ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Μηχανική μάθηση για την αναγνώριση και πρόβλεψη χειρόγραφου κειμένου

Πτυχιακή εργασία

ΣΤΕΛΙΟ ΜΠΟΜΠΑΪ

Αθήνα, Ιούλιος 2020

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

Καραγιώργου Σοφία

Επισκέπτης Λέκτορας,

Τμήμα Πληροφορικής και Τηλεματικής,

Χαροκόπειο Πανεπιστήμιο

Βαρλάμης Ηρακλής

Επίκουρος Καθηγητής,

Τμήμα Πληροφορικής και Τηλεματικής,

Χαροκόπειο Πανεπιστήμιο

Μιχαήλ Δημήτρης

Επίκουρος Καθηγητής,

Τμήμα Πληροφορικής και Τηλεματικής,

Χαροκόπειο Πανεπιστήμιο

Μηχανική μάθηση για την αναγνώριση και πρόβλεψη χειρόγραφου κειμένου

Στέλιο Μπομπάϊ

Ο Στέλιο Μπομπάϊ

δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

Σελίδα για Αφιέρωση (προαιρετικά)

Σελίδα για γνωμικό (προαιρετικά)

ΕΥΧΑΡΙΣΤΙΕΣ

Ολοκληρώνοντας τη πτυχιακή μου εργασία, θα ήθελα να ευχαριστήσω όλους όσους βοήθησαν σε όλα τα στάδια ανάπτυξής της. Αρχικά, θα ήθελα να ευχαριστήσω την επιβλέπουσα καθηγήτρια κ. Καραγιώργου Σοφία για την εξαιρετική συνεργασία καθώς και τις εύστοχες υποδείξεις καθ' όλη τη διάρκεια της εκπόνησης της εργασίας. Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου και τους κοντινούς μου ανθρώπους για την στήριξη κατά την διάρκεια εκπόνησης της συγκεκριμένης πτυχιακής εργασίας.

Πίνακας Περιεχομένων

Περίληψη

Σήμερα, η ακριβής αναγνώριση των εκτυπωμένων χαρακτήρων θεωρείται ως επιλυμένο πρόβλημα. Πολλά εμπορικά προϊόντα επικεντρώνονται προς αυτήν την κατεύθυνση, επιτυγχάνοντας υψηλά ποσοστά αναγνώρισης. Ωστόσο, η αναγνώριση χειρόγραφου χαρακτήρα είναι συγκριτικά δυσκολότερη. Έτσι, η αναγνώριση χειρόγραφων εγγράφων εξακολουθεί να αποτελεί αντικείμενο ενεργού έρευνας.

Επιπλέον, η πρόβλεψη λέξης ή τα γλωσσικά μοντέλα είναι η εργασία της πρόβλεψης των πιο πιθανών λέξεων με βάση τις προηγούμενες λέξεις. Έχει πολλές εφαρμογές, όπως η πρόταση της επόμενης λέξης κατά την εισαγωγή κειμένου, ως βοήθημα στην επίλυση της ασάφειας στην αναγνώριση ομιλίας και γραφής και στη μηχανική μετάφραση.

Αυτή η εργασία στοχεύει στην ανάπτυξη μιας αλληλουχίας ως εξής: προ-επεξεργασία μιας εικόνας εισόδου, ανίχνευση κειμένου, τμηματοποίηση γραμμής, τμηματοποίηση χαρακτήρων, αναγνώριση χαρακτήρων και τέλος πρόβλεψη λέξεων. Τα δύο κύρια στάδια αυτού του αγωγού είναι το στάδιο αναγνώρισης και το στάδιο πρόβλεψης. Στο στάδιο αναγνώρισης το Computer Vision μαζί με το CNN, που αναπτύχθηκε με την Keras βιβλιοθήκη, χρησιμοποιήθηκαν για την προ-επεξεργασία και την αναγνώριση κειμένου από την εικόνα εισόδου, ενώ στο στάδιο της πρόβλεψης υπάρχουν δύο βασικές μεθοδολογίες, το Markov μοντέλο και το LSTM, που χρησιμοποιούνται για την πρόβλεψη κειμένου με βάση το αποτέλεσμα του προηγούμενου σταδίου.

Συνοψίζοντας, επιτεύχθηκε ακρίβεια **99,834%** σε **120 λεπτά** για το μοντέλο CNN και με την κατάλληλη προ-επεξεργασία στην εικόνα εισόδου μπορεί να υπάρξει ακριβής αναγνώριση των χαρακτήρων της εικόνας. Επιπλέον, αν και, τα αποτελέσματα από το μοντέλο Markov έχουν νόημα στην αγγλική γλώσσα και είναι πολύ γρήγορα να ληφθούν (λίγα δευτερόλεπτα), δεν είναι πολύ λογικά. Αντιθέτως, τα αποτελέσματα από το μοντέλο LSTM, το οποίο πέτυχε ακρίβεια **14.751%** σε **150 λεπτά**, είναι πιο συντακτικά και γραμματικά σωστά και θα μπορούσαν να είναι ένα πραγματικό κείμενο.

Λέξεις κλειδιά: μηχανική μάθηση, νευρωνικά δίκτυα, επεξεργασία φυσικής γλώσσας, αναγνώριση εικόνας

Abstract

Nowadays, the accurate recognition of machine printed characters is considered largely a solved problem. A lot of commercial products are focused towards that direction, achieving high recognition rates. However, handwritten character recognition is comparatively difficult. So, the recognition of handwritten documents is still a subject of active research.

In addition, Word prediction, or *language modeling*, is the task of predicting the most likely words following the preceding text. It has many applications, such as suggesting the next word as text is entered, as an aid in resolving ambiguity in speech and handwriting recognition, and in machine translation.

This thesis aims at developing a pipeline as followed: preprocessing an input image, text detection, line segmentation, character segmentation, character recognition and finally word prediction. The two main stages of this pipeline are the recognition stage and prediction stage. At the recognition stage Computer Vision along with CNN, developed with Keras, were used to preprocess and recognize text from the input image, while at the prediction stage there are two main methodologies, Markov Models and LSTM (Long short-term memory), used to predict text based on the output of the previous stage.

To sum up, an accuracy of **99.834%** in **120 minutes** was achieved for the CNN model and with the proper preprocessing in the input image there can be an accurate recognition of the characters of the image. Moreover, although, the results from the Markov model make sense in the English language and are really fast to obtain (a few seconds), are not very logical. In contrast, the results from the LSTM model, which achieved an accuracy of **14.751%** in **150 minutes**, are more syntactically and grammatically correct and they could be a real-world text.

Keywords: machine learning, neural networks, natural language processing, image classification

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

1 *Εισαγωγή*

Με τα συνεχώς αυξανόμενα δεδομένα σε ηλεκτρονική μορφή, η ανάγκη για αυτοματοποιημένες μεθόδους ανάλυσης δεδομένων συνεχίζει να αυξάνεται. Ο στόχος της μηχανικής μάθησης είναι η ανάπτυξη μεθόδων που μπορούν να ανιχνεύσουν αυτόματα τα πρότυπα στα δεδομένα και στη συνέχεια να χρησιμοποιήσουν αυτά τα πρότυπα για να προβλέψουν μελλοντικά δεδομένα ή άλλα ενδιαφέροντα αποτελέσματα. Επομένως, η μηχανική μάθηση συνδέεται στενά με τον τομέα της στατιστικής και της εξόρυξης δεδομένων, αλλά διαφέρει ελαφρώς όσον αφορά την έμφαση και την ορολογία της.

Μέρος μιας οικογένειας μεθόδων της μηχανικής μάθησης είναι η βαθιά μάθηση, η οποία είναι βασισμένη στα τεχνητά νευρωνικά δίκτυα. Η ονομασία “βαθιά”, προέρχεται από την χρήση πολλαπλών επιπέδων στο μοντέλο. Είδη νευρωνικών δικτύων, όπως τα Συνελικτικά Νευρωνικά Δίκτυα και τα Επαναλαμβανόμενα Νευρωνικά Δίκτυα, είναι ευρέως γνωστά τα τελευταία χρόνια λόγω της απόδοσης τους σε εφαρμογές όπως η αυτόματη αναγνώριση ομιλίας, η αναγνώριση εικόνας και η επεξεργασία φυσικής γλώσσας.

Μια ειδική περίπτωση εφαρμογής αναγνώρισης εικόνας είναι οι εικόνες που αποτελούνται από μεμονωμένα χειρόγραφα γράμματα ή και ψηφία. Στην εποχή μας, η ακριβής αναγνώριση των εκτυπωμένων χαρακτήρων θεωρείται ως επιλυμένο πρόβλημα. Πολλά εμπορικά προϊόντα επικεντρώνονται προς αυτήν την κατεύθυνση, επιτυγχάνοντας υψηλά ποσοστά αναγνώρισης. Ωστόσο, η αναγνώριση χειρόγραφου χαρακτήρα είναι συγκριτικά δυσκολότερη. Έτσι η αναγνώριση χειρόγραφων εγγράφων εξακολουθεί να αποτελεί αντικείμενο ενεργού έρευνας.

Επιπλέον, μια ειδική περίπτωση εφαρμογής της επεξεργασίας φυσικής γλώσσας είναι η πρόβλεψη της επόμενης ή των επόμενων λέξεων δεδομένου προηγούμενων λέξεων. Παραδοσιακά, τα n-gram μοντέλα ήταν η τυπική προσέγγιση στα γλωσσικά μοντέλα (language models). Παρόλα αυτά τα τελευταία

χρόνια, λόγω των νευρωνικών δικτύων, τα Επαναλαμβανόμενα Νευρωνικά Δίκτυα έχουν εφαρμοστεί με επιτυχία στο language modeling.

1.1 Αντικείμενο εργασίας

Αντικείμενο της παρούσας εργασίας είναι να αξιοποιήσει τα Συνελικτικά Νευρωνικά Δίκτυα, για την εκμάθηση ενός μοντέλου αναγνώρισης χειρόγραφων χαρακτήρων, και με την κατάλληλη προ-επεξεργασία μίας εικόνας εισόδου να αναγνωρίσει το κείμενο που αναδεικνύεται στην εικόνα. Έπειτα για την πρόβλεψη της επόμενης ή επόμενων λέξεων με βάση το κείμενο της εικόνας που έχει αναγνωριστεί, γίνεται αξιοποίηση ενός Επαναλαμβανόμενου Νευρωνικού Δικτύου και ενός στατιστικού στοχαστικού μοντέλου (Markov model) και γίνεται μία σύγκριση των αποτελεσμάτων των δύο μοντέλων όσον αφορά τον χρόνο και τη συντακτική λογική τους.

1.2 Οργάνωση κειμένου

Η συνέχεια της παρούσας εργασίας οργανώνεται σύμφωνα με την περιγραφή που ακολουθεί. Στο κεφάλαιο 2 γίνεται εισαγωγική αναφορά στην μηχανική εκμάθηση. Στο κεφάλαιο 3 γίνεται εισαγωγική αναφορά στην βαθιά μάθηση. Το κεφάλαιο 4 καλύπτει τις βιβλιοθήκες, τα σύνολα δεδομένων και τις τεχνολογίες που χρησιμοποιήθηκαν. Το κεφάλαιο 5 αναλύει την μεθοδολογία που ακολουθήθηκε για την διεκπεραίωση της εργασίας. Το κεφάλαιο 6 παρουσιάζει τα αποτελέσματα που λήφθηκαν. Το κεφάλαιο 7 παραθέτει τα συμπεράσματα από αυτήν την εργασία και προτεινόμενες μελλοντικές επεκτάσεις. Στο κεφάλαιο 8 παρατίθεται η βιβλιογραφία και τέλος στο κεφάλαιο 9 παρατίθεται ο κώδικας που υλοποιήθηκε για την εργασία αυτή.

2 *Μηχανική Μάθηση*

Η βαθιά μάθηση είναι ένα συγκεκριμένο είδος μηχανικής μάθησης. Για να κατανοηθεί καλά η βαθιά μάθηση, πρέπει να υπάρχει μια καλή κατανόηση των βασικών αρχών της μηχανικής μάθησης. Οι περισσότεροι αλγόριθμοι μάθησης μηχανών έχουν ρυθμίσεις που καλούνται **υπερπαραμετρικά**, τα οποία πρέπει να προσδιοριστούν εκτός του αλγορίθμου μάθησης. Η μηχανική μάθηση αποτελεί ουσιαστικά μια μορφή εφαρμοσμένης στατιστικής με μεγαλύτερη έμφαση στη χρήση υπολογιστών για στατιστική εκτίμηση περίπλοκων λειτουργιών και μειωμένη έμφαση που αποδεικνύει διαστήματα εμπιστοσύνης γύρω από αυτές τις λειτουργίες. Οι περισσότεροι αλγόριθμοι μάθησης μηχανών μπορούν να χωριστούν στις κατηγορίες της εποπτευόμενης μάθησης και της μη εποπτευόμενης μάθησης. Οι περισσότεροι αλγόριθμοι βαθιάς μάθησης βασίζονται σε έναν αλγόριθμο βελτιστοποίησης που ονομάζεται *stochastic gradient descent*.

2.1 Αλγόριθμοι μάθησης

Ένας αλγόριθμος μάθησης μηχανών είναι ένας αλγόριθμος που μπορεί να μάθει από τα δεδομένα. Ο Mitchell το 1997 (Mitchell, 1997) παρέχει έναν συνοπτικό ορισμό: " Ένα πρόγραμμα ηλεκτρονικού υπολογιστή λέγεται ότι μαθαίνει από την εμπειρία E σε σχέση με κάποια τάξη εργασιών T και μέτρηση απόδοσης P, αν η απόδοσή του στις εργασίες T, όπως μετράται με P, βελτιώνεται με την εμπειρία E ". Μπορούμε να φανταστούμε μια ευρεία ποικιλία εμπειριών E, εργασιών T και μέτρων επιδόσεων P. Καλά παραδείγματα και περιγραφές αυτών των εργασιών, μέτρων απόδοσης και εμπειριών προσφέρει ο Goodfellow (Goodfellow, Bengio, & Courville, 2016).

2.1.1 Η εργασία Τ

Η μηχανική μάθηση μας δίνει τη δυνατότητα να αντιμετωπίσουμε εργασίες που είναι υπερβολικά δύσκολες να επιλυθούν με καθορισμένα προγράμματα γραπτά και σχεδιασμένα από ανθρώπους. Από επιστημονική και φιλοσοφική άποψη, η μηχανική μάθηση είναι ενδιαφέρουσα, διότι η ανάπτυξη της κατανόησής μας για αυτήν συνεπάγεται με την ανάπτυξη της κατανόησης των αρχών που διέπουν τη νοημοσύνη.

Σε αυτόν τον σχετικά επίσημο ορισμό της λέξης «εργασία», η ίδια διαδικασία της μάθησης δεν είναι εργασία. Η μάθηση είναι το μέσο μας για την επίτευξη της ικανότητας εκτέλεσης της εργασίας. Για παράδειγμα, εάν θέλουμε ένα ρομπότ να μπορεί να περπατάει, τότε το περπάτημα είναι η εργασία. Θα μπορούσαμε να προγραμματίσουμε το ρομπότ να μάθει να περπατάει, ή θα μπορούσαμε να προσπαθήσουμε να γράψουμε άμεσα ένα πρόγραμμα που καθορίζει πώς να περπατήσει χειροκίνητα.

Οι εργασίες μηχανικής μάθησης περιγράφονται συνήθως ως προς τον τρόπο με τον οποίο το σύστημα μηχανικής μάθησης πρέπει να επεξεργάζεται ένα παράδειγμα. Ένα **παράδειγμα** είναι μια συλλογή χαρακτηριστικών που έχουν μετρηθεί ποσοτικά από κάποιο αντικείμενο ή γεγονός που θέλουμε το σύστημα μηχανικής μάθησης να επεξεργαστεί. Αντιπροσωπεύουμε συνήθως ένα παράδειγμα ως ένα διάνυσμα $x \in R^n$ όπου κάθε καταχώριση x_i του διανύσματος είναι ένα άλλο χαρακτηριστικό. Για παράδειγμα, τα χαρακτηριστικά μιας εικόνας είναι συνήθως οι τιμές των εικονοστοιχείων στην εικόνα.

Πολλά είδη εργασιών μπορούν να επιλυθούν με τη μηχανική μάθηση. Μερικές από τις πιο συχνές εργασίες μηχανικής μάθησης περιλαμβάνουν τα ακόλουθα :

2.1.1.1 Ταξινόμηση

Σε αυτού του τύπου την εργασία, το πρόγραμμα του υπολογιστή ζητείται να προσδιορίσει σε πια από τις κατηγορίες *k* ανήκει κάποια είσοδος. Για να λυθεί αυτή η εργασία, ο αλγόριθμος μάθησης συνήθως ζητείται να παράγει μια συνάρτηση τύπου $f : R^n => \{ 1,…., k\}$. Όταν $y = f(x)$, το μοντέλο αναθέτει μία είσοδο που

περιγράφεται από το διάνυσμα *x* σε μία κατηγορία που προσδιορίζεται από τον αριθμητικό κωδικό *y*. Υπάρχουν και άλλες παραλλαγές της εργασίας ταξινόμησης, για παράδειγμα, όπου το *f* εξάγει μια κατανομή πιθανότητας σε τάξεις. Ένα παράδειγμα μιας εργασίας ταξινόμησης είναι η αναγνώριση αντικειμένου, όπου η είσοδος είναι μια εικόνα (συνήθως περιγράφεται ως ένα σύνολο τιμών φωτεινότητας εικονοστοιχείων) και η έξοδος είναι ένας αριθμητικός κωδικός που προσδιορίζει το αντικείμενο στην εικόνα. Για παράδειγμα, το ρομπότ Willow Garage PR2 μπορεί να λειτουργήσει ως σερβιτόρος που μπορεί να αναγνωρίσει διάφορα είδη ποτών και να τα παραδώσει σε ανθρώπους με εντολή (Goodfellow, et al., 2010). Η σύγχρονη αναγνώριση αντικειμένων επιτυγχάνεται καλύτερα με τη βαθιά μάθηση (Krizhevsky, Sutskever, & Hinton, 2012; Ioffe & Szegedy, 2015). Η αναγνώριση αντικειμένων είναι η ίδια βασική τεχνολογία που επιτρέπει στους υπολογιστές να αναγνωρίζουν πρόσωπα (Taigman, Yang, Ranzato, & Wolf, 2014) το οποίο μπορεί να χρησιμοποιηθεί για αυτόματη προσθήκη ετικετών σε συλλογές φωτογραφιών και για υπολογιστές να αλληλεπιδρούν πιο φυσικά με τους χρήστες τους.

2.1.1.2 Ταξινόμηση με ελλιπείς εισόδους

Η ταξινόμηση γίνεται πιο δύσκολη εάν το πρόγραμμα υπολογιστή δεν είναι εγγυημένο ότι στο διάνυσμα εισόδου του, κάθε μέτρηση θα παρέχεται πάντα. Για να λυθεί αυτή η εργασία ταξινόμησης, ο αλγόριθμος μάθησης πρέπει να καθορίσει μόνο μία συνάρτηση που να συνδέεται από ένα διάνυσμα εισόδου σε ένα κατηγορικό αποτέλεσμα. Όταν όμως κάποιες από τις εισόδους μπορεί να λείπουν, αντί να παρέχει μια μοναδική συνάρτηση, ο αλγόριθμος μάθησης πρέπει να μάθει ένα σύνολο από συναρτήσεις. Κάθε συνάρτηση αντιστοιχεί στο να ταξινομήσει το *x* με διαφορετικό υποσύνολο των εισόδων που της λείπουν. Ένας τρόπος για να προσδιοριστεί αποτελεσματικά ένα τόσο μεγάλο σύνολο συναρτήσεων είναι να βρεθεί μια κατανομή πιθανότητας για όλες τις σχετικές μεταβλητές και έπειτα να λυθεί η εργασία ταξινόμησης περιθωριοποιώντας τις ελλείπουσες μεταβλητές.

2.1.1.3 Παλινδρόμηση

Σε μία τέτοια εργασία, το πρόγραμμα υπολογιστή ζητείται να προβλέψει μία αριθμητική τιμή αφού δοθεί κάποια είσοδος. Έτσι για να λυθεί, ο αλγόριθμος μάθησης ζητείται να παράγει μια συνάρτηση $f : R^n => R$. Αν και αυτή η εργασία είναι παρόμοια με την εργασία ταξινόμησης, η διαφορά τους είναι η μορφή της εξόδου. Ένα παράδειγμα μιας εργασίας παλινδρόμησης είναι η πρόβλεψη του αναμενόμενου ποσού αξίωσης που θα κάνει ο ασφαλισμένος ή η πρόβλεψη μελλοντικών τιμών ασφάλειας. Αυτά τα είδη προβλέψεων χρησιμοποιούνται επίσης για αλγοριθμικές συναλλαγές.

