Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Ανάπτυξη Συστήματος Ταυτοποίησης Προσώπων σε Ροές Βίντεο με χρήση Συνελικτικών Νευρωνικών Δικτύων** **Development of a Face Identification System for Video Streams using Convolutional Neural Networks**
Ονοματεπώνυμο Φοιτητή	**Στυλιανός Καρανίκας**
Πατρώνυμο	**Θεόδωρος**
Αριθμός Μητρώου	**ΜΠΠΛ/ 13032**
Επιβλέπων	**Άγγελος Πικράκης, Επίκουρος Καθηγητής**

Ημερομηνία Παράδοσης **Οκτώβριος 2017**

Τριμελής Εξεταστική Επιτροπή

(υπογραφή)	(υπογραφή)	(υπογραφή)
Άγγελος Πικράκης Επίκ. Καθηγητής (Επιβλέπων)	Θεμιστοκλής Παναγιωτόπουλος Καθηγητής	Δημήτριος Αποστόλου Αναπλ. Καθηγητής

Περίληψη – Abstract

Στο πλαίσιο αυτής τη μεταπτυχιακής Διατριβής, ολοκληρώθηκε η ανάπτυξη μιας εφαρμογής αναγνώρισης και ταυτοποίησης προσώπων σε αρχεία βίντεο, με την χρήση Συνελικτικών Νευρωνικών Δικτύων μέσω της βιβλιοθήκης OpenFace. Σκοπός της Διατριβής είναι η παρουσίαση μιας υλοποίησης τεχνολογιών ανοιχτού κώδικα που συναγωνίζεται παρόμοιες εφαρμογές που χρησιμοποιούνται στην σημερινή βιομηχανία, ενώ παράλληλα θα μπορεί να χρησιμοποιηθεί από συσκευές με χαμηλές δυνατότητες επεξεργαστικής ισχύος. Η εφαρμογή παρέχει ένα ολοκληρωμένο framework για την εύρεση, προεπεξεργασία, εκπαίδευση και εκτέλεση ενός ταξινομητή προσώπων χωρίς την ανάγκη χρήσης εξωτερικών εργαλείων. Για το τελικό στάδιο της αναγνώρισης προσώπων χρησιμοποιήθηκε ο αλγόριθμος «Γραμμικών Μηχανών Διανυσματικής Στήριξης» ο οποίος συγκριτικά με άλλους παρουσίασε τα πιο αξιόπιστα αποτελέσματα, διατηρώντας ταυτόχρονα σε χαμηλά επίπεδα τους χρόνους πρόβλεψης.

The subject of the present Master Thesis is the development of a Face Recognition application for video files, implementing Convolutional Neural Networks through the use of the OpenFace library. The goal of this Thesis is to present an implementation of open source technologies that could provide comparable results to other state-of-the-art private face recognition systems, while being well-suited for mobile scenarios. The application that is presented, provides the user with a fully realized framework for the search, preprocessing, training and subsequent classification of faces, without the need for external tools. For the training stage of the classifier the Linear SVM algorithm was chosen, because it provided the best results compared to other algorithms while providing low prediction times.

Περιεχόμενα

1. Εισαγωγή - Σύντομη Περιγραφή Αντικειμένου

Δεδομένου του αυξανόμενου ενδιαφέροντος των τελευταίων ετών και της συνεχούς ανάπτυξης νέων τεχνολογιών για αναγνώριση και ταυτοποίηση προσώπων από εταιρίες, όπως η Facebook και η Google, που έχουν την δυνατότητα και την υποδομή να διαχειρίζονται μεγάλους όγκους δεδομένων προσώπων, έχει προκύψει η ανάγκη για την δημιουργία τεχνολογιών ανοιχτού κώδικα που να μπορούν να συναγωνιστούν σε απόδοση τις ήδη υπάρχουσες. Στην παρούσα διατριβή, παρουσιάζεται η ανάπτυξη μιας εφαρμογής αναγνώρισης και ταυτοποίησης προσώπων σε αρχεία βίντεο χρησιμοποιώντας την βιβλιοθήκη της OpenFace. Η OpenFace είναι μία βιβλιοθήκη ανοιχτού κώδικα και γενικού σκοπού, η οποία προσφέρει τεχνολογίες «face recognition» και προεκπαιδευμένα μοντέλα Συνελικτικών Νευρωνικών Δικτύων, για την εκπαίδευση αγνώστων dataset για χρήση σε συσκευές χαμηλής υπολογιστικής ισχύος.

Στο 1^{ο} κεφάλαιο, αναλύεται ο όρος «Μηχανική Μάθηση» (Machine Learning) και οι τομείς που τον απαρτίζουν. Παρουσιάζονται οι επιμέρους κατηγορίες στις οποίες χωρίζεται και γίνεται ανάλυση των αλγόριθμων ταξινόμησης και των εφαρμογών τους στον πραγματικό κόσμο, όπως ταξινόμηση εγγράφων, εικόνων και προσώπων. Στη συνέχεια, παρουσιάζονται οι βασικές έννοιες της Μηχανικής Μάθησης, όπως «Μείωση Διαστάσεων», «Παραμετρικά και Μη-Παραμετρικά Μοντέλα» και τέλος επεξηγείται ο τρόπος λειτουργίας των αλγόριθμων «k-κοντινότερου γείτονα» και «Γραμμικής Παλινδρόμησης».

Στο 2^{ο} κεφάλαιο, γίνεται εμβάθυνση στον τομέα του «Image Classification» (Ταξινόμηση Εικόνων) και παρουσιάζονται τα προβλήματα ταξινόμησης που είναι πιθανό να προκύψουν και ο τρόπος προσέγγισης τέτοιων προβλημάτων. Ακολούθως, παρουσιάζεται ο τρόπος λειτουργίας ενός ταξινομητή Κοντινότερου Γείτονα και γίνεται περαιτέρω ανάλυση στους δύο αλγόριθμους, «Μηχανών Διανυσματικής Στήριξης (SVM)» και «Δέντρων Αποφάσεων (Decision Tree)», με βάση τους οποίους θα γίνει η εκπαίδευση του ταξινομητή της εφαρμογής.

Στο 3^{ο} κεφάλαιο, επεξηγούνται τα Νευρωνικά Δίκτυα, ο τρόπος λειτουργίας τους και παρουσιάζονται επιγραμματικά οι βασικότερες συναρτήσεις ενεργοποίησης Νευρωνικών Δικτύων.

Στο 4^{ο} κεφάλαιο, παρουσιάζεται ο όρος «Συνελικτικά Νευρωνικά Δίκτυα», γίνεται επεξήγηση των Βασικών Εννοιών των Συνελικτικών Νευρωνικών Δικτύων, όπως είναι τα «Sparse Interactions», «Parameter Sharing» και «Equivariant Represantations». Στη συνέχεια, αναλύεται η αρχιτεκτονική ενός CNN και αναφέρονται τα πιο διαδεδομένα μοντέλα αρχιτεκτονικών που χρησιμοποιούνται σήμερα.

Στο 5^{ο} κεφάλαιο, γίνεται η παρουσίαση της βιβλιοθήκης OpenFace, της αρχιτεκτονικής του νευρωνικού δικτύου που χρησιμοποιεί, καθώς και του τρόπου λειτουργίας του για εξαγωγή των απαραίτητων features για ταξινόμηση προσώπων. Στη συνέχεια, αναλύεται ο τρόπος υλοποίησης της Openface, ο τρόπος επεξεργασίας των δεδομένων εισόδου, η μέθοδος εκπαίδευσης του νευρωνικού της δικτύου και τέλος, παρουσιάζονται συγκριτικά αποτελέσματα απόδοσης της OpenFace σε ακαδημαϊκά dataset, όπως είναι το LFW.

Στο τελευταίο κεφάλαιο, παρουσιάζεται η εφαρμογή ταυτοποίησης προσώπων που αναπτύχθηκε ως θέμα της παρούσας διατριβής, και γίνεται ανάλυση των εργαλείων που χρησιμοποιήθηκαν για την ανάπτυξη της, ο τρόπος λειτουργίας της για εύρεση και προεπεξεργασία εικόνων, και η μέθοδος εξαγωγής embeddings από τα δεδομένα του training dataset. Τέλος, γίνεται επεξήγηση του τρόπου εκτέλεσης της εφαρμογής και παρουσιάζονται συγκριτικά αποτελέσματα για την εκτέλεση της για δύο διαφορετικούς αλγόριθμους, μαζί με τα αντίστοιχα συμπεράσματα.

2. Μηχανική Μάθηση (Machine Learning)

Ο όρος μηχανική μάθηση, ή machine learning όπως είναι ευρέως γνωστός χρησιμοποιήθηκε για πρώτη φορά το 1959 από τον Arthur Samuel ο οποίος εργαζόταν στην IBM. [1] Συγκεκριμένα, ως μηχανική μάθηση ορίζεται το σύνολο των μεθόδων που χρησιμοποιούνται για την αυτόματη ανεύρεση προτύπων σε έναν συγκεκριμένο όγκο δεδομένων με απώτερο στόχο την πρόβλεψη μελλοντικών δεδομένων ή την λήψη αποφάσεων με ένα συγκεκριμένο βαθμό αβεβαιότητας, όπως π.χ. ο σχεδιασμός συλλογής αντίστοιχων συνόλων δεδομένων όπως τα αρχικά. Άλλες μορφές αβεβαιότητας μπορούν να θεωρηθούν προβλήματα όπως: Ποιες θα ήταν οι καλύτερες προβλέψεις για το μέλλον με βάση τα παρελθοντικά δεδομένα που έχουμε στην διάθεση μας; Ποιες μετρήσεις θα χρειαστεί να γίνουν στα ήδη υπάρχοντα δεδομένα; Ποιο είναι το ιδανικό μοντέλο που μπορούμε να χρησιμοποιήσουμε για την επεξήγηση των δεδομένων μας;

Η προτεινόμενη λύση για τέτοιου είδους προβλήματα είναι η εφαρμογή της θεωρίας πιθανοτήτων σε περιβάλλοντα μηχανικής μάθησης. Κάτω από τον όρο μηχανική μάθηση, κατατάσσεται μια ευρεία ποικιλία αλγόριθμων εκμάθησης και εφαρμογής μοντέλων πιθανοτήτων.

2.1 Κατηγορίες Μηχανικής Μάθησης

Οι πιο διαδεδομένες μέθοδοι μηχανικής μάθησης χωρίζονται σε δύο μεγάλες κατηγορίες:

1) **Supervised Learning (Εκμάθηση με επίβλεψη)**

2) **Unsupervised Learning (Εκμάθηση χωρίς επίβλεψη)**

Στις μεθόδους εκμάθησης με επίβλεψη ο σκοπός είναι συνήθως η ταξινόμηση του πλήθους δεδομένων εισόδου **x** σε κατηγορίες δεδομένων **y** όπου χρησιμοποιούμε μία συλλογή ζευγαριών τα οποία έχουν συγκεκριμένες **ονομασίες (labels)** της μορφής $\mathbf{D = \{(x_i,y_i)\}^N_{i=1}}$**.** Την συλλογή **D** την αποκαλούμε **συλλογή εκπαίδευσης (training set)** και ως **N** ορίζουμε τον αριθμό των **παραδειγμάτων εκπαίδευσης (training examples).**

Στην πιο απλή εφαρμογή μιας τέτοιας μεθόδου κάθε δεδομένο εισόδου της μορφής $\mathbf{x_i}$ είναι ένα **D-διάστατο** διάνυσμα αριθμών τα οποία, παραδείγματος χάριν, θα μπορούσαν να αντιπροσωπεύουν το ύψος και το βάρος ενός ατόμου. Αυτά τα δεδομένα αποκαλούνται **χαρακτηριστικά (features ή attributes).** Συνήθως σε εφαρμογές μηχανικής μάθησης, ως δεδομένα $\mathbf{x_i}$ συναντάμε πιο πολύπλοκα αντικείμενα όπως εικόνες, email, γραφήματα ή μοριακά σχήματα.

Αντίστοιχα, το είδος ενός δεδομένου εξόδου ή **μεταβλητής απόκρισης (response variable)** $\mathbf{y_i}$ συνήθως χωρίζεται σε δύο κατηγορίες. Στην πρώτη κατηγορία μπορεί να αποτελεί μια μεταβλητή **ονομαστική (nominal)** ή **κατηγορική (categorical)** η οποία θα παίρνει τιμές από μία πεπερασμένη συλλογή όπου $\mathbf{y_i \in \{1, . . . , C\}}$ όπως π.χ. αρσενικό ή θηλυκό ή θα μπορούσε να είναι μία **πραγματική αξία (real-valued scalar)** όπως π.χ. το εισόδημα ενός ατόμου.. Η μορφή των δεδομένων y καθορίζει την ονομασία του προς επίλυση προβλήματος, οπότε για ονομαστικά δεδομένα το πρόβλημα που πρέπει να επιλυθεί χαρακτηρίζεται ως **ταξινόμηση (classification) ή αναγνώριση προτύπων (pattern recognition), ενώ** σε αντίθετη περίπτωση χαρακτηρίζεται ως **οπισθοδρόμηση (regression)**.

Για την κατηγορία μεθόδων χωρίς επίβλεψη το **training set** που έχουμε αποτελείται μονάχα από δεδομένα εισόδου $\mathbf{D = \{ x_i \}^N_{i=1}}$ και ο στόχος μας είναι να βρούμε αξιοπρόσεκτα πρότυπα μέσα στα δεδομένα. Η διαδικασία αυτή αποκαλείται ορισμένες φορές ως **«ανακάλυψη γνώσης» (knowledge discovery).** Γενικότερα, αυτή η κατηγορία προβλημάτων δεν είναι τόσο καλά ορισμένη, καθώς δεν είναι γνωστά εξ' αρχής τα πρότυπα τα οποία θέλουμε να ανακαλύψουμε και ως συνέπεια δεν μπορούμε να έχουμε μία μέθοδο μέτρησης των λανθασμένων συμπερασμάτων που μπορεί να εξάγουμε. Η δυσκολία που παρουσιάζεται για την εξαγωγή σωστών συμπερασμάτων γίνεται πιο εμφανής, αν χρειαστεί να την συγκρίνουμε με

μεθόδους με επίβλεψη, όπου η πρόβλεψη για ένα δεδομένο μπορεί να συγκριθεί άμεσα με την κλάση στην οποία ανήκει.

Επιγραμματικά, υπάρχει και μία τρίτη μέθοδος μηχανικής μάθησης η οποία αποκαλείται **μάθηση μέσω ενίσχυσης-υποστήριξης (reinforcement learning)** και η χρήση της οποίας δεν είναι ιδιαίτερα διαδεδομένη. Η μέθοδος αυτή στηρίζεται σε σήματα ανταμοιβής ή τιμωρίας για την εκμάθηση μιας συγκεκριμένης συμπεριφοράς ενός αλγορίθμου. Ο σκοπός της συγκεκριμένης μεθόδου είναι η δοκιμή μεθόδων ώστε να καταλήξουμε σε αυτή η οποία παρέχει τα καλύτερα αποτελέσματα για το εκάστοτε πρόβλημα. Η κάθε δοκιμή μπορεί να μην επηρεάζει μονάχα το άμεσο αποτέλεσμα, αλλά και τις μετέπειτα καταστάσεις οι οποίες απαιτούν την βελτιστοποίηση της διαδικασίας. Επομένως, τα κύρια χαρακτηριστικά της μεθόδου είναι η **trial-and-error** προσέγγιση και η **delayed reward** όσον αφορά την βέλτιστη επιλογή. Παραδείγματα αυτής της μεθόδου μπορεί να αποτελεί αν ένα ρομπότ που συλλέγει σκουπίδια θα αποφασίσει να μπει σε ένα νέο δωμάτιο ή αν θα επιστρέψει στην βάση του για να επαναφορτιστεί. Η απόφαση αυτή θα εξαρτηθεί από την ταχύτητα και την ευκολία που είχε στο παρελθόν για να επιστρέψει στην βάση του. [2] [3] [4] [5] [6]

Στη συνέχεια θα εξετάσουμε σε μεγαλύτερο βάθος τους τρόπους λειτουργίας και τις εφαρμογές των παραπάνω αλγόριθμων εκμάθησης.

2.2 Classification Algorithms (Αλγόριθμοι Ταξινόμησης)

Η σημαντικότερη κατηγορία supervised learning και μία από τις πιο διαδεδομένες σε εφαρμογές είναι η μέθοδος **classification (ταξινόμηση)**.

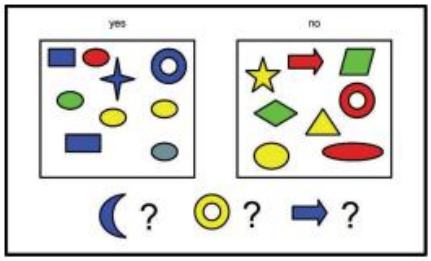
Classification ονομάζεται η διαδικασία ταξινόμησης δεδομένων σε κλάσεις οι οποίες μας είναι γνωστές από πριν, είναι δηλαδή προκαθορισμένες. Μετά από τον ορισμό ενός σετ δεδομένων με το ίδιο πλήθος χαρακτηριστικών, πραγματοποιείται αρχικά μία διαδικασία εκπαίδευσης (training) όπου ορίζει ο χρήστης ορισμένες ετικέτες (labels 1 ή -1) στα δεδομένα και με βάση αυτές γίνεται η εκπαίδευση του αλγόριθμου. Όταν έρθει εις πέρας η εκπαίδευση, δημιουργείται ένα «μοντέλο πρόβλεψης», όπου βάση αυτού θα γίνει στη συνέχεια η δοκιμή των νέων «άγνωστων» δεδομένων ώστε να κατηγοριοποιηθούν από τον αλγόριθμο σε αντίστοιχες κλάσεις. Συμπερασματικά, ο στόχος μας με την χρήση ενός τέτοιου αλγόριθμου είναι ο ορισμός ενός πολύ καλού γενικευμένου μοντέλου όπου θα μπορεί να ταξινομεί σωστά τα νέα «άγνωστα» δεδομένα.

Ας εξετάσουμε αναλυτικότερα την μέθοδο που χρησιμοποιούμε για ταξινόμηση. Πρακτικά, ο στόχος μας είναι να καθορίσουμε για κάθε δεδομένο εισόδου x μία τιμή y που του αντιστοιχεί για κάθε $y \in \{1, \ldots, C\}$, όπου C ορίζουμε τον αριθμό των κλάσεων. Σε περίπτωση που το $C = 2$ άρα το $y \in \{0, 1\}$ τότε η μέθοδος αποκαλείται **binary classification**. Σε περίπτωση όπου $C > 2$ τότε η μέθοδος αποκαλείται **multiclass classification**. Σε περίπτωση που για ένα x μπορεί να αντιστοιχούν πάνω από ένα y αποκαλούμε την μέθοδο **multi-label classification**. Εναλλακτικά, μπορούμε να προσεγγίσουμε την συγκεκριμένη κατηγορία ως διάφορες μεθόδους **binary classification** οι οποίες σχετίζονται μεταξύ τους. Η μέθοδος την οποία χρησιμοποιούμε στην εφαρμογή και στην οποία θα αναφερόμαστε ως μέθοδο ταξινόμησης στην συνέχεια είναι η multiclass classification.

Για να μπορέσουμε να ορίσουμε καλύτερα το πρόβλημα που θέλουμε να λύσουμε μέσω ταξινόμησης θα χρησιμοποιήσουμε τον όρο **προσεγγιστική συνάρτηση (function approximation)**. Υποθέτουμε λοιπόν, πως για μία άγνωστη συνάρτηση **f, y = f(x)** [9]**.** Ο στόχος μας είναι να υπολογίσουμε την συνάρτηση f με βάση ένα training set το οποίο έχει προκαθορισμένες κλάσεις για κάθε τιμή του, και με βάση αυτό το set να κάνουμε προβλέψεις χρησιμοποιώντας το $y_{\text{εκτίμησης}} = f_{\text{εκτίμησης}}(x)$ [9]. Ο κύριος σκοπός μας είναι να κάνουμε προβλέψεις με βάση άγνωστα x, σύμφωνα με την αρχική πρόβλεψη που έχουμε κάνει για την συνάρτηση εκτίμησης. Η μέθοδος αυτή αποκαλείται **γενίκευση (generilization)**.

Για την καλύτερη κατανόηση της μεθόδου γενίκευσης, παρατίθεται το παρακάτω παράδειγμα.

Εικόνα 1. Αριστερά έχουμε ορισμένα παραδείγματα (training examples) από χρωματιστά σχήματα, καθώς και τρία σχήματα το οποία δεν ανήκουν σε κάποια κλάση. Δεξιά έχουμε μία απεικόνιση των δεδομένων μας σε ένα πίνακα N x D, όπου N είναι οι περιπτώσεις και D τα χαρακτηριστικά των δεδομένων μας για κάθε x. Η στήλη Label αντιστοιχεί στις κλάσεις των δεδομένων για εύρος y ∈ {0, 1}. [9]



D features (attributes)

N cases

Color	Shape	Size (cm)	Label
Blue	Square	10	1
Red	Ellipse	2.4	1
Red	Ellipse	20.7	0

Όπως βλέπουμε στην πάνω εικόνα θέλουμε να ταξινομήσουμε τα τρία παιχνίδια που δεν αποτελούν μέρος του training set. Επομένως αυτό αποτελεί ένα πρόβλημα generalization. Μία πιθανή πρόβλεψη είναι πως το μισοφέγγαρο θα μπορούσε να έχει y = 1, καθώς στα δεδομένα του training set τα μπλε σχήματα ανήκουν στην αντίστοιχη κλάση 1. Για τα επόμενα δύο σχήματα είναι δύσκολο να πραγματοποιηθεί μια βέβαιη πρόβλεψη καθώς ορισμένα κίτρινα σχήματα παίρνουν την τιμή 1 και ορισμένα παίρνουν την τιμή 0. Η ίδια αβεβαιότητα για το αποτέλεσμα προκύπτει και για το μπλε βέλος. Για την επίλυση τέτοιων προβλημάτων αβεβαιότητας πρέπει να στραφούμε στην θεωρία πιθανοτήτων.

Σκοπός είναι να διαχωριστεί το εύρος των πιθανοτήτων για κάθε κλάση που μπορεί να έχουν τα δεδομένα x, όπως και το training set D. Επομένως έχουμε p(y|x, D). Θα θεωρήσουμε ως C το διάνυσμα το οποίο απεικονίζει το p(y|x, D). Για προβλήματα όπως αυτό που έχουμε μόνο δύο κλάσεις μπορούμε να επιστρέψουμε τον αριθμό p(y = 1|x, D) εφόσον ισχύει πως p(y = 1|x, D) + p(y =0|x, D) = 1 [9]. Επομένως για να κάνουμε την καλύτερη δυνατή εκτίμηση με βάση τα παραπάνω πρέπει να έχουμε:

$Y_{εκτίμησης} = f_{εκτίμησης}(x) = \text{argmax}^{C}_{c=1}\, p(y = c|x, D)$ (1) [9]

Η Εξ.1 αντιστοιχεί στην πιο πιθανή κλάση που μπορούν να ανήκουν τα παραπάνω αντικείμενα. Στην περίπτωση όμως αντικειμένων όπως ο κίτρινος κύκλος, που η πιθανότητα δεν βρίσκεται κοντά στην τιμή 1 και επικρατεί αβεβαιότητα όσον αφορά την απάντηση μας είναι καλύτερο να θεωρηθεί πως δεν μπορεί να προβλεφθεί η σωστή απάντηση παρά να καταταχθεί σε μία κλάση για την οποία δεν υπάρχει βεβαιότητα. [9]

2.3 Εφαρμογές προβλημάτων ταξινόμησης στον πραγματικό κόσμο

Προηγουμένως εξετάστηκε ένα θεωρητικό παράδειγμα ταξινόμησης, τέτοιου είδους προβλήματα προκύπτουν κατά κόρον και σε πραγματικές εφαρμογές. Τα κυριότερα παραδείγματα αυτών αναφέρονται παρακάτω.

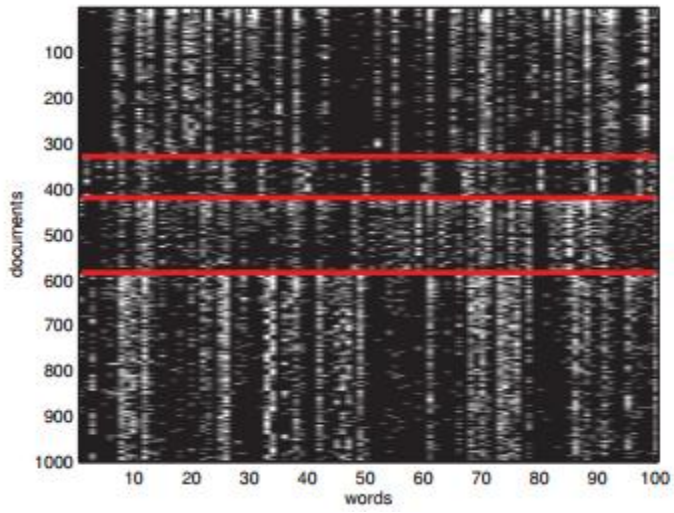
2.3.1 Ταξινόμηση εγγράφων και φιλτράρισμα κακόβουλων e-mail (spam)

Σε περιπτώσεις ταξινόμησης εγγράφων ο στόχος είναι έχοντας ως δεδομένο ένα έγγραφο, π.χ. μία ιστοσελίδα ή ένα email, να καταταχθεί σε μία από τις κλάσεις C, χρησιμοποιώντας ως x ένα κομμάτι από το κείμενο που περιέχεται στο έγγραφο. Μια υποκατηγορία αυτού του

προβλήματος είναι το φιλτράρισμα κακόβουλων email. Σκοπός είναι να χαρακτηρίσουμε το email ως κακόβουλο (y = 1) ή ως μη κακόβουλο (y = 0).

Οι περισσότεροι ταξινομητές θεωρούν πως το διάνυσμα εισόδου x είναι σταθερού μήκους. Για την απεικόνιση έγγραφων μεταβλητού μεγέθους σε μορφή διανύσματος χρησιμοποιείται η απεικόνιση **bag of words**. Η κεντρική ιδέα της μεθόδου είναι ότι θεωρείται πως $x_{ij} = 1$ μόνο στην περίπτωση όπου η λέξη j εμφανιστεί στο έγγραφο i. Αν για κάθε έγγραφο του dataset που χρησιμοποιείται εφαρμοστεί η παραπάνω αντιστοιχία θα προκύψει ως αποτέλεσμα ένα πίνακας εμφάνισης λέξης ανά έγγραφο όπως φαίνεται στην παρακάτω εικόνα:

Εικόνα 2. Η παρακάτω εικόνα απεικονίζει 1000 γραμμές που κάθε μία αποτελείται από ένα διάνυσμα bag-of-words και 100 στήλες όπου κάθε στήλη αντιστοιχεί σε μία λέξη. Οι κόκκινες γραμμές διαχωρίζουν 4 διαφορετικές κλάσεις. Ο διαχωρισμός τους σε κάθε κλάση καθορίζεται από ένα υποσύνολο λέξεων που μπορεί να εμφανίζονται ή να απουσιάζουν από το κάθε έγγραφο. [9]



Στη περίπτωση φιλτραρίσματος κακόβουλων email, το έγγραφο θα χαρακτηριστεί ως κακόβουλο εάν περιέχει λέξεις όπως: «φθηνό», «προσφορά», «αγοράστε».

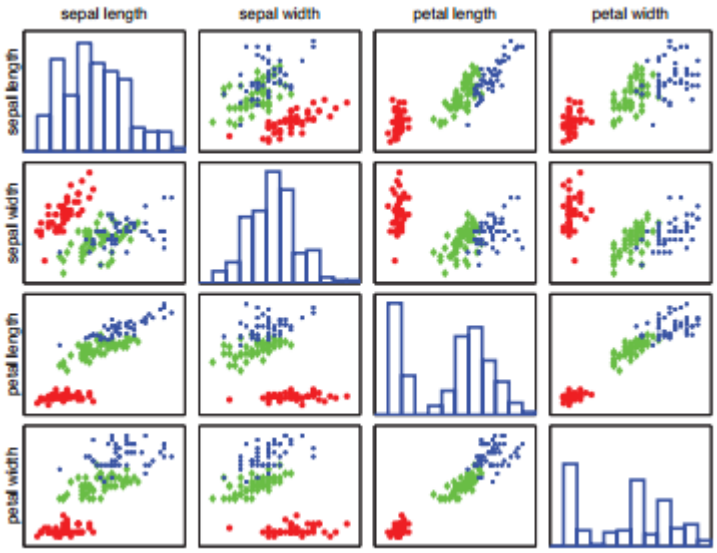
2.3.2 Ταξινόμηση λουλουδιών

Το πρόβλημα ταξινόμησης λουλουδιών προήλθε αρχικά από τον στατιστικολόγο Ronald Fisher. Ο στόχος του ήταν δυνατότητα αναγνώρισης τριών διαφορετικών ειδών ίριδας, των setosa, versicolor και virginica, όπως φαίνεται και στην παρακάτω εικόνα.

Εικόνα 3. Παρακάτω απεικονίζονται τα τρία διαφορετική είδη ίριδας. [7]

Σε συνεργασία με έναν βοτανολόγο όρισαν τέσσερα διαφορετικά είδη χαρακτηριστικών: μήκος και πλάτος ανθών, και μήκος και πλάτος πετάλων. Ο ορισμός των εξαγόμενων χαρακτηριστικών (feature extraction) αποτελεί την βάση της διαδικασίας ταξινόμησης, συνεπώς είναι ένα εξαιρετικά σημαντικό εργαλείο αλλά αντίστοιχα και πολύ δύσκολο. Πολλές μέθοδοι μηχανικής μάθησης χρησιμοποιούν χαρακτηριστικά τα οποία επιλέγονται από ανθρώπους. Αντίστοιχα, υπάρχουν και μέθοδοι μηχανικής μάθηση οι οποίες ειδικεύονται στην εξαγωγή χρήσιμων χαρακτηριστικών. Όσον αφορά το θέμα που εξετάζουμε, με την βοήθεια της απεικόνισης σε διαγράμματα, όπως φαίνεται παρακάτω, των ξεχωριστών χαρακτηριστικών της κάθε εικόνας μπορούμε να καταλήξουμε σε σαφή συμπεράσματα για τον διαχωρισμό των ειδών. Συγκεκριμένα, οι setosas μπορούν να ταξινομηθούν εξετάζοντας αν το μήκος και πλάτος των πετάλων της βρίσκεται κάτω από ένα διακριτό όριο. Οι δύο υπόλοιπες τάξεις είναι δυσκολότερα αναγνωρίσιμες και θα χρειαστεί να συνδυαστούν τα δεδομένα από τουλάχιστον δύο χαρακτηριστικά για να ληφθεί μια απόφαση. Τέλος, είναι χρήσιμο να απεικονίζονται τα δεδομένα σε διαγράμματα προτού εφαρμοστεί μία μέθοδος μηχανικής μάθησης. [9]

Εικόνα 4. Οπτικοποίηση των δεδομένων των λουλουδιών ως ζεύγη διαγραμμάτων διασποράς. Τα διαγώνια διαγράμματα περιέχουν τα οριακά ιστογράμματα των 4 χαρακτηριστικών. Τα υπόλοιπα διαγράμματα περιέχουν τα διαγράμματα διασποράς όλων των πιθανών ζευγαριών χαρακτηριστικών για τα οποία έχουμε την αντιστοιχία: κόκκινο = setosa, πράσινο = versiolor, μπλε = virginica [9]



2.3.3 Ταξινόμηση εικόνων (Image classification) και αναγνώριση γραφής (handwriting recognition)

Η ταξινόμηση εικόνων αποτελεί ένα από τα πιο ευρέως διαδεδομένα θέματα ταξινόμησης και θα αναλυθεί σε βάθος σε ξεχωριστό κεφάλαιο. Ως ταξινόμηση εικόνων θα μπορούσε να θεωρηθεί η περίπτωση όπου εξετάζεται ένα σύνολο εικόνων και θέλουμε να δούμε εάν περιέχει κάποιο ζώο ή όχι, αν πρόκειται για εικόνα εξωτερικού ή εσωτερικού χώρου, αν περιέχει κάποιο αμάξι ή όχι.

Η υποπερίπτωση που θέλουμε να εξετάσουμε είναι αν η εικόνα που αναλύεται περιέχει γράμματα ή ψηφία τα οποία έχουν γραφτεί με το χέρι όπως για παράδειγμα μια διεύθυνση ή ένας ταχυδρομικός κώδικας σε ένα γράμμα. Για τις περιπτώσεις αυτές χρησιμοποιείται ταξινόμηση αναγνώρισης γραφής. Ένα dataset το οποίο μπορεί να χρησιμοποιηθεί είναι το **MNIST,** ή αλλιώς “Modified National Institute of Standards”. [8] Στο dataset αυτό περιέχονται

10.000 εικόνες από τα ψηφία 0 έως 9 τα οποία έχουν γραφτεί με το χέρι από διάφορα άτομα. Η κάθε εικόνα είναι μεγέθους 28x28 και οι τιμές τους στην κλίμακα grayscale είναι από 0 έως 255. Στην παρακάτω εικόνα φαίνονται κάποια παραδείγματα των ψηφίων που περιέχονται στο dataset.

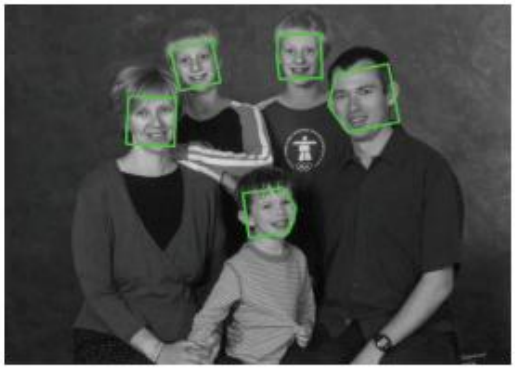
Εικόνα 5. Τα πρώτα 9 ψηφία του MNIST dataset [41]

2.3.4. Εντοπισμός εικόνων και αναγνώριση προσώπων (Face detection and recognition)

Ένα δυσκολότερο πρόβλημα σε περιπτώσεις ανάλυσης εικόνων είναι ο εντοπισμός αντικειμένων που περιέχονται σε μια εικόνα, ή αλλιώς **object detection.** Μία εξειδίκευση αυτής της περίπτωσης προβλημάτων είναι ο εντοπισμός προσώπων την οποία θα αναλύσουμε σε ξεχωριστό κεφάλαιο. Επιγραμματικά, μια προσέγγιση για την επίλυση ενός τέτοιου προβλήματος είναι ο διαχωρισμός της εικόνας σε πολλά μικρότερα κομμάτια τα οποία επικαλύπτουν το ένα το άλλο σε διαφορετικές θέσεις, κλίμακες, και κατευθύνσεις. Στην συνέχεια πραγματοποιείται η ταξινόμηση κάθε κομματιού εικόνας με βάση αν περιέχει χαρακτηριστικά που αντιστοιχούν σε μία εικόνα ή όχι. Αυτή η μέθοδος αποκαλείται **μέθοδος συρόμενου παραθύρου (sliding window detector)**. Το σύστημα στη συνέχεια επιστρέφει τις θέσεις όπου η πιθανότητα να υπάρχει πρόσωπο είναι μεγάλη. Εφαρμογές τέτοιων μεθόδων συναντώνται σε σύγχρονες ψηφιακές κάμερες όπου οι θέσεις των προσώπων βοηθούν στην εστίαση που θα κάνει η κάμερα, ή σε εφαρμογές όπως το Google Street View όπου χρησιμοποιείται για την προσθήκη φίλτρου «μωσαϊκού» στα πρόσωπα που εμφανίζονται για την προστασία της ιδιωτικότητας τους.

Εικόνα 6. Παράδειγμα οπτικοποίησης μεθόδου συρόμενου παραθύρου. Η εικόνα διαχωρίζεται σε επικαλυπτόμενα κομμάτια της εικόνας και σε κάθε ξεχωριστό κομμάτι εφαρμόζεται ο ταξινομητής. [87]

Εικόνα 7. Παράδειγμα εντοπισμού προσώπων με χρήση ταξινομητή συρόμενου παραθύρου. Στα αριστερά φαίνεται η αρχική εικόνα και στα δεξιά το αποτέλεσμα του ταξινομητή ο οποίος εντόπισε 5 πρόσωπα σε διαφορετικές θέσεις και πόζες. [9]



Εφόσον έχει ολοκληρωθεί ο εντοπισμός των προσώπων σε μια εικόνα θα πρέπει στην συνέχεια να εφαρμοστεί η αναγνώριση προσώπων, δηλαδή η ταξινόμηση κάθε προσώπου σε μία κλάση η οποία θα είναι το ονοματεπώνυμό του. Σε αυτή την περίπτωση οι πιθανές κλάσεις για αναγνώριση μπορεί να είναι πολύ μεγάλες σε αριθμό. Αντίστοιχα τα χαρακτηριστικά τα οποία μπορεί να χρησιμεύουν για την αναγνώριση, πιθανότατα θα είναι διαφορετικά από αυτά για τον εντοπισμό προσώπων. Για παράδειγμα, οι διαφορές στο κούρεμα ενός ατόμου μπορεί να είναι χρήσιμες πληροφορίες για την αναγνώριση προσώπων, αλλά όχι για τον εντοπισμό ενός προσώπου, καθώς η μέθοδος οφείλει να επικεντρώνεται στην ταξινόμηση αντικειμένων σε πρόσωπα και μη – πρόσωπα. [9]

Εικόνα 8. Παράδειγμα 25 εικόνων μεγέθους 64x64 pixel διαφορετικών ταυτοτήτων προσώπων από ένα συγκεκριμένο dataset [9]

2.4 Βασικές Έννοιες Μηχανικής Μάθησης

Στη συνέχεια θα αναλυθούν ορισμένες βασικές έννοιες που χρησιμοποιούνται στην μηχανική μάθηση και οι οποίες θα χρειαστούν στην συνέχεια κατά την ανάλυση των νευρωνικών δικτύων που χρησιμοποιούνται από την εφαρμογή.

2.4.1 Μείωση Διαστάσεων (Dimensionality Reduction)

Σε αρκετές εφαρμογές μηχανικής μάθησης και ειδικότερα σε εφαρμογές ανάλυσης εικόνων υψηλής ανάλυσης έχουμε να κάνουμε με δεδομένα υψηλών διαστάσεων (high dimensional data). Στόχος είναι να εξαχθούν όσο το δυνατόν πιο χρήσιμα χαρακτηριστικά (features) από τα δεδομένα ώστε να επεξεργάζονται αποτελεσματικά και γρήγορα.

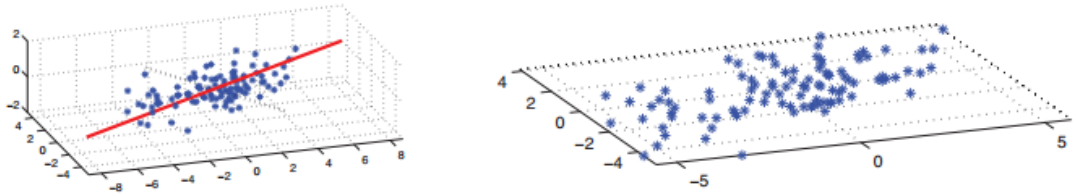
Ένας τρόπος για επιτευχθεί είναι η προβολή των δεδομένων υψηλής ανάλυσης σε μια περιοχή λιγότερων διαστάσεων. Η τεχνική αυτή λέγεται **dimensionality reduction.** Η κύρια λογική πίσω από αυτή την τεχνική είναι πως παρόλο που τα δεδομένα μπορεί εκ πρώτης όψεως να φαίνονται πως επηρεάζονται από πολυποίκιλους παράγοντες, στην πραγματικότητα η πολυμορφία τους μπορεί να είναι συνέπεια ορισμένων, αρκετά μικρότερων σε αριθμό, λανθανόντων παραγόντων. Συγκεκριμένα σε θέματα αναγνώρισης εικόνων αυτοί οι κρυμμένοι παράγοντες μπορεί να είναι η πόζα των εικόνων, ο φωτισμός τους ή η ταυτότητα των προσώπων, και όχι τόσο η αντίθεση των χρωμάτων, ή η ανάλυση των εικόνων.

Αυτή η τεχνική, όταν χρησιμοποιείται στα δεδομένα εισόδου στατιστικών μοντέλων μπορεί να αποφέρει ακριβέστερα αποτελέσματα γιατί με την προβολή των δεδομένων σε χαμηλότερη ανάλυση μπορούμε να επικεντρωθούμε στα σημαντικά στοιχεία που καθορίζουν και διαχωρίζουν τα δεδομένα το ένα από το άλλο, ενώ παράλληλα φιλτράρονται χαρακτηριστικά τα οποία μπορεί να κάνουν πιο δυσδιάκριτες αυτές τις διαφορές. Ένα ακόμα πλεονέκτημα αυτής της τεχνικής είναι πως αν επιτευχθεί η απεικόνιση των δεδομένων σε δισδιάστατα περιβάλλοντα, μπορούν στη συνέχεια να οπτικοποιηθούν με μεγάλη ευκολία σε διαγραμματικές απεικονίσεις.

