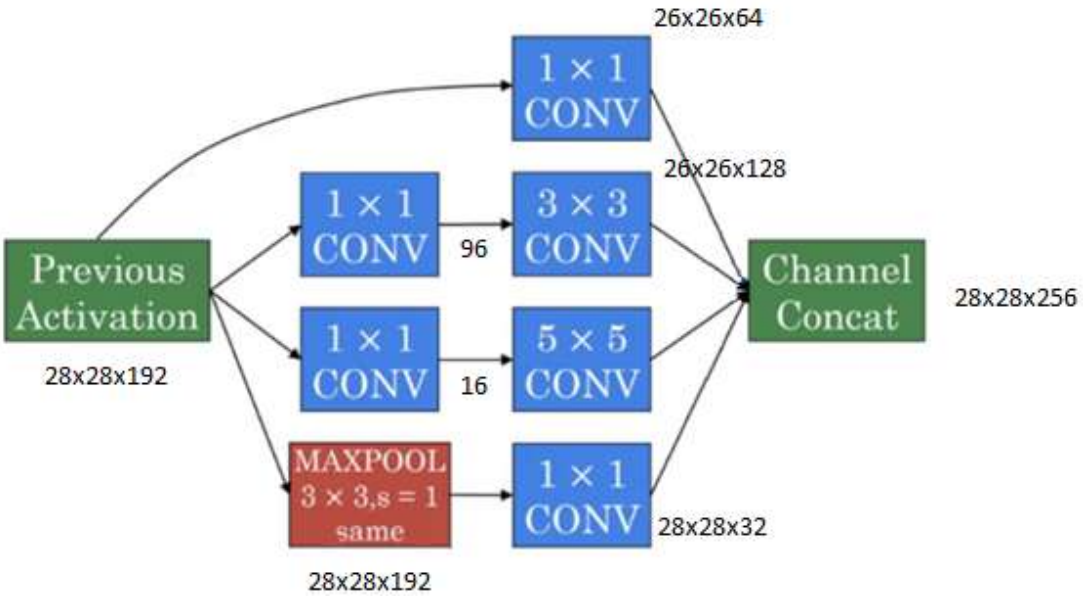
Μειώνουμε δηλαδή τις διαστάσεις μέσω του convolution layer 1x1 στα 16 κανάλια και έπειτα εφαρμόζουμε το convolution layer 5x5 32 φίλτρων στις μειωμένες διαστάσεις.

Για να υπολογίσουμε τις παραμέτρους:

28x28x16x1x1x192 = 2.408.448 παράμετροι για το πρώτο convolutional layer

28x28x32x5x5x16 = 10.035.200 παράμετροι για το δεύτερο convolutional layer

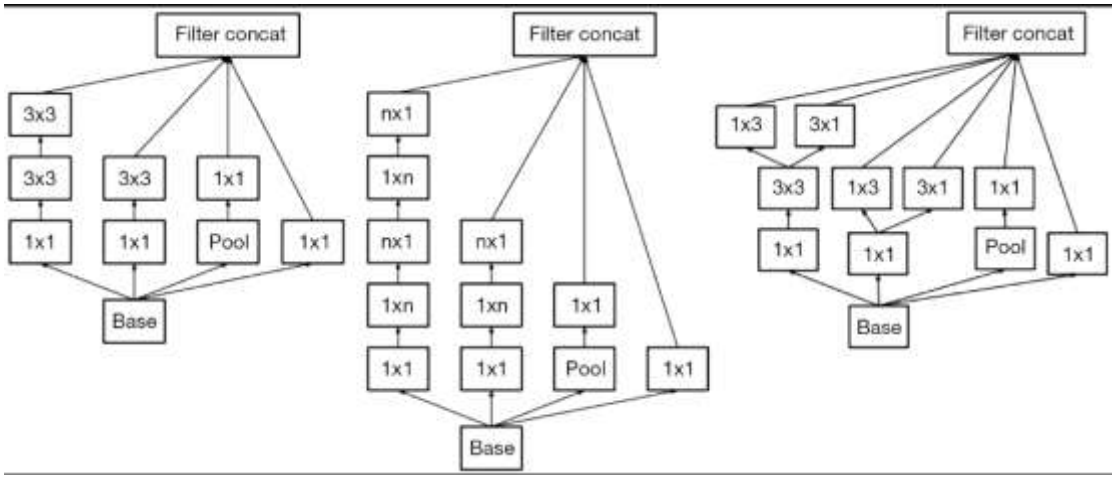
Συνολικά έχουμε 12.443.648 παραμέτρους που είναι σχεδόν το ένα δέκατο των προηγούμενων υπολογισμών[17]



Εικόνα 43 Πράξεις Inception Module

Στην εικόνα 44 βλέπουμε την είσοδο (αριστερα) να είναι η έξοδος της προηγούμενης συνάρτησης ενεργοποίησης, επείτα τροφοδοτείται στα επίπεδα που εξηγήσαμε πριν και τελικά παραθέτουμε τα αποτελέσματα στο τελευταίο επίπεδο (δεξιά)

6.1.4 Modules που χρησιμοποιήθηκαν για το Inception V3



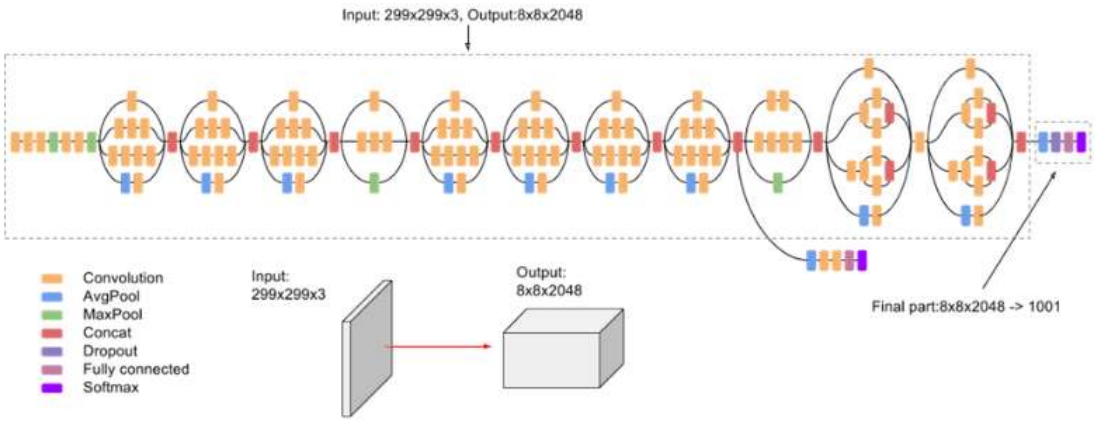
Εικόνα 44 Διαφορετικές αρχιτεκτονικές Inception Modules

Στην εικόνα 45 βρίσκουμε τα διαφορετικά module που αξιοποιεί το InceptionV3 μοντέλο

Στο πρώτο module βλέπουμε πως στη θέση του convolution layer 5x5 έχουν τοποθετηθεί στη θέση τους 2 convolution layers 3x3 και αυτό έγινε για τη μείωση του υπολογιστικού κόστους.

Επίσης στο μεσαίο module μπορούμε να παρατηρήσουμε ακόμα και τα 3x3 convolution layer έχουν αλλαχθεί σε 3x1 που ακολουθείται από ένα 1x3. Και αυτό έχει ως αποτέλεσμα τη δραματική μείωση του υπολογιστικού κόστους.

Όμοια αρχιτεκτονική έχει και το 3ο module.[6]



Εικόνα 45 Διάγραμμα υψηλού επιπέδου του μοντέλου InceptionV3

Το χαρακτηριστικό του δικτύου αυτού είναι ότι δεν αποτελείται από «στοιβαγμένα» επίπεδα συνέλιξης, MaxPooling και πλήρως συνδεδεμένα επίπεδα, Αλλά από σύνολα αυτών όπως φαίνεται στην εικόνα 38. Στην πράξη αυτό σημαίνει ότι η είσοδος περνάει από τα modules που εξηγήσαμε και αναπαρίστανται στην εικόνα 45.

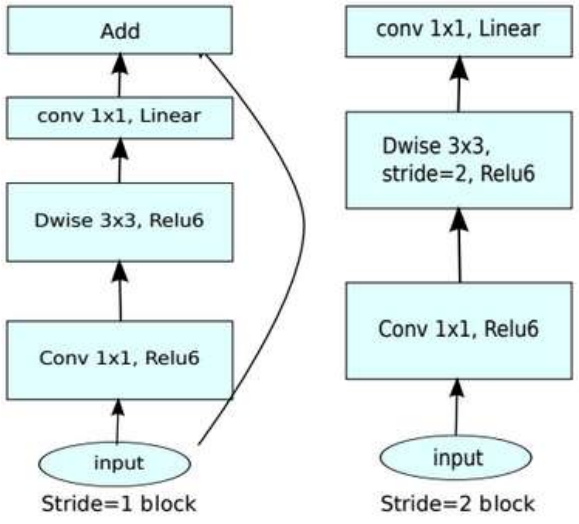
Ενώ στην εικόνα 38 φαίνεται η τελική αρχιτεκτονική του δικτύου

6.2 MobileNetV2

Το MobileNetV2 είναι ένα CNN του οποίου η αρχιτεκτονική είναι τέτοια ώστε να λειτουργεί με τον βέλτιστο τρόπο σε κινητές συσκευές ή συσκευές με περιορισμένους πόρους. Για τον λόγο αυτό προσθέτουμε ένα νέο επίπεδο. Βασίζεται σε έναν ανεστραμμένο τρόπο υπολογισμού από τα συνηθισμένα δίκτυα. Το επίπεδο αυτό ονομάζεται Inverted Residual Block. Τα επίπεδα αυτά είναι τοποθετημένα ανάμεσα σε «λεπτά» bottleneck επίπεδα.[8]

Η βάση στη οποία σχεδιάστηκε το MobileNetV2 και τα inverted residual επίπεδα είναι ότι α) οι πίνακες χαρακτηριστικών είναι δυνατόν να κωδικοποιηθούν σε μικρότερες διαστάσεις και β) οι μη γραμμικές συναρτήσεις ενεργοποίησης έχουν ως αποτέλεσμα την έλλειψη πληροφορίας παρά το γεγονός ότι αυξάνουν τον τρόπο με τον οποίο απεικονίζονται (γραφικά). Έχοντας ως βάση αυτές τις δύο αρχές, οι δημιουργοί του μοντέλου οδηγήθηκαν στη σχεδίαση ενός νέου επίπεδου συνέλιξης.

Το νέο επίπεδο αυτό παίρνει ως είσοδο ένα δείγμα χαμηλών διαστάσεων με k κανάλια και πραγματοποιεί 3 διαφορετικές συνελίξεις (3 convolutional επίπεδα). Αρχικά περνά από ένα επίπεδο 1x1 convolution το οποίο χρησιμοποιείται για να επεκτείνει τις χαμηλές διαστάσεις της εισόδου μετά χρησιμοποιείται η συνάρτηση ReLU6 (η ReLU6 είναι η συνάρτηση ενεργοποίησης ReLU *βλέπε κεφάλαιο 3.6* περιορισμένη στο διαστημα [0,6]). Στη συνέχεια χρησιμοποιούνται φίλτρα 3x3 και τα οποία ακολουθεί και πάλι η ReLU6 πετυχαίνοντας ένα φιλτράρισμα της υψηλών διαστάσεων εισόδου. Τέλος η προηγούμενη έξοδος επιστρέφει στις αρχικές διαστάσεις μέσω ενός ακόμα 1x1 convolution επιπέδου. Αυτό από μόνο του σημαίνει έλλειψη πληροφορίας, επομένως, είναι σημαντικό η συνάρτηση ενεργοποίησης στο τελευταίο βήμα να είναι γραμμική. Αφού οι πίνακες χαρακτηριστικών στην αρχή και στο τέλος έχουν τις ίδιες διαστάσεις προστίθεται μια σύνδεση για να βοηθήσει στη διόρθωση λαθών κατά την οπισθοδιάδωση σφάλματος. Μία οπτικοποίηση αυτών βρίσκεται στην εικόνα 46.