2.1.1.4 Μεταγραφή

Σε μία τέτοια εργασία, το σύστημα μηχανικής μάθησης ζητείται να παρατηρήσει μία σχετικά αδόμητη αναπαράσταση κάποιου είδους δεδομένων και να μεταγράψει την πληροφορία σε διακριτή μορφή κειμένου. Για παράδειγμα, σε μη οπτική αναγνώριση χαρακτήρων, στο πρόγραμμα υπολογιστή εμφανίζεται μια φωτογραφία που περιέχει μια εικόνα κειμένου και καλείται να επιστρέψει αυτό το κείμενο με τη μορφή ακολουθίας χαρακτήρων. Ένα άλλο παράδειγμα είναι η αναγνώριση ομιλίας, όπου στο πρόγραμμα υπολογιστή παρέχεται μια ηχητική κυματομορφή και εκπέμπεται μια ακολουθία χαρακτήρων ή κωδικών ID λέξεων που περιγράφουν τις λέξεις που εκφωνήθηκαν στην ηχογράφηση. Η βαθιά μάθηση είναι ένα κρίσιμο στοιχείο των σύγχρονων συστημάτων αναγνώρισης ομιλίας που χρησιμοποιούνται σε μεγάλες εταιρείες, συμπεριλαμβανομένων Microsoft, IBM και Google (Hinton, et al., 2012).

2.1.1.5 Μηχανική μετάφραση

Σε μια εργασία μηχανικής μετάφρασης, η είσοδος αποτελείται ήδη από μια ακολουθία συμβόλων σε κάποια γλώσσα και το πρόγραμμα υπολογιστή πρέπει να το μετατρέψει σε μια ακολουθία συμβόλων σε άλλη γλώσσα. Αυτό εφαρμόζεται συνήθως σε φυσικές γλώσσες, όπως μετάφραση από Αγγλικά σε Γαλλικά.

2.1.1.6 Δομημένο αποτέλεσμα

Οι δομημένες εργασίες αποτελέσματος περιλαμβάνουν κάθε εργασία όπου το αποτέλεσμα είναι ένα διάνυσμα (ή άλλη δομή δεδομένων που περιέχει πολλές τιμές) με σημαντικές σχέσεις μεταξύ των διάφορων στοιχείων. Αυτή είναι μια ευρεία κατηγορία και υπάγει την μεταγραφή και τις εργασίες μετάφρασης. Ένα παράδειγμα είναι η ανάλυση - αντιστοίχιση μιας φράσης φυσικής γλώσσας σε ένα δέντρο που περιγράφει τη γραμματική του δομή επισημαίνοντας τους κόμβους των δέντρων ως ρήματα, ουσιαστικά, επίρρημα και ούτω καθεξής. Αυτές οι εργασίες ονομάζονται δομημένες εργασίες αποτελέσματος επειδή το πρόγραμμα πρέπει να παράγει πολλές τιμές που είναι στενά αλληλένδετες. Για παράδειγμα, οι λέξεις που παράγονται από ένα πρόγραμμα λεζάντας εικόνας πρέπει να αποτελούν μια έγκυρη πρόταση.

2.1.2 Το μέτρο απόδοσης, P

Για να αξιολογήσουμε τις ικανότητες ενός αλγορίθμου μηχανικής μάθησης, πρέπει να σχεδιάσουμε ένα ποσοτικό μέτρο της απόδοσής του. Συνήθως αυτό το μέτρο απόδοσης P είναι συγκεκριμένο για την εργασία T που εκτελείται από το σύστημα.

Για εργασίες όπως η ταξινόμηση, η ταξινόμηση με ελλιπείς εισόδους και η μεταγραφή, συχνά μετράμε την ακρίβεια του μοντέλου. Η ακρίβεια είναι απλά η αναλογία των παραδειγμάτων για τα οποία το μοντέλο παράγει το σωστό αποτέλεσμα. Μπορούμε επίσης να λάβουμε ισοδύναμες πληροφορίες μετρώντας το **ποσοστό σφάλματος**, την αναλογία των παραδειγμάτων για τα οποία το μοντέλο παράγει ένα λανθασμένο αποτέλεσμα.

Συνήθως άξιο ενδιαφέροντος είναι το πόσο καλά ο αλγόριθμος μηχανικής μάθησης αποδίδει σε δεδομένα που δεν έχει δει ποτέ, αφού αυτό καθορίζει το πόσο καλά θα λειτουργήσει όταν αναπτυχθεί στον πραγματικό κόσμο. Επομένως, αξιολογούνται αυτά τα μέτρα απόδοσης χρησιμοποιώντας ένα δοκιμαστικό σύνολο δεδομένων που είναι ξεχωριστό από τα δεδομένα που χρησιμοποιούνται για την εκπαίδευση του συστήματος μηχανικής εκμάθησης.

Η επιλογή του μέτρου απόδοσης μπορεί να φαίνεται απλή και αντικειμενική, αλλά συχνά είναι δύσκολο να επιλεχθεί ένα μέτρο απόδοσης που αντιστοιχεί καλά στην επιθυμητή συμπεριφορά του συστήματος. Σε ορισμένες περιπτώσεις, αυτό οφείλεται στο γεγονός ότι είναι δύσκολο να αποφασιστεί τι πρέπει να μετρηθεί. Για παράδειγμα, κατά την εκτέλεση μιας εργασίας μεταγραφής, πρέπει να μετρηθεί η ακρίβεια του συστήματος κατά τη μεταγραφή ολόκληρων αλληλουχιών ή θα πρέπει να χρησιμοποιηθεί ένα πιο λεπτομερές μέτρο απόδοσης που δίνει μερική πίστωση για τη σωστή διόρθωση ορισμένων στοιχείων αυτών των σειρών; Κατά την εκτέλεση μιας εργασίας παλινδρόμησης, θα πρέπει να τιμωρείται περισσότερο το σύστημα εάν κάνει συχνά μεσαίου μεγέθους λάθη ή αν σπάνια κάνει πολύ μεγάλα λάθη; Αυτά τα είδη επιλογών σχεδιασμού εξαρτώνται από την εφαρμογή.

2.1.3 Η εμπειρία Ε

Οι αλγόριθμοι μηχανικής μάθησης μπορούν να κατηγοριοποιηθούν ευρέως ως **μη εποπτευόμενοι** ή **εποπτευόμενοι** από το είδος της εμπειρίας που τους επιτρέπεται να έχουν κατά τη διάρκεια της μαθησιακής διαδικασίας. Ένα σύνολο δεδομένων είναι μια συλλογή πολλών παραδειγμάτων, όπως ορίζεται στην ενότητα 2.1.1 . Μερικές φορές τα παραδείγματα καλούνται και σημεία δεδομένων.

Ένα από τα παλαιότερα σύνολα δεδομένων που μελετήθηκαν από τους στατιστικολόγους και τους ερευνητές της μηχανικής μάθησης είναι το σύνολο δεδομένων Iris (Fisher, 1936). Πρόκειται για μια συλλογή μετρήσεων διαφορετικών τμημάτων 150 φυτών ίριδας. Κάθε μεμονωμένο φυτό αντιστοιχεί σε ένα παράδειγμα. Τα χαρακτηριστικά σε κάθε παράδειγμα είναι οι μετρήσεις κάθε μέρους του φυτού: το μήκος του σέπαλου, το πλάτος του σέπαλου, το μήκος του πέταλου και το πλάτος του πέταλου. Το σύνολο δεδομένων καταγράφει επίσης σε ποιο είδος ανήκε κάθε φυτό. Τρία διαφορετικά είδη αντιπροσωπεύονται στο σύνολο δεδομένων.

Οι μη εποπτευόμενοι αλγόριθμοι μάθησης αντιμετωπίζουν ένα σύνολο δεδομένων που περιέχει πολλές δυνατότητες και, στη συνέχεια, μαθαίνουν χρήσιμες ιδιότητες της δομής αυτού του συνόλου δεδομένων. Στο πλαίσιο της βαθιάς μάθησης, συνήθως είναι θεμιτό να μαθευτεί ολόκληρη η κατανομή

πιθανότητας που δημιούργησε ένα σύνολο δεδομένων, είτε ρητά όπως στην εκτίμηση πυκνότητας ή σιωπηρά για εργασίες όπως η σύνθεση. Ορισμένοι άλλοι μη εποπτευόμενοι αλγόριθμοι μάθησης εκτελούν άλλους ρόλους, όπως ομαδοποίηση, ο οποίος αποτελείται από την διαίρεση του συνόλου δεδομένων σε ομάδες παρόμοιων παραδειγμάτων.

Οι εποπτευόμενοι αλγόριθμοι μάθησης αντιμετωπίζουν ένα σύνολο δεδομένων που περιέχει χαρακτηριστικά, αλλά κάθε παράδειγμα σχετίζεται επίσης με μία **ετικέτα** (**label**) ή έναν **στόχο**. Για παράδειγμα, το σύνολο δεδομένων Iris σχολιάζεται με το είδος κάθε φυτού ίριδας. Ένας εποπτευόμενος αλγόριθμος μάθησης μπορεί να μελετήσει το σύνολο δεδομένων Iris και να μάθει να ταξινομεί τα φυτά ίριδας σε τρία διαφορετικά είδη με βάση τις μετρήσεις τους.

Παραδοσιακά, οι άνθρωποι αναφέρονται σε προβλήματα παλινδρόμησης, ταξινόμησης και δομημένων αποτελεσμάτων ως εποπτευόμενη μάθηση. Αν και η μη εποπτευόμενη μάθηση και η εποπτευόμενη μάθηση δεν είναι εντελώς τυπικές ή διακριτές έννοιες, βοηθούν στην κατηγοριοποίηση ορισμένων πραγμάτων που γίνονται με τους αλγόριθμους μηχανικής μάθησης.

Ορισμένοι αλγόριθμοι μηχανικής μάθησης δεν αντιμετωπίζουν μόνο ένα σταθερό σύνολο δεδομένων. Για παράδειγμα, οι αλγόριθμοι **ενισχυτικής μάθησης** αλληλεπιδρούν με ένα περιβάλλον, οπότε υπάρχει ένας βρόχος ανατροφοδότησης μεταξύ του συστήματος μάθησης και των εμπειριών του.

2.2 Εποπτευόμενοι αλγόριθμοι μάθησης

Μία από τις πιο αποτελεσματικές προσεγγίσεις στην εποπτευόμενη μάθηση είναι τα Support Vector Machines (SVMs) (Boser, Guyon, & Vapnik, 1992; Cortes & Vapnik, 1995).

Αυτό το μοντέλο είναι παρόμοιο με τη λογιστική παλινδρόμηση στο ότι οδηγείται από μια γραμμική συνάρτηση $w^T x + b$. Σε αντίθεση με την λογιστική παλινδρόμηση, τα SVMs δεν παρέχουν πιθανότητες, αλλά παράγουν μόνο μία κλάση ταυτότητας. Τα SVMs προβλέπουν ότι η θετική κλάση είναι παρούσα όταν το $w^T x + b$ είναι θετικό. Ομοίως, προβλέπουν ότι η αρνητική κλάση είναι παρούσα όταν το $w^T x + b$ είναι αρνητικό. Μια βασική καινοτομία που σχετίζεται με τα SVMs είναι

το κόλπο του πυρήνα (**kernel trick**). Το kernel trick αποτελείται από την παρατήρηση ότι πολλοί αλγόριθμοι μηχανικής μάθησης μπορούν να γραφτούν αποκλειστικά ως προϊόν εσωτερικού γινομένου (dot products) μεταξύ των παραδειγμάτων.

2.3 Μη εποπτευόμενοι αλγόριθμοι μάθησης

Σχεδόν όλη η βαθιά μάθηση τροφοδοτείται από έναν πολύ σημαντικό αλγόριθμο: στοχαστική κατάβαση κλίσης ή stochastic gradient descent (SGD). Ένα επαναλαμβανόμενο πρόβλημα στη μηχανική μάθηση είναι ότι τα μεγάλα σετ δεδομένων εκπαίδευσης είναι απαραίτητα για καλή γενίκευση, αλλά είναι επίσης πιο υπολογιστικά ακριβά. Η συνάρτηση κόστους που χρησιμοποιείται από έναν αλγόριθμο μηχανικής μάθησης αποσυντίθεται συχνά ως άθροισμα των παραδειγμάτων εκπαίδευσης για κάποια συνάρτηση απώλειας ανά παράδειγμα.

Η διορατικότητα του στοχαστικής κλίσης είναι ότι η κλίση είναι μια προσδοκία. Η προσδοκία μπορεί να εκτιμηθεί περίπου χρησιμοποιώντας ένα μικρό σύνολο δειγμάτων. Συγκεκριμένα, σε κάθε βήμα του αλγορίθμου, μπορούμε να δοκιμάσουμε ένα μικρό δείγμα (**minibatch**) παραδειγμάτων αντλούμενα ομοιόμορφα από το σετ δεδομένων εκπαίδευσης. Το μέγεθος του minibatch επιλέγεται συνήθως ως ένας σχετικά μικρός αριθμός παραδειγμάτων, που κυμαίνεται από 1 έως μερικές εκατοντάδες. Συνήθως το μέγεθος του minibatch διατηρείται σταθερό καθώς αυξάνεται το μέγεθος του σετ δεδομένων εκπαίδευσης. Μπορούμε λοιπόν να εκπαιδεύσουμε ένα σετ δεδομένων εκπαίδευσης με δισεκατομμύρια παραδείγματα χρησιμοποιώντας ενημερώσεις που υπολογίζονται σε εκατό παραδείγματα.

2.4 Functions

Ένα πιο συγκεκριμένο παράδειγμα ταξινόμησης είναι η ταξινόμηση εικόνας. Σε αυτό το σημείο θα πρέπει να αναφερθούν κάποιες βασικές συναρτήσεις και έννοιες.

2.4.1 Score function

Το **score function** είναι η συνάρτηση που αντιστοιχίζει τις τιμές εικονοστοιχείων μιας εικόνας σε confidence scores για κάθε κλάση. Ας υποθέσουμε ένα σύνολο δεδομένων εκπαίδευσης εικόνων $x_i \in R^D$, και καθένα σχετίζεται με ένα label y_i. Έχουμε $i = 1 \ldots \mathrm{N}$ και $y_i \in 1 \ldots K$, με το **N** να είναι ο αριθμός των παραδειγμάτων (καθένα με διάσταση **D**) και το **K** να είναι οι ξεχωριστές κατηγορίες. Επομένως, ορίζουμε το score function $f : R^D \rightarrow R^K$ που αντιστοιχίζει τα raw εικονοστοιχεία της εικόνας σε scores κλάσεων.

Linear classifier (ο πιο απλός classifier, ο γραμμικός):

$$\boldsymbol{f(x_i, W, b) = Wx_i + b}$$

Ο πίνακας **W** (με μέγεθος [K x D]), το διάνυσμα **b** (με μέγεθος [K x 1]) είναι οι **παράμετροι** της συνάρτησης. Οι παράμετροι **W** συνήθως καλούνται τα βάρη (**weights**), και το **b** συνήθως καλείται **bias vector** επειδή επηρεάζει τα output scores, αλλά χωρίς την αλληλεπίδραση του με τα x_i.

2.4.2 Loss function

Στο προηγούμενο κεφάλαιο, ορίστηκε το score function από τα εικονοστοιχεία της εικόνας στα scores κλάσεων, τα οποία παραμετροποιούνται από ένα σετ από βάρη W. Επιπλέον, είναι εμφανές ότι δεν έχουμε έλεγχο στα δεδομένα (x_i, y_i) τα οποία είναι σταθερά, αλλά έχουμε έλεγχο πάνω στα βάρη και θέλουμε να τα ορίσουμε έτσι ώστε τα προβλεπόμενα class scores να είναι συνεπή με τα ground truth labels στα δεδομένα εκπαίδευσης.

Επομένως, εάν τα βάρη δεν είναι τα ιδανικά για το συγκεκριμένο σύνολο δεδομένων και τα class scores δεν είναι καλά, τότε θα μετρήσουμε αυτήν την "δυσαρέσκεια" μας με τέτοια αποτελέσματα με ένα **loss function** (ή **cost function** όπως αναφέρεται κάποιες φορές ή και **objective**). Άρα, το loss θα είναι μεγάλο εάν δεν κάνουμε καλή ταξινόμηση του συνόλου δεδομένων εκπαίδευσης και αντιθέτως μικρό εάν κάνουμε καλή ταξινόμηση. Στην ουσία, το loss function ποσοτικοποιεί την "συμφωνία" μεταξύ των προβλεπόμενων scores και των ground truth labels.

Ένας διάσημος ταξινομητής είναι ο **Softmax Classifier**. Στην περίπτωση του ταξινομητή αυτού, η συνάρτηση που αντιστοιχίζει $f(x_i; W) = Wx_i$ παραμένει

αμετάβλητη, αλλά αυτά τα scores ερμηνεύονται ως οι μη ομαλοποιημένες λογαριθμικές πιθανότητες για κάθε κλάση και χρησιμοποιείται το **cross-entropy loss** το οποίο ορίζεται ως εξής:

$$\boldsymbol{L_i} = -\mathbf{log}\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right) \quad \text{ή αλλιώς}$$

$$\boldsymbol{L_i} = -\boldsymbol{f_{y_i}} + \mathbf{log}\sum_j \boldsymbol{e^{f_j}}$$

Το $\boldsymbol{f_j}$ είναι το j[th] στοιχείο του διανύσματος των class scores *f*. Το πλήρως loss για το σύνολο δεδομένων είναι το μέσο όλων των $\boldsymbol{L_i}$ στα σύνολα δεδομένων εκπαίδευσης. Η συνάρτηση $\boldsymbol{f_j(z)} = \frac{e^{z_j}}{\sum_k e^{z_k}}$ ονομάζεται **softmax function**. Η συνάρτηση αυτή παίρνει ένα διάνυσμα αυθαίρετων πραγματικών scores (στο *z*) και το συμπιέζει σε ένα διάνυσμα τιμών μεταξύ του μηδέν και του ενός που αθροίζονται σε ένα.

2.5 Overfitting και Underfitting

Η κεντρική πρόκληση στη μηχανική μάθηση είναι ότι πρέπει να αποδίδουμε καλά σε νέες, προηγουμένως μη παρατηρημένες εισόδους - όχι μόνο σε εκείνες στις οποίες εκπαιδεύτηκε το μοντέλο μας. Η ικανότητα να έχει καλή απόδοση σε προηγούμενες μη παρατηρημένες εισόδους ονομάζεται **γενίκευση**.

Συνήθως, κατά την εκπαίδευση ενός μοντέλου μηχανικής μάθησης, έχουμε πρόσβαση σε ένα **σετ δεδομένων εκπαίδευσης**, μπορούμε να υπολογίσουμε κάποιο μέτρο σφάλματος στο σετ που ονομάζεται σφάλμα εκπαίδευσης και μειώνουμε αυτό το σφάλμα εκπαίδευσης. Αυτό που διαχωρίζει τη μηχανική εκμάθηση από τη βελτιστοποίηση είναι ότι θέλουμε το σφάλμα γενίκευσης, που ονομάζεται επίσης σφάλμα δοκιμής, να είναι επίσης χαμηλό. Το σφάλμα γενίκευσης ορίζεται ως η αναμενόμενη τιμή του σφάλματος σε μια νέα είσοδο. Εδώ η προσδοκία μεταφέρεται σε διάφορες πιθανές εισόδους, που αντλούνται από την κατανομή των εισόδων που περιμένουμε να συναντήσει το σύστημα στην πράξη. Εκτιμούμε τυπικά το σφάλμα γενίκευσης ενός μοντέλου μηχανικής μάθησης μετρώντας την απόδοσή του σε ένα **σετ δεδομένων δοκιμής** παραδειγμάτων που συλλέχθηκαν ξεχωριστά από το σετ δεδομένων εκπαίδευσης.