Η πιο συνηθισμένη τεχνική για την μείωση διαστάσεων ονομάζεται **ανάλυση κύριων στοιχείων ή principal components analysis (PCA).** Η συγκεκριμένη τεχνική είναι αρκετά διαδεδομένη διότι πρόκειται για μία από τις απλούστερες και πιο αποτελεσματικές μεθόδους μείωσης διαστάσεων. Η λογική πίσω από την συγκεκριμένη μέθοδο είναι η εύρεση των προβολών των δεδομένων οι οποίες μεγιστοποιούν την διακύμανση τους. Το πρώτο κύριο στοιχείο (first principal component) είναι η κατεύθυνση στον χώρο κατά την οποία οι προβολές έχουν την μέγιστη διακύμανση. Το δεύτερο κύριο στοιχείο είναι αυτό του οποίου η κατεύθυνση μεγιστοποιεί την διακύμανση προς όλες τις κατευθύνσεις οι οποίες σχηματίζουν ορθή γωνία με το πρώτο κύριο στοιχείο. Επομένως, για διανύσματα **p-διαστάσεων** τα οποία θέλουμε να οριοθετήσουμε σε ένα **q-διάστατο** υπο-χώρο, ορίζουμε ως τα κύρια στοιχεία μας τις προβολές των αρχικών διανυσμάτων σε q κατευθύνσεις. Συνολικά, υπάρχουν p κύρια στοιχεία τα οποία μπορούμε να χρησιμοποιήσουμε. Ένας τρόπος για να υπολογιστεί η μέγιστη διακύμανση είναι να βρεθεί η προβολή στον υπο-χώρο όπου η μέση απόσταση των αρχικών διανυσμάτων ως προς τα κύρια στοιχεία μας είναι ελάχιστη.

Οι εφαρμογές της PCA, καθώς και άλλων μεθόδων μείωσης διαστάσεων, εφαρμόζονται σε διάφορες ερευνητικές περιοχές, όπως σε θέματα βιολογίας, όπου η PCA χρησιμοποιείται για την αποκωδικοποίηση δεδομένων των γονιδίων τα οποία είναι αλληλοσχετιζόμενα μεταξύ τους λόγω της συμπεριφοράς παρότι βρίσκονται σε διαφορετικά βιολογικά «μονοπάτια». Αντίστοιχα σε επεξεργασία φυσικής γλώσσας όπου για την ανεύρεση εγγράφων χρησιμοποιείται μία μέθοδος παραπλήσια της PCA, γνωστή ως «latent symantic analysis». Τέλος, σε θέματα γραφικών υπολογιστή χρησιμοποιώντας τα δεδομένα από καταγραφή κινήσεων (motion capturing) πραγματικών ατόμων , δημιουργούνται animations αυτών των κινήσεων. [10]

Εικόνα 9. Παράδειγμα dimensionality reduction για δεδομένα τριών διαστάσεων που στην συνέχεια προβάλλονται σε δισδιάστατο περιβάλλον. Η κόκκινη γραμμή δείχνει το 1^{ο} κύριο στοιχείο και η διακεκομμένη μαύρη διαγώνιος το 2^{ο} κύριο στοιχείο. [9]



2.4.2 Παραμετρικά και μη-παραμετρικά μοντέλα

Οι μέθοδοι που χρησιμοποιούνται για μηχανική μάθηση αποτελούνται κατά κύριο λόγο από μοντέλα πιθανοτήτων τα οποία διαχωρίζονται ανάλογα με το αν χρησιμοποιούνται για εκμάθηση με επίβλεψη ή χωρίς. Ο πιο σημαντικός διαχωρισμός αυτών των μοντέλων είναι αν πρόκειται για **παραμετρικά** ή **μη-παραμετρικά μοντέλα**.

Ως **παραμετρικά μοντέλα** ορίζονται αυτά για τα οποία έχουμε καθορισμένο αριθμό παραμέτρων, ενώ ως **μη-παραμετρικά μοντέλα** ορίζονται αυτά για τα οποία ο αριθμός των παραμέτρων αυξάνεται με το πλήθος των δεδομένων που χρησιμοποιούμε για την εκπαίδευση του μοντέλου. Το πλεονέκτημα των παραμετρικών μοντέλων είναι πως μπορούν να επεξεργαστούν δεδομένα με μεγαλύτερη ταχύτητα, αλλά έχουν ως μειονέκτημα το γεγονός πως πρέπει να γίνουν αρκετές υποθέσεις όσον αφορά την φύση των δεδομένων. Αντίθετα, τα μη-παραμετρικά μπορούν να είναι πιο ελαστικά όσον αφορά τα δεδομένα, αλλά είναι αρκετά δύσχρηστα από πλευράς των απαιτούμενων υπολογισμών που πρέπει να γίνουν καθώς απαιτούν μεγάλο αριθμό υπολογιστικών πόρων. Τα πλεονεκτήματα και τα μειονεκτήματα των δύο μοντέλων είναι τα εξής:

Παραμετρικά Μοντέλα

Πλεονεκτήματα:

1) Απλούστερη δομή
2) Ταχύτητα στην εκμάθηση από τα δεδομένα εισόδου
3) Δεν απαιτούν μεγάλο αριθμό δεδομένων για εκπαίδευση και δεν απαιτούν ακρίβεια στα χαρακτηριστικά τους.

Μειονεκτήματα:

1) Είναι περιοριστικά στον τρόπο προσέγγισης των δεδομένων
2) Δεν λειτουργούν με ακρίβεια για πολυσύνθετα προβλήματα
3) Σε πρακτικές εφαρμογές οι υποθέσεις που γίνονται για την συνάρτηση του προβλήματος σπάνια ταυτίζονται με την πραγματική συνάρτηση

Μη-παραμετρικά Μοντέλα

Πλεονεκτήματα:

1) Μεγαλύτερη ελαστικότητα για την πρόβλεψη της μορφής της συνάρτησης του προβλήματος
2) Δεν απαιτεί υποθέσεις για την εύρεση της συνάρτησης
3) Μπορεί να καταλήξει σε πιο αποδοτικά μοντέλα προβλέψεων

Μειονεκτήματα:

1) Απαιτεί μεγάλο πλήθος δεδομένων εισόδου
2) Η διαδικασία για την εκπαίδευση του αλγόριθμου είναι αρκετά χρονοβόρα
3) Ενέχει κινδύνους overfitting των δεδομένων προς εκπαίδευση

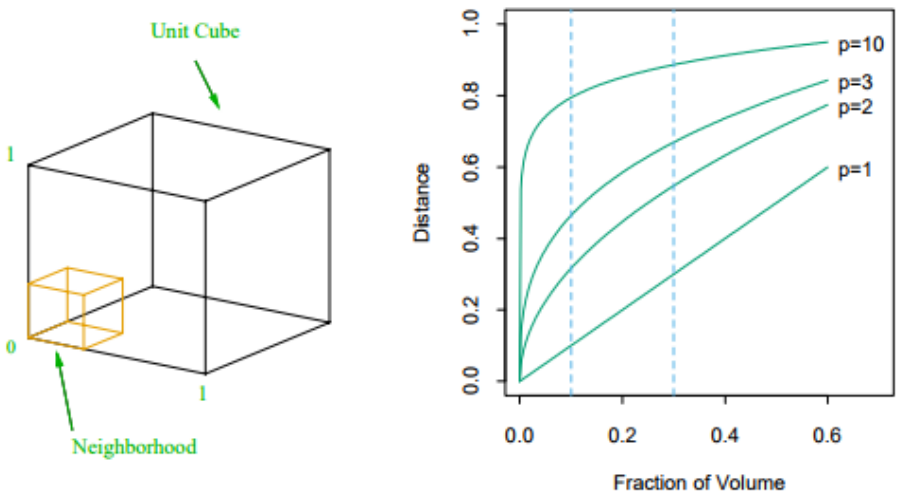
Στη συνέχεια θα δούμε επιγραμματικά έναν αλγόριθμο από κάθε κατηγορία για να δούμε πως εφαρμόζονται τα παραπάνω στοιχεία του καθενός. [10]

2.4.3. Αλγόριθμος K-κοντινότερου γείτονα (K-nearest neighbors)

Ο αλγόριθμος K-κοντινότερου γείτονα χρησιμοποιείται κυρίως σε θέματα classification και είναι μη-παραμετρικός. Η λογική του αλγόριθμου είναι πως αν παρατηρηθούν ορισμένα σημεία K του training set τα οποία είναι τα κοντινότερα στο x δεδομένο εισόδου, και διαπιστωθεί σε ποιες κλάσεις ανήκουν, μπορεί να προβλεφθεί και η κλάση στην οποία ανήκει το x. Αυτή η μέθοδος εκμάθησης αποκαλείται διαφορετικά και ως **memory-based learning** ή **instance-based learning**. Η πιο απλή μέθοδος για να υπολογιστούν οι αποστάσεις μεταξύ των σημείων είναι η Ευκλείδεια μετρική συνάρτηση. Ο συγκεκριμένος αλγόριθμος είναι αρκετά απλός ώστε να μπορεί να είναι εύκολα εφαρμόσιμος και προσφέρει ακριβή αποτελέσματα αν έχει μεγάλο πλήθος ταξινομημένων δεδομένων. [11] Το πρόβλημα που παρατηρείται με την συγκεκριμένη μέθοδο είναι πως σε θέματα υψηλών διαστάσεων δεν μπορεί να αποδώσει με ακρίβεια σαφή αποτελέσματα. Αυτό το πρόβλημα είναι γνωστό και ως **«Curse of Dimensionality»**.

Για να γίνει κατανοητό το μέγεθος του προβλήματος, ας θεωρηθεί πως θα εφαρμοστεί ο συγκεκριμένος αλγόριθμος σε δεδομένα τα οποία έχουν ίση διασπορά μέσα σε ένα κύβο p διαστάσεων. Αν θέλουμε να υπολογίσουμε τους κοντινότερους γείτονες του σημείου x θα πρέπει να σχηματίσουμε ένα υπέρ-κυβο γύρω από το σημείο x μέχρι να περιλαμβάνει ένα κλάσμα r από όλα τα σημεία. Εφόσον το r απεικονίζει ένα κλάσμα του συνολικού όγκου του κύβου το μήκος των άκρων αντιστοιχεί σε $e_p(r) = r^{1/p}$. Σε περίπτωση όπου ο κύβος είναι 10-διάστατος και θέλουμε να βασιστούμε σε ποσοστό 10% των δεδομένων μας έχουμε $e_{10}(0.1) = 0.80$, επομένως πρέπει να μεγαλώσουμε τον κύβο κατά 80% προς κάθε διάσταση γύρω από το x για να υπολογίσουμε την κλάση του. Αυτό έχει ως αποτέλεσμα να μην ασχολούμαστε για δεδομένα τα οποία είναι κοντά το ένα στο άλλο όπως φαίνεται και από την παρακάτω εικόνα. [12]

Εικόνα 10. Στην αριστερή εικόνα έχουμε τον κύβο που θέλουμε να χρησιμοποιήσουμε για να υπολογίσουμε τους κοντινότερους γείτονες του p-διάστατου κύβου που περιέχει όλα τα δεδομένα. Στο διπλανό διάγραμμα παρατηρούμε ποιο πρέπει να είναι το μέγεθος των άκρων του κύβου για να καλύψουμε ένα κλάσμα του αρχικού όγκου του κύβου για διάφορα p. Όπως φαίνεται για p = 10, πρέπει να καλύψουμε το 80% του μήκους για να υπολογίσουμε το 10% από τα δεδομένα μας. [12]



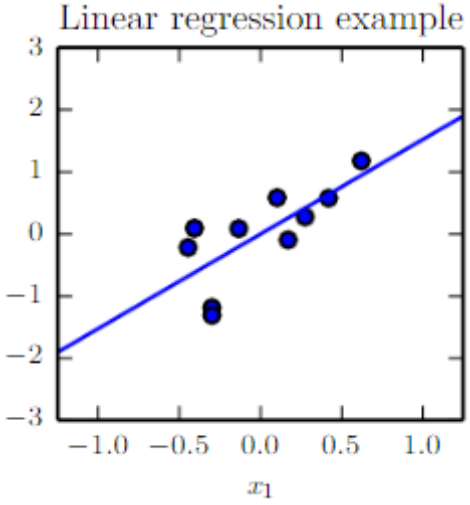
2.4.4 Αλγόριθμος Γραμμικής Παλινδρόμησης (Linear Regression)

Για αποφευχθούν θέματα για δεδομένα υψηλών διαστάσεων, χρησιμοποιούνται παραμετρικές μέθοδοι, καθώς βασίζονται στην δημιουργία υποθέσεων για την φύση των στοιχείων που έχουμε στην διάθεση μας. Ένας από τους βασικούς αλγόριθμους τέτοιων μοντέλων είναι ο αλγόριθμος της γραμμικής παλινδρόμησης. Η αρχική υπόθεση στην οποία βασίζεται σε μία γραμμική συνάρτηση των δεδομένων εισόδου μας και επομένως εκφράζεται ως εξής:

$y = w^T x + b$, όπου $w \in R^n$ και είναι ένα διάνυσμα παραμέτρων [91]

Ως παραμέτρους ορίζονται οι τιμές οι οποίες ελέγχουν την συμπεριφορά του συστήματος. Επομένως μπορούν να θεωρηθούν τα w ως συντελεστές βαρύτητας τα οποία καθορίζουν κατά πόσο τα χαρακτηριστικά των δεδομένων καθορίζουν την πρόβλεψη. Αν ο συντελεστής βαρύτητας είναι θετικός τότε αυξάνεται η τιμή της πρόβλεψης y, ενώ αν είναι αρνητικός μειώνεται. Αντίστοιχα, αν είναι μηδενικός ο συντελεστής δεν επηρεάζει καθόλου την πρόβλεψη, ενώ αν είναι πολύ μεγάλος την διαμορφώνει σε σημαντικό βαθμό. Ως b ορίζεται την κλίση της συνάρτησης. [13]

Εικόνα 11. Στην παρακάτω εικόνα βλέπουμε ένα παράδειγμα linear regression όπου έχουμε 10 σημεία x από το training set το οποίο περιέχει από ένα χαρακτηριστικό. Επειδή υπάρχει μόνο ένα χαρακτηριστικό υπάρχει μόνο ένα βάρος w το οποίο το επηρεάζει. Όπως παρατηρούμε ο αλγόριθμος εκπαιδεύεται ώστε να ορίζει στο w έτσι ώστε η ευθεία y = wx να διαπερνά όσο το δυνατόν κοντύτερα από όλα τα σημεία. [91]

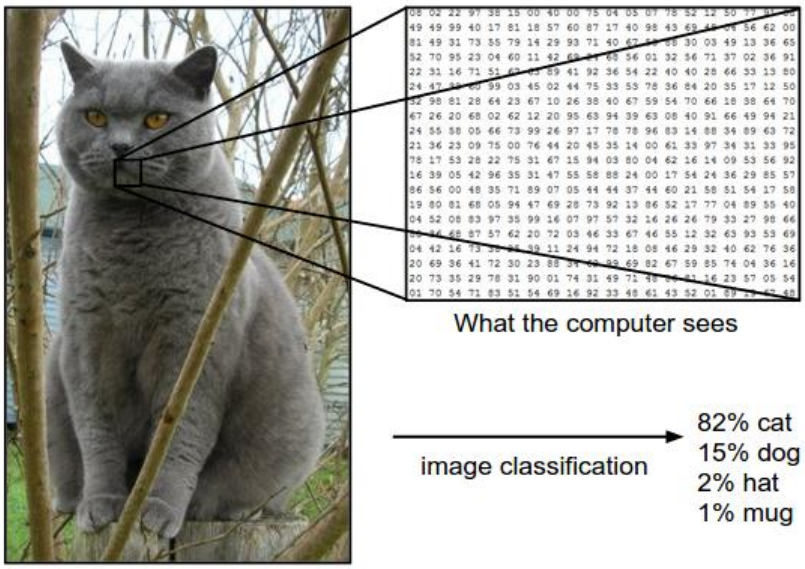


3. Ταξινόμηση Εικόνων (Image Classification)

Εφόσον εξετάσαμε στην προηγούμενη ενότητα τις βασικές κατηγορίες και έννοιες της μηχανικής μάθησης, στην συνέχεια θα αναλύσουμε τον τρόπο λειτουργίας ενός ταξινομητή εικόνων, των προβλημάτων που παρουσιάζονται και τον τρόπο αντιμετώπισης τους καθώς και πως γίνεται η εφαρμογή ενός αλγόριθμου για να βελτιστοποιηθεί η απόδοση του ταξινομητή.

Όπως περιγράφηκε προηγουμένως το πρόβλημα που πρέπει να επιλυθεί μέσω του Image Classification είναι να ανατεθεί σε μια εικόνα που δίνεται ως δεδομένο εισόδου, μία αντίστοιχη ονομασία, από μία προκαθορισμένη ομάδα κλάσεων.

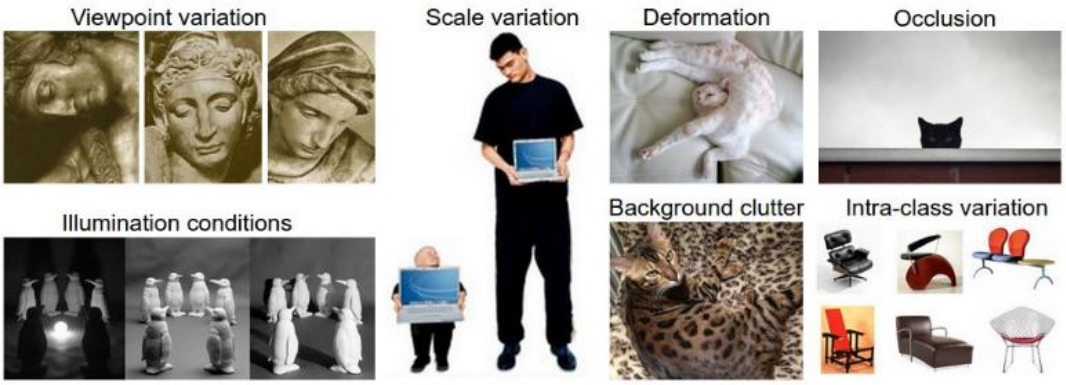
Εικόνα 12. Παράδειγμα εικόνας η οποία δίνεται ως είσοδος σε ένα σύστημα ταξινόμησης εικόνων στην οποία μπορούν να δοθούν οι εξής κατηγορίες: γάτα (cat), σκύλος (dog), καπέλο (hat), κούπα (mug). Για να απεικονίσουμε την εικόνα σε ένα υπολογιστικό περιβάλλον θα πρέπει να ορίσουμε την εικόνα ως ένα 3-διάστατο πίνακα. Εφόσον η εικόνα μας είναι μεγέθους 248x400 και αποτελείται από των συνδυασμό τριών χρωμάτων του μοντέλου RGB (κόκκινο, πράσινο, μπλε) έχουμε ένα πίνακα μεγέθους 248x400x3 = 297.600. Κάθε ακέραιος αριθμός από τους 297.600 έχει εύρος από 0 έως 255 όπου ως 0 είναι το μαύρο χρώμα και 255 το άσπρο. [88]



3.1 Προβλήματα Ταξινόμησης

Για να μπορέσει ένας αλγόριθμος ταξινόμησης εικόνων να αναγνωρίσει επιτυχημένα την κλάση στην οποία ανήκει η παραπάνω εικόνα θα πρέπει να λάβουμε υπόψη μας τα προβλήματα που πρέπει να ξεπεραστούν κατά τον σχεδιασμό ενός τέτοιου μοντέλου για να αποφέρει σωστά αποτελέσματα. Τα προβλήματα που μπορεί να προκύψουν είναι: η διαφοροποίηση στον προσανατολισμό του αντικειμένου προς την κάμερα **(viewpoint variation)**, η κλίμακα που μπορεί να έχει το αντικείμενο της εικόνας σε σχέση με το πραγματικό μέγεθος του **(scale variation)**, η μη πλήρης απεικόνιση του αντικειμένου σε μια εικόνα καθώς μπορεί μόνο ορισμένα pixel του να είναι εμφανή (**occlusion**), οι συνθήκες φωτισμού της εικόνας **(illumination conditions)**, η ύπαρξη πολλών αντικειμένων στο φόντο της εικόνας που ίσως να κάνουν πιο δυσδιάκριτο το αντικείμενο **(background clutter)**, και τέλος η ποικιλομορφία που μπορεί να έχει η κάθε κλάση που να εμποδίζει τον ακριβή καθορισμό της **(intra-class variation)**.

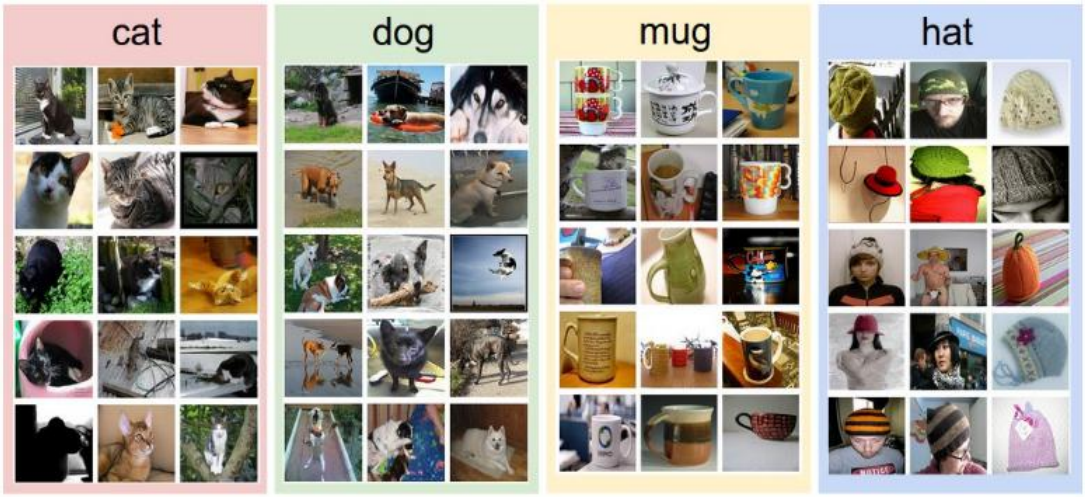
Εικόνα 13. Παράδειγμα προβλημάτων ταξινόμησης εικόνων. [88]



3.2 Προσέγγιση με γνώμονα τα δεδομένα

Ο σχεδιασμός ενός μοντέλου ταξινόμησης εικόνων θα πρέπει να δημιουργηθεί έτσι ώστε να λαμβάνει υπόψη τα παραπάνω προβλήματα και να μπορεί να τα ξεπεράσει. Ο πιο εύκολος τρόπος να επιτευχθεί αυτό είναι με την υιοθέτηση μιας προσέγγισης με γνώμονα τα δεδομένα για την εκπαίδευση του μοντέλου. Ο σκοπός μας θα πρέπει να είναι να μπορέσουμε να προσφέρουμε στο μοντέλο με όσο περισσότερα δεδομένα εκπαίδευσης μπορούμε, τα οποία να χαρακτηρίζουν και να απεικονίζουν ακριβέστερα την εξωτερική εμφάνιση της κάθε κλάσης, ώστε να μπορεί μέσω της διαδικασίας εκμάθησης, να αναπτυχθεί ένα ακριβές μοντέλο ταξινόμησης εικόνων. Συνεπώς, ένα από τα σημαντικότερα κομμάτια για την ταξινόμηση εικόνων είναι η δημιουργία ενός κατάλληλου **training dataset** από ήδη ταξινομημένες εικόνες που θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου. [14]

Εικόνα 14. Παράδειγμα ενός training dataset χωρισμένων ανά κατηγορία κλάσεων που επιθυμούμε να ταξινομούμε με το αλγοριθμικό μοντέλο. Σε μια πραγματική εφαρμογή ο αριθμός των εικόνων που θα ανήκει σε κλάση του dataset θα αντιστοιχούσε σε εκατοντάδες χιλιάδες εικόνες. [88]

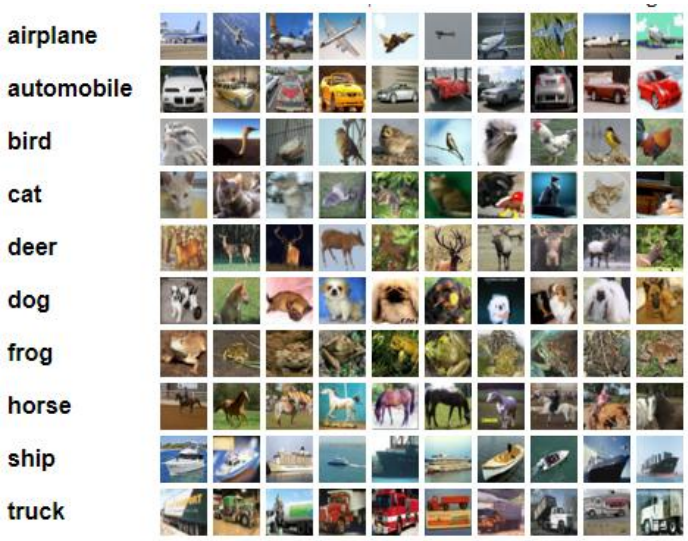


3.3 Ταξινομητής Κοντινότερου Γείτονα (Nearest Neighbor Classifier)

Για να μπορέσουμε να προσεγγίσουμε τον σχεδιασμό ενός ταξινομητή θα πρέπει να ορίσουμε τον αλγόριθμο με βάση τον οποίο θα γίνεται η ταξινόμηση των εικόνων ανά κλάση. Για το παράδειγμα μας, θα χρησιμοποιήσουμε τον αλγόριθμο του κοντινότερου γείτονα και κ-κοντινότερου γείτονα.

Το training dataset που θα χρησιμοποιηθεί για τον ταξινομητή είναι το **CIFAR-10** [15] , ένα από τα πιο διαδεδομένα dataset για χρήση σε ταξινομητές εικόνων. Το CIFAR-10 dataset αποτελείται από 60.000 εικόνες μεγέθους 32x32 pixel που διαχωρίζονται σε 10 κατηγορίες κλάσεων. Από αυτές, οι 50.000 χρησιμοποιούνται για την εκπαίδευση ενός αλγόριθμου και οι υπόλοιπες χρησιμοποιούνται για τον έλεγχο αποτελεσματικότητας του ταξινομητή. Αντίστοιχα, με το CIFAR-10 υπάρχει και το CIFAR-100 dataset το οποίο αποτελείται από 100 κλάσεις που περιέχουν 600 εικόνες ανά κλάση, όπου κάθε κλάση ανήκει σε μία από 20 υπερκλάσεις.

Εικόνα 15. Παράδειγμα των κλάσεων που περιέχονται στο CIFAR-10 με 10 τυχαίες εικόνες ανά κατηγορία. [88]



Για να εφαρμοστεί ο αλγόριθμος του κοντινότερου γείτονα θα έπρεπε να πάρουμε την απεικόνιση στοιχεία πίνακα της εικόνας που θέλουμε να ταξινομήσουμε, να την συγκρίνουμε ανά pixel με κάθε εικόνα του training set, και να αθροίσουμε τις διαφορές τους ώστε να καταλήξουμε στην εικόνα με το μικρότερο άθροισμα. Η κλάση στην οποία θα ανήκει η εικόνα με το μικρότερο άθροισμα θα είναι η κλάση στην οποία θα ταξινομηθεί η εικόνα εισόδου. Εναλλακτικά, αν θεωρήσουμε ως δύο διανύσματα **I_1 και I_2** τις εικόνες που συγκρίνουμε θα μπορούσαμε να υπολογίσουμε την διαφορά τους μέσω της **L1 απόστασης**, η οποία ορίζεται ως εξής: [16] [17]

$$d_1(I_1,I_2)=\sum_p|I_1^p-I_2^p| \text{ [17]}$$

Εικόνα 16. Παράδειγμα χρήσης απόστασης L1 για την σύγκριση δύο εικόνων για την $1^η$ διάσταση του πίνακα. Στα αριστερά είναι τα στοιχεία της εικόνας προς ταξινόμηση από την οποία αφαιρούμε την εικόνα από το CIFAR-10 για να καταλήξουμε σε ένα πίνακα που περιέχει τις διαφορές τους ανά pixel. Στο τέλος, αθροίζουμε όλα τα στοιχεία του πίνακα. Σε περίπτωση που το αποτέλεσμα είναι 0 τότε οι δύο εικόνες είναι πανομοιότυπες και αντίστοιχα αν υπάρχει μεγάλη διαφορά ανάμεσα τους το άθροισμα θα είναι αντίστοιχα μεγάλο. [88]

56	32	10	18
90	23	128	133
24	26	178	200
2	0	255	220

−

10	20	24	17
8	10	89	100
12	16	178	170
4	32	233	112

=

46	12	14	1
82	13	39	33
12	10	0	30
2	32	22	108

→ 456

Εναλλακτικά, θα μπορούσαμε να χρησιμοποιήσουμε κάποιον άλλο τρόπο υπολογισμού αποστάσεων μεταξύ διανυσμάτων, όπως η **L2 απόσταση,** η οποία είναι η ευκλείδεια απόσταση

μεταξύ δύο διανυσμάτων. Ουσιαστικά πρόκειται για την τετραγωνική ρίζα των αθροισμάτων των τετραγώνων της διαφοράς των στοιχείων των δύο πινάκων, και ορίζεται ως εξής:

$$d_1(I_1,I_2)=\sqrt{\sum p(I1p - I2p|)2} \quad [18]$$

Σε αρκετές περιπτώσεις μηχανικής μάθησης, προτιμάται η χρήση L2 αποστάσεων σε σύγκριση με την L1 καθώς προσφέρει πιο ευδιάκριτα αποτελέσματα κατά την εύρεση σφαλμάτων ταξινόμησης και διαφορών στοιχείων μεταξύ δύο εικόνων. [19]

Εικόνα 17. Αποτελέσματα του αλγόριθμου κοντινότερου γείτονα σε δέκα εικόνες του CIFAR-10. Στα δεξιά της κάθε εικόνας εμφανίζονται τα 10 πρώτα αποτελέσματα του αλγορίθμου εφόσον υπολογίστηκε η διαφορά της κάθε εικόνας ανά pixel. Όπως φαίνεται, στην εικόνα του αλόγου ο συγκεκριμένος αλγόριθμος δεν επιστρέφει αξιόπιστα αποτελέσματα για κάθε κλάση. [88]

Όπως παρατηρείται, ο αλγόριθμος του κοντινότερου γείτονα δεν επιστρέφει ικανοποιητικά αποτελέσματα κατά την ταξινόμηση εικόνων. Στην πραγματικότητα ο συγκεκριμένος αλγόριθμος πρόκειται για μια υποπερίπτωση του αλγόριθμου **k-κοντινότερου γείτονα (k-nearest neighbor)** που εξετάστηκε προηγουμένως, και συγκεκριμένα την περίπτωση όπου ορίζεται **k = 1** στον αλγόριθμο. Σε περίπτωση όπου **k > 1,** ο αλγόριθμος θα εντοπίσει τους k-κοντινότερους γείτονες και θα ταξινομήσει την εικόνα με βάση την κλάση στην οποία ανήκουν οι περισσότεροι γείτονες**,** αυξάνοντας κατά αυτό τον τρόπο και την αποδοτικότητα του ταξινομητή.

Εικόνα 18. Παράδειγμα εφαρμογής ενός ταξινομητή κοντινότερου γείτονα (κέντρο) και ενός ταξινομητή 5 κοντινότερων γειτόνων (δεξιά) σε ένα dataset 3 κλάσεων (αριστερά). Στην περίπτωση του ταξινομητή κοντινότερου γείτονα με μέτρηση L2 αποστάσεων παρατηρούμε πως υπάρχουν κοντινότεροι γείτονες σε σημεία αρκετά απομακρυσμένα από την υπόλοιπη κλάση τους, κάτι που πιθανά δείχνει λάθος ταξινόμηση των στοιχείων, ενώ στην περίπτωση όπου έχουμε 5 κοντινότερους γείτονες εξομαλύνονται αυτές οι περιπτώσεις και οδηγούν σε καλύτερα αποτελέσματα για τον ταξινομητή. [88]

Συμπερασματικά, ο αλγόριθμος κοντινότερου γείτονα έχοντας τα πλεονεκτήματα της ταχύτητας στην εκπαίδευση του μπορεί να επιφέρει αρκετά ικανοποιητικά αποτελέσματα σε εικόνες μικρών διαστάσεων. Δυστυχώς, για πρακτικές εφαρμογές όπου τα δεδομένα εισόδου είναι εικόνες αρκετά υψηλής ανάλυσης, η αποτελεσματικότητα του μειώνεται δραματικά. Επιπλέον, έχει το μειονέκτημα πως είναι αρκετά χρονοβόρος κατά την ταξινόμηση καθώς πρέπει να υπολογίσει τις διαφορές της εικόνας εισόδου με όλες τις εικόνες του training dataset. Το αποτέλεσμα αυτό είναι το αντίθετο από το επιθυμητό καθώς σε μία real-time εφαρμογή μας ενδιαφέρει περισσότερο η ταχύτητα κατά την ταξινόμηση, παρά κατά την εκπαίδευση. Όπως θα δούμε στην συνέχεια με την χρήση νευρωνικών δικτύων μπορούμε να επιτύχουμε μεγαλύτερη ταχύτητα και ακρίβεια κατά την ταξινόμηση εις βάρος όμως του χρόνου εκπαίδευσης. [88]

3.4 Αλγόριθμοι Εκπαίδευσης Ταξινομητή

Στη συνέχεια, θα γίνει η ανάλυση και επεξήγηση του τρόπου λειτουργίας των αλγόριθμων που χρησιμοποιεί η εφαρμογή ταξινόμησης προσώπων για την εκπαίδευση των training dataset.

3.4.1 Μηχανές Διανυσματικής Στήριξης (SVM)

Τα Support Vector Machines ή Μηχανές Διανυσματικής Στήριξης ορίζονται ως μία ομάδα μοντέλων εκμάθησης με επίβλεψη (supervised learning methods). Χρησιμοποιούνται κατά κύριο λόγο, για την ανάλυση δεδομένων και την αναγνώριση προτύπων σε θέματα ταξινόμησης (μηχανική μάθηση) και συγκεκριμένα, στην περίπτωση της εφαρμογής που εξετάζουμε, σε θέματα αναγνώρισης προσώπων (facial recognition). Ο αρχικός αλγόριθμος Μηχανών Διανυσμάτων Υποστήριξης υλοποιήθηκε από τους Vladimir Vapnik και Alexey Ya. Chervonenkis το 1963 και η τωρινή μορφή του (soft margin) προτάθηκε από τους Corinna Cortes και Vladimir Vapnik το 1993 και εκδόθηκε το 1995.[20]

Η ανάπτυξη των Μηχανών Διανυσματικής Στήριξης είχε την αντίστροφη πορεία από τον τρόπο εξέλιξης των νευρωνικών δικτύων. Οι Μηχανές Διανυσματικής Στήριξης είχαν πρωτίστως θεωρητικό υπόβαθρο κατά την ανάπτυξη τους και στη συνέχεια έγινε η εφαρμογή τους σε πειράματα σε αντίθεση με τα νευρωνικά δίκτυα τα οποία ανέπτυξαν την θεωρία τους μέσα από εκτενή πειράματα και διάφορες εφαρμογές. [21] Η επικρατούσα γνώμη για τις Μηχανές Διανυσματικής Στήριξης ήταν πως παρόλο που το θεωρητικό τους υπόβαθρο ήταν αδιαμφισβήτητο δεν θα μπορούσαν να αξιοποιηθούν σε πρακτικές εφαρμογές. Η μετέπειτα δημοτικότητα ή αποδοχή τους οφείλεται στο μεγάλο ποσοστό επιτυχίας που είχε για εφαρμογές αναγνώρισης χειρόγραφων ψηφίων. Το ποσοστό λάθους του ήταν περίπου 1,1%, ποσοστό αντίστοιχο ενός χρονοβόρου σε υλοποίηση νευρωνικού δικτύου. [22] Στην σημερινή εποχή, οι Μηχανές Διανυσματικής Στήριξης αποφέρουν συγκριτικά καλύτερα αποτελέσματα σε σχέση με άλλα νευρωνικά δίκτυα ή άλλα στατιστικά μοντέλα σε προβλήματα που χρησιμοποιούνται ως σημεία αναφοράς για την αποτελεσματικότητα ενός αλγόριθμου. [23] [24] [25] [26]

Ο αλγόριθμος Μηχανών Διανυσματικής Στήριξης λοιπόν στοχεύει στην επίλυση ταξινόμησης προσώπων, και επομένως το ερώτημα που προσπαθεί να απαντήσει είναι αν το πρόσωπο το οποίο εξετάζεται ανήκει σε μία από τις ορισμένες κλάσεις και να το κατατάξει κατά βέλτιστο τρόπο σε μία από αυτές. Ο διαχωρισμός αυτός μπορεί να γίνει δίνοντας την τιμή 1 σε περίπτωση που ανήκει στην συγκεκριμένη κλάση και -1 σε περίπτωση που δεν ανήκει. Συμπεραίνουμε λοιπόν πως για επίλυση του συγκεκριμένου προβλήματος θα πρέπει να χρησιμοποιήσουμε ένα αλγόριθμο γραμμικής ταξινόμησης.

Οι Μηχανές Διανυσματικής Στήριξης στοχεύουν στο να βρουν το βέλτιστο υπερεπίπεδο (hyperplane) το οποίο θα διαχωρίζει τα δεδομένα των κλάσεων σε ένα ν-διάστατο χώρο (feature space). Από την στιγμή που τα δεδομένα είναι γραμμικά διαχωρίσιμα, το βέλτιστο υπερεπίπεδο είναι αυτό το οποίο θα έχει την μεγαλύτερη απόσταση από τα δεδομένα των δύο κλάσεων τα οποία έχουν την κοντινότερη απόσταση μεταξύ τους. Αρά, μπορούμε να θεωρήσουμε το υπερεπίπεδο ως μία ευθεία γραμμή η οποία θα διαχωρίζει τις κλάσεις ώστε να έχουμε την μικρότερη δυνατή πιθανότητα λάθους (training error). Στη συνέχεια θα εξετάσουμε τον τρόπο

κατά τον οποίο λειτουργούν οι Μηχανές Διανυσματικής Στήριξης για να βρει το βέλτιστο υπερεπίπεδο. [27]

Αρχικά θα θεωρήσουμε πως τα δεδομένα εισόδου μας (training data) είναι της παρακάτω μορφής:

$(x_1,y_1),(x_2,y_2), \ldots ,(x_l,y_l), x \in R^n, y \in \{+1, -1\}$ [89]

Για λόγους καλύτερης απεικόνισης θα θεωρήσουμε ένα δισδιάστατο επίπεδο στο οποίο περιέχονται τα δεδομένα εισόδου το οποίο ορίζουμε ως :

$x \in R^2$

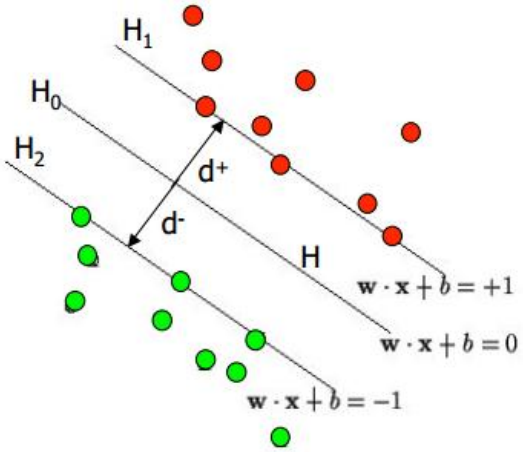
Με αυτό τον τρόπο τα δεδομένα μας μπορούν να διαχωριστούν γραμμικά και προκύπτουν αρκετά διαφορετικά υπερεπίπεδα τα οποία μπορούν να τα διαχωρίσουν. Τα υπερεπίπεδα μπορούν να οριστούν ως:

$wx_i + b >= 1$ όταν $y_i = +1$ (1) [89]

$wx_i + b <= -1$ όταν $y_i = -1$ (2) [89]

Με βάση την παρακάτω εικόνα παρατηρούμε πως:

Εικόνα 19. Διαγραμματική Απεικόνιση SVM. [89]



Τα υπερεπίπεδα H_1 και H_2 ορίζονται ως :

$H_1 : wx_i + b = +1$ [89]

$H_2 : wx_i + b = -1$ [89]

Τα σημεία τα οποία τέμνονται από τις δύο ευθείες είναι οι κορυφές των Διανυσμάτων Υποστήριξης. Αντίστοιχα, η ευθεία H_0 , αποτελεί το μέσο υπερεπίπεδο το οποίο διαχωρίζει τα δεδομένα και ορίζεται ως:

$H_0 : wx_i + b = 0$ [21]

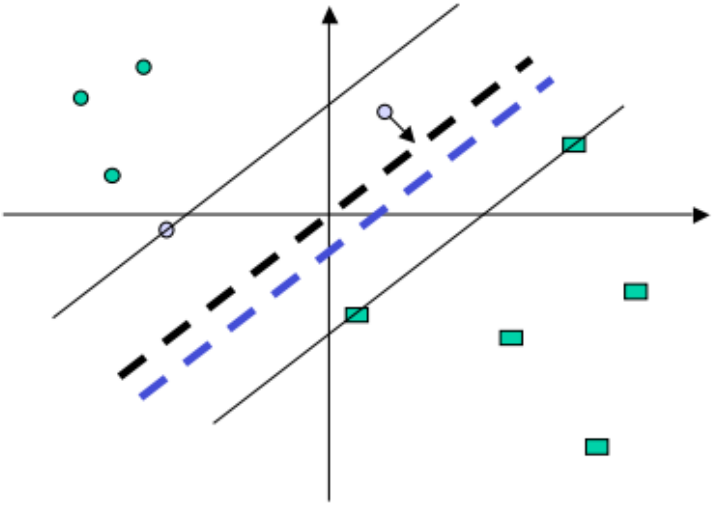
d+ = η μικρότερη απόσταση από το κοντινότερο θετικό σημείο

d- = η μικρότερη απόσταση από το κοντινότερο αρνητικό σημείο

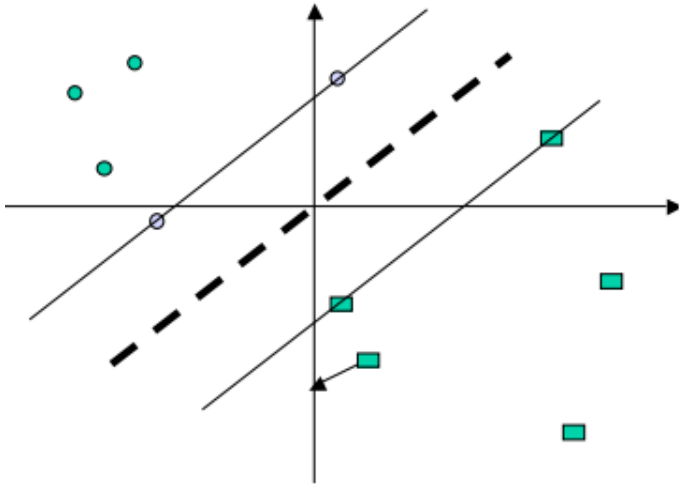
Ο αλγόριθμος βελτιστοποίησης που καθορίζει τα αντίστοιχα βάρη (w) λειτουργεί με τέτοιο τρόπο ώστε μόνο τα Διανύσματα Υποστήριξης να καθορίζουν τα βάρη και ακολούθως το υπερεπίπεδο που τα διαχωρίζει. Από τις παρακάτω εικόνες συνεπάγεται πως η μετακίνηση

ενός Διανύσματος Υποστήριξης επηρεάζει όλο το όριο απόφασης ενώ η μετακίνηση οποιουδήποτε άλλου διανύσματος δεν έχει κάποιο αποτέλεσμα.

