ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΦΥΣΙΚΗΣ

ΕΡΓΑΣΤΗΡΙΟ ΗΛΕΚΤΡΟΝΙΚΗΣ

ΕΞΑΓΩΓΗ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΓΙΑ ΕΠΑΛΗΘΕΥΣΗ OFF-LINE ΥΠΟΓΡΑΦΩΝ ΜΕΣΩ ΣΥΝΕΛΙΚΤΙΚΩΝ ΝΕΥΡΩΝΙΚΩΝ ΔΙΚΤΥΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΤΣΑΓΚΟΠΟΥΛΟΣ ΝΙΚΟΛΑΟΣ

Α.Μ. 5387

Επιβλέπων: Γεώργιος Οικονόμου

Πάτρα, Ιούλιος 2017

Στην οικογένειά μου

Περίληψη

Αντικείμενο της παρούσας διπλωματικής εργασίας είναι η μελέτη των τεχνητών νευρωνικών δικτύων στα πλαίσια της μηχανικής μάθησης που ανθίζει τα τελευταία χρόνια. Συγκεκριμένα, εστιάσαμε στα συνελικτικά νευρωνικά δίκτυα και στους τρόπους με τους οποίους μπορούμε να εξάγουμε χαρακτηριστικά από αυτά.

Η επαλήθευση της *off-line* υπογραφής έχει μελετηθεί αρκετά τα τελευταία χρόνια, χρησιμοποιώντας τεχνικές από διάφορους κλάδους όπως την ψηφιακή επεξεργασία σήματος, την γραφολογία, την υπολογιστική όραση και άλλες. Εκμεταλλευόμενοι την ιδιαίτερη επιτυχία των ταξινομητών προσπαθήσαμε να εκπαιδεύσουμε έναν ώστε να είναι σε θέση να εξάγει χαρακτηριστικά από πλαστές και γνήσιες υπογραφές.

Η προσέγγισή μας έγινε με τη δημιουργία ενός συνελικτικού δικτύου δύο κλάσεων, το οποίο εκπαιδεύτηκε πάνω σε ένα *training set D,* από το οποίο στο τέλος εξαγάγαμε χαρακτηριστικά γνησίων και πλαστών υπογραφών (*writer independent*). Στη συνέχεια, σε ένα δεύτερο *training set E,* εξαγάγαμε χαρακτηριστικά μέσω του *cnn* και εκπαιδεύσαμε έναν *writer dependent* ταξινομητή.

Για τη γενίκευση, από το *testing set E,* χρησιμοποιήσαμε καινούρια δείγματα για την εξαγωγή χαρακτηριστικών με σκοπό την επαλήθευση από τον *writer dependent classifier.*

Μετρήσαμε μέσο *EER* για 10 γραφείς ίσο με 13,54% για πιστά αντίγραφα υπογραφών δείχνοντας ότι η μελέτη των συνελικτικών δικτύων για την εξαγωγή χαρακτηριστικών αποφέρει αποτελέσματα.

Abstract

This B.Sc. thesis focuses on the research of off-line signature verification with use of convolutional neural networks. Specially, we tried to extract features through them.

Off-line signature verification has been researched over the last few decades using techniques from multiple areas like digital signal processing, graphology, computer vision and more. Taking advantage the great results of classifiers we trained a cnn which extracts features from genuine and forged signatures.

Our approach was through the creation of a convolutional neural network with two classes, and trained on a training set D. Then we extracted features based on genuine and forgeries signatures (writer independent). Continuing, based on a second training set E, we extracted features through the first cnn and we trained a writer dependent classifier.

To this end, from the testing set E we used new samples for feature extracting targeting the verification from the writer dependent classifier. We obtained mean *EER* for 10 writers equal to 13,54% based on skill forgeries.

Ευχαριστίες

Στα πλαίσια αυτής διπλωματικής εργασίας, είχα την τιμή να συνεργαστώ με τον κ. Οικονόμου του τμήματος φυσικής, που μου έδωσε τη δυνατότητα να εργαστώ πάνω σε ένα θέμα το οποίο ακμάζει ιδιαίτερα τα τελευταία χρόνια. Ιδιαίτερη βοήθεια παρείχε ο κ. Ζώης Ηλίας, καθηγητής του Τ.Ε.Ι Αθήνας, ο οποίος παρείχε αστείρευτη βιβλιογραφία, κώδικες, καθώς και προτάσεις για διαφορετικούς τρόπους προσέγγισης. Θα ήθελα να ευχαριστήσω επίσης, τον Ηλία Θεοδωρακόπουλο και τη Γιάννα Ντίνου από το εργαστήριο ηλεκτρονικής, οι οποίοι ήταν πάντα διαθέσιμοι σε κάθε δυσκολία και τέλος την οικογένειά μου, που μου στάθηκε από την αρχή των σπουδών μου έως τώρα.

Περιεχόμενα

Κεφάλαιο 1: Εισαγωγή στη βιομετρία

Η ανάγκη αναγνώρισης του ατόμου, υπήρχε από πάντα στην κοινωνία και αποτελούσε απαραίτητη τόσο στην καθημερινή συναναστροφή του, όσο και στις υπηρεσίες που την απαρτίζουν. Οι άνθρωποι, τυπικά, αναγνωρίζονται μεταξύ τους βάσει σωματικών χαρακτηριστικών (π.χ. πρόσωπο, φωνή, στάση του σώματος) καθώς και άλλων συναφών πληροφοριών (π.χ. εθνικότητα, γλώσσα, ρουχισμός). Τέτοια χαρακτηριστικά συγκροτούν την ταυτότητά του. Υπηρεσίες που δίνουν πρόσβαση ηλεκτρονική (όπως αρχεία, αλληλογραφία, τραπεζικές συναλλαγές, πανεπιστημιακές υπηρεσίες) τόσο και φυσική (πρόσβαση σε χώρες, τοποθεσίες ασφαλείας κρατικές ή μη) απαιτούν ασφαλέστερη διαχείριση ταυτότητας.

Η θεμελιώδης διαδικασία στον έλεγχο ταυτοποίησης είναι η καθιέρωση μιας σύνδεσης μεταξύ ενός ατόμου και της προσωπικής του ταυτότητας. Η αναγνώριση ενός ατόμου βασίζεται στις ακόλουθες τρεις μεθόδους:

i. τι γνωρίζει
ii. τι βρίσκεται στην κατοχή του
iii. ποιος είναι

Η πρώτη μέθοδος βασίζεται στη γνώση κάποιας πληροφορίας. όπως κωδικών πρόσβασης, αριθμών εγγράφων (ταυτότητας, άδειας οδήγησης, διαβατηρίου) καθώς η δεύτερη σε αντικείμενα που κατέχει (κρυπτογραφημένων usb, καρτών εισόδου, κλειδιών, κινητών και άλλων προσωπικών αντικειμένων) και η τρίτη στο ποιος είναι πραγματικά.

Πληροφορίες και αντικείμενα που δίνουν πρόσβαση, είναι εύκολα αντιληπτό ότι μπορούν να κλαπούν, να μεταβιβαστούν από τον κάτοχό τους σε τρίτους ακόμη και να αρνηθούν από τον ίδιο προς την υπηρεσία. Οπότε, η μέθοδος που βασίζεται στο ποιος είναι κάποιος πραγματικά, καθιστά την πιο έγκυρη και αξιόπιστη μορφή ταυτοποίησης.

Το ποιος είναι κάποιος, καθορίζεται από τα βιομετρικά του χαρακτηριστικά και η ταυτοποίηση μέσω αυτής της μεθόδου ονομάζεται βιομετρική αναγνώριση (biometric recognition). Ως βιομετρικά, θεωρείται το σύνολο των χαρακτηριστικών που κατέχει κάποιος εκ γενετής και των χαρακτηριστικών που οφείλονται στη συμπεριφορά του. Συνεπώς, η βιομετρική αναγνώριση αποτελεί την πιο αξιόπιστη μέθοδο, καθώς τέτοια χαρακτηριστικά είναι δύσκολο να αλλοιωθούν, να χαθούν, να χειραγωγηθούν και να μεταβιβαστούν σε τρίτους, δημιουργώντας μια ισχυρή σύνδεση μεταξύ του ατόμου και της ταυτότητάς του.

Κάθε άτομο που παρουσιάζει κάποιο βιομετρικό χαρακτηριστικό του σε κάποιο σύστημα για αναγνώριση ονομάζεται *χρήστης του συστήματος*. Η διαδικασία αναγνώρισης απαιτεί τη φυσική παρουσία του χρήστη, με αποτέλεσμα να τον αποτρέπει από την ψευδή δήλωση στοιχείων και ταυτόχρονα καθιστά αδύνατη την καταγραφή του στο σύστημα

δύο φορές (σε περίπτωση απάτης) όπως και την προσπάθεια εισόδου υποδυόμενος διαφορετικό χρήστη.

1.1 Βιομετρικά συστήματα

Ένα βιομετρικό σύστημα συλλέγει ένα ή περισσότερα χαρακτηριστικά, όπως δαχτυλικό αποτύπωμα, ίριδα, retina, γεωμετρία παλάμης (ή αυτιού), DNA, οσμή, μορφή οδοντοστοιχίας, φωνή, υπογραφή και πρόσωπο με σκοπό να επαληθεύσει ή να ταυτοποιήσει κάποιον χρήστη. Τα χαρακτηριστικά αυτά αναφέρονται στην αγγλική βιβλιογραφία ως *traits, indicators, identifiers* ή *modalities*.

Η διαδικασία αναγνώρισης που ακολουθεί κάθε τέτοιο σύστημα αποτελείται από δύο μέρη: εγγραφή (*enrollment*) και αναγνώριση (*recognition)*. Κατά τη διάρκεια της εγγραφής, δεδομένα (*raw data)* συλλέγονται από τον χρήστη και μέσω κάποιας διαδικασίας εξάγονται χαρακτηριστικά (*feature extraction*). Τα αρχικά δεδομένα απορρίπτονται, καθώς η πληροφορία έχει ήδη εξαχθεί σε πιο χρήσιμη και αξιοποιήσιμη μορφή και τέλος, αποθηκεύονται.

- Κατά τη διαδικασία της ταυτοποίησης, η προηγούμενη διαδικασία επαναλαμβάνεται και τα χαρακτηριστικά που εξήχθησαν συγκρίνονται με όλα τα δεδομένα της βάσης. Τέλος, το σύστημα βρίσκει σε ποιον χρήστη αντιστοιχεί.
- Κατά τη διαδικασία της επαλήθευσης ο χρήστης δηλώνει ποιος είναι, και τα χαρακτηριστικά που εξάγονται εκείνη τη στιγμή, συγκρίνονται με τα αποθηκευμένα του χρήστη που ισχυρίζεται πως είναι. Το σύστημα επαληθεύει την ταυτότητά του ή τον απορρίπτει.

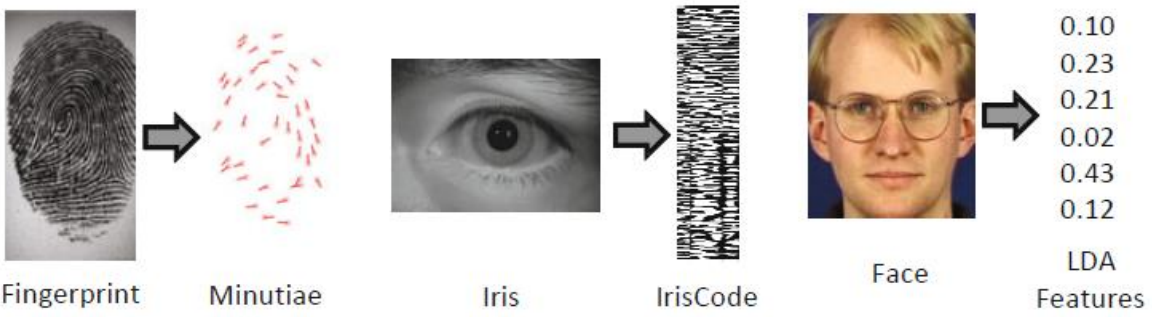
Το πλήθος των δεδομένων εισαγωγής, η πολυπλοκότητα εξαγωγής χαρακτηριστικών και η δυσκολία απόφασης από το σύστημα για το αν πρέπει να δοθεί πρόσβαση ή όχι στον χρήστη, εξαρτάται από τον σκοπό για τον οποίο απαιτείται η ταυτοποίηση. Παραδείγματος χάριν, για την έκδοση διαβατηρίου τα βιομετρικά που συνήθως χρησιμοποιούνται είναι μια απλή φωτογραφία προσώπου (προσοχή, όχι βιομετρική αναγνώριση προσώπου), ύψος, δαχτυλικό αποτύπωμα, υπογραφή και χρώμα ματιών,· ενώ για την πρόσβαση σε χώρο πυρηνικών όπλων ίσως απαιτούνται πολλά περισσότερα, όπως βιομετρική ανάλυση προσώπου, γεωμετρία χεριού-αυτιού, ανάλυση retina και αναγνώριση φωνής. Επίσης, στη δεύτερη περίπτωση, το σύστημα θα εξετάσει λεπτομερέστερα τα δείγματα του χρήστη με εκείνα που έχει αποθηκευμένα.

Συνεπώς, ένα βιομετρικό σύστημα, αποτελείται από τέσσερα βασικά μέρη (*building blocks*): τον αισθητήρα *(sensor)*, την εξαγωγή των χαρακτηριστικών *(feature extractor)*, τη βάση δεδομένων *(database)*, και τον συγκριτή *(matcher)*.

Θα προσπαθήσουμε να περιγράψουμε αναλυτικότερα τα βασικά μέρη ενός βιομετρικού συστήματος:

Sensor: Πρόκειται για το βασικότερο κομμάτι του συστήματος εφόσον είναι υπεύθυνο για τη συλλογή δεδομένων. Τα δεδομένα συνήθως είναι μια εικόνα διδιάστατης μορφής, εκτός αν συλλέγει φωνή (μονοδιάστατο σήμα) ή κάποιο πολυπλοκότερο χαρακτηριστικό όπως ηλεκτρονική υπογραφή (πολυδιάστατο σήμα). Ένας αισθητήρας θα πρέπει να συλλέγει δεδομένα υψηλής ανάλυσης και ευκρίνειας.

Feature extraction: Συνήθως (σχεδόν πάντα), τα δεδομένα που συλλέχθηκαν από τον αισθητήρα δεν μπορούν να χρησιμοποιηθούν από το σύστημα για ταυτοποίηση, καθώς περιέχουν πολλή άχρηστη πληροφορία (ή κοινή πληροφορία με τα δείγματα των υπόλοιπων χρηστών) οπότε το σύστημα εξάγει κάποια χαρακτηριστικά τα οποία θεωρεί απαραίτητα και απορρίπτει την αρχική εικόνα εισαγωγής. Η μορφή των χαρακτηριστικών εξαρτάται από το είδος των βιομετρικών. Στην εικόνα διακρίνονται τρεις τύποι εξαγμένων χαρακτηριστικών για διαφορετικά βιομετρικά. Η διαδικασία αυτή αποτελεί το κύριο θέμα της εργασίας αυτής.

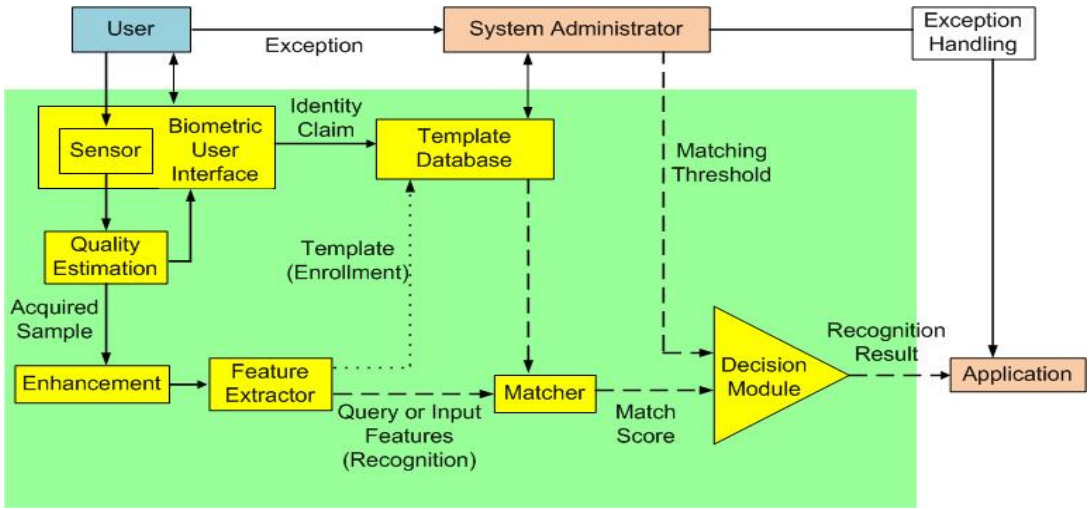


Εικόνα 1.1: Δεδομένα εισαγωγής από τον αισθητήρα και δεδομένα που εξήχθησαν από αυτά. Η μορφή του εξαγόμενου χαρακτηριστικού εξαρτάται από τη φύση του βιομετρικού εισαγωγής.

Database: Όλα τα δεδομένα που προέρχονται από το προηγούμενο block αποθηκεύονται στη βάση δεδομένων η οποία αποθηκεύει παράλληλα το όνομα των χρηστών όπως και κάθε άλλου είδους χρήσιμη πληροφορία (διεύθυνση, τηλέφωνο, φύλο) και είναι άμεσα προσβάσιμη από το σύστημα. Η βάση δεδομένων μπορεί να είναι είτε συγκεντρωτική, είτε αποκεντρωτική και η επιλογή του είδους της αποτελεί σημαντική απόφαση. Μια συγκεντρωτική βάση δεδομένων είναι αποθηκευμένη τοπικά στο σύστημα και πιθανές επιπτώσεις αυτού είναι η απαίτηση μεγάλου αποθηκευτικού χώρου, ενώ στη δεύτερη περίπτωση, μια αποκεντρωτική μπορεί να βρίσκεται οπουδήποτε (π.χ. σε κάποιον server). Η δεύτερη απαιτεί συνεχή επικοινωνία με το σύστημα και βασικό μειονέκτημά της είναι η πιθανή παραβίασή της από χάκερς.

Matching Module: Αποφασίζει αν το δείγμα προς εξέταση μπορεί να επαληθεύσει την ταυτότητα του χρήστη. Ένας συγκριτής που εξετάζει ομοιότητες, έχει ως έξοδο ένα σκορ

αντιστοίχισης του αμφισβητούμενου δείγματος (questioned sample) με τα αποθηκευμένα δείγματα. Πολλές φορές, η τελική απόφαση για την ταυτοποίηση λαμβάνεται από το σύστημα (αν είναι εφοδιασμένο με decision making modules), διαφορετικά, η ταυτοποίηση οφείλεται στην κρίση φυσικού προσώπου το οποίο εξετάζει τα matching scores.



Εικόνα 1.2: Βασικά στοιχεία ενός βιομετρικού συστήματος

1.2 Βιομετρικά χαρακτηριστικά

Τέλος, θα κάνουμε μια συνοπτική περιγραφή των πιο κοινών βιομετρικών χαρακτηριστικών που συλλέγονται.

Δαχτυλικό αποτύπωμα (fingerprint): Το πιο διαδεδομένο βιομετρικό χαρακτηριστικό που χρησιμοποιείται για ταυτοποίηση εκατοντάδες χρόνια, κυρίως από την αστυνομία και κυβερνητικές υπηρεσίες. Είναι εξαιρετικά σπάνιο να βρεθούν άτομα με το ίδιο αποτύπωμα ακόμη και αν πρόκειται για συγγενικά πρόσωπα. Βρίσκεται στις άκρες των δακτύλων, αποτελείται από κορυφογραμμές και κοιλάδες και διαμορφώνεται τους πρώτους μήνες ανάπτυξης. Παλαιότερα η συλλογή του γινόταν με εμβάπτιση του δαχτύλου σε μελάνι και στη συνέχεια η αποτύπωση γινόταν σε χαρτί. Η σύγκριση δύο δαχτυλικών αποτυπωμάτων γινόταν με οπτική σύγκριση. Τις τελευταίες δεκαετίες χρησιμοποιείται ψηφιακός αισθητήρας, συλλέγονται δείγματα περισσοτέρων του ενός δαχτύλου, και η σύγκριση γίνεται βάση των μικρολεπτομερειών τους. Το δαχτυλικό αποτύπωμα μερικές φορές δεν μπορεί να συλλεχθεί αν το δάχτυλο έχει αιχμές, κοψίματα, τραυματισμούς.

Αποτύπωμα παλάμης (palmprint): Στην ουσία πρόκειται για επέκταση του δακτυλικού αποτυπώματος εφόσον έχουν την ίδια μορφή. Η διαφορά είναι ότι καλύπτει μεγαλύτερη περιοχή, συνεπώς απαιτούνται μεγαλύτεροι αισθητήρες (μεγαλύτερο κόστος) και λόγω του πλήθους των δεδομένων σε σύγκριση με το δαχτυλικό αποτύπωμα, μεγαλύτερη υπολογιστική ισχύ.

Ίρις (iris): Η ίριδα του ματιού αποτελεί το χρωματικό μέρος του που περιβάλλεται εξωτερικά από τον σκληρό χιτώνα (ασπράδι) και εσωτερικά από την κόρη. Σταθεροποιείται χρωματικά περίπου στα δύο πρώτα χρόνια της ανάπτυξης. Η πολύπλοκη συνοχή της εμπεριέχει αρκετή πληροφορία και οφείλεται σε γενετικούς παράγοντες.

Πρόσωπο (face): Χρησιμοποιείται ενστικτωδώς από τους ανθρώπους για να αναγνωρίσουν ο ένας τον άλλον. Τα περισσότερα έγγραφα που υποδηλώνουν την ταυτότητα κάποιου, συμπεριλαμβάνουν και κάποια φωτογραφία του, της οποίας η λήψη εξαρτάται από περιορισμούς (όπως φωτισμός, γωνία οπτικής λήψης, χρωματισμό). Οι διαφορετικές γωνίες λήψης του ίδιου προσώπου, πολλές φορές, καθιστά δύσκολη την ταυτοποίηση από αυτόματα συστήματα. Τα τελευταία χρόνια χρησιμοποιούνται τεχνικές που κάνουν πιο περίπλοκη τη συλλογή της πολύτιμης πληροφορίας εξετάζοντας την γεωμετρία του προσώπου.

Γεωμετρία χεριού (hand geometry): Σύμφωνα με αυτή, εξετάζεται το χέρι από τον καρπό μέχρι τα άκρα και οι πληροφορίες εξάγονται βάσει του πάχους των δακτύλων, τη μεταξύ τους απόσταση και το σχήμα της παλάμης. Εφαρμόζεται σε περιπτώσεις που οι χρήστες του συστήματος είναι σχετικά λίγοι, τόσο από άποψης όγκου δεδομένων, όσο και πιθανότητας να βρεθούν δείγματα τα οποία μοιάζουν αρκετά μεταξύ τους.

Στάση του σώματος (gait): Η αναγνώριση πόζας μελετά το περπάτημα και τις κινήσεις του ατόμου με σκοπό την αναγνώρισή του. Άλλη μια μέθοδος μαζί με αυτή του προσώπου που χρησιμοποιείται και από τους ανθρώπους ενστικτωδώς. Ο αισθητήρας συλλογής του δείγματος απαιτεί το πέρας κάποιου χρόνου, προφανώς τόσο στη φάση εγγραφής (enrollment), όσο και στη φάση ταυτοποίησης. Υπάρχουν αρκετοί παράγοντες που επηρεάζουν τη στάση του σώματος όπως, υποδήματα, ρουχισμός, έδαφος και φυσική κατάσταση ποδιών (τραυματισμοί, ενοχλήσεις).

Βιομετρία αυτιού (ear biometrics): Βασίζεται στη γεωμετρία και το σχήμα του αυτιού. Μπορεί να προσεγγιστεί είτε μελετώντας τις αποστάσεις σημείων πάνω στο αυτί, είτε μελετώντας γενικότερα το σχήμα του. Μερικές φορές συνδυάζεται με την αναγνώριση προσώπου σε φωτογραφίες προφίλ.

Αναγνώριση φωνής (voice recognition): Η φωνή καθορίζεται τόσο από γενετικούς παράγοντες (φωνητικές χορδές, κοιλότητα του στόματος, οδοντοστοιχία) όσο και από επίκτητους (άρθρωση, τρόπος ομιλίας). Άλλη μια μέθοδος που είναι αποτελεσματικότερη σε μικρές ομάδες χρηστών. Συνήθως το σύστημα ζητά από τον χρήστη να επαναλάβει δυνατά κάποια φράση αρκετές φορές με σκοπό την αποφυγή απάτης. Ένα μειονέκτημά της είναι ότι το μικρόφωνο επηρεάζεται αρκετά από θόρυβο υποβάθρου.

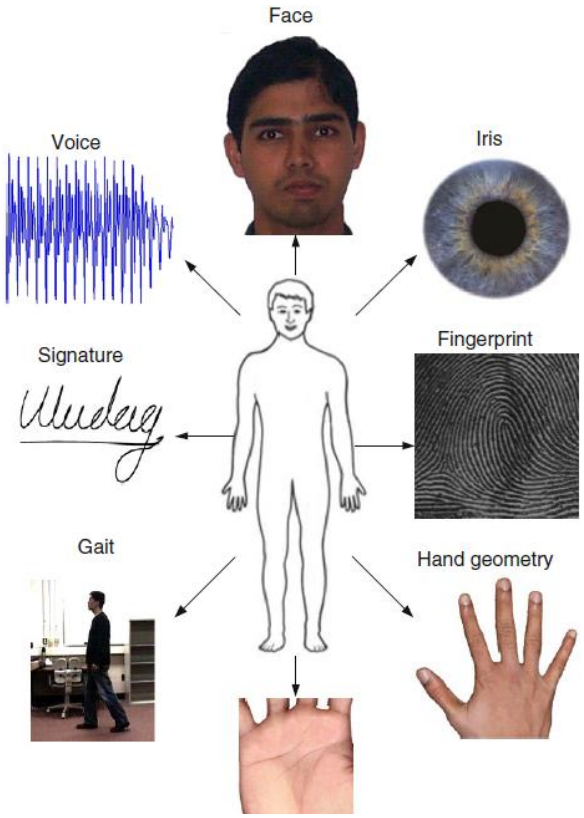
Πληκτρολόγηση (keystroke): Πρόκειται για χαρακτηριστικό συμπεριφοράς και όχι γενετικό καθώς και η συλλογή του μπορεί να γίνει χωρίς τη φυσική παρουσία του χρήστη. Είναι σαφές ότι η πληκτρολόγηση δεν έχει μοναδικότητα και αυτό την καθιστά

περισσότερο ως μέσο επαλήθευσης παρά ως μέσο ταυτοποίησης. Χρησιμοποιείται περισσότερο κατά τη διάρκεια μιας συνεδρίας, ώστε το σύστημα να επαληθεύει κάθε χρονική στιγμή ότι ο χρήστης δεν άλλαξε. Εφαρμόζεται σε υπηρεσίες που απαιτούν μικρότερο επίπεδο ασφάλειας, είτε συνδυαστικά με άλλα χαρακτηριστικά.