Όταν χρησιμοποιούμε έναν αλγόριθμο μηχανικής μάθησης, δεν καθορίζουμε τις παραμέτρους εκ των προτέρων και στην συνέχεια κάνουμε δειγματοληψία και τα δύο σύνολα δεδομένων. Κάνουμε δειγματοληψία στο σετ δεδομένων εκπαίδευσης και στη συνέχεια το χρησιμοποιούμε για να επιλέξουμε τις παραμέτρους για να μειώσουμε το σφάλμα εκπαίδευσης και μετά να κάνουμε δειγματοληψία στο σετ δεδομένων δοκιμής. Στο πλαίσιο αυτής της διαδικασίας, το αναμενόμενο σφάλμα δοκιμής είναι μεγαλύτερο ή ίσο με την αναμενόμενη τιμή του σφάλματος εκπαίδευσης. Οι παράγοντες που καθορίζουν πόσο καλά θα αποδώσει ένας αλγόριθμος μηχανικής μάθησης είναι η ικανότητά του να:

1. Κάνει το σφάλμα εκπαίδευσης μικρό.
2. Μειώσει το χάσμα μεταξύ του σφάλματος εκπαίδευσης και δοκιμής.

Αυτοί οι δύο παράγοντες αντιστοιχούν στις δύο κεντρικές προκλήσεις στη μηχανική μάθηση: **underfitting** και **overfitting**. Το underfitting συμβαίνει όταν το μοντέλο δεν είναι σε θέση να αποκτήσει αρκετά χαμηλή τιμή σφάλματος στο σετ δεδομένων δοκιμής. Το overfitting συμβαίνει όταν το χάσμα μεταξύ του σφάλματος εκπαίδευσης και του σφάλματος δοκιμής είναι πολύ μεγάλο.

2.6 Προκλήσεις που ενθαρρύνουν τη βαθιά μάθηση

Οι απλοί αλγόριθμοι μηχανικής μάθησης που περιγράφηκαν στις ενότητες 2.2 και 2.3 αλλά και άλλοι, λειτουργούν πολύ καλά σε μια μεγάλη ποικιλία σημαντικών προβλημάτων. Ωστόσο, δεν κατάφεραν να επιλύσουν τα κεντρικά προβλήματα της τεχνητής νοημοσύνης, όπως η αναγνώριση ομιλίας (speech recognition) ή η αναγνώριση αντικειμένου (object recognition).

Συγκεκριμένα για την αναγνώριση και ανίχνευση αντικειμένου, οι μέθοδοι της εμπίπτουν γενικά είτε σε προσεγγίσεις που βασίζονται σε μηχανική μάθηση είτε σε προσεγγίσεις που βασίζονται στην **βαθιά μάθηση** (τεχνητά νευρωνικά δίκτυα με πολλαπλά επίπεδα μεταξύ των επιπέδων εισόδου και αποτελέσματος). Για προσεγγίσεις μηχανικής μάθησης, καθίσταται αναγκαίο να οριστούν πρώτα χαρακτηριστικά χρησιμοποιώντας μία από τις μεθόδους παρακάτω και μετά να χρησιμοποιηθεί μία τεχνική όπως ο εποπτευόμενος αλγόριθμος μάθησης SVM για την ταξινόμηση. Αντιθέτως, τεχνικές βαθιάς μάθησης είναι ικανά να κάνουν

ανίχνευση αντικειμένου από την αρχή έως το τέλος χωρίς να ορίζουν συγκεκριμένα χαρακτηριστικά και βασίζονται συνήθως σε συνελικτικά νευρωνικά δίκτυα.

- Μέθοδοι μηχανικής μάθησης
    - Πλαίσιο ανίχνευσης αντικειμένων Viola – Jones με βάση τα χαρακτηριστικά του Haar. (Viola & Jones, Rapid object detection using a boosted cascade of simple features, 2001) (Viola, Jones, & others, Robust real-time object detection, 2001)
    - Μετασχηματισμός δυνατοτήτων κλίμακας (SIFT) (Lowe, 1999)
    - Histogram of oriented gradients (HOG) features (Dalal & Triggs, 2005)
- Μέθοδοι βαθιάς μάθησης
    - Region Proposals (R-CNN (Girshick, Donahue, Darrell, & Malik, 2014), Fast R-CNN (Girshick, Fast r-cnn, 2015), Faster R-CNN (Ren, He, Girshick, & Sun, 2015), cascade R-CNN (Pang, et al., 2019).)
    - Single Shot MultiBox Detector (SSD) (Liu, et al., 2016)
    - You Only Look Once (YOLO) (Redmon, Divvala, Girshick, & Farhadi, 2016)
    - Single-Shot Refinement Neural Network for Object Detection (RefineDet) (Zhang, Wen, Bian, Lei, & Li, 2018)
    - Retina-Net (Lin, Goyal, Girshick, He, & Dollár, 2017) (Pang, et al., 2019)
    - Deformable convolutional networks (Dai, et al., 2017)Η ανάπτυξη της βαθιάς μάθησης οφείλεται εν μέρει στην αποτυχία των παραδοσιακών αλγορίθμων να γενικευθούν καλά σε τέτοιες εργασίες τεχνητής νοημοσύνης. Κάποιες από αυτές τις προκλήσεις θα αναφερθούν παρακάτω (Goodfellow, Bengio, & Courville, 2016).

2.6.1 Η κατάρα της διαστατικότητας

Πολλά προβλήματα μηχανικής μάθησης γίνονται εξαιρετικά δυσκολότερα όταν ο αριθμός των διαστάσεων στα δεδομένα είναι μεγάλος. Αυτό το φαινόμενο είναι γνωστό ως η κατάρα της διαστατικότητας. Ιδιαίτερη ανησυχία είναι ότι ο αριθμός των πιθανών διακριτών συνθέσεων ενός συνόλου μεταβλητών αυξάνεται εκθετικά καθώς αυξάνεται ο αριθμός των μεταβλητών. Η κατάρα της διαστατικότητας εμφανίζεται σε πολλά μέρη στην επιστήμη των υπολογιστών, και ιδιαίτερα στην μηχανική μάθηση.

2.6.2 Τοπική ρύθμιση σταθερότητας και ομαλότητας

Προκειμένου να γενικευτούν καλά, οι αλγόριθμοι μηχανικής μάθησης πρέπει να καθοδηγούνται από προηγούμενες πεποιθήσεις (**prior beliefs**) σχετικά με το είδος της συνάρτησης που πρέπει να μάθουν. Πιο ανεπίσημα, μπορούν επίσης να συζητηθούν οι προηγούμενες πεποιθήσεις ως άμεσα επηρεάζοντας την ίδια τη συνάρτηση και ενεργώντας έμμεσα στις παραμέτρους μόνο μέσω της επίδρασής τους στη συνάρτηση.

Επιπρόσθετα, μπορούν να συζητηθούν ανεπίσημα οι προηγούμενες πεποιθήσεις ως εκφρασμένες σιωπηρά, επιλέγοντας αλγόριθμους που είναι προκατειλημμένοι προς την επιλογή κάποιας κατηγορίας συναρτήσεων έναντι άλλης, παρόλο που αυτές οι προκαταλήψεις μπορεί να μην εκφραστούν όσον αφορά την κατανομή πιθανότητας που αντιπροσωπεύει το βαθμό πεποίθησης σε διάφορες συναρτήσεις.

Μεταξύ των πιο ευρέως χρησιμοποιούμενων αυτών «priors» είναι η πεποίθηση ομαλότητας (**smoothness prior**) ή η πεποίθηση τοπικής σταθερότητας (**local constancy prior**). Πολλοί απλούστεροι αλγόριθμοι βασίζονται αποκλειστικά σε αυτό το prior για να γενικεύουν καλά, και ως αποτέλεσμα δεν καταφέρνουν να κλιμακώσουν τις στατιστικές προκλήσεις που εμπλέκονται στην επίλυση εργασιών τεχνητής νοημοσύνης.

2.6.3 Manifold learning

Ένα **πολύπτυγμα** (manifold) είναι μια συνδεδεμένη περιοχή. Μαθηματικά, είναι ένα σύνολο σημείων, που σχετίζεται με μια γειτονιά γύρω από κάθε σημείο.

Από οποιοδήποτε δεδομένο σημείο, το πολύπτυγμα τοπικά φαίνεται να είναι ευκλείδειος χώρος. Στην καθημερινή ζωή, βιώνουμε την επιφάνεια του κόσμου ως 2-D επίπεδο, αλλά στην πραγματικότητα είναι ένα σφαιρικό πολύπτυγμα στον 3-D χώρο. Ο ορισμός μιας γειτονιάς που περιβάλλει κάθε σημείο υποδηλώνει την ύπαρξη μετασχηματισμών που μπορούν να εφαρμοστούν για να μετακινήσουν το πολύπτυγμα από μία θέση σε μία γειτονική θέση. Στο παράδειγμα της επιφάνειας του κόσμου ως πολύπτυγμα, κάποιος μπορεί να περπατήσει βόρεια, νότια, ανατολικά ή δυτικά.

Αν και υπάρχει μια επίσημη μαθηματική έννοια του όρου «πολύπτυγμα», στη μηχανική μάθηση τείνει να χρησιμοποιείται πιο χαλαρά για τον προσδιορισμό ενός συνδεδεμένου συνόλου σημείων που μπορεί να προσεγγιστεί καλά λαμβάνοντας υπόψη μόνο έναν μικρό αριθμό βαθμών ελευθερίας ή διαστάσεων, ενσωματωμένο σε έναν υψηλότερο διαστατικό χώρο. Κάθε διάσταση αντιστοιχεί σε μια τοπική κατεύθυνση μεταβολής.

Πολλά προβλήματα μηχανικής μάθησης φαίνονται απελπιστικά αν περιμένουμε ο αλγόριθμος μηχανικής μάθησης να μάθει συναρτήσεις με ενδιαφέρουσες μεταβολές σε όλο το R^n. Οι αλγόριθμοι μάθησης πολυπτυγμάτων ξεπερνούν αυτό το εμπόδιο υποθέτοντας ότι το μεγαλύτερο μέρος του R^n αποτελείται από μη έγκυρες εισόδους, και ότι ενδιαφέρουσες είσοδοι εμφανίζονται μόνο κατά μήκος μιας συλλογής πολυπτυγμάτων που περιέχουν ένα μικρό υποσύνολο σημείων, με ενδιαφέρουσες μεταβολές στο αποτέλεσμα της συνάρτησης που ανέπτυξε που συμβαίνει μόνο σε κατευθύνσεις που βρίσκονται στο πολύπτυγμα, ή με ενδιαφέρουσες μεταβολές που συμβαίνουν μόνο όταν μετακινούμαστε από το ένα πολύπτυγμα στο άλλο.

3 *Βαθιά μάθηση*

3.1 Τεχνητά νευρωνικά δίκτυα (Artificial Neural Networks)

Οι μέθοδοι μάθησης νευρωνικών δικτύων παρέχουν μια ισχυρή προσέγγιση για την προσέγγιση πραγματικών, διακριτών και διανυσματικών στοχευόμενων συναρτήσεων. Για ορισμένους τύπους προβλημάτων, όπως το να μάθει κανείς να ερμηνεύει σύνθετα πραγματικά δεδομένα αισθητήρα, τα τεχνητά νευρωνικά δίκτυα είναι από τις πιο αποτελεσματικές μεθόδους μάθησης που είναι γνωστές.

Η μελέτη των τεχνητών νευρικών δικτύων (**ΤΝΔ**) έχει εμπνευστεί εν μέρει από την παρατήρηση ότι τα βιολογικά συστήματα μάθησης είναι κατασκευασμένα από πολύπλοκους ιστούς διασυνδεδεμένων νευρώνων. Προσεγγιστικά μιλώντας, τα τεχνητά νευρικά δίκτυα κατασκευάζονται από ένα πυκνά διασυνδεδεμένο σύνολο απλών μονάδων, όπου κάθε μονάδα παίρνει έναν αριθμό από πραγματικές εισόδους (πιθανώς τα αποτελέσματα άλλων μονάδων) και παράγει ένα πραγματικό αποτέλεσμα (που μπορεί να γίνει η είσοδος πολλών άλλων μονάδων).

Η βασική δομή των ΤΝΔ, είναι ένα δίκτυο μικρών μονάδων επεξεργασίας ή κόμβων, τα οποία συνδέονται μεταξύ τους με σταθμισμένες συνδέσεις. Το δίκτυο ενεργοποιείται παρέχοντας μία είσοδο σε μερικούς ή και όλους τους κόμβους και στην συνέχεια αυτή η ενεργοποίηση εξαπλώνεται σε όλο το δίκτυο κατά μήκος των σταθμισμένων συνδέσεων. Υπάρχουν δύο κυρίως είδη ΤΝΔ, αυτά που είναι κυκλικά και αυτά που δεν είναι κυκλικά. Τα κυκλικά συνήθως ονομάζονται ως feedback, recursive, ή recurrent νευρωνικά δίκτυα, ενώ τα μη κυκλικά ως feed-forward νευρωνικά δίκτυα (FNN). Το πιο γνωστό και διαδομένο FNN, είναι το πολυεπίπεδο perceptron (Bishop & others, Neural networks for pattern recognition, 1995).

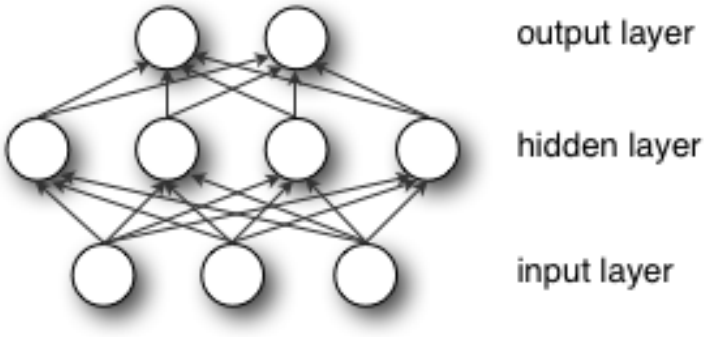


Figure 1 Multilayer Perceptron

3.2 Convolutional Neural Network (Συνελικτικά Νευρωνικά Δίκτυα)

Τα συνελικτικά νευρωνικά δίκτυα (LeCun & others, Generalization and network design strategies, 1989; Goodfellow, Bengio, & Courville, 2016), γνωστά και ως convolutional neural networks ή CNNs, είναι ένα εξειδικευμένο είδος νευρωνικού δικτύου για την επεξεργασία δεδομένων που έχει μια γνωστή τοπολογία τύπου πλέγματος. Παραδείγματα περιλαμβάνουν τα δεδομένα χρονοσειρών, τα οποία μπορούν να θεωρηθούν ως ένα μονοδιάστατο πλέγμα που λαμβάνουν δείγματα σε τακτά χρονικά διαστήματα, και δεδομένα εικόνας, τα οποία μπορούν να θεωρηθούν ως ένα δισδιάστατο πλέγμα εικονοστοιχείων. Το όνομα «convolutional neural network» υποδηλώνει ότι το δίκτυο χρησιμοποιεί μια μαθηματική πράξη που λέγεται **συνέλιξη** (convolution). Η συνέλιξη είναι ένα εξειδικευμένο είδος γραμμικής πράξης. Στην ουσία τα CNNs είναι απλά νευρωνικά δίκτυα που χρησιμοποιούν συνέλιξη αντί του γενικού πίνακα πολλαπλασιασμού σε τουλάχιστον ένα από τα επίπεδα τους.

3.2.1 Η πράξη της συνέλιξης

Στην πιο γενική του μορφή, η συνέλιξη είναι μια πράξη σε δύο συναρτήσεις ενός ορίσματος με πραγματική τιμή. Ας υποθέσουμε ότι παρακολουθούμε τη θέση ενός διαστημικού σκάφους με έναν αισθητήρα λέιζερ. Ο αισθητήρας λέιζερ παρέχει ένα αποτέλεσμα *x(t)*, τη θέση του διαστημικού σκάφους στο χρόνο *t*. Και τα δύο *x* και *t* είναι πραγματικές τιμές, δηλαδή μπορούμε να έχουμε μια διαφορετική τιμή από τον αισθητήρα λέιζερ ανά πάσα στιγμή. Ας υποθέσουμε τώρα ότι ο αισθητήρας λέιζερ μας είναι κάπως θορυβώδης. Για να λάβουμε μια λιγότερο θορυβώδη

εκτίμηση της θέσης του διαστημικού σκάφους, το κάνουμε με μια συνάρτηση στάθμισης *w(a)*, όπου το *a* είναι η ηλικία μιας μέτρησης. Εάν εφαρμοστεί μια τόσο σταθμισμένη μέση πράξη σε κάθε χρονική στιγμή, λαμβάνεται μια νέα συνάρτηση που παρέχει μια ομαλή εκτίμηση της θέσης του διαστημικού σκάφους:

$$s(t) = \int x(a)w(t-a)\,da$$

Αυτή η πράξη λέγεται **συνέλιξη**. Η πράξη της συνέλιξης χαρακτηρίζεται συνήθως με αστερίσκο:

$$s(t) = (x * w)(t)$$

Σε γενικές γραμμές, η συνέλιξη καθορίζεται για οποιεσδήποτε πράξεις για τις οποίες έχει καθοριστεί το παραπάνω ολοκλήρωμα και μπορεί να χρησιμοποιηθεί και για άλλους σκοπούς εκτός από τη λήψη σταθμισμένων μέσων όρων. Στην ορολογία των CNNs, το πρώτο όρισμα της συνέλιξης συχνά αναφέρεται ως είσοδος και το δεύτερο όρισμα ως πυρήνας. Στις εφαρμογές μηχανικής μάθησης, η είσοδος είναι συνήθως ένας πολυδιάστατος πίνακας δεδομένων και ο πυρήνας είναι συνήθως ένας πολυδιάστατος πίνακας παραμέτρων που προσαρμόζονται από τον αλγόριθμο μάθησης.

3.2.2 Αρχιτεκτονική

Η βάση για τα CNNs (LeCun, Bottou, Bengio, & Haffner, 1998; Bishop, 2006) ήταν η προσέγγιση για την δημιουργία μοντέλων που είναι αμετάβλητα σε ορισμένες μετατροπές των εισόδων, έτσι ώστε να οικοδομήσουν τις ιδιότητες αμεταβλητότητας στη δομή ενός νευρωνικού δικτύου. Αυτή η προσέγγιση έχει εφαρμοστεί ευρέως σε δεδομένα εικόνας.

Ας αναλογιστούμε την συγκεκριμένη εργασία της αναγνώρισης χειρόγραφων ψηφίων. Κάθε εικόνα εισόδου περιλαμβάνει ένα σύνολο τιμών έντασης εικονοστοιχείων και το επιθυμητό αποτέλεσμα είναι μια μεταγενέστερη κατανομή πιθανότητας στις δέκα κλάσεις ψηφίων (0-9). Είναι γνωστό ότι η ταυτότητα του ψηφίου είναι αμετάβλητη σε μεταφράσεις, σε κλιμάκωση καθώς και (μικρές) περιστροφές. Μια απλή προσέγγιση θα ήταν να αντιμετωπιστεί η εικόνα ως είσοδο

σε ένα πλήρως συνδεδεμένο δίκτυο, και εξασφαλίζοντας του ένα αρκετά μεγάλο σύνολο δεδομένων εκπαίδευσης να δώσει μια καλή λύση στο πρόβλημα.

Ωστόσο, αυτή η προσέγγιση αγνοεί μια βασική ιδιότητα των εικόνων, δηλαδή ότι τα κοντινά εικονοστοιχεία συσχετίζονται πιο έντονα από τα πιο απομακρυσμένα εικονοστοιχεία. Πολλές από τις πιο πρόσφατες προσεγγίσεις εκμεταλλεύονται αυτήν την ιδιότητα εξάγοντας τοπικά χαρακτηριστικά που εξαρτώνται μόνο από μικρές υπό-περιοχές της εικόνας. Οι πληροφορίες από αυτά τα χαρακτηριστικά μπορούν στην συνέχεια να συγχωνευτούν σε μεταγενέστερα στάδια επεξεργασίας προκειμένου να ανιχνευθούν χαρακτηριστικά υψηλότερης τάξης και τελικά να αποδοθούν πληροφορίες για την εικόνα στο σύνολό της. Επίσης, τα τοπικά χαρακτηριστικά που είναι χρήσιμα σε μια περιοχή της εικόνας είναι πιθανό να είναι χρήσιμα σε άλλες περιοχές της εικόνας, για παράδειγμα εάν το αντικείμενο ενδιαφέροντος μεταφράζεται.