Εικόνα 20. Διαγραμματική Απεικόνιση SVM. [89]



Εικόνα 21. Διαγραμματική Απεικόνιση SVM. [89]



Με βάση τα παραπάνω η μορφή της εξίσωσης η οποία ορίζει το βέλτιστο υπερεπίπεδο είναι η εξής:

$w^T x + b = 0$ [89]

όπου,

w = διάνυσμα βάρους (weight vector), x = διάνυσμα εισόδου (input vector), b = κλίση (bias)

Επομένως για κάθε απόσταση d_i έχουμε:

$w^T x + b >= 0$ για $d_i = +1$ [89]

$w^T x + b < 0$ για $d_i = -1$ [89]

όπου,

d = περιθώριο διαχωρισμού (margin of separation)

Για την εύρεση του βέλτιστου υπερεπιπέδου (optimal hyperplane), δηλαδή το υπερεπίπεδο εκείνο στο οποίο το d είναι μέγιστο, πρέπει να βρούμε την μέγιστη απόσταση ανάμεσα στα H_1 και H_2. Η απόσταση τους μπορεί να υπολογιστεί ως:

$$\begin{cases} wx_1 + b = +1 \\ wx_1 + b = -1 \end{cases} \Rightarrow w(x_1 - x_2) = 2 \Rightarrow \frac{w}{\|w\|(x_1 - x_2)} = \frac{2}{\|w\|} \text{ [28]}$$

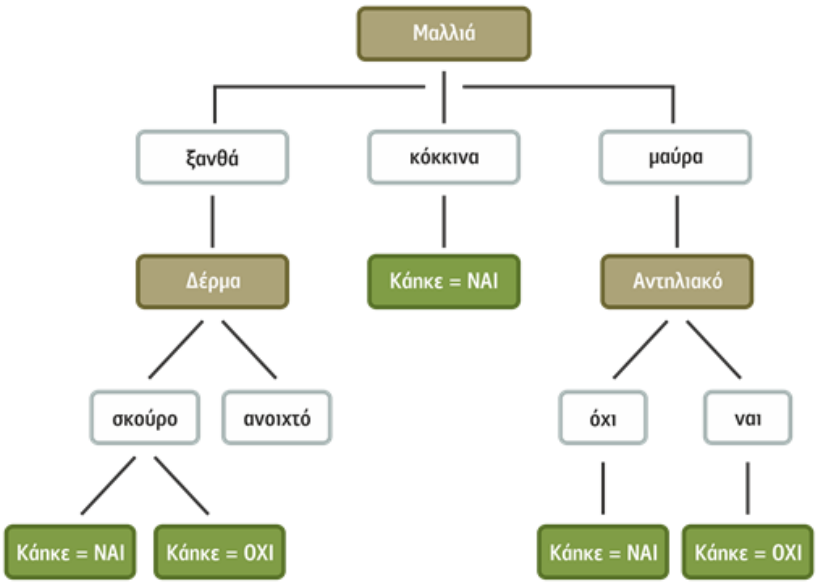
Συμπεραίνουμε λοιπόν πως για την μεγιστοποίηση του ορίου οι Μηχανές Διανυσματικής Στήριξης πρέπει να υπολογίσουν το υπερεπίπεδο κατά το οποίο το **||w||** είναι ελάχιστο.

3.4.2 Δέντρο Αποφάσεων (Decision Tree)

Η χρήση αλγορίθμων Δέντρου Αποφάσεων για πρακτικές μηχανικής μάθησης αποσκοπεί στην απεικόνιση συναρτήσεων διακριτών τιμών με την χρήση δενδροειδών απεικονίσεων. Τα δέντρα απόφασης είναι μέθοδοι εκμάθησης με επίβλεψη και χρησιμοποιούνται συχνά σε θέματα ταξινόμησης. Εναλλακτικά, τα τελικά δέντρα αποφάσεων μπορούν να απεικονιστούν και με την χρήση κανόνων εάν-τότε (if-then) για την βελτίωση της αναγνωσιμότητας τους. Η συγκεκριμένη αλγοριθμική μέθοδος έχει εφαρμοστεί με επιτυχία σε τομείς από υπολογισμός πιστωτικού ρίσκου για δανειολήπτες έως διαγνώσεις ιατρικών περιστατικών.

Η δομή ενός δέντρο απόφασης χαρακτηρίζεται από την ταξινόμηση της κάθε πιθανής περίπτωσης κατά μήκος του δέντρου, ξεκινώντας από την ρίζα και καταλήγοντας σε κόμβους φύλλων. Κάθε κόμβος του δέντρου απεικονίζει ένα χαρακτηριστικό και κάθε κλαδί το οποίο ακολουθείται από αυτό αντιστοιχεί σε μία από όλες τις πιθανές τιμές που μπορεί να δεχτεί το συγκεκριμένο χαρακτηριστικό. Η διαδικασία ταξινόμησης ξεκινάει από την ρίζα του δέντρου απόφασης και με βάση τις τιμές του κάθε χαρακτηριστικού προχωράει στον επόμενο κόμβο και στη συνέχεια επαναλαμβάνει την διαδικασία σε κάθε υπό-δέντρο που σχηματίζεται μέχρι να καταλήξει σε ένα τελικό κόμβο φύλλου, όπου θα αντιστοιχεί σε μία από τις κλάσεις του training dataset. [29]

Εικόνα 22. Δένδρο απόφασης για πρόβλημα χρήσης αντηλιακού. [30]



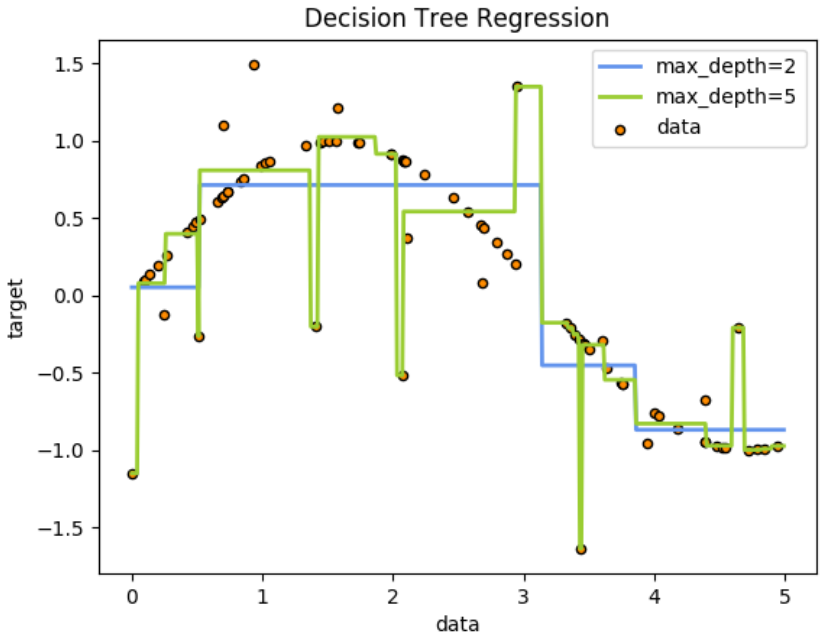
Το παραπάνω δέντρο απεικονίζει την πιθανότητα να καεί ένα άτομο και χωρίζεται ανάλογα με το δέρμα και την χρήση αντηλιακού. Για παράδειγμα, η περίπτωση όπου «χρώμα μαλλιών = ξανθά, Δέρμα = σκούρο» θα ταξινομούταν στον κόμβο που βρίσκεται στα πιο αριστερά του δέντρου και θα είχε ταξινομούταν ως αρνητική περίπτωση, «Κάηκε = Ναι».

Σε γενικές γραμμές, τα δέντρα αποφάσεων απεικονίζουν ένα μοντέλο διαχωρισμού των συνδυασμών των περιορισμών με βάση τις πιθανές τιμές χαρακτηριστικών ανά περίπτωση. Κάθε διαδρομή κατά μήκος του δέντρου αντιστοιχεί και σε ένα τέτοιο συνδυασμό των χαρακτηριστικών, και το πλήρες δέντρο απεικονίζει τον τρόπο διαχωρισμού αυτών των συνδυασμών.

Οι αλγόριθμοι δέντρων απόφασης συνήθως χρησιμοποιούνται για την λύση προβλημάτων με τα εξής χαρακτηριστικά:

- Περιπτώσεις όπου μπορούν να απεικονιστούν από ζεύγη τιμών των χαρακτηριστικών τους. Οι περιπτώσεις αυτές διαχωρίζονται σε μία προκαθορισμένη ομάδα χαρακτηριστικών όπως, π.χ. θερμοκρασία, και στις τιμές που μπορούν να δεχτούν π.χ. κρύο, ζέστη. Ιδανικές περιπτώσεις προβλημάτων είναι αυτές όπου σε κάθε χαρακτηριστικό αντιστοιχεί ένας μικρός αριθμός εύκολα διακριτών τιμών.
- Περιπτώσεις όπου οι τελικές τιμές των κόμβων είναι διακριτές. Το παράδειγμα δέντρου που παρουσιάστηκε προηγουμένως είναι ένα παράδειγμα Boolean ταξινόμησης (Ναι ή Όχι) για το κάθε δεδομένο εισόδου. Τα δέντρα αποφάσεων μπορούν να επεκταθούν και στην εκμάθηση συναρτήσεων με περισσότερες από δύο τελικές τιμές, όπως π.χ. σύνολο πραγματικών τιμών παρότι τέτοιες περιπτώσεις δεν εφαρμόζονται στην πραγματικότητα.
- Περιπτώσεις όπου μπορούν να παρουσιάζονται σφάλματα στο training dataset. Οι αλγόριθμοι δέντρων απόφασης μπορούν να ανταπεξέλθουν χωρίς προβλήματα σε περιπτώσεις όπου υπάρχουν σφάλματα, τόσο σε λάθος ταξινομήσεις του training dataset όσο και στις τιμές που μπορεί να έχουν εξαχθεί από αυτά.
- Περιπτώσεις όπου λείπουν δεδομένα από το training dataset. Σε περίπτωση που από το training dataset λείπουν δεδομένα ή περιέχονται άγνωστες τιμές για κάποιες περιπτώσεις κλάσεων, ένας αλγόριθμος δέντρου απόφασης μπορεί να εκτελεστεί και να κάνει πρόβλεψη.

Εικόνα 23. Παράδειγμα χρήσης αλγόριθμου δέντρου απόφασης όπου έχει γίνει εκμάθηση σε δεδομένα ώστε να προσεγγιστεί μία ημιτονοειδής καμπύλα, με την χρήση μιας ομάδας κανόνων If-Then. Όσο πιο μεγάλο βάθος έχει το δέντρο, τόσο πιο πολύπλοκοι είναι οι κανόνες που το απαρτίζουν και τόσο πιο ακριβές το τελικό μοντέλο. [90]



Τα πλεονεκτήματα που έχουν τα δέντρα απόφασης είναι:

- Απλά στην κατανόηση και απεικόνιση. Τα δέντρα αποφάσεων μπορούν να οπτικοποιηθούν χωρίς να γίνονται δυσνόητα.
- Δεν απαιτούν μεγάλο βαθμό προεπεξεργασίας των αρχικών δεδομένων, όπως data normalization, αφαίρεση δεδομένων με έλλειψη τιμών, δημιουργία dummy μεταβλητών.
- Μπορούν να διαχειριστούν και κατηγορικά και αριθμητικά δεδομένα σε αντίθεση με άλλες αλγοριθμικές μεθόδους που είναι εξειδικευμένες σε ένα τύπο δεδομένων.
- Είναι μοντέλα φιλοσοφίας «white box». Σε αντίθεση με «black box» μεθόδους, στα δέντρα απόφασης κάθε πιθανή κατάσταση μπορεί να παρατηρηθεί με την εξέταση του μοντέλου και να εξηγηθεί με την χρήση boolean τιμών.
- Μπορεί να επαληθευθεί η αποδοτικότητα του μοντέλου με την χρήση στατιστικών ελέγχων.

Αντίστοιχα, στα μειονεκτήματα περιλαμβάνονται:

- Μπορεί να παρουσιαστούν περιπτώσεις δημιουργίας πολύπλοκων δέντρων με αποτέλεσμα της κακή γενίκευση των δεδομένων. Σε αυτές τις περιπτώσεις απαιτείται ο ορισμός του μέγιστου βάθους του δέντρου μέσω μεθόδων pruning.
- Παρουσιάζουν αστάθεια καθώς μικρές διακυμάνσεις στα δεδομένα μπορεί να έχουν ως αποτέλεσμα την δημιουργία πολύ διαφορετικών δέντρων.
- Παρουσιάζουν δυσκολία στην απεικόνιση εννοιών όπως προβλήματα XOR ή ισοτιμίας.
- Η εκμάθηση ενός βελτιστοποιημένου δέντρου αποτελεί πρόβλημα, ακόμα και για απλά προβλήματα. Οι περισσότερες εφαρμογές δέντρων απόφασης βασίζονται σε ευρετικούς αλγόριθμους (heuristic algorithms) οι οποίοι παίρνουν τοπικά βέλτιστες αποφάσεις ανά κόμβο, χωρίς να εξετάζεται αν το συνολικό δέντρο απόφασης είναι βέλτιστο. [90]

Ο τρόπος σχηματισμού ενός δέντρου απόφασης για ταξινόμηση δεδομένων είναι ο εξής.

Για διανύσματα εκπαίδευσης $x_i \in R^n$, όπου i=1,…, I και διάνυσμα κλάσης $y \in R^l$ το δέντρο απόφασης αναδρομικά θα διαχωρίσει τα δεδομένα ώστε αυτά τα οποία ανήκουν στην ίδια κλάση να ομαδοποιηθούν.

Τα δεδομένα σε κάθε κόμβο m συμβολίζονται ως Q. Για κάθε πιθανή ακμή του δέντρου $\theta = (j, t_m)$ όπου j είναι ένα χαρακτηριστικό και η τιμή κατωφλίου είναι t_m, τα δεδομένα διαχωρίζονται στις υπό-κατηγορίες $Q_{left}(\theta)$ και $Q_{right}(\theta)$ για τις οποίες ισχύει

$$Q_{left}(\theta) = (x, y)|x_j <= t_m$$
$$Q_{right}(\theta) = Q \setminus Q_{left}(\theta)$$ [90]

Η περίπτωση λάθος ομαδοποίησης σε ένα κόμβο m υπολογίζεται από την χρήση μίας συνάρτησης $H()$, η οποία εξαρτάται από τον τύπο προβλήματος που πρέπει να επιλυθεί όπως π.χ. ταξινόμηση.

Συνεπώς, για δεδομένα εκπαίδευσης X_m ενός κόμβου m υπάρχουν οι αντίστοιχες μέθοδοι μέτρησης σφαλμάτων ομαδοποίησης.

- **Συνάρτηση Gini**

$$H(X_m) = \sum_k p_{mk}(1 - p_{mk})$$ [90]

- **Συνάρτηση Cross-Entropy**

$$H(X_m) = -\sum_k p_{mk} \log(p_{mk})$$ [90]

- **Συνάρτηση Misclassification**

$$H(X_m) = 1 - \max(p_{mk})$$ [90]

Για τις οποίες, για διακριτές τιμές εύρους 0,1,…,K-1 για ένα κόμβο m , με N_m πλήθος παρατηρήσεων, ορίζεται το ποσοστό των παρατηρήσεων της κλάσης k ως

$$p_{mk} = 1/N_m \sum_{x_i \in R_m} I(y_i = k)$$ [90]

Στη συνέχεια και με βάση την

$$G(Q, \theta) = \frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta))$$ [90]

υπολογίζονται οι παράμετροι

$$\theta^* = \text{argmin}_\theta \, G(Q, \theta)$$ [90]

που ελαχιστοποιούν το σφάλμα.

Τελικώς, γίνεται αναδρομή της διαδικασίας για τα

$Q_{left}(\theta^*)$ και $Q_{right}(\theta^*)$ μέχρι να επιτευχθεί η μεγιστοποίηση του βάθους όπου θα ισχύει, [31] [32] [33]

$$N_m < \text{min}_{samples}$$ [90]

ή

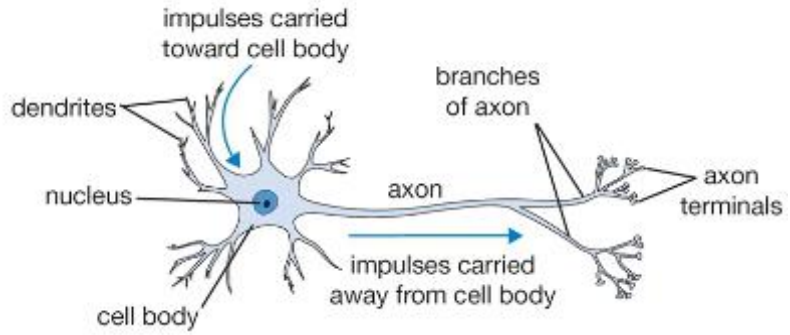
$$N_m = 1.$$ [90]

4. Νευρωνικά Δίκτυα (Neural Networks)

Τα Τεχνητά Νευρωνικά Δίκτυα προέκυψαν από την ιδέα προσομοίωσης των βιολογικών νευρωνικών συστημάτων του ανθρώπινου οργανισμού και εφαρμόζονται κατά κόρον σε θέματα μηχανικής και όπως στην περίπτωση μας, σε θέματα μηχανικής μάθησης.

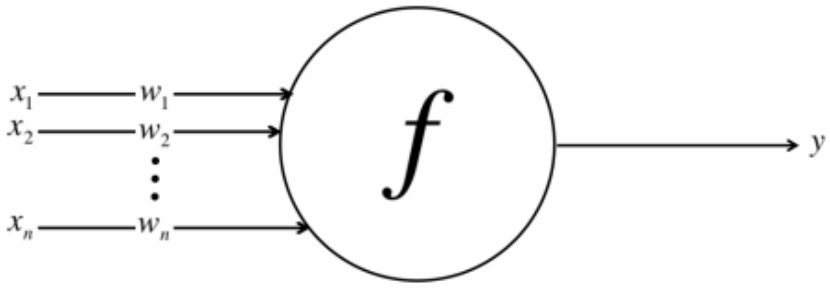
Αρχικά ας εξετάσουμε πως λειτουργεί το σύστημα των νευρώνων που υπάρχουν στον άνθρωπο. Ένα ανθρώπινο νευρικό σύστημα αποτελείται από περίπου 86 δισεκατομμύρια νευρώνες οι οποίοι συνδέονται μεταξύ τους από περίπου 10^{14} με 10^{15} συνάψεις. Όπως φαίνεται στην παρακάτω εικόνα κάθε νευρώνας δέχεται σήματα εισόδου από τους δενδρίτες και παράγει ένα σήμα εξόδου κατά μήκος του άξονα του. Στη συνέχεια, ο άξονας διακλαδώνεται και μέσω των συνάψεων του ενώνεται με άλλους δενδρίτες από διαφορετικούς νευρώνες.

Εικόνα 24. Απεικόνιση των τμημάτων ενός βιολογικού νευρώνα [88]



Αντίστοιχα, το κύριο συστατικό ενός Νευρωνικού Δικτύου είναι οι τεχνητοί νευρώνες του ή αλλιώς **αισθητήρες (perceptron).** Η ιδέα των perceptrons αναπτύχθηκε κατά τις δεκαετίες 1950 και 1960 από τον Frank Rosenblatt, ο οποίος εμπνεύστηκε από το έργο των Warren McCulloch και Walter Pitts και βασίζεται στην θεωρία ενός βιολογικού νευρώνα. [34]

Εικόνα 25. Απεικόνιση ενός αισθητήρα νευρωνικού δικτύου [35]



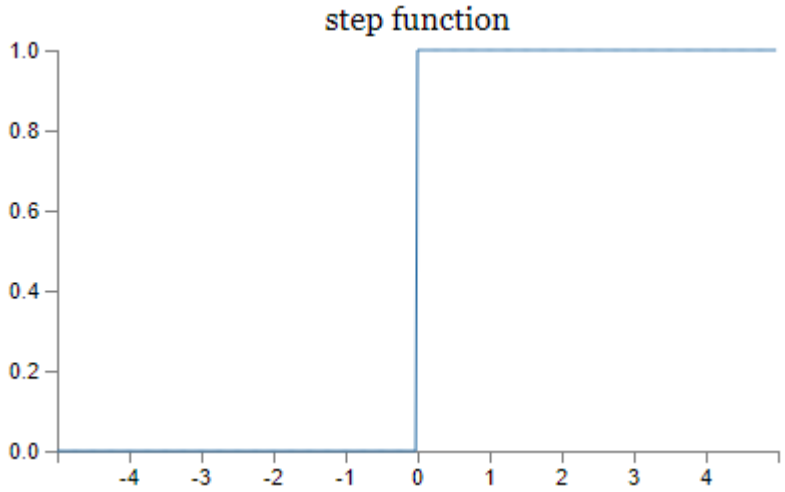
Επομένως, ένα perceptron δέχεται ως είσοδο διάφορα δεδομένα δυαδικά (0 ή 1) δεδομένα εισόδου x_1, x_2, x_3,..., x_n και από αυτά εξάγει μία μόνο δυαδική τιμή εξόδου (0 ή 1). Στην παρακάτω εικόνα απεικονίζεται ένα perceptron το οποίο δέχεται τρία σήματα ως είσοδο. Η μέθοδος που πρότεινε ο Rosenblatt για την εκτίμηση του σήματος εξόδου είναι η εισαγωγή ενός συντελεστή βάρους **w** για κάθε τιμή εισόδου που δέχεται το perceptron και το οποίο θα καθορίζει το πόσο σημαντικό είναι το κάθε σήμα εισόδου σε σχέση με το αποτέλεσμα. Το αν το σήμα εξόδου θα έχει την τιμή 0 ή 1 καθορίζεται επομένως από αν το αθροιστικό βάρος των $\Sigma_j w_j x_j$ είναι μεγαλύτερο ή μικρότερο από μία **οριακή τιμή**. Αυτή η οριακή τιμή αποτελεί μία παράμετρο του νευρώνα και καθορίζει άμεσα το αποτέλεσμα που θα έχει η τιμή εξόδου του perceptron. Επομένως, το αποτέλεσμα της εξόδου θα μπορούσε να απεικονιστεί ως εξής:

$$\text{output} = \begin{cases} 0 & \text{if } w \cdot x + b \leq 0 \\ 1 & \text{if } w \cdot x + b > 0 \end{cases}$$ [94]

όπου, ως b ή bias ορίζουμε την τιμή όπου θα καθορίζει πόσο εύκολο είναι να έχουμε ως έξοδο την τιμή 1.

Όταν οι συγκεκριμένες τιμές εισόδου περνάνε μέσα από τον αισθητήρα χρησιμοποιούνται από μία συνάρτηση f για να έχουμε το αποτέλεσμα **y,** όπου **y = f(w*x + b) [35].** Κάθε νευρώνας καθορίζεται από την συνάρτηση που χρησιμοποιεί και μπορεί να κάνει μονάχα τους υπολογισμούς που καθορίζονται από την συνάρτηση του. Η πιο απλή μορφή ενός νευρώνα είναι αυτή που εξετάσαμε προηγουμένως και περιλαμβάνει μια γραμμική ή συνάρτηση βήματος (step function). Αυτής της μορφής οι νευρώνες μπορούν να χρησιμοποιηθούν για την προβλήματα classification. [35]

Εικόνα 26. Διαγραμματική απεικόνιση μιας step function. Βλέπουμε πως για την τιμή 0 και πάνω η συνάρτηση επιστρέφει την τιμή 1 ενώ για τιμές κάτω του μηδενός επιστρέφεται η τιμή 0 [88]



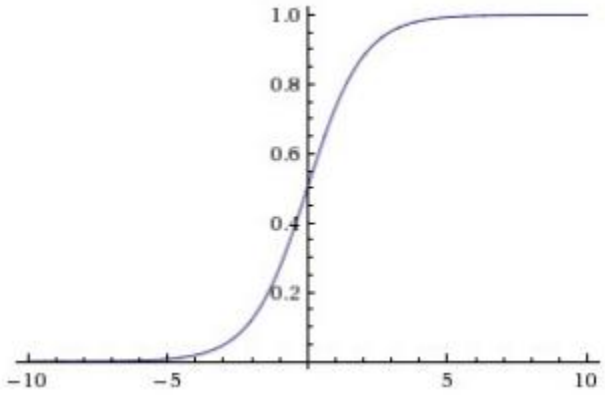
4.1 Συναρτήσεις Ενεργοποίησης (Activation Functions)

Το πρόβλημα που αντιμετωπίζουμε με τους γραμμικούς αισθητήρες είναι πως μία πολύ μικρή αλλαγή στις παραμέτρους της συνάρτησης μπορεί να αλλάξει πλήρως την συμπεριφορά ενός νευρωνικού δικτύου. Για να αποφευχθούν απότομες αλλαγές κατά την σταδιακή μεταβολή των παραμέτρων της συνάρτησης μπορούμε να χρησιμοποιήσουμε συναρτήσεις οι οποίες θα διευκολύνουν στην εκπαίδευση των νευρωνικών δικτύων. Στην συνέχεια, θα εξετάσουμε ορισμένες από τις πιο σημαντικές **συναρτήσεις ενεργοποίησης (activation functions)**.

4.1.1 Σιγμοειδής συνάρτηση (Sigmoid function)

Η σιγμοειδής συνάρτηση είναι της μορφής $\sigma(x)=1/(1+e^{-x})$ **[35]** και ήταν αρκετά διαδεδομένη για την εκπαίδευση δικτύων, καθώς επιτρέπει την σταδιακή αύξηση των τιμών από 0 έως 1, ενώ για πολύ μεγάλους αρνητικούς και θετικούς αριθμούς επιστρέφει τις τιμές 0 και 1 αντίστοιχα.

Εικόνα 27. Παράδειγμα σιγμοειδούς συνάρτησης [35]

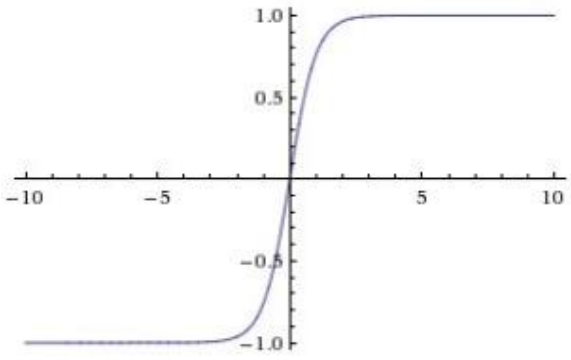


Πλέον, για πρακτικές εφαρμογές η συγκεκριμένη συνάρτηση χρησιμοποιείται σπάνια γιατί έχει ένα σημαντικό μειονέκτημα. Εφόσον, για πολύ μεγάλους αριθμούς επιστρέφεται πάντα η τιμή 0 ή 1, απαιτείται μεγάλη προσοχή κατά τον ορισμό των τιμών για τα βάρη της συνάρτησης ούτως ώστε να μην είναι πολύ υψηλές και εμποδίζει την αποτελεσματική εκπαίδευση του δικτύου.

4.1.2 Συνάρτηση Tanh

Η συγκεκριμένη συνάρτηση ενεργοποίησης έχει αρκετές ομοιότητες με την σιγμοειδή, με την διαφορά ότι λαμβάνει τιμές εύρους από -1 έως 1 και για αυτό τον λόγο προτιμάται σε εφαρμογές από την σιγμοειδή. Η συνάρτηση είναι της μορφής **$\tanh(x)=2\sigma(2x)-1\tanh(x)=2\sigma(2x)-1$ [35].**

Εικόνα 28. Παράδειγμα συνάρτησης Tanh [35]

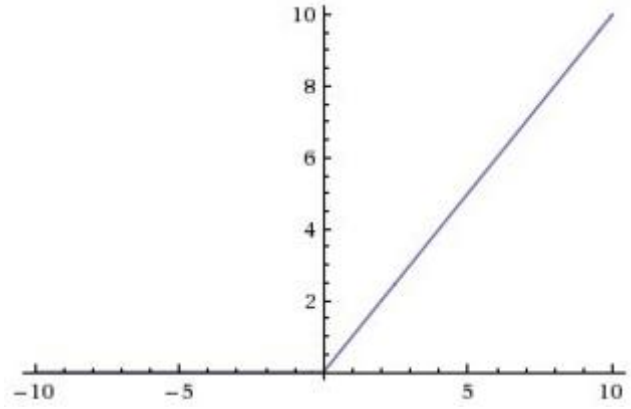


+

4.1.3 Συνάρτηση ReLU (Rectified Linear Unit)

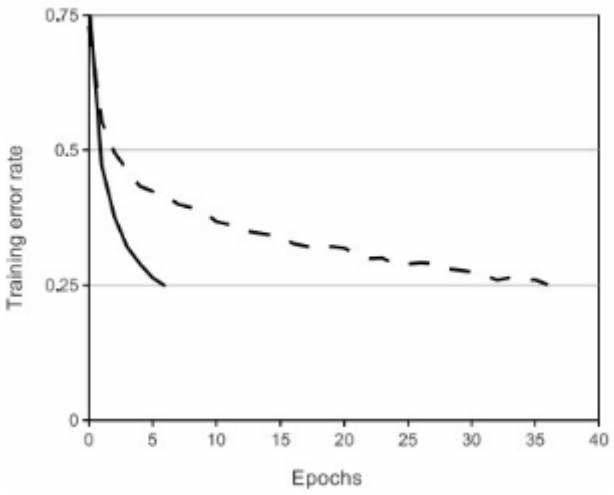
Η συνάρτηση ReLu είναι αντίστοιχη της γραμμικής συνάρτησης μόνο που ο νευρώνας ενεργοποιείται μόνο για τιμές πάνω από το 0. Η συνάρτηση είναι της μορφής **f(x)=max(0,x) [35]**.

Εικόνα 29. Παράδειγμα συνάρτησης ReLU [35]



Τα πλεονεκτήματα της συνάρτησης είναι πως σε σύγκριση με της σιγμοειδείς/tanh συναρτήσεις μειώνει κατά 6 φορές τον ρυθμό σφαλμάτων σε ένα νευρωνικό δίκτυο. Επιπλέον λόγω της γραμμικής της φύσης απαιτεί πολύ λιγότερη υπολογιστική ισχύ για την εκπαίδευση ενός δικτύου. [36]

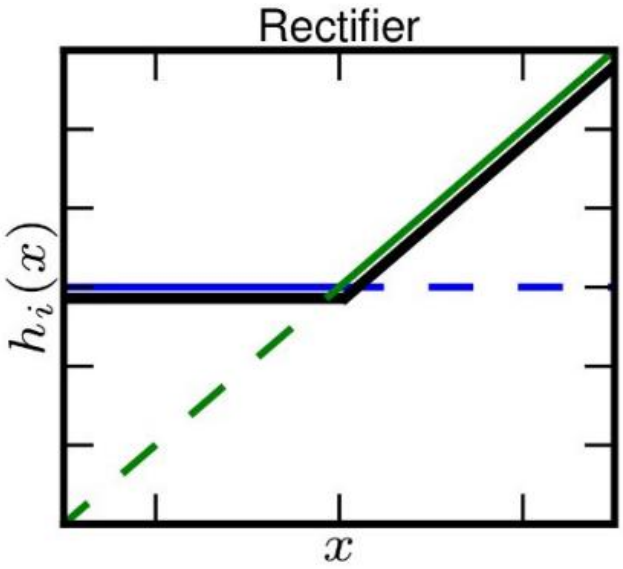
Εικόνα 30. Διαγραμματική απεικόνιση της ReLU (ενιαία γραμμή) και της Tanh (διακεκομμένη γραμμή) όπου απεικονίζεται η ταχύτητα μείωσης του ρυθμού σφαλμάτων κατά την εκπαίδευση του δικτύου [36]



4.1.4 Συνάρτηση Maxout

Η συνάρτηση **maxout** η οποία δημιουργήθηκε από τον Ian Goodfellow αποτελεί μια γενίκευση της συνάρτησης ReLU και είναι της μορφής: **$\max(w^T_1 x+b_1, w^T_2 x+b^2)$ [37]**. Αντίθετα από την ReLU όπου υπολογίζεται το μέγιστο από το 0 και την συνάρτηση, στην maxout χρησιμοποιούνται οι διπλάσιες παράμετροι για να αποφευχθούν προβλήματα μη ενεργοποίησης του νευρώνα. [37]

Εικόνα 31. Απεικόνιση της εφαρμογής της συνάρτησης maxout σε μία συνάρτηση ReLU [37]

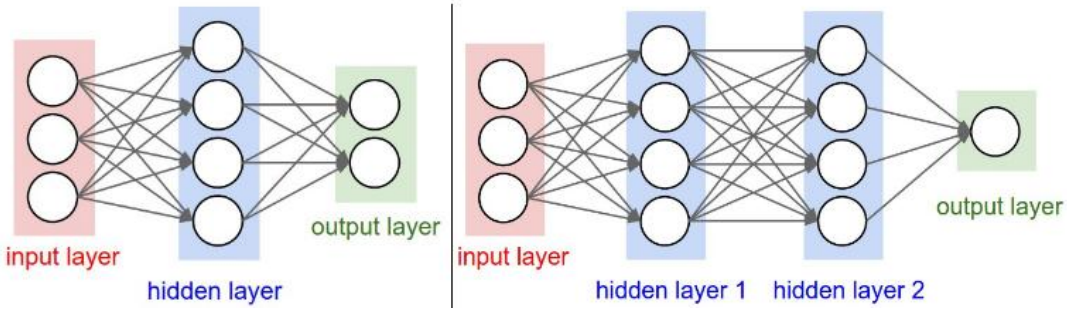


4.2 Πολυεπίπεδοι Αισθητήρες (Multi-layer Perceptrons)

Εφόσον είδαμε επιγραμματικά πως λειτουργεί ένας αισθητήρας γίνεται άμεσα κατανοητό πως για να μπορέσουμε να τους χρησιμοποιήσουμε αποτελεσματικά θα πρέπει να τους οργανώσουμε σε ένα νευρωνικό δίκτυο. Όπως και στον ανθρώπινο εγκέφαλο, έτσι και στην κατασκευή τεχνητών νευρωνικών δικτύων, οι νευρώνες χωρίζονται σε επίπεδα ούτως ώστε ορισμένοι αισθητήρες να μπορούν να εξάγουν σήματα τα οποία θα λαμβάνονται ως δεδομένα εισόδου από το επόμενο επίπεδο. Κάθε επίπεδο αποτελείται από νευρώνες οι οποίοι δεν συνδέονται μεταξύ τους, αλλά όλοι είναι συνδεδεμένοι με τους νευρώνες του επόμενου επιπέδου τους οποίους τροφοδοτούν. Σε αυτό τον σχηματισμό που αποκαλούμε ως **πολυεπίπεδους αισθητήρες (multi-layer perceptron)** ή **feed-forward neural networks** τα δεδομένα δεν επιστρέφουν μέσω ανάδρασης στα αρχικά επίπεδα για επανεπεξεργασία αλλά οι πληροφορίες μεταφέρονται μονάχα προς μία κατεύθυνση.

Η ονομασία ενός Ν-επίπεδου νευρωνικού δικτύου εξαρτάται άμεσα από το πόσα κρυφά επίπεδα υπάρχουν μετά το αρχικό, το οποίο περιέχει τα δεδομένα εισόδου. Για να ονομάσουμε ένα νευρωνικό δίκτυο δεν λαμβάνουμε υπόψη μας το πρώτο επίπεδο. Για παράδειγμα, ένα μονού-επιπέδου νευρωνικό δίκτυο δεν έχει κάποιο κρυφό επίπεδο να παρεμβάλλεται ανάμεσα στο επίπεδο εισόδου και στο επίπεδο εξόδου. Παράλληλα, το επίπεδο εξόδου δεν περιλαμβάνει στους αισθητήρες του κάποια συνάρτηση επεξεργασίας των δεδομένων γιατί σε αυτό το επίπεδο συνήθως γίνεται η ταξινόμηση των αρχικών μας δεδομένων.

Εικόνα 32. Παραδείγματα τεχνητών νευρωνικών δικτύων 2 και 3 επιπέδων [88]



Γενικά, τα δύο χαρακτηριστικά τα οποία χρησιμοποιούνται συνήθως για να ορίσουμε το μέγεθος ενός τεχνητού νευρωνικού δικτύου είναι ο αριθμός των νευρώνων, ή ο αριθμός των παραμέτρων. Με βάση την παραπάνω εικόνα το $1^{ο}$ νευρωνικό δίκτυο αποτελείται από 2 επίπεδα και 6 νευρώνες (καθώς το πρώτο επίπεδο δεν λαμβάνεται υπόψιν) και από (3*4) + (4*2) = 20 βάρη και 4+2 = 6 biases. Άρα αποτελείται συνολικά από 26 παραμέτρους εκμάθησης.

Εναλλακτικά, θα μπορούσαμε να πούμε πως ο στόχος ενός τέτοιου δικτύου είναι η προσέγγιση μιας συνάρτησης f*, όπως π.χ. στην περίπτωση ενός ταξινομητή όπου η συνάρτηση y = f*(x) χρησιμοποιείται για την ταξινόμηση ενός δεδομένου x σε μία κατηγορία y. Στην περίπτωση ενός feedforward δικτύου ορίζουμε ένα y = f(x;θ) , όπου το δίκτυο προσπαθεί να μάθει τις καλύτερες τιμές των παραμέτρων θ, οι οποίες θα έχουν ως αποτέλεσμα την καλύτερη προσέγγιση της επιθυμητής συνάρτησης.

Για να απεικονίσουμε καλύτερα την μορφή δικτύου θα μπορούσαμε να ορίσουμε το f(x) ως μία αλυσίδα τριών συναρτήσεων $f^{(1)}$, $f^{(2)}$, $f^{(3)}$ όπου κάθε συνάρτηση απεικονίζει ένα επίπεδο του δικτύου και ως αποτέλεσμα έχουμε $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$, όπου $f^{(1)}$ είναι το $1^{ο}$ επίπεδο, $f^{(2)}$ το $2^{ο}$ επίπεδο και ούτω καθεξής. Κατά την εκπαίδευση του δικτύου προσπαθούμε να επιτύχουμε f(x) = f*(x). Τα δεδομένα τα οποία εξάγονται από το κάθε επίπεδο μας προσφέρουν προσεγγιστικά αποτελέσματα της f*(x), για διαφορετικά σημεία εκπαίδευσης x μέσα στο δίκτυο, όπου για κάθε x έχουμε y ≈ f*(x). Σκοπός αυτών των αποτελεσμάτων είναι να ορίσουν απευθείας την συμπεριφορά που θα πρέπει να έχει το output layer του δικτύου για κάθε σημείο x, η οποία θα είναι να παράγει μία τιμή η οποία θα πρέπει να προσεγγίζει ένα αντίστοιχο y. Η συμπεριφορά των υπόλοιπων δικτύων πέραν του τελευταίου δεν καθορίζεται απευθείας από τα δεδομένα εκπαίδευσης που δίνουμε στο δίκτυο, αλλά με βάση τον αλγόριθμο εκμάθησης που χρησιμοποιείται, θα πρέπει να αποφασιστεί πως θα συμπεριφερθεί το κάθε δίκτυο ώστε να εξάγει τα επιθυμητά αποτελέσματα ώστε να επιτευχθεί η καλύτερη δυνατή προσέγγιση του f*(x). Ακριβώς επειδή τα δεδομένα εκπαίδευσης δεν καθορίζουν το επιθυμητό αποτέλεσμα του κάθε επιπέδου, αποκαλούνται **κρυφά επίπεδα (hidden layers)**. [91]

Παρότι η γενικότερη ιδέα των νευρωνικών δικτύων ξεκίνησε ως μία προσπάθεια προσέγγισης του τρόπου λειτουργίας των νευρώνων σε ένα νευρικό σύστημα, οι πρακτικές εφαρμογές και ο τρόπος σχεδιασμού των μοντέρνων νευρωνικών δικτύων μας οδηγούν στο συμπέρασμα πως είναι σωστότερο να θεωρήσουμε τα feedforward δίκτυα ως μηχανές προσεγγιστικών συναρτήσεων, οι οποίες είναι σχεδιασμένες ούτως ώστε να επιτύχουν στατιστικές γενικεύσεις για τα δεδομένα εισόδου τους χρησιμοποιώντας χαρακτηριστικά από τον τρόπο λειτουργίας ενός νευρικού συστήματος, παρά ως ενός μοντέλου απεικόνισης της λειτουργίας ενός εγκεφάλου.