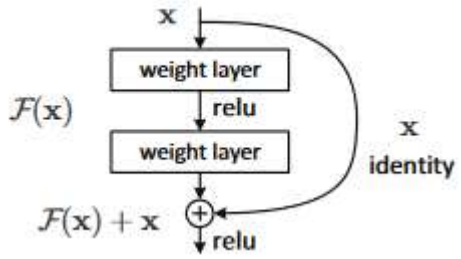


Εικόνα 46 Residual Block του MobileNetV2[8]

Αυτού του είδους ο σχεδιασμός παίρνει μια είσοδο συγκεκριμένων διαστάσεων και αυξάνει τα κανάλια της και κατ΄ επέκταση τις διαστάσεις της εισόδου. [21]

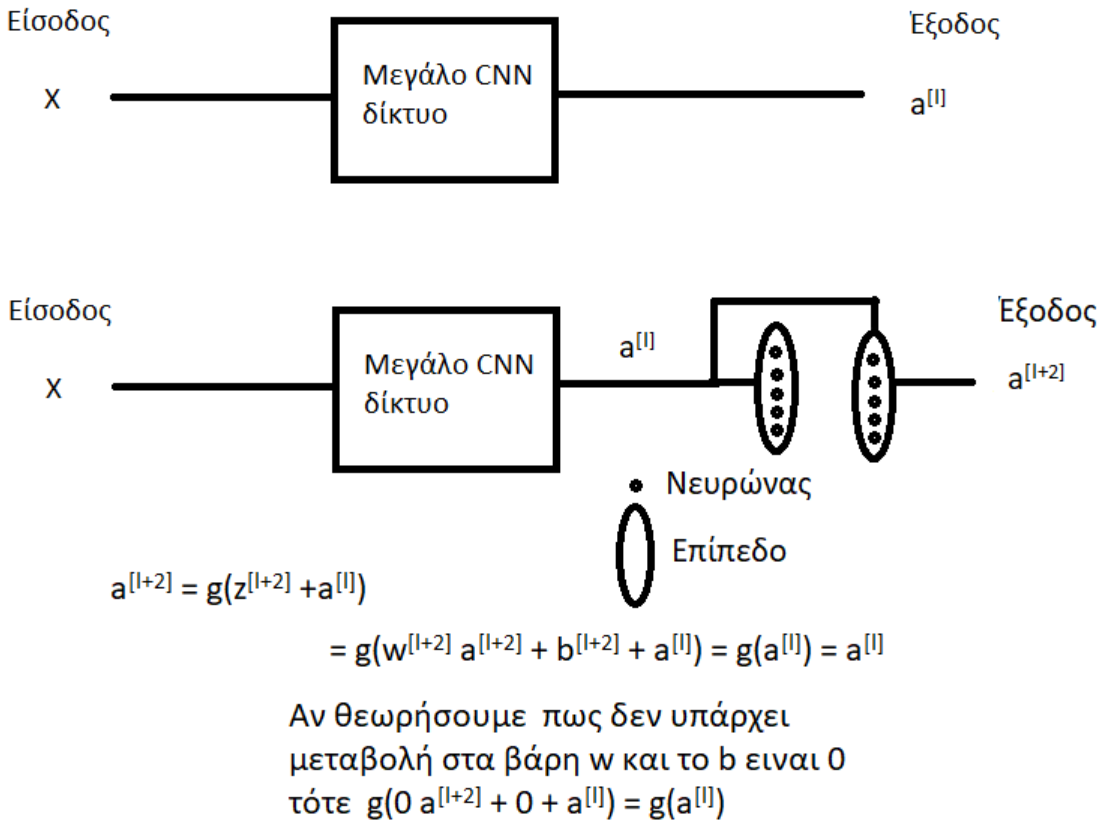
6.3 ResNetV2

Το επόμενο μοντέλο το οποίο χρησιμοποιήθηκε είναι το ResNetV2. Το ResNetV2 χρησιμοποιεί Residual Blocks. Τα Blocks αυτά παρέχουν στο μοντέλο μια «παράκαμψη» κατά την διάρκεια της εκπαίδευσης όπως φαίνεται και στην εικόνα 49. Τα δίκτυα αυτής της αρχιτεκτονικής είναι πιο εύκολο να βελτιστοποιηθούν και να αποκτήσουν ακρίβεια σε σχέση με ένα δίκτυο μεγάλου βάθους.



Εικόνα 47 Shortcut Module του ResNetV2

Στην περίπτωση της εικόνας 49, αν τα τελικά βάρη (F(x)+x) δίνουν το αποτέλεσμα που είναι ίδιο με το αρχικό (x) τότε στη διαδικασία διόρθωσης των βαρών το δίκτυο μας, παρακάμπτει το συγκεκριμένο επίπεδο (δεξιά καμπύλη).[7]



Εικόνα 48 Πως λειτουργεί η παράκαμψη (Shortcut)

Στην εικόνα 50 έχουμε στο πάνω μέρος μία είσοδο x η οποία περνάει από ένα δίκτυο CNN και μας δίνει μια έξοδο $a^{[l]}$

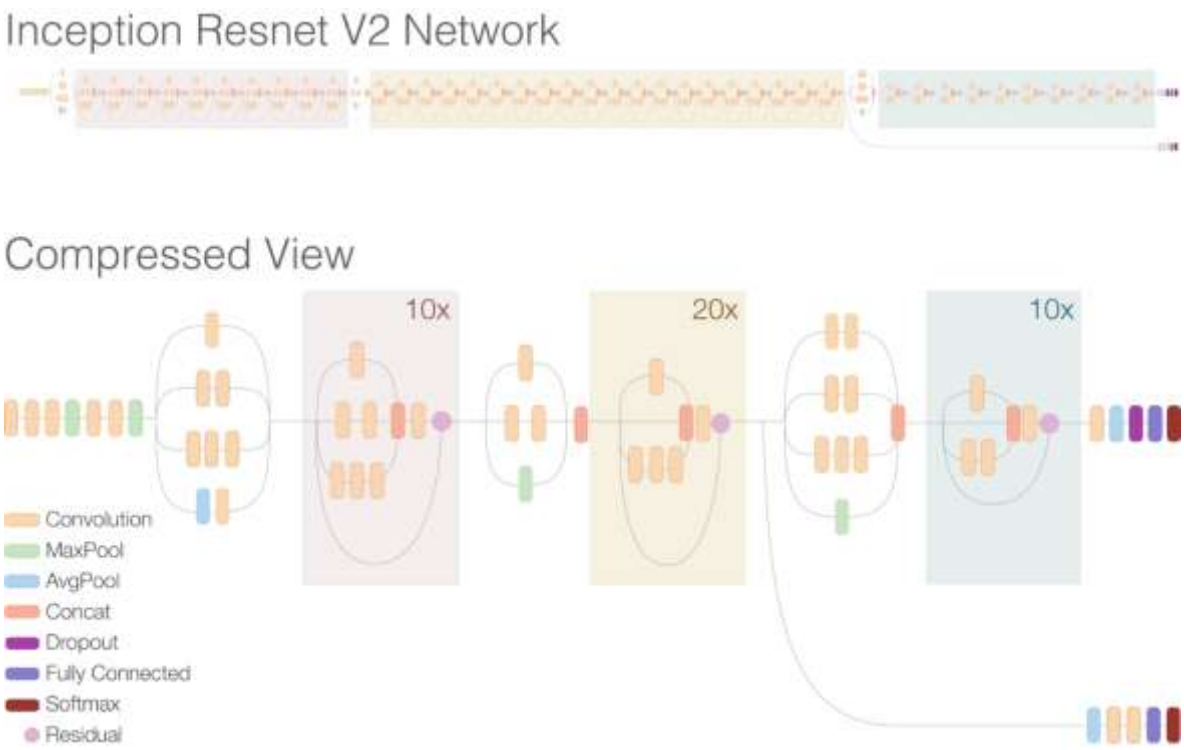
Επίσης στη μέση έχουμε την είσοδο είσοδο x που την περνάμε από ένα ίδιο βαθύ δίκτυο CNN παίρνοντας την ίδια έξοδο αλλά μετά περνάμε την έξοδο $a^{[l]}$ σε ένα residual block όπου αυτό θα μας δώσει μια έξοδο $a^{[l+2]}$ όταν τώρα χρειαστεί να διορθώσουμε τα βάρη του μοντέλου μας κατά τη διάρκεια της εκπαίδευσης θα πρέπει να κάνουμε την πράξη

$$g(w^{[l+2]} a^{[l+2]} + b^{[l+2]} + a^{[l]})$$

αν θεωρήσουμε ότι δεν υπάρχει κάποια αλλαγή στα βάρη και ότι το b είναι 0 τότε σημαίνει ότι η έξοδος του residual block είναι ακριβώς ίδια με την είσοδο του επομένως το συγκεκριμένο επίπεδο μπορεί απλά να παρακαμφθεί.

Το b (bias) είναι ένας σταθερός όρος ανεξάρτητος από τις τιμές εισόδου ο οποίος προστίθεται στον υπολογισμό της εξόδου.

Με την τεχνική-αρχιτεκτονική αυτή έχουμε τη δυνατότητα να δημιουργήσουμε δίκτυα με πάρα πολύ βάθος και σημαντική σταθερότητα. Χαρακτηριστικό παράδειγμα είναι το δίκτυο που χρησιμοποιήθηκε στον διαγωνισμό του ImageNet όπου το δίκτυο αποτελούνταν από 152 επίπεδα, πράγμα που το κάνει 8 φορές βαθύτερο από τα δίκτυα VGG.



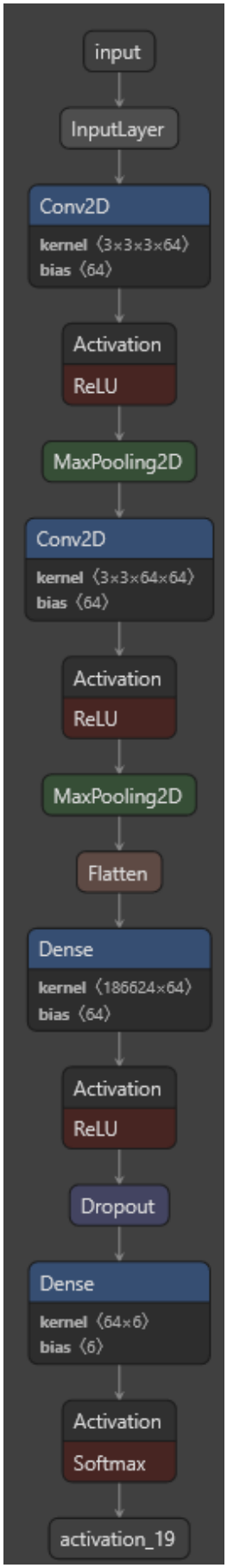
Εικόνα 49 Διάγραμμα υψηλού επιπέδου του ResNetV2 πηγή: https://ai.googleblog.com/2016/08/improving-inception-and-image.html

Στην εικόνα 51 βλέπουμε τον τελικό σχεδιασμό του μοντέλου (πάνω μέρος) ενώ στην πιο συμπιεσμένη προβολή του μοντέλου (μέση) φαίνεται ο αριθμός των επιπέδων που χρησιμοποιήθηκαν.