Αυτές οι έννοιες ενσωματώνονται στα CNNs μέσω τριών μηχανισμών : *τοπικά δεκτικά πεδία*, *κατανομή βάρους* και υπό-*δειγματοληψία*. Η δομή ενός CNN απεικονίζεται στο Figure 2. Στο συνελικτικό επίπεδο (Convolution layer) οι μονάδες οργανώνονται σε επίπεδα, καθένα από τα οποία ονομάζονται *feature maps*. Οι μονάδες σε ένα feature map λαμβάνουν κάθε είσοδο μόνο από μια μικρή υπό-περιοχή της εικόνας και όλες οι μονάδες σε ένα feature map περιορίζονται στο να μοιράζονται τις ίδιες τιμές βάρους. Για παράδειγμα, ένα feature map μπορεί να αποτελείται από 100 μονάδες διατεταγμένες σε πλέγμα *10×10*, με κάθε μονάδα να λαμβάνει εισόδους από μία *5×5* επιφάνεια εικονοστοιχείων της εικόνας.

Ολόκληρο το feature map επομένως έχει 25 ρυθμιζόμενες παραμέτρους βάρους συν μία παράμετρο ρυθμιζόμενης μεροληψίας (bias). Οι τιμές εισόδου από μία επιφάνεια συνδυάζονται γραμμικά χρησιμοποιώντας τα βάρη και την μεροληψία, και το αποτέλεσμα μετασχηματίζεται από μία σιγμοειδή μη γραμμική ενεργοποίηση. Αν σκεφτούμε τις μονάδες ως ανιχνευτές χαρακτηριστικών, τότε όλες οι μονάδες σε ένα feature map ανιχνεύουν το ίδιο μοτίβο αλλά σε διαφορετικά σημεία στην εικόνα εισόδου. Λόγω της κατανομής βάρους, η αξιολόγηση των ενεργοποιήσεων αυτών των μονάδων είναι ισοδύναμη με μια συνέλιξη των

εντάσεων των εικονοστοιχείων εικόνας με έναν «πυρήνα» που περιλαμβάνει τις παραμέτρους βάρους. Εάν η εικόνα εισόδου μετατοπιστεί, οι ενεργοποιήσεις του feature map θα μετατοπιστούν κατά το ίδιο ποσό, αλλά θα παραμείνουν αμετάβλητες.

Επειδή συνήθως θα χρειαστεί να εντοπιστούν πολλά χαρακτηριστικά για να δημιουργήσουν ένα αποτελεσματικό μοντέλο, γενικά θα υπάρχουν πολλά feature maps στο συνελικτικό επίπεδο, καθένα από τα οποία έχει το δικό του σύνολο παραμέτρων βάρους και μεροληψίας. Τα αποτελέσματα των συνελικτικών μονάδων σχηματίζουν τις εισόδους στο επίπεδο υπό-δειγματοληψίας του δικτύου.

Για κάθε feature map στο συνελικτικό επίπεδο, υπάρχει ένα επίπεδο μονάδων στο επίπεδο υπό-δειγματοληψίας και κάθε μονάδα λαμβάνει εισόδους από ένα μικρό δεκτικό πεδίο στο αντίστοιχο feature map του συνελικτικού επιπέδου. Αυτές οι μονάδες εκτελούν υπό-δειγματοληψία. Για παράδειγμα, κάθε μονάδα υπό-δειγματοληψίας μπορεί να λάβει εισόδους από μία *2×2* περιοχή μονάδων στο αντίστοιχο feature map και θα υπολογίσει τον μέσο όρο αυτών των εισόδων, πολλαπλασιαζόμενο με ένα προσαρμοστικό βάρος με την προσθήκη μιας παραμέτρου προσαρμοστικής μεροληψίας, και στη συνέχεια μετασχηματίζεται χρησιμοποιώντας μια σιγμοειδή μη γραμμική συνάρτηση ενεργοποίησης. Τα δεκτικά πεδία επιλέγονται να είναι συνεχόμενα και μη αλληλεπικαλυπτόμενα έτσι ώστε να υπάρχει ο μισός αριθμός σειρών και στηλών στο επίπεδο υπό-δειγματοληψίας σε σύγκριση με το συνελικτικό επίπεδο. Με αυτόν τον τρόπο, η απόκριση μιας μονάδας στο επίπεδο υπό-δειγματοληψίας θα είναι σχετικά μη ευαίσθητη σε μικρές μετατοπίσεις της εικόνας στις αντίστοιχες περιοχές του χώρου εισόδου.

Σε μια πρακτική αρχιτεκτονική, μπορεί να υπάρχουν αρκετά ζεύγη συνελικτικών επιπέδων και επιπέδων υπό-δειγματοληψίας. Σε κάθε στάδιο υπάρχει μεγαλύτερος βαθμός αμεταβλητότητας μετασχηματισμού εισόδου σε σύγκριση με το προηγούμενο επίπεδο. Μπορεί να υπάρχουν διάφορα feature maps σε ένα δεδομένο συνελικτικό επίπεδο για κάθε επίπεδο μονάδων στο προηγούμενο επίπεδο υπό-δειγματοληψίας, έτσι ώστε η σταδιακή μείωση της χωρικής ανάλυσης

να αντισταθμίζεται από έναν αυξανόμενο αριθμό χαρακτηριστικών. Το τελικό επίπεδο του δικτύου θα ήταν τυπικά ένα πλήρως συνδεδεμένο, πλήρως προσαρμοσμένο επίπεδο, με αποτέλεσμα *softmax* μη γραμμικότητας στην περίπτωση ταξινόμησης πολλαπλών κλάσεων.

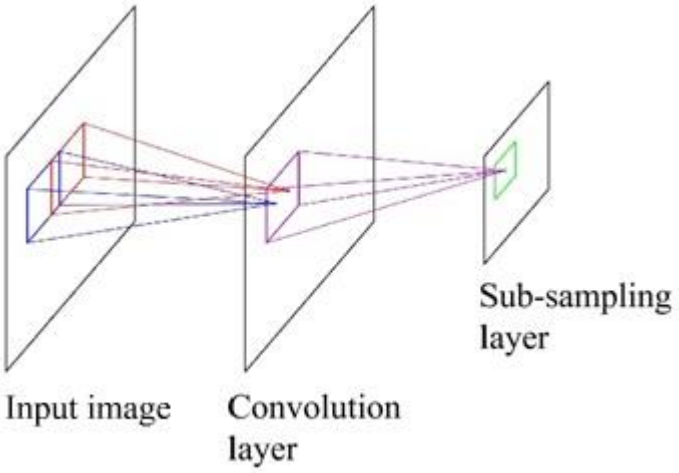


Figure 2 Διάγραμμα που απεικονίζει μέρος του CNN, δείχνοντας ένα επίπεδο συνελικτικών μονάδων και ακολουθεί ένα επίπεδο από μονάδες υπό-δειγματοληψίας. Μπορούν να χρησιμοποιηθούν πολλά διαδοχικά ζεύγη τέτοιων επιπέδων.

3.3 Dropout

Regularization είναι η διαδικασία προσθήκης πληροφοριών για την αποτροπή του overfitting και χρησιμοποιείται σε objective functions σε προβλήματα βελτιστοποίησης**.** Μία τεχνική του Regularization είναι το **Dropout**.

Η τεχνική dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014) χρησιμοποιείται για την αποτροπή του overfitting και παρέχει έναν τρόπο συνδυασμού εκθετικά πολλών διαφορετικών αρχιτεκτονικών νευρωνικών δικτύων αποτελεσματικά. Ο όρος «dropout» αναφέρεται σε εγκατάλειψη μονάδων (κρυφών και ορατών) σε ένα νευρωνικό δίκτυο. Με την εγκατάλειψη μιας μονάδας, εννοείται η προσωρινή κατάργησή της από το δίκτυο, μαζί με όλες τις εισερχόμενες και εξερχόμενες συνδέσεις του, όπως φαίνεται στο Figure 3. Η επιλογή των μονάδων που θα εγκαταλειφτούν είναι τυχαία. Στην απλούστερη περίπτωση, κάθε μονάδα διατηρείται με μια σταθερή πιθανότητα *p* ανεξάρτητη από άλλες μονάδες, όπου το *p* μπορεί να επιλεγεί χρησιμοποιώντας ένα σύνολο δεδομένων δοκιμής ή μπορεί

απλά να οριστεί στο 0.5, το οποίο φαίνεται να είναι σχεδόν βέλτιστο για ένα ευρύ φάσμα δικτύων και εργασιών. Ωστόσο, για τις μονάδες εισόδου, η βέλτιστη πιθανότητα διατήρησης είναι συνήθως πιο κοντά στο 1 παρά στο 0,5.

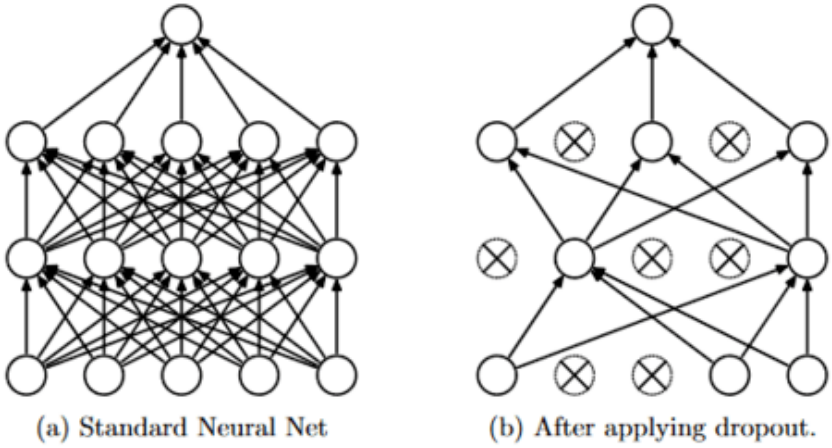


Figure 3 Dropout μοντέλο νευρωνικού δικτύου. Αριστερά: είναι ένα τυπικό νευρωνικό δίκτυο με δύο κρυφά επίπεδα.Δεξιά: είναι ένα παράδειγμα αραιωμένου δικτύου που παράχθηκε με την εφαρμογή dropout στο δίκτυο στα αριστερά. Οι διασταυρούμενες μονάδες εγκαταληφθεί.

3.4 Rectified Linear Unit (Relu)

Ο τυπικός τρόπος (Krizhevsky, Sutskever, & Hinton, 2012) μοντελοποίησης του αποτελέσματος ενός νευρώνα f ως συνάρτηση της εισόδου του x γίνεται με $f(x) = \tanh(x)$ ή $f(x) = (1 + e^{-x})^{-1}$ (**sigmoid**). Όσον αφορά τον χρόνο εκπαίδευσης με κατάβαση κλίσης, αυτές οι κορεσμένες μη γραμμικότητες είναι πολύ πιο αργές από τη μη κορεσμένη γραμμικότητα $f(x) = \max(0, x)$. Με βάση (Nair & Hinton, 2010), αναφερόμαστε σε νευρώνες με αυτή τη μη γραμμικότητα ως Rectified Linear Units (ReLUs). Τα βαθιά CNNs με ReLUs εκπαιδεύονται πολλές φορές γρηγορότερα από τα ισοδύναμα τους με μονάδες $\tan h$.

3.5 Recurrent Neural Network

Τα Επαναλαμβανόμενα νευρωνικά δίκτυα (Recurrent neural network or RNN) (Rumelhart, Hinton, & Williams, 1986) (Goodfellow, Bengio, & Courville, 2016) είναι ένα είδος νευρωνικών δικτύων για την επεξεργασία αλληλουχίας δεδομένων. Όπως και τα CNN είναι ένα εξειδικευμένο είδος νευρωνικού δικτύου για την επεξεργασία ενός πλέγματος τιμών ***X*** όπως μία εικόνα, ένα RNN είναι ένα νευρωνικό δίκτυο

ειδικό για την επεξεργασία μιας αλληλουχίας τιμών $x^{(1)},\ldots,x^{(\tau)}$. Τα περισσότερα RNN μπορούν να επεξεργαστούν αλληλουχίες μεταβλητού μήκους.

Η μετάβαση από τα δίκτυα πολλαπλών επιπέδων σε RNN, στηρίχτηκε στο πλεονέκτημα από τις αρχικές ιδέες της μηχανικής μάθησης και των στατιστικών μοντέλων: την κοινή χρήση παραμέτρων σε διάφορα μέρη ενός μοντέλου. Η κοινή χρήση παραμέτρων καθιστά δυνατή την επέκταση και την εφαρμογή του μοντέλου σε παραδείγματα διαφορετικού μήκους και στην γενίκευση τους.

Ένα πρόβλημα που αναδύεται, είναι ότι οι κλίσεις που διαδίδονται σε πολλά στάδια τείνουν είτε να εξαφανίζονται είτε να εκρήγνυνται. Ακόμα και εάν ειπωθεί ότι οι παράμετροι είναι τέτοιες ώστε το RNN να είναι σταθερό, η δυσκολία με μακροπρόθεσμες εξαρτήσεις (**long-term dependencies**) προκύπτει από τα εκθετικά μικρότερα βάρη που δίδονται σε μακροπρόθεσμες αλληλεπιδράσεις σε σύγκριση με τις βραχυπρόθεσμες. Αυτό το πρόβλημα, που αφορά συγκεκριμένα τα RNN, ονομάζεται και **Vanishing and exploding gradient problem** (Bengio, Simard, & Frasconi, 1994; Pascanu, Mikolov, & Bengio, 2013).

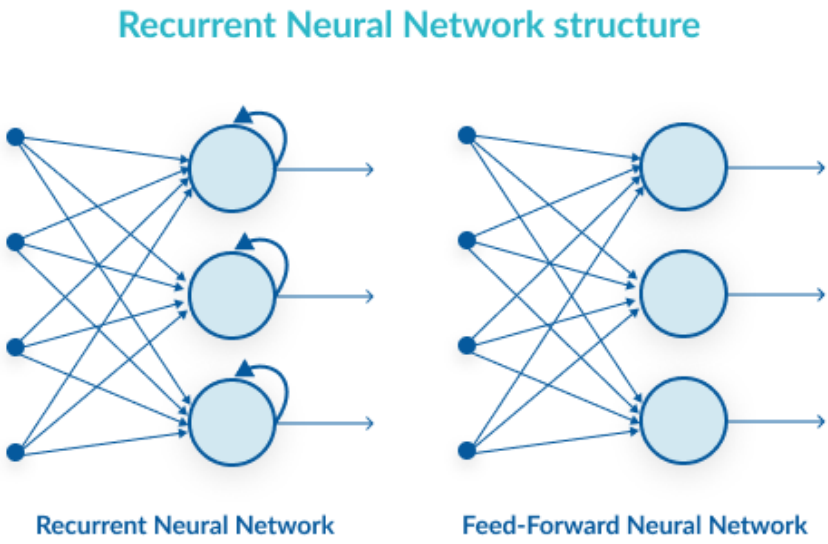


Figure 4 (LEFT): RNN architecture, (RIGHT) FNN architecture

3.5.1 Long-short term memory (LSTM)

Η πιο αποτελεσματική λύση στο *Vanishing and exploding gradient problem*, είναι η αρχιτεκτονική του **Long Short Term Memory (LSTM)** (Hochreiter & Schmidhuber, 1997).

3.5.1.1 Αρχιτεκτονική

Σύμφωνα και με (Graves, 2012), η αρχιτεκτονική ενός LSTM αποτελείται από ένα σύνολο επαναλαμβανόμενων συνδεδεμένων υπό-δικτύων, γνωστά και ως μπλοκ μνήμης (**memory blocks)**. Κάθε μπλοκ περιέχει ένα ή περισσότερα αυτό-συνδεδεμένα (self-connected) κελιά μνήμης και τρεις πολλαπλασιαστικές μονάδες — τα input, output και forget gates — που παρέχουν συνεχή ανάλογα λειτουργιών εγγραφής, ανάγνωσης και επαναφοράς για τα κελιά. Μια απεικόνιση ενός μπλοκ μνήμης LSTM με ένα μόνο κελί παρέχει το Figure 5.

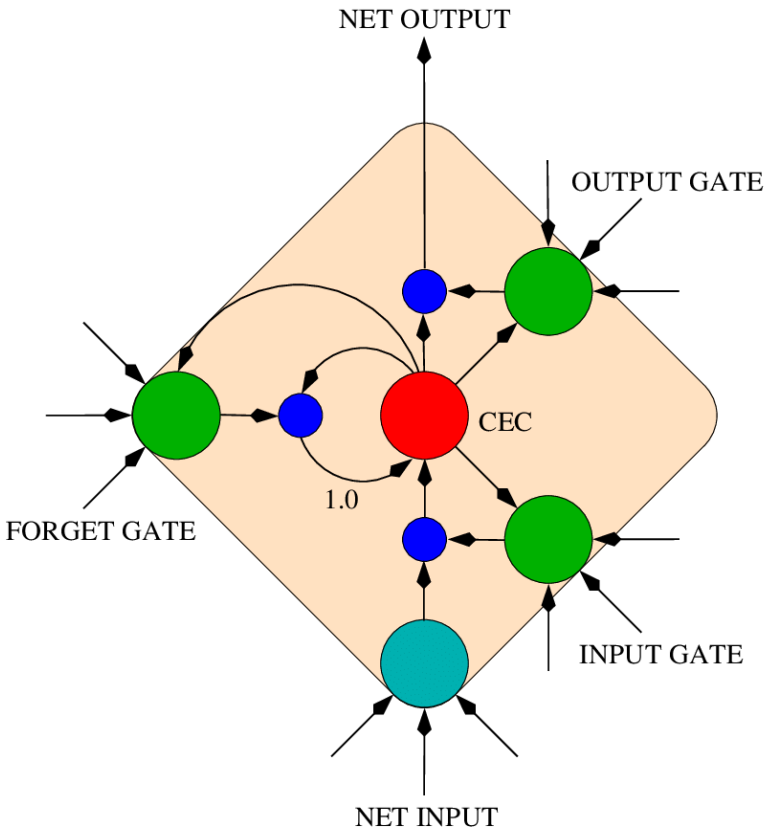


Figure 5 LSTM memory block with one cell. The internal state of the cell is maintained with a recurrent connection of fixed weight 1.0. The three gates collect activations from inside and outside the block, and control the cell via multiplicative units (small circles). The input and output gates scale the input and output of the cell while the forget gate scales the internal state. The cell input and output activation functions (g and h) are applied at the indicated places.

Ένα δίκτυο LSTM σχηματίζεται ακριβώς όπως ένα απλό RNN, με την διαφορά ότι οι μη γραμμικές μονάδες στο κρυφό επίπεδο αντικαθίστανται από μπλοκ μνήμης. Επίσης, όπως και με άλλα RNN, το κρυφό επίπεδο μπορεί να συνδεθεί με οποιονδήποτε τύπο διαφοροποιήσιμου επιπέδου αποτελέσματος (output layer), ανάλογα με την απαιτούμενη εργασία (παλινδρόμηση, ταξινόμηση κ.λπ.).

Οι πολλαπλασιαστικές πύλες επιτρέπουν στα κελιά μνήμης LSTM να αποθηκεύουν και να έχουν πρόσβαση σε πληροφορίες για μεγάλα χρονικά διαστήματα, αποφεύγοντας έτσι το *Vanishing gradient problem.*

3.6 Markov model

Ως Markov μοντέλο ονομάζεται το μοντέλο στο οποίο η επόμενη κατάσταση εξαρτάται μόνο από την τρέχουσα. Μαθηματικά, η υπό όρους κατανομή πιθανότητας εξαρτάται από την τρέχουσα κατάσταση και όχι από τις προηγούμενες καταστάσεις. Επιπλέον ονομάζεται και ως 1^{st}-order Markov μοντέλο.

Τα Markov chains (Shannon, 1948) είναι μαθηματικά συστήματα που μεταπηδούν από μία κατάσταση στην άλλην. Για παράδειγμα, εάν υλοποιηθεί ένα Markov chain μοντέλο της συμπεριφοράς ενός ανθρώπου, θα μπορούσε να συμπεριληφθεί το “να φάει”, το “να κοιμηθεί” και το “να ασκηθεί” ως καταστάσεις, οι οποίες σε συνδυασμό και με άλλες συμπεριφορές θα δημιουργούσαν μία κατάσταση χώρου (**state space**), δηλαδή μία λίστα όλων των πιθανών καταστάσεων.