DNA: Το DNA είναι γενετικό χαρακτηριστικό το οποίο είναι μοναδικό (εκτός από την περίπτωση μονοζυγωτικών διδύμων) και μπορεί να προσδώσει ταυτότητα σε κάποιον. Τρεις παράγοντες καθιστούν αμφιλεγόμενη τη χρήση του: (i) μπορεί να συλλεχθεί εύκολα από κάποιον χωρίς να το γνωρίζει και να γίνει κατάχρηση, (ii) η επεξεργασία του με τεχνολογία αιχμής απαιτεί ανάμιξη από ειδικούς και απαιτεί χρόνο, (iii) δίνει πρόσβαση σε ευαίσθητες πληροφορίες, όπως τη δεκτικότητα του ατόμου σε ασθένειες. Χρησιμοποιείται κυρίως σε εξιχνιάσεις εγκλημάτων και σε περιπτώσεις αναγνώρισης συγγενικού δεσμού.

Θερμογραφήματα προσώπου (facial infrared thermogram): Μέθοδος η οποία δεν απαιτεί επαφή, καθώς γίνεται με τη λήψη μιας φωτογραφίας προσώπου με υπέρυθρη κάμερα (αντί του ορατού) και μελετάται η κατανομή της θερμοκρασίας. Εκτός του προσώπου, θερμογράφηση μπορεί να γίνει και στο χέρι ή στις φλέβες του χεριού. Μειονέκτημα αποτελεί το γεγονός ότι πρέπει να γίνεται σε χώρο συγκεκριμένης θερμοκρασίας καθώς και το κόστος του εξοπλισμού.

Σάρωση αμφιβληστροειδούς (retinal scan): Η πληροφορία που βρίσκεται στη δομή των αγγείων του αμφιβληστροειδούς χιτώνα είναι αρκετά μοναδική ώστε να δώσει ταυτότητα



Εικόνα 1.3: Παραδείγματα βιομετρικών χαρακτηριστικών

σε κάποιον. Η απόκτησή της γίνεται ζητώντας από τον χρήστη να εστιάσει σε κάποιο σημείο του αισθητήρα ώστε να σαρωθεί η δομή των αγγείων του.[1]

Υπογραφή (signature): Η μελέτη της υπογραφής ως βιομετρικό χαρακτηριστικό και η χρήση της ως μέσο επαλήθευσης αλλά και ταυτοποίησης θα γίνει εκτενώς στο επόμενο κεφάλαιο.

Κεφάλαιο 2: Η αναγνώριση της υπογραφής

Η εγκυρότητα εγγράφων, τόσο πολιτικών όσο και δικαστικών, εξαρτάται από τον συγγραφέα και τη γνησιότητά τους. Υπήρξαν πολλές φορές που σημαντικά έγγραφα ή έργα παραμείναν ανυπόγραφα με τη γνησιότητά τους να αμφισβητείται. Αυτές οι περιστάσεις επέβαλλαν την παρουσία μαρτύρων κατά τη διάρκεια συγγραφής ή υπογραφής ώστε η γνησιότητα να γίνει ευρέως αποδεκτή. Η παρουσία μαρτύρων όπως είναι φυσικό δεν ήταν πάντα εύκολη υπόθεση. Οπότε, στην πορεία ανάπτυξης ενός αποδεκτού προτύπου που θα ήταν αρκετό ώστε να υποστηρίξει τη γνησιότητα ενός χειρόγραφου προέκυψαν επιμέρους θέματα. Είναι πρέπον οι δικαστές ή οι ένορκοι να ελέγχουν από μόνοι τους τη γνησιότητα των εγγράφων; Και αν ναι, ποια είναι τα κριτήρια;

Η απάντηση στην πρώτη ερώτηση ήταν καταφατική, υπό την προϋπόθεση της προσκόμισης κάποιων επιβεβαιωτικών στοιχείων. Η επιβεβαίωση αυτή αναζητήθηκε από κάποιον μάρτυρα αναγνώρισης. Κάποιου εξοικειωμένου με τα χειρόγραφα του φερόμενου συγγραφέα. Οι μάρτυρες αναγνώρισης ήταν αυτοί που θα έθεταν τα κριτήρια σε κάθε χειρόγραφο ώστε να επιβεβαιώσουν αν οι προσφερόμενες αποδείξεις ήταν αρκετές για να σφραγίσουν την εγκυρότητα του εγγράφου. Η έκταση της γνωριμίας κάποιου με τη γραφή ενός ατόμου, και πως αποκτήθηκε αυτή, τέθηκε υπό αμφισβήτηση.

Όμως, κάθε άτομο που έχει παρατηρήσει για χρόνια ή έστω και μία φορά τον γραφικό χαρακτήρα κάποιου, ήταν ευπρόσδεκτος να καταθέσει τη μαρτυρία του για την αυθεντικότητά του. Παρ' όλη την αδυναμία αυτής της τακτικής, στάθηκε από μόνη της να υπερισχύσει των απαραίτητων προδιαγραφών. Η ανάγκη ήταν μεγαλύτερη των προτύπων.

Στο πέρασμα των χρόνων, αυτές οι τακτικές γινόντουσαν όλο και λιγότερο ευνοϊκές, δίνοντας βήμα στην ανάπτυξη της επιστήμης που βρίσκεται πίσω από την εξέταση της γνησιότητας των εγγράφων.[2]

2.1 Κατηγορίες μελέτης υπογραφών

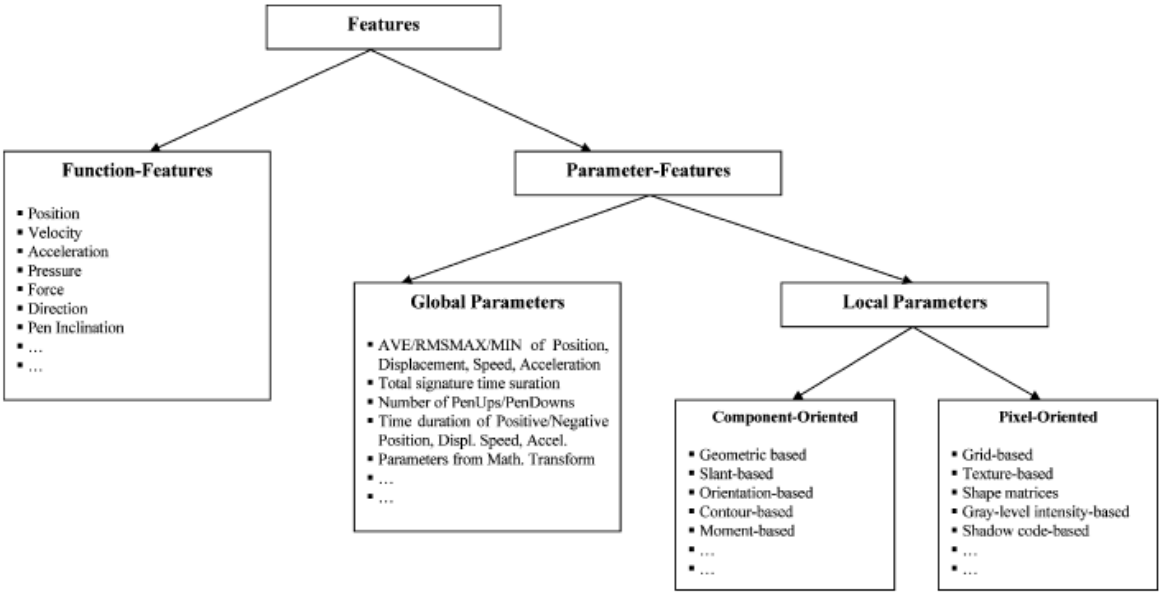
Η μελέτη της υπογραφής χωρίζεται σε δύο κατηγορίες, *off-line* και *on-line*. Η κύρια διαφορά της πρώτης από τη δεύτερη είναι ότι η πρώτη βασίζεται σε στατικές εικόνες δύο διαστάσεων, ενώ η δεύτερη στη μελέτη πολυδιάστατων σημάτων. Αντικείμενο μελέτης παρουσιάζουν και οι δύο, μη δείχνοντας ιδιαίτερη προτίμηση, και αυτό γιατί βρίσκουν διαφορετική εφαρμογή στον πραγματικό κόσμο.

Η μελέτη της στατικής υπογραφής είναι απαραίτητη όχι μόνο επειδή οι περισσότερες υπογραφές είναι στατικές (προς το παρόν), αλλά και για τη μελέτη του παρελθόντος. Όπως αναφέραμε προηγουμένως, πολλά χειρόγραφα ή έργα τέχνης φέρουν υπογραφές των οποίων οι συγγραφείς είναι υπό αμφισβήτηση. Συνεπώς, η ανάπτυξη συστημάτων που είναι σε θέση με λίγα δείγματα στην κατοχή τους, να μπορούν να επαληθεύσουν την κυριότητα κάποιου είναι πολύ σημαντική.

Αντιθέτως, η μελέτη της δυναμικής υπογραφής λόγω των πολλών της διαστάσεων και κατ' επέκταση χαρακτηριστικών, καθιστά ευκολότερη την έρευνα οδηγώντας σε αξιοσημείωτα αποτελέσματα. Επιπλέον, όσο η τεχνολογία προχωρά, όλο και περισσότερες υπηρεσίες και εφαρμογές στρέφονται στη χρήση δυναμικών συστημάτων. Τα πλεονεκτήματα ενός τέτοιου συστήματος είναι προφανή, εκτός από τη δυνατότητα μη φυσικής αποθήκευσης της υπογραφής δίνει τη δυνατότητα και της υπογραφής εξ' αποστάσεως.

Στατική (*off-line*) υπογραφή: Θεωρείται ως μια εικόνα (*grayscale*) δύο διαστάσεων και απεικονίζεται ως ένα διάνυσμα $S(x,y)_{0\leq x\leq X, 0\leq y\leq Y}$, όπου το $S(x,y)$ υποδηλώνει το επίπεδο του γκρι στη θέση (x,y). Κύριο γνώρισμά της καθαρά η γεωμετρία της.

Δυναμική (*online*) υπογραφή: Πραγματοποιείται πάνω σε ηλεκτρονικές συσκευές που δημιουργούν αναπαριστάμενα σήματα κατά τη διάρκεια της γραφής. Παριστάνονται ως μια ακολουθία $S(n)_{n=0,1,\dots,N}$ όπου η τιμή $S(n)$ είναι η τιμή του σήματος που δειγματοληπτήθηκε τη στιγμή $n\Delta T$. Είναι εμφανές ότι στα γνωρίσματά της εμπλέκεται και ο χρόνος. Στην εικόνα βλέπουμε την ποικιλία και την πολυπλοκότητα των χαρακτηριστικών που μπορούμε να εξάγουμε από μια δυναμική υπογραφή.[3]



Εικόνα 2.1: Τα κυριότερα χαρακτηριστικά και παράμετροι της δυναμικής υπογραφής

2.2 Η εξέταση της υπογραφής

Όπως προαναφέραμε στο πρώτο κεφάλαιο, οι όροι αναγνώρισης και επαλήθευσης είναι διαφορετικοί και ειδικά στο θέμα της υπογραφής γίνεται συχνά κατάχρηση της αναγνώρισης. Η επαλήθευση, προσπαθεί να επαληθεύσει την ταυτότητα κάποιου σε ένα κείμενο π.χ. αν η υπογραφή αυτή ανήκει σε κάποιον συγκεκριμένο. Αρχικά αυτό γινόταν με τη σύγκριση μίας προς μίας και πολλές φορές με αρκετές προσεγγίσεις. Κατά τη διάρκεια της ταυτοποίησης ο σκοπός είναι "σε ποιόν ανήκει η συγκεκριμένη υπογραφή".

Στη διαδικασία της επαλήθευσης η επεξεργασία της υπογραφής χωρίζεται σε τρεις φάσεις:

- γίνεται μία λεπτομερής εξέταση πάνω στα ιδιαίτερα χαρακτηριστικά της υπογραφής
- σύγκριση των χαρακτηριστικών της υπο-εξέταση υπογραφής με γνωστά δείγματα για την εύρεση διαφορών και ομοιοτήτων και,
- αξιολόγηση των χαρακτηριστικών, οι ομοιότητες και οι διαφορές οδηγούν σε ένα συμπέρασμα για την αυθεντικότητα.

Η τελευταία φάση είναι η πιο καθοριστική στην ισχύ της απόδειξης. Η μοναδικότητα των χαρακτηριστικών εξαρτάται από τη σπανιότητά της και τη σχέση της με τις κοινές υπογραφές. Ένας εξεταστής θα πρέπει να γνωρίζει τη διακύμανση που υπάρχει μεταξύ των γραφέων σε ένα σχετικό δείγμα πληθυσμού. Επίσης, θα πρέπει να έχει την εμπειρία να ξεχωρίσει εκείνα τα μοναδικά χαρακτηριστικά που την κάνουν περισσότερο ή λιγότερο μοναδική.

Συμπεραίνουμε λίγο πολύ, η τελική απόφαση για το αν είναι ή όχι το δείγμα αυθεντικό, εκφράζεται πιθανοτικά. Αν για παράδειγμα η ετυμηγορία επαληθεύει την αυθεντικότητα θα πρέπει μετά να εξετάσουμε και ποιο είναι το πιθανότερο σενάριο:

- το δείγμα είναι μια συνηθισμένη υπογραφή του συγγραφέα
- η ύπαρξη του δείγματος υποκινήθηκε από κάποιον άλλον χωρίς την επιλογή του συγγραφέα
- το δείγμα σχετίζεται με την υπογραφή άλλου συγγραφέα.

Τα σενάρια αντίστοιχα που προκύπτουν όταν ο εξεταστής αμφιβάλλει για τη γνησιότητα του δείγματος είναι:

- ο συγγραφέας προσπάθησε να παραποιήσει την υπογραφή του
- ο συγγραφέας χρησιμοποίησε διαφορετική υπογραφή
- η διαδικασία της υπογραφής διαταράχθηκε από εξωτερικούς παράγοντες
- η υπογραφή είναι μια κακή απομίμηση
- το δείγμα σύγκρισης δεν είναι αντιπροσωπευτικό του συγγραφέα.[4]

2.3 Ποιότητα αντιγράφων

Το μειονέκτημα των υπογραφών είναι ότι είναι ευάλωτες στην πλαστογραφία, περισσότερο μάλιστα απ' ότι τα χειρόγραφα κείμενα. Συχνά, χρήστες πλαστογραφούν υπογραφές με στόχο την παραποίηση της ταυτότητάς τους. Οι εξεταστές γνησιότητας λοιπόν, βασίζονται στην ποιότητα των δειγμάτων. Η ποιότητα της υπογραφής εξαρτάται από διάφορους φυσικούς παράγοντες, όπως την κίνηση του καρπού και του ώμου, τον συνδυασμό των δακτύλων κ.α. Η προσπάθεια αναπαραγωγής της υπογραφής απαιτεί οπτική αλληλεπίδραση και αυτό επιδρά με δύο διαφορετικούς τρόπους στο συνολικό αποτέλεσμα. Είτε ο παραχαράκτης θα δώσει έμφαση στο δείγμα προς πλαστογράφηση με αποτέλεσμα την επιβράδυνση της διαδικασίας οδηγώντας σε μια ασταθή ροή μελάνης, είτε θα προσπαθήσει να μιμηθεί πολύ φυσικά την κίνηση του αρχικού απέχοντας πολύ από το γνήσιο δείγμα. Παρατηρούμε όμως, ότι και τα γνήσια δείγματα μεταξύ τους έχουν μικρές αποκλίσεις και αυτό οφείλεται σε μια αλληλουχία γράψιμου και παύσεως, καθώς και στην πνευματική-ψυχολογική κατάσταση του γραφέα εκείνη τη στιγμή. Είναι συνετό να περιμένουμε ότι οι αντιγραφές διαφέρουν με τις γνήσιες από τρεις απόψεις: μορφή, ποιότητα γραμμής της μελάνης και συνέχεια μολυβιάς.[4]

2.4 Πολυπλοκότητα της υπογραφής

Θεωρούμε την πολυπλοκότητα της υπογραφής ως την ενδογενή διαταραχή της μορφής ενός δείγματος και της διακύμανσης του σε ένα σετ αυθεντικών δειγμάτων. Η υπογραφή αποτελεί μια χειρονομία που εκτελείται πάρα πολύ γρήγορα. Η ταχύτητα αυτή προσδίδει διαφορετική πολυπλοκότητα σε κάθε χρήστη. Για παράδειγμα, κάποιων ατόμων οι υπογραφές χαρακτηρίζονται από ευανάγνωστα καλλιγραφικά γράμματα, ενώ κάποιων άλλων χαρακτηρίζονται από συνεχόμενη γραφή χωρίς να σηκώνεται το μολύβι από το χαρτί. Για τον λόγο αυτό, η πολυπλοκότητα αποτελεί σημαντικός παράγοντας που χαρακτηρίζει τους γραφείς μέσω της υπογραφής τους. Από την άλλη, οι συμπεριφορικοί παράγοντες οδηγούν σε μεγαλύτερες αποκλίσεις μεταξύ διαφορετικών δειγμάτων. Ωστόσο, αυτές οι αποκλίσεις εξαρτώνται από τους γραφείς, καθώς υπάρχουν γραφείς των οποίων αποκλίνουν όλα τα δείγματα μεταξύ τους και γραφείς των οποίων τα σχετικά δείγματα είναι πανομοιότυπα.[5]

Έχουν προταθεί τρεις θεωρητικές σχέσεις, μόλις ληφθεί η απόφαση ότι το υπό αμφισβήτηση δείγμα είναι παρόμοιο ή συμβατό με το πρότυπο δείγμα, τη λεγόμενη "θεωρία πολυπλοκότητας". Αυτές οι σχέσεις είναι:

Ο αριθμός των αλυσιδωτών μολυβιών και η πολυπλοκότητα

Η πρώτη σχέση είναι ο αριθμών των μολυβιών ως δείκτης πολυπλοκότητας. Όσο περισσότερο η διεύθυνση του μολυβιού αλλάζει χωρίς να σηκώνεται από το χαρτί, τόσο οπτικά πιο πολύπλοκη είναι η υπογραφή.

Πολυπλοκότητα και πιθανή αντιστοίχιση σε δείγμα

Αυτή η σχέση εξάγεται από την πρώτη, δεδομένου ότι όλα τα δείγματα μοιράζονται κοινά στοιχεία, π.χ. συνεχόμενες μολυβιές, και ο αριθμός των οποίων συμβάλλουν στην πολυπλοκότητα. Η πολυπλοκότητα αυξάνει την πιθανότητα τα δείγματα να αποκλίνουν μεταξύ τους. Για πιο απλά δείγματα η πιθανότητα να βρεθεί συσχέτιση μεταξύ τους είναι μεγαλύτερη.

Πολυπλοκότητα και η ευκολία αντιγραφής

Όσο το δείγμα γίνεται περισσότερο πολύπλοκο, αναμένεται να είναι πιο δύσκολο να αναπαραχθεί. Είναι προφανές ότι μια υπογραφή που αποτελείται από απλές γραμμές είναι ευκολότερο να αναπαραχθεί από μία πιο καλλιγραφική. Επομένως, ένα δείγμα πολύπλοκου χωροχρονικού σχεδίου (όπου η έννοια του χρόνου αφορά στο χρόνο που χρειάζεται η ολοκλήρωση της υπογραφής) θεωρείται ότι είναι πιο δύσκολο να αναπαραχθεί γιατί οι αποκλίσεις του από τα υπόλοιπα δείγματα είναι πιο εύκολο να εντοπιστούν από τους εξεταστές.

Η θεωρία της πολυπλοκότητας (που προτάθηκε από τους *Found and Rogers*), προτείνει ότι όσο πιο πολύπλοκη η υπογραφή, τόσο λιγότερο πιθανό να γίνει τέλεια η πλαστογράφηση χωρίς να βρεθούν ίχνη της. Επιπροσθέτως, όταν βρίσκονται ομοιότητες μεταξύ πολύπλοκων δειγμάτων, τόσο πιθανότερο είναι να αποτελεί το εξεταζόμενο δείγμα γνήσιο.[4]

Ως παράγοντες πολυπλοκότητας μπορούμε να αναφέρουμε:

- ο αριθμός των σημείων καμπής
- μήκος γραμμής στην οποία υπάρχουν τα σημεία καμπής
- αριθμός σημείων τομής της γραμμής (καθώς και επικαλυπτόμενων γραμμών)
- αριθμός φορών που το μολύβι σηκώθηκε από το χαρτί
- το πάχος της γραμμής σαν δείκτη της πίεσης που ασκήθηκε
- η έλλειψη μοναδικότητας χαρακτήρων (π.χ. διπλότυποι χαρακτήρες)
- χωρική μορφή
- απόκλιση δειγμάτων ίδιου γραφέα.[4]

Κεφάλαιο 3: Μηχανική Μάθηση

Η μηχανική μάθηση είναι κλάδος της επιστήμης των υπολογιστών σχετιζόμενος με την τεχνητή νοημοσύνη και γνώρισε ιδιαίτερη άνθιση από το 1990 και μετά. Το 1959 ορίζεται από τον Samuel Arthur ως *«πεδίο μελέτης που δίνει τη δυνατότητα στους υπολογιστές την ικανότητα να μαθαίνουν, χωρίς να έχουν προγραμματιστεί ρητά»*. Αποτέλεσμά της είναι η αναγνώριση μοτίβων στα δεδομένα και η λήψη αποφάσεων μέσω αλγορίθμων οι οποίοι έχουν εκπαιδευτεί κατάλληλα τροφοδοτούμενοι από μια σειρά κατηγοριοποιημένων δεδομένων. Παραδείγματα εφαρμογών αποτελούν: τα φίλτρα ηλεκτρονικού ταχυδρομείου *(spam filters)*, η οπτική αναγνώριση χαρακτήρων *(OCR)*, η αναγνώριση σημάτων οδικής κυκλοφορίας από αυτοκίνητα κ.α.

Ανάλογα με τον τύπο του προβλήματος οι εργασίες εκμάθησης χωρίζονται σε 3 κατηγορίες:

- Επιτηρούμενη μάθηση (Supervised learning): Τα δεδομένα που χρησιμοποιούνται κατά την εκπαίδευση είναι προσημασμένα ως προς τον τύπο τους *(labeled data)*. Στόχος του αλγορίθμου είναι να προσαρμόσει κάποιον γενικό κανόνα διαχωρισμού, ώστε να μπορεί να κατηγοριοποιήσει δεδομένα που δέχεται στην είσοδο.
- Μη επιτηρούμενη μάθηση (Unsupervised learning): Τα δεδομένα εισαγωγής δεν φέρουν ταμπέλες ως προς την κατηγορία τους *(unlabeled data)* με αποτέλεσμα να δίνεται στον αλγόριθμο η δυνατότητα να ανακαλύψει μόνος του κάποιο κρυφό μοτίβο διαχωρισμού των δεδομένων, ή να εξάγει κάποια χαρακτηριστικά.
- Ημι-επιτηρούμενη μάθηση (Semi-supervised learning): Δίνονται στον αλγόριθμο μικρό πλήθος προσημασμένων δεδομένων καθώς και μεγάλο πλήθος αγνώστων δειγμάτων αφήνοντας τον αλγόριθμο να βρει μόνος του κάποιο μοτίβο διαχωρισμού που σχετίζεται με τα ήδη κατηγοριοποιημένα δείγματα που του έχουν δοθεί.

Υπάρχουν διάφορες προσεγγίσεις με σκοπό την εκμάθηση των μηχανών. Μερικές από αυτές είναι: τα δέντρα λήψης αποφάσεων, οι κανόνες συσχέτισης, η βαθιά μάθηση *(Deep Learning)*, οι μηχανές διανυσμάτων υποστήριξης *(SVM)*, τα δίκτυα Bayes, η ομαδοποίηση *(clustering)*, γενετικοί αλγόριθμοι και τεχνητά νευρωνικά δίκτυα *(Neural Networks, ή για συντομία NN)*. Η δική μας προσέγγιση θα γίνει μέσω των τελευταίων.[6]

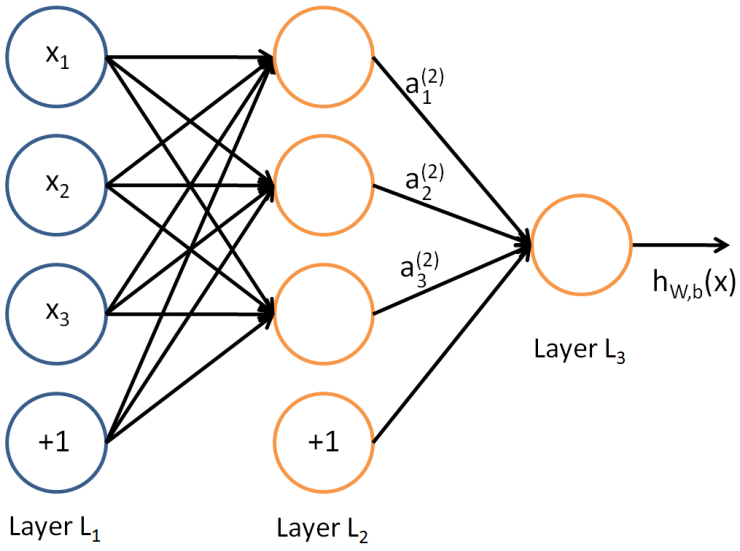
3.1 Τεχνητά νευρωνικά δίκτυα

Το όνομα τους είναι εμπνευσμένο από τους βιολογικούς νευρώνες τους οποίους και προσπαθούν να μιμηθούν. Πρόκειται στην ουσία για ένα σύνολο μονάδων, που ονομάζονται νευρώνες, τοπικά συνδεδεμένων μεταξύ τους, συγκροτώντας ένα δίκτυο.