Όπως είδαμε προηγουμένως, για να μπορέσουμε να κατανοήσουμε τον τρόπο αλληλεπίδρασης δύο μεταβλητών εισόδου στο δίκτυο δεν μπορούμε να χρησιμοποιήσουμε γραμμικά μοντέλα νευρωνικών δικτύων και προσπαθούμε να εφαρμόσουμε μοντέλα γραμμικότητας όχι στο δεδομένο x απευθείας αλλά σε ένα μετασχηματισμό του που θα ορίσουμε ως φ(x), όπου φ είναι η συνάρτηση μη-γραμμικού μετασχηματισμού του x. Θα μπορούσαμε να θεωρήσουμε πως το φ παρέχει μία ομάδα χαρακτηριστικών τα οποία περιγράφουν το x ή ως ένα διαφορετικό τρόπο απεικόνισης του x.

Το πρόβλημα που παρουσιάζεται είναι πως θα διαλέξουμε το κατάλληλο φ, για το δίκτυο. Πλέον, με την ανάπτυξη μεθόδων deep learning μπορούμε να εκπαιδεύσουμε το δίκτυο ούτως ώστε να βρει το φ, κάτι που παλαιότερα απαιτούσε δεκαετίες ανθρώπινης προσπάθειας για κάθε ξεχωριστό θέμα που προέκυπτε και ειδικούς από κάθε τομέα ειδίκευσης, όπως αναγνώριση ομιλίας. Με βάση την παραπάνω προσέγγιση καταλήγουμε στο μοντέλο $\mathbf{y=f(x;\theta, w) =\varphi(x;\theta)^{T}w}$ **[91]**. Ως **θ** ορίζουμε τις παραμέτρους τις οποίες χρησιμοποιούμε για να καταλήξουμε στο **φ** από όλες τις πιθανές συναρτήσεις και τις παραμέτρους **w** οι οποίες αντιστοιχούν το κάθε φ(**x**) στο επιθυμητό αποτέλεσμα. Με το παραπάνω μοντέλο μπορούμε να ορίσουμε ένα feedforward δίκτυο όπου κάθε φ αντιστοιχεί και σε ένα κρυφό επίπεδο του νευρωνικού δικτύου. Για να μπορέσουμε να βελτιώσουμε τον συγκεκριμένο τρόπο προσέγγισης θα μπορούσαμε να χρησιμοποιήσουμε και άτομα ειδικά στα θέματα που θέλουμε να εξετάσουμε τα οποία θα μπορούσαν να σχεδιάσουν ποικίλα **φ(x;θ)** τα οποία θα πρέπει σύμφωνα με την

προϋπάρχουσα γνώση να αποφέρουν προσεγγιστικά καλά αποτελέσματα. Το πλεονέκτημα αυτής της μεθόδου είναι πως ο σχεδιαστής ενός τέτοιου μοντέλου αρκεί να βρει την σωστή «οικογένεια» συναρτήσεων που θα αποδίδουν καλά σε ένα δίκτυο και μέσω της εκπαίδευσης του δικτύου να γίνει η προσέγγιση της κατάλληλης συνάρτησης για κάθε επίπεδο ώστε να επιτύχουμε το βέλτιστο y. [91]

5. Συνελικτικά Νευρωνικά δίκτυα (CNN)

Τα συνελικτικά νευρωνικά δίκτυα ή CNNs [38], είναι μια εξειδικευμένη μορφή νευρωνικών δικτύων τα οποία εξειδικεύονται σε δεδομένα εικόνων π.χ. ως 2-διάστατες απεικονίσεις pixel. Ονομάζονται συνελικτικά δίκτυα γιατί εφαρμόζουν μία μαθηματική γραμμική διαδικασία γνωστή ως **συνελιγμό (convolution)**. Συγκεκριμένα, ως συνελικτικά networks ορίζονται τα νευρωνικά δίκτυα τα οποία αντί να χρησιμοποιήσου πολλαπλασιασμό πινάκων χρησιμοποιούν την διαδικασία συνελιγμού σε τουλάχιστον ένα από τα επίπεδα που τα απαρτίζουν.

5.1 Συνελιγμός

Στην πιο γενική του μορφή ο συνελιγμός είναι μία διαδικασία η οποία εφαρμόζεται σε δύο συναρτήσεις ενός πραγματικού προβλήματος. Για παράδειγμα, έστω ότι επιθυμούμε να εντοπίσουμε την θέση ενός αντικειμένου που βρίσκεται σε κίνηση, με ένα αισθητήρα. Ο αισθητήρας μας δίνει ένα αποτέλεσμα x(t) όπου απεικονίζει την θέση του αντικειμένου την χρονική στιγμή t. Τόσο το x όσο και το t έχουν πραγματικές τιμές, και έτσι για κάθε διαφορετική χρονική στιγμή δεχόμαστε μία διαφορετική τιμή για την θέση του αντικειμένου. Σε περίπτωση όπου ο αισθητήρας επιστρέφει αποτελέσματα με θόρυβο (noise) θα πρέπει να χρησιμοποιήσουμε τον μέσο όρο από διάφορες μετρήσεις για να μπορέσουμε να εντοπίσουμε την θέση του αντικειμένου χωρίς την παρεμβολή θορύβου. Εφόσον, οι πιο πρόσφατες μετρήσεις μας θα είναι οι πιο αντιπροσωπευτικές, θα χρησιμοποιήσουμε συντελεστές βάρους για να μπορέσουμε να εστιάσουμε στις πιο πρόσφατες μετρήσεις. Για να το επιτύχουμε αυτό θα εφαρμόσουμε μία συνάρτηση συντελεστή βάρους w(a), όπου ως a ορίζουμε το πόσο πρόσφατη είναι η κάθε μέτρηση. Αν σε κάθε χρονική στιγμή βρούμε τον σταθμισμένο μέσο όρο θα έχουμε μία καινούρια συνάρτηση **s** η οποία θα μας δώσει μία πιο ακριβή πρόβλεψη για την θέση που βρίσκεται το αντικείμενο. [91]

Επομένως ως **συνελιγμό** ορίζουμε:

$$s(t) = (x * w)(t)$$ **[91]**

Σε γενικές γραμμές, ως συνελιγμός ορίζεται οποιαδήποτε συνάρτηση της παραπάνω μορφής και δεν περιορίζεται μόνο σε περιπτώσεις σταθμισμένων μέσων. Συνήθως, σε περιπτώσεις CNNs η πρώτη συνάρτηση (x(t)) αποκαλείται δεδομένο εισόδου (input) και η δεύτερη (w(a)) ορίζεται ως πυρήνας (kernel). Επίσης το αποτέλεσμα (s(t)) ορίζεται ως χάρτης χαρακτηριστικών (feature map).

Σε συνέχεια του παραδείγματος προς εξέταση θα θεωρηθεί για λόγους αληθοφάνειας πως ο αισθητήρας, σε αντιστοιχία με τα δεδομένα που θα είχαμε στη διάθεσή μας, δέχεται δεδομένα ανά συγκεκριμένα χρονικά διαστήματα π.χ. μία μέτρηση ανά δευτερόλεπτο. Επομένως το t μας δέχεται μόνο ακέραιες τιμές. Συνεπώς, αν θεωρήσουμε πως τα x και w ορίζονται μόνο με βάση έναν ακέραιο t έχουμε:

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a)$$ **[91]**

Γενικότερα, σε εφαρμογές μηχανικής μάθησης, ως δεδομένα εισόδου έχουν πολυδιάστατους πίνακες δεδομένων, και ως πυρήνα έχουν πολυδιάστατους πίνακες με παραμέτρους που χρησιμοποιούνται από τον αλγόριθμο εκπαίδευσης. Αυτοί οι πολυδιάστατοι πίνακες συχνά αποκαλούνται και **tensors.** Κάθε στοιχείο των δεδομένων εισόδου αλλά και του πυρήνα πρέπει να αποθηκευτεί ξεχωριστά και για αυτό τον λόγο θεωρούμε πως για κάθε

συνάρτηση επιστρέφει μηδενικό αποτέλεσμα, εκτός από τα ακριβή σημεία στα οποία αποθηκεύουμε τις τιμές των πινάκων. Αυτό σημαίνει πως από την προηγούμενη συνάρτηση μπορούμε πρακτικά να ορίσουμε το άπειρο άθροισμα της προηγούμενης εξίσωσης ως το άθροισμα πλήθους όσο και το πλήθος των στοιχείων των πινάκων. Επιπλέον μπορούμε να εφαρμόσουμε συνελιγμούς σε πάνω από μία διάσταση. [91]

Επομένως, αν θεωρήσουμε ως την δισδιάστατη εικόνα **I** ως το δεδομένο εισόδου μας και τον δισδιάστατο πυρήνα **K** έχουμε:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(m, n)K(i - m, j - n) \text{ [91]}$$

Αντίστοιχα, καθώς η συνάρτηση συνελιγμού μπορεί να μετασχηματιστεί μπορούμε να μετατρέψουμε την παραπάνω συνάρτηση ως εξής:

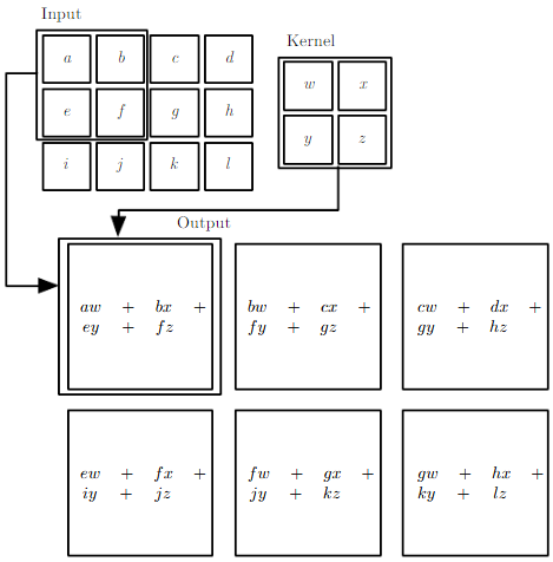
$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i - m, j - n)K(m,n) \text{ [91]}$$

Συνήθως η δεύτερη μορφή της συγκεκριμένης εξίσωσης είναι ευκολότερο να χρησιμοποιηθεί σε μία βιβλιοθήκη μηχανικής μάθησης, καθώς υπάρχει μικρό πλήθος τιμών ανάμεσα σε m και n. Αντίστοιχα σε πολλές βιβλιοθήκες νευρωνικών δικτύων χρησιμοποιείται και η συνάρτηση **cross-correlation** η οποία είναι η ίδια με την συνάρτηση συνελιγμού, αλλά χωρίς την αντίστροφη σχέση των τιμών εισόδου - πυρήνα, και απεικονίζεται ως εξής:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m,n) \text{ [91]}$$

Σε πολλές περιπτώσεις μπορούμε να παρατηρήσουμε πως η cross-correlation χαρακτηρίζεται ως συνελικτική, αλλά διευκρινίζεται αν αντιστρέφεται ο πυρήνας ή όχι ώστε να ξεχωρίζουν.

Εικόνα 33. Παράδειγμα δισδιάστατου συνελιγμού tensor χωρίς αντιστροφή πυρήνα. Περιορίζουμε τις τιμές εξόδου μόνο στις περιπτώσεις όπου ο πυρήνας αντιστοιχεί σε μία θέση του πίνακα εισόδου και χρησιμοποιούμε τα βέλη για να αντιστοιχίσουμε το αποτέλεσμα του πολλαπλασιασμού των πινάκων της πάνω αριστερά περιοχής του πίνακα εισόδου. [91]



5.2 Βασικές Έννοιες Συνελικτικών Δικτύων

Η θεωρία των **συνελικτικών neural networks** ενσωματώνει τρείς βασικές έννοιες για την βελτίωση των μοντέλων μηχανικής μάθησης στα οποία εφαρμόζεται.

Οι έννοιες αυτές είναι:

1) **Αραιές Αλληλεπιδράσεις (Sparse Interactions)**

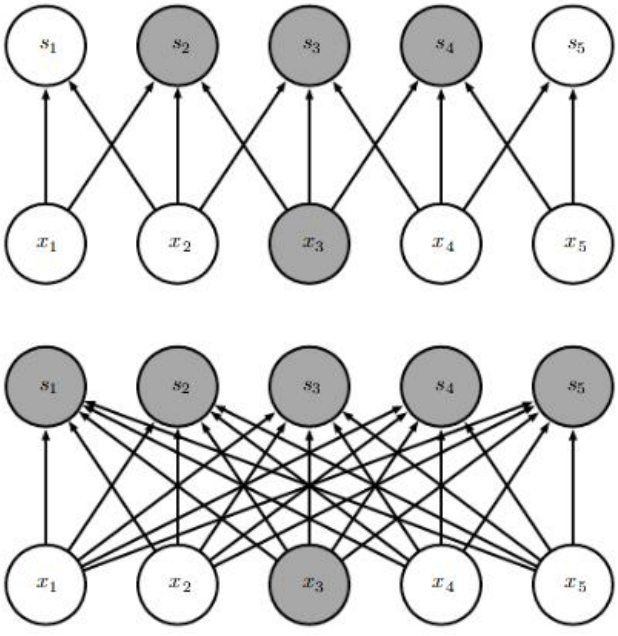
2) **Κοινή Χρήση Παραμέτρων (Parameter Sharing)**

3) **Εξισωτικές Αναπαραστάσεις (Equivariant Represantations)**

5.2.1 Αραιές Αλληλεπιδράσεις (Sparse Interactions)

Τα παραδοσιακά επίπεδα νευρωνικών δικτύων χρησιμοποιούν την μέθοδο πολλαπλασιασμού πινάκων μεταξύ ενός πίνακα παραμέτρων και μίας ξεχωριστής παραμέτρου, η οποία θα περιγράφει την αλληλεπίδραση μεταξύ των δεδομένων εισόδου και του αποτελέσματος. Συνεπώς, σε μια τέτοια δομή ενός νευρωνικού δικτύου κάθε δεδομένο εισόδου αλληλεπιδρά με κάθε δεδομένο εξόδου. Αυτές οι αλληλεπιδράσεις σε συνελικτικά νευρωνικά δίκτυα είναι πολύ πιο αραιές συνήθως, εξηγώντας έτσι τον όρο των **sparse interactions**, το οποίο επιτυγχάνεται έχοντας πολύ μικρότερο tensor πυρήνα από ότι tensor εισόδου. Για παράδειγμα, σε θέματα αναγνώρισης προσώπων, η εικόνα εισόδου μπορεί να αποτελείται από χιλιάδες pixel, αλλά με την εφαρμογή kernels οι οποίοι μπορεί να αποτελούνται από δεκάδες pixel μπορούμε να εξάγουμε τα επιθυμητά χαρακτηριστικά, όπως η ύπαρξη άκρων στην εικόνα. Αυτό έχει ως αποτέλεσμα την αποθήκευση λιγότερων παραμέτρων οι οποίες μπορούν να μειώσουν τόσο τις ανάγκες πόρων του δικτύου, αλλά και την βελτίωση των αποτελεσμάτων με λιγότερη επεξεργασία των δεδομένων. Σε περίπτωση όπου έχουμε m δεδομένα εισόδου και n εξόδου θα πρέπει για τον πολλαπλασιασμό των πινάκων να έχουμε m*n παραμέτρους, ενώ σε περίπτωση όπου μειώσουμε τον αριθμό των δεδομένων που πρέπει να αντιστοιχηθούν με τον πίνακα n σε k, τότε θα χρειαστούμε μονάχα k*n παραμέτρους για να εξάγουμε τα αποτελέσματα μας. [91]

Εικόνα 34. Συγκριτικό παράδειγμα συνελικτικού νευρωνικού δικτύου (πάνω) και κλασικού νευρωνικού δικτύου (κάτω) με χρήση πολλαπλασιασμού πινάκων. Για το δεδομένο εισόδου x_3 στο οποίο χρησιμοποιείται πυρήνας πλάτους 3, μόνο τρία δεδομένα εξόδου επηρεάζονται από το x λόγω των αραιών αλληλεπιδράσεων. Αντίθετα, σε περιπτώσεις πολλαπλασιασμού πινάκων όλα τα δεδομένα εξόδου επηρεάζονται από το x. [91]

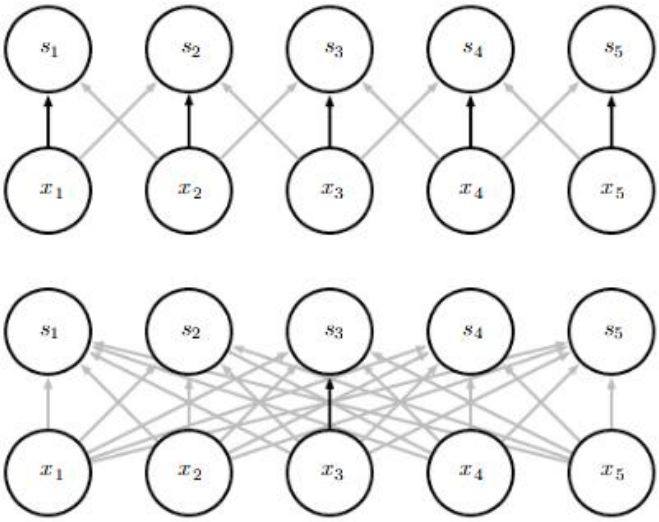


5.2.2 Κοινή Χρήση Παραμέτρων (Parameter Sharing)

Ως parameter sharing αναφερόμαστε στην χρήση των ίδιων παραμέτρων παραπάνω από μία φορά στο μοντέλο νευρωνικού δικτύου που χρησιμοποιούμε, σε αντίθεση με περιπτώσεις

παραδοσιακών νευρωνικών δικτύων, όπου κάθε στοιχείο ενός πίνακα με βάρη χρησιμοποιείται μονάχα μία φορά για να γίνει ο υπολογισμός των δεδομένων εξόδου του συγκεκριμένου επιπέδου στο οποίο εφαρμόζεται. Θα μπορούσαμε, συμπερασματικά, να θεωρήσουμε πως τα βάρη που χρησιμοποιεί ένα συνελικτικό νευρωνικό δίκτυο είναι αλληλένδετα καθώς η τιμή ενός βάρους μπορεί να επαναχρησιμοποιηθεί για τον υπολογισμό ενός δεδομένου εξόδου. Συγκεκριμένα, όπως είδαμε παραπάνω ο πίνακας πυρήνα χρησιμοποιείται σε κάθε θέση του πίνακα εισόδου, το οποίο σημαίνει πως αντί να γίνεται εύρεση των απαραίτητων παραμέτρων σε κάθε θέση έχουμε μόνο ένα σετ από παραμέτρους, εξοικονομώντας κατά αυτό τον τρόπο τις απαιτήσεις αποθηκευτικού χώρου του δικτύου σε k παραμέτρους, όπου το k είναι αρκετές φορές μικρότερο από τον αρχικό πίνακα m. Παρ' όλα αυτά σε περιπτώσεις εφαρμογών αναγνώρισης προσώπων, όπου θέλουμε να εφαρμόσουμε διαφορετικές συναρτήσεις στο πάνω μέρος της εικόνας και στο κάτω μέρος της εικόνας, για να εντοπίσουμε τα μάτια και το στόμα αντίστοιχα, δεν θα χρησιμοποιήσουμε κοινές παραμέτρους γιατί θέλουμε να εξάγουμε διαφορετικά χαρακτηριστικά σε διαφορετικές θέσεις στην εικόνα. [91]

Εικόνα 35. Συγκριτικό παράδειγμα μοντέλου κοινής χρήσης παραμέτρων. Τα μαύρα βέλη απεικονίζουν τις συνδέσεις που χρησιμοποιούν την ίδια παράμετρο σε δύο διαφορετικά μοντέλα. Στο πάνω διάγραμμα τα μαύρα βέλη απεικονίζουν την χρήση της κεντρικής παραμέτρου ενός πυρήνα με 3 στοιχεία σε ένα μοντέλου συνελικτικού νευρωνικού δικτύου. Επειδή υπάρχει κοινή χρήση παραμέτρων στο συγκεκριμένο μοντέλο χρησιμοποιείται η ίδια παράμετρος για κάθε δεδομένο εισόδου. Αντίθετα, στο κάτω μοντέλο νευρωνικού δικτύου (fully connected) χρησιμοποιείται η κεντρική τιμή του πίνακα που περιέχει τα βάρη, μόνο μία φορά καθώς δεν υπάρχει parameter sharing. [91]



5.2.3 Εξισωτικές Αναπαραστάσεις (Equivariant Represantations)

Σε εφαρμογές συνελικτικών νευρωνικών δικτύων, η κοινή χρήση παραμέτρων έχει ως αποτέλεσμα να υπάρχει για το αντίστοιχο επίπεδο του δικτύου **ισοδύναμη αντιστοιχία (equivariance)** μεταξύ των δεδομένων εισόδου και εξόδου. Συνεπώς, κάθε μεταβολή σε ένα δεδομένο εισόδου αντιστοιχεί σε αντίστοιχη μεταβολή του αποτελέσματος εξόδου. Αν θεωρήσουμε μία συνάρτηση f(x) που χρησιμοποιείται σε ένα επίπεδο του δικτύου, υπάρχει equivariance με μία συνάρτηση g όταν f(g(x)) = g(f(x)) [91]. Σε ένα αντίστοιχο παράδειγμα εφαρμογής εικόνων, ας θεωρήσουμε την συνάρτηση I η οποία περιλαμβάνει τιμές φωτεινότητας της εικόνας σε συγκεκριμένες συντεταγμένες, και την συνάρτηση g η οποία αντιστοιχεί μία συνάρτηση εικόνας σε μία άλλη συνάρτηση ώστε, I' = g(I) οπού ισχύει I'(x,y) = I(x-1,y) μετατοπίζοντας έτσι κάθε pixel του I κατά μία μονάδα. Αν εφαρμόσουμε την μέθοδο συνελιγμού στην I' και μετασχηματίσουμε με την χρήση της g τα δεδομένα εξόδου, θα έχουμε το ίδιο αποτέλεσμα με το να εφαρμόζαμε τον μετασχηματισμό στην συνάρτηση I και στη συνέχεια

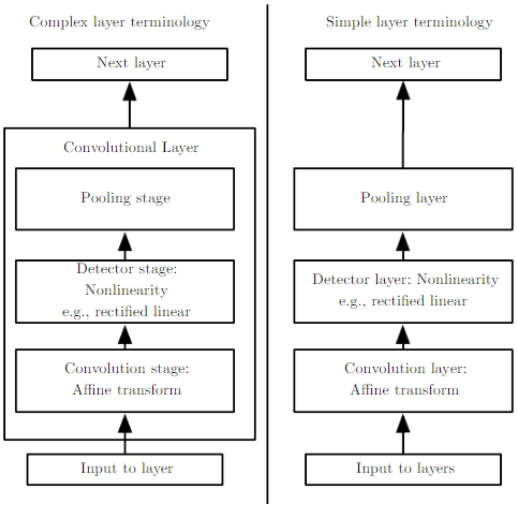
εφαρμόζαμε συνελιγμό. Η συγκεκριμένη ιδιότητα είναι πολύ χρήσιμη καθώς μπορούμε να γνωρίζουμε σε ποια σημεία παρουσιάζονται συγκεκριμένα χαρακτηριστικά ανάλογα με τα αντίστοιχα δεδομένα εισόδου. [91]

Εικόνα 36. Παράδειγμα αποδοτικότητας ανίχνευσης άκρων σε μία εικόνα. Η δεξιά εικόνα σχηματίστηκε με την χρήση κάθε pixel της αριστερής εικόνας και την αφαίρεση , στη συνέχεια, κάθε γειτονικού pixel στα αριστερά. Το μέγεθος της αριστερής εικόνας είναι 320x280 pixel ενώ η δεξιά αποτελείται από 319x280 pixel. Με την χρήση ενός μοντέλου συνελικτικού δικτύου ο μετασχηματισμός της εικόνας μπορεί να περιγράφει με ένα πυρήνα δύο στοιχείων και απαιτεί 319 * 280 * 3 = 267.960 επεξεργασίες (δύο πολλαπλασιασμούς και μία πρόσθεση ανά pixel). Αντίστοιχα, για να γίνει η ίδια διαδικασία με πολλαπλασιασμό πινάκων θα χρειαζόταν 320 * 280 * 319 * 280 = 8.003.072.000 στοιχεία. [91]

5.3 Συγκέντρωση (Pooling)

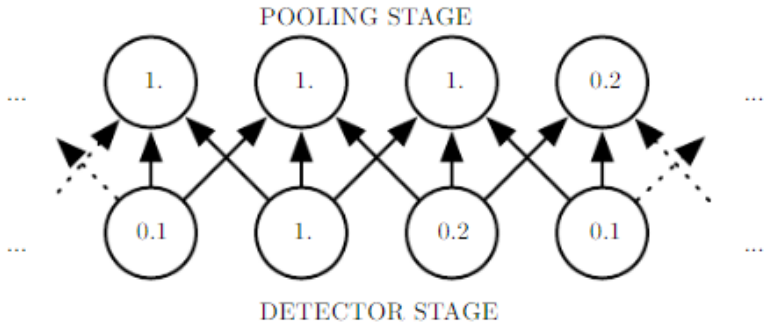
Η τυπική περίπτωση ενός επιπέδου που περιέχεται σε ένα CNN αποτελείται από τρία στάδια επεξεργασίας. Στο πρώτο στάδιο, το επίπεδο πραγματοποιεί παράλληλα ορισμένους συνελιγμούς ώστε να παράγει μία ομάδα γραμμικών απεικονίσεων. Στο δεύτερο στάδιο, οι γραμμικές αυτές απεικονίσεις μετασχηματίζονται μέσα από μία μη-γραμμική συνάρτηση ενεργοποίησης. Το στάδιο αυτό αποκαλείται και ως **στάδιο ανίχνευσης (detector stage)**. Στο τρίτο και τελευταίο στάδιο, χρησιμοποιείται μία **συνάρτηση συγκέντρωσης (pooling function)**, ούτως ώστε να επεξεργαστούμε σε μεγαλύτερο βαθμό τα δεδομένα εξόδου του συγκεκριμένου επιπέδου του δικτύου.

Εικόνα 37. Διαγραμματικές απεικονίσεις ενός τυπικού CNN. Υπάρχουν δύο τρόποι απεικόνισης συνελικτικών νευρωνικών δικτύων. Αριστερά, μπορούμε να απεικονίσουμε το CNN ως ένα αριθμό σχετικά πολύπλοκων επιπέδων, όπου κάθε επίπεδο αποτελείται από διαφορετικά στάδια. Σε αυτή την απεικόνιση, υπάρχει αντιστοίχιση ένα-προς-ένα μεταξύ των tensor πυρήνα και των επιπέδων του δικτύου. Δεξιά, το δίκτυο απεικονίζεται ως ένας μεγαλύτερος αριθμός απλούστερων επιπέδων, όπου κάθε βήμα επεξεργασίας αντιστοιχεί σε ένα επίπεδο. Αυτό έχει ως αποτέλεσμα να μην υπάρχουν παράμετροι για κάθε επίπεδο του δικτύου. [91]



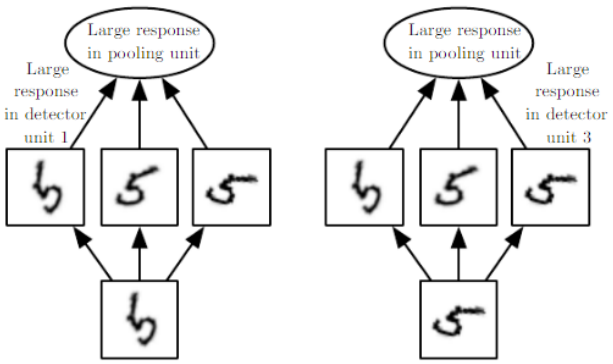
Μία συνάρτηση συγκέντρωσης αντί να επιστρέφει τα δεδομένα εξόδου του δικτύου σε μία συγκεκριμένη θέση, επιστρέφει το στατιστικό άθροισμα των γειτονικών δεδομένων εξόδου. Ένα παράδειγμα μιας συνάρτησης συγκέντρωσης είναι η συνάρτηση **μέγιστης συγκέντρωσης (max pooling)** [39] η οποία επιστρέφει την μέγιστη τιμή εξόδου από μία ομάδα γειτονικών δεδομένων σε σχήμα παραλληλόγραμμου. Αντίστοιχες, συναρτήσεις συγκέντρωσης υπολογίζουν τον μέσο όρο σε μία ομάδα δεδομένων ή τον σταθμισμένο μέσο με βάση την απόσταση ενός κεντρικού σημείου.

Εικόνα 38. Παράδειγμα εφαρμογής max pooling σε ένα επίπεδο ενός νευρωνικού δικτύου. Στο στάδιο ανίχνευσης βλέπουμε τα αποτελέσματα της εφαρμογής της μη-γραμμικής συνάρτησης. Στο στάδιο της συγκέντρωσης βλέπουμε το αποτέλεσμα της εφαρμογής μέγιστης συγκέντρωσης ανά pixel με πλάτος συγκέντρωσης 3 pixel. [91]



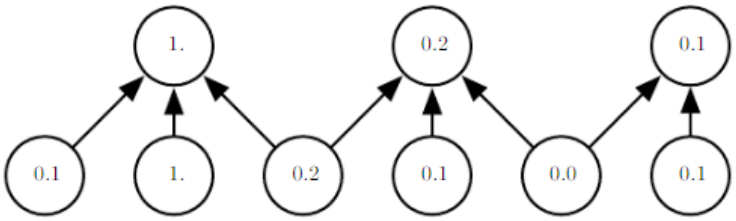
Η χρήση συναρτήσεων συγκέντρωσης επιτυγχάνει τον αδιάβλητο μετασχηματισμό των δεδομένων, συνεπώς ακόμα και αν υπάρξει μείωση δεδομένων εισόδου το αποτέλεσμα εξόδου θα παραμείνει αμετάβλητο. Με αυτό τον τρόπο μπορούμε να εξάγουμε χρήσιμα αποτελέσματα αν μας ενδιαφέρει να γνωρίζουμε την ύπαρξη ενός χαρακτηριστικού στα δεδομένα μας, παρά την ακριβή θέση στην οποία εντοπίζεται το συγκεκριμένο χαρακτηριστικό. Σε θέματα αναγνώρισης προσώπων, δεν είναι απαραίτητο να γνωρίζουμε την ακριβή θέση του κάθε ματιού, καθώς μας αρκεί να γνωρίζουμε πως σε κάθε μεριά του προσώπου μπορούμε να εντοπίσουμε από ένα μάτι. Σε περίπτωση όμως, που θέλουμε να εντοπίσουμε το σημείο στο οποίο συναντώνται δύο άκρες σε μία εικόνα θα πρέπει να έχουμε διατηρήσει την ακριβή θέση του συγκεκριμένου σημείου. Εναλλακτικά, μπορούμε να χρησιμοποιήσουμε μεθόδους pooling σε ορισμένα δεδομένα εξόδου ώστε να πετύχουμε την αμεταβλησία των δεδομένων μας. [91]

Εικόνα 39. Παράδειγμα εκμάθησης αμεταβλησίας σε ένα επίπεδο δικτύου. Με την χρήση φίλτρων και μίας μονάδας μέγιστης συγκέντρωσης μπορούμε να εξασφαλίσουμε την αμεταβλητότητα κατά την περιστροφή μιας εικόνας του αριθμού 5. [91]



Εφαρμόζοντας μεθόδους συγκέντρωσης (pooling) σε μία ολόκληρη περιοχή δεδομένων, και άρα χρησιμοποιώντας λιγότερες μονάδες συγκέντρωσης από ότι μονάδες ανίχνευσης όπως π.χ. το να χρησιμοποιηθεί το άθροισμα για περιοχές συγκέντρωσης που έχουν απόσταση k pixel αντί για 1 μόνο pixel, βελτιώνεται η αποδοτικότητα του επόμενου επιπέδου, καθώς θα δεχτεί λιγότερα δεδομένα εισόδου προς επεξεργασία.

Εικόνα 40. Παράδειγμα εφαρμογής pooling με μείωση δείγματος. Στο παρακάτω διάγραμμα χρησιμοποιούμε την μέθοδο max pooling με πλάτος συγκέντρωσης 3 ανά 2 pixel. Με αυτό τον τρόπο, επιτυγχάνεται η μείωση των επεξεργαστικών απαιτήσεων του επόμενου επιπέδου. [91]



Για εφαρμογές εικόνων, πολλές φορές κρίνεται απαραίτητη η χρήση μεθόδων pooling για να ταξινομηθούν εικόνες οι οποίες μπορεί να διαφοροποιούνται ανάλογα με το μέγεθος τους, ενώ το μέγεθος των δεδομένων εισόδου για το επίπεδο ταξινόμησης πρέπει να έχει ένα προκαθορισμένο μέγεθος. Μετατοπίζοντας το εύρος του pooling που γίνεται ανά εικόνα μπορεί να επιτευχθεί η εξομάλυνση των μεγεθών των εικόνων και να λαμβάνουμε τον ίδιο αριθμό στατιστικών αθροισμάτων ανεξαρτήτως του αρχικού μεγέθους της εικόνας.

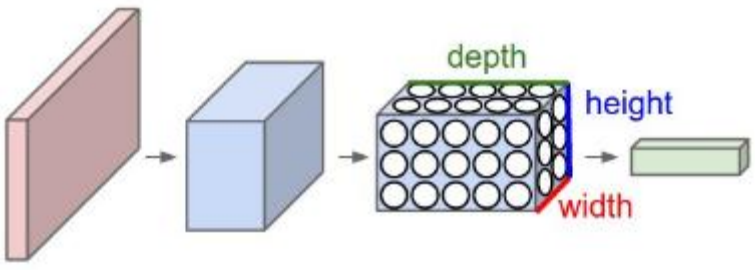
Για την εύρεση της ιδανικής μεθόδου συγκέντρωσης που θα πρέπει να εφαρμόσουμε μπορούμε να ανατρέξουμε σε προϋπάρχουσες μελέτες που αναλύουν τις κατάλληλες μεθόδους που μπορούν να εφαρμοστούν σε κάθε περίπτωση. [40]

5.4 Αρχιτεκτονική CNN για Αναγνώριση Εικόνων

Όπως είδαμε προηγουμένως τα Συνελικτικά Δίκτυα δημιουργήθηκαν για την επεξεργασία εικόνων καθώς έχουν χαρακτηριστικά τα οποία βοηθούν στην αποτελεσματικότερη επεξεργασία των δεδομένων σε σχέση με fully connected νευρωνικά δίκτυα. Στη συνέχεια, θα εξετάσουμε ποια είναι η αρχιτεκτονική που εφαρμόζεται σε αυτά τα δίκτυα και πως επιλύει τα προβλήματα που θα μπορούσαν να παρουσιαστούν αν είχαν μία κλασική δομή δικτύου.

Αρχικά, τα layers ενός συνελικτικού δικτύου οργανώνονται σε 3 διαστάσεις, κατά ύψος, κατά πλάτος και κατά βάθος. Αυτό μας διευκολύνει καθώς μπορούμε να αναλύσουμε μια εικόνα σύμφωνα με την θέση του pixel (μήκος, πλάτος) και σύμφωνα με το χρώμα του, χωρισμένα με βάση την κλίμακα RGB (βάθος). Επομένως, θα μπορούσαμε να έχουμε μια εικόνα η οποία να αποτελείται είναι μεγέθους 32x32x3, επομένως θα χρειαζόταν 3072 βάρη ο πρώτος νευρώνας του δικτύου. Σε περίπτωση όμως που έχουμε μια εικόνα μεγαλύτερης ανάλυσης π.χ. 200x200x3 θα χρειαζόμασταν 120.000 βάρη ανά νευρώνα, με αποτέλεσμα την έλλειψη αποδοτικότητας του δικτύου. Σε ένα CNN με την χρήση pooling και parameter sharing το επεξεργαστικό φορτίο του δικτύου θα ήταν σημαντικά μικρότερο. [88]

Εικόνα 41. Παράδειγμα CNN το οποίο χωρίζει τους νευρώνες σε 3 διαστάσεις ανά επίπεδο. Κάθε επίπεδο δέχεται ως είσοδο ένα 3-διάστατο όγκο δεδομένων και εξάγει ένα αντίστοιχο 3-διάστατο αποτέλεσμα. Στο πρώτο επίπεδο του παραδείγματος μπορούμε να συμπεράνουμε πως το ύψος και το πλάτος του θα είναι του μεγέθους των διαστάσεων της εικόνας και το βάθος του θα είναι 3 για το μπλε, κόκκινο, πράσινο κανάλι χρωμάτων της εικόνας. [88]



Για τον σχεδιασμό της αρχιτεκτονικής ενός CNN χρησιμοποιούνται τρεις διαφορετικοί τύποι κρυφών επιπέδων:

1) **Επίπεδο Συνελιγμού**

2) **Επίπεδο Pooling**

3) **Fully-Connected Επίπεδο**

Ας εξετάσουμε ένα παράδειγμα αρχιτεκτονικής ενός CNN όπου θα χρησιμοποιήσουμε ως δεδομένα για επεξεργασία εικόνες μεγέθους 32x32x3. Η αρχιτεκτονική ενός τέτοιου δικτύου θα είναι ως εξής:

Επίπεδο Εισόδου: Στο 1^{ο} επίπεδο θα περιλαμβάνονται τα pixel και το χρώμα της εικόνας όπως είδαμε παραπάνω. Σε αυτό το επίπεδο, συνήθης πρακτική είναι η χρήση διαστάσεων εικόνας η οποία να είναι διαιρέσιμη τουλάχιστον 2 φορές. Συνήθως χρησιμοποιούνται μεγέθη 32x32 , 64x64, 96x96 , 224x224 , 384x384 και 512x512.

Επίπεδο Συνελιγμού: Σε αυτό το επίπεδο θα χρησιμοποιήσουμε την τεχνική του συνελιγμού όπου μέσω του parameter sharing θα καταλήξουμε σε ένα όγκο δεδομένων 32x32x12, σε περίπτωση που χρησιμοποιήσουμε 12 φίλτρα.

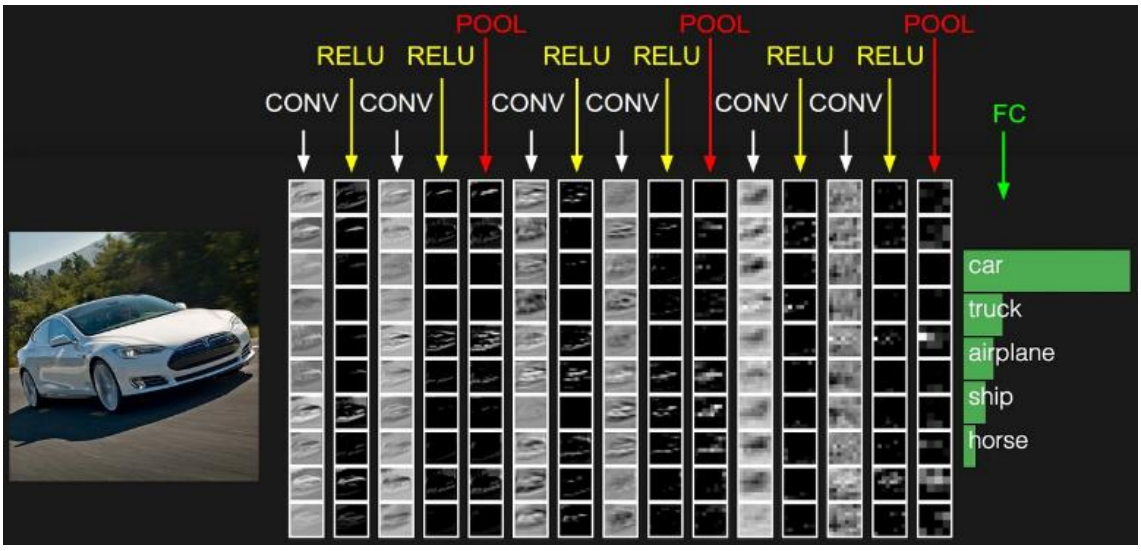
Relu Επίπεδο: Σε αυτό το επίπεδο θα χρησιμοποιήσουμε μια συνάρτηση ενεργοποίησης Relu, όπως π.χ η max(0,x) για τα δεδομένα του επιπέδου.

Pooling Επίπεδο: Χρησιμοποιώντας την διαδικασία pooling θα κάνουμε downsampling των διαστάσεων της εικόνας, κατά ύψος και πλάτος, και θα καταλήξουμε σε ένα όγκο δεδομένων 16x16x12.

Fully-Connected επίπεδο: Στο τελευταίο επίπεδο θα καταλήξουμε τα δεδομένα των κλάσεων προς ταξινόμηση και έτσι θα έχουμε έναν όγκο 1x1x10, εφόσον ορίζουμε πως έχουμε 10 κλάσεις για να ταξινομήσουμε την εικόνα.