6.4 MotorSkillsCNN

Τέλος δημιουργήσαμε ένα δίκτυο το οποίο είναι μια υλοποίηση της κλασικής αρχιτεκτονικής των δικτύων αυτών. Αυτό σημαίνει ότι έχουμε «στοιβαγμένα» επίπεδα συνέλιξης, επιλογής χαρακτηριστικών και συναρτήσεων ενεργοποίησης.

Το δίκτυο μας αποτελείται από 1 dense layer με 64 φίλτρα μεγέθους 3x3 και δύο convolution επίπεδα.



Εικόνα 50 διάγραμμα υψηλού επιπέδου του μοντέλου MotorSkillsCNN

Για τον σχεδιασμό του μοντέλου MotorSkillsCNN χρησιμοποιήθηκε το εργαλείο netron.app του Lutz Roeder. Το netron.app είναι μία δωρεάν δικτυακή και desktop εφαρμογή οποία μας παρέχει την οπτικοποίηση του μοντέλου του οποίου βάζουμε ως είσοδο. Για να δούμε οπτικά αποτελέσματα αφού επισκεφτούμε την σελίδα netron.app επιλέγουμε το αποθηκευμένο μοντέλο μας και το φορτώνουμε στην σελίδα.

Για να αποθηκεύσουμε το μοντέλο μας πρέπει μετά την εκπαίδευση του να τρέξουμε την εντολή model.save("ModelName.model") το model είναι η μεταβλητή που περιέχει το εκπαιδευμένο μας μοντέλο ενώ το .save είναι εντολή που μας παρέχει το framework TensorFlow τέλος το ModelName είναι το όνομα με το οποίο θα αποθηκευτεί το μοντέλο μας και η κατάληξη .model είναι ο τύπος του αρχείου που θα δημιουργηθεί.

Μια αναλυτικότερη εικόνα του μοντέλου φαίνεται στον πίνακα 3 όπου στην πρώτη στήλη φαίνεται ο τύπος και το όνομα του επιπέδου, στη δεύτερη στήλη φαίνονται οι διαστάσεις της εξόδου του συγκεκριμένου φίλτρου και τέλος στην τρίτη στήλη φαίνεται ο αριθμός των παραμέτρων.

Για να καταλήξουμε την αρχιτεκτονική αυτή δοκιμάστηκαν όλοι οι πιθανοί συνδυασμοί με τα παρακάτω δεδομένα.

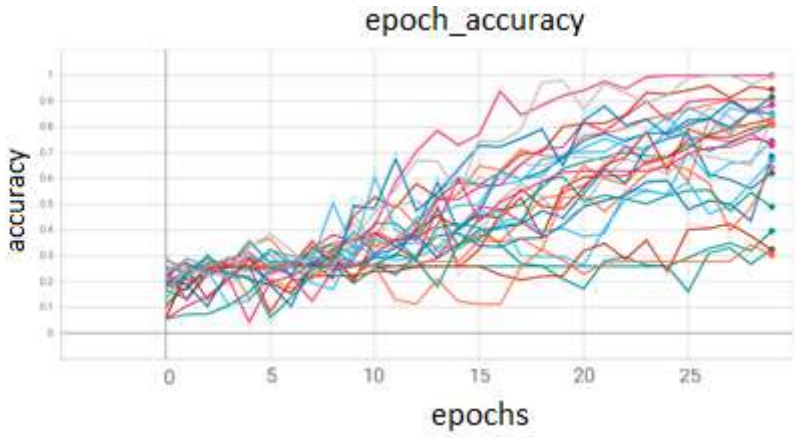
Πλήρως συνδεδεμένα επίπεδα = [1, 2, 3]

Μέγεθος Φίλτρων = [32, 64, 128]

Επίπεδα συνέλιξης (Convolutional layers) = [1, 2, 3]

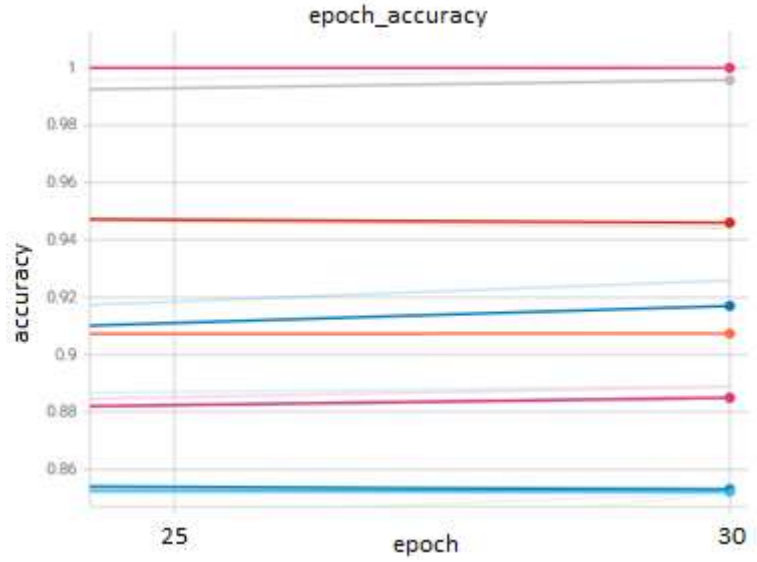
για τον κάθε συνδυασμό εκπαιδεύσαμε το μοντέλο με τα δεδομένα μας και επιλέξαμε την αρχιτεκτονική με την καλύτερη ακρίβεια και το ελάχιστο λάθος.

Το γράφημα το οποίο προέκυψε από όλες τις διαδικασίες εκπαίδευσης είναι το εξής



Γράφημα 4 Πορεία εκπαίδευσης όλων των μοντέλων

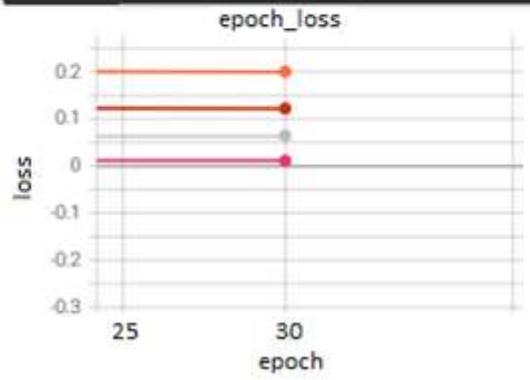
Με μια πιο προσεκτική ματιά στην κορυφή του γραφήματος 4 παρατηρούμε το εξής:



Γράφημα 5 Καλύτερες ακρίβειες μοντέλων

Και ομοίως στο διάγραμμα με το λάθος ανά epoch

Name	Smoothed	Value	Step	Time	Relative
1-conv-128-nodes-1-dense-1614248331\train	0.06476	0.0578	29	Thu Feb 25, 12:20:19	1m 22s
2-conv-64-nodes-1-dense-1614248216\train	0.01121	0.01065	29	Thu Feb 25, 12:17:52	53s
3-conv-64-nodes-2-dense-1614248870\train	0.2004	0.1956	29	Thu Feb 25, 12:28:50	56s
3-conv-64-nodes-3-dense-1614249472\train	0.1222	0.1179	29	Thu Feb 25, 12:38:52	56s



Γράφημα 6 Σφάλμα μοντέλων

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 222, 222, 64)	1792
activation (Activation)	(None, 222, 222, 64)	0
max_pooling2d (MaxPooling2D)	(None, 111, 111, 64)	0
conv2d_1 (Conv2D)	(None, 109, 109, 64)	36928
activation_1 (Activation)	(None, 109, 109, 64)	0
max_pooling2d_1 (MaxPooling2	(None, 54, 54, 64)	0
flatten (Flatten)	(None, 186624)	0
dense (Dense)	(None, 64)	11944000
activation_2 (Activation)	(None, 64)	0
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 6)	390
activation_3 (Activation)	(None, 6)	0

Total params: 11,983,110
Trainable params: 11,983,110
Non-trainable params: 0

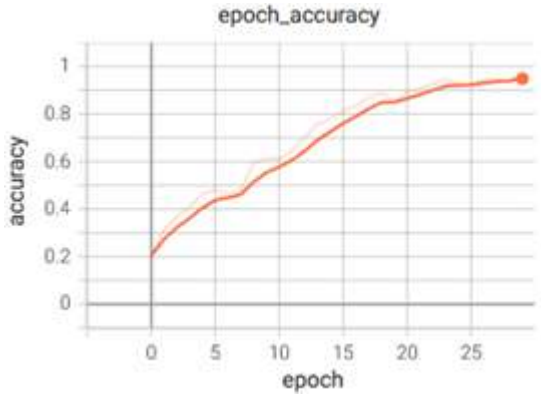
Πίνακας 3 αναλυτική περιγραφή του μοντέλου MotorSkillsCNN

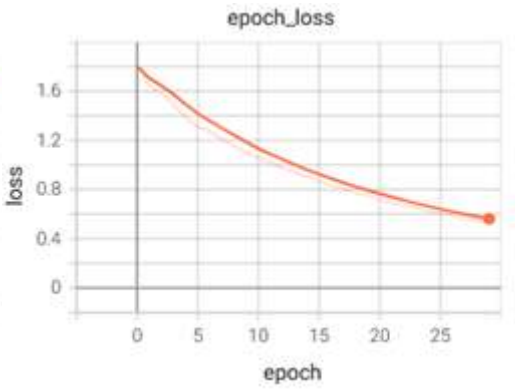
Κεφάλαιο 7. Εκπαίδευση

Κατά τη διάρκεια της εκπαίδευσης χρησιμοποιήθηκαν 54 φωτογραφίες για την εκπαίδευση των μοντέλων και κρατήθηκαν 10 για επαλήθευση. Οι 10 φωτογραφίες της επαλήθευσης δεν χρησιμοποιήθηκαν στην εκπαίδευση για να δούμε την απόδοση του μοντέλου σε φωτογραφίες οι οποίες του ήταν άγνωστες.

7.1 ResNetV2

Εδώ θα δούμε τη διαδικασία της εκπαίδευσης. Η ακρίβεια και το λάθος του ResNetV2.





Γράφημα 7. Accuracy κατά την εκπαίδευση ResNetV2

Γράφημα 8 Loss κατά την εκπαίδευση ResNetV2

Στα γραφήματα 7 και 8 βλέπουμε τη διαδικασία της εκπαίδευσης και πιο συγκεκριμένα στο γράφημα 4 φαίνεται η ακρίβεια του μοντέλου (άξονας y) και τα 30 epochs εκπαίδευσης (άξονα x) Στο γράφημα 5 απεικονίζεται το loss (άξονας x) και τα epochs εκπαίδευσης (άξονας x). Στα διαγράμματα 7 και 8 υπάρχει μία ομαλοποίηση του 20% (γραμμή με έντονο χρώμα) ενώ απεικονίζονται και τα αρχικά δεδομένα χωρίς κάποια επεξεργασία (αχνή γραμμή)

Κατά την επαλήθευση χρησιμοποιήσαμε 10 εικόνες άγνωστες για το μοντέλο. Βάλαμε το μοντέλο να προβλέψει την κλάση στην οποία ανήκουν (για εμάς γνωστή) και είδαμε πως το μοντέλο είχε ακρίβεια 30% και τον εξής πίνακα σύγχυσης.