Κάθε νευρώνας δέχεται ένα σήμα στην είσοδό του και μέσω μιας συνάρτησης ενεργοποίησης *(activation function)* που περιέχεται σε αυτόν παράγεται μια έξοδος.

Το μέγεθος ενός νευρωνικού δικτύου καθορίζεται από τα επίπεδά του. Ένα επίπεδο αποτελείται από ένα σύνολο νευρώνων κοινών χαρακτηριστικών οι οποίοι δέχονται το ίδιο σήμα στην είσοδό τους και ταυτόχρονα τροφοδοτούν τους επόμενους. Ανάλογα με το πλήθος των επιπέδων, η αρχιτεκτονική του δικτύου λέγεται ρηχή *(shallow neural network)* ή βαθιά *(deep neural network)*. Τα ενδιάμεσα επίπεδα ονομάζονται κρυφά επίπεδα *(hidden layers)*.

Κάθε νευρώνας μπορεί να έχει συνάψεις με πολλούς νευρώνες ταυτόχρονα είτε αφορά την είσοδό του, είτε την έξοδό του. Το σήμα στην είσοδο κάθε νευρώνα μεταφέρεται μέσω των συνάψεων αυτών, και ο σκοπός τους είναι να το ενισχύσουν ή να το αποσβέσουν. Κατά συνέπεια, βλέπουμε ότι κάθε σύνδεση έχει διαφορετικό συντελεστή βάρους και **συμπερασματικά ο σκοπός κάθε νευρωνικού δικτύου είναι ο προσδιορισμός αυτών των βαρών που θα οδηγεί στις προβλεπόμενες τιμές.**



Εικόνα 3.1: Παράδειγμα νευρωνικού δικτύου τριών επιπέδων

3.2 Συναρτήσεις ενεργοποίησης

Στην πιο απλή μορφή του, ένας νευρώνας δέχεται στην είσοδό του μια τιμή x και μέσω μιας συνάρτησης ενεργοποίησης $f(\cdot)$ παράγει μια έξοδο $f(x)$. Οι συναρτήσεις ενεργοποίησης στοχεύουν σε μια έξοδο ανάμεσα στο $[0,1]$ ή $[-1,1]$ ανάλογα με το πρόβλημα. Οι πιο συνηθισμένες συναρτήσεις ενεργοποίησης είναι:

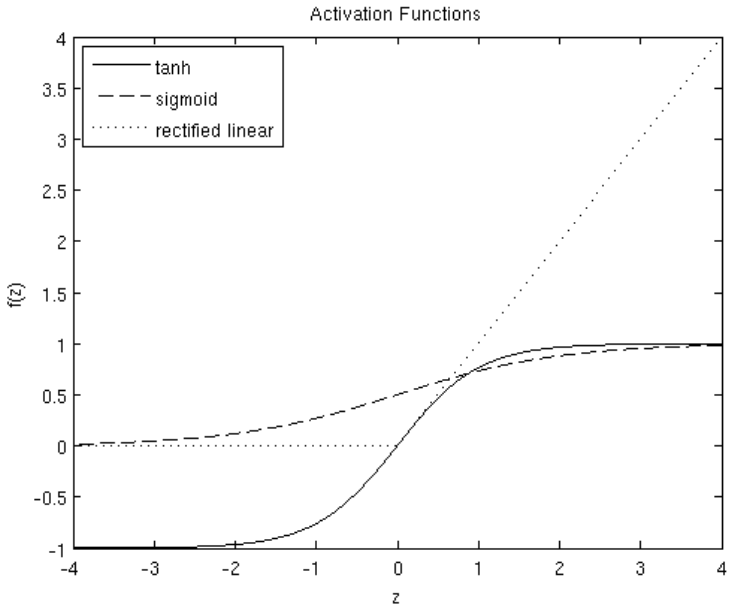
$$f(x) = \frac{1}{1+e^{-x}} \qquad \textit{σιγμοειδής} \tag{3.1}$$

$$f(x) = \frac{e^x + e^{-x}}{e^x + e^{-x}} \ \ \upsilon\pi\varepsilon\rho\beta o\lambda\iota\kappa\acute{\eta}\ \varepsilon\phi\alpha\pi\tau o\mu\acute{\varepsilon}\nu\eta \tag{3.2}$$

Καθώς και η *rectified linear unit (ReLU):*

$$f(x) = max(0, x) \tag{3.3}$$

Η τελευταία χρησιμοποιείται κυρίως στα βαθιά νευρωνικά δίκτυα και η διαφορά της με τις υπόλοιπες είναι ότι δεν είναι διαφορίσιμη.



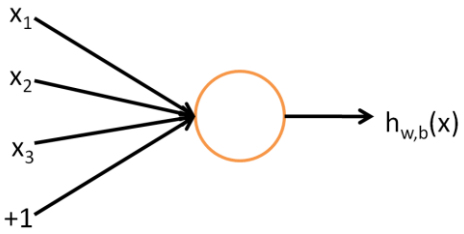
(3.4)

Εικόνα 3.2: Οι τρεις συναρτήσεις ενεργοποίησης

Ας υποθέσουμε όμως τώρα ότι ο νευρώνας βρίσκεται στην πιο γενική σύνδεσή του και δέχεται N τιμές στην είσοδό του: $x_1, x_2, \dots, x_N$. Κάθε μία από αυτές τις τιμές x_i φτάνει μέσω μιας σύναψης με βάρος W_i, με αποτέλεσμα $W_1, W_2, \dots, W_N$ τα βάρη του νευρώνα. Η έξοδος του νευρώνα είναι:

$$h_{W,b}(x) = f(W^T x) = f\left(\sum_{i=1}^{N} W_i x_i + b\right) \ \ ,\acute{o}\pi o\upsilon\ f: \mathbb{R} \to \mathbb{R} \tag{3.5}$$

Η $h_{W,b}(x)$ καλείται *hypothesis function* και ο όρος b *bias term*.



Εικόνα 3.3: Απλός νευρώνας 3 εισόδων

Bias term: Πρόκειται για έναν σταθερό όρο, του οποίου η χρήση είναι πολύ σημαντική καθώς όταν εισέρχεται στο $W^T x$ επιτρέπει στη συνάρτηση ενεργοποίησης να μετακινηθεί προς τα αριστερά ή τα δεξιά δίνοντάς της την ευκαιρία να προσεγγίσει ακόμη καλύτερα την ιδανική μορφή.

Με μία άλλη ματιά, μπορεί κάποιος να πει ότι το άθροισμα $\sum_{i=1}^{N} W_i x_i + b$, εξυπηρετεί μια συνάρτηση $y = ax + b$ οπότε χωρίς την εισαγωγή του *bias* θα περνάει πάντα από την αρχή (0,0) δίνοντας ίσως φτωχά αποτελέσματα.

3.3 Μοντέλο πλήρους συνδεδεμένου νευρωνικού δικτύου

Στην *εικόνα 3.1* παρατηρούμε ένα απλό νευρωνικό δίκτυο τριών επιπέδων όπου το πρώτο επίπεδο αντιστοιχεί στην είσοδο των δεδομένων στο δίκτυο, το δεύτερο αποτελεί το ενδιάμεσο επίπεδο, ενώ το τρίτο αποτελείται από έναν μόνο νευρώνα και η έξοδός του αντιπροσωπεύει την έξοδο όλου του δικτύου.

Συμβολίζουμε ως n τον αριθμό των επιπέδων του δικτύου, άρα $n = 3$. Ονομάζουμε τα επίπεδα ως L_l, οπότε L_1 το επίπεδο εισόδου και L_3 η έξοδος του δικτύου.

Οι παράμετροι του δικτύου είναι οι $(W, b) = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)})$ όπου $W_{ij}^{(l)}$ το βάρος της σύναψης μεταξύ του νευρώνα j στο επίπεδο l και του νευρώνα i στο επίπεδο $l+1$. Στο παράδειγμα, $W^{(1)} \in \mathbb{R}^{3\times3}$ γιατί είναι ένας πίνακας 3×3 όπου κάθε σειρά αντιστοιχεί στα βάρη καθενός από τους τρεις νευρώνες και κάθε στήλη στις τρεις συνάψεις του κάθε νευρώνα με το προηγούμενο επίπεδο, ενώ $W^{(2)} \in \mathbb{R}^{1\times3}$ επειδή έχουμε $j = 3$ νευρώνες που συνδέονται με $i = 1$ νευρώνα από το επόμενο επίπεδο. (Σημειώστε ότι τα *bias terms* δεν έχουν συμπεριληφθεί καθώς η έξοδός τους είναι 1).

Η ενεργοποίηση *(activation)*, ή αλλιώς έξοδος κάθε νευρώνα i από το επίπεδο l συμβολίζονται ως a_i^l. Για ήδη προσδιορισμένες παραμέτρους οι έξοδοι υπολογίζονται βάσει της προηγούμενης σχέσης ως:

$$a_1^{(2)} = f\left(W_{11}^{(1)} x_1 + W_{12}^{(1)} x_2 + W_{13}^{(1)} x_3 + b_1^{(1)}\right) \tag{3.6}$$

$$a_2^{(2)} = f\left(W_{21}^{(1)} x_1 + W_{22}^{(1)} x_2 + W_{23}^{(1)} x_3 + b_2^{(1)}\right) \tag{3.7}$$

$$a_3^{(2)} = f\left(W_{31}^{(1)} x_1 + W_{32}^{(1)} x_2 + W_{33}^{(1)} x_3 + b_3^{(1)}\right) \tag{3.8}$$

με αποτέλεσμα η *hypothesis function* να παράγει:

$$h_{W,b}(x) = a_1^{(3)} = f\left(W_{11}^{(2)} a_1^{(2)} + W_{12}^{(2)} a_2^{(2)} + W_{13}^{(2)} a_3^{(2)} + b_1^{(2)}\right) \tag{3.9}$$

Στη συνέχεια, συμπεριλαμβάνοντας τους όρους *bias* συμβολίζουμε ως $z_i^{(l)}$ το σταθμισμένο άθροισμα των εισόδων στον νευρώνα i στο επίπεδο l (π.χ. $z_i^{(2)} = \sum_{j=1}^{n} W_{ij}^{(1)} x_j + b_i^{(1)}$) έτσι ώστε $f\left(z_i^{(l)}\right) = a_i^{(l)}$ και εφαρμόζοντας τη συνάρτηση ενεργοποίησης σε μορφή διανυσμάτων ώστε $f([z_1, z_2, z_3]) = [f\left(z_1\right), f(z_2), f(z_3)]$, καταλήγουμε σε μια πιο συμπαγή μορφή των παραπάνω:

$$\mathrm{z}^{(2)} = \mathrm{W}^{(1)}\mathrm{x} + \mathrm{b}^{(1)} \tag{3.10}$$
$$\mathrm{a}^{(2)} = \mathrm{f}\left(\mathrm{z}^{(2)}\right) \tag{3.11}$$
$$\mathrm{z}^{(3)} = \mathrm{W}^{(2)}\mathrm{a}^{(2)} + \mathrm{b}^{(2)} \tag{3.12}$$
$$\mathrm{h}_{\mathrm{W,b}}(\mathrm{x}) = \mathrm{a}^{(3)} = \mathrm{f}(\mathrm{z}^{(3)}) \tag{3.13}$$

Αυτό ονομάζεται διάδοση προς τα εμπρός *(forward propagation)*. Οργανώνοντας τις παραμέτρους μας σε πίνακες, εκμεταλλευόμενοι τις ιδιότητες της γραμμικής άλγεβρας, επιταχύνουμε τους υπολογισμούς. Η αρχιτεκτονική που χρησιμοποιήσαμε στο προηγούμενο παράδειγμα ονομάζεται πλήρως συνδεδεμένη *(fully connected)* καθώς προϋποθέτει τη σύνδεση κάθε νευρώνα με όλους του προηγούμενου και του επόμενου επιπέδου. Αυτός ο τρόπος σύνδεσης των νευρώνων δεν είναι ο μοναδικός αλλά ο πιο διαδεδομένος, επίσης ο αριθμός των νευρώνων στην έξοδο εξαρτάται από το εκάστοτε πρόβλημα.

3.4 Μοντέλα επιτηρούμενης μάθησης

Τα πιο διαδεδομένα προβλήματα που μοντελοποιούνται από νευρωνικά δίκτυα είναι προβλήματα *γραμμικής, λογιστικής και softmax παλινδρόμησης*. Η βασική ιδέα είναι η ίδια και οι διαφορές εντοπίζονται στη συνάρτηση κόστους και στον αριθμό των νευρώνων στο τελευταίο επίπεδο.

Γραμμική παλινδρόμηση

Στόχος σε ένα τέτοιο πρόβλημα είναι η πρόβλεψη μιας τιμής στην έξοδο για ένα διάνυσμα δεδομένων $x \in \mathbb{R}^n$ που εισάγονται στο δίκτυο. Κάθε x_i που αντιστοιχεί σε ένα δείγμα καλείται χαρακτηριστικό ή αλλιώς *feature* και το πλήθος αυτών ορίζει τις διαστάσεις του. Είναι απαραίτητο κάθε δείγμα να αποτελείται από τον ίδιο αριθμό χαρακτηριστικών.

Η τιμή της εξόδου του δικτύου είναι αποτέλεσμα μιας συνάρτησης $y = h_\theta(x)$ (όπου θ οι εκάστοτε παράμετροι) οπότε στην πράξη έχουμε $y^{(i)} \approx h_\theta(x^{(i)})$ για κάθε δείγμα. Ορίζουμε σαν συνάρτηση κόστους μια συνάρτηση της οποίας η τιμή αποτελεί μέτρο για το πόσο απέχει η $h_\theta(\cdot)$ από τη ζητούμενη:

$$J(\theta) = \frac{1}{2}\sum_i \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2 \tag{3.14}$$

η οποία είναι συνάρτηση των παραμέτρων θ αφού όπως προαναφέρθηκε η $h_\theta(\cdot)$ είναι συνήθως συνάρτηση βαρών και *bias*. Προσπαθούμε να βρούμε τις κατάλληλες παραμέτρους που ελαχιστοποιούν τη συνάρτηση κόστους.

Λογιστική παλινδρόμηση

Σε πολλές εφαρμογές η τιμή y των δειγμάτων δεν είναι απαραίτητα μια συνεχής τιμή, αλλά διακριτή. Υπάρχουν δηλαδή προβλήματα που απαιτούν από το δίκτυο να ταξινομήσει τα δεδομένα ανάμεσα σε δύο *κλάσεις* οι οποίες μπορούν να αναπαριστούν δύο οποιεσδήποτε κατηγορίες (π.χ. *"ναι/όχι", "άσπρο/μαύρο", "σκύλος/γάτα" κ.λπ.*).

Στα προβλήματα ταξινόμησης η ετικέτα παίρνει δυαδικές τιμές $\left(y^{(i)} \in \{0,1\}\right)$. Επομένως, η έννοια της υποθετικής συνάρτησης αποκτά άλλο νόημα, συγκεκριμένα μας δίνει την πιθανότητα το δείγμα να ανήκει στην κλάση "1" έναντι της πιθανότητας να ανήκει στην τάξη "0". Ειδικότερα:

όπου σ η σιγμοειδής συνάρτηση της οποίας το σύνολο τιμών είναι $[0,1]$. Ο στόχος είναι

$$P(y = 1|x) = h_\theta(x) = \sigma(W^T x + b) \tag{3.15}$$

$$P(y = 0|x) = 1 - P(y = 1|x) = 1 - h_\theta(x) \tag{3.16}$$

η $P(y = 1|x)$ να παίρνει μεγάλες τιμές όταν το x ανήκει στην κλάση "1" και μικρές όταν το x ανήκει στην κλάση "0". Για ένα σετ δειγμάτων *(binary labeled)* $\left\{\left(x^{(i,)}, y^{(i)}\right): i = 1, \dots, m\right\}$ η συνάρτηση κόστους έχει τη μορφή:

$$J(\theta) = -\sum_i \left(y^{(i)}\, log\left(h_\theta\left(x^{(i)}\right)\right) + \left(1 - y^{(i)}\right) log\left(1 - h_\theta\left(x^{(i)}\right)\right)\right) \tag{3.17}$$

Αν παρατηρήσει κανείς, εύκολα θα δει ότι μόνο ο ένας από τους δύο όρους δεν μηδενίζεται στο άθροισμα για κάθε δείγμα, οπότε στην ελαχιστοποίησή της όταν $y^{(i)} = 1$ πρέπει να αυξήσουμε πολύ την $h_\theta\left(x^{(i)}\right)$, ενώ αντίστοιχα όταν $y^{(i)} = 0$ πρέπει να αυξήσουμε πολύ την $1 - h_\theta\left(x^{(i)}\right)$.

Softmax παλινδρόμηση

Η *softmax regression* είναι μια γενίκευση της λογιστικής παλινδρόμησης στην περίπτωση που θέλουμε να χειριστούμε περισσότερες από δύο κλάσεις. Σε αυτή την

προσέγγιση οι προσημασμένες ετικέτες στα δεδομένα μας είναι της μορφής $y^{(i)} \in \{1, \dots, K\}$ για K αριθμό κλάσεων.

Δοσμένου ενός δείγματος x θέλουμε μέσω της $h_\theta(x)$ να εκτιμήσουμε την πιθανότητα $P(y = k|x)$ για κάθε κλάση. Έτσι, η υποθετική $h_\theta(x)$ θα εξάγει ένα K-διάστατο διάνυσμα (του οποίου τα στοιχεία αθροίζουν στο 1, αφού μιλάμε για πιθανότητες) της μορφής:

$$h_{W,b}(x) = \begin{bmatrix} P(y = 1|x;\theta) \\ P(y = 2|x;\theta) \\ \vdots \\ P(y = K|x;\theta) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} exp(\theta^{(j)^T} x)} \begin{bmatrix} exp(\theta^{(1)^T} x) \\ exp(\theta^{(2)^T} x) \\ \vdots \\ exp(\theta^{(K)^T} x) \end{bmatrix} \tag{3.18}$$

Αν το κάθε δείγμα έχει n χαρακτηριστικά είναι λογικό ότι $\theta^{(1)}, \theta^{(2)}, \dots, \theta^{(K)} \in \mathbb{R}^n$. Το εκθετικό άθροισμα στον παρονομαστή κανονικοποιεί τον πίνακα ώστε το άθροισμα των στοιχείων του να κάνει 1, δηλαδή την αρχική μας απαίτηση.

Η παράμετρος θ σε διανυσματική μορφή περιγράφεται από έναν πίνακα $n \times K$.

$$\theta = \begin{bmatrix} \theta_{11} & \theta_{12} & \cdots & \theta_{1K} \\ \theta_{21} & \theta_{22} & \dots & \theta_{2K} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{n1} & \theta_{n2} & \cdots & \theta_{nK} \end{bmatrix} \tag{3.19}$$

Για να περιγράψουμε τη συνάρτηση κόστους θα χρειαστούμε μια συνάρτηση ελέγχου για την οποία ισχύει: $\delta = \begin{cases} 1, & y^{(i)} = k \\ 0, & y^{(i)} \neq k \end{cases}$ (3.20)

Άρα:

$$J(\theta) = -\left[\sum_{i=1}^{m}\sum_{k=1}^{K} \delta \cdot log \frac{exp\left(\theta^{(k)^T} x^{(i)}\right)}{\sum_{j=1}^{K} exp\left(\theta^{(j)^T} x^{(i)}\right)}\right] \tag{3.21}$$

όπου m το πλήθος των δειγμάτων. Η συνάρτηση κόστους είναι η γενικευμένη μορφή της συνάρτησης κόστους στη λογιστική παλινδρόμηση με τη διαφορά ότι τώρα αθροίζουμε σε K διαφορετικές πιθανές τιμές κλάσεων. Επιπλέον,

$$P\left(y^{(i)} = k|x^{(i)};\theta\right) = \frac{\exp\left(\theta^{(k)^T} x^{(i)}\right)}{\sum_{j=1}^{K} \exp\left(\theta^{(j)^T} x^{(i)}\right)} \tag{3.22}$$

Για $K = 2$ η συνάρτηση $J(\theta)$ ανάγεται στη γνωστή σχέση της λογιστικής παλινδρόμησης. Παρατηρούμε ότι η *softmax* συνάρτηση είναι κανονικοποιημένη και δίνει μια διαίσθηση των πιθανοτήτων, ενώ η συνάρτηση κόστους δίνει τις ενεργοποιήσεις (ή αλλιώς *score*) και λέγεται *cross-entropy*.

3.5 Ο αλγόριθμος Backpropagation

Όπως προαναφέραμε, ο σκοπός ενός *NeuralNet* είναι να δώσει στα βάρη του τιμές τέτοιες ώστε για κάθε δεδομένο που δέχεται στην είσοδό του, να παράγει την επιθυμητή έξοδο. Για να το πετύχει αυτό πρέπει να περάσει από ένα στάδιο εκπαίδευσης ή αλλιώς εκμάθησης, με άλλα λόγια, να επαναπροσδιορίσει αρκετές φορές τις τιμές τους για να φτάσει στο επιθυμητό αποτέλεσμα.

Συνάρτηση κόστους *(cost function)*: Ας υποθέσουμε λοιπόν ότι έχουμε ένα σετ εκπαίδευσης $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$ m δειγμάτων εκπαίδευσης. Ορίσαμε τη συνάρτηση κόστους ως μια συνάρτηση της οποίας η τιμή αντιπροσωπεύει την απόδοση του δικτύου στην πρόβλεψη των επιθυμητών εξόδων με την πιο γενική μορφή:

$$J(W, b; x, y) = \frac{1}{2}\left\|h_{W,b}(x) - y\right\|^2 \tag{3.23}$$

Επομένως για ένα σύνολο m δειγμάτων *(batch)* η ολική συνάρτηση κόστους παίρνει τη μορφή:

$$\begin{aligned} J(W, b) &= \left[\frac{1}{m}\sum_{i=1}^{m} J\left(W, b; x^{(i)}, y^{(i)}\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(W_{ji}^{(l)}\right)^2 \\ &= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}(x) - y\right\|^2\right)\right] + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l+1}\left(W_{ji}^{(l)}\right)^2 \end{aligned} \tag{3.24}$$

Weight decay: Παρατηρούμε ότι στη συνάρτηση κόστους εκτός από το μέσο τετραγωνικό σφάλμα έχουμε προσθέσει έναν επιπλέον όρο που ονομάζεται *weight decay* και προσπαθεί να ελαττώσει εκθετικά τα βάρη στο μηδέν. **Ο σκοπός αυτού, είναι να αποφύγει την υπερπροσαρμογή του δικτύου *(overfitting)*.** Άρα έχουμε εισάγει μία γκαουσσιανή με σκοπό να παραμετροποιήσει τη συνάρτηση κόστους. Στην πράξη αυτό, "τιμωρεί" τα μεγάλα βάρη περιορίζοντας την ελευθερία του συστήματος.

Gradient descent: Εφόσον η απόδοση του δικτύου πάνω στο πρόβλημα αντιπροσωπεύεται από τη συνάρτηση κόστους, στόχος μας είναι η ελαχιστοποίηση της. Η συνάρτηση κόστους έχει ως ανεξάρτητες παραμέτρους τα βάρη και τα *bias* και ως μη κοίλη παρουσιάζει ελάχιστο, άρα ένας αλγόριθμος βελτιστοποίησης αποτελεί ο *gradient descent* που παίρνει τις μικρότερες τιμές του κοντά σε τοπικά ελάχιστα.

Πρώτα, πρέπει να αρχικοποιηθούν τυχαία τα βάρη και τα *bias* με μικρές τιμές γύρω από το μηδέν. Η αρχικοποίηση με τυχαίο τρόπο εξυπηρετεί δημιουργώντας μια

ασυμμετρία στις εξόδους των νευρώνων. Αν ορίζαμε την ίδια τιμή σε όλα τα βάρη τότε όλοι οι νευρώνες θα είχαν την ίδια έξοδο.

Σε κάθε επανάληψη, βάσει της *gradient descent* τα βάρη και τα *bias* αλλάζουν ως εξής:

$$W_{ij}^{(l)} := W_{ij}^{(l)} - \alpha^{1} \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) \tag{3.25}$$

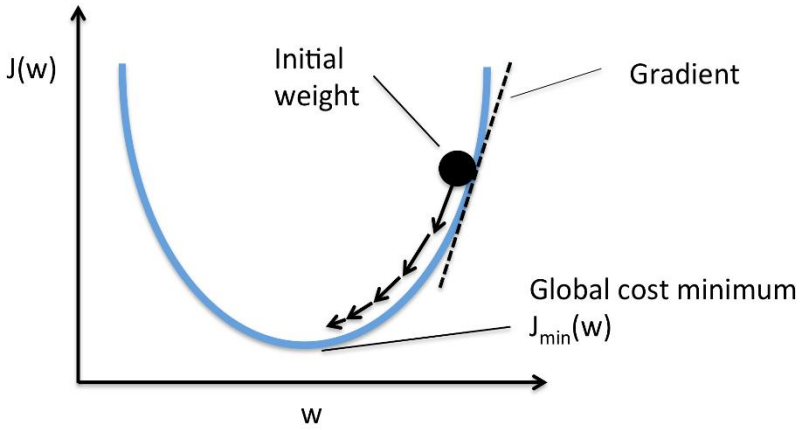
$$b_i^{(l)} := b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b) \tag{3.26}$$

Παρατηρούμε ότι, θετική παράγωγος στη συνάρτηση κόστους δείχνει ότι η κλίση είναι θετική, το οποίο σημαίνει πως απομακρυνόμαστε από τοπικό ελάχιστο, τα βάρη παίρνουν μικρότερη τιμή από 'κείνη που είχαν στην προηγούμενη επανάληψη. Αντιθέτως για αρνητική κλίση, πλησιάζουμε στο τοπικό ελάχιστο, άρα τα βάρη παίρνουν μεγαλύτερες τιμές από τις προηγούμενες προσπαθώντας να το "φτάσουν".