Επιπλέον, τα Markov chains ενημερώνουν για την πιθανότητα μεταπήδησης ή μετάβασης (**transition**) από την μία κατάσταση στην άλλη. Ένα απλό Markov chain δύο καταστάσεων παρουσιάζεται παρακάτω:

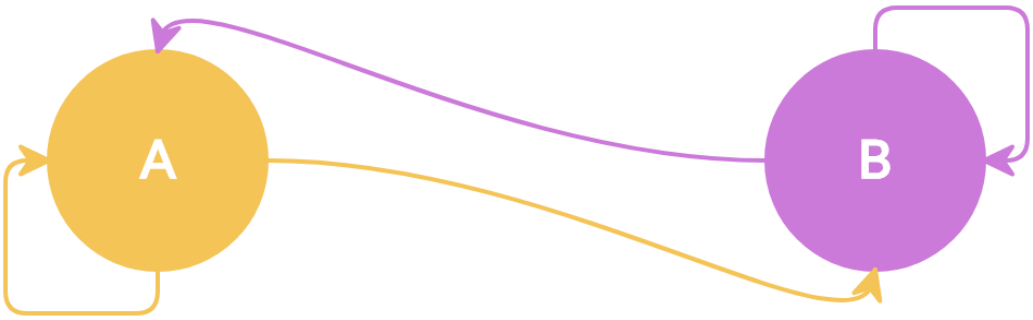


Figure 6

Με δύο καταστάσεις A και B στην κατάσταση χώρου, υπάρχουν τέσσερα πιθανά transitions (αντί για δύο είναι τέσσερα λόγω του ότι μία κατάσταση μπορεί να κάνει transition στον εαυτό της). Αν κάποιος είναι στο A, μπορεί να κάνει transition στο B ή να μείνει στο A. Στο συγκεκριμένο παράδειγμα, το οποίο είναι ένα 1^{st}-order Markov μοντέλο, η πιθανότητα να κάνεις transition από την μία κατάσταση στην άλλην είναι 0.5.

Στην πράξη, συνήθως, χρησιμοποιούνται τα “transition matrix” για τον υπολογισμό των πιθανοτήτων μετάβασης. Κάθε κατάσταση στην κατάσταση χώρου περιλαμβάνεται μία φορά ως σειρά και ξανά ως στήλη και κάθε κελί στο matrix υποδεικνύει την πιθανότητα να κάνεις transition από την κατάσταση της γραμμής του στην κατάσταση της στήλης του. Με συνέπεια, τα κελιά να κάνουν την ίδια δουλεία με το τα βέλη στο Figure 6.

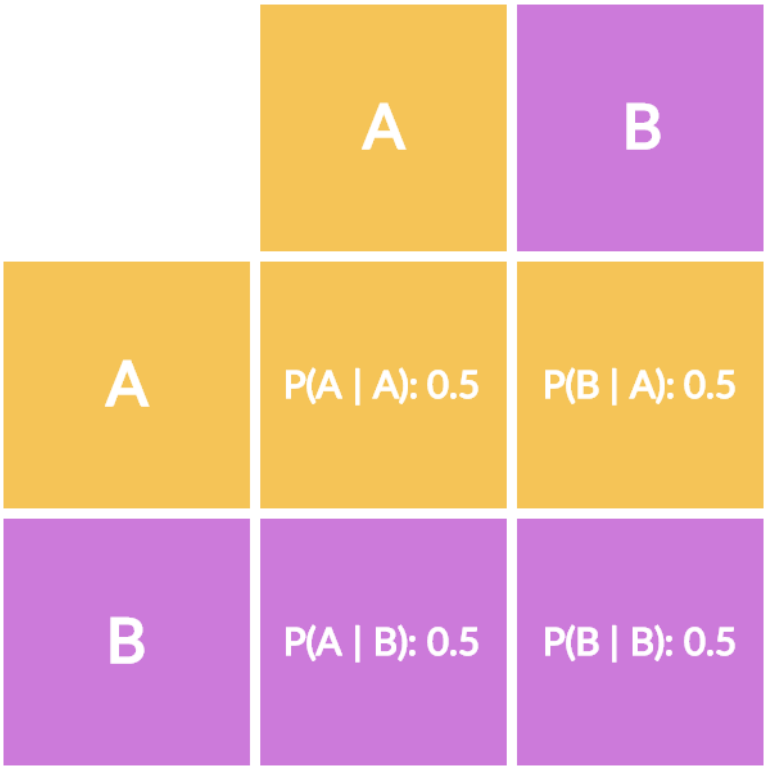


Figure 7

Ουσιαστικά, εάν η παρούσα κατάσταση βασίζεται σε *n* προηγούμενες καταστάσεις τότε ονομάζεται και ως *n-th* order Markov μοντέλο. Για παράδειγμα σε ένα 2nd-order Markov μοντέλο, η παρούσα κατάσταση εξαρτάται ΜΟΝΟ από τις δύο προηγούμενες χωρίς να επηρεάζεται από τις υπόλοιπες καταστάσεις στην κατάσταση χώρου. Συνεπώς, μία αλληλουχία γεγονότων που ακολουθούν το μοντέλο Markov συντελεί το Markov chain.

Συγκεκριμένα, στην εφαρμογή της πρόβλεψης της επόμενης λέξης, κάθε λέξη αντιμετωπίζεται ως κατάσταση και η πρόβλεψη επιτυγχάνεται με βάση την προηγούμενη κατάσταση.

3.7 Language modeling

Μια σημαντική εφαρμογή των Markov models είναι η δημιουργία στατιστικών γλωσσικών μοντέλων (**statistical language models**), τα οποία είναι κατανομές πιθανότητας σε αλληλουχίες λέξεων (Murphy, 2012). Ορίζεται το state space ως

όλες οι λέξεις στα Αγγλικά ή σε κάποια άλλη γλώσσα. Εάν χρησιμοποιηθεί ένα 1st-order Markov μοντέλο, τότε ονομάζεται **bigram model.** Εάν χρησιμοποιηθεί ένα 2nd-order Markov μοντέλο, τότε ονομάζεται **trigram model**. Και ούτω καθεξής. Γενικότερα αυτά τα μοντέλα λέγονται **n-gram models**.

Ένα άλλο είδος language model, εκτός του στατιστικού, είναι και το νευρωνικό. Πρόκειται για ένα γλωσσικό μοντέλο που βασίζεται σε νευρωνικά δίκτυα, αξιοποιώντας την ικανότητά τους να μάθουν κατανεμημένες παραστάσεις για να μειώσουν το αντίκτυπο της κατάρας της διαστατικότητας.

Τα language models, μπορούν να χρησιμοποιηθούν για πολλά πράγματα, όπως τα εξής:

- **Ολοκλήρωση πρότασης** Ένα γλωσσικό μοντέλο μπορεί να προβλέψει την επόμενη λέξη δεδομένου των προηγούμενων λέξεων σε μια πρόταση.
- **Αυτόματη συγγραφή δοκιμίων** Παραγωγή τεχνητού κειμένου.
- **Ταξινόμηση κειμένου** Οποιοδήποτε μοντέλο πυκνότητας μπορεί να χρησιμοποιηθεί ως πυκνότητα υπό όρους και ως εκ τούτου μετατρέπεται σε (παραγωγική) ταξινόμηση.

Στα πλαίσια της εργασίας, το Markov model και το LSTM μοντέλο χρησιμοποιούνται για την πρόβλεψη των επόμενων 15 λέξεων με βάση ένα input seed που θα έχουν τον ρόλο των προηγούμενων λέξεων.

4 *Βιβλιοθήκες*

4.1 Τεχνολογικό υπόβαθρο

Όπως έχει προαναφερθεί, ο σκοπός της παρούσας εργασίας είναι η δημιουργία ενός μοντέλου για την αναγνώριση χειρόγραφων χαρακτήρων, όπου θα χρησιμοποιηθεί για την αποτελεσματική αναγνώριση χειρόγραφου κειμένου μίας εικόνας και επιπλέον η εικόνα αυτή θα είναι και το αποτέλεσμα το οποίο θα χρησιμοποιηθεί στην πρόβλεψη της επόμενης λέξης ή λεξεων. Για την επίτευξη του παραπάνω στόχου, η μελέτη χωρίστηκε σε δύο στάδια, καθένα από τα οποία απαιτούσε λεπτομερή προσοχή σε κάθε βήμα με σκοπό τη βέλτιστη λειτουργία και απόδοση του αλγορίθμου τόσο σε επίπεδο χρόνου εκτέλεσης αλλά και ποιοτικών αποτελεσμάτων.

Αρχικά, πραγματοποιήθηκε η εκμάθηση μοντέλου αναγνώρισης χαρακτήρων με χρήση του Συνελικτικού Νευρωνικού Δικτύου (Convolutional Neural Network), στη συνέχεια η προ-επεξεργασία της εικόνας εισόδου και αναγνώριση του κειμένου της εικόνας. Επακόλουθα, για το στάδιο της πρόβλεψης της επόμενης λέξης ή λέξεων, πραγματοποιήθηκε ένα 2^{nd}-order Markov μοντέλο και ένα Long short-term memory (LSTM) μοντέλο, τα οποία λαμβάνουν ως είσοδο το αποτέλεσμα του προηγούμενου σταδίου για την πρόβλεψη. Τέλος, γίνεται μια σύγκριση των αποτελεσμάτων των δύο μοντέλων ως προς τον χρόνο εκτέλεσης και την συντακτική λογική τους.

4.1.1 Σύνολα Δεδομένων

Το σύνολο δεδομένων που χρησιμοποιήθηκε για το στάδιο της εκμάθησης του μοντέλου αναγνώρισης χαρακτήρων, είναι από την ιστοσελίδα ‘www.kaggle.com’ με όνομα ‘A-Z Handwritten Alphabets in .csv format’. Το σύνολο δεδομένων αυτό περιέχει 26 φάκελους (A-Z) περιέχοντας χειρόγραφες εικόνες με μέγεθος 28*28

εικονοστοιχείων, κάθε αλφάβητος στην εικόνα είναι κεντραρισμένο σε 20*20 εικονοστοιχεία και κάθε εικόνα είναι αποθηκευμένη στο επίπεδο του γκρι.

Το σύνολο δεδομένων που χρησιμοποιήθηκε για το στάδιο της πρόβλεψης της επόμενης λέξης, δημιουργήθηκε από την συσσώρευση πολλαπλών λογοτεχνικών συγγραμμάτων από την ιστοσελίδα 'www.Gutenberg.org', το οποίο παρέχει δωρεάν eBooks, σε ένα ενιαίο t.x.t αρχείο 'data.txt'. Επειδή τα λογοτεχνικά συγγράμματα ως έχουν, περιέχουν αρκετό θόρυβο (ονομασία κεφαλαίων κ.α.), προ-επεξεργάζονται κρατώντας μόνο τα κομμάτια κειμένου. Έπειτα, γίνεται αντικατάσταση όλων των '!','?',':' συμβόλων σε '.', όλων των '--', ';', ' " ' συμβόλων σε ' <κενό> ' και αφαίρεση κενών γραμμών για την καλύτερη εκμάθηση των μοντέλων. Τελικά, το σύνολο δεδομένων αυτό, φτάνει να έχει 23706 προτάσεις και 261514 λέξεις, το οποίο είναι σχετικά μικρό για language modeling εργασίες.

Μερικά παραδείγματα:

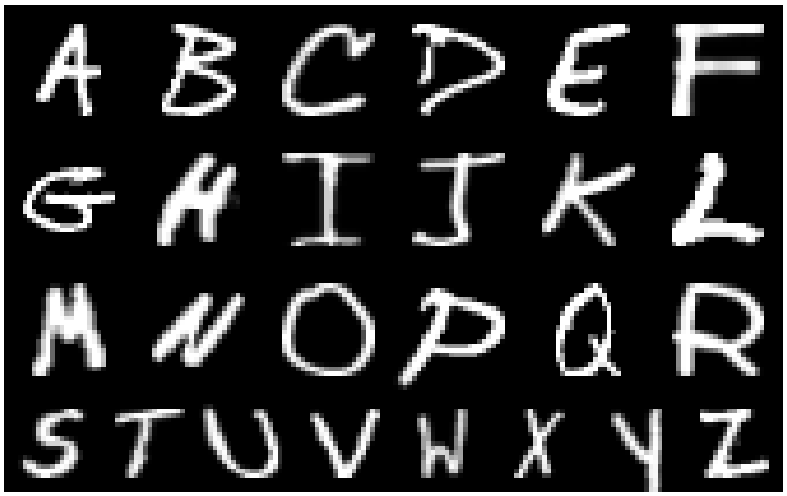


Figure 8: A-Z Handwritten Alphabets in .csv format

Happy families are all alike, every unhappy family is unhappy in its own way.
Everything was in confusion in the Oblonskys house. The wife had discovered that the husband was carrying on an intrigue with a French girl, who had been a governess in their family, and she had announced to her husband that she could not go on living in the same house with him. This position of affairs had now lasted three days, and not only the husband and wife themselves, but all the members of their family and household, were painfully conscious of it. Every person in the house felt that there was no sense in their living together, and that the stray people brought together by chance in any inn had more in common with one another than they, the members of the family and household of the Oblonskys. The wife did not leave her own room, the husband had not been at home for three days. The children ran wild all over the house, the English governess quarreled with the housekeeper, and wrote to a friend asking her to look out for a new situation for her, the man-cook had walked off the day before just at dinner time, the kitchen-maid, and the coachman had given warning.
Three days after the quarrel, Prince Stepan Arkadyevitch Oblonsky—Stiva, as he was called in the fashionable world—woke up at his usual hour, that is, at eight oclock in the morning, not in his wifes bedroom, but on the leather-covered sofa in his study. He turned over his stout, well-cared-for person on the springy sofa, as though he would sink into a long sleep again, he vigorously embraced the pillow on the other side and buried his face in it, but all at once he jumped up, sat up on the sofa, and opened his eyes.
Yes, yes, how was it now. he thought, going over his dream. Now, how was it. To be sure. Alabin was giving a dinner at Darmstadt, no, not Darmstadt, but something American. Yes, but then, Darmstadt was in America. Yes, Alabin was giving a dinner on glass tables, and the tables sang, Il mio tesoro—not Il mio tesoro though, but something better, and there were some sort of little decanters on the table, and they were women, too, he remembered.

Figure 9 'data.txt' sample sentences.

4.1.2 Τεχνολογίες

Για την διεκπεραίωση της εργασίας, χρησιμοποιήθηκε το open source εργαλείο Anaconda, το οποίο είναι μια δωρεάν και ανοιχτή διανομή των γλωσσών προγραμματισμού Python και R για επιστημονικούς υπολογιστές που στοχεύει στην απλούστευση της διαχείρισης και ανάπτυξης πακέτων.

Figure 10: Anaconda

Επιπλέον, το Jupyter Notebook, το οποίο είναι μια διαδικτυακή εφαρμογή ανοιχτού κώδικα που επιτρέπει να δημιουργείτε και να μοιράζεστε έγγραφα που περιέχουν ζωντανό κώδικα, εξισώσεις, απεικονίσεις και αφηγηματικό κείμενο, ήταν χρήσιμο για την δημιουργία των scripts για όλα τα στάδια.

Figure 11: Jupyter Notebook

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε ήταν η python. Η επιλογή της γλώσσας αυτής έγινε με βάση την εκτεταμένη συλλογή βιβλιοθηκών και πλαισίων, την απλότητα του, την αφθονία υποστήριξης και κυρίως τον χρόνο απόδοσης του.

Figure 12: Python

Οι προδιαγραφές για τα μοντέλα που δημιουργήθηκαν στον προσωπικό μου υπολογιστή ήταν:

- Windows edition : Windows 10 Pro
- Processor : AMD Ryzen 5 1600 Six-Core Processor 3.50 Hz
- RAM : 8.00 GB

- System type : 64-bit
- Graphics Card : NVIDIA GeForce GTX 1060 6GB

Πιο συγκεκριμένα οι βιβλιοθήκες/modules που χρησιμοποιήθηκαν στα python scripts θα αναλυθούν στο επόμενο κεφάλαιο.

4.2 Python

Αρχικά, μία πολύ σημαντική βιβλιοθήκη της python για την αποτελεσματική προ-επεξεργασία της εικόνας εισόδου συντέλεσε το *OpenCV* (Pulli, Baksheev, Kornyakov, & Eruhimov, 2012). Το OpenCV *(Open Source Computer Vision)* είναι μια βιβλιοθήκη λειτουργιών προγραμματισμού που στοχεύει κυρίως σε πραγματικού χρόνου computer – vision. Αναπτύχθηκε από την Intel, υποστηρίχθηκε αργότερα από το Willow Garage και τώρα διατηρείται από τον Itseez. Η βιβλιοθήκη είναι πολλαπλής πλατφόρμας και είναι δωρεάν για χρήση υπό την άδεια ανοικτού κώδικα BSD. Το OpenCV χρησιμοποιεί το Numpy, το οποίο είναι μια εξαιρετικά βελτιστοποιημένη βιβλιοθήκη για αριθμητικές λειτουργίες με σύνταξη τύπου MATLAB. Όλες οι δομές πίνακα OpenCV μετατρέπονται σε και από συστοιχίες Numpy. Αυτό διευκολύνει επίσης την ενσωμάτωση με άλλες βιβλιοθήκες που χρησιμοποιούν το Numpy όπως το SciPy και το Matplotlib.

Επιπλέον, για την εκμετάλλευση της ταχύτητας της κάρτας γραφικών, χρησιμοποιήθηκε και η βιβλιοθήκη NVIDIA cuDNN, η οποία είναι μια GPU-accelerated βιβλιοθήκη πρωτόγονων για βαθιά νευρωνικά δίκτυα. Το cuDNN παρέχει πολύ συντονισμένες εφαρμογές για τυπικές ρουτίνες όπως συνέλιξη μπρος ή πίσω, pooling, ομαλοποίηση και επίπεδα ενεργοποίησης.

Επιπροσθέτως, μία πλατφόρμα που χρησιμοποιήθηκε ήταν το TensorFlow. Το TensorFlow είναι μια ολοκληρωμένη πλατφόρμα ανοιχτού κώδικα για μηχανική μάθηση. Διαθέτει ένα ολοκληρωμένο, ευέλικτο οικοσύστημα εργαλείων, βιβλιοθηκών και κοινοτικών πόρων που επιτρέπει στους ερευνητές να προωθήσουν την τελευταία λέξη της τεχνολογίας στη μηχανική μάθηση και οι προγραμματιστές να δημιουργούν και να αναπτύσσουν εύκολα εφαρμογές μηχανικής μάθησης. Επίσης, το Keras είναι άλλη μια πολύ σημαντική βιβλιοθήκη. Πιο συγκεκριμένα, το

Keras είναι ένα API βαθιάς μάθησης γραμμένο στην γλώσσα προγραμματισμού Python, που τρέχει πάνω από το TensorFlow.

4.3 Modules

Σύντομη περιγραφή των modules που χρησιμοποιήθηκαν στα scripts.