Η παραπάνω μορφή είναι μία από τις πιο απλές αλλά και πιο διαδεδομένες αρχιτεκτονικές συνελικτικών νευρωνικών δικτύων όπου δέχονται ως είσοδο μια εικόνα και επιστρέφουν την κλάση στην οποία ανήκει ως αποτέλεσμα. [88]

Εικόνα 42. Παράδειγμα ενός τυπικού Συνελικτικού Νευρωνικού δικτύου 16 επιπέδων στα οποία εναλλάσσονται Συνελικτικά, Pooling και Relu επίπεδα, μέχρι να καταλήξουμε στο Fully Connected Layer στο οποίο γίνεται η ταξινόμηση της αρχικής εικόνας. [88]



5.5 Διαδεδομένες αρχιτεκτονικές CNN

LeNet: Οι πρώτες επιτυχημένες αρχιτεκτονικές CNN εφαρμόστηκαν την δεκαετία του 1990 από τον Yann LeCun. Η πιο διαδεδομένη από αυτές είναι η αρχιτεκτονική LeNet η οποία χρησιμοποιήθηκε για την αναγνώριση χειρόγραφων αριθμών. [41]

AlexNet: Το AlexNet τα οποίο αναπτύχθηκε από τους Alex Krizhevsky, Ilya Sutskever και Geoff Hinton είναι από τα πρώτα τα οποία αύξησαν το ενδιαφέρον για την χρήση CNNs για θέματα Computer Vision. Έχει παρόμοια αρχιτεκτονική με το LeNet αλλά ήταν μεγαλύτερο και με μεγαλύτερο βάθος καθώς είχε αρκετά επίπεδα συνελιγμού σε ακολουθία. Κέρδισε την πρώτη θέση στο ImageNet Large Scale Visual Recognition Challenge το 2012. [28]

ZF Net: Το ZF Net ή Zeiler & Fergus Net κέρδισε την πρώτη θέση στο ILSVRC του 2013 και αποτελεί βελτίωση του AlexNet. Βελτιστοποίησε τις παραμέτρους που χρησιμοποιούσε το δίκτυο και ταυτόχρονα μεγάλωσε το μέγεθος των επιπέδων συνελιγμού που βρίσκονταν στην μέση, σύμφωνα με την αρχιτεκτονική του δικτύου, ενώ παράλληλα μείωσε το μέγεθος των φίλτρων του 1^{ου} επιπέδου. [42]

GoogLeNet: Το GoogLeNet το οποίο κέρδισε το ILSVRC του 2014 σχεδιάστηκε από τους Szegedy et al. οι οποίοι εργάζονταν στην Google. Η κύρια συνεισφορά τους ήταν η ανάπτυξη ενός Inception Module το οποίο μείωσε τις παραμέτρους του δικτύου στα 4.000.000, σε σύγκριση με τις 60.000.000 παραμέτρους του AlexNet. Παράλληλα, χρησιμοποίησε επίπεδα pooling μέσης τιμής αντί για ένα fully-connected επίπεδο. Το συγκεκριμένο δίκτυο έχει διαφορετικές βελτιωμένες εκδόσεις και με την πιο πρόσφατη την **Inception-v4**.[43] [44]

VGGNet: Το VGGNet δημιουργήθηκε από τους Karen Simonyan και Andrew Zisserman, και πήρε την δεύτερη θέση στο ILSVRC του 2014. Η συγκεκριμένη αρχιτεκτονική ανέδειξε την σημασία του βάθους του δικτύου ως βασικό παράγοντα της καλής απόδοσής του. Το τελικό δίκτυο αποτελείται από 16 Συνελικτικά/Fully-Connected επίπεδα τα οποία χρησιμοποιούν συνελιγμούς 3x3 και pooling 2x2 κατά μήκος όλου του δικτύου. Το μειονέκτημα του συγκεκριμένου δικτύου είναι πως χρησιμοποιεί μεγάλο όγκο μνήμης και παραμέτρων (14.000.000). Από αυτές τις παραμέτρους οι περισσότερες βρίσκονται στα πρώτα fully-connected επίπεδα και αποδείχτηκε πως μπορούν τα συγκεκριμένα επίπεδα να αφαιρεθούν χωρίς να μειωθεί η απόδοση του δικτύου. [45]

ResNet: Το ResNet ή Residual Network αναπτύχθηκε από τους Kaiming He et al. και κέρδισε τον διαγωνισμό ILSVRC του 2015. Η συγκεκριμένη αρχιτεκτονική αποτελεί πλέον την πιο διαδεδομένη αρχιτεκτονική για εφαρμογή σε CNNs, ιδίως μετά τις βελτιώσεις που έγιναν στην συγκεκριμένη αρχιτεκτονική από τον Μάρτιο του 2016. [46] [47]

Εικόνα 43. Παράδειγμα ενός VGGNet δικτύου το οποίο χρησιμοποιεί συνελιγμούς 3x3 και επίπεδα pooling τα οποία χρησιμοποιούν max pooling 2x2. Σε κάθε επίπεδο βλέπουμε το μέγεθος του κάθε επιπέδου, των βαρών του κάθε επιπέδου καθώς και της μνήμης που απαιτείται. Παρατηρείται πως η περισσότερη μνήμη απαιτείται στα πρώτα επίπεδα συνελιγμού και πως οι πιο πολλές παράμετροι βρίσκονται στα fully-connected επίπεδα. [88]

```
INPUT: [224x224x3]        memory:  224*224*3=150K   weights: 0
CONV3-64: [224x224x64]  memory:  224*224*64=3.2M   weights: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64]  memory:  224*224*64=3.2M   weights: (3*3*64)*64 = 36,864
POOL2: [112x112x64]  memory:  112*112*64=800K   weights: 0
CONV3-128: [112x112x128]  memory:  112*112*128=1.6M   weights: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128]  memory:  112*112*128=1.6M   weights: (3*3*128)*128 = 147,456
POOL2: [56x56x128]  memory:  56*56*128=400K   weights: 0
CONV3-256: [56x56x256]  memory:  56*56*256=800K   weights: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256]  memory:  56*56*256=800K   weights: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256]  memory:  56*56*256=800K   weights: (3*3*256)*256 = 589,824
POOL2: [28x28x256]  memory:  28*28*256=200K   weights: 0
CONV3-512: [28x28x512]  memory:  28*28*512=400K   weights: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512]  memory:  28*28*512=400K   weights: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512]  memory:  28*28*512=400K   weights: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512]  memory:  14*14*512=100K   weights: 0
CONV3-512: [14x14x512]  memory:  14*14*512=100K   weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]  memory:  14*14*512=100K   weights: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512]  memory:  14*14*512=100K   weights: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512]  memory:  7*7*512=25K  weights: 0
FC: [1x1x4096]  memory:  4096  weights: 7*7*512*4096 = 102,760,448
FC: [1x1x4096]  memory:  4096  weights: 4096*4096 = 16,777,216
FC: [1x1x1000]  memory:  1000 weights: 4096*1000 = 4,096,000

TOTAL memory: 24M * 4 bytes ~= 93MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters
```

6. Openface

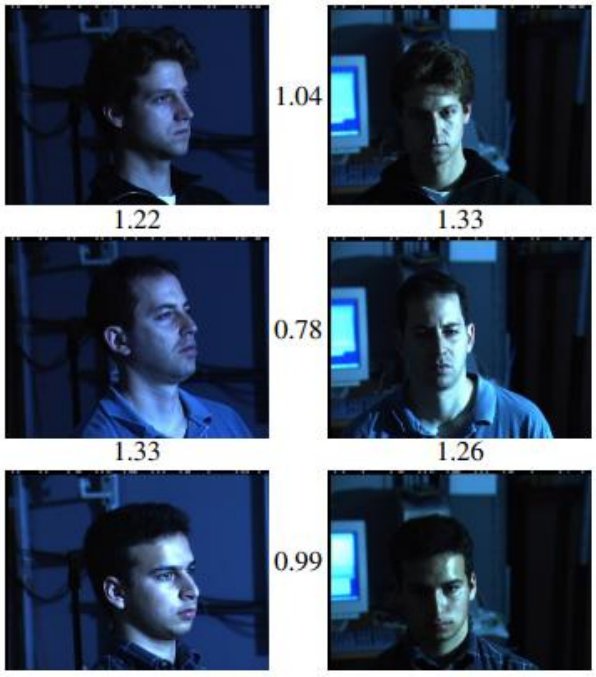
Η εφαρμογή αναγνώρισης προσώπων σε αρχεία βίντεο, της οποίας θέμα είναι η συγκεκριμένη διατριβή, βασίζεται στην βιβλιοθήκη OpenFace. Πρόκειται για μία υλοποίηση ανοιχτού κώδικα νευρωνικών δικτύων αναγνώρισης προσώπων μέσω Python και Torch και έχει αναπτυχθεί από τους Brandon Amos, Bartosz Ludwiczuk, και Mahadev Satyanarayanan. Το νευρωνικό δίκτυο με βάση το οποίο έχει εκπαιδευτεί η OpenFace βασίζεται στο **CVPR** paper του 2015 με τίτλο **FaceNet: A Unified Embedding for Face Recognition and Clustering,** των Florian Schroff, Dmitry Kalenichenko, και James Philbin, το οποίο αναπτύχθηκε στην Google. [48]

6.1 FaceNet

Το FaceNet αποτελεί ένα σύστημα το οποίο μπορεί να χρησιμοποιηθεί για αναγνώριση, ταυτοποίηση και ομαδοποίηση προσώπων. Η μέθοδος στην οποία βασίζεται είναι η εύρεση μιας Ευκλείδειας ενσωμάτωσης (embedding) ανά εικόνα μέσω της χρήσης ενός συνελικτικού νευρωνικού δικτύου βαθιάς μάθησης (deep convolutional network ή DNN). Το DNN εκπαιδεύεται ούτως ώστε να μπορεί να εντοπίσει τις L2 αποστάσεις και σύμφωνα με αυτές να συμπεράνει κατά πόσο υπάρχει ομοιότητα μεταξύ προσώπων. Ως αποτέλεσμα, εικόνες του ίδιου ατόμου θα έχουν μικρότερες αποστάσεις μεταξύ τους από ότι αν γίνει σύγκριση με εικόνες διαφορετικών ατόμων. Εφόσον, εκπαιδευτεί κατάλληλα το δίκτυο, το πρόβλημα της ταξινόμησης εικόνων μπορεί να επιλυθεί με την χρήση ενός αλγόριθμου k-NN.

Οι υπόλοιπες προσεγγίσεις νευρωνικών δικτύων χρησιμοποιούν ένα επίπεδο ταξινόμησης και στην συνέχεια εφαρμόζουν ένα επίπεδο bottleneck για την γενίκευση των αποτελεσμάτων αναγνώρισης, με αποτέλεσμα την μη αποδοτικότητα και αξιοπιστία του δικτύου. [49] Αντιθέτως, το FaceNet εξάγει μέσα από την εκπαίδευση του δικτύου, ένα embedding 128 διαστάσεων χρησιμοποιώντας μία συνάρτηση απώλειας βασισμένη σε τριάδες εικόνων σύμφωνα με τον ταξινομητή **LMNN (large margin nearest neighbor)**. Οι τριάδες που χρησιμοποιούνται αποτελούνται από δύο μικρογραφίες του ίδιου προσώπου και μία μικρογραφία ενός διαφορετικού προσώπου και με την χρήση της συνάρτησης απώλειας επιτυγχάνεται ο διαχωρισμός του σωστού ζεύγους από το λανθασμένο, σύμφωνα με ένα ικανοποιητικό περιθώριο απόστασης. Οι μικρογραφίες που χρησιμοποιούνται πρόκειται για κοντινά πλάνα των προσώπων απεικονιζόμενα χωρίς να έχει γίνει στοίχιση ή άλλη τροποποίηση τους. Η σωστή επιλογή των τριάδων είναι καθοριστική όσον αφορά την επίτευξη καλής απόδοσης του συστήματος. Με την μέθοδο αυτή επιτυγχάνεται η αναγνώριση προσώπων σε εικόνες, με ποικιλία φωτισμού και στάσεων προσώπου. [49]

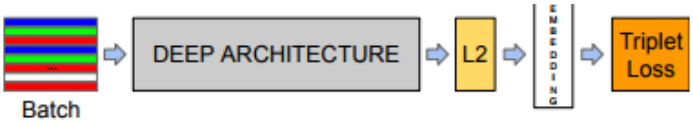
Εικόνα 44. Παράδειγμα αποτελεσμάτων του FaceNet για ζεύγη προσώπων σε διαφορετικές συνθήκες φωτισμού και στάσης. Οι αποστάσεις ανάμεσα σε κάθε ζεύγος εικόνων αντιπροσωπεύουν το κατά πόσο πρόκειται για το ίδιο άτομο ή όχι, με 0.0 να είναι η τιμή απόλυτης ταυτοποίησης και 4.0 η πλήρης διαφοροποίηση. Όπως φαίνεται από τα αποτελέσματα του FaceNet αν ορίσουμε την τιμή κατωφλίου για ταυτοποίηση ενός προσώπου σε 1.1 θα είχαμε ορθή ταξινόμηση των ατόμων που εμφανίζονται σε κάθε εικόνα. [69]



6.1.1 Μεθοδολογία

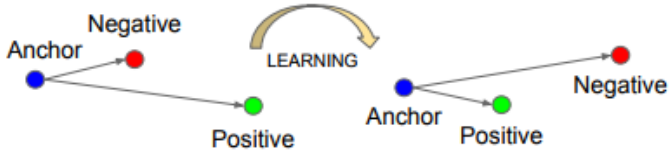
Όπως είδαμε προηγουμένως το σημαντικότερο κομμάτι της μεθοδολογίας για την εκπαίδευση του δικτύου είναι η χρήση της μεθόδου της συνάρτησης απώλειας με χρήση τριάδων εικόνων (**triplet loss function**), η οποία αντικατοπτρίζει πλήρως τον στόχο που πρέπει να επιτευχθεί για την αναγνώριση προσώπων. Το ιδανικό αποτέλεσμα της μεθόδου είναι η εύρεση ενός embedding **f(x)** για κάθε εικόνα **x**, όπου το τετράγωνο των αποστάσεων μεταξύ όλων των εικόνων, ανεξαρτήτως συνθηκών εικόνας, θα είναι ελάχιστο μεταξύ των ίδιων προσώπων σε σχέση με τις αποστάσεις μεταξύ εικόνων διαφορετικών προσώπων. Ακριβέστερα έχουμε $f(x) \in R^d$, με την ενσωμάτωση του x να γίνεται σε έναν Ευκλείδειο χώρο d-διαστάσεων.

Εικόνα 45. Παράδειγμα οργάνωσης του συστήματος FaceNet. Το δίκτυο που χρησιμοποιείται έχει ως επίπεδο εισόδου ένα σύνολο εικόνων και στη συνέχεια εφόσον γίνει εκπαίδευση του δικτύου υπολογίζεται η απόσταση L2 της εικόνας, γίνεται η ενσωμάτωση των χαρακτηριστικών της εικόνας σε πίνακα, και τελικώς εφαρμόζεται η τεχνική του triplet loss για την εξαγωγή των αποτελεσμάτων. [69]



Γίνεται κατανοητό πως η μέθοδος εκπαίδευσης μέσω σύγκρισης τριάδων εικόνων βασίζεται στην μέθοδο ταξινόμησης κοντινότερου γείτονα. Με αυτό τον τρόπο θέλουμε να διασφαλίσουμε πως μία εικόνα x_i^a ενός συγκεκριμένου ατόμου βρίσκεται κοντύτερα σε όλες τις εικόνες x_i^p του ίδιου ατόμου από ότι οι εικόνες x_i^n οποιουδήποτε διαφορετικού ατόμου, όπως φαίνεται και από την παρακάτω εικόνα. [50]

Εικόνα 46. Παράδειγμα εκπαίδευσης μέσω triplet loss όπου μέσω εκπαίδευσης ελαχιστοποιείται η απόσταση μεταξύ εικόνων x_i^a (anchor), x_i^p (positive) που περιέχουν το ίδιο άτομο ενώ ταυτόχρονα μεγιστοποιείται η απόσταση μεταξύ εικόνων x_i^a (anchor), x_i^n (negative). [69]



Επομένως το επιθυμητό αποτέλεσμα είναι,

$$\|x_i^a - x_i^p\|_2^2 + \alpha < \|x_i^a - x_i^n\|_2^2, \forall (x_i^a, x_i^p, x_i^n) \in \mathcal{T}$$ [69]

όπου ως **α,** ορίζεται το περιθώριο που υπάρχει ανάμεσα στα θετικά και αρνητικά ζεύγη. Ως **T** ορίζεται όλες οι πιθανές τριάδες στο training dataset που χρησιμοποιείται από ένα σύνολο **N**. Τελικώς, η απώλεια που πρέπει να ελαχιστοποιηθεί είναι της μορφής:

$$L = \sum_i^N \left[\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha \right]_+$$ [69]

Παρότι υπάρχουν και άλλες μέθοδοι που χρησιμοποιούν σύγκριση μεταξύ εικόνων των ίδιων προσώπων σε σχέση με αυτές διαφορετικών όπως η χρήση της μεθόδου triplet loss παραμένει πιο αποδοτική καθώς λαμβάνει υπόψη της και προσπαθεί να υπολογίσει και ένα περιθώριο ανάμεσα σε πρόσωπα που δεν ταυτοποιούνται, συμβάλλοντας έτσι στον διαχωρισμό των διαφορετικών προσώπων που περιέχονται σε κάθε dataset. [51]

6.1.2 Επιλογή Τριάδων

Σε περίπτωση μεγάλων dataset από εικόνες, όπως θα πρέπει να είναι ένα dataset που χρησιμοποιείται για την εκπαίδευση ενός CNN, θα υπάρχουν πολλές περιπτώσεις όπου μία τριάδα εικόνων θα ικανοποιεί την αρχική υπόθεση αποστάσεων μεταξύ ίδιων και διαφορετικών προσώπων. Σε αυτές τις περιπτώσεις, αυτές οι τριάδες δεν θα είχαν σημαντική συνεισφορά στην εκπαίδευση του δικτύου, ενώ παράλληλα θα το επιβάρυναν σημαντικά από πλευράς απόδοσης. Επομένως, για να αυξήσουμε την απόδοση του δικτύου θα πρέπει να επιλέξουμε τριάδες που θα βελτιώσουν την εκπαίδευσή του.

Ιδανικά θα πρέπει να βρούμε τριάδες όπου για μία εικόνα x_i^a θα επιλεγεί το x_i^p για το οποίο θα ισχύει:

$$\text{argmax}_{x_i^p} \|f(x_i^a) - f(x_i^p)\|_2^2$$ [69]

και αντίστοιχα ένα x_i^n όπου,

$$\text{argmin}_{x_i^n} \|f(x_i^a) - f(x_i^n)\|_2^2$$ [69]

Συνεπώς στόχος μας είναι ο εντοπισμός των τριάδων εκείνων που δεν θα ικανοποιούν την αρχική συνθήκη της μικρότερης απόστασης ανάμεσα στα θετικά ζεύγη και μεγαλύτερης ανάμεσα στα αρνητικά. Βέβαια κάτι τέτοιο είναι αδύνατο να επιτευχθεί μέσα σε όλες τις τριάδες που μπορεί να προκύψουν από το dataset, καθώς σε αυτή την περίπτωση θα είχαμε πολλαπλά λάθη ταξινόμησης. Για να μπορέσει να ξεπεραστεί αυτό το πρόβλημα το FaceNet χρησιμοποιεί μικρές ομαδοποιήσεις εικόνων κατά μήκος της εκπαίδευσης και μέσα από αυτές τις υποομάδες υπολογίζει τα argmax και argmin των αποστάσεων. Για την καλύτερη εκπαίδευση των υποομάδων επιλέχθηκαν κυρίως τα argmin των αρνητικών ζευγών και έγινε σύγκριση μεταξύ

όλων των θετικών ζευγών και όχι μόνο αυτών που είχαν argmax, οδηγώντας έτσι σε πιο σταθερά αποτελέσματα ταξινόμησης και αύξηση της σύγκλισης των αποστάσεων κατά την αρχή της εκπαίδευσης του δικτύου. [69]

6.1.3 Σχεδιασμός και Εκπαίδευση των CNN

Ο σχεδιασμός των συνελικτικών νευρωνικών δικτύων (convolutional neural network) που χρησιμοποιείται από το FaceNet βασίζεται σε δύο από τις βασικές αρχιτεκτονικές τις οποίες είδαμε συνοπτικά σε προηγούμενη ενότητα. Το πρώτο δίκτυο βασίζεται στην αρχιτεκτονική του **ZFNet** [42] και το δεύτερο βασίζεται σε πρόσφατες εκδόσεις ενός δικτύου **Inception** αρχιτεκτονικής. [43]

Οι κύριες διαφορές των δύο δικτύων εντοπίζονται στις διαφορές του αριθμού των **παραμέτρων** και των **FLOPS (floating point operations per second)** που απαιτούνται κατά την εκτέλεση τους. Ανάλογα με το πλαίσιο εφαρμογής των δικτύων κάθε μοντέλο μπορεί να είναι καταλληλότερο, καθώς αν επιθυμούμε να χρησιμοποιήσουμε το δίκτυο σε ένα κινητό τηλέφωνο θα πρέπει να χρησιμοποιήσουμε πολύ λιγότερες παραμέτρους, ώστε να επαρκεί η μνήμη, σε σύγκριση με ένα server όπου θα απαιτείται μεγαλύτερος αριθμός παραμέτρων και θα είναι αυξημένος ο αριθμός των FLOPS.

Στο μοντέλο που χρησιμοποιήθηκε με βάση το ZFNet ανάμεσα στα επίπεδα που χρησιμοποιούν συνελιγμούς προστίθενται αντίστοιχα επίπεδα συνελιγμού μεγέθους 1x1xd σύμφωνα με τους M. Lin et al [51]. Ως αποτέλεσμα έχουμε ένα δίκτυο 22 επιπέδων με 140.000.000 παραμέτρους και με περίπου 1.600.000.000 FLOPS ανά εικόνα που θα ονομάσουμε **NN1**.

Εικόνα 47. Διάγραμμα αρχιτεκτονικής του NN1 μοντέλου που βασίζεται στο Zeiler & Fergus με επίπεδα συνελιγμού 1x1 με βάση την πρόταση από τους Lin et al. Τα μεγέθη των επιπέδων εισόδου (size-in) και εξόδου (size-out) περιγράφονται ως γραμμές x στήλες x φίλτρα. Το μέγεθος του πυρήνα (kernel) είναι γραμμές x στήλες, μέγεθος βήματος και το μέγεθος του maxout pooling ορίζεται ως p=2. [37]

layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32×128	maxout p=2	34M	34M
fc7128	1×32×128	1×1×128		524K	0.5M
L2	1×1×128	1×1×128		0	
total				140M	1.6B

Η δεύτερη κατηγορία μοντέλων που χρησιμοποιήθηκαν, βασίζονται στα μοντέλα Inception του GoogLeNet. Τα συγκεκριμένα μοντέλα έχουν κατά 20 φορές λιγότερες

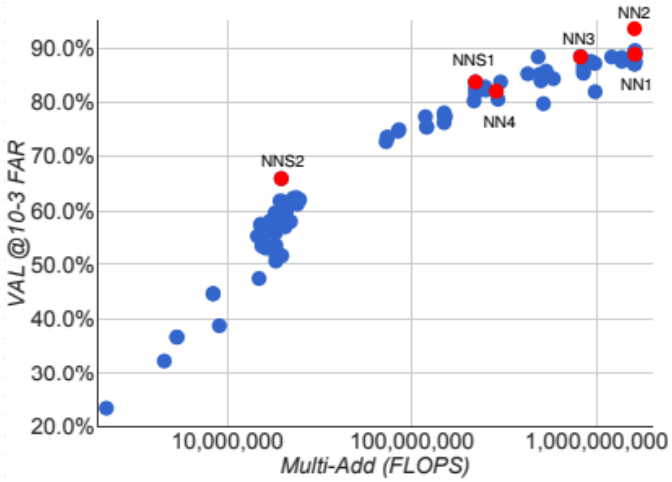
παραμέτρους (γύρω στα 6.600.000 – 7.500.00) και κατά 5 φορές λιγότερα FLOPS (έως και 500.000.000). Ορισμένα από αυτά τα μοντέλα είναι σχεδιασμένα να έχουν πολύ μικρότερο μέγεθος, ούτως ώστε να μπορούν να τρέξουν σε κινητά τηλέφωνα με επιτυχία. Η πρώτη μορφή του μοντέλου (**NNS1**) αποτελείται από 26.000.000 παραμέτρους και απαιτεί 220.000.000 FLOPS ανά εικόνα. Το **NNS2** αποτελείται από 4.300.000 παραμέτρους και μονάχα 20.000.000 FLOPS. Τα **NN2,** που θα δούμε αναλυτικά στο κάτω διάγραμμα**,** και το **NN3** έχουν παρόμοια αρχιτεκτονική αλλά το δεύτερο έχει μειωμένο επίπεδο εισόδου μεγέθους 160x160. Τέλος, το **NN4** έχει μέγεθος εισόδου 96x96 μειώνοντας ακόμα περισσότερο τις απαιτήσεις σε FLOPS σχέση με το **NN2** (285.000.000 σε σχέση με 1.600.000.000). Επιπλέον, επειδή έχει μικρότερο μέγεθος εισόδου, έχουν αφαιρεθεί οι συνελιγμοί μεγέθους 5x5 καθώς είναι περιττά και μπορούν να αφαιρεθούν με πολύ μικρή μείωση στην αποδοτικότητα του δικτύου. [69]

Εικόνα 48. Διάγραμμα αρχιτεκτονικής του NN2 μοντέλου το οποίο βασίζεται σε Inception GoogLeNet αρχιτεκτονικές. Το συγκεκριμένο μοντέλο είναι σχεδόν ίδιο με αυτό που έχει προταθεί από τους Szegedy et al. Οι δύο κύριες διαφορές εντοπίζονται στην χρήση pooling L2 αποστάσεων σε ορισμένα σημεία σε σχέση με max pooling. Το μέγεθος του pooling είναι σε όλα τα επίπεδα 3x3, εκτός από το τελευταίο pooling οπού γίνεται pooling μέσου όρου, και γίνεται παράλληλα με τα συνελικτικά modules του Inception μοντέλου. Ως p ορίζεται η μείωση των διαστάσεων σε κάθε επίπεδο μετά από κάθε διαδικασία pooling. [69]

type	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj (p)	params	FLOPS
conv1 (7×7×3, 2)	112×112×64	1							9K	119M
max pool + norm	56×56×64	0						m 3×3, 2		
inception (2)	56×56×192	2		64	192				115K	360M
norm + max pool	28×28×192	0						m 3×3, 2		
inception (3a)	28×28×256	2	64	96	128	16	32	m, 32p	164K	128M
inception (3b)	28×28×320	2	64	96	128	32	64	L_2, 64p	228K	179M
inception (3c)	14×14×640	2	0	128	256,2	32	64,2	m 3×3,2	398K	108M
inception (4a)	14×14×640	2	256	96	192	32	64	L_2, 128p	545K	107M
inception (4b)	14×14×640	2	224	112	224	32	64	L_2, 128p	595K	117M
inception (4c)	14×14×640	2	192	128	256	32	64	L_2, 128p	654K	128M
inception (4d)	14×14×640	2	160	144	288	32	64	L_2, 128p	722K	142M
inception (4e)	7×7×1024	2	0	160	256,2	64	128,2	m 3×3,2	717K	56M
inception (5a)	7×7×1024	2	384	192	384	48	128	L_2, 128p	1.6M	78M
inception (5b)	7×7×1024	2	384	192	384	48	128	m, 128p	1.6M	78M
avg pool	1×1×1024	0								
fully conn	1×1×128	1							131K	0.1M
L2 normalization	1×1×128	0								
total									7.5M	1.6B

Όλα τα μοντέλα των CNN έχουν εκπαιδευτεί με **Stochastic Gradient Descent (SGD)** [38] με **standard backdrop** [53]. και την χρήση **AdaGrad**. [54] Όλα τα μοντέλα έχουν τυχαία αρχικοποίηση σύμφωνα με τους Szegedy et al [34] και εκπαιδεύτηκαν μέσω ενός cluster υπολογιστών για 1.000 με 2.000 ώρες. Μετά από 500 ώρες εκπαίδευσης η αύξηση της ακρίβειας μειώνεται σε μεγάλο βαθμό, αλλά η επιπρόσθετη εκπαίδευση αυξάνει σημαντικά την αποδοτικότητα. Το περιθώριο a ορίζεται σε 0,2. Για την εκπαίδευση των μοντέλων, χρησιμοποιήθηκαν από 100.000.000 έως 200.000.000 μικρογραφίες εικόνων από 8.000.000 διαφορετικά άτομα. Για κάθε εικόνα χρησιμοποιήθηκε ένας αλγόριθμος face detection και οι μικρογραφίες τροποποιήθηκαν σύμφωνα με το απαιτούμενο επίπεδο εισόδου του κάθε δικτύου. Τα μεγέθη των μικρογραφιών έχουν εύρος από 96x96 έως 224x224 pixels.

Εικόνα 49. Διάγραμμα απεικόνισης της σχέσης μεταξύ FLOPS και Ακρίβειας των CNN. Τα μοντέλα που χρησιμοποιούνται από το FaceNet απεικονίζονται με κόκκινες κουκίδες. [69]



Με βάση το παραπάνω διάγραμμα, μπορούμε να δούμε πως συμπεριφέρεται η ακρίβεια του κάθε μοντέλου σε σχέση με τον αριθμό των παραμέτρων που έχει το κάθε μοντέλο. Για παράδειγμα, το μοντέλο **NN2** έχει πολύ κοντινά επίπεδα απόδοσης σε σχέση με το **NN1** παρότι έχει μόνο το 1/20 αριθμό παραμέτρων. Συνεπώς, παρατηρούμε πως το μοντέλο με βάση την αρχιτεκτονική Zeiler & Fergus προσφέρει αντίστοιχη απόδοση με το μεγαλύτερο από τα μοντέλο βασισμένα στην αρχιτεκτονική Inception. Επιπλέον, γίνεται αντιληπτό πως και μερικά από τα πιο μικρά μοντέλα όπως το NN3 έχουν υψηλή απόδοση, ενώ έχουν αρκετά μειωμένες απαιτήσεις μνήμης και υπολογιστικής ισχύος. Ο παρακάτω πίνακας δείχνει αναλυτικά τα αποτελέσματα των επιτυχημένων ταξινομήσεων των ζευγών.

Εικόνα 50. Διάγραμμα απόδοσης του κάθε μοντέλου δικτύου του FaceNet. Ως VAL ορίζεται το μέσο ποσοστό σωστής ταυτοποίησης προσώπων. Επίσης περιλαμβάνεται και το τυπικό σφάλμα για κάθε μοντέλο δικτύου. [69]

architecture	VAL
NN1 (Zeiler&Fergus 220×220)	87.9% ± 1.9
NN2 (Inception 224×224)	89.4% ± 1.6
NN3 (Inception 160×160)	88.3% ± 1.7
NN4 (Inception 96×96)	82.0% ± 2.3
NNS1 (mini Inception 165×165)	82.4% ± 2.4
NNS2 (tiny Inception 140×116)	51.9% ± 2.9

Σημαντικός παράγοντας για την σωστή αναγνώριση ενός προσώπου είναι και η ποιότητα της εικόνας που εξετάζεται. Για ποιότητα εικόνας της μορφής JPEG, με ποιότητα από 20 και πάνω η απόδοση κυμαίνεται στα ίδια επίπεδα.

Εικόνα 51. Πίνακας που απεικονίζει το ποσοστό απόδοσης του δικτύου NN1 για εικόνες JPEG με διαφορετικά επίπεδα ποιότητας. [69]

jpeg q	val-rate
10	67.3%
20	81.4%
30	83.9%
50	85.5%
70	86.1%
90	86.5%

Για εικόνες που είναι μεγέθους μέχρι 120x120 pixels η απόδοση του μοντέλου παραμένει σε υψηλά επίπεδα και ακόμη και σε εικόνες 80x80 pixels το δίκτυο έχει οριακά αποδεκτή απόδοση. Αυτά τα επίπεδα απόδοσης είναι αξιοσημείωτα, καθώς το δίκτυο εκπαιδεύτηκε με εικόνες μεγέθους 220x220 pixels και μπορούμε συνεπώς να θεωρήσουμε πως αν η εκπαίδευση γινόταν με εικόνες χαμηλότερης ανάλυσης, τότε η αποδοτικότητα θα μπορούσε να είναι αρκετά αυξημένη.

Εικόνα 52. Πίνακας που απεικονίζει το ποσοστό απόδοσης του δικτύου NN1 για εικόνες JPEG με διαφορετική ανάλυση εικόνας. [69]

#pixels	val-rate
1,600	37.8%
6,400	79.5%
14,400	84.5%
25,600	85.7%
65,536	86.4%

Ένας ακόμη παράγοντας που πρέπει να εξεταστεί, είναι τα αποτελέσματα του δικτύου σε διαφορετικές διαστάσεις των embeddings που εξάγονται από τα μοντέλα. Όπως φαίνεται στον παρακάτω πίνακα, η αύξηση των διαστάσεων πάνω από 128 μειώνει σε ένα ελάχιστο βαθμό την αποδοτικότητα του δικτύου. Αυτό πιθανώς οφείλεται στο γεγονός ότι για μεγαλύτερο αριθμό διαστάσεων θα απαιτούταν μεγαλύτερος χρόνος εκπαίδευσης, παρότι οι διαφορές στην απόδοση είναι στατιστικά μη σημαντικές. Επίσης, αξιοσημείωτο είναι το γεγονός πως και οι μικρότερες διαστάσεις έχουν ίδια μεγέθη απόδοσης με μικρή απόκλιση, για δίκτυα που θα μπορούσαν να χρησιμοποιηθούν σε κινητές συσκευές.

Εικόνα 53. Πίνακας που απεικονίζει το ποσοστό απόδοσης του δικτύου NN1 για εικόνες με διαφορετικό αριθμό διαστάσεων embeddings. Επίσης περιλαμβάνεται και το τυπικό σφάλμα για κάθε αριθμό διαστάσεων. [69]

#dims	VAL
64	$86.8\% \pm 1.7$
128	$87.9\% \pm 1.9$
256	$87.7\% \pm 1.9$
512	$85.6\% \pm 2.0$

Τέλος, σημαντικό αντίκτυπο στην απόδοση του δικτύου έχει ο αριθμός των εικόνων που χρησιμοποιούνται από το δίκτυο για εκπαίδευση. Από τον παρακάτω πίνακα είναι εμφανές πως χρησιμοποιώντας δεκάδες εκατομμύρια εικόνων για εκπαίδευση αυξάνεται η ακρίβεια των μοντέλων. Αν χρησιμοποιήσουμε αντίστοιχα εκατοντάδες εκατομμύρια εικόνες για εκπαίδευση αυξάνεται και πάλι η ακρίβεια του μοντέλου αλλά όχι σε τέτοιο βαθμό όσο η μετάβαση από εκατομμύρια σε δεκάδες εκατομμύρια. [69]

Εικόνα 54. Πίνακας που απεικονίζει το ποσοστό απόδοσης ενός δικτύου παρόμοιου με το NN2 για εικόνες μεγέθους 96x96 pixel, μετά από 700 ώρες εκπαίδευσης. [69]

#training images	VAL
2,600,000	76.3%
26,000,000	85.1%
52,000,000	85.1%
260,000,000	86.2%

6.1.4 Απόδοση Δικτύου FaceNet σε Ακαδημαϊκά Dataset

Ο έλεγχος των μοντέλων δικτύων που χρησιμοποιεί το FaceNet για τις μετρήσεις αποδοτικότητας του γίνεται σε δύο διαδεδομένο ακαδημαϊκά dataset. Το πρώτο είναι το **Labeled Faces in the Wild (LFW),** το οποίο είναι το de-facto dataset που χρησιμοποιείται για

ελέγχους αναγνώρισης προσώπων. [55] Το δεύτερο είναι το **Youtube Faces DB,** το οποίο πρόκειται για ένα dataset το οποίο είναι παρόμοιας φιλοσοφίας με το LFW αλλά αντί να χρησιμοποιεί ζεύγη εικόνων, χρησιμοποιεί ζεύγη βίντεο.

Τα αποτελέσματα του FaceNet για το μοντέλο δικτύου **NN1** στο LFW για ταξινόμηση εικόνων είχαν ποσοστό ακρίβειας **99.87%** σε περίπτωση εφαρμογής μόνο crop από το κέντρο της μικρογραφίας του LFW, ενώ σε περίπτωση όπου έγινε αναγνώριση προσώπου στις μικρογραφίες και στη συνέχεια alignment του προσώπου το ποσοστό ανέβηκε στο **99.63%** με **0,09 τυπικό σφάλμα**. [56] Το ποσοστό επιτυχίας και ο βαθμός απόκλισης του NN1 ήταν σημαντικά μικρότερο από το **DeepFace** [57] και κατά 30% μικρότερο σε σχέση με το state-of-the-art δίκτυο **DeepId2+**. [58]

Εικόνα 55. Σφάλματα κατά την εκτέλεση του NN1 στο dataset LFW. Στο πάνω μέρος της εικόνας φαίνονται τα ζεύγη τα οποία ταξινομήθηκαν λάθος ως ίδιο πρόσωπο (false accept) και στο κάτω μέρος φαίνονται τα ζεύγη που απορρίφθηκαν λανθασμένα (false reject). [69]

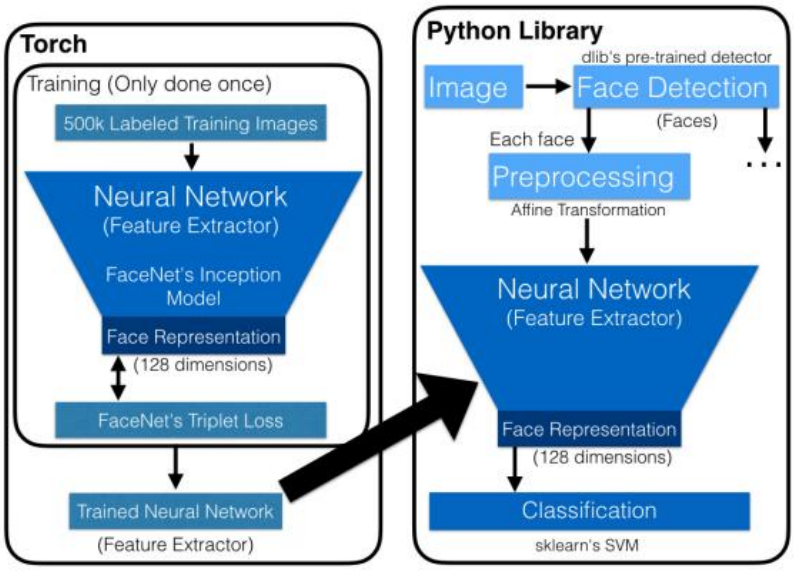
Για την απόδοση του **NN1** σε σχέση με το **Youtube Faces DB,** χρησιμοποιήθηκαν τα ζεύγη από τα πρώτα 100 καρέ του κάθε βίντεο στα οποία εντοπίζονται πρόσωπα. Το ποσοστό ακρίβειας κατά την ταξινόμηση εικόνων με αυτή τη μεθοδολογία ήταν **95,12%** με απόκλιση 0,39. Χρησιμοποιώντας τα πρώτα 1000 καρέ το τελικό ποσοστό ήταν **95,18%.** Σε σύγκριση με το DeepFace όπου είχε ποσοστό ακρίβειας 91,4%, με χρήση 100 καρέ, και με το DeepId2+ το οποίο είχε ποσοστό 93,2%, παρατηρούμε πως τα αποτελέσματα των πειραμάτων συνάδουν με την βελτίωση των ποσοστών που παρουσίασε και ο έλεγχος στο LFW. [69]

6.2 Σχεδιασμός και υλοποίηση της OpenFace

Η **OpenFace** σχεδιάστηκε με σκοπό την χρήση σε περιβάλλοντα φορητών συσκευών, όπου θα πρέπει να γίνει αναγνώριση προσώπων σε πραγματικό χρόνο και το σύστημα οφείλει να προσαρμοστεί ανάλογα με το γενικό πλαίσιο της εφαρμογής, με απώτερο σκοπό την υλοποίηση

ενός συστήματος που θα πρέπει να έχει υψηλή απόδοση, ενώ παράλληλα θα έχει αυξημένη ταχύτητα εκπαίδευσης και πρόβλεψης.