Η μορφή της συνάρτησης κόστους στο χώρο εξαρτάται από το πλήθος των βαρών του προβλήματος. Οι μερικές παράγωγοι της συνάρτησης κόστους στην περίπτωση της *stochastic gradient descent* δηλαδή όταν η ίδια υπολογίζεται σε κάθε επανάληψη πάνω σε ένα δείγμα υπολογίζονται ως:

$$\frac{\partial J(W, b)}{\partial W_{ij}^{(l)}} = \left[\frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial W_{ij}^{(l)}} J\left(W, b; x^{(i)}, y^{(i)}\right) \right] + \lambda W_{ij}^{(l)} \tag{3.27}$$

$$\frac{\partial J(W, b)}{\partial b_i^{(l)}} = \left[\frac{1}{m} \sum_{i=1}^{m} \frac{\partial}{\partial b_i^{(l)}} J\left(W, b; x^{(i)}, y^{(i)}\right) \right] \tag{3.28}$$

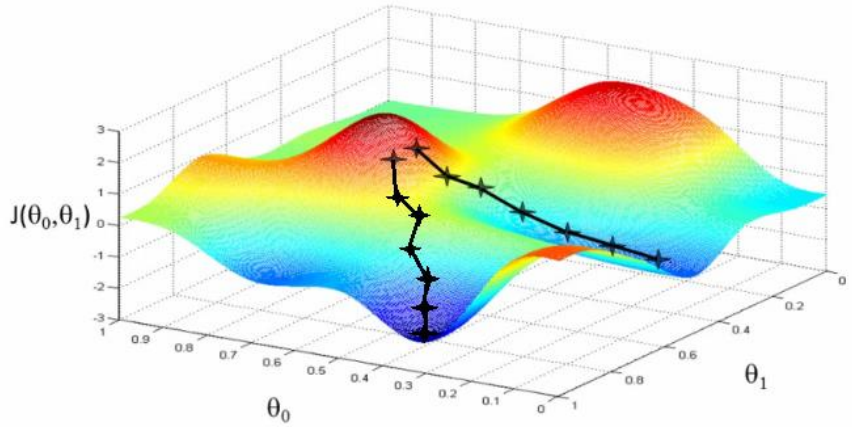


Εικόνα 3.4: Μονοδιάστατο πρόβλημα με θετική κλίση στη συνάρτηση κόστους

[1] Όπου α ο ρυθμός εκμάθησης.

Εύκολα γίνεται αντιληπτό ότι ο όρος *weight decay* δεν εξαρτάται από *bias term.*

Η συνάρτηση κόστους στον νευρώνα εξόδου υπολογίζεται εύκολα συγκρίνοντας την προβλεπόμενη τιμή με την έξοδο του νευρώνα, δηλαδή την $h_{W,b}(x)$. Τι γίνεται όμως με τα ενδιάμεσα επίπεδα; Πρέπει να ορίσουμε έναν όρο $\delta_i^{(l)}$ ο οποίος να αντιπροσωπεύει πόσο διαφέρει η έξοδός του από την αναμενόμενη τιμή (του συγκεκριμένου νευρώνα),



Εικόνα 3.5: Δισδιάστατο πρόβλημα με 2 ανεξάρτητες παραμέτρους βαρών θ_0 και θ_1

όσο μεγαλύτερος είναι τόσο περισσότερο ευθύνεται για την απόκλιση της συνολικής εξόδου από την αναμενόμενη τιμή καθώς αυτό το σφάλμα διαδίδεται στα επόμενα επίπεδα. Ο αλγόριθμος οπισθοδιάδοσης σφάλματος *(backpropagation algorithm)* δίνεται στα εξής τέσσερα βήματα:

1. κατά τη διάδοση προς τα εμπρός υπολογίζονται οι ενεργοποιήσεις κάθε επιπέδου μέχρι και την τελική έξοδο $(L_2, L_3, \dots, L_{nl})$
2. για κάθε νευρώνα εξόδου η μερική παράγωγος της συνάρτησης εξόδου, σύμφωνα με τα παραπάνω, υπολογίζεται ως:

$$\delta_i^{(nl)} = \frac{\partial}{\partial z_i^{(nl)}} \frac{1}{2} \left\| y - h_{W,b}(x) \right\|^2 = -\left(y_i - a_i^{(nl)}\right) \cdot f'\left(z_i^{(nl)}\right) \tag{3.29}$$

3. στη συνέχεια, το δ των ενδιάμεσων νευρώνων υπολογίζεται από τα τελευταία επίπεδα προς τα πίσω (εξού και ο όρος "οπισθοδιάδοση"). Οπότε για κάθε κόμβο i σε κάθε επίπεδο l:

$$\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \, \delta_j^{(l+1)} \right) \cdot f'(z_i^{(l)}) \tag{3.30}$$

4. και οι μερικές τους παράγωγοι δίνονται:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b;, x, y) = a_j^{(l)} \delta_i^{(l+1)} \tag{3.31}$$

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b;, x, y) = \delta_i^{(l+1)} \tag{3.32}$$

5. στο τελευταίο βήμα γίνεται η ενημέρωση των βαρών και των *bias*:

$$\Delta W_{ij}^{(l)} = -\alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) \tag{3.33}$$

$$\Delta b_i^{(l)} = -\alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b) \tag{3.34}$$

Όπου $\Delta W_{ij}^{(l)}$ η διαφορά της καινούριας τιμής των βαρών με την προηγούμενη (αντίστοιχα για το *bias*).[7-9]

3.6 Εκπαίδευση δικτύου

Πριν συνεχίσουμε θα πρέπει να αποσαφηνίσουμε κάποιες έννοιες. Το δίκτυο λέγεται ότι εκπαιδεύεται όσο ελαχιστοποιεί τη συνάρτηση κόστους του. Κάθε φορά που εφαρμόζεται η *gradient descent* τα βάρη μεταβάλλονται λίγο μετακινούμενα στο τοπικό (*ή ολικό*) ελάχιστο. Επομένως, ο αλγόριθμος *backpropagation* πρέπει να εκτελεστεί αρκετές φορές. Αυτό μας οδηγεί στην εισαγωγή των όρων:

Εποχή: Μια εποχή διαρκεί όσο εκτελείται ένα *forward* και ένα *backward pass* **σε όλο** το *training set* μέσα από το δίκτυο.

Batch size: Έτσι ονομάζεται ο αριθμός των δειγμάτων εκπαίδευσης στα οποία συμβαίνει ένα *forward* και *backward pass* ταυτόχρονα. Αποτελούν δηλαδή, ένα υποσύνολο του *training set*.

Iteration: Σε κάθε επανάληψη ένα *batch* έχει υποστεί *forward* και *backward pass*.

Για παράδειγμα σε ένα σετ εκπαίδευσης που περιέχει 500 δεδομένα και είναι χωρισμένο σε *batch size* των 250 θα χρειαστεί δύο επαναλήψεις για να ολοκληρώσει μία εποχή. Όταν η συνάρτηση κόστους δεν ελαχιστοποιείται περισσότερο, λέμε ότι ολοκληρώθηκε η εκπαίδευση του δικτύου.

3.7 Βελτιστοποίηση Gradient Descent

Ο αλγόριθμος *gradient descent* αποτελεί έναν από τους πιο διαδεδομένους αλγόριθμους που επιδέχονται βελτιστοποίηση, καθώς και ο πιο κοινός τρόπος να βελτιώσει κανείς ένα νευρωνικό δίκτυο. Ο λόγος για τον οποίο επιχειρείται η βελτιστοποίηση είναι στην επιτάχυνση της εκπαίδευσης και των υπολογισμών.

Οι πιο βασικές παραλλαγές της είναι:

Batch Gradient Descent:

Υπολογίζει τις παραγώγους της συνάρτησης κόστους των παραμέτρων θ για ολόκληρο το δείγμα εκπαίδευσης και ενημερώνονται σε κάθε εποχή:

$$\theta = \theta - \alpha \nabla_\theta J(\theta) \tag{3.35}$$

Εφόσον ο υπολογισμός γίνεται σε ολόκληρο το *set* οι υπολογισμοί μπορεί να είναι πολύ αργοί ή δύσχρηστοι για πολύ μεγάλο δείγμα δεδομένων αλλά αν η επιφάνεια της συνάρτησης κόστους είναι κυρτή τότε θα συγκλίνει στο ολικό ελάχιστο ή στο τοπικό ελάχιστο αν δεν είναι κυρτή.

Stochastic Gradient Descent:

Σε αντίθεση με την *BGD* οι παράμετροι υπολογίζονται και ενημερώνονται για κάθε δείγμα που μπαίνει στο δίκτυο $(x^{(i)}, y^{(i)})$:

$$\theta = \theta - \alpha \nabla_\theta J\left(\theta; x^{(i)}, y^{(i)}\right) \tag{3.36}$$

Οι υπολογισμοί γίνονται πολύ γρήγορα εφόσον αφορούν ένα δείγμα σε κάθε επανάληψη αλλά μερικοί από αυτούς είναι περιττοί. Η όλη διαδικασία είναι πολύ αργή.

Mini-batch Gradient Descent:

Σε αυτή την περίπτωση, παίρνουμε έναν συνδυασμό των *BGD* και *SGD* καθώς ο υπολογισμός των παραμέτρων γίνεται πάνω σε ένα *mini-batch* δεδομένων, οπότε σε κάθε εποχή, όσο μεγαλύτερο το *mini-batch* τόσες περισσότερες ενημερώσεις.

$$\theta = \theta - \alpha \nabla_\theta J\left(\theta; x^{(i:i+n)}, y^{(i:i+n)}\right) \tag{3.37}$$

Adam:

Adaptive Moment Estimation είναι μια μέθοδος η οποία προσαρμόζει τον ρυθμό εκμάθησης για κάθε παράμετρο. Αποθηκεύει έναν εκθετικά αποσβαινόμενο μέσο των προηγούμενων παραγώγων m_t, καθώς των τετραγωνισμένων παραγώγων υ_t όμοια με την ορμή:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{3.38}$$

$$\upsilon_t = \beta_2 \upsilon_{t-1} + (1 - \beta_2) g_t^2 \tag{3.39}$$

m_t και υ_t είναι εκτιμήσεις της πρώτης στιγμής (μέσης) και της δεύτερης στιγμής (της μη προσανατολισμένης διακύμανσης) των παραγώγων αντίστοιχα. Επειδή είναι αρχικοποιημένα ως μηδενικά διανύσματα, οι δημιουργοί της μεθόδου παρατήρησαν ότι μεροληπτούν προς το μηδέν, ειδικά τις πρώτες χρονικές στιγμές και όταν οι μειωτικοί όροι είναι πολύ μικροί (δηλαδή όταν $\beta_{1,2} \cong 1$).

Αντιδρούν σε αυτή τη συμπεριφορά διορθώνοντας τους όρους εκτίμησης της πρώτης και της δεύτερης χρονικής στιγμής:

$$\widehat{m}_t = \frac{m_t}{1-\beta_1^t} \tag{3.40}$$

$$\widehat{\upsilon_t} = \frac{\upsilon_t}{1-\upsilon_2^t} \tag{3.41}$$

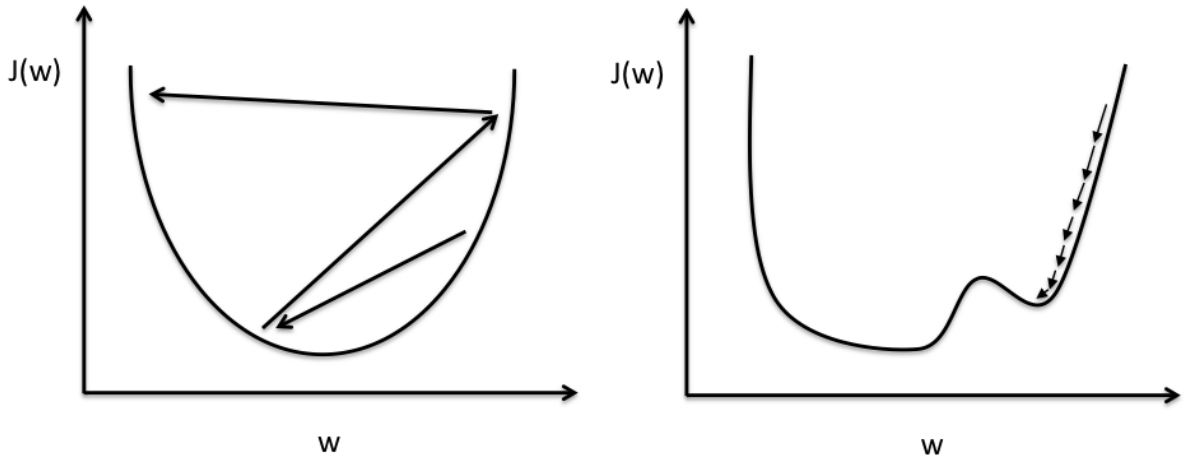
Ύστερα, ενημερώνουν τις παραμέτρους όπως ορίζει η μέθοδος:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\widehat{\upsilon_t}} + \varepsilon}\widehat{m_t} \tag{3.42}$$

Οι δημιουργοί προτείνουν ως προκαθορισμένες τιμές: $\beta_1 = 0.9, \beta_2 = 0.999$ και $\varepsilon = 10^{-8}$. Ο αλγόριθμος *Adam* έχει δείξει ότι λειτουργεί καλά σε σύγκριση με άλλους αλγορίθμους προσαρμοστικής μάθησης.[9]

Εκτός των παραλλαγών στον αλγόριθμο μπορούμε να αλλάξουμε και άλλες παραμέτρους που οδηγούν στη βελτιστοποίηση ενός δικτύου. Αυτές καλούνται *hyperparameters* και μπορούμε να τις αλλάξουμε μόνο αλλάζοντας την τιμή τους. Μία από αυτές είναι ο παράγοντας λ που συναντήσαμε στην εξασθένιση του βάρους (*weight decay*). Άλλες είναι:

Ρυθμός εκμάθησης: *Learning rate* ή *Lr* λέγεται ο παράγοντας α που μπαίνει μπροστά από τις παραγώγους των παραμέτρων και καθορίζει το μέγεθος του βήματος προς το ελάχιστο. Η σωστή επιλογή του έχει μεγάλη σημασία καθώς πολύ μεγάλες τιμές οδηγούν σε μεγάλα βήματα που εμποδίζουν τη σύγκλιση (*overshooting*) όπως και πολύ μικρές τιμές οδηγεί σε πολύ μικρά βήματα που καθυστερούν αρκετά την εκπαίδευση. Μια μέση λύση αποτελεί ο μεταβλητός ρυθμός εκμάθησης. Διαλέγοντας έναν ρυθμό εκμάθησης που μειώνεται σταδιακά σε κάθε εποχή, οδηγεί σε σύγκλιση με πολύ μικρές διακυμάνσεις στη διάρκεια εκπαίδευσης. Ένας άλλος τρόπος αποτελεί η επιλογή ενός *scheduler* που προγραμματίζει πριν την εκπαίδευση τον τρόπο με τον οποίο θα μεταβάλλεται ο α, ή κατά τη διάρκεια εκπαίδευσης όταν η συνάρτηση κόστους ξεπερνάει μια τιμή κατωφλίου.



Εικόνα 3.6: Μεγάλος ρυθμός εκμάθησης και πολύ μικρός αντίστοιχα.

Ορμή: Η *SGD* αντιμετωπίζει προβλήματα καθώς περνάει από τις απότομες περιοχές της συνάρτησης κόστους (σημεία δηλαδή που η κλίση είναι πιο απότομη σε μία διάσταση παρά στις άλλες) οι οποίες είναι συχνότερες γύρω από κάποιο ακρότατο. Οπότε, αρχίζει να ταλαντώνεται στις πλαγιές κατευθυνόμενη διστακτικά προς το ελάχιστο. Μπορούμε να επιταχύνουμε αυτή την πορεία προς την επιθυμητή κατεύθυνση χρησιμοποιώντας μια παράμετρο που λέγεται ορμή.

Εικόνα 3.7 Αριστερά: χωρίς ορμή. Δεξιά: με ορμή.

Στο διάνυσμα της ταχύτητας u_t, το οποίο έχει ίδιες διαστάσεις με το θ εισάγουμε μια παράμετρο $\gamma \in (0,1]$. Η ταλάντωση σε 'κείνα τα σημεία οφείλεται στις μεγάλες τιμές του $\nabla_\theta J(\theta)$ που αντιπροσωπεύει την κλίση, κατά συνέπεια ο ρυθμός εκμάθησης θα πρέπει να είναι μικρός. Τελικά, η ορμή προσδίδει μια ταχύτητα, αναγκάζοντας τη συνάρτηση να κάνει μεγαλύτερα βήματα για να μην εγκλωβιστεί σε κάποιο τοπικό ελάχιστο και να συνεχίσει να ψάχνει το ολικό. Ιδανικά η τιμή της θα πρέπει να είναι σταθερή σε κάποια ενδιάμεση τιμή σε όλη την πορεία της εκπαίδευσης και να αυξάνεται όταν η σύγκλιση αρχίζει να σταθεροποιείται.[9, 10]

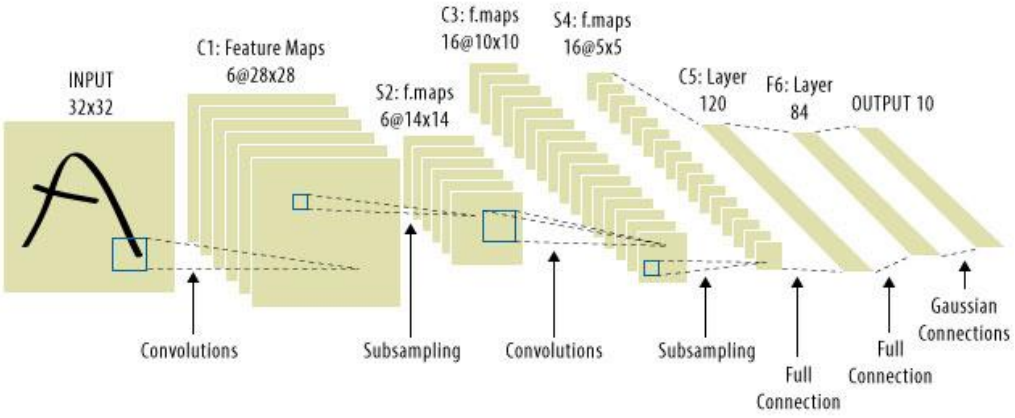
$$u_t = \gamma u_{t-1} + \alpha \nabla_\theta J(\theta) \tag{3.43}$$

$$\theta = \theta - u_t \tag{3.44}$$

3.8 Συνελικτικά Νευρωνικά Δίκτυα (*CNN*)

Τα *CNN* χρησιμοποιούνται για προβλήματα ταξινόμησης εικόνων. Στα πλήρως συνδεδεμένα νευρωνικά δίκτυα είδαμε ότι κάθε νευρώνας ήταν πλήρως συνδεδεμένος με κάθε δεδομένο εισαγωγής όπως και με τους νευρώνες του επόμενου επιπέδου. Για μικρές εικόνες, τύπου 28×28 οι υπολογισμοί ήταν εφικτοί για την εύρεση χαρακτηριστικών πάνω σε ολόκληρη την εικόνα. Η εξέταση 100 χαρακτηριστικών για παράδειγμα θα απαιτούσε τον υπολογισμό $28 \times 28 \times 100 \approx 8 \cdot 10^4$ παραμέτρων, ενώ αν θέλαμε να εξετάσουμε τον ίδιο αριθμό χαρακτηριστικών σε εικόνες 96×96 θα χρειαζόμασταν 10^6 παραμέτρους και ο αλγόριθμος *backpropagation* θα ήταν 100 φορές πιο αργός.[7]

Η κεντρική ιδέα πίσω από ένα *ConvNet* είναι ο περιορισμός των συνδέσεων μεταξύ των νευρώνων και των δεδομένων εισαγωγής, επιτρέποντας σε κάθε νευρώνα να συνδεθεί μόνο με ένα μέρος των δεδομένων. Η τοπικότητα των νευρώνων είναι εμπνευσμένη από την ανακάλυψη ενός οπτικού μηχανισμού, του οπτικού φλοιού, στον εγκέφαλο. Ο οπτικός φλοιός περιέχει κύτταρα τα οποία είναι υπεύθυνα για την ανίχνευση του φωτός σε μικρές "υπο-περιοχές" του οπτικού πεδίου. Ένα συνελικτικό δίκτυο προσπαθεί να μιμηθεί αυτή τη λειτουργία.[11]



Εικόνα 3.8: Τυπική τοπολογία ενός CNN

3.9 Αρχιτεκτονική των *CNN*

Οι νευρώνες στα συνελικτικά δίκτυα αναπαριστούν τρισδιάστατους όγκους σε κάθε επίπεδο σε αντίθεση με τους νευρώνες των προηγούμενων δικτύων που αναπαριστούσαν ανεξάρτητες μονάδες σε κάθε επίπεδο. Αυτό συμβαίνει για να εκμεταλλευτούν το γεγονός ότι τα δεδομένα αναπαριστούν εικόνες. Συγκεκριμένα, κάθε επίπεδο ενός *CNN* είναι διευθετημένο στις 3 διαστάσεις (*ύψος, πλάτος, βάθος*). Η διάσταση του βάθους υπεισέρχεται λόγω της τρίτης διάστασης της εικόνας, τα 3 χρωματικά κανάλια για παράδειγμα (μπορεί η διάσταση να είναι και μία). Επιπλέον, η έξοδος του δικτύου θα

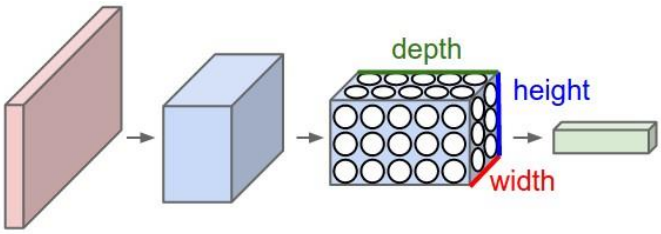
αποτελείται από ένα επίπεδο νευρώνων με διάσταση $1 \times 1 \times K$, όπου K ο αριθμός των κλάσεων.

Στην εικόνα 3.9, αν το κόκκινο κομμάτι αναπαριστά μια εικόνα διαστάσεων $H \times W \times D$ η διάσταση βάθους του μπλε νευρώνα είναι ίση με την τρίτη διάσταση της εικόνας, και το επίπεδο στην έξοδο μετασχηματίζει την αρχική εικόνα σε ένα διάνυσμα διάστασης $1 \times 1 \times K$ που σε κάθε K η πιθανότητα να ανήκει στην αντίστοιχη κλάση.

Άρα λοιπόν, ένα *ConvNet* είναι μια αλληλουχία επιπέδων που μετασχηματίζουν έναν όγκο ενεργοποιήσεων σε έναν άλλον μέσω κάποιας συνάρτησης.[12]

Παράδειγμα: Για μια εικόνα 96×96 και έναν ανιχνευτή 8×8 που παριστάνει κάποιο χαρακτηριστικό, ξεκινώντας από την πάνω αριστερή γωνία (1,1) και σαρώνοντας την εικόνα προς τα δεξιά πραγματοποιώντας συνελίξεις για κάθε περιοχή 8×8 που διασχίζει, θα τερματίσει στο σημείο (89,89) έχοντας συλλέξει 89×89 ενεργοποιήσεις.

Η τιμή κάθε ενεργοποίησης στον τελικό πίνακα, δείχνει το **κατά πόσο ανιχνεύθηκε** το συγκεκριμένο χαρακτηριστικό πάνω στην εικόνα.



Εικόνα 3.9: CNN του οποίου τα επίπεδα αναπαριστούν όγκους.

Η αρχιτεκτονική ενός συνελικτικού δικτύου χρησιμοποιεί τα εξής βασικά επίπεδα:

Convolutional Layer: Πρόκειται για το βασικό επίπεδο ενός *CNN* καθώς και το επίπεδο που δέχεται τον μεγαλύτερο υπολογιστικό φόρτο. Όπως αναφέραμε στην αρχή, η κεντρική ιδέα είναι ο περιορισμός των συνδέσεων του νευρώνα με τα δεδομένα εισαγωγής (την εικόνα).

Οι φυσικές εικόνες έχουν την ιδιότητα να μένουν στατικές, πράγμα που διευκολύνει την εξαγωγή ίδιων χαρακτηριστικών από διαφορετικές περιοχές πάνω στην εικόνα. Συγκεκριμένα, χρησιμοποιώντας έναν ανιχνευτή (π.χ. 8×8) που παριστάνει κάποιο χαρακτηριστικό, μπορούμε να συλλέξουμε διαφορετικές ενεργοποιήσεις για το ίδιο χαρακτηριστικό μετακινούμενοι σε διαφορετικά σημεία πάνω στην εικόνα.

Τυπικότερα, έχοντας εικόνες διάστασης $r \times c$ (x_{large}), πρώτα εκπαιδεύουμε έναν τυπικό *autoencoder*[2] πάνω σε μικρότερες περιοχές $a \times b$ (x_{small}) των αρχικών εικόνων, μαθαίνοντας k χαρακτηριστικά $f = \sigma\left(W^{(1)}x_{small} + b^{(1)}\right)$ (με σ τη σιγμοειδή και $W^{(1)}, b^{(1)}$ τα συνολικά βάρη του). Για κάθε μικρή περιοχή $a \times b$ (x_s) , υπολογίζουμε $f_s = \sigma\left(W^{(1)}x_s + b^{(1)}\right)$, δίνοντάς μας έναν όγκο $k \times (r - a + 1) \times (c - b + 1)$ χαρακτηριστικά (που προήλθαν από συνέλιξη).

Κάθε *autoencoder* δύο διαστάσεων που αναπαριστά κάποιο από τα k χαρακτηριστικά καλείται φίλτρο ή μάσκα. Ένα *convolutional layer* αποτελείται από k φίλτρα. Επομένως, αν οι εικόνες στην είσοδο, έχουν m κανάλια χρώματος και θέλουμε να ερευνήσουμε χαρακτηριστικά σε όλο τον όγκο της εικόνας, το *convLayer* θα είναι $k \times a \times b \times m$ διαστάσεων. Ή αλλιώς από k όγκους $a \times b \times m$ σε αναλογία με το πρώτο επίπεδο της εικόνας 3.8.[7, 12]

Σε ένα συνελικτικό επίπεδο μπορούμε να ορίσουμε τις εξής *hyperparameters*:

- Stride: Το βήμα του φίλτρου κάθε φορά σε μία διάσταση. Π.χ. όταν είναι s μετακινείται s pixels τη φορά προς τα δεξιά.