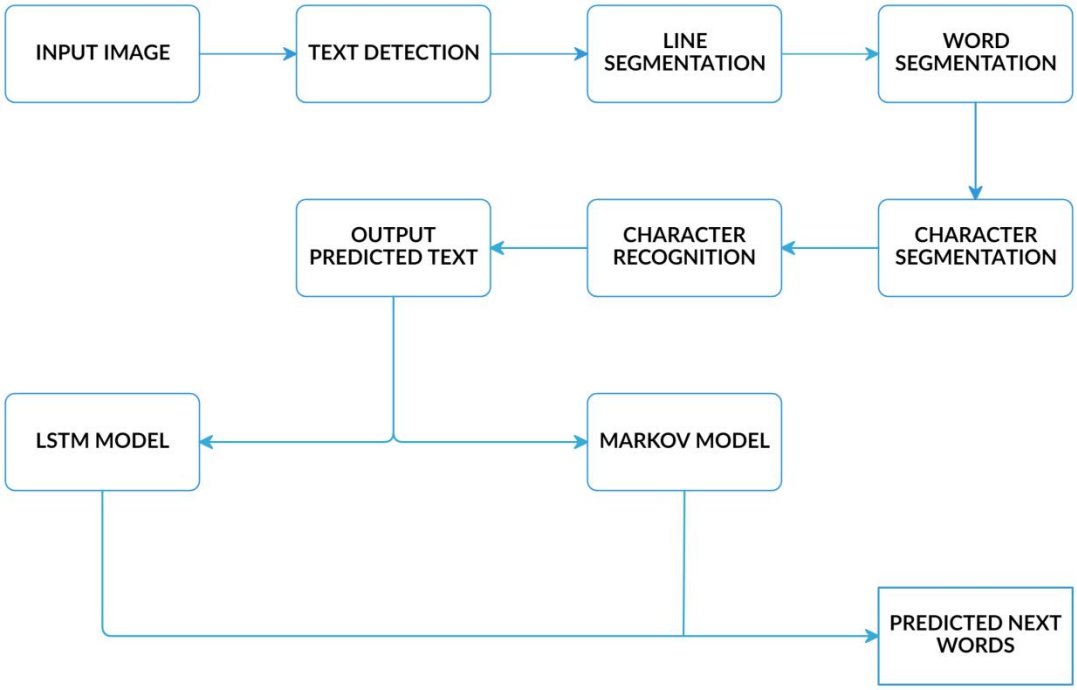
- **from heapq import nlargest:** Επιστρέφει μια λίστα με τα N μεγαλύτερα στοιχεία από το σύνολο δεδομένων που ορίζεται.
- **from keras.callbacks import ModelCheckpoint:** Κλήση για να αποθηκευτεί το μοντέλο Keras ή τα βάρη του μοντέλου σε κάποια συχνότητα.
- **from keras.layers import Dense:** Το κανονικό πυκνά συνδεδεμένο επίπεδο νευρωνικού δικτύου.
- **from keras.layers import Dropout:** Εφαρμογή Dropout στην είσοδο.
- **from keras.layers import Embedding:** Μετατρέπει τους θετικούς ακέραιους αριθμούς (ευρετήρια) σε πυκνά διανύσματα σταθερού μεγέθους. Αυτό το επίπεδο μπορεί να χρησιμοποιηθεί μόνο ως το πρώτο επίπεδο σε ένα μοντέλο.
- **from keras.layers import CuDNNLSTM:** Γρήγορη εκτέλεση LSTM που υποστηρίζεται από το cuDNN.
- **from keras.layers import Flatten:** Ισοπεδώνει την είσοδο. Δεν επηρεάζει το μέγεθος του batch size.
- **from keras.layers.convolutional import Conv2D:** 2D επίπεδο συνέλιξης (π.χ. χωρική συνέλιξη πάνω σε εικόνες).
- **from keras.layers.convolutional import MaxPooling2D:** Max pooling (υπό-δειγματοληπτική διαδικασία) για 2D χωρικά δεδομένα.
- **from keras.models import Sequential:** Το Sequential μοντέλο είναι μια γραμμική στοίβα επιπέδων όπου κάθε επίπεδο έχει ακριβώς έναν tensor εισόδου και ένα tensor εξόδου.
- **from keras.optimizers import Adam:** Optimizer που εφαρμόζει τον αλγόριθμο Adam. Η βελτιστοποίηση Adam είναι μία μέθοδος

stochastic gradient descent που βασίζεται σε προσαρμοστική εκτίμηση των στιγμών πρώτης τάξης και δεύτερης τάξης.

- **from keras.preprocessing.sequence import pad_sequences:** Padding αλληλουχιών στο ίδιο μήκος.
- **from keras.preprocessing.text import Tokenizer:** Κλάση βοηθητικού προγράμματος tokenization κειμένου.
- **from keras.utils import np_utils:** Μετατρέπει ένα διάνυσμα κλάσης (ακέραιοι αριθμοί) σε δυαδικό matrix κλάσης.
- **from random import randint:** Επιστρέφει έναν τυχαίο ακέραιο N.
- **from scipy import ndimage:** Με την μέθοδο *ndimage.center_of_mass* υπολογίζεται το κέντρο μάζας των τιμών ενός πίνακα στις ετικέτες.
- **import cv2:** Εισαγωγή της βιβλιοθήκης OpenCV.
- **import keras:** Εισαγωγή της βιβλιοθήκης Keras.
- **import math:** Παρέχει πρόσβαση στις μαθηματικές συναρτήσεις που ορίζονται στο πρότυπο C.
- **import numpy as np:** Εισαγωγή της βιβλιοθήκης Numpy.
- **import pickle:** Εκτελεί δυαδικά πρωτόκολλα για σειριοποίηση και από-σειριοποίηση μιας δομής αντικειμένου Python.
- **import string:** Η μέθοδος *string.punctuation* είναι μια συμβολοσειρά χαρακτήρων ASCII που θεωρούνται χαρακτήρες στίξης στις τοπικές ρυθμίσεις C.
- **import tensorflow as tf:** Εισαγωγή της πλατφόρμας TensorFlow.

5 *Μεθοδολογία*

Η επισκόπηση των βασικών πρακτικών σταδίων που ακολουθήθηκαν στη παρούσα εργασία περιγράφονται με την ακόλουθη αλληλουχία (pipeline):



Η ιδέα είναι να εισαχθεί μια εικόνα, να ανιχνευθεί κείμενο σε αυτήν, να ταξινομηθεί με γραμμές, λέξεις και τελικά με χαρακτήρες και να αναγνωριστούν. Η αναγνώριση επιτυγχάνεται με την εκπαίδευση ενός CNN μοντέλου. Όταν αναγνωριστούν όλοι οι χαρακτήρες, λαμβάνεται αυτό το αποτέλεσμα και χρησιμοποιείται ως είσοδο σε δύο διαφορετικά μοντέλα. Αυτά τα μοντέλα είναι ένα 2^{nd}–order μοντέλο Markov και ένα μοντέλο LSTM για την πρόβλεψη της επόμενης λέξης ή των επόμενων λέξεων.

- Τα δύο μοντέλα (Markov model και LSTM) εκπαιδεύονται με το ίδιο σύνολο δεδομένων.

5.1 Εκμάθηση μοντέλου αναγνώρισης χαρακτήρων με χρήση του CNN

A. Εισαγωγή των απαραίτητων modules.

B. Αρχικοποίηση μιας τιμής που θα χρησιμοποιηθεί για τον τυχαίο διαχωρισμό δεδομένων εκπαίδευσης και δοκιμής, ώστε να μπορούν να αναπαραχθούν τα ίδια αποτελέσματα.

C. Φόρτωση του συνόλου δεδομένων 'A_Z Handwritten Data.csv' και διαχώριση σε X (είσοδος) και Y (έξοδος).

D. Διαχώριση του συνόλου δεδομένων σε δεδομένα εκπαίδευσης και δεδομένα δοκιμής.

E. Ανασχηματισμός των καινούργιων δεδομένων στην κατάλληλη μορφή που απαιτείται από τον Conv2D.

F. Ανάμειξη των δεδομένων με την αλλαγή της θέσης τους για το στοιχείο της τυχαιότητας στην εκμάθηση του μοντέλου.

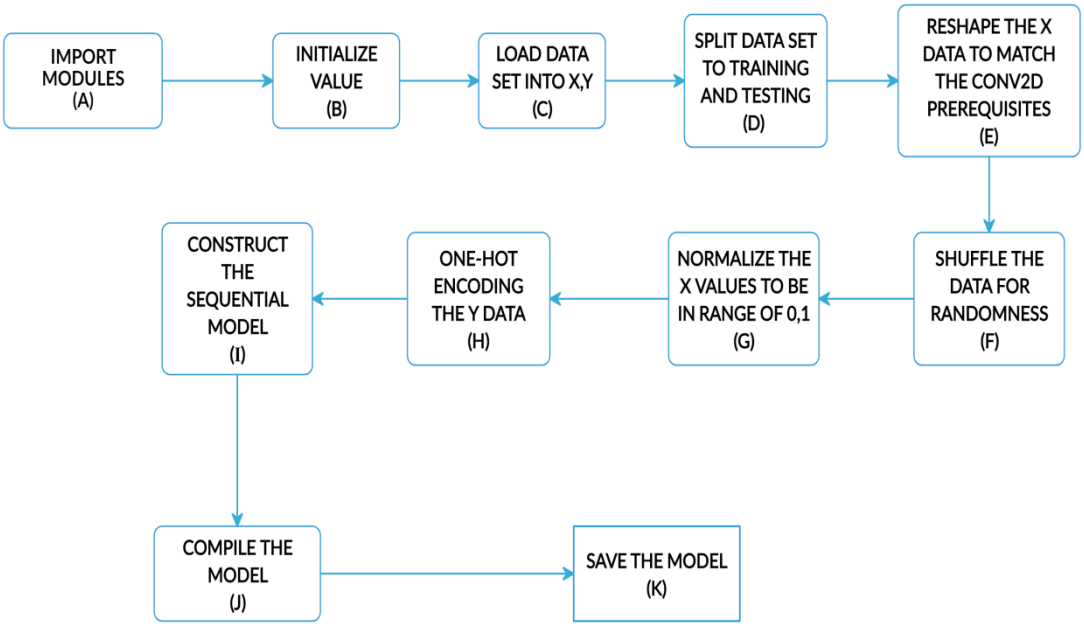
G. Ομαλοποίηση των δεδομένων έτσι ώστε κάθε τιμή εικονοστοιχείου να κυμαίνεται από 0 έως 1, και διαίρεση με το 255 διότι οι τιμές είναι από 0 έως 255.

H. One-Hot Encoding ώστε οι τιμές Y να είναι σε μια πιο φιλική μορφή για τον υπολογιστή, διότι περιέχουν τα πραγματικά γράμματα A-Z.

I. Κατασκευή του Sequential μοντέλου.

J. Μεταγλωττισμός του μοντέλου.

K. Αποθήκευση του μοντέλου στο σύστημα.


IMPORT MODULES (A)
INITIALIZE VALUE (B)
LOAD DATA SET INTO X,Y (C)
SPLIT DATA SET TO TRAINING AND TESTING (D)
RESHAPE THE X DATA TO MATCH THE CONV2D PREREQUISITES (E)
SHUFFLE THE DATA FOR RANDOMNESS (F)
NORMALIZE THE X VALUES TO BE IN RANGE OF 0,1 (G)
ONE-HOT ENCODING THE Y DATA (H)
CONSTRUCT THE SEQUENTIAL MODEL (I)
COMPILE THE MODEL (J)
SAVE THE MODEL (K)

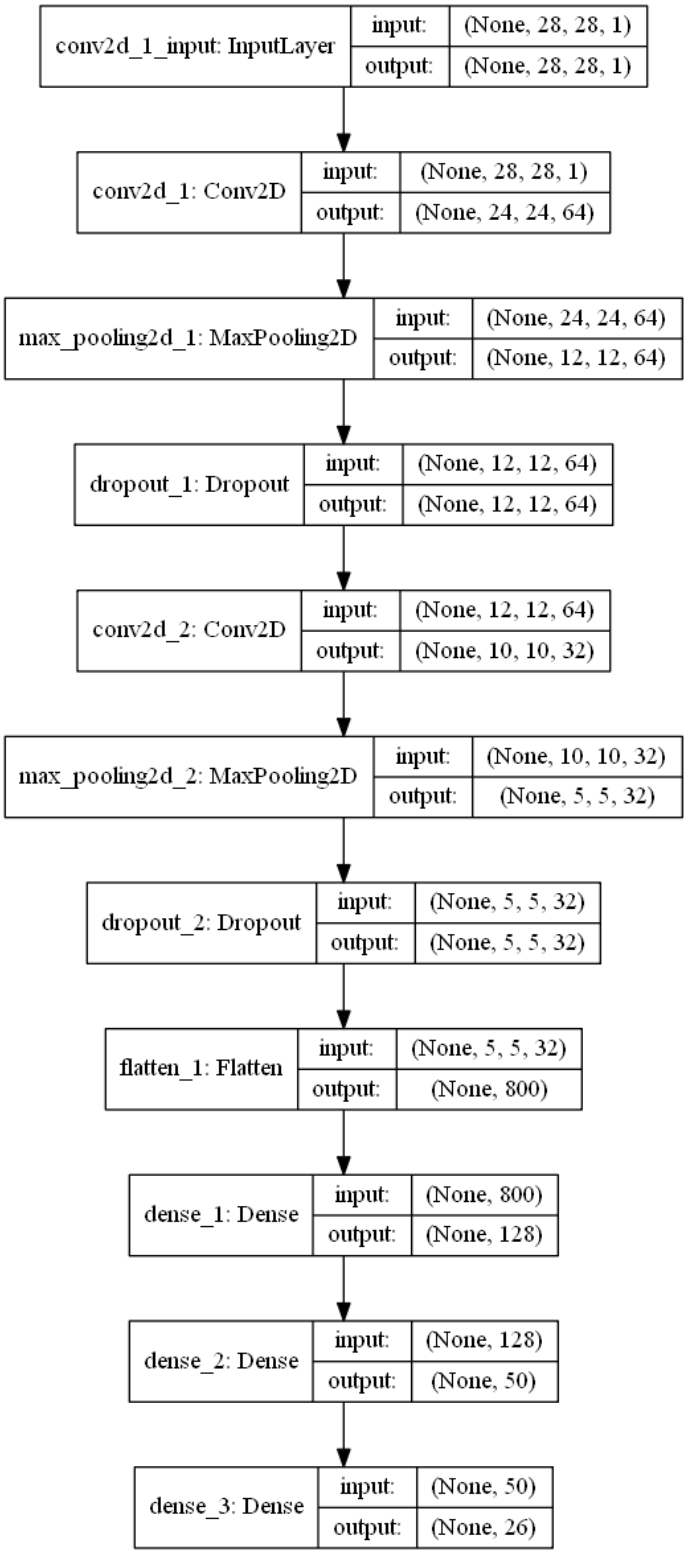
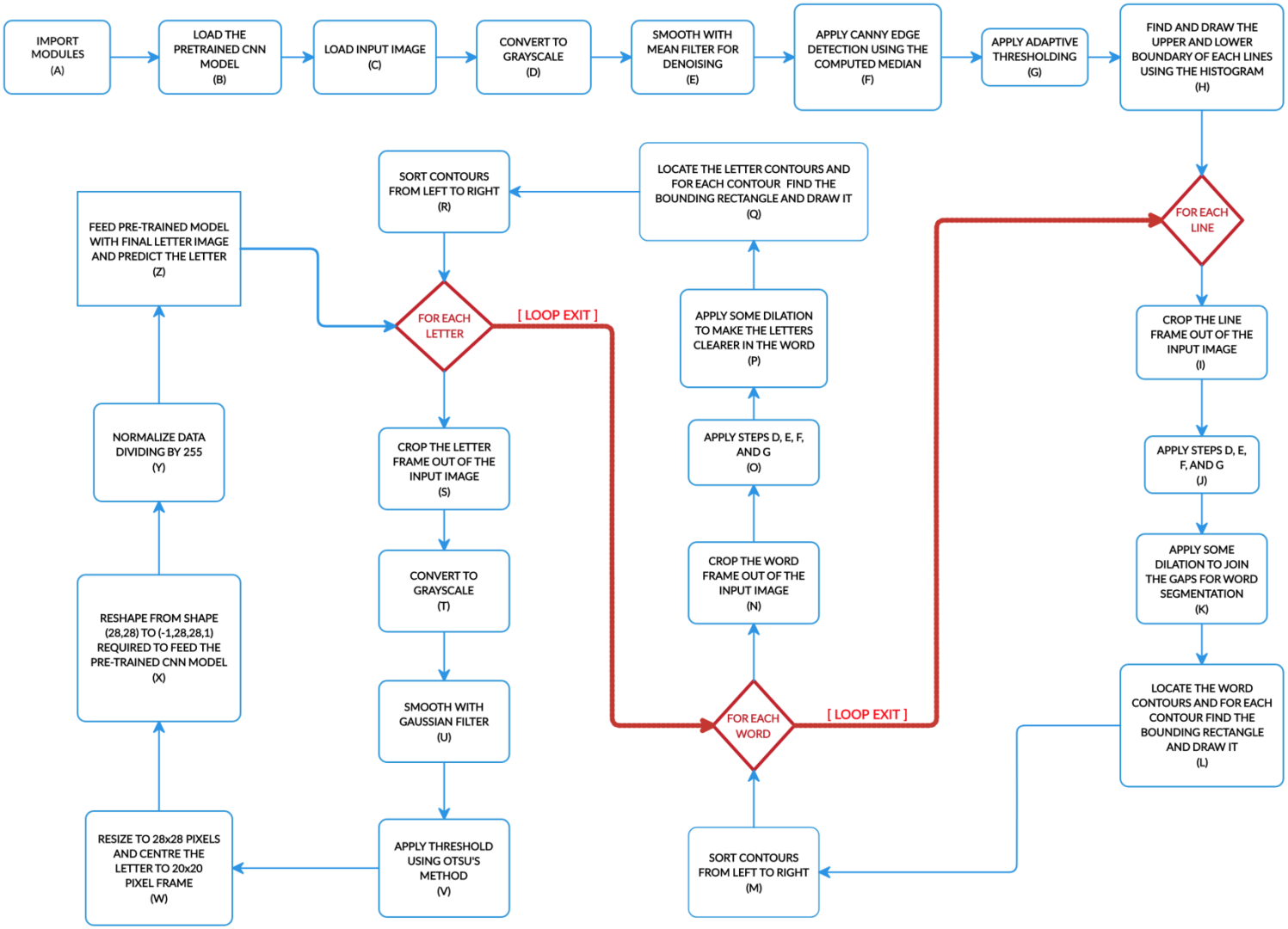


Figure 13 CNN Sequential Model

5.2 Προ-επεξεργασία της εικόνας εισόδου και αναγνώριση του κειμένου της εικόνας

A. Εισαγωγή των απαραίτητων modules.
B. Φόρτωση του εκπαιδευμένου CNN μοντέλου με την βοήθεια του module pickle.
C. Φόρτωση της επιλεγμένης εικόνας και αποθήκευση της σε μεταβλητή.
D. Μετατροπή της εικόνας σε κλίμακα του γκρι.
E. Θόλωση της εικόνας χρησιμοποιώντας φίλτρο μέσου για τυχόν θόρυβο.
F. Εφαρμογή αυτόματης ανίχνευσης ακμής Canny χρησιμοποιώντας τον υπολογισμένο μέσο.
G. Εφαρμογή προσαρμοστικού κατωφλιού (adaptive threshold) για μετατροπή της εικόνας από κλίμακα του γκρι σε μια δυαδική εικόνα.
H. Με βάση το ιστόγραμμα της εικόνας, εντοπίζεται και σχεδιάζεται το άνω και κάτω όριο κάθε γραμμής με αποτέλεσμα να χωριστούν τα περιεχόμενα της εικόνας σε γραμμές.
I. Για κάθε γραμμή, γίνεται περικοπή των πλαισίων της γραμμής της αρχικής εικόνας δημιουργώντας καινούργιες εικόνες (κάθε γραμμή είναι και μία νέα εικόνα).
J. Για κάθε τέτοια εικόνα, γίνεται εφαρμογή των τεχνικών που αναφέρθηκαν στο D, E, F, G.
K. Εφαρμογή κάποιας διαστολής για την ένωση των κενών ανάμεσα των χαρακτήρων, ώστε να εμφανιστούν και να διαχωριστούν οι λέξεις μέσα στην γραμμή.
L. Για κάθε περίγραμμα λέξης εντοπίζεται η ορθογώνια οριοθέτηση και σχεδιάζεται (γίνεται έλεγχος για την μη σχεδίαση των ψευδών θετικών που δεν είναι κείμενο π.χ. κουκκίδες στο χαρτί).
M. Ταξινόμηση των περιγραμμάτων από αριστερά προς τα δεξιά για την ταξινόμηση των λέξεων με την σειρά ανά γραμμή.
N. Σε αυτό το σημείο για κάθε λέξη, γίνεται περικοπή των πλαισίων της λέξης της αρχικής εικόνας δημιουργώντας καινούργιες εικόνες (κάθε λέξη είναι και μία νέα εικόνα).
O. Για κάθε τέτοια εικόνα, γίνεται εφαρμογή των τεχνικών που αναφέρθηκαν στο D, E, F, G.
P. Εφαρμογή κάποιας διαστολής για πιο εμφανή γράμματα στην λέξη.
Q. Για κάθε περίγραμμα γράμματος εντοπίζεται η ορθογώνια οριοθέτηση και σχεδιάζεται (γίνεται έλεγχος για την μη σχεδίαση των ψευδών θετικών που δεν είναι κείμενο π.χ. κουκκίδες στο χαρτί).
R. Ταξινόμηση των περιγραμμάτων από αριστερά προς τα δεξιά για την ταξινόμηση των γραμμάτων με την σειρά ανά λέξη.