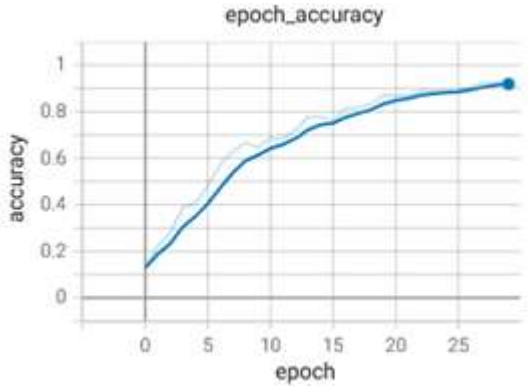
```
[0, 0, 1, 0, 0, 0]
[0, 1, 0, 0, 0, 0]
[0, 0, 1, 0, 1, 0]
[0, 0, 0, 0, 1, 1]
[0, 0, 0, 0, 1, 1]
[0, 0, 0, 1, 1, 0]
```

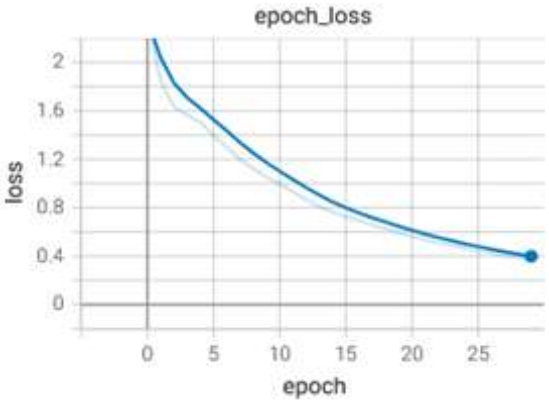
Πίνακας 4 Πίνακας σύγχησης ResNetV2

Στον πίνακα 4 (πίνακας σύγχυσης του ResNetV2) βλέπουμε έναν πίνακα 6x6 στον πίνακα αυτόν φαίνονται οι κλάσεις στις οποίες ανήκουν τα δεδομένα (σειρές) και στις κλάσεις που τα κατηγοριοποίησε το μοντέλο μας (στήλες). Η ιδανική μορφή του πίνακα θα ήταν να είναι όλα τα δεδομένα στην διαγώνιο. Τώρα όμως που δε γίνεται αυτό, βλέπουμε (πρώτη σειρά) ότι ενώ το 1 θα έπρεπε να βρίσκεται στην $1^{η}$ στήλη ($1^{η}$ κλάση) το μοντέλο μας, λανθασμένα το κατηγοριοποίησε στην $3^{η}$ στήλη ($3^{η}$ κλάση)

7.2 MobileNetV2

Υπό τις ίδιες συνθήκες εκπαίδευσης μπορούμε να δούμε την ακρίβεια και το λάθος για το MobileNetV2.





Γράφημα 9 Accuracy κατά την εκπαίδευση MobileNetV2 *Γράφημα 10 Loss κατά την εκπαίδευση MobileNetV2*

Στα γραφήματα 9 και 10 βλέπουμε τη διαδικασία της εκπαίδευσης και πιο συγκεκριμένα στο γράφημα 9 φαίνεται η ακρίβεια του μοντέλου (άξονας y) και τα 30 epochs εκπαίδευσης (άξονα x) Στο γράφημα 10 απεικονίζεται το loss (άξονας x) και τα epochs εκπαίδευσης (άξονας x). Στα διαγράμματα 9 και 10 υπάρχει μία

ομαλοποίηση του 20% (γραμμή με έντονο χρώμα) ενώ απεικονίζονται και τα αρχικά δεδομένα χωρίς κάποια επεξεργασία (αχνή γραμμή)

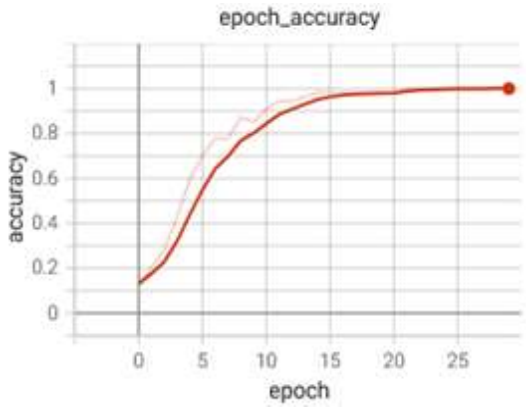
Παρά την υψηλή ακρίβεια εκπαίδευσης, στην επαλήθευση με τα άγνωστα για το μοντέλο δεδομένα έχουμε 20% ακρίβεια και τον εξής πίνακα σύγχυσης.

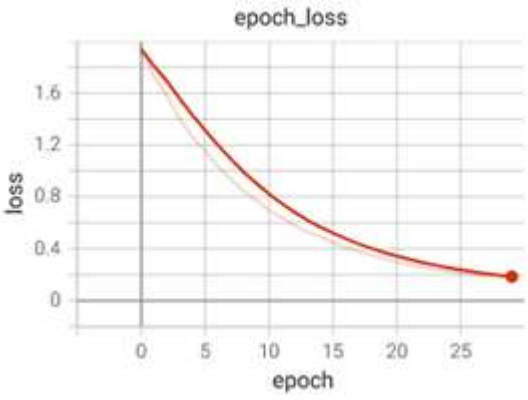
```
[1, 0, 0, 0, 0, 0]
[1, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 1, 0]
[0, 0, 0, 0, 1, 1]
[0, 0, 0, 0, 0, 2]
[0, 0, 0, 0, 2, 0]
```

Πίνακας 5 Πίνακας Συγχυσης MobileNetV2

Εδώ βλέπουμε πως το μοντέλο μας κατηγοριοποιεί σωστά το πρώτο δείγμα (στην πρώτη σειρά). Στην τρίτη σειρά επίσης βλέπουμε πως έχουμε 2 δείγματα αυτής της κλάσης και ότι το ένα έχει κατηγοριοποιηθεί σωστά στην 3^{η} στήλη ενώ το άλλο όχι.

7.3 InceptionV3





Γράφημα 11 Accuracy κατά την εκπαίδευση InceptionV3

Γράφημα 12 Loss κατά την εκπαίδευση InceptionV3

Στα γραφήματα 11 και 12 βλέπουμε τη διαδικασία της εκπαίδευσης και πιο συγκεκριμένα στο γράφημα 11 φαίνεται η ακρίβεια του μοντέλου (άξονας y) και τα 30 epochs εκπαίδευσης (άξονα x) Στο γράφημα 12 απεικονίζεται το loss (άξονας x) και τα epochs εκπαίδευσης (άξονας x). Στα διαγράμματα 11 και 12 υπάρχει μία ομαλοποίηση του 20% (γραμμή με έντονο χρώμα) ενώ απεικονίζονται και τα αρχικά δεδομένα χωρίς κάποια επεξεργασία (αχνή γραμμή)

Επίσης στο γράφημα 11 βλέπουμε την ακρίβεια του μοντέλου να φτάνει στο 100% το οποίο σημαίνει ότι το μοντέλο μας αναγνωρίζει το κάθε δείγμα και ξέρει την κλάση στην οποία πρέπει να κατηγοριοποιηθεί.

Παρόλα αυτά έχουμε ακρίβεια 50% κατά την επαλήθευση και τον εξής πίνακα σύγχυσης.

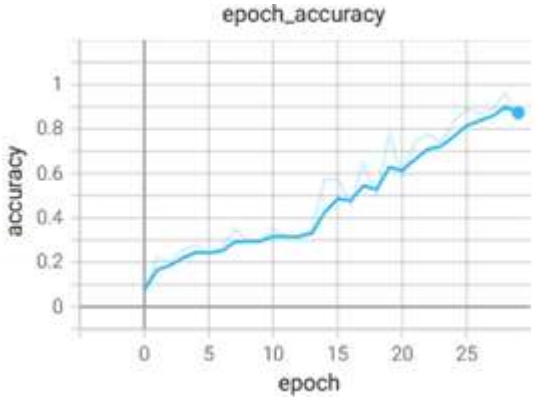
```
[1, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 1, 1]
[0, 0, 0, 0, 1, 1]
[0, 0, 0, 0, 2, 0]
[0, 0, 0, 0, 0, 2]
```

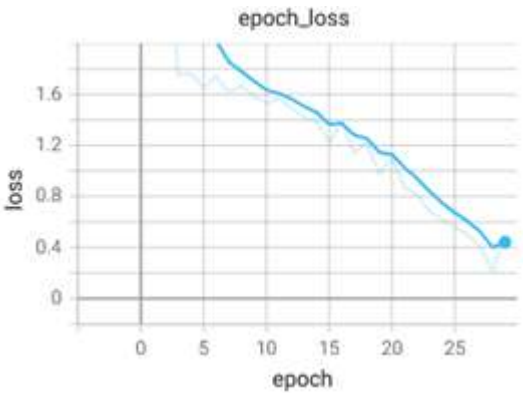
Πίνακας 6 Πίνακας σύγχυσης InceptionV3

Στον εξής πίνακα σύγχυσης βλέπουμε πως έχουν κατηγοριοποιηθεί σωστά 5 από τα 10 δείγματα (αυτά που είναι στη διαγώνιο) και τα υπόλοιπα που δεν έχουν κατηγοριοποιηθεί σωστά (σειρές 2, 3, 4) τα αναγνώρισε ως χαρακτηριστικά υψηλότερης κλάσης.