- Zero-padding: Γεμίζοντας μηδενικά γύρω από την εικόνα, μας διευκολύνει στο να επιλέγουμε κάθε φορά τις διαστάσεις των ενεργοποιήσεων κάθε φίλτρου χωρίς να αλλοιώνουμε τα χαρακτηριστικά στην εικόνα (αφού τα μηδενικά δεν παράγουν ενεργοποιήσεις). Οι σειρές των μηδενικών συμβολίζονται με p.

Το μέγεθος του όγκου, των ενεργοποιήσεων, καθώς το φίλτρο σαρώνει την εικόνα είναι:

$$\frac{(W - F + 2 \times P)}{S} + 1 \tag{3.45}$$

όπου W οι διαστάσεις της εικόνας, F του φίλτρου, P του *zero-padding* και S του *stride* στην περίπτωση που οι διαστάσεις φίλτρου και εικόνας είναι τετραγωνικές. Διαφορετικά με τον ίδιο τύπο υπολογίζεται η κάθε διάσταση ξεχωριστά.

- Παρατηρούμε λοιπόν, ότι η εισαγωγή της συνέλιξης, περιορίζει πάρα πολύ τον αριθμό των παραμέτρων που χρειάζονται να υπολογιστούν και να αποθηκευτούν στη μνήμη. Τα τελευταία χρόνια που έχουν αυξηθεί οι δυνατότητες των καρτών γραφικών (διατηρώντας χαμηλό το κόστος), ο τομέας του *computer vision* έχει ραγδαίες εξελίξεις σε εφαρμογές συνελικτικών νευρωνικών δικτύων.

[2] Νευρωνικό δίκτυο που χρησιμοποιείται για μη-επιτηρούμενη μάθηση.

Pooling Layer: Μετά την απόκτηση των χαρακτηριστικών από κάθε εικόνα, θέλουμε να προχωρήσουμε σε ταξινόμηση. Συνηθίζεται να υποδειγματολειπτούμε τα δεδομένα με σκοπό την απόρριψη περιττών υπολογιστικών παραμέτρων που μπορεί να αποδειχθούν πολύ απαιτητικοί ή να οδηγήσουν σε *overfitting*.

Η συνηθέστερη μορφή *pooling* είναι η εφαρμογή ενός φίλτρου 2×2 με *stride* 2. Στην ουσία, το φίλτρο σαρώνει τα δεδομένα κρατώντας τη **μέγιστη**[3] τιμή σε κάθε "παράθυρο" κάθε φορά, και μετακινείται ανά 2. Αυτό που επιτυγχάνει είναι η απόκτηση της μέγιστης ενεργοποίησης σε ίσες διαιρεμένες περιοχές στο σύνολο των δεδομένων, απορρίπτοντας το 75% των δεδομένων.

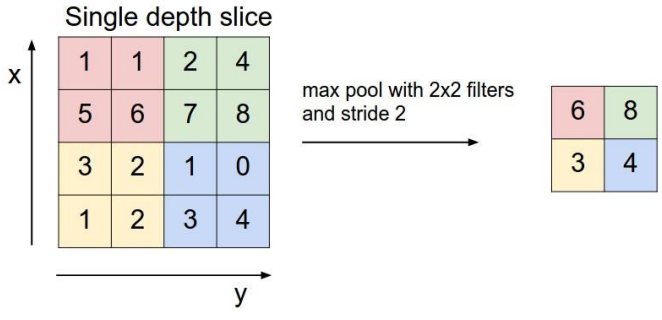
Οι υποδειγματολειπτημένες περιοχές δίνονται από:

$$\frac{(W - F)}{S} + 1 \tag{3.46}$$

Χαρακτηριστικά των *pooling layers*:

- Δέχονται όγκους δεδομένων $H \times W \times D$
- Ορίζονται από 2 *hyperparameters*, το χωρικό μέγεθος F και το βήμα S
- Παράγουν έναν όγκο χωρίς να επεμβαίνουν στην τρίτη διάσταση
- Δεν εισάγουν καινούριες παραμέτρους αφού το αποτέλεσμα προκύπτει από τις τιμές των δεδομένων

Επίσης, δεν συνηθίζεται να εφαρμόζεται *zero-padding*, καθώς ούτε και φίλτρα με πολύ μεγάλο δεκτικό πεδίο γιατί αποδεικνύονται καταστροφικά. Συνηθισμένη τακτική είναι η



Εικόνα 3.10: Εφαρμογή max pooling. Από έναν χώρο 4×4 πηγαίνουμε σε χώρο 2×2

[3] Άλλες συναρτήσεις που χρησιμοποιούνται για *pooling* είναι η *average* (εύρεση της μέσης τιμής των ενεργοποιήσεων) και η *sum* (άθροισμα των ενεργοποιήσεων). Ως επί το πλείστον χρησιμοποιείται η *max*.

χρήση φίλτρων με $F > S$, δηλαδή φίλτρα που επικαλύπτουν περιοχές παραπάνω από μία φορά (*overlapping*).[7, 12]

Μη γραμμικότητα (Non-linearity): Τα νευρωνικά δίκτυα και πιο συγκεκριμένα τα συνελικτικά, βασίζονται σε μη γραμμικές συναρτήσεις για την αναγνώριση σημάτων στα κρυφά επίπεδα. Η εφαρμογή τέτοιων συναρτήσεων αποσκοπεί στην ανόρθωση των χωρικών χαρακτηριστικών σιγουρεύοντας ότι θα είναι θετικά. Οι πιο συνηθισμένες συναρτήσεις που χρησιμοποιούνται είναι οι *sigmoid, tanh* (*hyperbolic tangent*) και η *ReLU*. Το κύριο θετικό της εφαρμογής τους είναι η επίτευξη πολύ γρηγορότερης εκπαίδευσης του δικτύου.

Τα περισσότερα *ConvNets* χρησιμοποιούν την *ReLU* η οποία κρατάει τις θετικές ενεργοποιήσεις θέτοντας τις υπόλοιπες με μηδέν.[13]

Normalization Layer: Πολλοί τρόποι κανονικοποίησης έχουν προταθεί για εφαρμογή εντός των χαρακτηριστικών με σκοπό τη μείωση της διακύμανσης στα δεδομένα και μπορούν να εφαρμοστούν πριν ή μετά το *pooling*. Παρ' όλα αυτά δεν έχουν διαδεδομένη χρήση καθώς δεν προσφέρουν ιδιαίτερα αποτελέσματα. [10, 12]

3.10 Οπισθοδιάδοση σε *CNN*

Θεωρούμε ότι ο όρος σφάλματος για το $(l+1)$ επίπεδο του δικτύου είναι ο $\delta^{(l+1)}$ με συνάρτηση κόστους $J(W, b; x, y)$, όπου (x, y) το υπο εκπαίδευση δείγμα. Αν το επίπεδο l είναι άμεσα συνδεδεμένο με το $(l+1)$ επίπεδο, τότε ο όρος σφάλματος του δεύτερου υπολογίζεται ως:

$$\delta^{(l)} = \left(\left(W^{(l)}\right)^T \delta^{l(+1)}\right) \bullet f'(z^{(l)}) \tag{3.47}$$

με "•" βαθμωτό γινόμενο μεταξύ στοιχείων (*element wise product*) και οι παράγωγοι:

$$\nabla_{W^{(l)}} J(W, b; x, y) = \delta^{(l+1)} \left(a^{(l)}\right)^T \tag{3.48}$$

$$\nabla_{b^{(l)}} J(W, b; x, y) = \delta^{(l+1)} \tag{3.49}$$

ακριβώς όπως είχαμε ξαναδεί στα κανονικά νευρωνικά δίκτυα.

Αν το l επίπεδο είναι *pooling layer* τότε το σφάλμα διαδίδεται ως:

$$\delta^l = upsample\left(\left(W_k^{(l)}\right)^T \delta_k^{l(+1)}\right) \bullet f'(z_k^{(l)}) \tag{3.50}$$

όπου k δείκτης του φίλτρου και $f'(z_k^{(l)})$ η παράγωγος της συνάρτησης ενεργοποίησης. Ο τελεστής *upsample* πρέπει να διαδώσει το σφάλμα μέσα από το *pooling layer* υπολογίζοντάς το ξεχωριστά για κάθε μονάδα που εισέρχεται σε αυτό. Για παράδειγμα, αν είχαμε *average pooling* τότε ο τελεστής απλά θα κατάνειμε ομοιόμορφα το σφάλμα σε κάθε μία μονάδα αναμεταξύ τους, που τροφοδοτούνται σε αυτό προς το προηγούμενο επίπεδο. Ενώ για *max pooling,* η μονάδα που επιλέγεται ως μεγαλύτερη λαμβάνει όλο το σφάλμα διότι, για μικρές αλλαγές στην είσοδο το αποτέλεσμα διαταράσσεται μόνο μέσω εκείνης της μονάδας.

Για να υπολογίσουμε τις παραγώγους, όσο αφορά τα συνελικτικά φίλτρα, βασιζόμαστε στη συνέλιξη με τη διαφορά ότι έχουμε περιστρέψει τα φίλτρα δύο φορές κατά 90° μέσω του τελεστή $rot90(\cdot,2)$ όπως επίσης το ίδιο κάνουμε και για τους πίνακες σφάλματος $\delta_\kappa^{(l)}$.

Άρα:

$$\nabla_{W_k^{(l)}} J(W,b;x,y) = \sum_{i=1}^{m} \left(a_i^{(l)}\right) * rot90(\delta_k^{(l+1)},2) \tag{3.51}$$

$$\nabla_{b_k^{(l)}} J(W,b;x,y) = \sum_{i=1}^{m} \left(\delta_k^{(l+1)}\right)_{a,b} \tag{3.52}$$

όπου $a_k^{(l)}$ η είσοδος στο επίπεδο[4] l. Η πράξη $\left(a_i^{(l)}\right) * \delta_k^{(l+1)}$ είναι η "έγκυρη" συνέλιξη μεταξύ της i-οστής εικόνας εισόδου στο επίπεδο l και το αντίστοιχο σφάλμα αφορά το k φίλτρο.[7]

[4] $a^{(1)}$ η εικόνα εισόδου.

Κεφάλαιο 4: Εξαγωγή χαρακτηριστικών υπογραφής με τη χρήση *CNN*

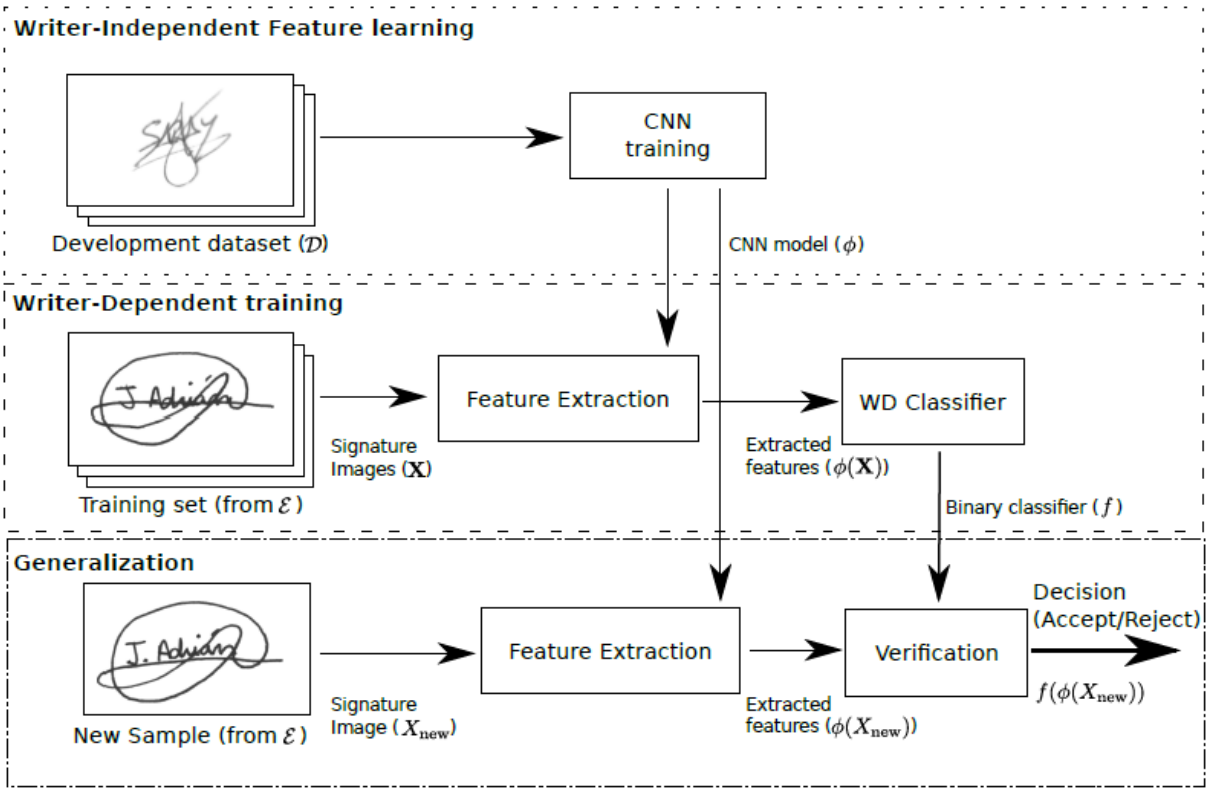
Η αυτόματη επαλήθευση της υπογραφής έχει ερευνηθεί τις τελευταίες δεκαετίες από διάφορες οπτικές γωνίες εμπλέκοντας διάφορους κλάδους όπως η υπολογιστική όραση, η ψηφιακή επεξεργασία σημάτος η γραφολογία κ.ά. μέσα σε αυτούς. Η ανάπτυξη ταξινομητών που ξεχωρίζουν γνήσιες από πλαστές υπογραφές, παρ 'ολα τα πλεονεκτήματα που προσφέρουν, είναι ακόμη δύσκολη. Στηριζόμαστε λοιπόν, στη μέθοδο που προτείνουν οι *Hafemann, Sabourin* και *Oliveira*,[14] στην εκμάθηση μέσω χαρακτηριστικών. Συγκεκριμένα, χαρακτηριστικών που προκύπτουν από τα δεδομένα και όχι χειροποίητων. Γενικότερα, όπως προτείνουν οι ίδιοι εστιάζουμε στην εκμάθηση γενικότερων χαρακτηριστικών που δεν εξαρτώνται από τον γραφέα, και χρησιμοποιούμε το μοντέλο αυτό για την εξαγωγή χαρακτηριστικών και στη συνέχεια στην ανάπτυξη ταξινομητών που εξαρτώνται από τον γραφέα.

Η βάση δεδομένων θα είναι η *CEDAR* η οποία περιέχει 48 δείγματα εκ των οποίων τα μισά (24) είναι γνήσια (*genuine*) και τα άλλα μισά είναι πιστά αντίγραφα (*skill forgeries*) για 55 γραφείς.

Στη συνέχεια θα χωρίσουμε τη βάση δεδομένων σε δύο σετ. Το πρώτο σετ "$\mathcal{D}$" θα περιέχει χρήστες μη εγγεγραμμένους στο σύστημα και θα χρησιμοποιηθεί στην εκπαίδευση του *CNN* για την εκμάθηση των χαρακτηριστικών, ενώ το δεύτερο σετ "$\mathcal{E}$" θα περιέχει τους χρήστες που είναι εγγεγραμμένοι στο σύστημα και θα χρησιμοποιηθεί στην εκπαίδευση των SVM.

Development set "$\mathcal{D}$": Θα αποτελείται από τις 24 γνήσιες υπογραφές 45 γραφέων, δηλαδή από 1080 διαφορετικά δείγματα. Μέσω αυτού του σετ θα γίνει η εκπαίδευση του συνελικτικού δικτύου ώστε να γίνει η αναπαράσταση των χαρακτηριστικών που θεωρείται ότι υπάρχουν στην υπογραφή. Η επιλογή των γραφέων θα γίνει με τέτοιο τρόπο ώστε να υπάρχει μια πληθώρα διαφορετικών μορφών δειγμάτων με κάποια πολυπλοκότητα. Σκοπός είναι η εύρεση κάθε χαρακτηριστικού που πιθανά χαρακτηρίζει μία υπογραφή, ανεξάρτητα από τον γραφέα (*writer independent*).

Exploitation set "$\mathcal{E}$": Αυτό το δείγμα θα περιέχει τους υπόλοιπους 10 χρήστες που είναι εγγεγραμένοι στο σύστημα. Μέσω αυτού θα εκπαιδευτούν 10 SVM (*writer dependent*), δηλαδή η εκπαίδευση θα εξαρτάται από το χρήστη, και στο τέλος της εκπαίδευσης (ιδανικά) θα είναι σε θέση να επαληθεύσει τη γνησιότητα κάθε υπογραφής. Συγκεκριμένα, για την εκπαίδευση θα χρησιμοποιηθούν 14 γνήσια δείγματα για κάθε χρήστη (ως θετικός πληθυσμός) και άλλα 28 τα οποία αποτελούν γνήσιες υπογραφές τυχαίων χρηστών από το σετ "$\mathcal{D}$" (ως αρνητικός πληθυσμός) ώστε να προσομοιώσει το σενάριο μιας πραγματικής εφαρμογής που υστερεί σε πιστά αντίγραφα[14]. Μετά την εκπαίδευση, το τεστ (επαλήθευση) θα γίνει στα υπόλοιπα 10 γνήσια δείγματα του κάθε χρήστη, σε 10 τυχαία πλαστά δείγματα άλλων χρηστών (από το σετ "$\mathcal{E}$") και στα 24 πιστά αντίγραφα που διαθέτουμε για κάθε χρήστη.[15]



Εικόνα 4.1: Η προτεινόμενη μέθοδος για την επαλήθευση off-line υπογραφών

Η μέθοδος περιγράφεται συνοπτικά ως εξής:

- Εκπαίδευση του *CNN* μέσω του σετ "$\mathcal{D}$" με σκοπό την εξαγωγή κάποιας συνάρτησης $\Phi(\cdot)$ που μαθεύτηκε από τα δεδομένα και προβάλλει τα δείγματα X σε έναν χώρο χαρακτηριστικών $\Phi(X) \in \mathbb{R}^m$, όπου m η διάσταση του προβαλλόμενου χώρου των χαρακτηριστικών.
- Εισαγωγή του σετ "$\mathcal{E}$" (*training part*) στο *CNN* και εξαγωγή των χαρακτηριστικών από κατάλληλο *layer*.
- Εκπαίδευση ταξινομητών δύο κλάσεων για κάθε γραφέα με σκοπό την εκμάθηση της συνάρτησης $f(\cdot)$ που κάνει το διαχωρισμό μεταξύ γνήσιων και πλαστών υπογραφών.
- Εισαγωγή του σετ "$\mathcal{E}$" (*testing part*) στο *CNN* κάθε δείγματος X_{NEW} και εξαγωγή των χαρακτηριστικών $\Phi(X_{NEW})$ και τέλος μέσω των *writer dependent* ταξινομητών λαμβάνουμε την τελική απόφαση $f\big(\Phi(X_{NEW})\big)$.

Στη συνέχεια, θα γίνει περιγραφή της μεθόδου βήμα προς βήμα, αναλύοντας τον τρόπο εισαγωγής των εικόνων, περιγραφή του συνελικτικού δικτύου, των ταξινομητών καθώς και συζήτηση των αποτελεσμάτων.

4.1 Προ-επεξεργασία των δεδομένων (*Preprocessing*)

Τα δείγματα της βάσης δεδομένων "CEDAR" δίνονται σε αρχεία εικόνων (*.png*) μεταβλητού μεγέθους διαστάσεων από 306×175 έως 824×796. Επειδή όμως σε ένα νευρωνικό δίκτυο όλα τα δεδομένα εισόδου πρέπει να έχουν το ίδιο μέγεθος αποφασίσαμε να τις αλλάξουμε τις διαστάσεις τους σε 80×120.

Για το *preprocessing* των εικόνων πραγματοποιήθηκαν με την εξής σειρά τα παρακάτω βήματα, τόσο για το σετ "$\mathcal{D}$" όσο και για το σετ "$\mathcal{E}$":

Image cropping: Έγινε περικοπή της περιττής πληροφορίας των εικόνων σύμφωνα με το κέντρο βάρους τους.

Invert image: Στη συνέχεια έγινε αντιστροφή των εικόνων ως προς την τιμή των *pixel*, δηλαδή για κάθε εικόνα Im πραγματοποιήθηκε $Im_{new} = 255 - Im$, ώστε το φόντο τους να έχει την τιμή 0, δηλαδή να είναι μαύρο.

Resizing: Αλλαγή μεγέθους σε 80×120 με τη μέθοδο *"bilinear interpolation"* ώστε να μπορούν να εισαχθούν στο *CNN*. Μειονέκτημα αποτελεί το γεγονός ότι χάνεται πληροφορία σε τόσο μικρό μέγεθος αλλά δεν διαθέταμε υλική υποστήριξη για μεγαλύτερες διαστάσεις.

Data augmentation: Όπως προτάθηκε από τον κ. Ηλία Θεοδωρακόπουλο, λόγω του μικρού αριθμού δειγμάτων, αποφασίσαμε να αυξήσουμε τον αριθμό των δεδομένων με τεχνητό τρόπο ώστε να επιτύχουμε καλύτερη εκπαίδευση του συστήματος με δύο τρόπους:

Περιστρέψαμε όλα τα δείγματα του σετ "$\mathcal{D}$" (γνήσια δείγματα) το καθένα κατά τυχαίο τρόπο σε ένα εύρος $\{-10, +10\}$ μοιρών. Η περιστροφή αυτή είχε ως αποτέλεσμα την αύξηση των διαστάσεων συνεπώς τις αλλάξαμε ξανά στις διαστάσεις αναφοράς. Αυτό είχε ως αποτέλεσμα τον διπλασιασμό του σετ "$\mathcal{D}$" σε 2160 δείγματα.

4.2 Δημιουργία *Development* και *Exploitation sets*.

Το σετ "$\mathcal{D}$" όπως αναφέραμε, θα χρησιμοποιηθεί στην εκπαίδευση του *CNN* με σκοπό την εκμάθηση εκείνης της συνάρτησης $\Phi(\cdot)$ που θα είναι χρήσιμη στην εξαγωγή πολύτιμων χαρακτηριστικών. Αποτελείται από τα γνήσια δείγματα των τελευταίων 45 γραφέων (1080 δείγματα), καθώς και εκείνων που προήλθαν από το *data augmentation*. Από αυτά, θέτουμε ετικέτες *training* και *validation* σε αναλογία 80% προς 20%, δηλαδή, το *CNN* θα χρησιμοποιήσει εκείνα με την ετικέτα *training* για να εκπαιδευτεί και εκείνα με την ετικέτα *validation* για να επαληθεύσει την κατάσταση εκπαίδευσης χωρίς να μεταβάλλει τις παραμέτρους του. Η επιλογή των ετικετών *train*, *validation* γίνεται τυχαία. Τέλος, κανονικοποιούμε τις τιμές, χωρίς να αφαιρούμε το μέσο όρο καθώς θέλουμε το φόντο να παραμείνει μαύρο.

Για την ανάπτυξη του *exploitation set* "$\mathcal{E}$", επιλέγουμε για εκπαίδευση τα 14 πρώτα δείγματα κάθε χρήστη (*genuine*), 14 τυχαία του "$\mathcal{D}$" σετ ως *random forgeries* και 24 *skill forgeries*. Συνολικά το *exploitation set* έχει 620 δείγματα($10 \times 14 + 10 \times 14$ με την ετικέτα *train* και$10 \times 10 + 10 \times 24$ με την ετικέτα *test*). Κανονικοποιούμε και αυτό το σετ γιατί κάθε δεδομένο που εισέρχεται στο *CNN* πρέπει να έχει υποστεί την ίδια προεργασία.

4.3 Εκπαίδευση του *CNN*

Χρησιμοποιούμε τη βιβλιοθήκη *MatConvNet* η οποία παρέχει έτοιμες συναρτήσεις για την υλοποίηση συνελικτικών νευρωνικών δικτύων μέσω του υπολογιστικού πακέτου *matlab* καθώς και έτοιμα δίκτυα για εκπαίδευση ή ήδη εκπαιδευμένα. Η υλοποίηση γίνεται πάνω σε *CPU* (*i7-4770*).

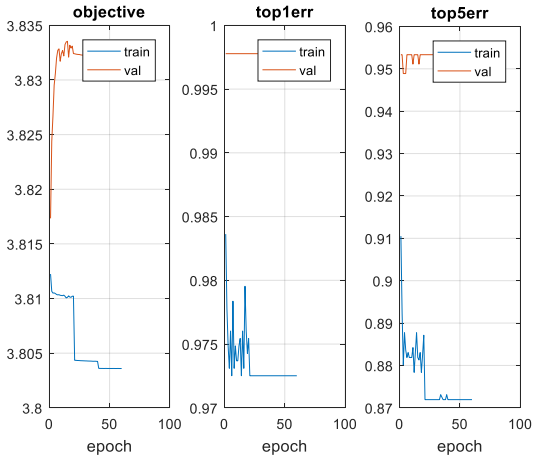
Η δομή του συνελικτικού νευρωνικού δικτύου που σχεδιάσαμε για την εξαγωγή χαρακτηριστικών αποτελείται από 11 επίπεδα *convolutional* και *pooling layers* και στην έξοδο υπάρχει η συνάρτηση *softmax*. Συγκεκριμένα, οι τύποι των επιπέδων με τις διαστάσεις τους δίνονται στον παρακάτω πίνακα:

input	80x120x1	output
conv	11x11x32 stride=2 pad=5	40x60x32
conv	5x5x32 stride=1 pad=2	40x60x32
conv	3x3x64 stride=1 pad=1	40x60x64
pooling	max	20x30x64
conv	3x3x128 stride=1 pad=1	20x30x128
pooling	max	10x15x128
conv	3x3x256 stride=1 pad=1	10x15x256
pooling	max	5x7x256
conv	3x3x512 stride=1 pad=1	5x7x512
pooling	average	1x1x512
Fully Connected	1x1x45	1x1x45
softmaxloss	-	activation

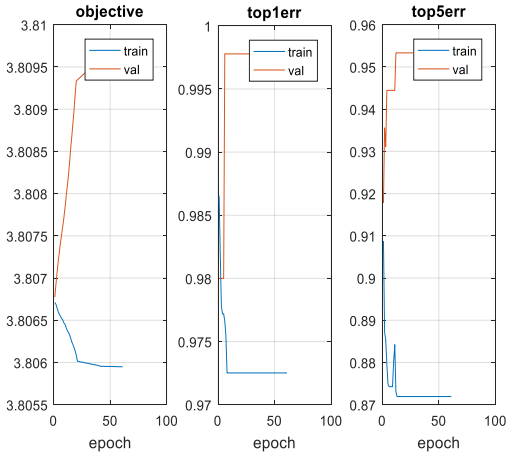
Τα χαρακτηριστικά τα εξάγουμε από το τελευταίο επίπεδο συνέλιξης, μετά το *average pooling*. Μετά την εκπαίδευση η συνάρτηση $\Phi(\cdot)$ δίνει ένα διάνυσμα $1 \times 1 \times 512$ δρώντας πάνω σε μία εικόνα X θεωρώντας ότι η πληροφορία εμπεριέχεται σε εκείνο το διάνυσμα.