S. Για κάθε γράμμα, γίνεται περικοπή των πλαισίων του γράμματος της αρχικής εικόνας δημιουργώντας καινούργιες εικόνες (κάθε γράμμα είναι και μία νέα εικόνα) και πρέπει να γίνει ρύθμιση στα πρότυπα του συνόλου δεδομένων MNIST.
T. Για κάθε εικόνα, γίνεται μετατροπή στην κλίμακα του γκρι.
U. Θόλωση της εικόνας χρησιμοποιώντας ένα Gaussian φίλτρο.
V. Εφαρμογή κατωφλιού (threshold) με συνδυασμό την μέθοδο του Otsu, για την μετατροπή της εικόνας από κλίμακα του γκρι σε μια δυαδική εικόνα.
W. Τροποποίηση του μεγέθους της εικόνας σε 28x28 εικονοστοιχεία και κεντράρισμα της εικόνας ώστε το γράμμα να είναι στο κέντρο της εικόνας όπως στο σύνολο δεδομένων MNIST σε ένα πλαίσιο 20x20.
X. Ανασχηματισμός ξανά την εικόνα από μέγεθος (28, 28) σε μέγεθος (1, 28, 28, 1) χωρίς να γίνει αλλαγή των δεδομένων.
Y. Ομαλοποίηση των δεδομένων έτσι ώστε κάθε τιμή εικονοστοιχείου να κυμαίνεται από 0 έως 1, και διαίρεση με το 255 διότι οι τιμές είναι από 0 έως 255.
Z. Τέλος, τροφοδότηση του προ-εκπαιδευμένου μοντέλου με την τελική μορφή της εικόνας και γίνεται πρόβλεψη του γράμματος για κάθε μία από τις εικόνες.

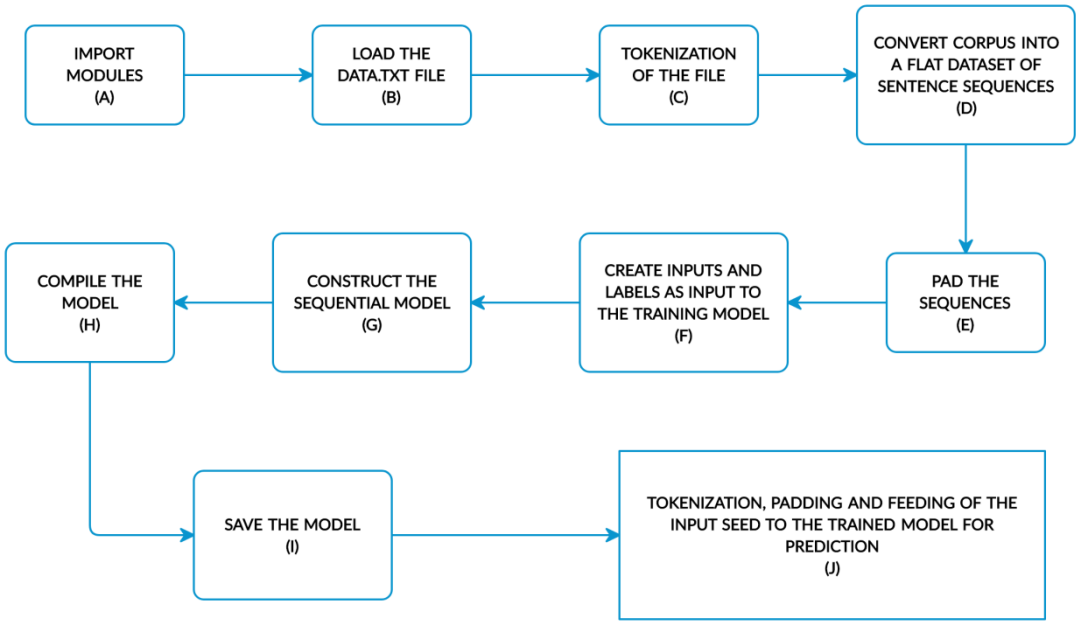


Μηχανική μάθηση για την αναγνώριση και πρόβλεψη χειρόγραφου κειμένου

Στέλιο Μπομπάϊ

5.3 LSTM

A. Εισαγωγή των απαραίτητων modules.
B. Φόρτωση του συνόλου δεδομένων ‘data.txt’.
C. Διεξαγωγή tokenization του αρχείου ‘data.txt’, δηλαδή η διαδικασία εξαγωγής των tokens (όροι / λέξεις) από ένα corpus.
D. Μετατροπή του corpus σε ένα επίπεδο σύνολο αλληλουχίας προτάσεων.
E. “Γέμισμα” (padding) των αλληλουχιών, με αποτέλεσμα εάν υπάρχουν διαφορετικές αλληλουχίες με διαφορετικά μήκη, να εξισωθούν.
F. Δημιουργία εισόδων και ετικετών από τις αλληλουχίες ως είσοδο στο μοντέλο εκμάθησης. Παραδείγματος χάρη για την αλληλουχία “This is a nice day”, θα οριστεί ως “This -> is”, “This is -> a”, “This is a -> nice” και “This is a nice -> day” όπου το αριστερό μέρος είναι οι είσοδοι και το δεξιά μέρος είναι οι ετικέτες.
G. Κατασκευή του Sequential μοντέλου.
H. Μεταγλωττισμός του μοντέλου
I. Αποθήκευση του μοντέλου στο σύστημα
J. Διεξαγωγή tokenization, padding του input seed και τροφοδότηση του στο εκπαιδευμένο μοντέλο για την πρόβλεψη των επόμενων λέξεων.


IMPORT MODULES (A)
LOAD THE DATA.TXT FILE (B)
TOKENIZATION OF THE FILE (C)
CONVERT CORPUS INTO A FLAT DATASET OF SENTENCE SEQUENCES (D)
PAD THE SEQUENCES (E)
CREATE INPUTS AND LABELS AS INPUT TO THE TRAINING MODEL (F)
CONSTRUCT THE SEQUENTIAL MODEL (G)
COMPILE THE MODEL (H)
SAVE THE MODEL (I)
TOKENIZATION, PADDING AND FEEDING OF THE INPUT SEED TO THE TRAINED MODEL FOR PREDICTION (J)

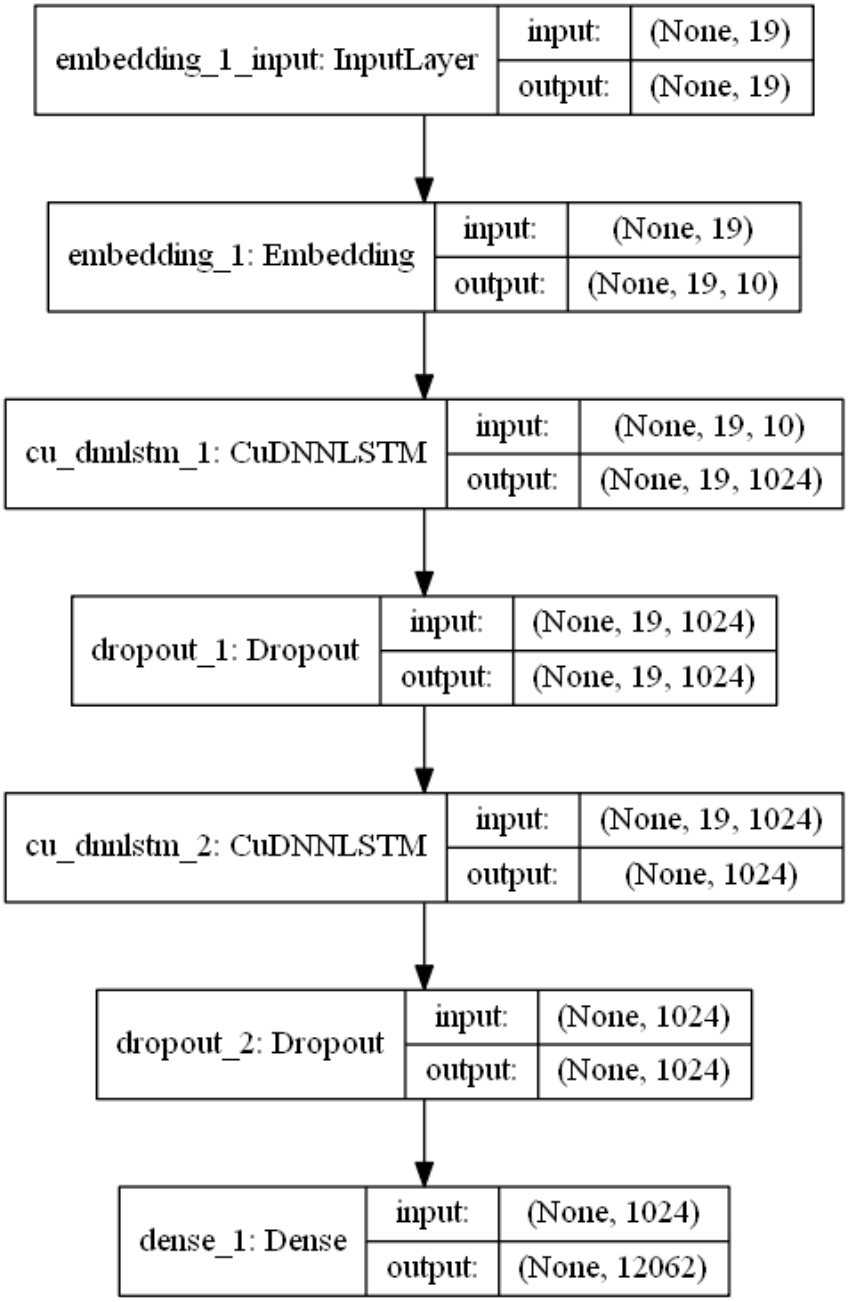


Figure 14 LSTM Sequential Model

5.4 Markov model

Εφόσον το μοντέλο θα είναι 2nd-order Markov model, η προηγούμενη κατάσταση θα είναι δύο λέξεις.

A. Εισαγωγή των απαραίτητων modules.

B. Φόρτωση του συνόλου δεδομένων 'data.txt'.

C. Tokenization του αρχείου ‘data.txt’, δηλαδή διαχώριση των προτάσεων σε λέξεις.

D. Υπολογισμός της αρχικής λέξης με την αρχική κατανομή κατάστασης.

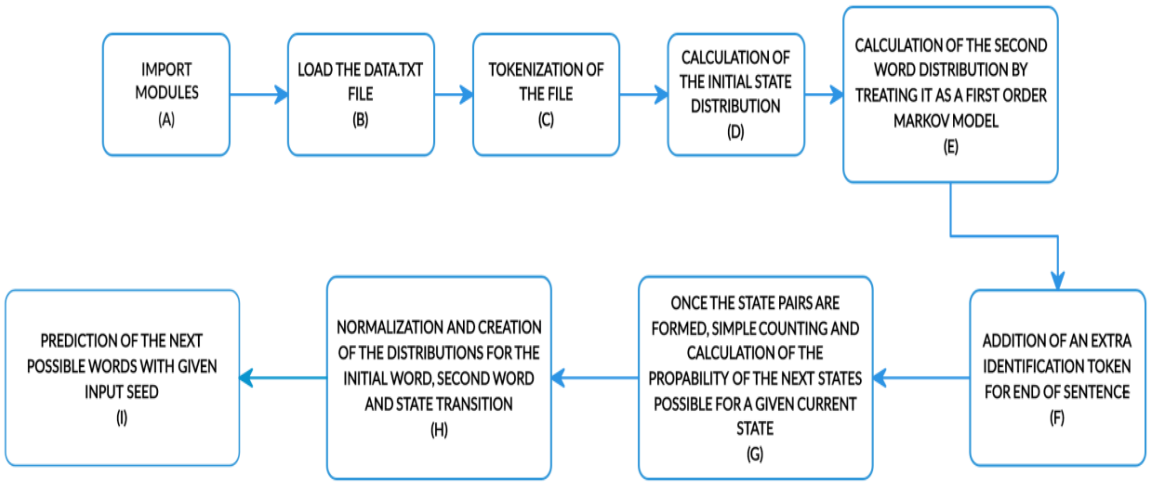
E. Υπολογισμός της δεύτερης λέξης ως 1st-order Markov model.

F. Πρόσθεση ενός επιπλέον token για την ένδειξη τέλους της γραμμής.

G. Εφόσον έχουν σχηματιστεί τα ζεύγη κατάστασης, πραγματοποιούνται προσθέσεις και γίνεται ο υπολογισμός της πιθανότητας των επόμενων πιθανών καταστάσεων για μια δεδομένη τρέχουσα κατάσταση.

H. Ομαλοποίηση και δημιουργία των κατανομών των αρχικών λέξεων, των δεύτερων λέξεων και των μεταβάσεων κατάστασης.

I. Πρόβλεψη επόμενων λέξεων με βάση ένα input seed.



6 *Αποτελέσματα*

6.1 Αποτελέσματα CNN

Στο μοντέλο που αναπτύχθηκε δοκιμάστηκαν αρκετοί παράμετροι για την καλύτερη ακρίβεια του μοντέλου και τα κυριότερα τμήματα για παραμετροποίηση ήταν ο αριθμός των επιπέδων Conv2d, του μεγέθους παρτίδας (batch size), του Dropout και του ποσοστού μάθησης (learning rate). Πιο συγκεκριμένα:

- Αριθμός των επιπέδων Conv2d : 1 , 2
- Αριθμός του μεγέθους παρτίδας (batch size) : 128, 256 , 512
- Αριθμός του Dropout : 0.2, 0.4, 0.5
- Αριθμός του ποσοστού μάθησης (learning rate) : 0.01, 0.001, 0.0005

Η λογική είναι να τρέξουν τα διάφορα μοντέλα με τις διάφορες παραμέτρους και να διαλέξουμε το μοντέλο με την μεγαλύτερη ακρίβεια χωρίς να έχουμε κάνει overfitting. Μερικά αποτελέσματα είναι οι παρακάτω εικόνες και στην περιγραφή τους οι διάφοροι παράμετροι.

	Data split 70-30							
	Layer	Batch size	Learning Rate	Dropout	Val.Acc	Val.loss	Speed/Epoch	Epochs
1st Run	1	512	0.001	0.4	0,99454	0,04387	14s	67
2nd Run	1	512	0.01	0.5	0,98171	0,06784	13s	66
3rd Run	2	512	0.001	0.4	0,99502	0,02234	13s	63
4th Run	2	256	0.001	0.4	0,99492	0,02326	16s	70
5th Run	2	128	0.001	0.4	0,99469	0,02250	22s	66
3.5th Run	2	512	0.001	0.4	0,99594	0,02164	13s	148
5.5th Run	2	128	0.001	0.4	0,99516	0,02361	22s	146
	Best Validation Accuracy				0,99594	Training	32.5 min	

Table 1 CNN Πίνακας Data split 70-30

	Data split 90-10							
	Layer	Batch size	Learning Rate	Dropout	Val.Acc	Val.loss	Speed/Epoch	Epochs
6th Run	1	512	0.001	0.4	0,99742	0,01989	14s	67
7th Run	1	256	0.001	0.5	0,99750	0,02118	13s	66
8th Run	2	512	0.001	0.4	0,99646	0,01413	13s	63
9th Run	2	256	0.001	0.4	0,99621	0,01422	16s	70
10th Run	2	128	0.001	0.4	0,99557	0,01676	22s	66
8.5th Run	2	512	0.001	0.4	0,99766	0,01222	13s	148
10.5th Run	2	128	0.0005	0.4	0,99834	0,00981	25s	290
	Best Validation Accuracy				0,99834	Training	120 min	

Table 2 CNN Πίνακας Data split 90-10

1st Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 1. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs αλλά προς τα τελευταία epochs αρχίζει να κάνει overfitting (αυξάνεται το validation loss ενώ το training loss συνεχίζει να μειώνεται) και δεν θα ήταν βέλτιστο για παραπάνω εκπαίδευση. Επιτυγχάνεται accuracy **99.454%** στο validation set και **99.849%** στο training set.

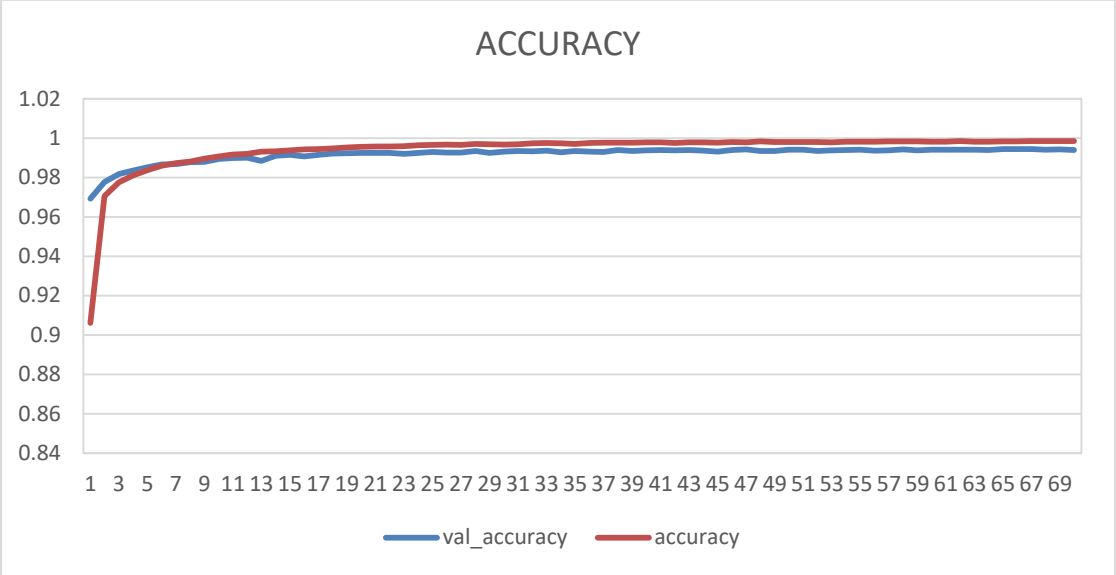


Figure 15 1st Run : 0,99454 Validation accuracy | 0,99849 Training accuracy

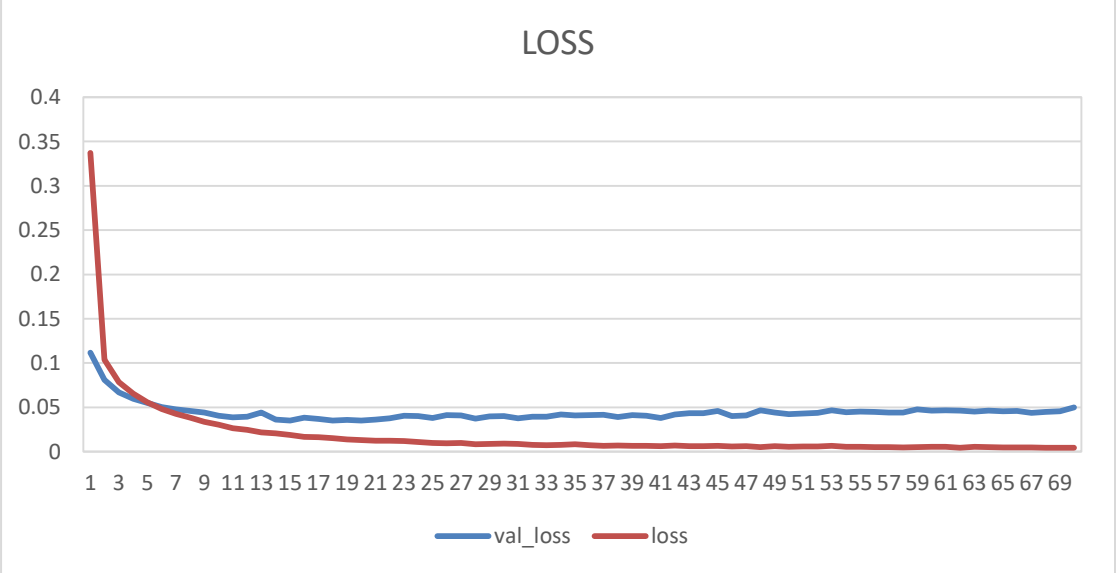


Figure 16 1st Run: 0,04387 Validation loss | 0,00472 Training loss

2nd Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 1. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs αλλά το μοντέλο φαίνεται να σταθεροποιείται στο loss και δεν φαίνεται να μειώνεται το χάσμα ανάμεσα τους. Επιτυγχάνεται accuracy **98.171%** στο validation set και **96.549%** στο training set.