7.4 MotorSkillsCNN

Όπως μπορούμε να δούμε στο γράφημα 13 η ακρίβεια φτάνει στο 95% και το λάθος στο γράφημα 14 έχει χαμηλότερη τιμή στο 0.22





Γράφημα 13 Accuracy κατά την εκπαίδευση MotorSkillsCN *Γράφημα 14 Loss κατά την εκπαίδευση MotorSkillsCNN*

Στα γραφήματα 13 και 14 βλέπουμε τη διαδικασία της εκπαίδευσης και πιο συγκεκριμένα στο γράφημα 13 φαίνεται η ακρίβεια του μοντέλου (άξονας y) και τα

30 epochs εκπαίδευσης (άξονα x) Στο γράφημα 14 απεικονίζεται το loss (άξονας x) και τα epochs εκπαίδευσης (άξονας x). Στα διαγράμματα 13 και 14 υπάρχει μία ομαλοποίηση του 20% (γραμμή με έντονο χρώμα) ενώ απεικονίζονται και τα αρχικά δεδομένα χωρίς κάποια επεξεργασία (αχνή γραμμή)

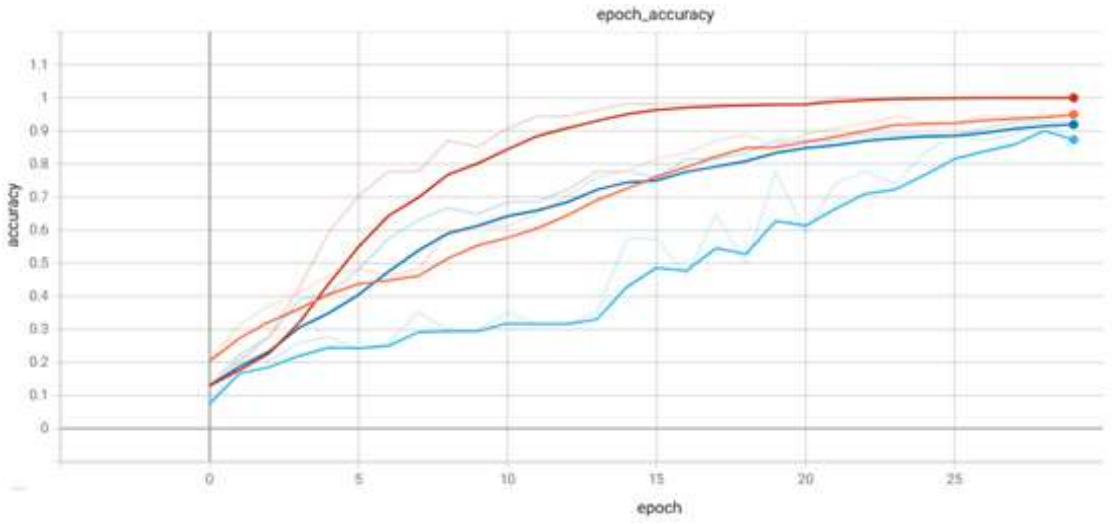
Επίσης κατά την επαλήθευση, έχουμε 40% ακρίβεια και τον ακόλουθο πίνακα σύγχυσης.

```
[0, 0, 1, 0, 0, 0]
[0, 0, 0, 0, 1, 0]
[0, 0, 2, 0, 0, 0]
[0, 0, 0, 0, 2, 0]
[0, 0, 0, 0, 2, 0]
[0, 0, 0, 0, 2, 0]
```

Πίνακας 7 Πίνακας σύγχυσης MotorSkillsCNN

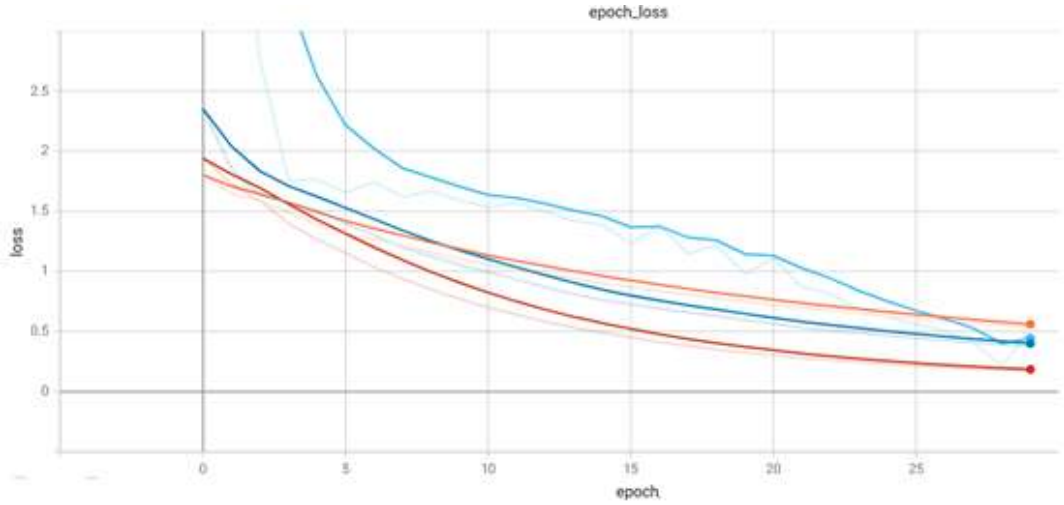
Στον δικό μας πίνακα σύγχυσης πίνακας 7 βλέπουμε να έχουν κατηγοριοποιηθεί σωστά τα 4 από τα 10 δείγματα (έχουν τοποθετηθεί στη διαγώνιο) γεγονός που μας δίνει ακρίβεια 40% κατά την επαλήθευση.

Κεφάλαιο 8. Συγκεντρωτικά αποτελέσματα



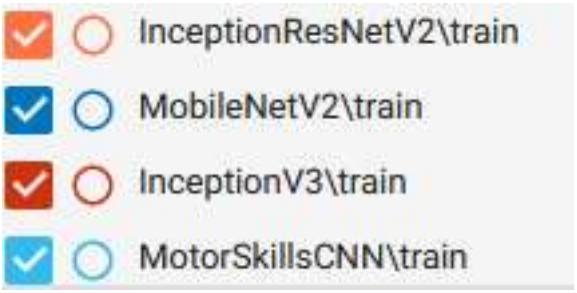
Γράφημα 15 Συγκεντρωτικά αποτελέσματα εκπαίδευσης των μοντέλων μετρώντας το Accuracy (άξονας χ) και τα epochs (άξονας y)

Στο γράφημα 15 βλέπουμε συγκεντρωτικά τα αποτελέσματα στη διαδικασία της εκπαίδευσης όλων των μοντέλων μαζί χρησιμοποιώντας τους ίδιους άξονες και την ίδια ομαλοποίηση (έντονες γραμμές)



Γράφημα 16 Συγκεντρωτικά αποτελέσματα εκπαίδευσης των μοντέλων μετρώντας το Loss (άξονας χ) και τα epochs (άξονας y)

Επίσης στο γράφημα 16 βλέπουμε την πορεία του loss κατά τη διάρκεια της εκπαίδευσης με το ίδιο ποσοστό ομαλοποίησης



Γράφημα 17 Υπόμνημα διαγραμμάτων

Τέλος εδώ είναι το υπόμνημα των γραφημάτων 16 και 17 όπου αντιστοιχείται το όνομα με το χρώμα του μοντέλου

8.1 Ανάλυση αποτελέσματων

Συγκρίνοντας τα παραπάνω αποτελέσματα προκύπτουν διάφορα συμπεράσματα. Αρχικά είναι πολύ σημαντικό να δούμε πως βάση αριθμών, το καλύτερο αποτέλεσμα ανήκει στο InceptionV3 μοντέλο. Σ’ αυτό μπορούμε να συναντήσουμε

την καλύτερη ακρίβεια με το ελάχιστο λάθος τόσο κατά την εκπαίδευση όσο και κατά την επαλήθευση. Επίσης σημαντικό είναι να δούμε πως τα μοντέλα δεν έχουν πολύ σημαντική διαφορά ως προς την ακρίβεια και το λάθος τόσο κατά την διάρκεια της εκπαίδευσης όσο και της επαλήθευσης. Αυτό σημαίνει ότι με κάθε μοντέλο θα έχουμε παρόμοια αποτελέσματα. Σε επόμενα στάδια έρευνας θα συνεχίσουμε με το δικό μας μοντέλο (MotorSkillsCNN) αφού έχοντας οι ίδιοι σχεδιάσει το μοντέλο είμαστε αρκετά πιο ευέλικτοι ως προς την χρήση καθώς και την παραμετροποίηση του. Σε επόμενο στάδιο θα μελετηθεί το πως τα διάφορα μοντέλα ανταποκρίνονται σε μεγαλύτερο εύρος δειγμάτων και το πόσο καλά μπορεί το μοντέλο μας να διαχωρίσει τα δείγματα στις κλάσεις τις οποίες όντως ανήκουν.

Στον πίνακα 8 βλέπουμε τα τελικά αποτελέσματα από τη διαδικασία εκπαίδευσης των μοντέλων. Από την διαδικασία εκπαίδευσης προκύπτει ότι την καλύτερη ακρίβεια (accuracy) την έχει το μοντέλο InceptionV3 100% και ακολουθεί το δικό μας μοντέλο MotorSkillsCNN έχοντας την ίδια ακρίβεια με το μοντέλο ResNetV2 της τάξης του 95% ενώ τελευταίο έρχεται το MobileNetV2 με ακρίβεια 92%. Κατά τη διάρκεια της εκπαίδευσης επίσης καταγράφηκε το Loss (λάθος). Και πάλι βλέπουμε ότι το μοντέλο InceptionV3 έρχεται πρώτο με το χαμηλότερο λάθος 0.168 και ακολουθεί το MobileNetV2 με 0.380 στη συνέχεια είναι το ResNetV2 με Loss 0.544 και τέλος το δικό μας μοντέλο με το μεγαλύτερο λάθος 0.568. Οι ίδιες μετρικές καταγράφηκαν και κατά τη διάρκεια της εκπαίδευσης, εκεί την καλύτερη ακρίβεια είχε και πάλι το InceptionV3 50% και ακολουθεί το MotorSkillsCNN με 40% στη συνέχεια το ResNetV2 με 30% και τέλος το MobileNetV2 με 20%. Τέλος το Loss της επαλήθευσης ακολουθεί τη σειρά που είχε και κατά την εκπαίδευση έχοντας ως πρώτο το InceptionV3 με το μικρότερο Loss 1.58, ακολουθεί το MobileNetV2 με 1.61, στη συνέχεια έχουμε το ResNetV2 με 1.64 και τέλος το MotorSkillsCNN με 2.03.

Όνομα μοντέλου	Accuracy Εκπαίδευσης	Loss Εκπαίδευσης	Accuracy Επαλήθευσης	Loss Επαλήθευσης
ResNetV2	0.95	0.544	0.3	1.64
MobileNetV2	0.92	0.380	0.2	1.61

InceptionNetV3	1	0.168	0.5	1.58
MotorSkillsCNN	0.95	0.568	0.4	2.03

Τελικά βλέπουμε πως συγκεντρωτικά την καλύτερη απόδοση την είχε το InceptionV3 μοντέλο έχοντας το μεγαλύτερο Accuracy και το μικρότερο Loss σε όλες τις φάσεις.

Πίνακας 8 Πίνακας συγκεντρωτικών αποτελεσμάτων εκπαίδευσης και επαλήθευσης

Κεφάλαιο 9. Εργαλεία που χρησιμοποιήθηκαν

Για την παρούσα έρευνα χρησιμοποιήθηκαν διάφορα εργαλεία, τόσο προγραμματιστικά όσο και hardware. Συγκεκριμένα η εκπαίδευση των μοντέλων καθώς και η προ επεξεργασία των δεδομένων έγιναν στο εξής υπολογιστικό σύστημα:

- CPU: AMD Ryzen 5 2600 Six-Core Processor 3.40 GHz
- RAM: 16,0 GB
- OS: Windows 10 Pro

Το προγραμματιστικό κομμάτι χρησιμοποίησε τη γλώσσα Python 3.8.5 με τη χρήση του προγράμματος Jupyter NoteBook. Επίσης Χρησιμοποιήθηκε η βιβλιοθήκη Tensorflow (CPU version) η οποία μας βοήθησε στην εκπαίδευση των μοντέλων. Επίσης χρησιμοποιήθηκε η βιβλιοθήκη Keras. Η βιβλιοθήκη αυτή είναι υπεύθυνη για την δημιουργία και την επεξεργασία των μοντέλων των οποίων χρησιμοποιήσαμε.

Τα μοντέλα αυτά είναι InceptionV3, MobileNetV2, ResNetV2 όπου προήλθαν από την βιβλιοθήκη Keras. Επίσης χρησιμοποιήθηκαν τα βάρη των μοντέλων από το ImageNet και δε χρησιμοποιήθηκε ο κατηγοριοποιητής του κάθε μοντέλου αλλά ένας τον οποίο κατασκευάσαμε εμείς και ανταποκρίνονταν στις απαιτήσεις μας (κατηγοριοποιούσε στις 6 επιθυμητές κλάσεις).