Σχεδιάζοντας διαφορετικά το δίκτυο θα μπορούσαμε να επιλέξουμε διαφορετικές διαστάσεις για τα εξαγόμενα χαρακτηριστικά.

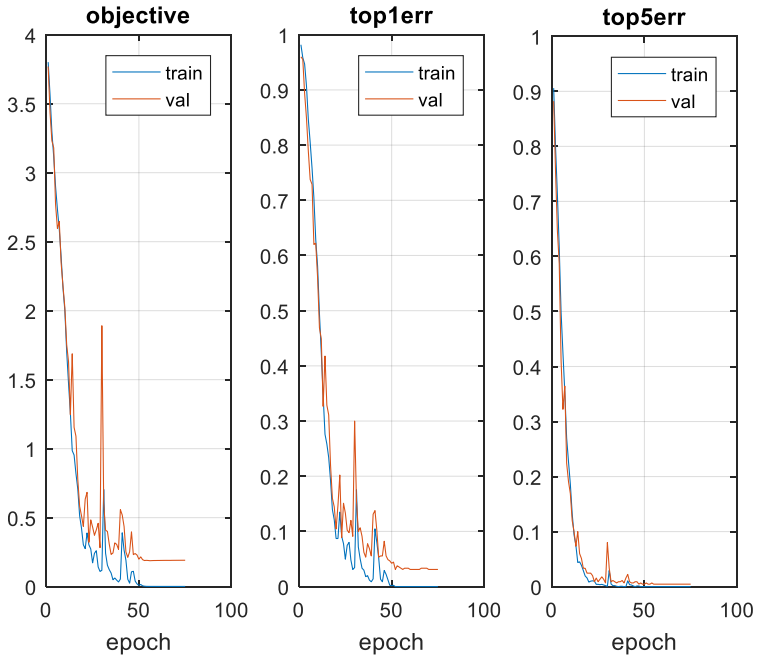
Διάφορες δομές προτάθηκαν στο στήσιμο του συνελικτικού δικτύου όπως οι συναρτήσεις *ReLU* οι οποίες δεν βοήθησαν καθόλου, οπότε δεν χρησιμοποιήθηκαν. Αρχικά, προσπαθήσαμε να εκπαιδεύσουμε το δίκτυο με *SGD* και *BGD* με τις οποίες δεν παρατηρήθηκε σύγκλιση οπότε καταλήξαμε στον αλγόριθμο *Adam* ο οποίος παρεχόταν από τη βιβλιοθήκη.



Εικόνα 4.2: Αποτελέσματα εκπαίδευσης με SGD

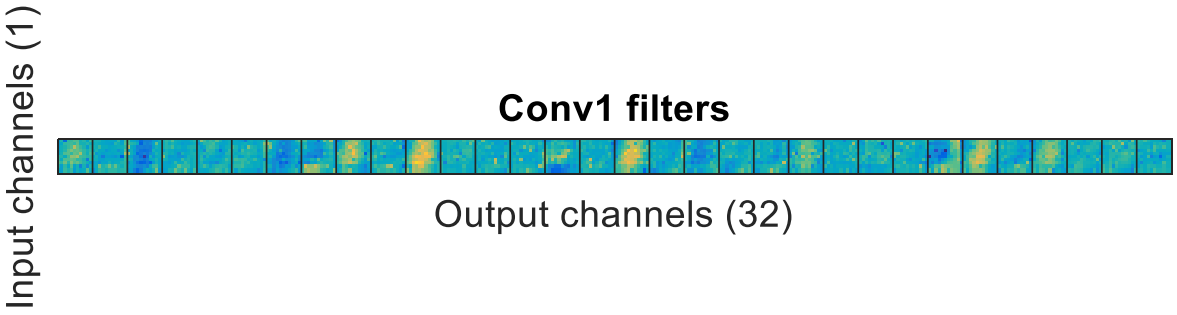


Εικόνα 4.3: Αποτελέσματα εκπαίδευσης με BGD

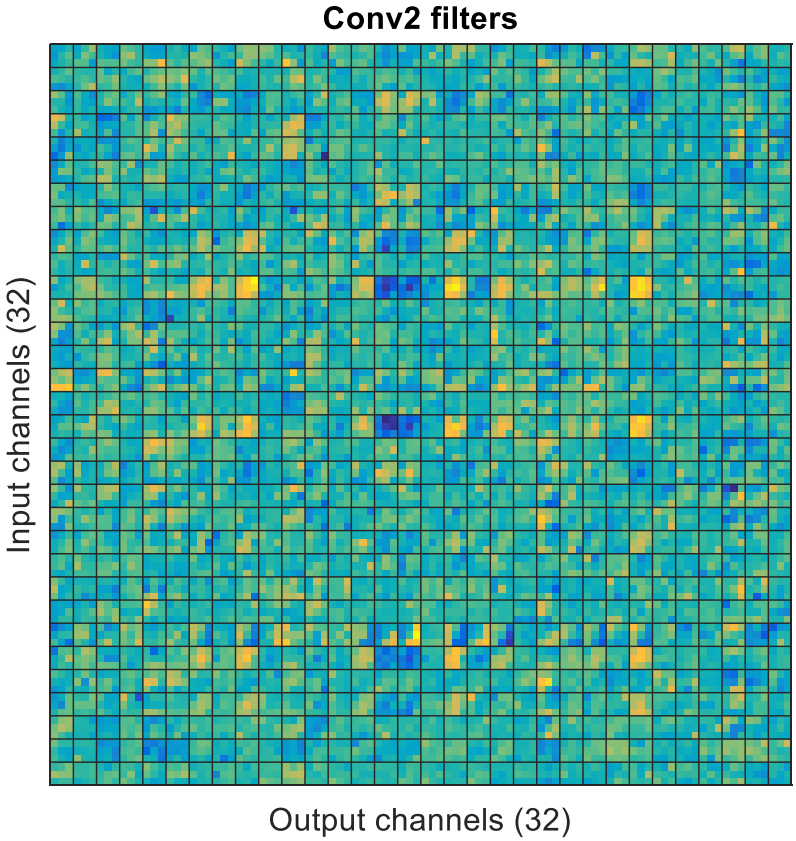


Εικόνα 4.4: Αποτελέσματα εκπαίδευσης με χρήση του αλγόριθμου Adam

Παρακάτω γίνεται απεικόνιση των φίλτρων πρώτου και δεύτερου επιπέδου:



Εικόνα 4.5: τριάντα δύο, διαστάσεων $11 \times 11 \times 1$*, χαμηλού επιπέδου ανιχνευτές χαρακτηριστικών στο πρώτο επίπεδο*



Εικόνα 4.6: τριάντα δύο, διαστάσεων $5 \times 5 \times 32$*, ανιχνευτές που εντοπίζουν χαρακτηριστικά υψηλότερου επιπέδου.*

4.4 Εκπαίδευση των *SVM*

Εφόσον το δίκτυο εκπαιδεύτηκε είμαστε σε θέση να εξάγουμε χαρακτηριστικά από το *exploitation set* για να τα εισάγουμε στον *classifier*. Θα χρησιμοποιήσουμε *SVM* (*libsvm*) τύπου *C-SVM* με *radial basis function kernel* (*RBF kernel*) ο οποίος έχει τροποποιηθεί κατάλληλα για τις ανάγκες της εφαρμογής.

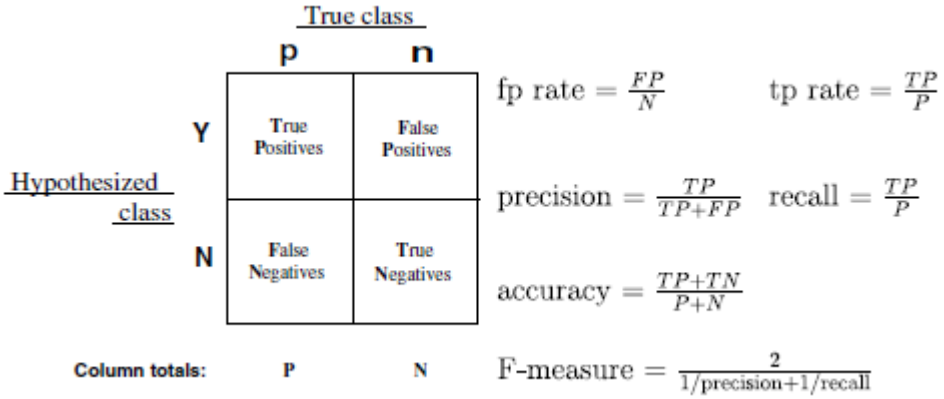
Ο *SVM* κατά τη διάρκεια της εκπαίδευσης επιλέγει από ένα εύρος παραμέτρων $\gamma \in 2^{[-8,5]}$ και $C \in 2^{[-3,10]}$ με βήμα 0.5. Για κάθε ζεύγος παραμέτρων κάνει *cross-validation* στα δεδομένα ώστε να επιλέξει τις βέλτιστες παραμέτρους.

Κατά την εκπαίδευσή τους για κάθε έναν ξεχωριστά, θα χρησιμοποιήσουμε το *exploitation set* "$\mathcal{E}$", δηλαδή, 14 γνήσια δείγματα και 28 τυχαία από το σετ "$\mathcal{D}$". Για το τεστ από την άλλη, 10 γνήσια δείγματα και 10 τυχαία ως *random forgeries* και 24 *skill forgeries*.

4.5 Αποτελέσματα

Για την απεικόνιση του πόσο καλά τα πήγε ο *SVM* κατά τη διάρκεια της εκπαίδευσης εξάγουμε καμπύλες *ROC*, το *false acceptance rate* (*FAR*), το *false rejection rate* (*FRR*) και από αυτά υπολογίζουμε το *equal error rate* (*ERR*).

Οι καμπύλες *ROC* είναι χρήσιμες στην οργάνωση των ταξινομητών (δύο κλάσεων) και στην απεικόνιση της απόδοσής τους. Για να αποκτήσουμε μια διαίσθηση του τρόπου που λειτουργούν ξεκινάμε από τον *confusion matrix* ενός ταξινομητή που έχει μία θετική και μία αρνητική κλάση.



Εικόνα 4.7: Ο confusion matrix και τα συνήθη μεγέθη που μετρούνται

Δίνοντας ένα δείγμα στον ταξινομητή υπάρχουν τέσσερις εκδοχές. Το δείγμα είναι θετικό και ταξινομείται ως θετικό (*TP*), ή αρνητικό(*FN*). Το δείγμα είναι αρνητικό και ταξινομείται ως αρνητικό (*TN*), ή θετικό (*FP*).

Στην περίπτωση των υπογραφών, το μέγεθος *FP* είναι το μέγεθος *FAR* και εκφράζει αυτό που υποδηλώνει το όνομά του, το ποσοστό των δειγμάτων που έγιναν αποδεκτά στο σύστημα (θεωρήθηκαν θετικά ενώ δεν ήταν). Αντίστοιχα το *FRR* είναι το μέγεθος *FN* (ποσοστό των υπογραφών που απορρίφτηκαν ενώ ήταν γνήσιες), και μπορεί αλλιώς να υπολογιστεί ως $FRR = 1 - TP$.

Στην εικόνα τα στοιχεία της κύριας διαγώνιας παριστάνουν τις σωστές αποφάσεις του ταξινομητή και της άλλης διαγώνιας τη σύγχυση (*confusion*).

Επομένως, οι καμπύλες *ROC* είναι δισδιάστατες γραφικές στις οποίες στον άξονα *X* βρίσκεται το *FAR*, ενώ στον άξονα *Y* το *TP* (*true positives*). Άρα παριστάνει μια σχέση μεταξύ του πλεονεκτήματος (*true positives*) και του κόστους (*false positives*). Είναι προφανές ότι τα σημεία πάνω αριστερά της καμπύλης είναι τα πιο επιθυμητά.

Τα σημεία που βρίσκονται πάνω στην κύρια διαγώνιο $y = x$ εκφράζουν τη χειρότερη δυνατή απόδοση ταξινομητή καθώς το ποσοστό των λάθος αντιστοιχισμένων θετικών δειγμάτων είναι ίσο με το ποσοστό των σωστά ταξινομημένων θετικών, δηλαδή έχουμε απόλυτη τυχαιότητα.

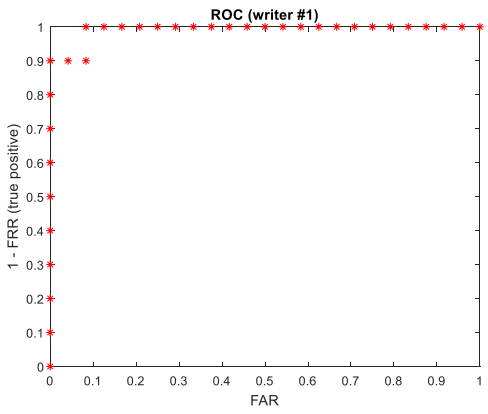
Ο δυαδικός ταξινομητής για να αποφασίσει σε ποια τάξη ανήκει ένα δείγμα παράγει ένα *score*, όταν η τιμή αυτή είναι πάνω από μία τιμή κατωφλίου (*threshold*), αντιστοιχεί το δείγμα στη θετική κλάση διαφορετικά στην αρνητική. Για κάθε διαφορετική τιμή κατωφλίου έχουμε ένα διαφορετικό σημείο στον χώρο *ROC*. Μια συνεχής καμπύλη παράγεται ιδανικά από άπειρα δείγματα. Επειδή στην πράξη τα δείγματα είναι περιορισμένα ουσιαστικά παράγεται μια συνάρτηση βήματος.

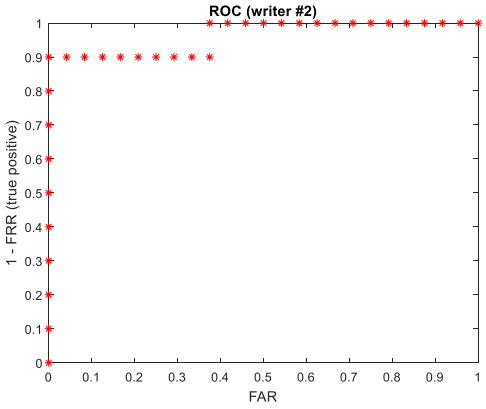
Αναφέραμε ότι οι καμπύλες *ROC* απεικονίζουν την απόδοση του ταξινομητή. Για να συγκρίνουμε όμως την απόδοση ταξινομητών μεταξύ τους είναι χρησιμότερο ένα βαθμωτό μέγεθος παρά μια συνάρτηση. Η πιο κοινή μέθοδος είναι ο υπολογισμός της περιοχής κάτω από την καμπύλη *AUC* (*area under curve*) της οποίας η τιμή κυμαίνεται από το 0 έως το 1. Για τον ταξινομητή του οποίου οι προβλέψεις είναι εντελώς τυχαίες (ευθεία γραμμή) η περιοχή κάτω από την καμπύλη είναι 0.5.[16]

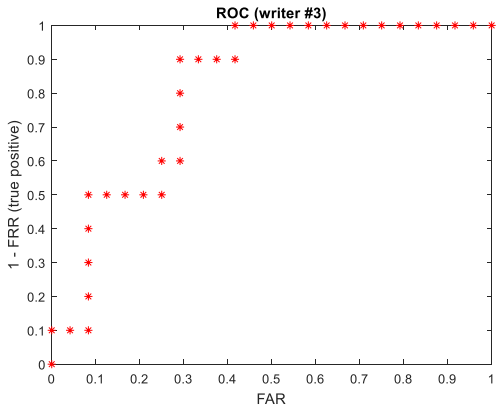
Τα αποτελέσματα χωρίζονται σε δύο κατηγορίες:

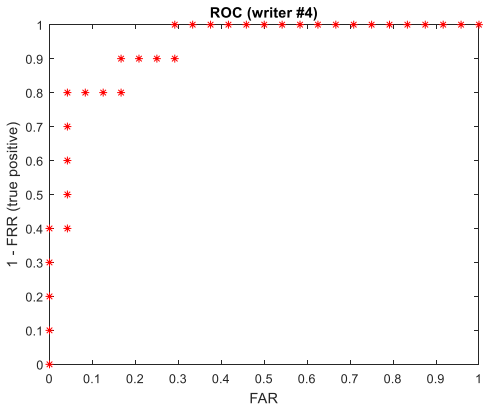
- *global*: αφορούν *random forgeries* και *skill forgeries*
- *skill forgeries*: αφορούν μόνο τις *skill forgeries*

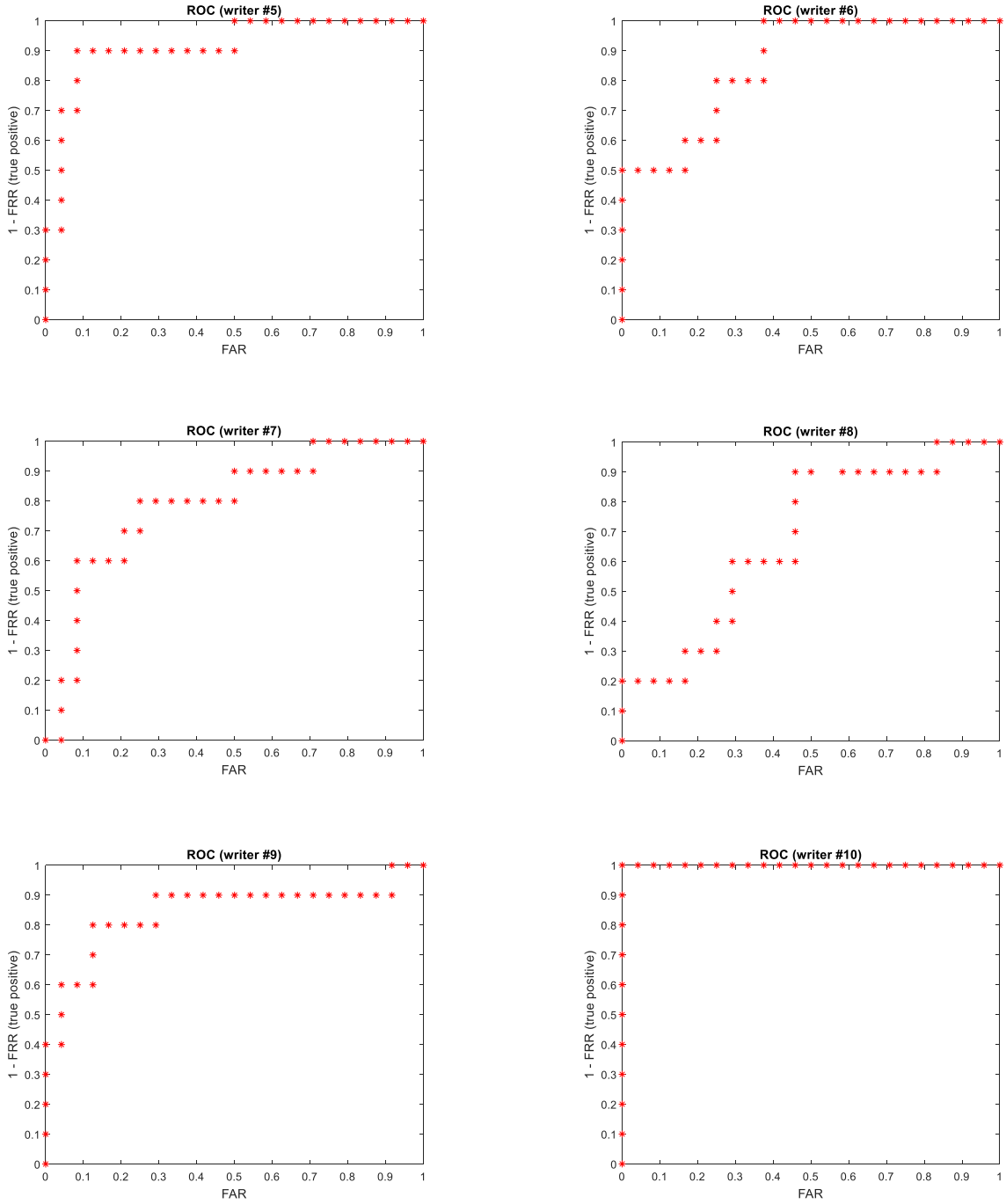
Παρακάτω παρουσιάζονται οι καμπύλες *ROC* των 10 γραφέων γραμμένων στο σύστημα που αφορούν μόνο τις *skill forgeries* καθώς και ο πίνακας με τις τιμές *AUC* υπολογισμένες για *global* και *skill forgery* επίπεδο.











Εικόνα 4.8: Καμπύλες ROC για κάθε έναν από τους 10 γραφείς.

AUC	1	2	3	4	5	6	7	8	9	10
global	0.9941	0.9735	0.8676	0.9559	0.9412	0.9000	0.8529	0.7412	0.8912	1
Sf	0.9917	0.9625	0.8125	0.9375	0.9167	0.8583	0.7917	0.6792	0.8458	1

Καλύτερη απόδοση είχαμε στον γραφέα 10 στον οποίο κάθε δείγμα αντιστοιχήθηκε σωστά, αμέσως επόμενος ήταν ο πρώτος και χειρότερη απόδοση παρατηρήθηκε στον γραφέα 8.

Επίσης μετρήθηκε το *EER* (*equal error rate*) για το οποίο ισχύει $EER = FAR = FRR$. Η τιμή του αφορά τις *skill forgery*.

EER	1	2	3	4	5	6	7	8	9	10
Sf	0.0417	0.0500	0.1958	0.1208	0.0917	0.1875	0.2250	0.2792	0.1625	0

Το μέσο *EER* για όλους τους γραφείς είναι $EER_{mean} = 0.1354$.

Δείγματα υπογραφών των εγγεγραμμένων χρηστών:

Εικόνα 4.9: Οι υπογραφές των 10 εγγεγραμμένων χρηστών στο σύστημα.