Εικόνα 56. Απεικόνιση της δομής υλοποίησης της Openface από το στάδιο της αρχικής εκπαίδευσης του νευρωνικού δικτύου σε Torch περιβάλλον, μέχρι το τελικό στάδιο της ταξινόμησης άγνωστων προσώπων. [48]

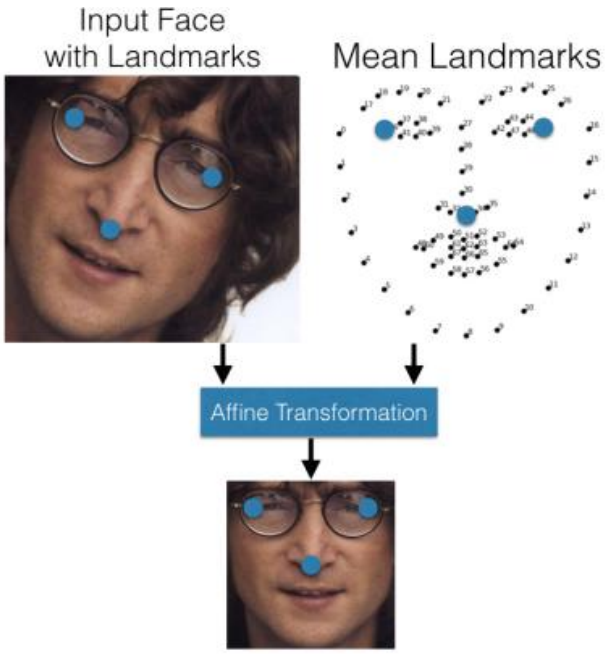


Για την εκπαίδευση του νευρωνικού δικτύου και την εξαγωγή των αποτελεσμάτων η OpenFace χρησιμοποιεί **Torch** περιβάλλοντα, [59] **Lua** [60] και **Luajit**. [61] βιβλιοθήκη της **Python** [62] χρησιμοποιεί την **numpy** πράξεις μεταξύ πινάκων και για χρήση γραμμικής άλγεβρας [63] και **scikit-learn** για την λειτουργίες ταξινόμησης. [64] Για τον εντοπισμό προσώπων χρησιμοποιείται ο προ-εκπαιδευμένος ανιχνευτής προσώπων της **dlib** [65] καθώς προσφέρει καλύτερα αποτελέσματα από τον αντίστοιχο της **OpenCV.**

6.2.1 Προεπεξεργασία Δεδομένων Εισόδου

Η OpenFace απαιτεί μια διαδικασία προεπεξεργασίας των εικόνων που θα δοθούν για την εκπαίδευση του νευρωνικού δικτύου. Κατά την διαδικασία του εντοπισμού προσώπου σε μια εικόνα μπορούν τα bounding boxes που περιβάλλουν μια εικόνα να επιστρέψουν εικόνες που θα έχουν διαφορετικές στάσεις και φωτισμούς προσώπων, δυσχεραίνοντας τον εντοπισμό τους και την απαραίτητα εξαγωγή των χαρακτηριστικών. Όπως είδαμε παραπάνω, η FaceNet μπορεί να αντιμετωπίσει το εξής πρόβλημα έχοντας ένα αρκετά μεγάλο training dataset, αλλά στην περίπτωση της OpenFace θα πρέπει να γίνει επεξεργασία της εικόνας ούτως ώστε για κάθε εικόνα, τα μάτια, μύτη και στόμα του κάθε προσώπου να εμφανίζονται σε παρόμοια σημεία. Η μέθοδος που χρησιμοποιείται για να επιτευχθεί αυτό λέγεται **affine transformation** και ανιχνεύει μέσω του **landmark detector** της **dlib.** [65] Πρόκειται για μια εφαρμογή των Kazemi et al [66] όπου εντοπίζει 68 σημεία του προσώπου και στην συνέχεια επεξεργάζεται την εικόνα ώστε οι γωνίες των ματιών και η μύτη να βρίσκονται στις προκαθορισμένες θέσεις. Επιπλέον, γίνεται η απαραίτητη επεξεργασία της εικόνας ώστε το μέγεθος της εικόνας που θα δοθεί στο νευρωνικό δίκτυο να είναι 96x96 pixel. [48]

Εικόνα 57. Προεπεξεργασία εικόνας της OpenFace με χρήση affine transformation. [48]



6.2.2 Εκπαίδευση του Νευρωνικού Δικτύου της OpenFace

Για την εκπαίδευση του Νευρωνικού Δικτύου η OpenFace συνδυάζει δεδομένα από τα δύο μεγαλύτερα dataset που χρησιμοποιούνται σε ερευνητικά επίπεδα για αναγνώριση προσώπων, τα **CASIA-WebFace.** [67] Η OpenFace εκπαιδεύεται με 500.000 εικόνες από τα παραπάνω dataset, ενώ η FaceNet χρησιμοποιεί 100.000.000 εικόνες [68] και η DeepFace που χρησιμοποιεί ένα dataset που αποτελείται από 4.400.000 εικόνες. [69]

Η OpenFace χρησιμοποιεί μία τροποποιημένη εκδοχή του **NN4** δικτύου της Facenet, με την ονομασία **nn4.small2** η οποία μειώνει τον αριθμό των παραμέτρων για χρήση σε μικρότερα dataset. Επιπλέον, το δίκτυο χρησιμοποιεί την τεχνική **triplet loss** της FaceNet.

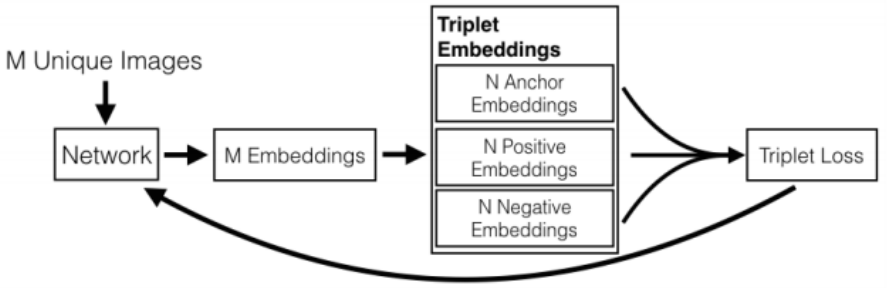
Εικόνα 58. Διαγραμματική απεικόνιση του νευρωνικού δικτύου nn4.small2 της OpenFace. [48]

type	output size	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj
conv1 ($7 \times 7 \times 3, 2$)	$48 \times 48 \times 64$						
max pool + norm	$24 \times 24 \times 64$						m $3 \times 3, 2$
inception (2)	$24 \times 24 \times 192$		64	192			
norm + max pool	$12 \times 12 \times 192$						m $3 \times 3, 2$
inception (3a)	$12 \times 12 \times 256$	64	96	128	16	32	m, 32p
inception (3b)	$12 \times 12 \times 320$	64	96	128	32	64	ℓ_2, 64p
inception (3c)	$6 \times 6 \times 640$		128	256,2	32	64,2	m $3 \times 3, 2$
inception (4a)	$6 \times 6 \times 640$	256	96	192	32	64	ℓ_2, 128p
inception (4e)	$3 \times 3 \times 1024$		160	256,2	64	128,2	m $3 \times 3, 2$
inception (5a)	$3 \times 3 \times 736$	256	96	384			ℓ_2, 96p
inception (5b)	$3 \times 3 \times 736$	256	96	384			m, 96p
avg pool	736						
linear	128						
ℓ_2 normalization	128						

Η εκπαίδευση του δικτύου της OpenFace γίνεται με τον διαχωρισμό εικόνων σε τριάδες και εφόσον υπολογιστεί το **triplet loss,** γίνεται επανατροφοδότηση του δικτύου με την χρήση

backpropagation. Για κάθε ομάδα εικόνων, χρησιμοποιείται ένα δείγμα από 20 εικόνες ανά άτομο, από ένα σύνολο 15 διαφορετικών ατόμων.

Εικόνα 59. Απεικόνιση της ροής λειτουργίας για την εκπαίδευση του Νευρωνικού Δικτύου της OpenFace [48]



6.2.3 Επαλήθευση Αποτελεσμάτων μέσω LFW

Για την επαλήθευση των αποτελεσμάτων της OpenFace χρησιμοποιείται και σε αυτή την περίπτωση το dataset LFW, το οποίο προσφέρει πειράματα τόσο για αναγνώριση προσώπου αλλά και για ταξινόμηση.

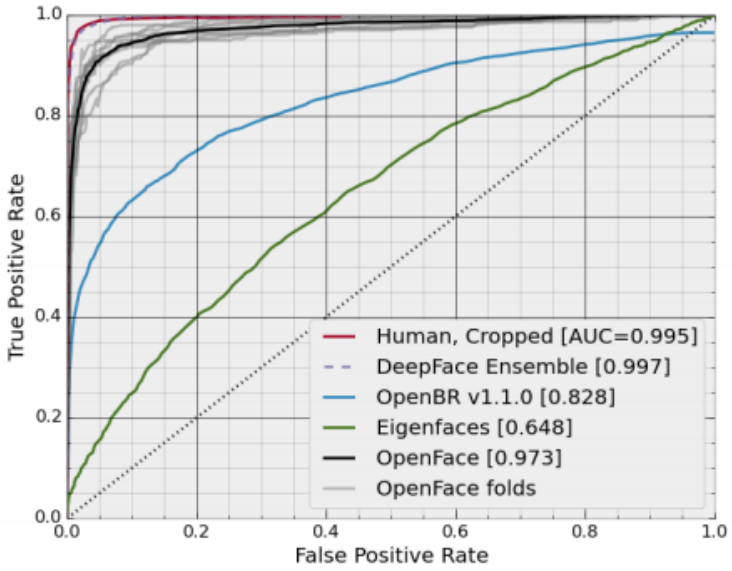
Το πείραμα της LFW για αναγνώριση προσώπων προβλέπει αν σε ένα ζεύγος προσώπων υπάρχει το ίδιο πρόσωπο ή όχι. Για το πείραμα διατίθενται 13.233 εικόνες από 5.750 άτομα και για το πείραμα παρέχονται 6.000 ζευγάρια εικόνων τα οποία χωρίζονται σε 10 τμήματα. Η ακρίβεια του πειράματος υπολογίζεται από τον μέσο όρο ακρίβειας 10 διαφορετικών πειραμάτων. Η διαδικασία περιλαμβάνει την χρήση των 9 τμημάτων για εκπαίδευση και του τελευταίου για τον έλεγχο των αποτελεσμάτων με την χρήση L2 αποστάσεων για την ταυτοποίηση ή όχι ενός προσώπου.

Εικόνα 60. Συγκριτική απεικόνιση αποτελεσμάτων του πειράματος LFW για διαφορετικά νευρωνικά δίκτυα σε σχέση με την OpenFace. [48]

Technique	Accuracy
Human-level (cropped)	0.9753
Eigenfaces (no outside data)	0.6002 ± 0.0079
FaceNet	0.9964 ± 0.009
DeepFace-ensemble	0.9735 ± 0.0025
OpenFace	0.9292 ± 0.0134

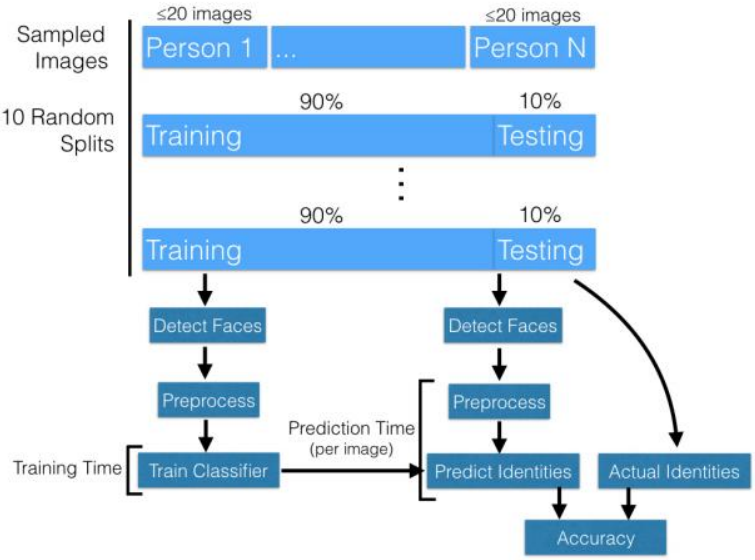
Η απεικόνιση του κατωφλίου ταυτοποίησης του παραπάνω πειράματος μπορεί να απεικονιστεί διαγραμματικά μέσω μίας καμπύλης **ROC (receiver operating characteristic)** η οποία απεικονίζει την σχέση μεταξύ του ποσοστού λανθασμένων ταυτοποιήσεων σε σχέση με τις θετικές. Η ιδανική καμπύλη θα έπρεπε να έχει την τιμή 1 κατά μήκος της και η επιφάνεια κάτω από την καμπύλη ROC **(AUC)** να απεικονίζει την πιθανότητα ενός ταξινομητή να προτιμήσει ένα ζεύγος εικόνων από τα ίδια άτομα, σε σχέση με ένα ζεύγος εικόνων από διαφορετικά άτομα. Τα αποτελέσματα της OpenFace πλησιάζουν σε μεγάλο βαθμό την ακρίβεια των state-of-the-art τεχνικών που χρησιμοποιούνται σε deep learning εφαρμογές. [48]

Εικόνα 61. Συγκριτική απεικόνιση ROC καμπυλών με βάση το πείραμα LFW για ταυτοποίηση προσώπων. [48]



Για την εκτέλεση του πειράματος ταξινόμησης εικόνων με χρήση ενός υποσυνόλου εικόνων από το LFW dataset έγινε σύγκριση των αποτελεσμάτων της OpenFace, με τις εξής τεχνικές της **OpenCV: Eigenfaces** [70]**, Fisherfaces** [71] [72] **και Local Binary Pattern Histograms (LBPH)**. [73] Για τα δεδομένα του πειράματος επιλέχθηκαν ως δείγμα 20 εικόνες από κάθε έναν αριθμό N ατόμων ή σε περίπτωση που δεν διατίθενται 20 εικόνες χρησιμοποιούνται όσες εικόνες είναι διαθέσιμες. Στην συνέχεια, τα δεδομένα που επιλέχθηκαν χωρίζονται τυχαία 10 φορές έτσι ώστε το 90% των εικόνων να αποτελεί το dataset εκπαίδευσης και το 10% τις εικόνες που θα πρέπει να ταξινομηθούν.

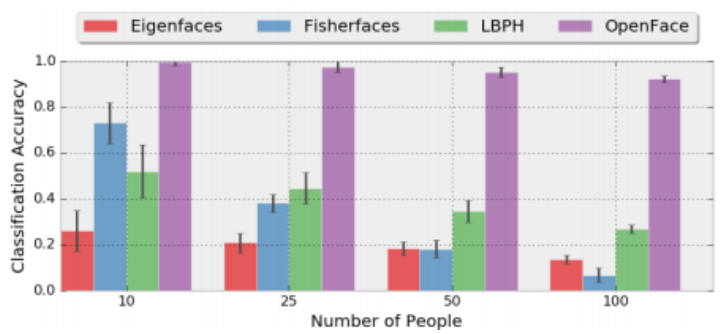
Εικόνα 62. Απεικόνιση της μεθοδολογίας του πειράματος ακρίβειας και απόδοσης ταξινόμησης με την χρήση LFW. [48]



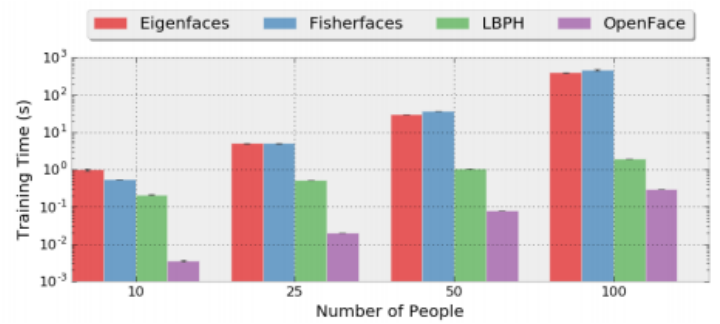
Η ταχύτητα που απαιτείται για την εκπαίδευση και την εκτέλεση του ταξινομητή, είναι ιδιαίτερα σημαντική για την OpenFace που αποσκοπεί στην χρήση σε φορητές συσκευές. Η προεπεξεργασία που απαιτείται για τον ταξινομητή είναι η μετατροπή της εικόνας σε μορφή grayscale και ακολούθως η τροποποίηση της μέσω affine transformation και alignment των

προσώπων, ώστε να δοθεί στο νευρωνικό δίκτυο και να εξαχθεί η απεικόνιση της σε 128 διαστάσεις. Ο ταξινομητής της OpenFace χρησιμοποιεί έναν γραμμικό αλγόριθμο Μηχανών Διανυσματικής Στήριξης με βάρος regularization ίσο με 1, καθώς αποδίδει καλύτερα σε σύγκριση με άλλα βάρη. Τα αποτελέσματα του πειράματος παρουσιάζονται στα παρακάτω διαγράμματα. [48]

Εικόνα 63. Συγκριτικό διάγραμμα απεικόνισης τεχνικών ταξινόμησης. Όσο αυξάνεται ο αριθμός των ατόμων μειώνεται η ακρίβεια του ταξινομητή. [48]



Εικόνα 64. Συγκριτικό διάγραμμα απεικόνισης ταχύτητας εκπαίδευσης του ταξινομητή. Όσο αυξάνεται ο αριθμός των ατόμων αυξάνεται ο απαιτούμενος χρόνος. [48]

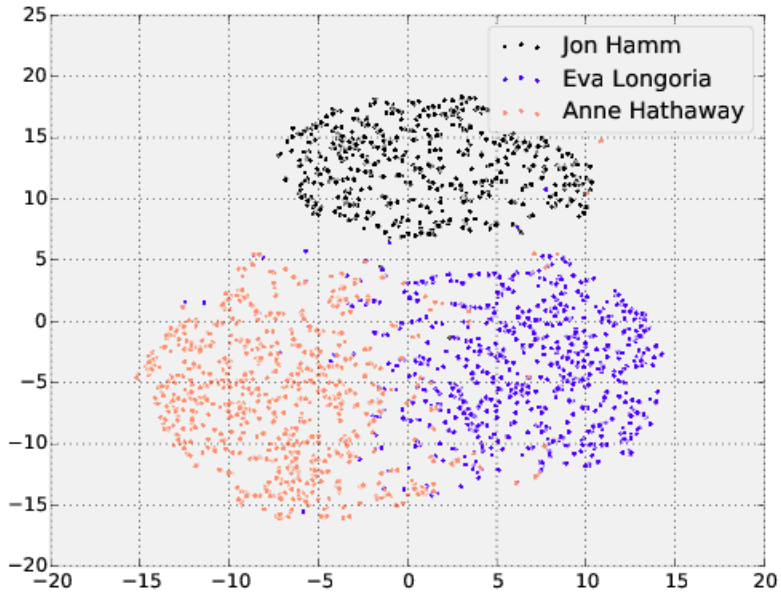


Εικόνα 65. Συγκριτικό διάγραμμα απεικόνισης ταχύτητας προβλέψεων ανά εικόνα του ταξινομητή. Όσο αυξάνεται ο αριθμός των προσώπων αυξάνεται και ο χρόνος πρόβλεψης όλων των ταξινομητών εκτός από της OpenFace. [48]

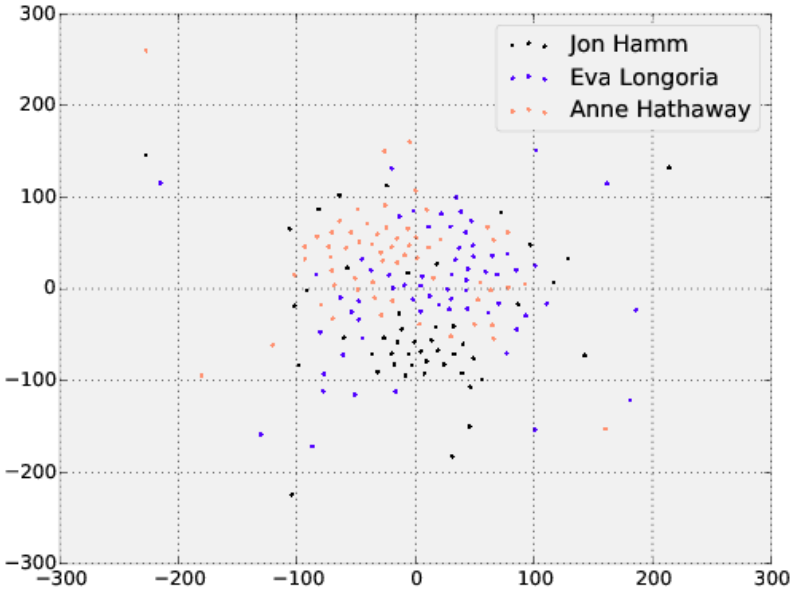
6.2.4 Απεικόνιση Representations μέσω t-SNE

Με την χρήση της μεθόδου **t-SNE**, μιας τεχνικής μείωσης διαστάσεων, είναι εφικτή η οπτικοποίηση των **128-διάστατων χαρακτηριστικών** που η OpenFace παράγει. Ακολουθούν παραδείγματα οπτικοποίησης τριών ατόμων, μέσα από ένα training και από ένα testing dataset. [74] [75]

Εικόνα 66. Παράδειγμα οπτικοποίησης από το Training Dataset. [92]



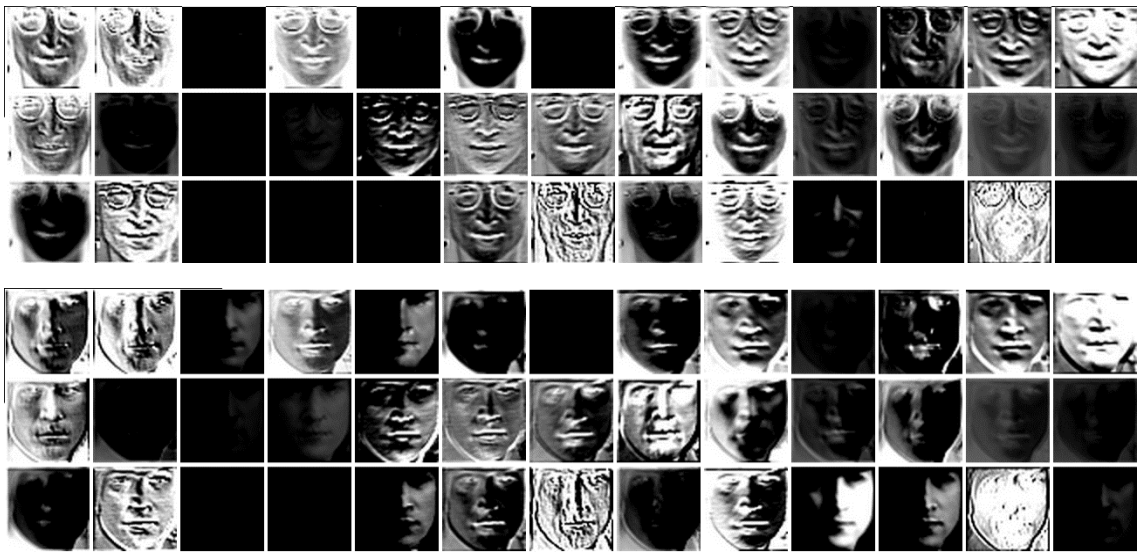
Εικόνα 67. Παράδειγμα οπτικοποίησης από το Testing Dataset. [92]



6.2.5 Απεικόνιση των χαρακτηριστικών του CNN

Για να γίνει πιο κατανοητός ο τρόπος επεξεργασίας μιας εικόνας και εξαγωγής των χαρακτηριστικών του κάθε επιπέδου του νευρωνικού δικτύου της OpenFace ακολουθούν τα αποτελέσματα από τα πρώτα 39 φίλτρα του πρώτου συνελικτικού επιπέδου του δικτύου. [76]

Εικόνα 68. Αποτελέσματα εξόδου των πρώτων 39 φίλτρων του πρώτου επιπέδου συνελιγμού της OpenFace για 2 εικόνες. [92]



6.2.6 Παρεμφερείς Υλοποιήσεις Νευρωνικών Δικτύων

Εκτός της **OpenFace** υπάρχουν στη συνέχεια αναφέρονται επιγραμματικά παρεμφερής υλοποιήσεις νευρωνικών δικτύων για μοντέλα μηχανικής μάθησης.

Oxford's VGG Face Descriptor:

Ο **VGG Face Descriptor** στο benchmark της LFW παρουσιάζει ακρίβεια σε ποσοστό 0.9727%. Το μοντέλο του το οποίο βασίζεται σε αλγόριθμους softmax δεν ενσωματώνει χαρακτηριστικά προσώπων όπως η FaceNet με αποτέλεσμα διαδικασίες clustering και classification να υλοποιούνται με μεγαλύτερη δυσκολία. Η υλοποίηση του βασίζεται σε αντίστοιχο μοντέλο triplet loss το οποίο δεν έχει γίνει διαθέσιμο για επεξεργασία. Το license του διατίθεται μονάχα για ερευνητικούς σκοπούς. [77]

Deep Face Representation

Η συγκεκριμένη υλοποίηση νευρωνικού δικτύου παρουσιάζει ακρίβεια σε ποσοστό .9777 με βάση το LFW benchmark. Αντίστοιχα, με τον VGG Face Descriptor δεν ενσωματώνει χαρακτηριστικά προσώπων όπως η FaceNet και δεν έχει αποκτήσει ακόμα licence. [78]

7. Επισκόπηση Εφαρμογής Face Identification

Η δομή της εφαρμογής ταξινόμησης ατόμων από αρχεία βίντεο που παρουσιάζεται σε αυτή την διατριβή μπορεί να χωριστεί σε τρία ξεχωριστά επίπεδα:

1) **Εύρεση Εικόνων για το Training Dataset**

2) **Προεπεξεργασία Εικόνων**

3) **Εκπαίδευση του Ταξινομητή**

4) **Διαδικασία Ταξινόμησης**

Στις μετέπειτα ενότητες, θα παρουσιαστεί αναλυτικότερα το κάθε στάδιο της εφαρμογής μέσω παραδειγμάτων εκτέλεσης και συγκριτικών αποτελεσμάτων. Για την διευκόλυνση της παρουσίασης των κλήσεων του κάθε προγράμματος, θεωρείται πως κάθε κλήση γίνεται μέσα από το **command line** των **Linux** και μέσα από τον φάκελο **facerec (ονομασία φακέλου που περιέχει το project της εφαρμογής)**, εκτός από τα σημεία που αναφέρεται το αντίθετο.

7.1 Εργαλεία Ανάπτυξης Εφαρμογής

Προτού γίνει η ανάλυση της εφαρμογής ταξινόμησης προσώπων σε αρχεία βίντεο και για την σαφέστερη επεξήγηση του τρόπου λειτουργίας της και των επιμέρους στοιχείων που την απαρτίζουν, παρατίθενται στη συνέχεια, επιγραμματικά, απαραίτητες έννοιες και ορισμοί των βιβλιοθηκών και των μεθόδων που χρησιμοποιούνται από αυτή.

7.1.1 Python

Η Python είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού υψηλού επιπέδου η οποία χρησιμοποιείται ευρέως σε ποικιλία εφαρμογών γενικού σκοπού. Η φιλοσοφία με βάση την οποία έχει σχεδιαστεί δίνει έμφαση στην αναγνωσιμότητα του κώδικα και η σύνταξη της επιτρέπει την ακρίβεια στον προγραμματισμό σε σχέση με άλλες γλώσσες όπως π.χ. η γλώσσα C++ ή η Java. Η δημιουργία της ξεκίνησε στις αρχές της δεκαετίας του 1990 από τον Guido Van Rossum στην Ολλανδία, στο CWI. Σήμερα, η γλώσσα ανήκει στο **Python Software Institute** η οποία εκδίδει την python ανοιχτού κώδικα.

Τα πλεονεκτήματα επιγραμματικά είναι τα εξής:

1) **Ταχύτητα**

2) **Υποστήριξη διαφορετικών τεχνολογιών**

3) **Φορητότητα ανάμεσα σε λειτουργικά συστήματα**

4) **Απλότητα**

Η Python μπορεί να χρησιμοποιηθεί για τον προγραμματισμό σε επίπεδο λειτουργικών συστημάτων, για την δημιουργία γραφικών περιβαλλόντων (GUI), για το προγραμματισμό δικτυακών εφαρμογών, για τον προγραμματισμό βάσεων δεδομένων καθώς και για τον προγραμματισμό μαθηματικών βιβλιοθηκών. [79]

7.1.2 Lua

Η Lua αποτελεί μία γλώσσα σεναρίων (scripting language) η οποία δημιουργήθηκε το 1993 στη Βραζιλία, στο PUC-Rio. Οι εφαρμογές της είναι πολυπληθής κυρίως στο πεδίο της βιομηχανίας όπως στη ρομποτική, σε επεξεργασία εικόνων, σε διακόπτες Ethernet, σε ανάπτυξη δικτύων καθώς στην ανάπτυξη παιχνιδιών. [80]

Τα πλεονεκτήματα της είναι τα εξής:

1) **Φορητότητα ανάμεσα σε λειτουργικά συστήματα**

2) **Ευκολία στην ενσωμάτωση κώδικα από άλλες γλώσσες**

3) **Μικρού Μεγέθους**

4) **Αποδοτικότητα**

7.1.3 Torch / Torch7

Το Torch/Torch7 είναι ένα ευέλικτο framework το οποίο έχει ενσωματωμένη μία βιβλιοθήκη για εφαρμογές μηχανικής μάθησης με την χρήση της Lua. Με την ενσωμάτωση τεχνολογιών CUDA και OpenMP/SSE για αριθμητικές ρουτίνες χαμηλού επιπέδου επιτυγχάνεται η υψηλή ικανότητα απόδοσης του. [59]

Τα κύρια χαρακτηριστικά που υποστηρίζει είναι: [81]

1) **Μοντέλα νευρωνικών δικτύων**

2) **Ρουτίνες indexing, slicing , transposing**

3) **Υποστήριξη για προγραμματισμό σε κάρτες γραφικών (GPU)**

4) **Ενσωμάτωση σε συσκευές Android και iOS**

7.1.4 Dlib

Η Dlib είναι μια γενικού-σκοπού βιβλιοθήκη ανοιχτού κώδικα που βασίζεται στη γλώσσα C++, η οποία δημιουργήθηκε από τον Davis King, το 2002. Ο σχεδιασμός της είναι βασισμένος στην λογική του component-based προγραμματισμού και για αυτό τον λόγο αποτελεί μια συλλογή από ανεξάρτητα τμήματα λογισμικού από διάφορους τομείς όπως γραφικά περιβάλλοντα, υπολογιστικά νήματα, Bayesian δίκτυα, επεξεργασία εικόνων, γραμμική άλγεβρα κ.α., με ιδιαίτερη έμφαση τα τελευταία χρόνια στον σχεδιασμό εργαλείων για στατιστική μέσω μηχανικής μάθησης. Επιπλέον, παρότι είναι ανοιχτού κώδικα, παρέχει εκτενές documentation για κάθε τμήμα κώδικα το οποίο περιλαμβάνεται σε αυτή. Οι βασικές φιλοσοφίες της είναι η ευκολία χρήσης και η φορητότητα του κώδικα σε διάφορα υπολογιστικά περιβάλλοντα. [65]

7.1.5 OpenCV

Η Open Source Computer Vision Library (OpenCV) είναι μια βιβλιοθήκη ανοιχτού κώδικα η οποία εξειδικεύεται σε θέματα computer vision και μηχανικής μάθησης που βασίζεται στη

γλώσσα C++. Η ανάπτυξη της άρχισε το 1999 από την Intel και πλέον διατηρείται από την Itseez. Στην OpenCV περιλαμβάνονται πάνω από 2.500 βελτιστοποιημένοι αλγόριθμοι, οι οποίοι αναφορικά μπορούν να χρησιμοποιηθούν για τον εντοπισμό και την αναγνώριση προσώπων, για την ταξινόμηση ατόμων σε βίντεο, για εφαρμογές ενισχυμένης πραγματικότητας (augmented reality). [82] [83]

7.1.6 Scikit–learn

Η scikit-learn είναι μία βιβλιοθήκη ανοιχτού κώδικα η οποία ειδικεύεται σε θέματα μηχανικής μάθησης για την γλώσσα προγραμματισμού Python. Αναπτύχθηκε αρχικά από τον David Cournapeau ως μία επέκταση της Scipy και η πρώτη σταθερή έκδοση της ήταν το 2010. Στα περιεχόμενα της περιλαμβάνονται αλγόριθμοι για εκμάθηση με επίβλεψη, εκμάθηση χωρίς επίβλεψη, μετασχηματισμούς dataset και υπολογιστική απόδοση. [64]

Οι βασικοί στόχοι της scikit-learn είναι:

1) Υψηλή Ποιότητα Κώδικα

2) Bare-bone Σχεδιασμός για το Framework

3) Ανάπτυξη μέσω της Κοινότητας Προγραμματιστών

4) Αναλυτικό Documentation

7.2 Εύρεση Εικόνων

Το πρώτο στάδιο για την δημιουργία ενός ταξινομητή (classifier) προσώπων είναι ο ορισμός των κλάσεων που θα χρησιμοποιηθούν και βάση των οποίων θα ταξινομούνται τα πρόσωπα που εισάγονται σε αυτόν. Για να οριστούν οι κλάσεις αυτές θα πρέπει αρχικά να αποφασιστεί ποια ομάδα προσώπων θα αποτελέσει το dataset για το οποίο θα πρέπει να βρεθούν τα απαραίτητα δεδομένα για την εκπαίδευση του ταξινομητή.

Η βιβλιοθήκη της Openface παρέχει την δυνατότητα εκπαίδευσης ενός ταξινομητή με την βοήθεια του νευρωνικού δικτύου το οποίο έχει προεκπαιδευτεί. Επομένως, το νευρωνικό δίκτυο μπορεί να αναγνωρίσει οποιοδήποτε καινούριο πρόσωπο δεχτεί και να κάνει εξαγωγή των απαιτούμενων χαρακτηριστικών (feature extraction) για την εκπαίδευση του ταξινομητή.

Για το πρώτο αυτό στάδιο η εφαρμογή παρέχει την δυνατότητα με την βοήθεια της μηχανής αναζήτησης της Google, αποθήκευσης σε ξεχωριστούς φακέλους των εικόνων από τα πρόσωπα που πρέπει να διαχωριστούν στις αντίστοιχες κλάσεις και από τις οποίες θα γίνει η εκπαίδευση. Με την χρήση ενός ή παραπάνω keywords η εφαρμογή μπορεί να χρησιμοποιηθεί για την αποθήκευση όσων εικόνων επιθυμούμε μέσω των Google Images.

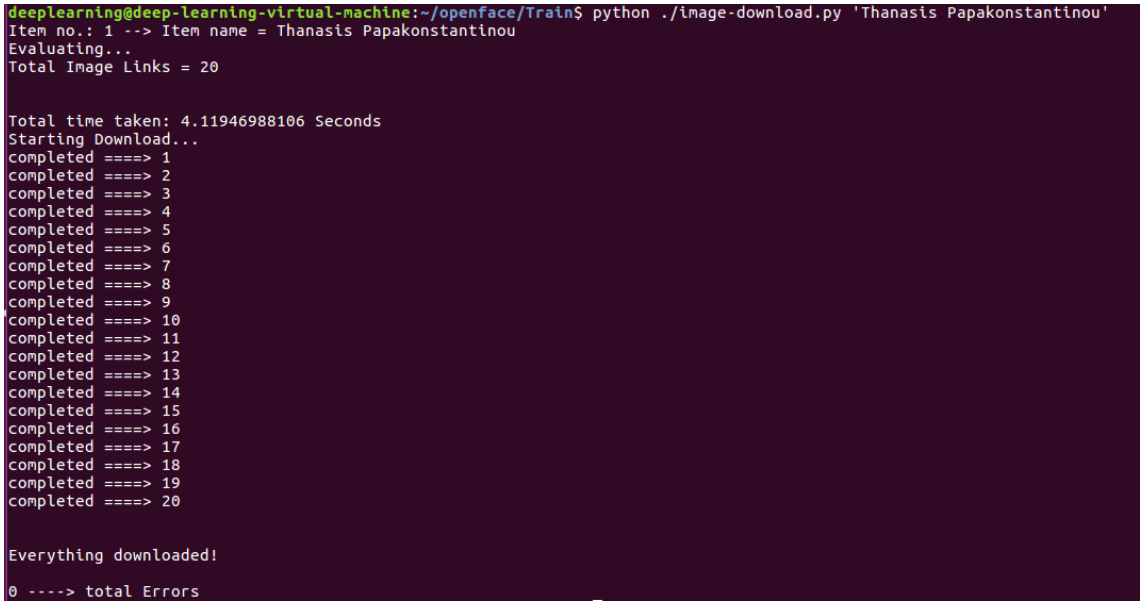
Ο αριθμός των αποτελεσμάτων που θα χρησιμοποιηθεί για το «κατέβασμα» των εικόνων μπορεί να οριστεί παραμετρικά μέσα στον κώδικα του προγράμματος για μεγαλύτερη ευκολία κατά την χρήση του ταξινομητή. Παράλληλα, η εφαρμογή μπορεί να παρέχει στον χρήστη πληροφορίες σχετικά με τους συνδέσμους που χρησιμοποιήθηκαν για την αποθήκευση των εικόνων καθώς και τον χρόνο τον οποίο διήρκησε η διαδικασία αναζήτησης και αποθήκευσης των εικόνων. Επιπλέον, παρέχει μηνύματα λαθών τα οποία μπορεί να προκύψουν κατά την εκτέλεση της διαδικασίας ώστε να μπορεί ο χρήστης της εφαρμογής να δει άμεσα τον αριθμό των εικόνων που αποθηκεύτηκαν και αν χρειαστεί να βελτιώσει τα keywords που χρησιμοποίησε ώστε να έχει τα βέλτιστα αποτελέσματα. Στην συνέχεια, θα περιγραφεί ο τρόπος λειτουργίας του προγράμματος και θα παρουσιαστεί ένα παράδειγμα σχετικά με τον

τρόπο λειτουργίας της εφαρμογής καθώς και όλων των δυνατοτήτων που προσφέρει στον χρήστη.

Μέσα στο φάκελο της εφαρμογής περιλαμβάνεται ο φάκελος με την ονομασία **«image-download»** στον οποίο περιέχεται το πρόγραμμα εύρεσης και αποθήκευσης εικόνων **«image-download.py»**. Από το τερματικό των Linux, εφόσον είμαστε στον φάκελο **image-download**, μπορούμε να το καλέσουμε με την εντολή **«python ./image-download.py '[keyword]'»** όπου μέσα στα brackets θα περιέχεται το keyword το οποίο θα πρέπει να χρησιμοποιήσουμε για την αναζήτηση των εικόνων. Τα μονά brackets είναι απαραίτητα σε περίπτωση όπου πρέπει να εισαχθούν κενά στο κλειδί αναζήτησης. Σε αντίθετη περίπτωση το πρόγραμμα θα αγνοήσει οτιδήποτε ακολουθείται από το κενό και θα θεωρήσει ως κλειδί αναζήτησης μόνο την πρώτη λέξη που θα χρησιμοποιηθεί.

Για το παράδειγμα μας που θα χρησιμοποιήσουμε, θα αναζητήσουμε και θα αποθηκεύσουμε τα αποτελέσματα για τον καλλιτέχνη «Θανάση Παπακωνσταντίνου». Επομένως η εντολή μας θα είναι της μορφής **«python ./image-download.py 'Thanasis Papakonstantinou'»** και ως αποτέλεσμα λαμβάνουμε την εξής εικόνα:

Εικόνα 69. Αποτελέσματα εφαρμογής αναζήτησης και αποθήκευσης εικόνων.



Με βάση την παραπάνω εικόνα παρατηρείται πως αρχικά το πρόγραμμα προσπαθεί να αναγνωρίσει τον αριθμό των κλειδιών αναζήτησης που δέχτηκε ως είσοδο και να τα διαχωρίσει. Εφόσον συμβεί αυτό θα εντοπίσει τους πρώτους 20 υπερσυνδέσμους που αντιστοιχούν στην κάθε εικόνα. Στην συνέχεια θα εμφανίσει ένα μήνυμα το οποίο περιγράφει τον χρόνο τον οποίο χρειάστηκε για την εύρεση των εικόνων. Εφόσον ολοκληρωθεί η διαδικασία, θα ξεκινήσει η αποθήκευση των εικόνων στον φάκελο τον οποίο θα δημιουργήσει αυτόματα η ονομασία του οποίου θα είναι ίδια με το κλειδί αναζήτησης το οποίο χρησιμοποιήθηκε. Για κάθε εικόνα η οποία αποθηκεύεται επιτυχώς εμφανίζεται το μήνυμα «completed», ενώ σε περίπτωση λάθους εμφανίζεται το αντίστοιχο σφάλμα για την ενημέρωση του χρήστη. Τα λάθη που μπορούν να προκύψουν χωρίζονται σε τρεις κατηγορίες: σφάλματα IO, σφάλματα HTTP και σφάλματα URL. Για οποιοδήποτε από αυτά τα σφάλματα θα εμφανιστεί το αντίστοιχο μήνυμα στον χρήστη. Με την ολοκλήρωση της αποθήκευσης των εικόνων το πρόγραμμα ενημερώνει τον χρήστη πως η διαδικασία ολοκληρώθηκε και εμφανίζει τον αριθμό των σφαλμάτων τα οποία προέκυψαν.

Στο παράδειγμα μας δεν εμφανίστηκε κάποιο λάθος, οπότε τα αποτελέσματα μας μπορούν να βρεθούν στον φάκελο με την ονομασία «Thanasis Papakonstantinou» και να είναι

διαθέσιμα για περαιτέρω επεξεργασία. Σε περίπτωση όπου κάποια εικόνα δεν περιέχει κάποιο πρόσωπο λόγω λάθους της μηχανής αναζήτησης δεν χρειάζεται η επέμβαση του χρήστη, καθώς στο επόμενο στάδιο επεξεργασίας θα γίνει η εξαγωγή των χαρακτηριστικών και θα αντιμετωπιστεί το συγκεκριμένο πρόβλημα.