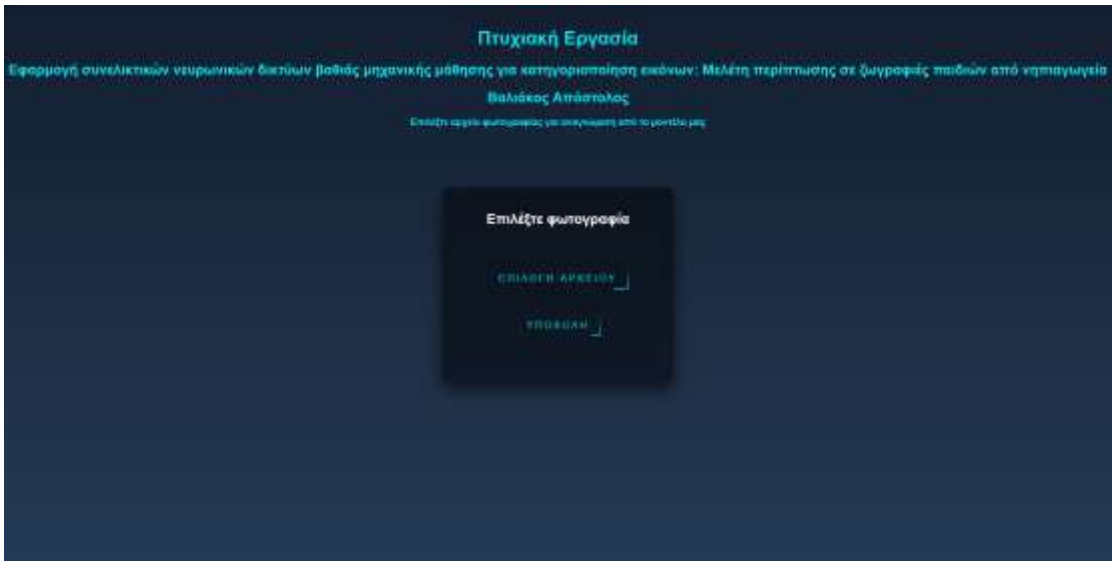
Για την οπτικοποίηση των αποτελεσμάτων χρησιμοποιήθηκε το TensorBoard το οποίο είναι ένα εργαλείο που αναπαριστά την διαδικασία εκμάθησης του εκάστοτε μοντέλου και άλλων παραμέτρων που μπορεί να επιλέξει ο χρήστης. Καθώς επίσης και μας δίνει την δυνατότητα σύγκρισης διάφορων μοντέλων όπως είδαμε στα γραφήματα 1,2,3.

Κεφάλαιο 10. Ιστοσελίδα για αυτόματη αναγνώριση φωτογραφιών

Στο πλαίσιο της παρούσας εργασίας δημιουργήθηκε και μια ιστοσελίδα στην οποία ο χρήστης μπορεί να ανεβάσει μία φωτογραφία την οποία θα επεξεργαστεί το καλύτερο μοντέλο βάση αποτελεσμάτων (Inception V3) και θα του εμφανίσει την κλάση στην οποία κατηγοριοποιείται η φωτογραφία αυτή.

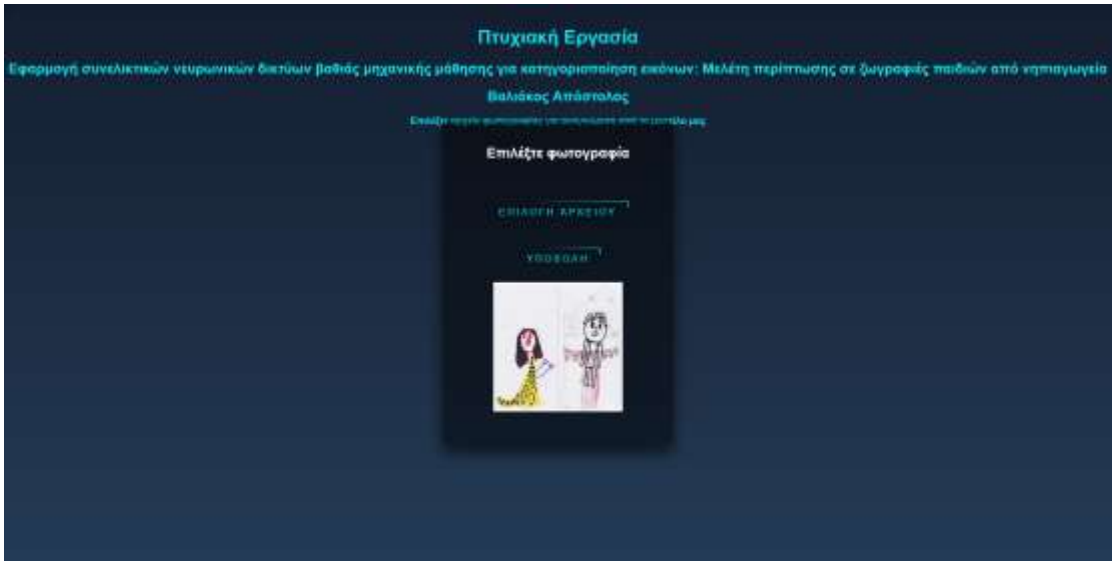
10.1 Στιγμιότυπα οθόνης από την ιστοσελίδα

Μόλις ο χρήστης επισκεφτεί την σελίδα, εμφανίζονται κάποιες βασικές πληροφορίες για τον σκοπό και τον δημιουργό της. Στην συνέχεια καλείται να επιλέξει ένα αρχείο από το αντίστοιχο κουμπί. (Εικόνα 52)



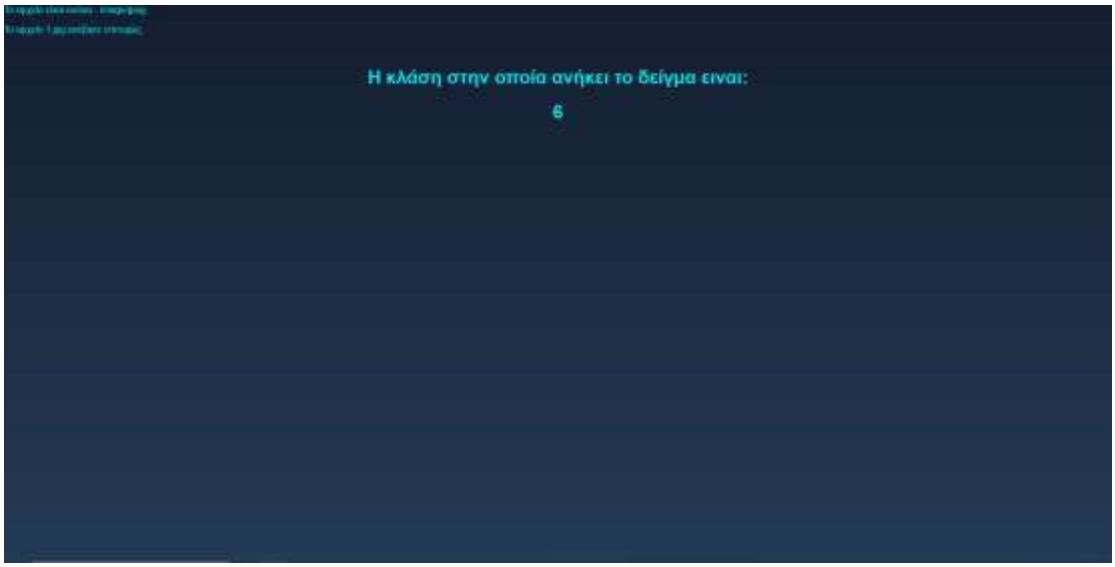
Εικόνα 51 Αρχική σελίδα

Αφού ο χρήστης επιλέξει την εικόνα που επιθυμεί τη βλέπει σε μικρογραφία κάτω από τα κουμπιά και καλείται να πατήσει το κουμπί υποβολή για να φορτωθεί η εικόνα στο μοντέλο και να γίνει η πρόβλεψη. (Εικόνα 53) ο κώδικας που υλοποιεί την συγκεκριμένη σελίδα με αναλυτικά σχόλια βρίσκεται στο παράρτημα Α.3



Εικόνα 52 Επιλογή και προεπισκόπηση δείγματος

Τέλος εμφανίζεται το αποτέλεσμα με την κλάση στην οποία έχει κατηγοριοποιηθεί το δείγμα του υπέβαλε ο χρήστης.



Εικόνα 53 Εμφάνιση αποτελέσματος

Ο κώδικας για το ανέβασμα της εικόνας και την πρόβλεψη από το μοντέλο βρίσκεται στο παράρτημα Α.4

Ενώ στο παράρτημα Α.5 βρίσκεται το Script που φορτώνει το μοντέλο και κάνει την πρόβλεψη.

Επίσης υπάρχουν τα αντίστοιχα μηνύματα λάθους στις εξής περιπτώσεις:

- Το αρχείο υπάρχει ήδη στον φάκελο
- Το αρχείο δεν είναι κάποιου αποδεκτού τύπου (Οι αποδεκτοί τύποι των αρχείων είναι JPG, JPEG, PNG & GIF)
- Έλεγχος για το μέγεθος του αρχείου

Αν όλοι αυτοί οι έλεγχοι είναι εντάξει τότε εμφανίζεται το μήνυμα με το αποτέλεσμα της κατηγοριοποίησης του δείγματος.

Εικόνα 54 Μήνυμα λάθους

Ενδεικτικό μήνυμα λάθους για την περίπτωση που το αρχείο υπάρχει ήδη

Συμπεράσματα και περαιτέρω έρευνα

Από τη συγκεκριμένη έρευνα προκύπτει ένα πλήθος συμπερασμάτων και πολλά εναύσματα για περαιτέρω έρευνα. Αρχικά μπορούμε να παρατηρήσουμε ότι τα μοντέλα έχουν διαφορές στην ακρίβεια επικύρωσης αλλά είναι πολύ κοντά στην ακρίβεια εκπαίδευσης. Αυτό μας επιτρέπει να πούμε ότι θα συνεχίσουμε την έρευνα με το μοντέλο MotorSkillsCNN με μεγαλύτερο σύνολο δεδομένων επειδή γνωρίζουμε την αρχιτεκτονική του σε μεγαλύτερο βάθος και μας είναι πιο εύκολο να εργαστούμε με αυτό. Παρ' όλα αυτά την καλύτερη ακρίβεια επικύρωσης την έχει το GoogLeNet με 50%. Αναγκαίο κρίνεται η έρευνα να συνεχιστεί με μεγαλύτερο πλήθος εικόνων για την βελτίωση του μοντέλου και την εξαγωγή ασφαλέστερων συμπερασμάτων όσον αφορά την κατηγοριοποίηση. Τα αποτελέσματα σε

μεγαλύτερο όγκο δεδομένων ίσως είναι διαφορετικά από αυτά της συγκεκριμένης εργασίας και αυτό οφείλεται στο μικρό αριθμό δεδομένων που ήταν διαθέσιμα.