Παράρτημα

Στήσιμο του συνελικτικού νευρωνικού δικτύου

```
function net = cnn_cedar_init(varargin)
% CNN_CEDAR_LENET Initialize a CNN similar for CEDAR based upon
cnn_mnist_init
opts.batchNormalization = true ;
opts.networkType = 'simplenn' ;
opts = vl_argparse(opts, varargin) ;

rng('default');
rng(0) ;

f=1/100 ;
net.layers = {} ;
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(11,11,1,32, 'single'),
zeros(1, 32, 'single')}}, ...
                           'stride', 2, ...
                           'pad', 5);
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(5,5,32,32, 'single'),
zeros(1, 32, 'single')}}, ...
                           'stride', 1, ...
                           'pad', 2);
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(3,3,32,64, 'single'),
zeros(1, 64, 'single')}}, ...
                           'stride', 1, ...
                           'pad', 1);
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [2 2], ...
                           'stride', 2);
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(3,3,64,128, 'single'),
zeros(1, 128, 'single')}}, ...
                           'stride', 1, ...
                           'pad', 1);
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [2 2], ...
                           'stride', 2);
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(3,3,128,256, 'single'),
zeros(1, 256, 'single')}}, ...
                           'stride', 1, ...
                           'pad', 1);
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'max', ...
                           'pool', [2 2], ...
                           'stride', 2);
net.layers{end+1} = struct('type', 'conv', ...
                           'weights', {{f*randn(3,3,256,512, 'single'),
zeros(1, 512, 'single')}}, ...
                           'stride', 1, ...
                           'pad', 1);
net.layers{end+1} = struct('type', 'pool', ...
                           'method', 'avg', ...
```

```
                            'pool', [5 7], ...
                            'stride', 5, ...
                            'pad', 0);
%------------------------> FEATURE EXTRACTION <-------------------------
--
net.layers{end+1} = struct('type', 'conv', ...
                            'weights', {{f*randn(1,1,512,45, 'single'),
zeros(1, 45, 'single')}}, ...
                            'stride', 1, ...
                            'pad', 0);
net.layers{end+1} = struct('type', 'softmaxloss');


% optionally switch to batch normalization
if opts.batchNormalization
  net = insertBnorm(net, 1) ;
  net = insertBnorm(net, 4) ;
  net = insertBnorm(net, 7) ;
end

% Meta parameters
net.meta.inputSize = [80 120 1] ;
net.meta.trainOpts.learningRate = 0.0001; %starting lr
net.meta.trainOpts.numEpochs = 75;
net.meta.trainOpts.batchSize = 100 ;

% Fill in defaul values
net = vl simplenn tidy(net) ;

% Switch to DagNN if requested
switch lower(opts.networkType)
  case 'simplenn'
    % done
  case 'dagnn'
    net = dagnn.DagNN.fromSimpleNN(net, 'canonicalNames', true) ;
    net.addLayer('top1err', dagnn.Loss('loss', 'classerror'), ...
      {'prediction', 'label'}, 'error') ;
    net.addLayer('top5err', dagnn.Loss('loss', 'topkerror', ...
      'opts', {'topk', 5}), {'prediction', 'label'}, 'top5err') ;
  otherwise
    assert(false) ;
end

% --------------------------------------------------------------------
function net = insertBnorm(net, l)
% --------------------------------------------------------------------
assert(isfield(net.layers{l}, 'weights'));
ndim = size(net.layers{l}.weights{1}, 4);
layer = struct('type', 'bnorm', ...
               'weights', {{ones(ndim, 1, 'single'), zeros(ndim, 1,
'single')}}, ...
               'learningRate', [1 1 0.05], ...
               'weightDecay', [0 0]) ;
net.layers{l}.weights{2} = [] ;  % eliminate bias in previous conv layer
net.layers = horzcat(net.layers(1:l), layer, net.layers(l+1:end)) ;
```

Αλγόριθμος εκκίνησης εκπαίδευσης

```
function [net, info] = cnn_cedar(varargin)
%CNN_CEDAR  Demonstrates MatConvNet on CEDAR based upon cnn_mnist

run(fullfile(fileparts(mfilename('fullpath')),...
  '..', '..', 'matlab', 'vl_setupnn.m')) ;

opts.batchNormalization = false ;
opts.network = [] ;
opts.networkType = 'simplenn' ;
[opts, varargin] = vl_argparse(opts, varargin) ;

sfx = opts.networkType ;
if opts.batchNormalization, sfx = [sfx '-bnorm'] ; end
opts.expDir = fullfile(vl_rootnn, 'data', ['cedar-adam-' sfx]) ;
[opts, varargin] = vl_argparse(opts, varargin) ;

opts.dataDir = fullfile(vl_rootnn, 'data', 'CEDAR','signatures') ;
opts.imdbPath = fullfile(opts.dataDir, 'development_set.mat');
opts.train = struct() ;
opts = vl_argparse(opts, varargin) ;
if ~isfield(opts.train, 'gpus'), opts.train.gpus = []; end;

% --------------------------------------------------------------------
%                                                         Prepare data
% --------------------------------------------------------------------

if isempty(opts.network)
  net = cnn_cedar_init('batchNormalization', opts.batchNormalization, ...
    'networkType', opts.networkType) ;
else
  net = opts.network ;
  opts.network = [] ;
end

if exist(opts.imdbPath, 'file')
load(opts.imdbPath) ;
else
% do nothing
%  imdb = getMnistImdb(opts) ;
%  mkdir(opts.expDir) ;
%  save(opts.imdbPath, '-struct', 'imdb') ;
end

net.meta.classes.name =
arrayfun(@(x)sprintf('%d',x),1:45,'UniformOutput',false) ;

% --------------------------------------------------------------------
%                                                               Train
% --------------------------------------------------------------------

switch opts.networkType
  case 'simplenn', trainfn = @cnn_train ;
  case 'dagnn', trainfn = @cnn_train_dag ;
end

[net, info] = trainfn(net, development, getBatch(opts), ...
```

```
  'expDir', opts.expDir, ...
  net.meta.trainOpts, ...
  opts.train, ...
  'val', find(development.images.set == 3)) ;

% -------------------------------------------------------------------
%                                                      Visualize filters
% -------------------------------------------------------------------
 figure(2); vl_tshow(net.layers{1}.weights{1}); title('Conv1 filters');
 figure(3); vl_tshow(net.layers{2}.weights{1}); title('Conv2 filters');
 figure(4); vl_tshow(net.layers{3}.weights{1}); title('Conv3 filters');
 figure(5); vl_tshow(net.layers{5}.weights{1}); title('Conv4 filters');
 figure(6); vl_tshow(net.layers{7}.weights{1}); title('Conv5 filters');
 figure(7); vl_tshow(net.layers{9}.weights{1}); title('Conv6 filters');
 figure(8); vl_tshow(net.layers{11}.weights{1}); title('Conv7 filters');

% -------------------------------------------------------------------
function fn = getBatch(opts)
% -------------------------------------------------------------------
switch lower(opts.networkType)
  case 'simplenn'
    fn = @(x,y) getSimpleNNBatch(x,y) ;
  case 'dagnn'
    bopts = struct('numGpus', numel(opts.train.gpus)) ;
    fn = @(x,y) getDagNNBatch(bopts,x,y) ;
end

% -------------------------------------------------------------------
function [images, labels] = getSimpleNNBatch(development, batch)
% -------------------------------------------------------------------
images = development.images.data(:,:,:,batch) ;
labels = development.images.labels(1,batch) ;

% -------------------------------------------------------------------
function inputs = getDagNNBatch(opts, development, batch)
% -------------------------------------------------------------------
images = development.images.data(:,:,:,batch) ;
labels = development.images.labels(1,batch) ;
if opts.numGpus > 0
  images = gpuArray(images) ;
end
inputs = {'input', images, 'label', labels} ;
```

Αλγόριθμος εκπαίδευσης δικτύου

```
function [net, stats] = cnn_train(net, imdb, getBatch, varargin)
%CNN_TRAIN  An example implementation of SGD for training CNNs
%    CNN_TRAIN() is an example learner implementing stochastic
%    gradient descent with momentum to train a CNN. It can be used
%    with different datasets and tasks by providing a suitable
%    getBatch function.
%
%    The function automatically restarts after each training epoch by
%    checkpointing.
%
%    The function supports training on CPU or on one or more GPUs
%    (specify the list of GPU IDs in the `gpus` option).

% Copyright (C) 2014-16 Andrea Vedaldi.
% All rights reserved.
%
% This file is part of the VLFeat library and is made available under
% the terms of the BSD license (see the COPYING file).
addpath(fullfile(vl_rootnn, 'examples'));

opts.expDir = fullfile(vl_rootnn, 'data','CEDAR-adam-') ;
opts.continue = true ;
opts.batchSize = 200 ;
opts.numSubBatches = 1 ;
opts.train = [] ;
opts.val = [] ;
opts.gpus = [] ;
opts.epochSize = inf;
opts.prefetch = false ;
opts.numEpochs = 30 ;
opts.learningRate = 0.001 ;
opts.weightDecay = 0.0005 ;

opts.solver = @adam;  % Empty array means use the default SGD solver
[opts, varargin] = vl_argparse(opts, varargin) ;
if ~isempty(opts.solver)
  assert(isa(opts.solver, 'function_handle') && nargout(opts.solver) ==
2,...
    'Invalid solver; expected a function handle with two outputs.') ;
  % Call without input arguments, to get default options
  opts.solverOpts = opts.solver() ;
end

opts.momentum = 0.3 ;
opts.saveSolverState = true ;
opts.nesterovUpdate = false ;
opts.randomSeed = 0 ;
opts.memoryMapFile = fullfile(tempdir, 'matconvnet.bin') ;
opts.profile = false ;
opts.parameterServer.method = 'mmap' ;
opts.parameterServer.prefix = 'mcn' ;

opts.conserveMemory = true ;
opts.backPropDepth = +inf ;
opts.sync = false ;
opts.cudnn = true ;
opts.errorFunction = 'multiclass' ;
```

```
opts.errorLabels = {} ;
opts.plotDiagnostics = false ;
opts.plotStatistics = true;
opts.postEpochFn = [];  % postEpochFn(net,params,state) called after each
epoch; can return a new learning rate, 0 to stop, [] for no change
opts = vl_argparse(opts, varargin) ;

if ~exist(opts.expDir, 'dir'), mkdir(opts.expDir) ; end
if isempty(opts.train), opts.train = find(imdb.images.set==1) ; end
if isempty(opts.val), opts.val = find(imdb.images.set==2) ; end
if isscalar(opts.train) && isnumeric(opts.train) && isnan(opts.train)
  opts.train = [] ;
end
if isscalar(opts.val) && isnumeric(opts.val) && isnan(opts.val)
  opts.val = [] ;
end

% -------------------------------------------------------------------------
                                                              Initialization
% -------------------------------------------------------------------------

net = vl_simplenn_tidy(net); % fill in some eventually missing values
net.layers{end-1}.precious = 1; % do not remove predictions, used for
error
vl_simplenn_display(net, 'batchSize', opts.batchSize) ;

evaluateMode = isempty(opts.train) ;
if ~evaluateMode
  for i=1:numel(net.layers)
    J = numel(net.layers{i}.weights) ;
    if ~isfield(net.layers{i}, 'learningRate')
      net.layers{i}.learningRate = ones(1, J) ;
    end
    if ~isfield(net.layers{i}, 'weightDecay')
      net.layers{i}.weightDecay = ones(1, J) ;
    end
  end
end

% setup error calculation function
hasError = true ;
if isstr(opts.errorFunction)
  switch opts.errorFunction
    case 'none'
      opts.errorFunction = @error_none ;
      hasError = false ;
    case 'multiclass'
      opts.errorFunction = @error_multiclass ;
      if isempty(opts.errorLabels), opts.errorLabels = {'top1err',
'top5err'} ; end
    case 'binary'
      opts.errorFunction = @error_binary ;
      if isempty(opts.errorLabels), opts.errorLabels = {'binerr'} ; end
    otherwise
      error('Unknown error function ''%s''.', opts.errorFunction) ;
  end
end

state.getBatch = getBatch ;
stats = [] ;
```

```
% -------------------------------------------------------------------------
%                                                              Train and Validate
% -------------------------------------------------------------------------

modelPath = @(ep) fullfile(opts.expDir, sprintf('net-epoch-%d.mat', ep));
modelFigPath = fullfile(opts.expDir, 'net-train.pdf') ;

start = opts.continue * findLastCheckpoint(opts.expDir) ;
if start >= 1
  fprintf('%s: resuming by loading epoch %d\n', mfilename, start) ;
  [net, state, stats] = loadState(modelPath(start)) ;
else
  state = [] ;
end

for epoch=start+1:opts.numEpochs

  % Set the random seed based on the epoch and opts.randomSeed.
  % This is important for reproducibility, including when training
  % is restarted from a checkpoint.

  rng(epoch + opts.randomSeed) ;
  prepareGPUs(opts, epoch == start+1) ;

  % Train for one epoch.
  params = opts ;
  params.epoch = epoch ;
  params.learningRate = opts.learningRate(min(epoch,
numel(opts.learningRate))) ;
  params.train = opts.train(randperm(numel(opts.train))) ; % shuffle
  params.train = params.train(1:min(opts.epochSize, numel(opts.train)));
  params.val = opts.val(randperm(numel(opts.val))) ;
  params.imdb = imdb ;
  params.getBatch = getBatch ;

  if numel(params.gpus) <= 1
    [net, state] = processEpoch(net, state, params, 'train') ;
    [net, state] = processEpoch(net, state, params, 'val') ;
    if ~evaluateMode
      saveState(modelPath(epoch), net, state) ;
    end
    lastStats = state.stats ;
  else
    spmd
      [net, state] = processEpoch(net, state, params, 'train') ;
      [net, state] = processEpoch(net, state, params, 'val') ;
      if labindex == 1 && ~evaluateMode
        saveState(modelPath(epoch), net, state) ;
      end
      lastStats = state.stats ;
    end
    lastStats = accumulateStats(lastStats) ;
  end

  stats.train(epoch) = lastStats.train ;
  stats.val(epoch) = lastStats.val ;
  clear lastStats ;
  if ~evaluateMode
```

```
    saveStats(modelPath(epoch), stats) ;
  end

  if params.plotStatistics
    switchFigure(1) ; clf ;
    plots = setdiff(...
      cat(2,...
      fieldnames(stats.train)', ...
      fieldnames(stats.val)'), {'num', 'time'}) ;
    for p = plots
      p = char(p) ;
      values = zeros(0, epoch) ;
      leg = {} ;
      for f = {'train', 'val'}
        f = char(f) ;
        if isfield(stats.(f), p)
          tmp = [stats.(f).(p)] ;
          values(end+1,:) = tmp(1,:)' ;
          leg{end+1} = f ;
        end
      end
      subplot(1,numel(plots),find(strcmp(p,plots))) ;
      plot(1:epoch, values','-') ;
      xlabel('epoch') ;
      title(p) ;
      legend(leg{:}) ;
      grid on ;
    end
    drawnow ;
    print(1, modelFigPath, '-dpdf') ;
  end

  if ~isempty(opts.postEpochFn)
    if nargout(opts.postEpochFn) == 0
      opts.postEpochFn(net, params, state) ;
    else
      lr = opts.postEpochFn(net, params, state) ;
      if ~isempty(lr), opts.learningRate = lr; end
      if opts.learningRate == 0, break; end
    end
  end
end

% With multiple GPUs, return one copy
if isa(net, 'Composite'), net = net{1} ; end

% -------------------------------------------------------------------------
--
function err = error_multiclass(params, labels, res)
% -------------------------------------------------------------------------
--
predictions = gather(res(end-1).x) ;
[~,predictions] = sort(predictions, 3, 'descend') ;

% be resilient to badly formatted labels
if numel(labels) == size(predictions, 4)
  labels = reshape(labels,1,1,1,[]) ;
end

% skip null labels
```

```
mass = single(labels(:,:,1,:) > 0) ;
if size(labels,3) == 2
  % if there is a second channel in labels, used it as weights
  mass = mass .* labels(:,:,2,:) ;
  labels(:,:,2,:) = [] ;
end

m = min(5, size(predictions,3)) ;

error = ~bsxfun(@eq, predictions, labels) ;
err(1,1) = sum(sum(sum(mass .* error(:,:,1,:)))) ;
err(2,1) = sum(sum(sum(mass .* min(error(:,:,1:m,:),[],3)))) ;

% -------------------------------------------------------------------------
function err = error_binary(params, labels, res)
% -------------------------------------------------------------------------
predictions = gather(res(end-1).x) ;
error = bsxfun(@times, predictions, labels) < 0 ;
err = sum(error(:)) ;

% -------------------------------------------------------------------------
function err = error_none(params, labels, res)
% -------------------------------------------------------------------------
err = zeros(0,1) ;

% -------------------------------------------------------------------------
function [net, state] = processEpoch(net, state, params, mode)
% -------------------------------------------------------------------------
% Note that net is not strictly needed as an output argument as net
% is a handle class. However, this fixes some aliasing issue in the
% spmd caller.

% initialize with momentum 0
if isempty(state) || isempty(state.solverState)
  for i = 1:numel(net.layers)
    state.solverState{i} = cell(1, numel(net.layers{i}.weights)) ;
    state.solverState{i}(:) = {0} ;
  end
end

% move CNN  to GPU as needed
numGpus = numel(params.gpus) ;
if numGpus >= 1
  net = vl_simplenn_move(net, 'gpu') ;
  for i = 1:numel(state.solverState)
    for j = 1:numel(state.solverState{i})
      s = state.solverState{i}{j} ;
      if isnumeric(s)
        state.solverState{i}{j} = gpuArray(s) ;
      elseif isstruct(s)
        state.solverState{i}{j} = structfun(@gpuArray, s,
'UniformOutput', false) ;
      end
    end
  end
end
if numGpus > 1
  parserv = ParameterServer(params.parameterServer) ;
  vl_simplenn_start_parserv(net, parserv) ;
else
```

```
  parserv = [] ;
end

% profile
if params.profile
  if numGpus <= 1
    profile clear ;
    profile on ;
  else
    mpiprofile reset ;
    mpiprofile on ;
  end
end

subset = params.(mode) ;
num = 0 ;
stats.num = 0 ; % return something even if subset = []
stats.time = 0 ;
adjustTime = 0 ;
res = [] ;
error = [] ;

start = tic ;
for t=1:params.batchSize:numel(subset)
  fprintf('%s: epoch %02d: %3d/%3d:', mode, params.epoch, ...
          fix((t-1)/params.batchSize)+1,
ceil(numel(subset)/params.batchSize)) ;
  batchSize = min(params.batchSize, numel(subset) - t + 1) ;

  for s=1:params.numSubBatches
    % get this image batch and prefetch the next
    batchStart = t + (labindex-1) + (s-1) * numlabs ;
    batchEnd = min(t+params.batchSize-1, numel(subset)) ;
    batch = subset(batchStart : params.numSubBatches * numlabs :
batchEnd) ;
    num = num + numel(batch) ;
    if numel(batch) == 0, continue ; end

    [im, labels] = params.getBatch(params.imdb, batch) ;

    if params.prefetch
      if s == params.numSubBatches
        batchStart = t + (labindex-1) + params.batchSize ;
        batchEnd = min(t+2*params.batchSize-1, numel(subset)) ;
      else
        batchStart = batchStart + numlabs ;
      end
      nextBatch = subset(batchStart : params.numSubBatches * numlabs :
batchEnd) ;
      params.getBatch(params.imdb, nextBatch) ;
    end

    if numGpus >= 1
      im = gpuArray(im) ;
    end

    if strcmp(mode, 'train')
      dzdy = 1 ;
      evalMode = 'normal' ;
```

```
    else
      dzdy = [] ;
      evalMode = 'test' ;
    end
    net.layers{end}.class = labels ;
    res = vl_simplenn(net, im, dzdy, res, ...
                      'accumulate', s ~= 1, ...
                      'mode', evalMode, ...
                      'conserveMemory', params.conserveMemory, ...
                      'backPropDepth', params.backPropDepth, ...
                      'sync', params.sync, ...
                      'cudnn', params.cudnn, ...
                      'parameterServer', parserv, ...
                      'holdOn', s < params.numSubBatches) ;

    % accumulate errors
    error = sum([error, [...
      sum(double(gather(res(end).x))) ;
      reshape(params.errorFunction(params, labels, res),[],1) ; ]],2) ;
  end

  % accumulate gradient
  if strcmp(mode, 'train')
    if ~isempty(parserv), parserv.sync() ; end
    [net, res, state] = accumulateGradients(net, res, state, params,
batchSize, parserv) ;
  end

  % get statistics
  time = toc(start) + adjustTime ;
  batchTime = time - stats.time ;
  stats = extractStats(net, params, error / num) ;
  stats.num = num ;
  stats.time = time ;
  currentSpeed = batchSize / batchTime ;
  averageSpeed = (t + batchSize - 1) / time ;
  if t == 3*params.batchSize + 1
    % compensate for the first three iterations, which are outliers
    adjustTime = 4*batchTime - time ;
    stats.time = time + adjustTime ;
  end

  fprintf(' %.1f (%.1f) Hz', averageSpeed, currentSpeed) ;
  for f = setdiff(fieldnames(stats)', {'num', 'time'})
    f = char(f) ;
    fprintf(' %s: %.3f', f, stats.(f)) ;
  end
  fprintf('\n') ;

  % collect diagnostic statistics
  if strcmp(mode, 'train') && params.plotDiagnostics
    switchFigure(2) ; clf ;
    diagn = [res.stats] ;
    diagnvar = horzcat(diagn.variation) ;
    diagnpow = horzcat(diagn.power) ;
    subplot(2,2,1) ; barh(diagnvar) ;
    set(gca,'TickLabelInterpreter', 'none', ...
      'YTick', 1:numel(diagnvar), ...
      'YTickLabel',horzcat(diagn.label), ...
      'YDir', 'reverse', ...
```

```
      'XScale', 'log', ...
      'XLim', [1e-5 1], ...
      'XTick', 10.^(-5:1)) ;
    grid on ; title('Variation');
    subplot(2,2,2) ; barh(sqrt(diagnpow)) ;
    set(gca,'TickLabelInterpreter', 'none', ...
      'YTick', 1:numel(diagnpow), ...
      'YTickLabel',{diagn.powerLabel}, ...
      'YDir', 'reverse', ...
      'XScale', 'log', ...
      'XLim', [1e-5 1e5], ...
      'XTick', 10.^(-5:5)) ;
    grid on ; title('Power');
    subplot(2,2,3); plot(squeeze(res(end-1).x)) ;
    drawnow ;
  end
end

% Save back to state.
state.stats.(mode) = stats ;
if params.profile
  if numGpus <= 1
    state.prof.(mode) = profile('info') ;
    profile off ;
  else
    state.prof.(mode) = mpiprofile('info');
    mpiprofile off ;
  end
end
if ~params.saveSolverState
  state.solverState = [] ;
else
  for i = 1:numel(state.solverState)
    for j = 1:numel(state.solverState{i})
      s = state.solverState{i}{j} ;
      if isnumeric(s)
        state.solverState{i}{j} = gather(s) ;
      elseif isstruct(s)
        state.solverState{i}{j} = structfun(@gather, s, 'UniformOutput',
false) ;
      end
    end
  end
end

net = vl_simplenn_move(net, 'cpu') ;

% -------------------------------------------------------------------------
function [net, res, state] = accumulateGradients(net, res, state, params,
batchSize, parserv)
% -------------------------------------------------------------------------
numGpus = numel(params.gpus) ;
otherGpus = setdiff(1:numGpus, labindex) ;

for l=numel(net.layers):-1:1
  for j=numel(res(l).dzdw):-1:1

    if ~isempty(parserv)
      tag = sprintf('l%d_%d',l,j) ;
      parDer = parserv.pull(tag) ;
```

```
    else
      parDer = res(l).dzdw{j}  ;
    end

    if j == 3 && strcmp(net.layers{l}.type, 'bnorm')
      % special case for learning bnorm moments
      thisLR = net.layers{l}.learningRate(j) ;
      net.layers{l}.weights{j} = vl_taccum(...
        1 - thisLR, ...
        net.layers{l}.weights{j}, ...
        thisLR / batchSize, ...
        parDer) ;
    else
      % Standard gradient training.
      thisDecay = params.weightDecay * net.layers{l}.weightDecay(j) ;
      thisLR = params.learningRate * net.layers{l}.learningRate(j) ;

      if thisLR>0 || thisDecay>0
        % Normalize gradient and incorporate weight decay.
        parDer = vl_taccum(1/batchSize, parDer, ...
                           thisDecay, net.layers{l}.weights{j}) ;

        if isempty(params.solver)
          % Default solver is the optimised SGD.
          % Update momentum.
          state.solverState{l}{j} = vl_taccum(...
            params.momentum, state.solverState{l}{j}, ...
            -1, parDer) ;

          % Nesterov update (aka one step ahead).
          if params.nesterovUpdate
            delta = params.momentum * state.solverState{l}{j} - parDer ;
          else
            delta = state.solverState{l}{j} ;
          end

          % Update parameters.
          net.layers{l}.weights{j} = vl_taccum(...
            1, net.layers{l}.weights{j}, ...
            thisLR, delta) ;

        else
          % call solver function to update weights
          [net.layers{l}.weights{j}, state.solverState{l}{j}] = ...
            params.solver(net.layers{l}.weights{j},
state.solverState{l}{j}, ...
            parDer, params.solverOpts, thisLR) ;
        end
      end
    end

    % if requested, collect some useful stats for debugging
    if params.plotDiagnostics
      variation = [] ;
      label = '' ;
      switch net.layers{l}.type
        case {'conv','convt'}
          if isnumeric(state.solverState{l}{j})
            variation = thisLR * mean(abs(state.solverState{l}{j}(:))) ;
          end
```

```
          power = mean(res(l+1).x(:).^2) ;
          if j == 1 % fiters
            base = mean(net.layers{l}.weights{j}(:).^2) ;
            label = 'filters' ;
          else % biases
            base = sqrt(power) ;%mean(abs(res(l+1).x(:))) ;
            label = 'biases' ;
          end
          variation = variation / base ;
          label = sprintf('%s_%s', net.layers{l}.name, label) ;
      end
      res(l).stats.variation(j) = variation ;
      res(l).stats.power = power ;
      res(l).stats.powerLabel = net.layers{l}.name ;
      res(l).stats.label{j} = label ;
    end
  end
end

% -------------------------------------------------------------------------
function stats = accumulateStats(stats_)
% -------------------------------------------------------------------------

for s = {'train', 'val'}
  s = char(s) ;
  total = 0 ;

  % initialize stats stucture with same fields and same order as
  % stats_{1}
  stats__ = stats_{1} ;
  names = fieldnames(stats__.(s))' ;
  values = zeros(1, numel(names)) ;
  fields = cat(1, names, num2cell(values)) ;
  stats.(s) = struct(fields{:}) ;

  for g = 1:numel(stats_)
    stats__ = stats_{g} ;
    num__ = stats__.(s).num ;
    total = total + num__ ;

    for f = setdiff(fieldnames(stats__.(s))', 'num')
      f = char(f) ;
      stats.(s).(f) = stats.(s).(f) + stats__.(s).(f) * num__ ;

      if g == numel(stats_)
        stats.(s).(f) = stats.(s).(f) / total ;
      end
    end
  end
  stats.(s).num = total ;
end

% -------------------------------------------------------------------------
--
function stats = extractStats(net, params, errors)
% -------------------------------------------------------------------------
--
stats.objective = errors(1) ;
for i = 1:numel(params.errorLabels)
  stats.(params.errorLabels{i}) = errors(i+1) ;
```

```
end

% -------------------------------------------------------------------------
function saveState(fileName, net, state)
% -------------------------------------------------------------------------
save(fileName, 'net', 'state') ;

% -------------------------------------------------------------------------
function saveStats(fileName, stats)
% -------------------------------------------------------------------------
if exist(fileName)
  save(fileName, 'stats', '-append') ;
else
  save(fileName, 'stats') ;
end

% -------------------------------------------------------------------------
function [net, state, stats] = loadState(fileName)
% -------------------------------------------------------------------------
load(fileName, 'net', 'state', 'stats') ;
net = vl_simplenn_tidy(net) ;
if isempty(whos('stats'))
  error('Epoch ''%s'' was only partially saved. Delete this file and try
again.', ...
        fileName) ;
end

% -------------------------------------------------------------------------
function epoch = findLastCheckpoint(modelDir)
% -------------------------------------------------------------------------
list = dir(fullfile(modelDir, 'net-epoch-*.mat')) ;
tokens = regexp({list.name}, 'net-epoch-([\d]+).mat', 'tokens') ;
epoch = cellfun(@(x) sscanf(x{1}{1}, '%d'), tokens) ;
epoch = max([epoch 0]) ;

% -------------------------------------------------------------------------
function switchFigure(n)
% -------------------------------------------------------------------------
if get(0,'CurrentFigure') ~= n
  try
    set(0,'CurrentFigure',n) ;
  catch
    figure(n) ;
  end
end
% -------------------------------------------------------------------------
function clearMex()
% -------------------------------------------------------------------------
%clear vl_tmove vl_imreadjpeg ;
disp('Clearing mex files') ;
clear mex ;
clear vl_tmove vl_imreadjpeg ;

% -------------------------------------------------------------------------
function prepareGPUs(params, cold)
% -------------------------------------------------------------------------
numGpus = numel(params.gpus) ;
if numGpus > 1
  % check parallel pool integrity as it could have timed out
  pool = gcp('nocreate') ;
```

```
  if ~isempty(pool) && pool.NumWorkers ~= numGpus
    delete(pool) ;
  end
  pool = gcp('nocreate') ;
  if isempty(pool)
    parpool('local', numGpus) ;
    cold = true ;
  end
end
if numGpus >= 1 && cold
  fprintf('%s: resetting GPU\n', mfilename) ;
  clearMex() ;
  if numGpus == 1
    disp(gpuDevice(params.gpus)) ;
  else
    spmd
      clearMex() ;
      disp(gpuDevice(params.gpus(labindex))) ;
    end
  end
end
```