Εικόνα 70. Παράδειγμα μορφής περιεχομένων του φακέλου «Thanasis Papankonstantinou»

Το πρόγραμμα παράλληλα με την αποθήκευση των εικόνων δημιουργεί και αποθηκεύει σε ένα αρχείο .txt με την ονομασία «output.txt» όλους τους υπερσυνδέσμους που χρησιμοποίησε για την εύρεση των εικόνων, ώστε ο χρήστης να μπορεί να έχει πλήρη εικόνα για τις αναζητήσεις που έχουν γίνει από το πρόγραμμα. Εφόσον ολοκληρωθεί η διαδικασία αναζήτησης των εικόνων που χρειαζόμαστε για την εκπαίδευση του ταξινομητή θα προχωρήσουμε στο επόμενο στάδιο το οποίο είναι η επεξεργασία των εικόνων. Οι εικόνες μπορεί να ποικίλλουν σε μέγεθος, σε αριθμό προσώπων που εμφανίζονται, στην θέση την οποία έχει το πρόσωπο, στην απόσταση που βρίσκεται, καθώς και αν είναι αναγνωρίσιμο το πρόσωπο, οπότε είναι απαραίτητο να πάρουν την κατάλληλη μορφή για την εξαγωγή χαρακτηριστικών από αυτές.

7.3 Προεπεξεργασία Εικόνων

Για να μπορέσει να γίνει η κατάλληλη επεξεργασία των εικόνων από το πρόγραμμα θα πρέπει το δένδρο φακέλων που έχουμε δημιουργήσει και αποθηκεύσει τα πρόσωπα τα οποία θα χρησιμοποιήσουμε για την εκπαίδευση του ταξινομητή, να είναι της εξής δομής για κάθε ξεχωριστό φάκελο:

Εικόνα 71. Δομή Φακέλων για τα Δεδομένα του Training Dataset [93]

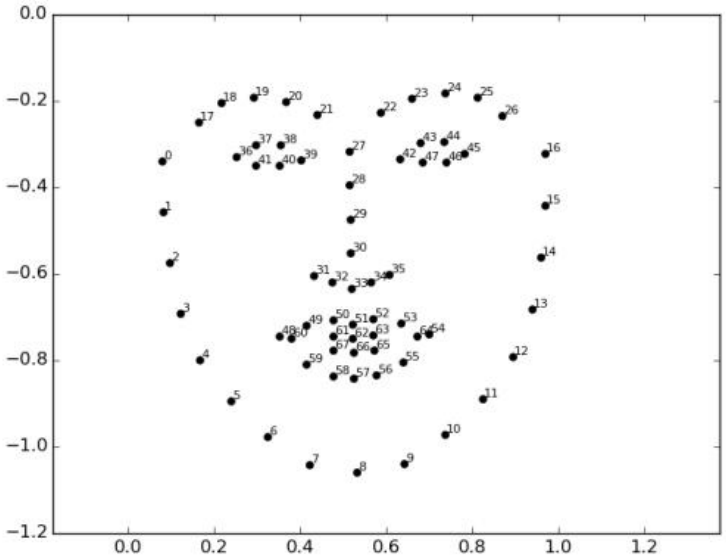
```
$ tree data/mydataset/raw
person-1
├── image-1.jpg
├── image-2.png
...
└── image-p.png

...

person-m
├── image-1.png
├── image-2.jpg
...
└── image-q.png
```

Η ονομασία των εικόνων δεν επηρεάζει κάπως το πρόγραμμα, εφόσον έχουμε διασφαλίσει προηγουμένως ότι οι εικόνες του κάθε φακέλου περιέχουν μόνο τις εικόνες του αντίστοιχου προσώπου. Όπως είδαμε προηγουμένως, το πρόγραμμα αναζήτησης και αποθήκευσης εικόνων φροντίζει για την δημιουργία φακέλων σύμφωνα με την απαιτούμενη

δομή. Η κάθε εικόνα που αποθηκεύεται μπορεί να είναι είτε σε μορφή **.jpeg,** είτε **.png** και η επέκταση του αρχείου **να μην περιέχει κεφαλαία**.

Εφόσον έχουμε καταλήξει στον αριθμό των προσώπων που θα μπορεί να αναγνωρίσει ο ταξινομητής και έχουμε διαμορφώσει κατάλληλα τους φακέλους μπορούμε να ξεκινήσουμε την επεξεργασία των εικόνων. Για την επεξεργασία των εικόνων θα χρησιμοποιήσουμε το πρόγραμμα με την ονομασία **«align-dlib.py»** το οποίο περιέχεται στον φάκελο **«./util».** Ο σκοπός του προγράμματος είναι για κάθε εικόνα να εντοπίσουμε την θέση που βρίσκεται το πρόσωπο και με την βοήθεια της βιβλιοθήκης **dlib,** να εντοπίσουμε τα 68 σημεία που βρίσκονται σε κάθε πρόσωπο (landmark estimation), καθώς και την θέση (pose detection) που έχουν. Αφού εντοπισθούν τα σημεία αυτά, θα αποκοπεί και θα ευθυγραμμιστεί μόνο το πρόσωπο από την συγκεκριμένη εικόνα στο κατάλληλο μέγεθος ώστε να γίνει η εκπαίδευση του ταξινομητή.

Εικόνα 72. Διαγραμματική Απεικόνιση των 68 Landmarks ενός προσώπου [93]



Η κλήση του προγράμματος είναι της παρακάτω μορφής:

./util/align-dlib.py <διαδρομή-φακέλων-εικόνων> align outerEyesAndNose <διαδρομή-φακέλων-ευθυγραμμισμένων-δεδομένων>--size 96 --skipMulti

όπου καλώντας το, το πρόγραμμα θα επεξεργαστεί κάθε εικόνα που βρίσκεται στην διαδρομή που ορίζουμε και εφόσον εντοπιστεί κάποιο πρόσωπο θα αποθηκευτεί μετά την επεξεργασία στον φάκελο που έχουμε ορίσει για τα ευθυγραμμισμένα δεδομένα.

Για τον ορισμό των landmarks όπου θα ευθυγραμμιστεί το κάθε πρόσωπο, το πρόγραμμα μας δίνει τρεις επιλογές:

1) **Περίγραμμα ματιών και μύτης (outerEyesAndNose)**

2) **Εσωτερικό περίγραμμα ματιών και κάτω χείλους (innerEyesAndBottomLip)**

3) **Περίγραμμα ματιών (eyes_1)**

Για το παράδειγμα μας θα χρησιμοποιήσουμε τα περιγράμματα ματιών και μύτης για την ευθυγράμμιση των εικόνων. Επιπρόσθετα, μπορούμε να καθορίσουμε το μέγεθος το οποίο

θα έχουν οι τελικές εικόνες όπως φαίνεται στο όρισμα **«--size 96»**. Στο παράδειγμα μας το μέγεθος των εικόνων που θα χρησιμοποιήσουμε είναι **96x96** καθώς συνδυάζεται το χαμηλό μέγεθος τους με την αξιόπιστη απόδοση τους για εκπαίδευση αντίστοιχων αλγορίθμων ταξινόμησης. [84] Όσον αφορά το όρισμα **«--skipMulti»** το χρησιμοποιούμε για να μην γίνει η επεξεργασία της εικόνας σε περίπτωση όπου υπάρχουν πάνω από 1 πρόσωπο σε μία εικόνα. Με αυτό τον τρόπο διασφαλίζουμε πως τα αρχικά μας δεδομένα για την εκπαίδευση του ταξινομητή δεν θα περιέχουν σφάλματα τα οποία θα επιστρέφουν λάθος αποτελέσματα στην συνέχεια. Μπορούμε για αύξηση της ταχύτητας της επεξεργασίας των αρχικών δεδομένων να εκτελέσουμε όσες διαδικασίες επιθυμούμε ταυτόχρονα. Ιδανικά, μπορούμε να εκτελέσουμε ταυτόχρονα μία διαδικασία για κάθε φάκελο που θέλουμε να ευθυγραμμίσουμε. Για να γίνει αυτό, θα πρέπει να εκτελέσουμε το πρόγραμμα με την παρακάτω εντολή:

for N in {1..[n]}; do ./util/align-dlib.py <διαδρομή-φακέλων-εικόνων> align outerEyesAndNose <διαδρομή-φακέλων-ευθυγραμμισμένων-δεδομένων>--size 96 –skipMulti & done

όπου ως n ορίζουμε τον αριθμό των διαδικασιών που θέλουμε να εκτελεστούν.

Παρόλα αυτά εγκυμονεί ο κίνδυνος ο αριθμός των απαραίτητων δεδομένων για την εκπαίδευση του ταξινομητή να είναι πολύ μικρός, σε περίπτωση που από τα αρχικά δεδομένα δεν μπορεί να γίνει ο εντοπισμός ή η ευθυγράμμιση των προσώπων. Για την αποφυγή του κινδύνου αυτού, μπορεί να οριστεί ένας ελάχιστος αριθμός εικόνων που θα εισάγεται σε κάθε φάκελο για επεξεργασία και σε περίπτωση που ο αριθμός των εικόνων δεν επαρκεί, θα διαγράφεται. Το πρόγραμμα το οποίο θα πρέπει να εκτελέσουμε περιέχεται στον φάκελο **«./util»** και έχει τον τίτλο **«prune-dataset.py»**.

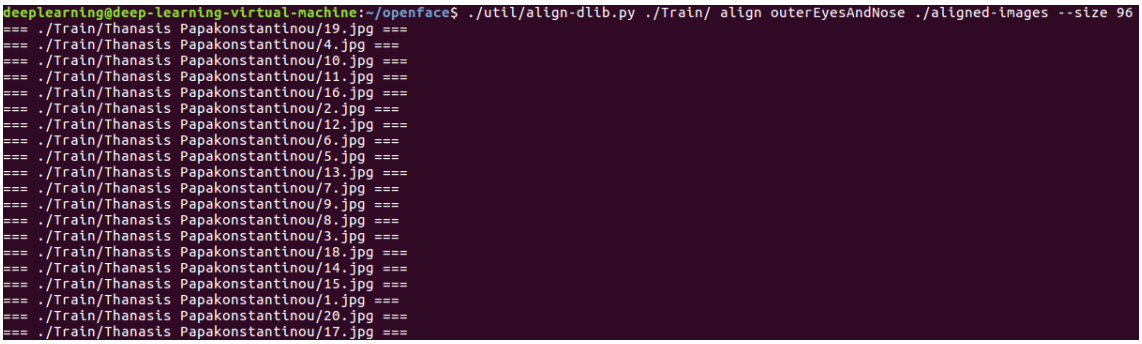
Η κλήση του γίνεται ως εξής:

./util/prune-dataset.py <διαδρομή-φακέλων-ευθυγραμμισμένων-δεδομένων> --numImagesThreshold <ελάχιστος-αριθμός-εικόνων-ανά-φάκελο>

Σε περίπτωση όπου δεν χρησιμοποιηθεί το όρισμα για τον ελάχιστο αριθμό εικόνων, το πρόγραμμα θα διαγράψει κάθε φάκελο με λιγότερες από 10 εικόνες. Ο προκαθορισμένος αυτός αριθμός μπορεί να αλλάξει μέσα από τον κώδικα του προγράμματος.

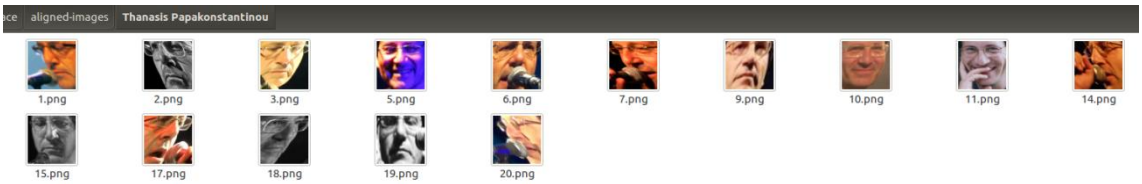
Συνεχίζοντας το παράδειγμα μας θα καλέσουμε το πρόγραμμα ως εξής:

Εικόνα 73. Αποτελέσματα ευθυγράμμισης εικόνων κατά την εκτέλεση της εφαρμογής.



Όπως παρατηρείται το πρόγραμμα για κάθε μία από τις εικόνες που βρίσκονται στον φάκελο που δημιουργήθηκε, τις επεξεργάστηκε και τις αποθήκευσε στον αντίστοιχο φάκελο με την ονομασία «aligned-images».

Εικόνα 74. Αποθηκευμένα αρχεία ύστερα από την εκτέλεση ευθυγράμμισης εικόνων.



Για μία πιο ακριβή σύγκριση ας εξετάσουμε την 1η εικόνα του κάθε φακέλου για να μπορέσουμε να καταλάβουμε καλύτερα τον τρόπο επεξεργασίας της και την τελική μορφή που πρέπει να έχει.

Εικόνα 75. Αρχική Εικόνα. [95]

Εικόνα 76. Εμφάνιση landmarks στην αρχική εικόνα. [95]

Εικόνα 77. Τελική εικόνα μετά από εντοπισμό landmarks και ευθυγράμμιση. [95]

Εφόσον ολοκληρωθεί η διαδικασία για όλες τις εικόνες που περιέχονται σε κάθε κλάση, ακολουθεί το στάδιο της εξαγωγής των embeddings της κάθε κλάσης, με βάση τα οποία, θα ακολουθήσει η εκπαίδευση του ταξινομητή.

7.4 Εξαγωγή Embeddings

Σε αυτό το στάδιο της εφαρμογής θα αξιοποιηθεί το νευρωνικό δίκτυο που παρέχεται από την OpenFace, το οποίο θα χρησιμοποιηθεί για την εξαγωγή των 128 embeddings, σύμφωνα με την διαδικασία που αναλύθηκε σε προηγούμενο κεφάλαιο. Από τις 4 εκδοχές προεκπαιδευμένων που είναι διαθέσιμες από το API της OpenFace επιλέχθηκε το **nn4.small.v1,** λόγω της μεγαλύτερης ακρίβειας που επέδειξε στο **LFW benchmark**.

Η κλήση του προγράμματος για την κλήση των χαρακτηριστικών γίνεται ως εξής:

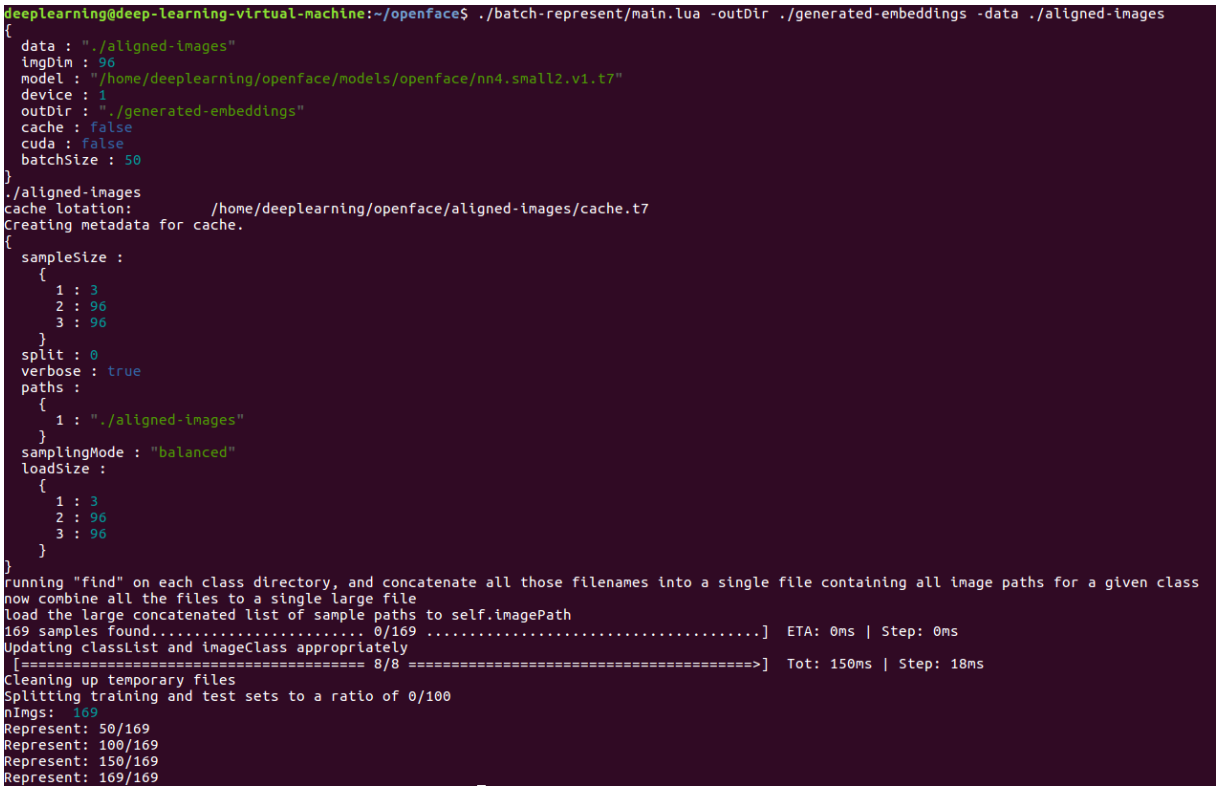
./batch-represent/main.lua -outDir <διαδρομή-φακέλου-αποθήκευσης-embeddings> --data <διαδρομή-φακέλου-ευθυγραμμισμένων-εικόνων>

Αναλυτικότερα καλείται το πρόγραμμα **main.lua** και δέχεται δύο ορίσματα:

1) Μέσω του ορίσματος **–outDir** καθοδηγείται η εφαρμογή στον φάκελο όπου θα αποθηκευτούν τα **embeddings** και τα **labels** της κάθε κλάσης.

2) Μέσω του **–data** δίνεται το path όπου έχουν αποθηκευτεί οι εικόνες στις οποίες έγινε προεπεξεργασία και οι οποίες αποτελούν το training dataset του ταξινομητή.

Εικόνα 78. Παράδειγμα κλήσης της εφαρμογής main.lua για την εξαγωγή των embeddings.



Με την ολοκλήρωση της κλήσης του **main.lua** , δημιουργούνται στον φάκελο **generated embeddings** δύο αρχεία:

- **labels.csv**
- **reps.csv**

Στο αρχείο **labels.csv** αποθηκεύονται, διαχωρίζονται και αριθμούνται οι ονομασίες της κάθε κλάσης. Στο αρχείο **reps.csv** αποθηκεύονται για κάθε κλάση τα **128 embeddings** που εξάχθηκαν.

Κατά την δημιουργία των **representations** δημιουργείται ένα **cache** αρχείο στον φάκελο που περιέχει τις εικόνες του dataset, σε περίπτωση που χρειαστεί να γίνει επανάληψη της διαδικασίας με **διαφορετικό μοντέλο cnn**. Για να γίνει επανάληψη της διαδικασίας με **καινούριο dataset** θα πρέπει να διαγραφεί το συγκεκριμένο αρχείο, αλλιώς θα παρουσιαστεί σφάλμα κατά την εκτέλεση του **ταξινομητή.**

Εικόνα 79. Παράδειγμα σφάλματος για training dataset που έχει αλλαχθεί χωρίς διαγραφή του cache αρχείου.

```
If your dataset has changed, delete the cache file.
nImgs:  169
Represent: 50/169
Represent: 100/169
Represent: 150/169
/home/deeplearning/torch/install/bin/luajit: .../deeplearning/torch/install/share/lua/5.1/image/init.lua:367: ./aligned-images
u/5.png: No such file or directory
stack traceback:
        [C]: in function 'error'
        .../deeplearning/torch/install/share/lua/5.1/image/init.lua:367: in function 'load'
        /home/deeplearning/openface/batch-represent/dataset.lua:330: in function 'sampleHookTest'
        /home/deeplearning/openface/batch-represent/dataset.lua:386: in function 'get'
        ...eeplearning/openface/batch-represent/batch-represent.lua:39: in function 'batchRepresent'
        ./batch-represent/main.lua:42: in main chunk
        [C]: in function 'dofile'
        ...ning/torch/install/lib/luarocks/rocks/trepl/scm-1/bin/th:150: in main chunk
        [C]: at 0x00405d50
```

7.5 Εκπαίδευση του Ταξινομητή

Το επόμενο στάδιο της εφαρμογής είναι η εκπαίδευση του ταξινομητή, με βάση τα embeddings τα οποία έχουν εξαχθεί προηγουμένως. Το μοντέλο ταξινόμησης, το οποίο προκύπτει με βάση το μοντέλο ταξινόμησης που θα επιλεγεί κατά την εκτέλεση του προγράμματος, θα αποθηκευτεί στο πρόγραμμα με την μορφή ενός αρχείου pickle της Python. Στο συγκεκριμένο παράδειγμα, οι δύο αλγόριθμοι που θα χρησιμοποιηθούν για την εκπαίδευση είναι ο **SVM** και ο **Decision Tree.**

Η κλήση του προγράμματος του ταξινομητή γίνεται ως εξής:

./Train/Trainer.py train <διαδρομή-φακέλου-αποθήκευσης-embeddings> --classifier <ονομασία-αλγόριθμου-εκπαίδευσης-ταξινομητή>

Το πρόγραμμα **Trainer.py** δέχεται δύο ορίσματα. Αρχικά, πρέπει να οριστεί ο φάκελος όπου είναι αποθηκευμένα τα αρχεία **labels.csv, reps.csv** που δημιουργήθηκαν προηγουμένως. Στην περίπτωση μας είναι αποθηκευμένα στο path **./generated-embeddings.** Στη συνέχεια, πρέπει να οριστεί το αλγοριθμικό μοντέλο ταξινόμησης με βάση το οποίο θα γίνει η εκπαίδευση. Το συγκεκριμένο όρισμα είναι προαιρετικό και σε περίπτωση που δεν οριστεί η εκπαίδευση γίνεται αυτόματα με την χρήση ενός αλγόριθμου **Γραμμικών Μηχανών Διανυσματικής Στήριξης (Linear SVM)** που παρέχεται από την **scikit-learn.**

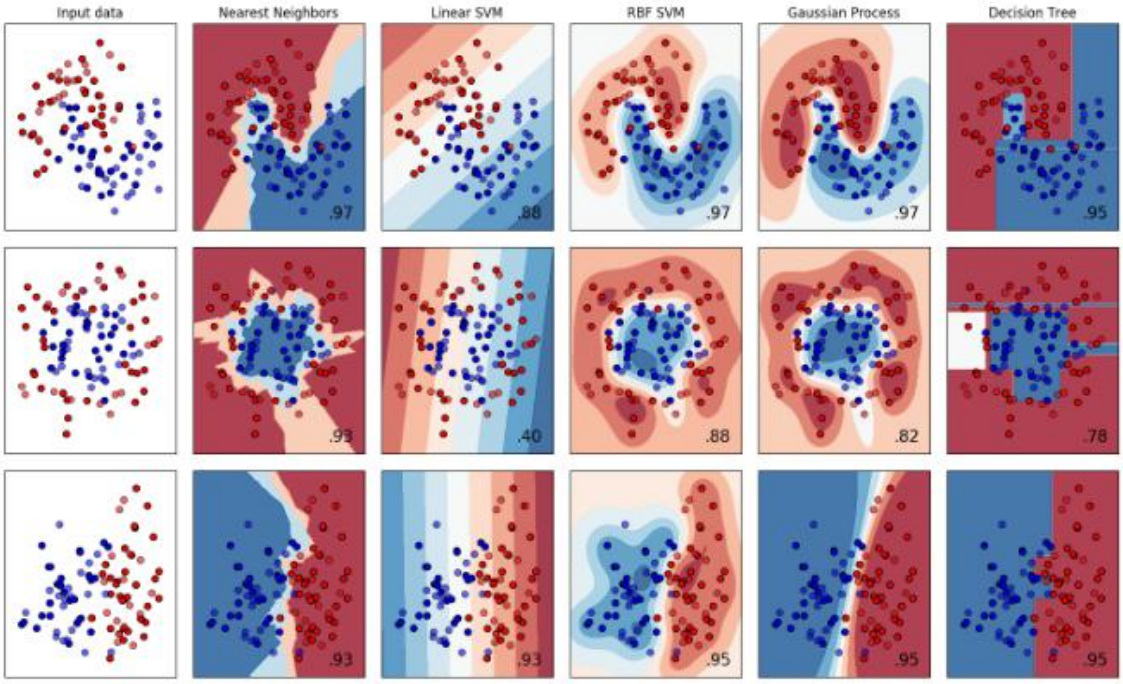
Οι αλγόριθμοι οι οποίοι παρέχονται για την εκπαίδευση του ταξινομητή είναι οι εξής:

1) Linear Svm
2) Grid Search Svm
3) Gaussian Mixture Models (GMM)
4) Radial Svm
5) Decision Tree
6) Gaussian NB
7) Deep Belief Network (DBN)

Ο πιο αξιόπιστος από πλευράς απόδοσης και ακρίβειας για εκπαίδευση του ταξινομητή είναι ο «**Linear SVM»**. Εξίσου γρήγορος είναι ο «**Decision Tree»** αλλά λόγω του τρόπου σχεδιασμού του δεν εξάγει αποτελέσματα με μορφή πιθανοτήτων με αποτέλεσμα να

παρουσιάζονται προβλήματα κατά την εκτέλεση του. Οι υπόλοιποι αλγόριθμοι, παρουσιάζουν αντίστοιχη ακρίβεια με τον **«Linear SVM»** αλλά υστερούν στον χρόνο που απαιτείται για την εκπαίδευση, λόγω της αυξημένης πολυπλοκότητας τους.

Εικόνα 80. Συγκριτική Απεικόνιση του Τρόπου Λειτουργίας των Ταξινομητών της Scikit-learn. Τα στοιχεία που χρησιμοποιούνται για έλεγχο απεικονίζονται με διαφάνεια, ενώ τα υπόλοιπα σημεία αποτελούν τα στοιχεία του training dataset. Κάτω δεξιά αναγράφεται το ποσοστό ακρίβειας του κάθε ταξινομητή. [85]



Εικόνα 81. Παράδειγμα κλήσης του προγράμματος εκπαίδευσης του ταξινομητή με τον αλγόριθμο Decision Tree

```
deeplearning@deep-learning-virtual-machine:~/openface$ ./Train/Trainer.py  train ./generated-embeddings --classifier DecisionTree
Loading embeddings.
Training for 7 classes.
Saving classifier to './generated-embeddings/classifier.pkl'
```

Το αποτέλεσμα του ταξινομητή αποθηκεύεται στην τοποθεσία **./generated-embeddings/** με την ονομασία **classifier.pkl.**

7.6 Διαδικασία Ταξινόμησης

Το τελευταίο στάδιο της διαδικασίας είναι η εκτέλεση του ταξινομητή και η εξαγωγή των αποτελεσμάτων από το πρόγραμμα. Ο ταξινομητής δέχεται ως είσοδο ένα αρχείο βίντεο το και εφόσον γίνει ταξινόμηση εξάγεται το ίδιο αρχείο βίντεο, στο οποίο για κάθε πρόσωπο που εντοπίστηκε, απεικονίζεται κάτω από ένα bounding box, η κλάση στην οποία ταξινομήθηκε. Δίπλα από το κάθε όνομα της τάξης απεικονίζεται και το ποσοστό βεβαιότητας του αλγορίθμου για την ταξινόμηση του προσώπου. Ο ταξινομητής μπορεί να κατατάξει ένα πρόσωπο μόνο σε κάποια από τις γνωστές κλάσεις και σε περίπτωση που εμφανιστεί ένα άγνωστο πρόσωπο τότε θα κάνει μια πρόβλεψη και θα το κατατάξει στην κοντινότερη κλάση από τις ήδη υπάρχουσες.

Η κλήση του ταξινομητή γίνεται ως εξής:

./Classify/Classifier.py –dlibFacePredictor <διαδρομή-μοντέλου-εντοπισμού-προσώπων> --networkModel <διαδρομή-μοντέλου-νευρωνικού-δικτύου-OpenFace> –imgDim <μέγεθος-εικόνας> --cuda --verbose infer <διαδρομή-φακέλου-pickle-αρχείου> <διαδρομή-αρχείου-βίντεο>.mp4 –multi --delete --step <βήμα επεξεργασίας frame>

Αναλυτικότερα:

--dlibFacePredictor (Προαιρετικό): Με το συγκεκριμένο όρισμα καθορίζεται η διαδρομή στην οποία έχει αποθηκευτεί το **.dat** αρχείο **«shape_predictor_68_face_landmarks»,** όπου περιέχεται το πρόγραμμα της dlib για τον εντοπισμό προσώπων με βάση τα 68 landmarks, όπως αναφέρθηκε προηγουμένως.

--networkModel (Προαιρετικό): Στο συγκεκριμένο όρισμα καθορίζεται η διαδρομή στην οποία είναι αποθηκευμένο το μοντέλο του νευρωνικού δικτύου **«nn4.small.v1.t7»** το οποίο θα χρησιμοποιηθεί για την ταξινόμηση των προσώπων κατά την εκτέλεση του προγράμματος.

imgDIm (Προαιρετικό): Στο όρισμα imgDIm δίνεται το προκαθορισμένο μέγεθος που θα πρέπει να έχει η εικόνα που θα πρέπει να επεξεργαστεί ο ταξινομητής για να γίνει σύγκριση με τις υπάρχουσες κλάσεις του dataset. Το όρισμα είναι προαιρετικό και σε περίπτωση που δεν χρησιμοποιηθεί το προκαθορισμένο μέγεθος της εικόνας θα είναι **96x96.**

--cuda (Προαιρετικό): Αν καλεστεί το όρισμα **cuda** τότε η επεξεργασία και ταξινόμηση των προσώπων θα εκτελεστεί από την κάρτα γραφικών του συστήματος. Σε αντίθετη περίπτωση, ο ταξινομητής θα αξιοποιήσει τον επεξεργαστή του υπολογιστικού συστήματος για να πραγματοποιήσει την ταξινόμηση.

--verbose (Προαιρετικό): Με την κλήση **verbose,** η εφαρμογή κατά την εκτέλεση της θα επιστρέφει ενημερωτικά μηνύματα στον χρήστη σχετικά με τον χρόνο που χρειάστηκε για να ολοκληρώσει τις διαδικασίες εντοπισμού και ταξινόμησης των προσώπων.

Infer: Με την κλήση **infer** καλείται ο subparser της python. Μετά από το συγκεκριμένο όρισμα, δίνεται η διαδρομή του φακέλου του αρχείου που δημιουργήθηκε από την εκπαίδευση του ταξινομητή. Στη συνέχεια, δίνεται η διαδρομή του αρχείου βίντεο που θα πρέπει να επεξεργαστεί η εφαρμογή.

--multi (Προαιρετικό): Με το όρισμα **multi** ο ταξινομητής για κάθε καρέ θα εντοπίσει όλα τα πρόσωπα και θα ταξινομήσει όλα τα πρόσωπα που εμφανίζονται στο βίντεο. Σε αντίθετη περίπτωση, ο ταξινομητής θα επεξεργαστεί μόνο το πρώτο πρόσωπο που θα εντοπίσει σε κάθε καρέ.

--delete (Προαιρετικό): Με την χρήση του ορίσματος **delete** ο φάκελος στον οποίο αποθηκεύονται τα ταξινομημένα καρέ του βίντεο διαγράφεται και εξάγεται μόνο το τελικό αρχείο βίντεο.

--step (Προαιρετικό): Με την χρήση του **step** μπορεί να οριστεί στον ταξινομητή το βήμα με βάση το οποίο θα επεξεργαστεί κάθε καρέ. Χρησιμοποιείται για αύξηση της ταχύτητας του ταξινομητή σε περιπτώσεις αρχείων βίντεο με υψηλό ρυθμό καρέ (framerate) όπου δεν είναι απαραίτητη η επεξεργασία κάθε καρέ βίντεο για την διαδικασία της ταξινόμησης. Σε περίπτωση που δεν οριστεί, ο ταξινομητής θα επεξεργαστεί κάθε καρέ του αρχείου εισόδου.

Εικόνα 82. Παράδειγμα κλήσης του Classifier με εμφάνιση πληροφοριών για τον χρόνο εκτέλεσης της κάθε διαδικασίας της εφαρμογής, της θέσης εντοπισμού του προσώπου και του ποσοστού βεβαιότητας για την πρόβλεψη.

```
deeplearning@deep-learning-virtual-machine:~/openface$ ./Classify/Classifier.py --verbose  infer ./generated-embeddings/classifier.pkl
 ./Videos/pavlidis.mp4
=== ./Videos/pavlidis/pavlidis0.jpg ===
  + Original size: (720, 960, 3)
Loading the image took 0.0812380313873 seconds.
Face detection took 1.64891386032 seconds.
Alignment took 0.0237519741058 seconds.
This bbox is centered at 571, 245
Neural network forward pass took 0.433490037918 seconds.
Prediction took 0.0379729270935 seconds.
Predict Pavlos Pavlidis with 1.00 confidence.
```

Εφόσον αρχίσει η εκτέλεση του ταξινομητή το πρόγραμμα θα ακολουθήσει την ακόλουθη διαδικασία. Το πρώτο βήμα είναι η προεπεξεργασία του αρχείου βίντεο που δίνεται ως είσοδος. Ο ταξινομητής θα δημιουργήσει ένα φάκελο στην διαδρομή όπου περιέχεται το αρχείο, με το ίδιο όνομα με αυτό. Με την βοήθεια της συνάρτησης **VideoCapture** της **OpenCV** βιβλιοθήκης, θα αποθηκεύσει κάθε καρέ στον φάκελο, και ως τίτλος θα δοθεί ο αριθμός του καρέ, για την ευκολότερη προσπέλαση των αρχείων.

Ακολούθως για κάθε καρέ, σύμφωνα με το βήμα που έχει οριστεί, θα γίνει η επεξεργασία και η ταξινόμηση του σε μία από τις κλάσεις του dataset. Η πρώτη συνάρτηση που καλείται κατά την επεξεργασία είναι η **getRep,** η οποία θα πρέπει να εντοπίσει τα πρόσωπα που μπορεί να υπάρχουν στο καρέ και να εξάγει τα **representations** τους, για την μετέπειτα ταξινόμηση τους. Αρχικά, με την χρήση των συναρτήσεων για image processing της **OpenCV** και της **dlib,** γίνεται ο εντοπισμός του προσώπου και ο σχεδιασμός του bounding box γύρω του. Μέσω της **OpenFace** γίνεται η επεξεργασία της εικόνας, με την εφαρμογή alignment και εύρεσης των landmarks του προσώπου που εντοπίστηκε και στην συνέχεια αποθηκεύονται τα **representations** και η θέση του προσώπου στο καρέ.

Το επόμενο βήμα είναι η σύγκριση των μετρήσεων που αποθηκεύτηκαν με τις αντίστοιχες μετρήσεις για την κάθε κλάση του dataset. Με την χρήση της **scikit-learn** εξάγεται για κάθε κλάση η πιθανότητα να ανήκει σε αυτή το πρόσωπο που έχει εντοπιστεί. Η κλάση με την μεγαλύτερη πιθανότητα είναι αυτή στην οποία ταξινομείται το πρόσωπο, και το όνομα της αποθηκεύεται κάτω από το bounding box της εικόνας μαζί με το ποσοστό βεβαιότητας της ταξινόμησης. Σε περιπτώσεις αλγορίθμων οι οποίοι επιστρέφουν ποσοστά προβλέψεων, όπως οι Μηχανές Διανυσματικής Στήριξης μπορεί να τεθεί στον κώδικα ένα threshold ποσοστού για κάτω από το οποίο δεν θα θεωρείται έγκυρη η πρόβλεψη.

Τέλος, εφόσον ολοκληρωθεί η παραπάνω διαδικασία για κάθε καρέ του βίντεο, με την χρήση του **VideoWriter** της **OpenCV** εξάγεται, στην ίδια τοποθεσία με το αρχικό βίντεο, το τελικό αρχείο βίντεο στο οποίο περιέχονται τα ταξινομημένα πρόσωπα.

Εικόνα 83. Παράδειγμα καρέ ύστερα από την εκτέλεση του ταξινομητή. Παρατηρείται το bounding box που έχει σχηματιστεί γύρω από το πρόσωπο που εντοπίστηκε και από κάτω του αναφέρεται το όνομα της κλάσης στην οποία ανήκει μαζί με το ποσοστό βεβαιότητας της πρόβλεψης. [96]

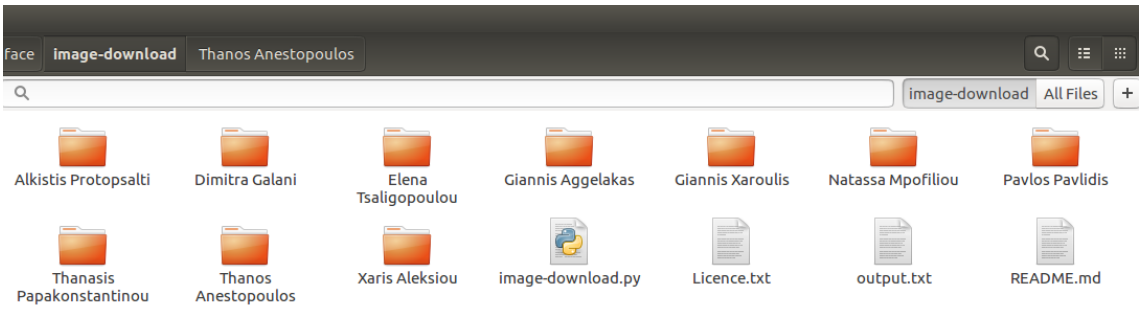
7.7 Συγκριτική Εκτέλεση Εφαρμογής με Ταξινομητές Μηχανών Διανυσματικής Στήριξης – Δέντρα Αποφάσεων

Σε αυτή την ενότητα θα παρουσιαστεί η εκτέλεση της διαδικασίας εκπαίδευσης του ταξινομητή αρχικά με ένα αλγόριθμο **«Δέντρων Αποφάσεων (Decision Tree)»** και στη συνέχεια με ένα **«Γραμμικών Μηχανών Διανυσματικής Στήριξης (Linear SVM)»**, και στη συνέχεια θα εκτελεστεί ο ταξινομητής ώστε να γίνει αναγνώριση προσώπων από τις κλάσεις του training dataset.

Το training dataset που θα χρησιμοποιηθεί, θα αποτελείται από 5 άνδρες και 5 γυναίκες καλλιτέχνες, και θα γίνει εκπαίδευση για 20, 50, 100 εικόνες ανά κλάση ώστε υπάρξει δυνατότητα εξαγωγής χρήσιμων συμπερασμάτων και για τις περιπτώσεις κλιμάκωσης των dataset. Επίσης, θα γίνει και η εκτέλεση του αλγόριθμου με όλες τις εικόνες που είναι εφικτό να εξαχθούν από την μηχανή αναζήτησης Google για κάθε κλάση. Το πρόβλημα που προκύπτει από την φύση του dataset, είναι πως πολλά από τα αποτελέσματα εικόνων που επιστρέφονται πιθανόν να περιέχουν διαφορετικούς μουσικούς, καλλιτέχνες, εξώφυλλα δίσκων και σε πολλές φορές αντικείμενα όπως μικρόφωνα θα εμφανίζονται στις εικόνες. Ο τρόπος επίλυσης τέτοιων θεμάτων είναι ο έλεγχος από τον χρήστη της εφαρμογής ώστε να εξασφαλιστεί η «καθαρότητα» του dataset.

Οι αλγόριθμοι θα εκτελεστούν σε virtual machine σε λειτουργικό περιβάλλον Ubuntu, με επεξεργαστή Intel Core i7 1,8 GHz.

Εικόνα 84. Τελική εικόνα του training dataset.



Το 1^{ο} βίντεο το οποίο θα χρησιμοποιηθεί στο εξής πείραμα ταξινόμησης προσώπων είναι μεγέθους 720x960 και σε αυτό εμφανίζεται ο καλλιτέχνης Παύλος Παυλίδης, για τον οποίο υπάρχει ήδη καταχωρημένη μία κλάση. Το βίντεο έχει διάρκεια 4 δευτερολέπτων και από αυτό εξάγονται **129** καρέ στα οποίο να εντοπίζεται κάποιο πρόσωπο.

Το 2^{ο} βίντεο το οποίο θα χρησιμοποιηθεί στο πείραμα είναι μεγέθους 360x636 και σε αυτό εμφανίζεται η καλλιτέχνιδα Χάρις Αλεξίου. Έχει διάρκεια 7 δευτερολέπτων και από αυτό εξάγονται **159** καρέ στα οποία απεικονίζεται κάποιο πρόσωπο.