Α.1 Προ επεξεργασία εικόνων

```
import numpy as np
import matplotlib.pyplot as plt
import os
import cv2
```

Εισάγουμε τις βιβλιοθήκες που χρειαστούμε για την προ – επεξεργασία των δεδομένων

```
DATADIR = "D:/ZOGRAFIES_Croped/ZOGRAFIES/train"
```

Ορίζουμε τον φάκελο με τα δεδομένα

```
CATEGORIES = ["0", "1", "2", "3", "4", "5"]
```

Ορίζουμε τα ονόματα των κλάσεων

```
for category in CATEGORIES:
```
// Για κάθε κατηγορία μπαίνουμε στον φάκελο με τις εικόνες που περιέχει

```
    path = os.path.join(DATADIR, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img), cv2.COLOR_RGB2BGR)
        plt.imshow(img_array)
        plt.show()
        break
    break
```

Δείχνουμε ενδεικτικά μία εικόνα

```
IMG_SIZE = 224
```

Ορίζουμε το μέγεθος της εικόνας

```
new_array = cv2.resize(img_array,(IMG_SIZE, IMG_SIZE))
```

```
plt.imshow(new_array)
plt.show()
```

Αναδιαμορφώνουμε το μέγεθος της εικόνας και την προβάλουμε

```
training_data = []
```

Δημιουργούμε τον πίνακα που θα φιλοξενεί τα δεδομένα μας.

```
def create_training_data():
    for category in CATEGORIES:
        path = os.path.join(DATADIR, category)
        class_num = CATEGORIES.index(category)
        for img in os.listdir(path):
            try:
                img_array                                    =
cv2.imread(os.path.join(path,img), cv2.COLOR_RGB2BGR)
                new_array                                    =
cv2.resize(img_array,(IMG_SIZE,
IMG_SIZE),interpolation=cv2.INTER_CUBIC)
                training_data.append([new_array,
class_num])
            except Exception as e:
                pass
```

Δημιουργούμε μία συνάρτηση η οποία περνά απ' όλους τους φακέλους και όλες τις κατηγορίες και αναδιαμορφώνει τις εικόνες, ενώ τελικά τις προσθέτει στον πίνακα που δημιουργήσαμε.

```
create_training_data()
```

καλούμε την συνάρτηση που δημιουργήσαμε

```
print(len(training_data))
```

εκτυπώνουμε το μέγεθος των δεδομένων για να είμαστε σίγουροι ότι είναι ίδιο με αυτό των δεδομένων στον φάκελο.

```
import random
random.shuffle(training_data)
```

ανακατεύουμε τα δεδομένα μας

```
X = []
y = []
```

δημιουργούμε 2 νέους πίνακες, ο πίνακας X θα περιέχει τα δεδομένα και ο πίνακας y περιέχει τις κλάσεις των δεδομένων.

```
for features, label in training_data:
    X.append(features)
    y.append(label)
```

για κάθε εικόνα αποθηκεύουμε τα δεδομένα και την κλάση του καθενός

```
X = np.array(X).reshape(-1, IMG_SIZE, IMG_SIZE, 3)
```

Διαμορφώνουμε τον νέο πίνακα

```
import pickle
pickle_out = open("X_train_zog_crop.pickle","wb")
pickle.dump(X, pickle_out)
pickle_out.close()
pickle_out = open("y_train_zog_crop.pickle","wb")
pickle.dump(y, pickle_out)
pickle_out.close()
```

αποθηκεύουμε τα δεδομένα σε αρχεία pickle και τα εξάγουμε.

Όμοια διαδικασία ακολουθούμε για τα δεδομένα επαλήθευσης

```
DATADIR = "D:/ZOGRAFIES_Croped/ZOGRAFIES/validation"
CATEGORIES = ["0", "1", "2", "3", "4", "5"]
for category in CATEGORIES:
    path = os.path.join(DATADIR, category)
    for img in os.listdir(path):
        img_array = cv2.imread(os.path.join(path,img),
cv2.COLOR_RGB2BGR)
        plt.imshow(img_array)
        plt.show()
        break
    break
IMG_SIZE = 224
```

```
new_array = cv2.resize(img_array,(IMG_SIZE, IMG_SIZE))
plt.imshow(new_array)
plt.show()
val_data = []
def create_validation_data():
    for category in CATEGORIES:
        path = os.path.join(DATADIR, category)
        class_num = CATEGORIES.index(category)
        for img in os.listdir(path):
            try:
                img_array                                    =
cv2.imread(os.path.join(path,img), cv2.COLOR_RGB2BGR)
                new_array                                    =
cv2.resize(img_array,(IMG_SIZE,
IMG_SIZE),interpolation=cv2.INTER_CUBIC)
                val_data.append([new_array, class_num])
            except Exception as e:
                pass
create_validation_data()
print(len(val_data))
import random
random.shuffle(val_data)
for sample in val_data[:10]:
    print(sample[1])
X_val = []
y_val = []
for features, label in val_data:
    X_val.append(features)
    y_val.append(label)
X_val  =  np.array(X_val).reshape(-1,  IMG_SIZE,  IMG_SIZE,
3)
import pickle
pickle_out = open("X_val_zog_crop.pickle","wb")
pickle.dump(X_val, pickle_out)
```

```
pickle_out.close()
pickle_out = open("y_val_zog_crop.pickle","wb")
pickle.dump(y_val, pickle_out)
pickle_out.close()
```

A.2 Επεξήγηση κώδικα για την εκπαίδευση των μοντέλων

//Imports

```
import pickle
import os
import numpy as np
import cv2
import matplotlib.pyplot as plt
import tensorflow as tf
keras = tf.keras
```

Στις πρώτες γραμμές του κώδικα, εισάγουμε τις βιβλιοθήκες τις οποίες θα χρησιμοποιήσουμε στο πρόγραμμα μας.

```
base_dir = 'drive/MyDrive/Data'
pickle_in = open("drive/MyDrive/Data/X_train_zog_crop.pickle","rb")
X_train = pickle.load(pickle_in)
pickle_in = open("drive/MyDrive/Data/y_train_zog_crop.pickle","rb")
y_train = pickle.load(pickle_in)


pickle_in = open("drive/MyDrive/Data/X_val_zog_crop.pickle","rb")
X_test = pickle.load(pickle_in)
pickle_in = open("drive/MyDrive/Data/y_val_zog_crop.pickle","rb")
y_test = pickle.load(pickle_in)
```

```
X_train=np.array(X_train/255.0)
X_test=np.array(X_test/255.0)
y_train=np.array(y_train)
y_test=np.array(y_test)
```

Στη συνέχεια φορτώνουμε τα δεδομένα μας τα οποία έχουμε προ – επεξεργαστεί και αποθηκεύουμε τα δεδομένα στις ανάλογες μεταβλητές.

```
IMG_SHAPE = (224, 224, 3)
```

Ορίζουμε τις διαστάσεις της εικόνας.

```
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE, include_top=False, weights='imagenet')
```

Φορτώνουμε το προ - εκπαιδευμένο μοντέλο ορίζοντας ότι το μέγεθος της εικόνας στην είσοδο θα είναι αυτό που ορίσαμε πιο πάνω, δε θα συμπεριλάβουμε τον τελικό κατηγοριοποιητή του μοντέλου (διαχωρίζει σε 1000 κλάσεις) και τέλος, ορίζουμε ότι τα βάρη που θα χρησιμοποιηθούν για το μοντέλο είναι αυτά που δημιουργήθηκαν στον διαγωνισμό 'imagenet'.

```
base_model.trainable = False
```

στην γραμμή αυτή δηλώνουμε ότι δε θα αλλάξουμε τα ήδη διαμορφωμένα βάρη.

```
global_average_layer = tf.keras.layers.GlobalAveragePooling2D()
```

ορίζουμε ένα global average επίπεδο και το αποθηκεύουμε στην μεταβλητή.

```
prediction_layer = keras.layers.Dense(6,activation="softmax")
```

δημιουργούμε έναν κατηγοριοποιητή που να αναγνωρίζει 6 κλάσεις και χρησιμοποιούμε ως συνάρτηση ενεργοποίησης την softmax

```
model = tf.keras.Sequential([
  base_model,
  global_average_layer,
  prediction_layer
])
```

Συνθέτουμε το τελικό μας μοντέλο και το αποθηκεύουμε στην μεταβλητη model

```
model.summary()
```

τελική επισκόπηση του μοντέλου που δημιουργήσαμε

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

κάνουμε compile το μοντελο μας ορίζοντας ως optimizer τον adam, ο τροπος με τον οποιο θα υπολογίζουμε το loss θα είναι η 'sparse_categorical_crossentropy' και τέλος κατά τη διάρκεια της εκπαίδευσης θέλουμε να βλέπουμε την ακρίβεια του μοντελου μας.

```
history = model.fit(X_train, y_train,
                    epochs=15)
```

Εδώ εκπαιδεύουμε το μοντέλο μας για 15 epochs (15 περαςσματα στα δεδομένα) τροφοδοτώντας το με τα δεδομένα που φορτώσαμε στην αρχή

```
acc = history.history['accuracy']
print(acc)
```

εκτυπώνουμε στην οθόνη την ακρίβεια του μοντέλου

```
val_loss, val_acc = model.evaluate(X_test, y_test)
print("Loss: ",val_loss)
print("Accuracy: ",val_acc)
```

υπολογίζουμε το loss και το accuracy με βάση τα δεδομένα που έχουμε ορίσει για έλεγχο.

```
from sklearn.metrics import confusion_matrix,
accuracy_score
confusion_matrix(y_test, y_train)
accuracy_score(y_test, y_train)
# Recall
from sklearn.metrics import recall_score
print("Recall: ")
recall_score(y_test1, y_pred1, average=None)
# Precision
print("Precision: ")
```

```
from sklearn.metrics import precision_score

precision_score(y_test1, y_pred1, average=None)
```

Υπολογίζουμε πίνακα σύγχησης, ακρίβεια και ευαισθησία.

Α.3 Επεξήγηση κώδικα σελίδας

```
<html>

<head>

     <meta charset="UTF-8">

     <script                                    class="jsbin"
src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery
.min.js"></script>

     <script                                    class="jsbin"
src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8.0/
jquery-ui.min.js"></script>

     <link href="style.css" rel="stylesheet">

</head>
```

Εισαγωγή απαραίτητων αρχείων JavaScript και CSS για την σωστή λειτουργία και όψη της σελίδας μας.

```
<body>

          <div class="container center">

               <h1>Πτυχιακή Εργασία</h1>

               <h2>Εφαρμογή συνελικτικών νευρωνικών
δικτύων βαθιάς μηχανικής μάθησης για κατηγοριοποίηση
εικόνων: Μελέτη περίπτωσης σε ζωγραφιές παιδιών από
νηπιαγωγεία</h2>

               <h2>Βαλιάκος Απόστολος</h2>

               <p><b>Επιλέξτε αρχείο φωτογραφίας για
αναγνώριση από το μοντέλο μας</b></p>
```

<!-- κείμενα τα οποία εμφανίζονται στην σελίδα μας ως τίτλοι -->

```
               <div class="login-box">

                    <h2>Επιλέξτε φωτογραφία</h2>

                    <form              action="upload.php"
method="post" enctype="multipart/form-data">
```