Αλγόριθμος *Adam* (*gradient descent optimizer*)

```
function [w, state] = adam(w, state, grad, opts, lr)
%ADAM
%   Adam solver for use with CNN_TRAIN and CNN_TRAIN_DAG
%
%   See [Kingma et. al., 2014](http://arxiv.org/abs/1412.6980)
%    |  ([pdf](http://arxiv.org/pdf/1412.6980.pdf)).
%
%   If called without any input argument, returns the default options
%   structure. Otherwise provide all input arguments.
%
%   W is the vector/matrix/tensor of parameters. It can be single/double
%   precision and can be a `gpuArray`.
%
%   STATE is as defined below and so are supported OPTS.
%
%   GRAD is the gradient of the objective w.r.t W
%
%   LR is the learning rate, referred to as \alpha by Algorithm 1 in
%   [Kingma et. al., 2014].
%
%   Solver options: (opts.train.solverOpts)
%
%   `beta1`:: 0.9
%      Decay for 1st moment vector. See algorithm 1 in [Kingma et.al.
2014]
%
%   `beta2`:: 0.999
%      Decay for 2nd moment vector
%
%   `eps`:: 1e-8
%      Additive offset when dividing by state.v
%
%   The state is initialized as 0 (number) to start with. The first call
to
%   this function will initialize it with the default state consisting of
%
%   `m`:: 0
%      First moment vector
%
%   `v`:: 0
%      Second moment vector
%
%   `t`:: 0
%      Global iteration number across epochs
%
%   This implementation borrowed from torch optim.adam

% Copyright (C) 2016 Aravindh Mahendran.
% All rights reserved.
%
% This file is part of the VLFeat library and is made available under
% the terms of the BSD license (see the COPYING file).

if nargin == 0 % Returns the default solver options
  w = struct('beta1', 0.9, 'beta2', 0.999, 'eps', 1e-8) ;
  return ;
end
```

```
if isequal(state, 0) % start off with state = 0 so as to get default
state
  state = struct('m', 0, 'v', 0, 't', 0);
end

% update first moment vector `m`
state.m = opts.beta1 * state.m + (1 - opts.beta1) * grad ;

% update second moment vector `v`
state.v = opts.beta2 * state.v + (1 - opts.beta2) * grad.^2 ;

% update the time step
state.t = state.t + 1 ;

% This implicitly corrects for biased estimates of first and second
moment
% vectors
lr_t = lr * (((1 - opts.beta2^state.t)^0.5) / (1 - opts.beta1^state.t)) ;

% Update `w`
w = w - lr_t * state.m ./ (state.v.^0.5 + opts.eps) ;
```

Τοπολογία του συνελικτικού νευρωνικού δικτύου.

layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
type	input	conv	relu	conv	relu	conv	relu	mpool	conv	relu	mpool	conv	relu	mpool	conv	apool	conv	softmxl
name	n/a	layer1	layer2	layer3	layer4	layer5	layer6	layer7	layer8	layer9	layer10	layer11	layer12	layer13	layer14	layer15	layer16	layer17
support	n/a	5	1	3	1	3	1	2	3	1	2	3	1	2	3	5x7	1	1
filt dim	n/a	1	n/a	32	n/a	32	n/a	n/a	64	n/a	n/a	128	n/a	n/a	256	n/a	512	n/a
filt dilat	n/a	1	n/a	1	n/a	1	n/a	n/a	1	n/a	n/a	1	n/a	n/a	1	n/a	1	n/a
num filts	n/a	32	n/a	32	n/a	64	n/a	n/a	128	n/a	n/a	256	n/a	n/a	512	n/a	45	n/a
stride	n/a	2	1	1	1	1	1	2	1	1	2	1	1	2	1	5	1	1
pad	n/a	2	0	1	0	1	0	0	1	0	0	1	0	0	1	0	0	0
rf size	n/a	5	5	9	9	13	13	15	23	23	27	43	43	51	83	147x179	147x179	147x179
rf offset	n/a	1	1	1	1	1	1	2	2	2	4	4	4	8	8	40x56	40x56	40x56
rf stride	n/a	2	2	2	2	2	2	4	4	4	8	8	8	16	16	80	80	80
data size	80x120	40x60	40x60	40x60	40x60	40x60	40x60	20x30	20x30	20x30	10x15	10x15	10x15	5x7	5x7	1	1	1
data depth	1	32	32	32	32	64	64	64	128	128	128	256	256	256	512	512	45	1
data num	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	1
data mem	4MB	29MB	29MB	29MB	29MB	59MB	59MB	15MB	29MB	29MB	7MB	15MB	15MB	3MB	7MB	200KB	18KB	4B
param mem	n/a	3KB	0B	36KB	0B	72KB	0B	0B	289KB	0B	0B	1MB	0B	0B	5MB	0B	90KB	0B

Περικοπή περιττής πληροφορίας από τις εικόνες στη βάση *CEDAR*

```
% This script crops the excess information from cedar database
% after this, images can directly insert to matlab.
% Need to execute once

sig_path = strcat(vl_rootnn,'\data\CEDAR\signatures\');
dbname = 'CEDAR';
thin_level = 1;
writer_num = 55;
sig_num = 24;

%% image_cropping & thinning & complement
for i=1:writer_num
    for j=1:sig_num
        imagepath = strcat(sig_path,'full_forg');
        [~,cropped_imgray{i,j}] = ...
            image_prepr_crop(dbname,imagepath,'false',i,j,thin_level);
    end
end
```

```
function
[cropped_image,thin_imagegray1]=image_prepr_crop(dbname,image_path,Origin
ality, writer_id, sid, thin_level)

% SOS implementation with MIN rectangle
if strcmp(dbname,'CEDAR')
    if strcmp(Originality,'original')
        sign_file=[image_path '\original_' num2str(writer_id) '_'
num2str(sid) '.png'];
    elseif strcmp(Originality,'false')
        sign_file=[image_path '\forgeries_' num2str(writer_id) '_'
num2str(sid) '.png'];
    end
elseif strcmp(dbname,'MCYT')
    if strcmp(Originality,'original')
        d=dir([image_path '\writer_' num2str(writer_id) '\*.bmp']);
        sign_file=[image_path 'writer_' num2str(writer_id) '\'
d(sid).name];
    elseif strcmp(Originality,'false')
        d=dir([image_path '\writer_' num2str(writer_id) '\*.bmp']);
        sign_file=[image_path '\writer_' num2str(writer_id) '\'
d(sid).name];
    end
elseif strcmp(dbname,'GPDS_synthetic')
    if strcmp(Originality,'original')
        %d=dir([image_path '\writer_' num2str(writer_id) '\*.bmp']);
        if writer_id<10
            if sid<10
                sign_file=[image_path '00' num2str(writer_id) '\c-00'
num2str(writer_id) '-0' num2str(sid) '.jpg'];
            else
                sign_file=[image_path '00' num2str(writer_id) '\c-00'
num2str(writer_id) '-' num2str(sid) '.jpg'];
            end
        elseif writer_id<100
            if sid<10
                sign_file=[image_path '0' num2str(writer_id) '\c-0'
num2str(writer_id) '-0' num2str(sid) '.jpg'];
```

```
            else
                sign_file=[image_path '0' num2str(writer_id) '\c-0'
num2str(writer_id) '-' num2str(sid) '.jpg'];
            end
        else
            if sid<10
                sign_file=[image_path num2str(writer_id) '\c-'
num2str(writer_id) '-0' num2str(sid) '.jpg'];
            else
                sign_file=[image_path num2str(writer_id) '\c-'
num2str(writer_id) '-' num2str(sid) '.jpg'];
            end
        end
    elseif strcmp(Originality,'false')
        if writer_id<10
            if sid<10
                sign_file=[image_path '00' num2str(writer_id) '\cf-00'
num2str(writer_id) '-0' num2str(sid) '.jpg'];
            else
                sign_file=[image_path '00' num2str(writer_id) '\cf-00'
num2str(writer_id) '-' num2str(sid) '.jpg'];
            end
        elseif writer_id<100
            if sid<10
                sign_file=[image_path '0' num2str(writer_id) '\cf-0'
num2str(writer_id) '-0' num2str(sid) '.jpg'];
            else
                sign_file=[image_path '0' num2str(writer_id) '\cf-0'
num2str(writer_id) '-' num2str(sid) '.jpg'];
            end
        else
            if sid<10
                sign_file=[image_path  num2str(writer_id) '\cf-'
num2str(writer_id) '-0' num2str(sid) '.jpg'];
            else
                sign_file=[image_path  num2str(writer_id) '\cf-'
num2str(writer_id) '-' num2str(sid) '.jpg' ];
            end
        end
    end
elseif strcmp(dbname,'GPDS300')
    if strcmp(Originality,'original')
        sign_file=[image_path 'writer' num2str(writer_id) '\simg'
num2str(sid) '.bmp'];
    elseif strcmp(Originality,'false')
        sign_file=[image_path 'writer' num2str(writer_id) '\simg'
num2str(sid) '.bmp'];
    end
elseif strcmp(dbname,'UTSig')
    if strcmp(Originality,'original')
        sign_file=[image_path num2str(writer_id) '\' num2str(sid)
'.tif'];
    elseif strcmp(Originality,'false_1')
        sign_file=[image_path num2str(writer_id) '\' num2str(sid)
'.tif'];
    elseif strcmp(Originality,'false_3')
        sign_file=[image_path num2str(writer_id) '\' num2str(sid)
'.tif'];
    elseif strcmp(Originality,'false_2')
        d=dir([image_path num2str(writer_id) '\*.tif']);
        sign_file=[image_path num2str(writer_id) '\' d(sid).name];
```

```
        %sign_file=[image_path num2str(writer_id) '\' num2str(sid)
'.tif'];

    end
end
simage=imread(sign_file);

%figure, subplot(2, 2, 1), imshow(simage), title('original image')

bw_level=graythresh(simage);
bw_image=im2bw(simage,bw_level);

%subplot(2, 2, 2), imshow(bw_image), title('black and white image')


thin_image = bwmorph(~bw_image, 'thin', thin_level);

thin_image = ~thin_image;
[x,y]=find(thin_image==0);
thin_imagegray1=uint8(255*ones(size(simage)));% gray level information
for i=1:length(x)
    thin_imagegray1(x(i),y(i))=simage(x(i),y(i));
end
%subplot(2, 2, 3), imshow(thin_image), title('thinned bw image')

% Crop the  thinned image
% ------------------------------------------------------------------------
---------------------
[~,x]=find(~thin_image);

% find the extreme x values and try to cut the outliers
% x in ascending order
xs=sort(x);

outlier_distance=3;
while xs(1)<(xs(2)-outlier_distance)
    xs(1)=[];
end
while xs(end)>(xs(end-1)+outlier_distance)
    xs(end)=[];
end

xmin=xs(1)-1;
xmax=xs(end)+1;

% crop the image
cropped_image=thin_image;
cropped_image(:,xmax:end)=[];
cropped_image(:,1:xmin)=[];

thin_imagegray1(:,xmax:end)=[];
thin_imagegray1(:,1:xmin)=[];

[y,~]=find(~cropped_image);
% y in ascending order
ys=sort(y);

while ys(1)<(ys(2)-outlier_distance)
```

```
    ys(1)=[];
end
while ys(end)>(ys(end-1)+outlier_distance)
    ys(end)=[];
end

ymin=ys(1)-1;
ymax=ys(end)+1;

cropped_image(ymax:end, :)=[];
cropped_image(1:ymin,:)=[];
thin_imagegray1(ymax:end, :)=[];
thin_imagegray1(1:ymin,:)=[];
%subplot(2, 2, 4), imshow(cropped_image), title('cropped image')
```

Data preparation και δημιουργία *Development set*

```
% This script prepares cropped data for training
% performing the following tasks:
% 1)complement
% 2)resize(80x120)
% 3)data_augmentation in the last 45 writers
% 4)Development set creation
% 5)Setting: train,val
% 6)Standardizing

imagepath =
strcat(vl rootnn,'\data\CEDAR\signatures\full org\cropped org');
load(imagepath)
writer_num = 55;
sig_num = 24;
%% complement
for i=1:writer_num
    for j=1:sig_num
        cropped_imgray{i,j} = imcomplement(cropped_imgray{i,j});
    end
end
%% resize
for i=1:writer_num
    for j=1:sig_num
        resized{i,j} = imresize(cropped_imgray{i,j},[80 120],'bilinear');
        res_labels{i,j} = i;
    end
end
%% data augmentation
%rotate
k=1;
for i=11:55
    m=1;
    for j=1:sig_num
        theta = randi([-10 10],1,1);
        [rotated{k,m}] = im_rotation(resized{i,j},theta);
        rot_labels{k,m} = i;
        m=m+1;
    end
    k=k+1;
end
```

```
%concatenating
images = cat(1,resized,rotated);
labels = cat(1,res_labels,rot_labels);
%% Development set "D"
DEVimages = cell(90,sig_num);
DEVlabels = cell(90,sig_num);
k=1;
for i=11:100
    for j=1:sig_num
        DEVimages{k,j} = images{i,j};
        DEVlabels{k,j} = labels{i,j};
    end
    k=k+1;
end
clearvars -except DEVimages DEVlabels
%% Setting train val on Development set
k=1;
for i=1:size(DEVimages,1)
    for j=1:size(DEVimages,2)
        development.images.data(:,:,:,k) = single(DEVimages{i,j});
        development.images.labels(k,1) = DEVlabels{i,j};
        k=k+1;
    end
end
development.meta.sets = {'train','val','test'};
development.meta.classes =
arrayfun(@(x)sprintf('%d',x),11:55,'uniformoutput',false);
% Pick random validation 80/20%
for i=1:numel(development.images.labels)
        development.images.set(i,1) = randi(100,1);
        if development.images.set(i,1) <= 20
            development.images.set(i,1) = 2;
        else development.images.set(i,1) = 1;
        end
end
%% Standardizing
samples = numel(development.images.labels);
z = reshape(development.images.data,[],samples);
n = std(z,0,1);
z = bsxfun(@rdivide, z, n);
development.images.data = reshape(z, 80, 120, 1, []);

clearvars -except development

function  [imrot] = im_rotation(im,theta)
[ref_h,ref_w,~] = size(im);

%% Rotation & setting white corners
Irot=imrotate(im,theta);
Mrot = ~imrotate(true(size(im)),theta);
Irot(Mrot&~imclearborder(Mrot)) = 0;


[imr_h,imr_w,~]=size(Irot);

%% going back to reference Height
sc=max(imr_h/ref_h,imr_w/ref_w);
imrot=imresize(Irot,1/sc);

%% going back to reference Width
```

```
hpad = round((ref_h - size(imrot,1))/2);
hpad(hpad<0) = 0;
wpad = round((ref_w - size(imrot,2))/2);
wpad(wpad<0) = 0;

imrot = padarray(imrot,[hpad wpad],0,'both');
imrot = imrot(1:80,1:120); %sometimes it creates 81x121 arrays
end
```

Δημιουργία *Exploitation set*

```
%% Exploitation set
load('development_set.mat');
load('skilled_forg.mat');
load('cropped_org.mat');
writer_num = 10;
sig_num = 24;
%% originals, random_forg
% complement
for i=1:writer_num
    for j=1:sig_num
        cropped_imgray{i,j} = imcomplement(cropped_imgray{i,j});
    end
end
% resize
k=1;
for i=1:writer_num
    %org
    for j=1:sig_num
        exploitation.images.data(:,:,:,k) =
single(imresize(cropped_imgray{i,j},[80 120],'bilinear'));
        exploitation.images.labels(1,k) = i;
        if j<=14
            exploitation.images.set(1,k) = 1;
        else exploitation.images.set(1,k) = 3;
        end
        exploitation.images.originality(1,k) = 1;
        k=k+1;
    end
    %rand forg
    for j=1:38
        exploitation.images.data(:,:,:,k) =
development.images.data(:,:,:,1+(j-1)*24);
        exploitation.images.labels(1,k) = i;
        exploitation.images.originality(1,k) = 2;
        if j<=28
           exploitation.images.set(1,k) = 1;
        else exploitation.images.set(1,k) = 3;
        end
        k=k+1;
    end
end
%% skilled_forg
exploitation.images.data =
cat(4,exploitation.images.data,skilled_forg.images.data);
exploitation.images.labels =
cat(2,exploitation.images.labels,skilled_forg.images.labels');
```

```
exploitation.images.set =
cat(2,exploitation.images.set,skilled_forg.images.set');
exploitation.images.originality =
cat(2,exploitation.images.originality,skilled_forg.images.originality');
%% meta
exploitation.meta.originality = {'genuine','random forg','skilled forg'};
exploitation.meta.sets = {'train','val','test'};
exploitation.meta.classes =
arrayfun(@(x)sprintf('%d',x),1:10,'uniformoutput',false);
%% Standardizing
samples = numel(exploitation.images.labels);
z = reshape(exploitation.images.data,[],samples);
n = std(z,0,1);
n(exploitation.images.originality==2) = 1;
z = bsxfun(@rdivide, z, n);
exploitation.images.data = reshape(z, 80, 120, 1, []);
```

Εξαγωγή χαρακτηριστικών και προετοιμασία για την εισαγωγή στον *SVM*

```
%% feature extraction

load(strcat(vl_rootnn,'/data/CEDAR-adam-simplenn/net.mat'))

net.layers{1,end}.type = 'softmax' ;
res = [];
for i=1:numel(exploitation.images.set)
    res = vl_simplenn(net,exploitation.images.data(:,:,:,i));
    exploitation.images.features(i,:) = double(squeeze(gather(res(end-
2).x))');
end

%% Split sets for every SVM
for i=1:10
    indices = find(exploitation.images.labels==i);
    writerID{i,1}.features = exploitation.images.features(indices,:);
    writerID{i,1}.set = exploitation.images.set(indices);
    writerID{i,1}.originality = exploitation.images.originality(indices);
    writerID{i,1}.labels(writerID{i,1}.originality==1) = 1;
    writerID{i,1}.labels(writerID{i,1}.originality==2) = 0;
    writerID{i,1}.labels(writerID{i,1}.originality==3) = 0;
end
```

Εκπαίδευση των *SVM* και εξαγωγή των αποτελεσμάτων

```
% fgtr : Original Train signatures features of writer
% frftr : Random Forgeries Train signatures features from #N_fr others
writers
% fgts : rest Original Test signatures features of writer
% fsfts : Skilled Forgeries Test signatures features of writer
% frfts : Random Forgeries Test signatures features from all others
writers
max_signers = 10;

for i=1:max_signers
fgtr = writerID{i,1}.features(writerID{i,1}.originality==1 &
writerID{i,1}.set==1,:);
frftr = writerID{i,1}.features(writerID{i,1}.originality==2 &
writerID{i,1}.set==1,:);
fgts = writerID{i,1}.features(writerID{i,1}.originality==1 &
writerID{i,1}.set==3,:);
fsfts = writerID{i,1}.features(writerID{i,1}.originality==3 &
writerID{i,1}.set==3,:);
frfts = writerID{i,1}.features(writerID{i,1}.originality==2 &
writerID{i,1}.set==3,:);

[smodl{i,1},dec{i,1},labels{i,1},far{i,1},pd{i,1},auc{i,1},dec_val{i,1},t
{i,1},predict{i,1},SVMmodel] =...
    trainGRF_test_with_CV_2classSVM(fgtr,frftr,fgts,fsfts,frfts);

end

% ROC
max_signers = 10;
for writer_id=1:max_signers
figure,plot(far{writer_id}.sf,pd{writer_id}.sf,'r*');
end

%%% EER
EER_writer_id = cell(1,max_signers);
for writer_id = 1:max_signers

% AVE ~ EER
% best average error ~ Equal Error Rate

% Skilled Forgeries and Original Forgeries
EER_writer_id{writer_id} = min(mean([far{writer_id}.sf 1-
pd{writer_id}.sf],2));

end

% mean EER for all the writers of the database
EER_db = mean(cell2mat(EER_writer_id));
```

Συνάρτηση εκπαίδευσης και *Cross Validation*

```
function
[smodl,dec,labels,far,pd,auc,dec_val,t,predict,SVMmodel]=trainGRF_test_wi
th_CV_2classSVM(fgtr,frftr,fgts,fsfts,frfts)
%% Training with features: genuine (fgtr), and random forgeries (frftr).
% Training with features: genuine (fgts), skilled (fsfts) and random
forgeries (frfts).
addpath(genpath('.\libsvm_code'))
LRS=size(fgtr,1);
mul_RF_LRS=2; % how many times x LRS samples are drawn for the random
forgery training.
train_label=[ones(1,LRS) -1*ones(1,mul_RF_LRS*LRS)]';

%%%%%%%% SVM %%%%%%%%
smodl=0;
gamma=2.^(-8:0.5:5); % was 2.^(-15:0.5:3)
C=2.^(-3:0.5:10);     % (-8:0.5:10)
rep=10;
for k=1:rep
    rng(k)
    train_data = [fgtr ;
frftr(randsample(size(frftr,1),mul_RF_LRS*LRS),:)];
    for g=1:27
        for n=1:27

[modl(g,n),dec(g,n).vals,labels(g,n).lab]=do_binary_cross_validation(trai
n_label, train_data, ['-s 0 -t 2 -c ' num2str(C(n)) ' -g '
num2str(gamma(g)) ' -q '], LRS);
        end
    end
   smodl=smodl+modl/rep;
end

[gi,Cj]=find(smodl==max(smodl(:)),1);

SVMmodel = svmtrain(train_label,train_data,['-s 0 -t 2 -c '
num2str(C(Cj)) ' -g ' num2str(gamma(gi)) ' -q ']);
testing_instance_matrix=[fgts;fsfts;frfts];
testing_label_vector=[ones(1,size(fgts,1)) -1*ones(1,size(fsfts,1)) -
1*ones(1,size(frfts,1))]';

[predict.labels,predict.acc, dec_val] = svmpredict(testing_label_vector,
testing_instance_matrix, SVMmodel, ' -q ');
%[far,pd,t,auc] = perfcurve(testing_label_vector ,dec_val,1);
[far.global,pd.global,t.global,auc.global] =
perfcurve(testing_label_vector ,dec_val,1);
[far.sf,pd.sf,t.sf,auc.sf] =
perfcurve(testing_label_vector(1:size(fgts,1)+size(fsfts,1))
,dec_val(1:size(fgts,1)+size(fsfts,1)),1);

rmpath(genpath('.\libsvm_code'))
end
```

```
function [ret,dec_values,labels]=do_binary_cross_validation(y, x, param,
nr_fold)
len = length(y);
rand_ind = randperm(len);
dec_values = [];
labels = [];
for i = 1:nr_fold % Cross training : folding
  test_ind = rand_ind([floor((i-
1)*len/nr_fold)+1:floor(i*len/nr_fold)]');
  train_ind = [1:len]';
  train_ind(test_ind) = [];
  model = svmtrain(y(train_ind),x(train_ind,:),param);
  [pred, acc, dec] = svmpredict(y(test_ind),x(test_ind,:),model,' -q ');
  if model.Label(1) < 0;
    dec = dec * -1;
  end
  dec_values = vertcat(dec_values, dec);
  labels = vertcat(labels, y(test_ind));
end
%disp(sprintf('Cross Validation:'));
%ret=validation_function_v1(dec_values, labels);
[far,pd,t,auc] = perfcurve(labels,dec_values,1);
ret=auc;
```

Αναφορές

1. A.K. Jain, A.A Ross, and K. Nandakumar, *Introduction to Biometrics*. 2011, New York: Springer.
2. R.A. Huber and A.M. Headrick, *Handwriting identification: facts and fundamentals*. 2010, Boca Raton: CRC Press.
3. Impedovo, D. and G. Pirlo, *Automatic Signature Verification: The State of the Art.* IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, 2008. **38**(5): p. 609-635.
4. Alewijnse, L.C., *Analysis of Signature Complexity, Master Thesis*. 2008, University of Amsterdam
5. Houmani, N. and S. Garcia-Salicetti, *Quality Measures for Online Handwritten Signatures*, in *Signal and Image Processing for Biometrics*, J. Scharcanski, H. Proença, and E. Du, Editors. 2014, Springer Berlin Heidelberg, . p. 255-283.
6. *Machine Learning*. [cited 2017; Available from: https://en.wikipedia.org/wiki/Machine_learning.
7. University, S. *UFLDL Tutorial*. [cited 2017; Available from: http://ufldl.stanford.edu/tutorial/.
8. Raschka, S. *Single-Layer Neural Networks and Gradient Descent*. [cited 2017; Available from: http://sebastianraschka.com/Articles/2015_singlelayer_neurons.html.
9. Ruder, S., *Single-Layer Neural Networks and Gradient Descent.* 2017.
10. Papadopoulos, A., *ΣΥΝΕΛΙΚΤΙΚΑ ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ ΣΤΗΝ ΥΠΟΛΟΓΙΣΤΙΚΗ ΌΡΑΣΗ*, in *Department of Physics*. 2016, University of Patras.
11. Samer Hijazi, R.K., Chris Rowen, *Using Convolutional Neural Networks for Image Recognition.* 2015.
12. Stanford, U.o. *CS231n Convolutional Neural Networks for Visual Recognition*. [cited 2017; Available from: http://cs231n.github.io/convolutional-networks/.
13. Matthew D Zeiler, R.F., *Visualizing and Understanding Convolutional Networks.* 2013.
14. Hafemann, L.G., R. Sabourin, and L.S. Oliveira. *Writer-independent feature learning for Offline Signature Verification using Deep Convolutional Neural Networks*. in *2016 International Joint Conference on Neural Networks (IJCNN)*. 2016.
15. Hafemann, L.G., R. Sabourin, and L.S. Oliveira, *Analyzing features learned for Offline Signature Verification using Deep CNNs.* arXiv preprint arXiv:1607.04573, 2016.
16. Fawcett, T., *An introduction to ROC analysis.* Pattern Recognition Letters, 2006. **27**(8): p. 861-874.