7.7.1 Εκτέλεση Ταξινομητή Δέντρου Αποφάσεων (Decision Tree)

Ύστερα από την εκτέλεση του ταξινομητή έχουμε τα εξής αποτελέσματα:

Πίνακας 1. 1^{ο} βίντεο με χρήση Decision Tree

Πρόσωπα Ανά Κλάση	Ποσοστό Ακρίβειας	Μέσος Χρόνος Πρόβλεψης (sec)
20	0,643	1,166
50	0,279	0,934
100	0,248	0,920
~150	0,077	1,018

Πίνακας 2. 2^{ο} βίντεο με χρήση Decision Tree

Πρόσωπα Ανά Κλάση	Ποσοστό Ακρίβειας	Μέσος Χρόνος Πρόβλεψης (sec)
20	0,270	0,589
50	0,628	0,437
100	0,691	0,442
~150	0,471	0,439

7.7.2 Εκτέλεση Ταξινομητή Γραμμικών Μηχανών Διανυσματικής Στήριξης (Linear SVM)

Ύστερα από την εκτέλεση του ταξινομητή έχουμε τα εξής αποτελέσματα:

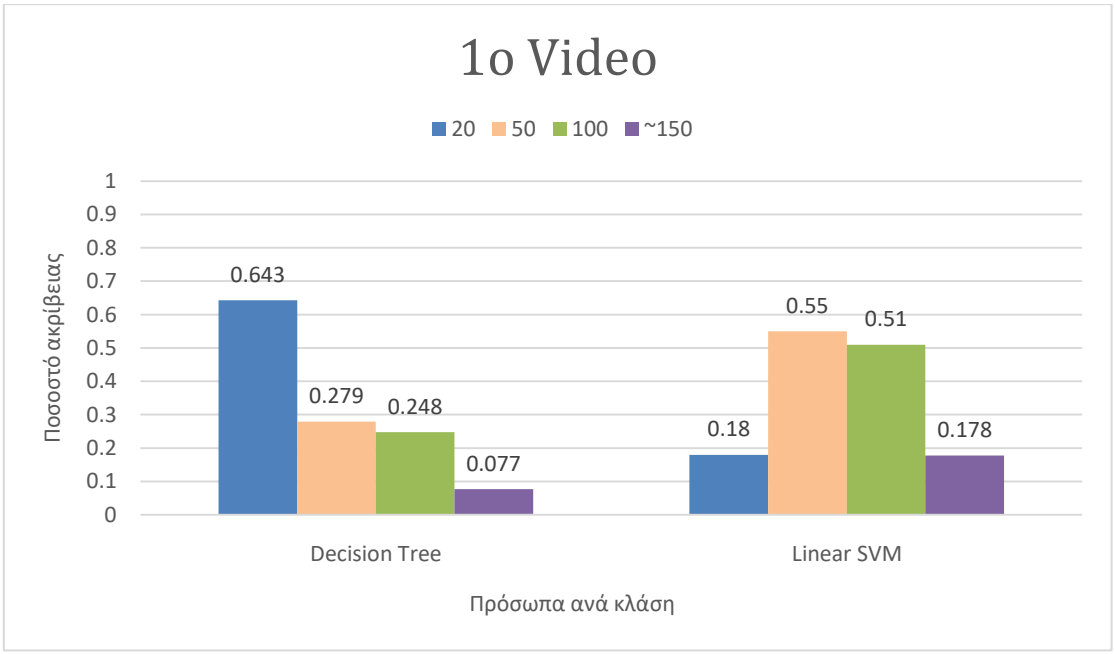
Πίνακας 3. 1ο βίντεο με χρήση Linear SVM

Πρόσωπα Ανά Κλάση	Ποσοστό Ακρίβειας	Μέσος Χρόνος Πρόβλεψης (sec)
20	0,180	1,195
50	0,550	0,975
100	0,510	1,089
~150	0.178	0,983

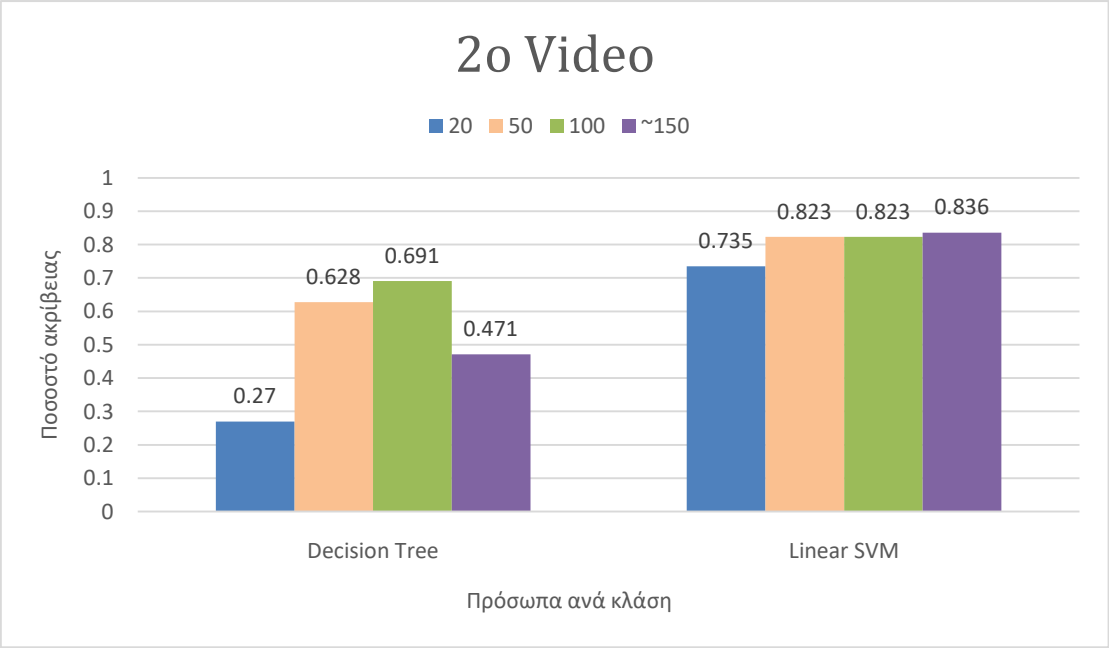
Πίνακας 4. 2ο βίντεο με χρήση Linear SVM

Πρόσωπα Ανά Κλάση	Ποσοστό Ακρίβειας	Μέσος Χρόνος Πρόβλεψης (sec)
20	0,735	0,562
50	0,823	0,401
100	0,823	0,420
~150	0,836	0,479

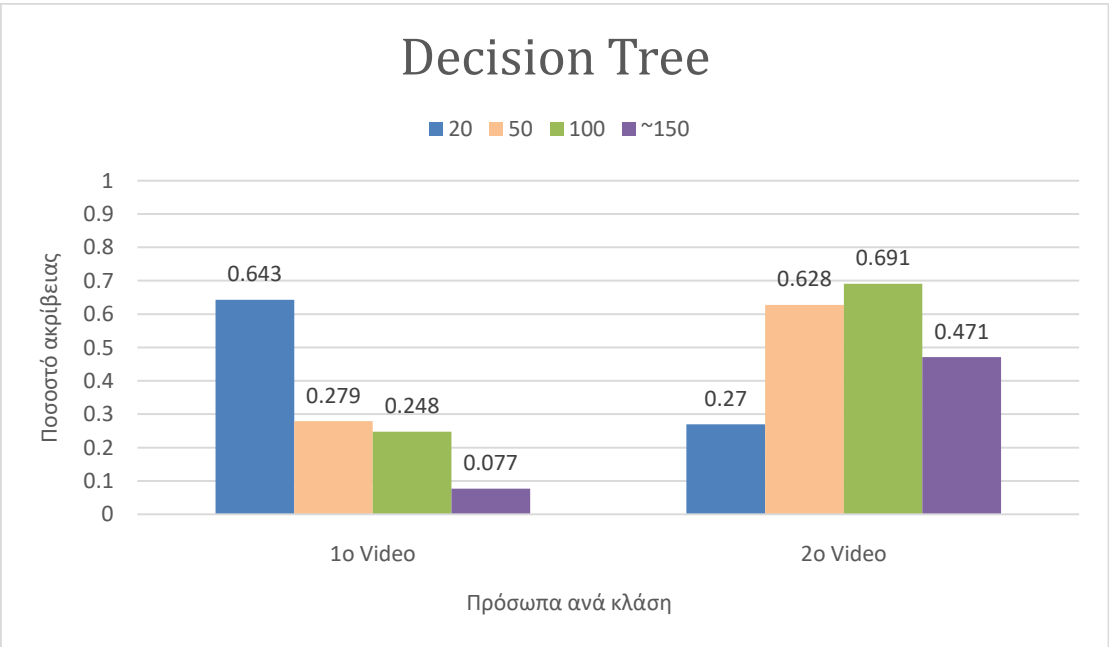
Διάγραμμα 1. Συγκριτική Απεικόνιση των Αποτελεσμάτων των Αλγόριθμων για το 1ο αρχείο βίντεο.



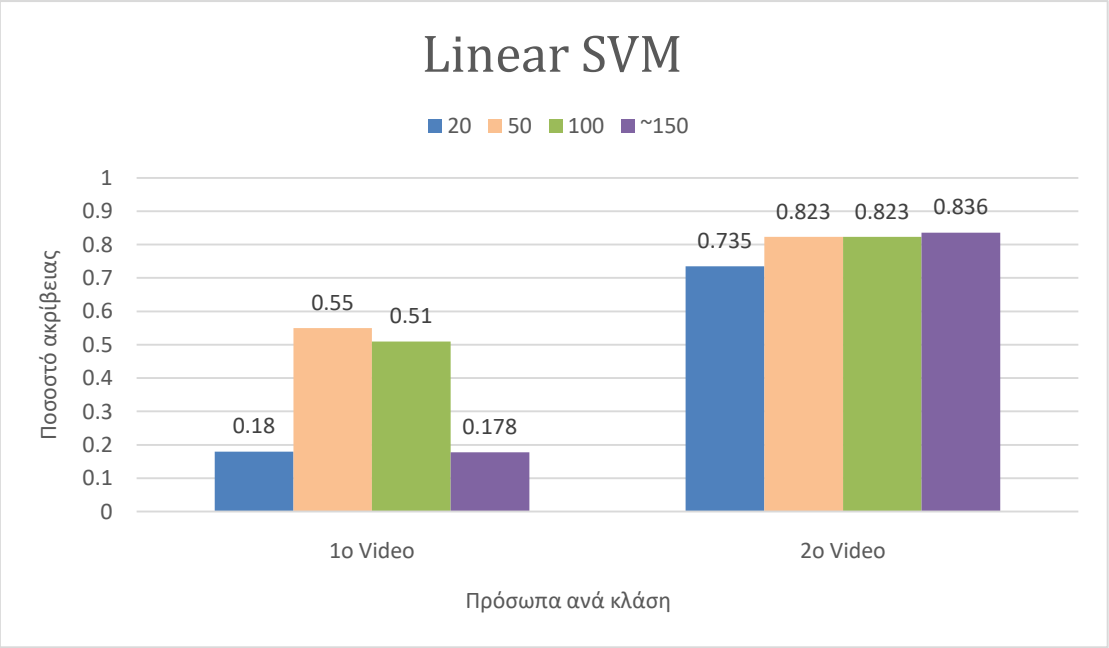
Διάγραμμα 2. Συγκριτική Απεικόνιση των Αποτελεσμάτων των Αλγόριθμων για το 2ο αρχείο βίντεο.



Διάγραμμα 3. Συγκριτική Απεικόνιση των Αποτελεσμάτων των αρχείων βίντεο με χρήση Δέντρων Αποφάσεων (Decision Tree).



Διάγραμμα 4. Συγκριτική Απεικόνιση των Αποτελεσμάτων των αρχείων βίντεο με χρήση Linear SVM (Γραμμικών Μηχανών Διανυσματικής Στήριξης).



7.8 Παρατηρήσεις - Συμπεράσματα

Με βάση τα αποτελέσματα των παραπάνω μετρήσεων, γίνεται άμεσα εμφανές πως αρκετές από τις υποθέσεις που έγιναν πριν την εκτέλεση του πειράματος επαληθεύτηκαν. Αρχικά, παρατηρούμε πως για ακραίες τιμές του training dataset παρουσιάζεται υψηλή διακύμανση στην αξιοπιστία των αποτελεσμάτων και η χρήση ενός dataset με ελάχιστα δεδομένα δεν είναι αξιόπιστη.

Επιπλέον, αντίστοιχη διακύμανση παρουσιάζεται και στην περίπτωση χρήσης όλων των διαθέσιμων δεδομένων, με αποτέλεσμα να αυξάνεται και ο αριθμός των σφαλμάτων των εικόνων του training dataset, όπως φαίνεται και από τα αποτελέσματα ακρίβειας του 1^{ου} βίντεο για τους δύο αλγόριθμους. Πρέπει να σημειωθεί επίσης πως η υψηλή ανάλυση του 1^{ου} βίντεο σε σύγκριση με το 2^{ο} έπαιξε σημαντικό ρόλο τόσο στην ποιότητα των αποτελεσμάτων, καθώς προκύπτουν θέματα λόγω dimensionality reduction, όσο και στον χρόνο επεξεργασίας από τον ταξινομητή.

Τα πιο αξιόπιστα αποτελέσματα και για τους δύο αλγόριθμους, φαίνεται πως εξάγονται με την χρήση ενός training dataset μεγέθους 50–100 εικόνες ανά κλάση, με αναλογικά αντίστοιχα ποσοστά ακριβείας και για τα δύο βίντεο. Όσον αφορά τους δύο αλγόριθμους, ο αλγόριθμος Μηχανών Διανυσματικής Στήριξης (Linear SVM) παρουσιάζει συγκριτικά με τον αλγόριθμο Δέντρων Αποφάσεων (Decision Tree) σημαντικά καλύτερα αποτελέσματα, εφόσον έχει γίνει εκπαίδευση των αλγορίθμων με ένα αξιόπιστο dataset και ειδικότερα στην περίπτωση του 2^{ου} βίντεο.

Τελικώς, λαμβάνοντας υπόψη όλα τα παραπάνω γίνεται σαφές πως η εφαρμογή για ταξινόμηση προσώπων που παρουσιάστηκε σε αυτή την μεταπτυχιακή διατριβή, εφόσον εκπαιδευτεί κατάλληλα, και με την χρήση ενός προεκπαιδευμένου Νευρωνικού Δικτύου όπως αυτό που παρέχεται από την OpenFace, έχει τη δυνατότητα να εξάγει αποτελέσματα άμεσα συγκρίσιμα με αντίστοιχες εφαρμογές που χρησιμοποιούν state-of-the-art συστήματα αναγνώρισης προσώπων με σημαντικά μεγαλύτερα dataset, πλησιάζοντας ακόμα και αποτελέσματα ανθρώπινης ακρίβειας. Είναι βέβαιο, πως με την ανάπτυξη νέων τεχνολογιών ανοιχτού κώδικα στο τομέα της Μηχανικής Μάθησης, όπως π.χ. η βιβλιοθήκη TensorFlow, είναι θέμα χρόνου μέχρι τα ποσοστά ανθρώπινης ακρίβειας να ξεπεραστούν από αντίστοιχες εφαρμογές αναγνώρισης και ταξινόμησης προσώπων.

8. Βιβλιογραφικές Αναφορές

1. Samuel, A., Some studies in machine learning using the game of checkers, IBM Journal of Research and Development, 1959:535-554
2. Kaelbing, L., Littman, Moore, A., Reinforcement learning: A Survey, Journal of Artificial Intelligence Research, 1996, 4:237-285
3. Sutton, R., Barto, A., Reinforcement Learning, 1998, 1st edition, MIT Press
4. Russell, S., Norvig, P., Artificial Intelligence: a modern approach, 2010, 3rd edition, Pearson
5. Szepersvari, C., Algorithms of Reinforcement Learning, 2010, 1st edition, Morgan & Claypool Publishers
6. Wiering, M., van Otterlo, M., Reinforcement Learning: State-of-the-Art, 2012, 1st edition, Springer
7. Fisher, R., The use of multiple measurements in taxonomic problems, Annals of Eugenics, 1936, 7(II):179-188
8. LeCun, Y., Cortes, C., Burges, C., The MNIST database of handwritten digits, online available: http://yann.lecun.com/exdb/mnist/
9. Murphy, K., Machine Learning: A probalistic perspective, 2012, 1st edition, MIT Press
10. Shalizi, C., 36-402, Undergraduate Advanced Data Analysis, 2016, Carnegie Mellon University, Online lecture notes, http://www.stat.cmu.edu/~cshalizi/uADA/16/
11. Covet, T., Hart, P., Nearest neighbor pattern classification, IEEE Transactions on Information Theory, 1967, 13(1):21-27
12. Hastie, T., Tibshirani, R., Friedman, J., The Elements of Statistical Learning, 2009, 2nd edition, Springer
13. Bishop, C., Pattern Recognition and Machine Learning, 2006, 1st edition, Springer
14. Domingos, P., A few useful things to know about Machine Learning, Communications of the ACM, 2012, 55(10):78-87
15. Krizhevsky, A., Learning multiple layers of features from tiny images, 2009, online available: http://www.cs.toronto.edu/~kriz/cifar.html
16. American Optimal Decisions, Portfolio Safeguard Help: L1 Distance, 2017, online available: http://www.aorda.com/html/PSG_Help_HTML/l1_distance.htm
17. Phillips, J., L7: Distances, Data Mining, University of Utua, 2013, online available: https://www.cs.utah.edu/~jeffp/teaching/cs5955/L7-Distances.pdf
18. Ambrosio, A., Vector p-norm, 2103, online available: http://planetmath.org/vectorpnorm
19. Differences between L1 and L2 as loss function and regulation, 2013, online available: http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/
20. Cortes, C., Vapnik, V., Support-Vector Networks, Machine Learning, 1995, 20(3), 273-297
21. Kecman, V., Wang, L., Support Vector Machines: Theory and Applications, 2005, 1st edition, Springer
22. Bottou, L., Cortes, C., Denker, L., Drucker, H., Guyon, I., Jackel, L., Lecun, Y., Muller, U., Sackinger, E., Simard P., Vapnik, V., Comparison of classifier methods: A case study in handwritten digit recognition, IARP (Ed.), Proceedings of the International Conference on Pattern Recognition, 1994, 2:77-82
23. Cherkassky, V., Mulier, F., Learning from Data: Concepts, Theory, and Methods, 2007, 2nd edition, Willey-IEEE Press

24. Nielsen, J., Robert, M., Usability Inspection Methods, 1994, 1st edition, John Wiley & Sons
25. Chu, F., Wang, L., Gene expression data analysis using support vector machines, Proceedings of IEEE International Joint Conference on Neural Networks, 2003:2268-2271
26. Meyer, D., Leisch, F., Hornik, K., The support vector machine under test, Neurocomputing, 2003, 55:169-186
27. Wang, L., Support Vector Machines: Theory and applications, 2005, 1st edition, Springer
28. Scholkopf, B., Smola, A., Learning with Kernels, 2001, 1st edition, MIT Press
29. Mitchell, T., Machine Learning, 1997, 1st edition, McGraw Hill
30. Γεωργούλη, Κ., Τεχνητή Νοημοσύνη, 2015, online available: http://repfiles.kallipos.gr/html_books/93/img_book/sxima_4,14.png
31. Breiman, L., Friedman, J., Olshen, R., Stone, C., Classification and Regression Trees. Wadsworth, Belmont, CA, 1984
32. Quinlan, J., R., C4. 5: programs for machine learning. Morgan Kaufmann, 1993
33. Hastie, T., Tibshirani, R., Friedman, J., Elements of Statistical Learning, 2009, 1st edition, Springer,
34. Van Der Malsburg, C., Rosenblatt, F., Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms, Brain Theory, 1986:245-248
35. Buduma, N., Lacascio, N., Fundamentals of Deep Learning, 2017, 1st edition, O'Reilly
36. Krizhevsky, A., Sutskever, I., Hinton, G., ImageNet Classification with Deep Convolutional Neural Networks, 2012, Advances in Neural Information Processing Systems
37. Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y., Maxout Networks, Proceedings of Machine Learning Research, 2013
38. LeCun, Y., Boser, B., Denker, J., Henderson, D., Howard, R., Hubbard, W., Jackel, L., Backpropagation applied to handwritten Zip Code recognition, Neural Computation, 1989, 1:541-551
39. Zhou, Y., Chellappa, R., Vaid, A., Jenkins, B., Image restoration using a neural network, IEEE Transactions on Acoustics, Speech, and Signal Processing, 1988, 36(7):1141-1151
40. Boureau, Y., Ponce, J., LeCun, Y., A theoretical analysis of feature pooling in visual recognition, Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, 111-118
41. LeCun, Y., Bottu, L., Bengio, Y., Haffner, P., Gradient-based learning applied to document recognition, Proceedings of the IEEE, 1998, 1-46 , 2278-2324
42. Zeiler, M., Fergus, R., Visualizing and understanding convolutional network, ECCV 2014: Lecture Notes in Computer Science, 8689:818-833
43. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erthan, D., Vanhoucke, V., Rabinovich, Going deeper with convolutions, Computer Vision and Pattern Recognition, 2015
44. Szegedy, C., Ioffe, S., Vanhouche, V., Alemi, A., Inception-v4, Inseption-ResNet and the Impact of residual connections on learning, Computer Vision and Pattern Recognition, 2016
45. Simonyan, K., Zisserman, A., Very Deep Convolutional networks for large-scale image recognition, Computer Vision and Pattern Recognition, 2014

46. He, K., Zhang, X., Ren, S., Sun, J., Deep Residual learning for image recognition, Computer Vision and Pattern Recognition, 2015
47. H., K., Zhang, X., Ren, S., Sun, J., Identity mappings in deep residual networks, Computer Vision-ECCV 2016-Lecture Notes in Computer Science book series, 2016, 9908
48. Amos, B., Ludwiczuk, B., Mahadev, S., OpenFace: A general-purpose face recognition library with mobile applications, CMU-CS-16-188, CMU School of Computer Science, Tech. Rep., 2016
49. Taigman, Y., Yang, M., Ranzato, M., Wolf, L., Deepface: Closing the gap to human-level perfomance in face verification, IEEE Conference on CVRP, 2014:1701-1708
50. Weinberger, K., Blitzer, J., Saul L., Distance metric learning for large margin nearest neighbor classification, Journal of Machine Learning Research, 2009, 10:207-244
51. Sun, Y., Wang, X., Tang, X., Deep learning face representation by joint identification-verification, NIPS'14 Proceedings of the 27th International Conference on Neural Information Processing, 2014, 2:1988-1996
52. Lin, M., Chen, Q., Yan, S., Network in network, CoRR, abs/1312.4400
53. Rumelhart, D., Hinton, G., Williams, R., Learning representations by back-propagating errors, Nature, 1986, 323:533-536
54. Duchi, J., Hazan, E., Singer, Y., Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 2011, 12:2121–2159
55. Huang, G., Ramesh, M., Berg, T., Learned-Miller, E., Labeled faces in the wild: A database for studying face recognition in unconstrained environments, Technical Report 07-49, University of Massachusetts, Amherst, 2007
56. Chen, D., Ren, S., Wei, Y., Cao, X., Sun, J., Joint cascade face detection and alignment, Proceedings ECCV, 2015
57. Taigman, Y., Yang, M., Ranzato, M., Wolf, L., Deepface: Closing the gap to human-level performance in face verification. In IEEE Conf. on CVPR, 2014
58. Sun, Y., Wang, X., Tang, X., Deeply learned face representations are sparse, selective, and robust. CoRR, 2014
59. Collobert, R., Kavukcuoglu, K., Farabet, C., Torch7: A matlab-like environment for machine learning, BigLearn, NIPS Workshop, 2011
60. Ierusalimschy, R., Henrique de Figueiredo, L., Celes Filho, W., Lua-An extensible extension language, Software: Practice and Experience, 1996, 26(6):635-652
61. Pall, M., The LuaJIT Project, 2017, online available: http://luajit.org/
62. Van Rossum, G., Jr. Drake, F., An introduction to Python, 2011, Network Theory Ltd.
63. Oliphant, T., Guide to NumPy, 2015, 2nd edition, CreateSpace Independent Publishing Platform
64. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blonder, M., Prettenhofer, P., Weiss, R., Dudourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchenay, E., Scikit-learn: Machine Learning in Python, THe Journal of Machine Learning Research, 2011, 12:2825-2830
65. Davis E King. Dlib-ml: A machine learning toolkit. The Journal of Machine Learning Research, 2009, 10:1755–1758
66. Kazemi, V., Sullivan, J., One millisecond face alignment with an ensemble of regression trees, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, 1867-1874

67. Yi, D., Lei, Z., Liao, S., Li, S., Learning face representation from scratch, arXin, 2014
68. Ng, H. W., Winkler, S., A data-driven approach to cleaning large face datasets, IEEE International Conference on Image Processing, 2014, 265(265):530
69. Schroff, F., Kalenichenko, D., Philbin, J., FaceNet: A unified embedding for face recognition and clustering, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, 815-823
70. Taigman, Y., Yang, M., Ranzato, M., Wolf, L., Deepface: Closing the gap to human-level performance in face verification, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014, 1701-1708
71. Turk, M., Pentland, A., Eigenfaces for recognition, Journal of Cognitive neuroscience, 1991, 3(1):71-86
72. Belhumeur, P., Hespanha, J., Kriegman, D., Eigenfaces vs Fisherfaces: Recognition using class specific linear projection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, 19(7):711-720
73. Ahonen, T., Hadid, A., Pietikainen, M., Face recognition with local binary patterns, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2006, 28(12):2037-2041
74. Van der Maaten, L., Accelerating t-SNE using Tree-Based algorithms, Journal of Machine Learning Research, 2014, 15(Oct):3221-3245
75. Van der Maaten, L., Hinton, G., Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research, 2008, 9(Nov):2579-2605
76. Amos, B., Visualizing representations with t-SNE, CMU-CS-16-188, CMU School of Computer Science, Tech. Rep., 2016
77. Parkhi, O., Vedaldi, A., Zisserman, A., Deep Face Recognition, British Machine Vision Conference, 2015
78. Wu, X., He, R., Sun, Zhenan, S., Tan, T., A light CNN for Deep Face Representation with Noisy Labels, arXiv, 2015
79. Nosrati, M., Python: An appropriate language for real world programming, World Applied Programming, 2011, 1(2):110-117
80. Ierusalimschy, R., Henrique de Figueiredo, L., Celes, W., The evolution of Lua, Proceedings of the 3rd ACM SIGPLAN conference on history of programming languages, 2007
81. Ronan, C., Koray, S., Torch: a scientific computing framework for LuaJIT, online available: http://torch.ch/
82. Pisarevsky, V., Bouguet, J., Open Source Computer Vision Library, 2006, 1st edition, Springer
83. OpenCVteam, Open Source Computer Vision Library, online available: https://opencv.org/about.html
84. Wechsler, H., Phillips, J., Bruce, V., Fogelman-Soulie, F., Huang T., Face Recognition: from theory to applications, 1998, 1st edition, Springer
85. scikitLearn, Classifier comparison, 2017, online available: http://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html#example-classification-plot-classifier-comparison-py
86. Dlib C++ Library, Real-Time Face Pose estimation, 2014, online available: http://blog.dlib.net/2014/08/real-time-face-pose-estimation.html

87. Douglas, C., Case Study: Extraction of color HAAR features, CMSoft, 2015, online available: http://www.cmsoft.com.br/opencl-tutorial/case-study-extraction-color-haar-features/
88. Karpathy, A., CS231n: Convolutional Neural Networks for Visual Recognition, 2017, University of Stanford
89. Berwick, R., An Idiot's Guide to Support Vector Machines (SVMs), 2003, Cambridge, MA: MIT, online available: http://web.mit.edu/6.034/wwwbob/svm-notes-long-08.pdf
90. Scikit-learn Documentation, Decision Trees, online available: http://scikit-learn.org/stable/modules/tree.html
91. Goodfellow, I., Bengio, Y., Courville, A., Deep Learning (Adaptive Computation and Machine Learning series), 2016, 1st edition, MIT Press
92. Amos, B., OpenFace, DNN Models, 2015-2016, Carnegie Mellon University, online available: https://cmusatyalab.github.io/openface/
93. Amos, B., OpenFace API Documentation, 2015-2016, Carnegie Mellon University, online available: http://openface-api.readthedocs.io/en/latest/index.html
94. Nielsen, M., Neural Networks and Deep Learning, 2015, 1st edition, Determination Press
95. https://www.patrasevents.gr/imgsrv/Misc/thanasis_papakonstantinou9.jpg
96. Συνέντευξη του Παύλου Παυλίδη στο elculture.gr online available: https://www.youtube.com/watch?v=xxBP0WULodE

Παράρτημα Α

Documentation Openface

Στη συνέχεια παρατίθενται τα σημαντικότερα functions που παρέχονται από την OpenFace. Εναλλακτικά μπορούν να βρεθούν στην τοποθεσία https://github.com/cmusatyalab/openface/tree/master/api-docs:

openface.AlignDlib class

***class* openface.AlignDlib(facePredictor)**

Χρησιμοποιείται για την ευθυγράμμιση προσώπων με βάση το πρότυπο που παρέχεται από την dlib. [86] Δημιουργεί ένα αντικείμενο 'AlignDlib'.

Παράμετροι: facePredictor (str) – Η διαδρομή για τον landmark estimator της dlib.

***function* align(imgDim, rgbImg, bb=None, landmarks=None, landmarkIndices=INNER_EYES_AND_BOTTOM_LIP)**

Μετασχηματισμός και ευθυγράμμιση ενός προσώπου σε μία εικόνα

Παράμετροι:

- **imgDim** (*int*) – Το μήκος των άκρων σε pixels του τετραγώνου, στο οποίο μετασχηματίζεται η εικόνα.
- **rgbImg** (*numpy.ndarray*) –RGB εικόνα προς επεξεργασία. Μορφή: (ύψος, πλάτος, 3)
- **bb** (*dlib.rectangle*) –Bounding box το οποίο σχηματίζεται γύρω από το πρόσωπο. Σχηματίζεται αυτόματα στο μεγαλύτερο πρόσωπο της κάθε εικόνας.
- **landmarks** (*list of (x,y) tuples*) – Σημεία στα οποία ανιχνεύτηκαν τα landmarks του προσώπου.
- **landmarkIndices** (*list of ints*) – Δείκτες βάσει των οποίων θα γίνει ο μετασχηματισμός.
- **skipMulti** (*bool*) – Σε περίπτωση που υπάρχει πάνω από ένα πρόσωπο προσπέρσμα της εικόνας χωρίς εντοπισμό προσώπου.

Επιστρέφεται: Η τελική ευθυγραμμισμένη RGB εικόνα. Μορφή εικόνας (imgDIm, imgDIm, 3)

Τύπος: numpy.ndarray

***function* findLandmarks(rgbImg, bb)**

Εύρεση όλων των landmarks ενός προσώπου

Παράμετροι:

- **rgbImg** (*numpy.ndarray*) – RGB εικόνα προς επεξεργασία. Μορφή: (ύψος, πλάτος, 3)
- **bb** (*dlib.rectangle*) – Bounding box το οποίο σχηματίζεται γύρω από το πρόσωπο για το οποίο θα εντοπιστούν τα landmarks.

Επιστρέφεται: Οι τοποθεσίας των landmarks που ανιχνεύτηκαν

Τύπος: Λίστα από tuples (x,y)

***function* getAllFaceBoundingBoxes(rgbImg)**

Εύρεση όλων των bounding boxes προσώπων σε κάθε εικόνα.

Παράμετροι:

- **rgbImg** (*numpy.ndarray*) – RGB εικόνα προς επεξεργασία. Μορφή: (ύψος, πλάτος, 3).

Επιστρέφεται: Τα bounding boxes για κάθε πρόσωπο που εντοπίστηκε στην εικόνα

Τύπος: dlib.rectangles

***function* getLargestFaceBoundingBox(rgbImg, skipMulti=False)**

Εύρεση του μεγαλύτερου bounding box σε μία εικόνα.

Παράμετροι:

- **rgbImg** (*numpy.ndarray*) – RGB εικόνα προς επεξεργασία. Μορφή: (ύψος, πλάτος, 3).
- **skipMulti** (*bool*) – Προσπέραση της εικόνας σε περίπτωση που εντοπιστούν παραπάνω από 1 πρόσωπα.

Επιστρέφεται: Το bounding box για το μεγαλύτερο πρόσωπο σε μία εικόνα ή Τίποτα.

Τύπος: dlib.rectangle

openface.TorchNeuralNet class

***class* openface.TorchNeuralNet(self, model=defaultModel, imgDim=96, cuda=False)**

Χρήση του subprocess του Torch για εξαγωγή χαρακτηριστικών. Δημιουργεί ένα αντικείμενο 'TorchNeuralNet'. Καλεί το πρόγραμμα **openface_server.lua.**

Παράμετροι:

- **model** (*str*) – Η διαδρομή για την τοποθεσία του μοντέλου Torch. (*defaultModel=* '/home/docs/checkouts/readthedocs.org/user_builds/openface-api/checkouts/latest/openface/../models/openface/nn4.small2.v1.t7')
- **imgDim** (*int*) – Το μήκος των άκρων του τετραγώνου της εικόνας εισόδου.
- **cuda** (*bool*) – Χρήση ή όχι του CUDA.

***function* forward(rgbImg)**

Εκτέλεση του νευρωνικού δικτύου με είσοδο μία εικόνα RGB.

Παράμετροι:

- **rgbImg** (*numpy.ndarray*) – RGB εικόνα προς επεξεργασία. Μορφή: (imgDim, imgDim, 3).

Επιστρέφεται: Διάνυσμα χαρακτηριστικών που εξάχθηκαν από το νευρωνικό δίκτυο.

Τύπος: numpy.ndarray

***function* forwardPath(imgPath)**

Εκτέλεση του νευρωνικού δικτύου με είσοδο μία εικόνα από τον δίσκο.

Παράμετροι:

- **imgPath** (str) – Η διαδρομή της τοποθεσίας της εικόνας.

Επιστρέφεται: Διάνυσμα χαρακτηριστικών που εξάχθηκαν από το νευρωνικό δίκτυο.

Τύπος: numpy.ndarray

openface.data module

Module για δεδομένα εικόνων.

***class* openface.data.Image(cls, name, path)**

Αντικείμενο το οποίο χρησιμοποιεί μεταδεδομένα (metadata) εικόνων. Δημιουργεί ένα αντικείμενο 'Image'.

Παράμετροι:

- **cls** (str) – Η κλάση της εικόνας, το όνομα του προσώπου
- **name** (str) – Η ονομασία της εικόνας
- **path** (str) – Η διαδρομή της τοποθεσίας της εικόνας στο δίσκο.

***function* getBGR()**

Ανάγνωση της εικόνας από το δίσκο σε μορφή BGR.

Επιστρέφεται: Εικόνα BGR. Μορφή: (ύψος, πλάτος, 3)

Τύπος: numpy.ndarray

***function* getRGB()**

Ανάγνωση της εικόνας από το δίσκο σε μορφή RGB.

Επιστρέφεται: Εικόνα RGB. Μορφή: (ύψος, πλάτος, 3)

Τύπος: numpy.ndarray

openface.data.iterImgs(directory)

Προσπέλαση εικόνων σε ένα φάκελο.

Οι εικόνες πρέπει να είναι αποθηκευμένες σε υποφακέλους με την ονομασία του κάθε προσώπου (κλάσης) με την παρακάτω μορφή.

Εικόνα 85. Παράδειγμα μορφής φακέλων ανά κλάση. [93]

```
$ tree directory
person-1
├── image-1.jpg
├── image-2.png
...
└── image-p.png

...

person-m
├── image-1.png
├── image-2.jpg
...
└── image-q.png
```

Παράμετροι: directory (str) – Η τοποθεσία στην οποία θα γίνει η προσπέλαση.

Επιστρέφεται: Προσπέλαση αντικειμένων εικόνων.

openface.helper module

Συναρτήσεις βοήθειας της OpenFace.

openface.helper.mkdirP(path)

Δημιουργία ενός φακέλου και μη επιστροφή σφάλματος σε περίπτωση όπου η διαδρομή υπάρχει ήδη. Σε περίπτωση που υπάρχει ήδη ο φάκελος, δεν γίνεται κάποια ενέργεια

Παράμετροι: path (str) – Ο φάκελος που θα δημιουργηθεί.

Παράρτημα Β

Συγκριτικά Μοντέλα CNN της Openface

Στη συνέχεια παρουσιάζεται τα μοντέλα νευρωνικών δικτύων που παρέχει η OpenFace για ελεύθερη χρήση.

Ορισμοί Μοντέλων

Μοντέλο	Αριθμός Παραμέτρων
nn4.small2	3733968
nn4.small1	5579520
nn4	6959088
nn2	7472144

Τα παραπάνω μοντέλα έχουν εκπαιδευτεί από τον συνδυασμό των δύο μεγαλύτερων dataset για αναγνώριση προσώπων που είναι διαθέσιμα για κοινή χρήση: **FaceScrub** και **CASIA-WebFace.** [67] [68]

Απόδοση των Μοντέλων

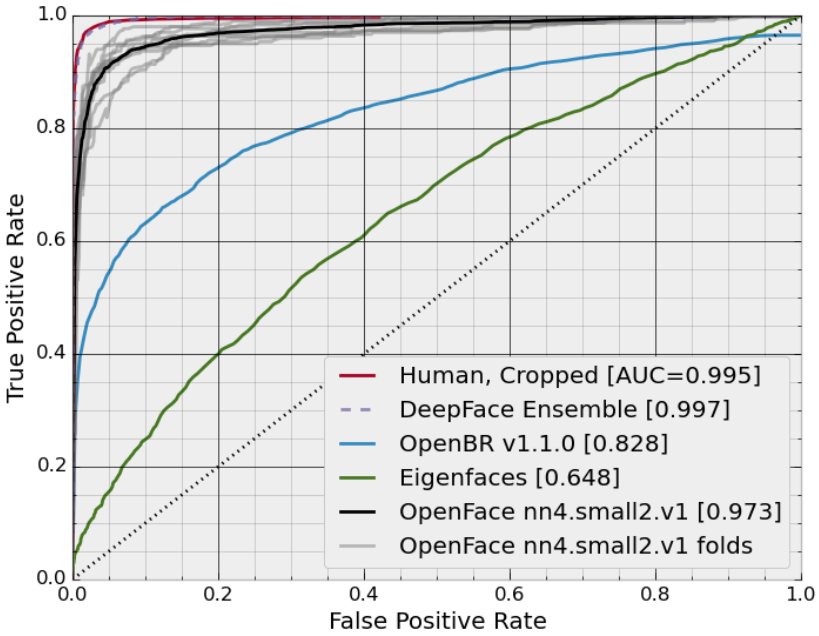
Τα αποτελέσματα των μετρήσεων για την απόδοση του κάθε μοντέλου προέρχονται από τον υπολογισμό του μέσου όρου για 500 προσπελάσεις του νευρωνικού δικτύου. Οι μετρήσεις έγιναν με την χρήση της **OpenBLAS** με έναν **8-πύρηνο επεξεργαστή 3,7 GHz** και με την χρήση μίας κάρτας γραφικών **Tesla K40**.

Μοντέλο	Χρόνος Εκτέλεσης CPU	Χρόνος Εκτέλεσης GPU
nn4.small2.v1	0.9292 ± 0.0134	0.973
nn4.small1.v1	0.9210 ± 0.0160	0.973
nn4.v2	0.9157 ± 0.0152	0.966
nn4.v1	0.7612 ± 0.0189	0.853
FaceNet (Reference)	0.9963 ± 0.009	-

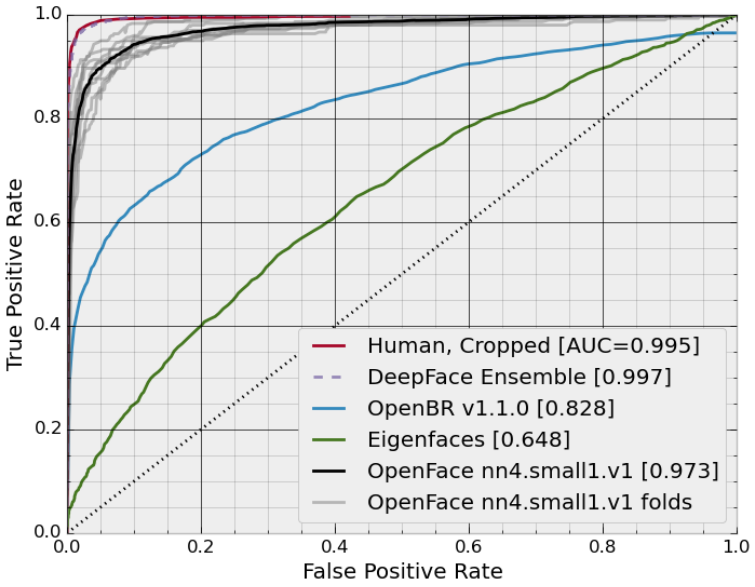
Καμπύλες ROC

Ακολουθούν οι καμπύλες ROC με συγκριτικά αποτελέσματα του κάθε μοντέλου σε σχέση με τα υπόλοιπα μοντέλα νευρωνικών δικτύων που είναι διαθέσιμα στο ευρύ κοινό.

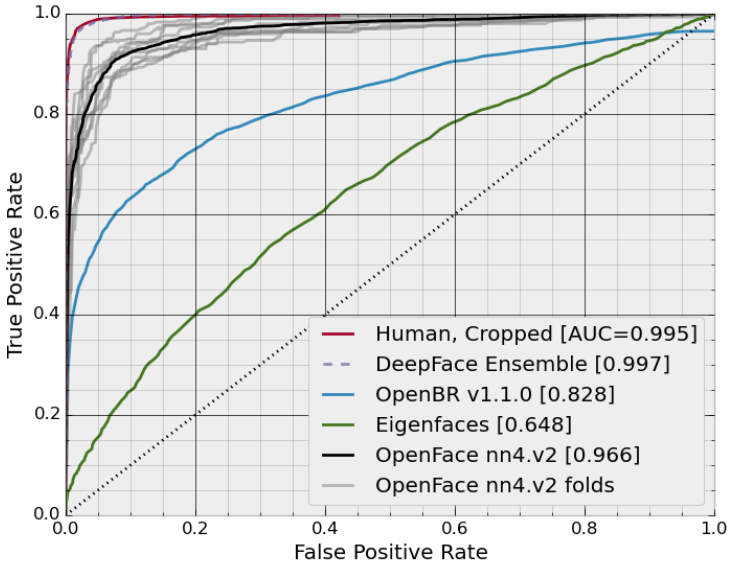
Εικόνα 86. nn4.small2.v1 [92]



Εικόνα 87. nn4.small1.v1 [92]



Εικόνα 88. Nn4.v2 [92]



Εικόνα 89. nn4.v1 [92]