<!—Στη σειρά αυτή ορίζουμε τι θα ακολουθήσει αφου κάνουμε υποβολή την φόρμα μας (με τη μέθοδο post θα στείλουμε το αρχείο που ανέβασε ο χρήστης στο upload.php) -->

```
                        <input                type="file"
id="fileToUpload"   name="fileToUpload"   accept="image/*"
onchange="readURL(this);"  hidden >
                        <div class = 'center' >
                            <label for = "fileToUpload"
class = "final-btn ">
                                <a>
                                    <span></span>
                                    <span></span>
                                    <span></span>
                                    <span></span>
                                    Επιλογη αρχειου
                                </a>
                            </label>
                            </br>
                        </div>
                        <div>
                            <input         type="submit"
value="Upload  Image"  name="submit"  id  =  "FileToSubmit"
hidden>
                            <div class = "center">
                            <label      for      =
"FileToSubmit" class = "final-btn">
                                <a>
                                    <span></span>
                                    <span></span>
                                    <span></span>
                                    <span></span>
                                    Υποβολη
                                </a>
                                </label>
```

```
                                    </div>
                               </div>
                          </br>
                               <img id="blah" src="#" alt="" />
                          </form>
```

<!-- ορίζουμε τα κουμπιά που θα έχει η σελίδα μας και τα τοποθετούμε στα κατάλληλα div για να μας βοηθήσει στο styling-->

```
                </div>
           </div>
<script>
```

<!-- ορίζουμε ένα JavaScript script το οποίο θα εμφανίζει την προεπισκόπιση της εικόνας μας.-->

```
     function readURL(input) {
        if (input.files && input.files[0]) {
             var reader = new FileReader();

             reader.onload = function (e) {
                  $('#blah')
                       .attr('src', e.target.result)
                       .width(224)
                       .height(224);
             };
             reader.readAsDataURL(input.files[0]);
        }
   }
</script>

</body>
</html>
```

A.4 UPLOAD.PHP

```
<html>
<head>
<title>Αποτελέσματα</title>
<link href="style.css" rel="stylesheet">
</head>
<body>
<?php
$target_dir = "uploads/"; //φάκελος που αποκηθεύονται οι εικόνες που ανεβάζουν οι χρήστες
$target_file = $target_dir . basename($_FILES["fileToUpload"]["name"]); //Όνομα αρχείου που ανεβάζουμε
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));
```
//τύπος αρχείου

//Έλεγχος αν η εικόνα είναι όντως εικόνα

```
if(isset($_POST["submit"])) {
  $check = getimagesize($_FILES["fileToUpload"]["tmp_name"]);
  if($check !== false) {
    echo  " <p style='text-align: left;'> Το αρχείο είναι εικόνα - " . $check["mime"] . ".  </p> ";
    // echo "<br>";
    $uploadOk = 1;
  } else {
    echo  " <p style='text-align: center; color: red;'> Το αρχείο δεν ειναι εικόνα. </p> ";
    // echo "<br>";
    $uploadOk = 0;
  }
}
```
//Έλεγχος αν το αρχείο υπάρχει ήδη στο φάκελο

```
if (file_exists($target_file)) {
  echo " <h2 style='text-align: center; color: red;'> Το αρχείο υπάρχει ήδη. </h2>";
  // echo "<br>";
  $uploadOk = 0;
}
```

//Έλεγχος για το αν το μέγεθος αρχείου ξεπερνά το μέγιστο

```
if ($_FILES["fileToUpload"]["size"] > 500000) {
  echo " <p style='text-align: center; color: red;'> Το αρχείο είναι πολύ μεγάλο. </p>";
  // echo "<br>";
  $uploadOk = 0;
}
```

// Επιτρεπτοί τύποι αρχείων

```
if($imageFileType != "jpg" && $imageFileType != "png" && $imageFileType != "jpeg"
&& $imageFileType != "gif" ) {
  echo " <p style='text-align: center;color: red;'>Οι αποδεκτοί τύποι των αρχείων είναι JPG, JPEG, PNG & GIF. </p>";
  // echo "<br>";
  $uploadOk = 0;
}
```

// Check if $uploadOk is set to 0 by an error

```
if ($uploadOk == 0) {
  echo " <h2 style='text-align: center; color: red;'> Το αρχείο δεν ανέβηκε επιτυχώς. </h2>";
  // echo "<br>";
```

// if everything is ok, try to upload file

```
} else {
  if (move_uploaded_file($_FILES["fileToUpload"]["tmp_name"], $target_file)) {
    echo " <p style='text-align: left;'> Το αρχείο ". htmlspecialchars( basename(
```

```
$_FILES["fileToUpload"]["name"])).  "  ανέβηκε  επιτυχώς.
</p>";

$command                                                =
escapeshellcmd("F:\MiniConda\\envs\\tensorflow\\python.ex
e     C:\Users\\Απόστολος\\Desktop\\test1.py     "     .
"F:\Haxm\\Xampp\\htdocs\\sel\\uploads\\".basename($_FILES
["fileToUpload"]["name"]));
```

//Δημιουργούμε ένα windows shell μεσα στο οποίο θα τρέξουμε το script μας με την εικόνα

//Το πρώτο όρισμα είναι το σημείο στο οποίο έχουμε εγκατεστημένη την python

//Το δεύτερο είναι το σημείο που έχουμε αποθηκευμένο το Script

//Τρίτο όρισμα είναι το path της εικόνας μας.

```
$output = shell_exec($command);

    echo "</br>";

    echo " <h1 style='text-align: center;'> Η κλάση στην
οποία ανήκει το δείγμα ειναι: </h1>";

    echo  "  <h1  style='text-align:  center;'>  $output
</h1>";

  } else {

    echo "<h1 style='text-align: center;'> Sorry, there
was an error uploading your file. </h1>";

    echo "<br>";

  }

}

?>

</body>

</html>
```

A.5 Python Script

```
#!/usr/bin/env python
# coding: utf-8
import cv2
import tensorflow
import sys
import numpy as np
```

Εισάγουμε τις απαραίτητες βιβλιοθήκες

```
args = sys.argv[1]
```

ορίζουμε ότι το πρώτο όρισμα που θα υπάρχει όταν καλούμε το script από command line θα είναι αποθηκευμένο στην μεταβλητή args

```
def prepare(filename):
    IMG_SIZE = 224 //μέγεθος εικόνας
    img_array = cv2.imread(filename, cv2.COLOR_BGR2RGB)
```
//διαβασμα εικόνας
```
    img_array = img_array/255.0 //μετατροπή σε πίνακα
    new_array = cv2.resize(img_array, (IMG_SIZE,
IMG_SIZE))
```
//μετατροπή στις επιθυμητές διαστάσεις
```
    return new_array.reshape(-1, IMG_SIZE, IMG_SIZE, 3)
model = tensorflow.keras.models.load_model(r"C:\Users\Απόστολος\inception.model");
```

Φορτώνουμε το ήδη εκπαιδευμένο μοντέλο μας στην μεταβλητή model

```
response = model.predict(prepare(args));
```

προετοιμάζουμε την εικόνα και την τροφοδοτούμε στο μοντέλο μας για να κάνει την πρόβλεψη

```
CATEGORIES = ["1","2","3","4","5","6"];
```

Ορίζουμε τις κατηγορίες για να εμφανίσουμε το αποτέλεσμα στον χρήστη

```
max = response.max(1)
result = np.where(response == max)
print(CATEGORIES[int(result[1])]);
```

από τις κατανομές πιθανότητας να αντιστοιχει η εικόνα στην κάθε κλάση επιλέγουμε την υψηλότερη τιμή και την εμφανίζουμε στον χρήστη

Βιβλιογραφία

1. Feldman HM, Chaves-Gnecco D. Developmental-behavioral pediatrics. In: Zitelli BJ, McIntire SC, Nowalk AJ, eds. *Zitelli and Davis' Atlas of Pediatric Physical Diagnosis*. 7th ed. Philadelphia, PA: Elsevier; 2018:chap 3.
2. Tan, S.; Hong, C.T.; Chen, J.-H.; Chan, L.; Chi, W.-C.; Yen, C.-F.; Liao, H.-F.; Liou, T.-H.; Wu, D. Hand Fine Motor Skill Disability Correlates with Cognition in Patients with Moderate-to-Advanced Parkinson's Disease. *Brain Sci.* **2020**, *10*, 337. https://doi.org/10.3390/brainsci10060337
3. Junaid KA, Fellowes S. Gender differences in the attainment of motor skills on the Movement Assessment Battery for Children. Phys Occup Ther Pediatr. 2006;26(1-2):5-11. PMID: 16938822.
4. Caramiaux Baptiste, Françoise Jules, Liu Wanyu, Sanchez Téo, Bevilacqua Frédéric Machine Learning Approaches for Motor Learning: A Short Review. Frontiers in Computer Science 2020 https://www.frontiersin.org/article/10.3389/fcomp.2020.00016
5. Peters, Jan. (2007). Machine Learning for Motor Skills in Robotics. Künstliche Intelligenz, v.2008, 41-43 (2008).
6. Szegedy, Christian & Vanhoucke, Vincent & Ioffe, Sergey & Shlens, Jon & Wojna, ZB. (2016). Rethinking the Inception Architecture for Computer Vision. 10.1109/CVPR.2016.308.
7. K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA, 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.
8. Sandler, Mark & Howard, Andrew & Zhu, Menglong & Zhmoginov, Andrey & Chen, Liang-Chieh. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. 4510-4520. 10.1109/CVPR.2018.00474.
9. Christian Szegedy and Wei Liu and Yangqing Jia and Pierre Sermanet and Scott Reed and Dragomir Anguelov and Dumitru Erhan and Vincent Vanhoucke and Andrew Rabinovich (2015). Going Deeper with Convolutions,

Computer Vision and Pattern Recognition (CVPR), http://arxiv.org/abs/1409.4842.

10. *Hodgkin, A. L.; Huxley, A. F. (1952-08-28). "A quantitative description of membrane current and its application to conduction and excitation in nerve". The Journal of Physiology. **117** (4): 500–544. doi:10.1113/jphysiol.1952.sp004764. PMC 1392413. PMID 12991237.*
11. *Wuraola, Adedamola; Patel, Nitish (2018), "Computationally Efficient Radial Basis Function", 2018 International Conference on Neural Information Processing (ICONIP), Siem reap Cambodia: Springer, pp. 103–112, doi:10.1007/978-3-030-04179-3_9*
12. *Yerushalmy J (1947). "Statistical problems in assessing methods of medical diagnosis with special reference to x-ray techniques". Public Health Reports. **62** (2): 1432–39. doi:10.2307/4586294. JSTOR 4586294. PMID 20340527.*
13. *Powers, David M W (2011). "Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness & Correlation" (PDF). Journal of Machine Learning Technologies. **2** (1): 37–63. Archived from the original (PDF) on 2019-11-14.*
14. *"Google: RankBrain Archives". Search Engine Land. Retrieved 2020-11-03.*
15. O'Carroll, B. (2017, October 24). What are the 3 types of AI? A guide to narrow, general, and super artificial intelligence.
16. "Collection of sources defining "singularity"". singularitysymposium.com. Archived from the original on 17 April 2019. Retrieved 17 April 2019.
17. Eden, Amnon H.; Moor, James H. (2012). Singularity hypotheses: A Scientific and Philosophical Assessment. Dordrecht: Springer. pp. 1–2. ISBN 9783642325601.
18. Pulkit Sharma, December 26, 2018 : A Comprehensive Tutorial to learn Convolutional Neural Networks from Scratch https://www.analyticsvidhya.com/blog/2018/12/guide-convolutional-neural-network-cnn/
19. https://image-net.org/challenges/LSVRC/2013/

20. https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c
21. https://medium.com/@luis_gonzales/a-look-at-mobilenetv2-inverted-residuals-and-linear-bottlenecks-d49f85c12423
22. https://www.image-net.org/
23. Princeton University "About WordNet." WordNet. Princeton University. 2010