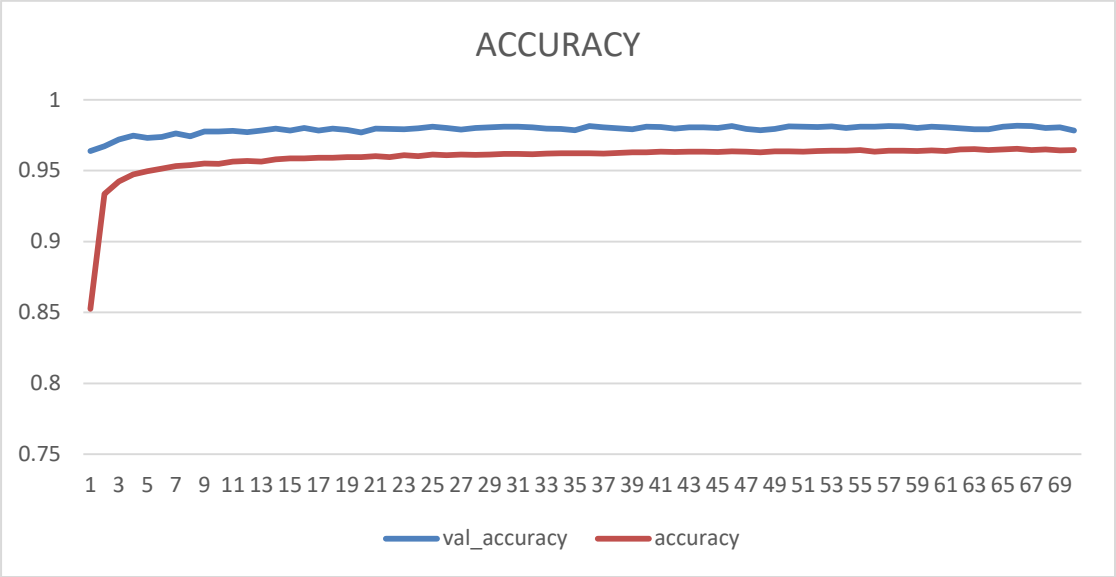


Figure 17 2nd Run : 0,98171 Validation accuracy | 0,96549 Training accuracy

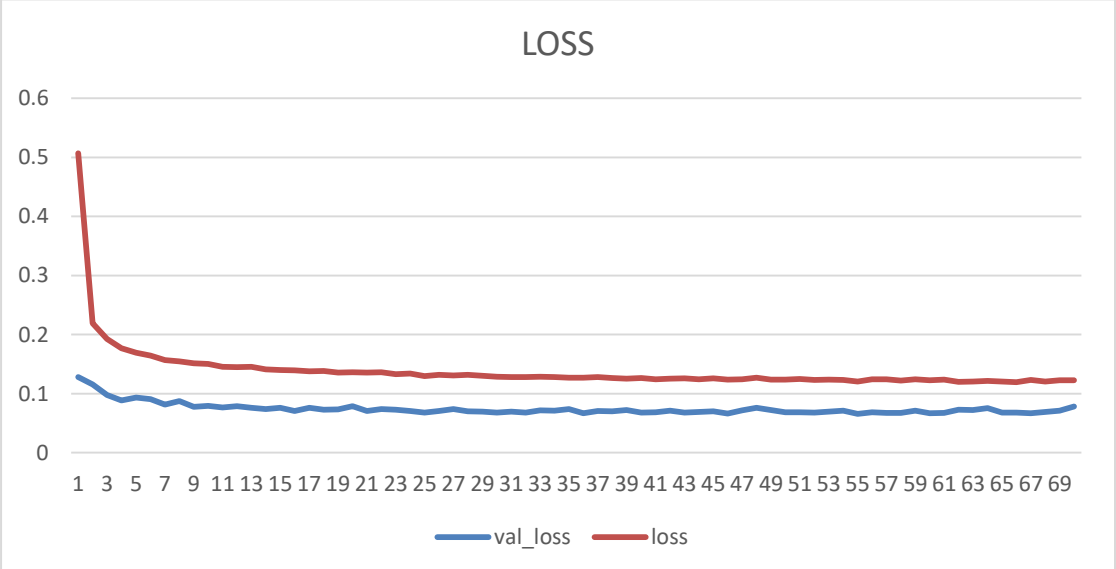


Figure 18 2nd Run : 0,06784 Validation loss | 0,11948 Training loss

3rd Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 1. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs και φαίνεται ότι θα μπορούσε να εκπαιδευτεί για παραπάνω epochs χωρίς overfitting. Επιτυγχάνεται accuracy **99.502%** στο validation set και **99.268%** στο training set.

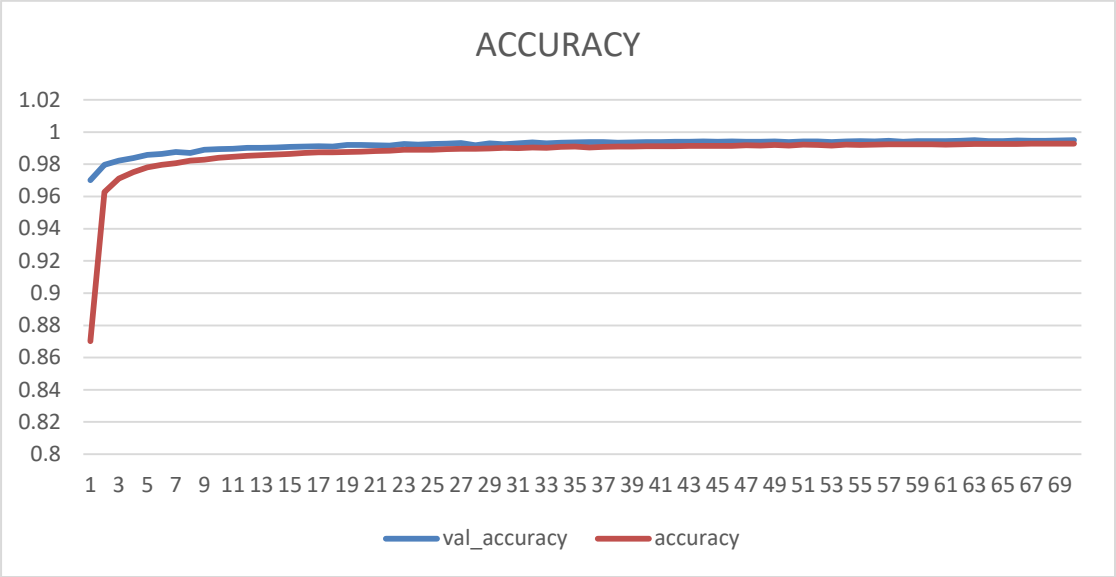


Figure 19 3rd Run : 0,99502 Validation accuracy | 0,99268 Training accuracy

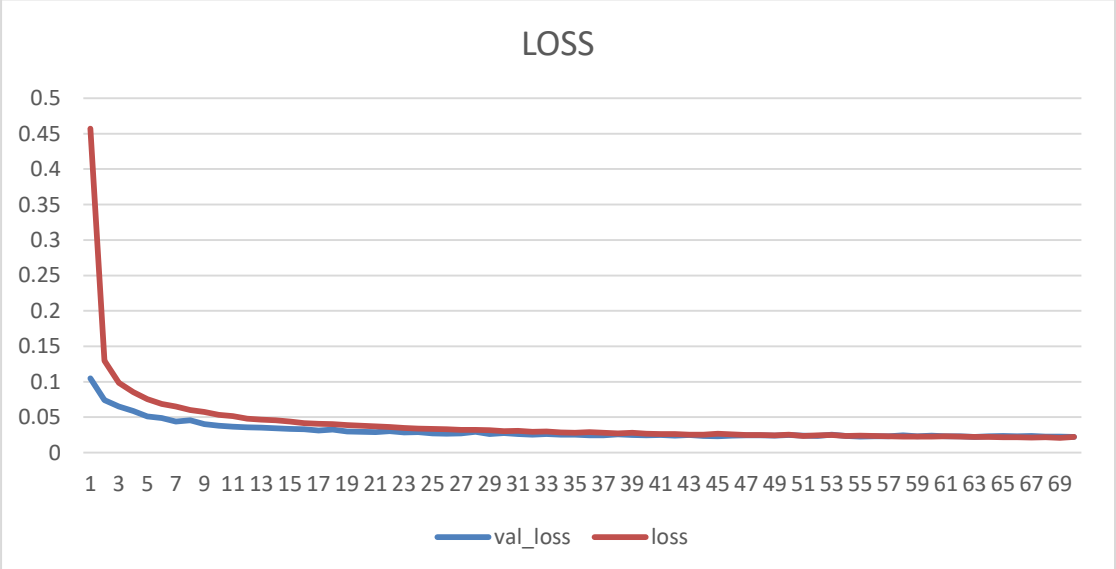


Figure 20 3rd Run : 0,02234 Validation loss | 0,02230 Training loss

3.5th Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 1. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs χωρίς overfitting. Το μοντέλο αυτό είναι ίδιο με το 3rd Run αλλά εκπαιδεύτηκε για παραπάνω epochs. Επιτυγχάνεται accuracy **99.594%** στο validation set και **99.408%** στο training set.

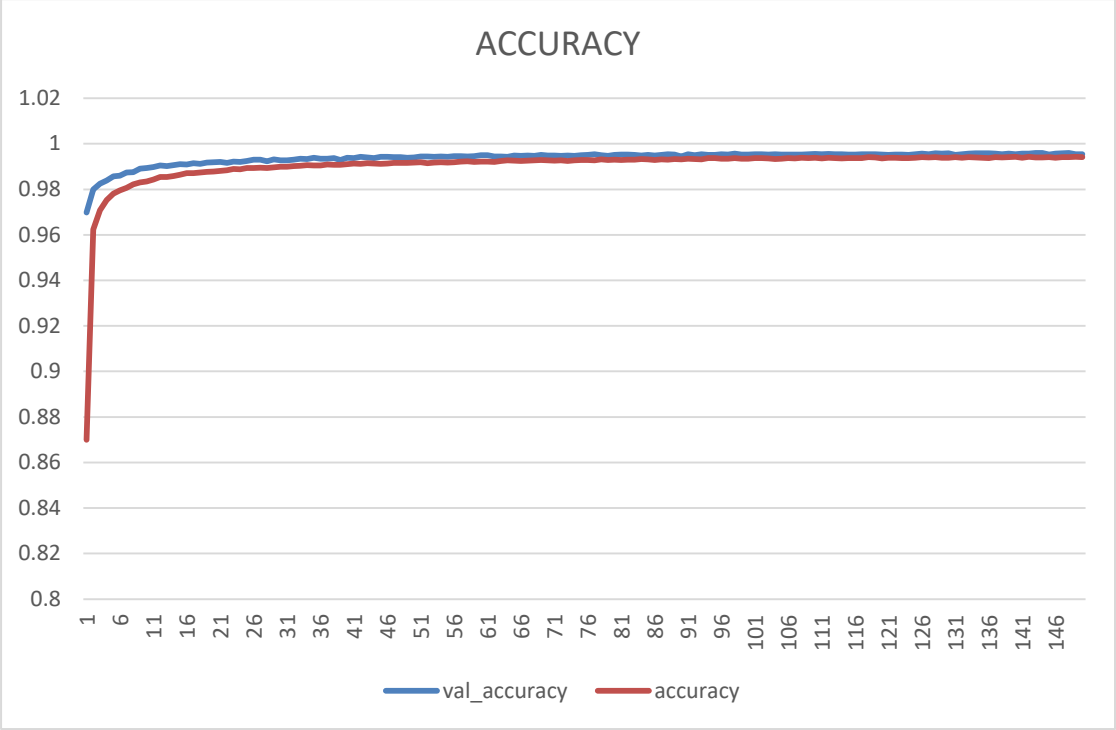


Figure 21 3.5th Run : 0,99594 Validation accuracy | 0,99408 Training accuracy

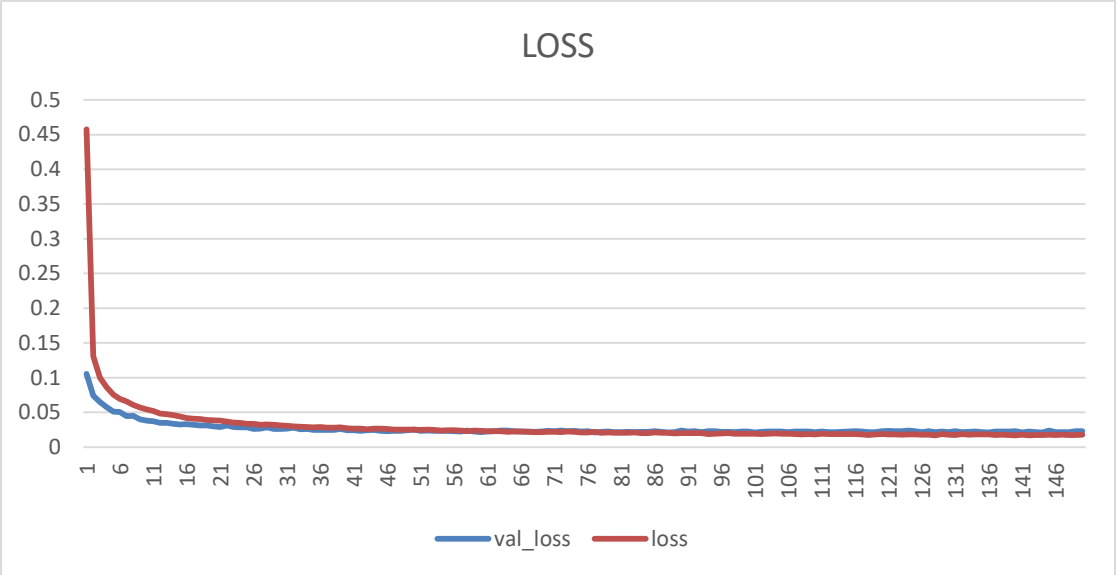


Figure 22 3.5th Run : 0,02164 Validation loss | 0,01733 Training loss

4th Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 1. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs και φαίνεται ότι θα μπορούσε να εκπαιδευτεί για παραπάνω epochs χωρίς overfitting. Επιτυγχάνεται accuracy **99.492%** στο validation set και **99.219%** στο training set.

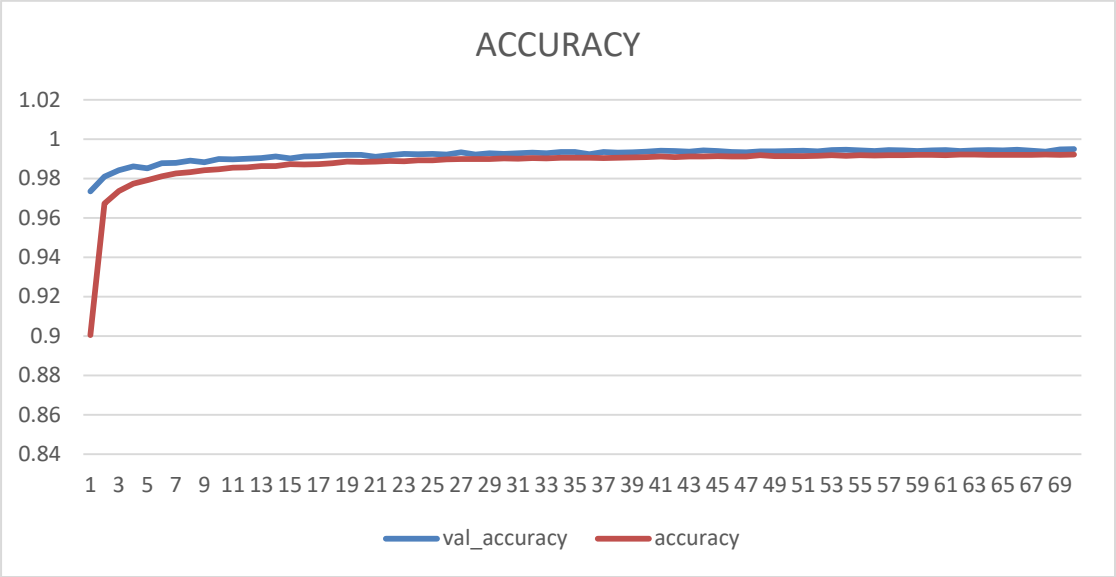


Figure 23 4th Run : 0,99492 Validation accuracy | 0,99219 Training accuracy

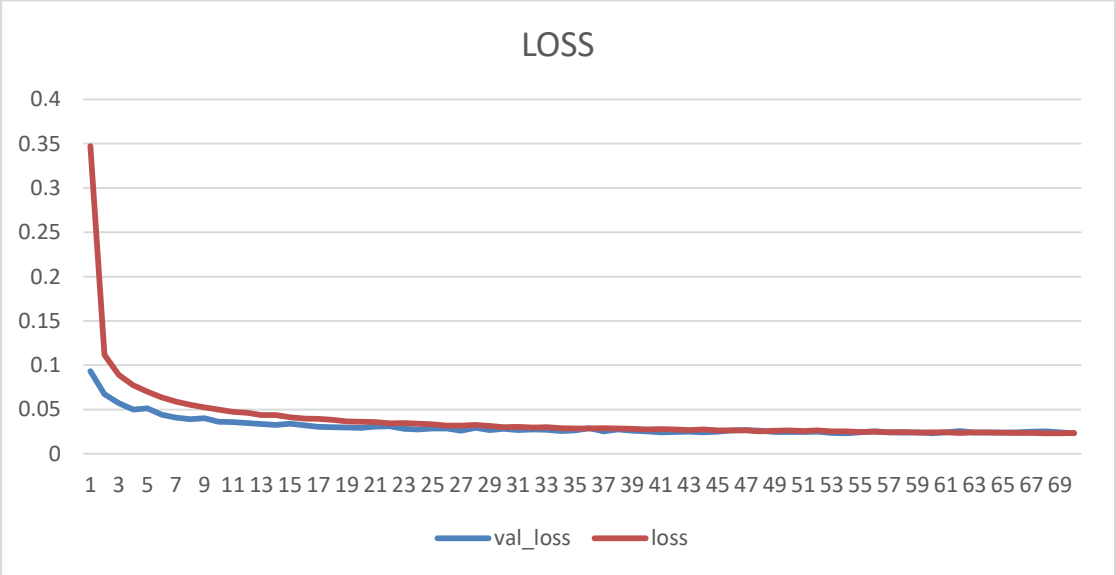


Figure 24 4th Run : 0,02326 Validation loss | 0,02346 Training loss

5th Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 1. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs και φαίνεται ότι θα μπορούσε να εκπαιδευτεί για παραπάνω epochs χωρίς overfitting. Επιτυγχάνεται accuracy **99.469%** στο validation set και **99.119%** στο training set.

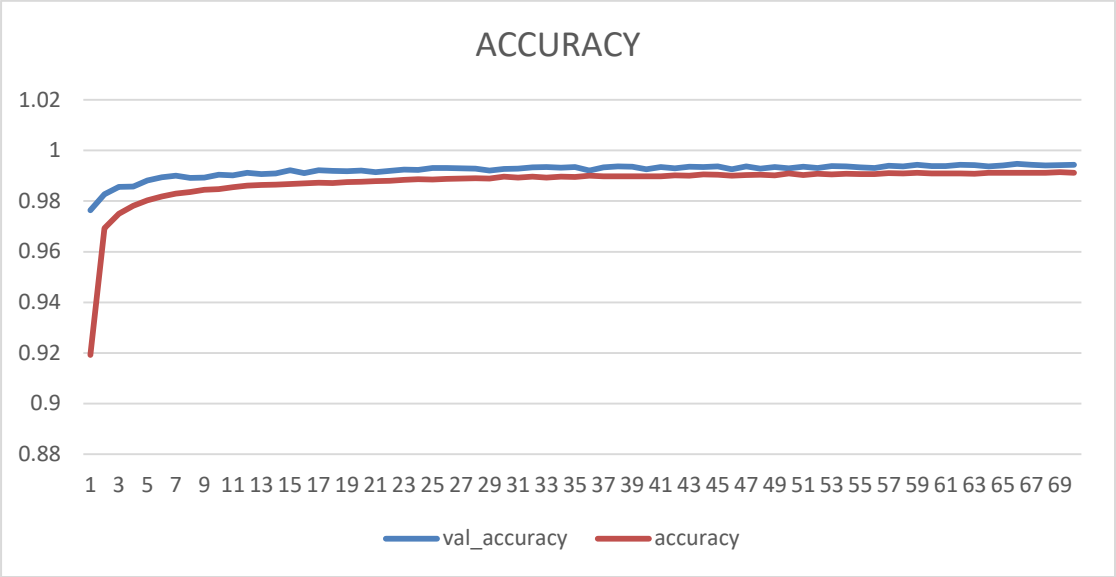


Figure 25 5th Run : 0,99469 Validation accuracy | 0,99119 Training accuracy

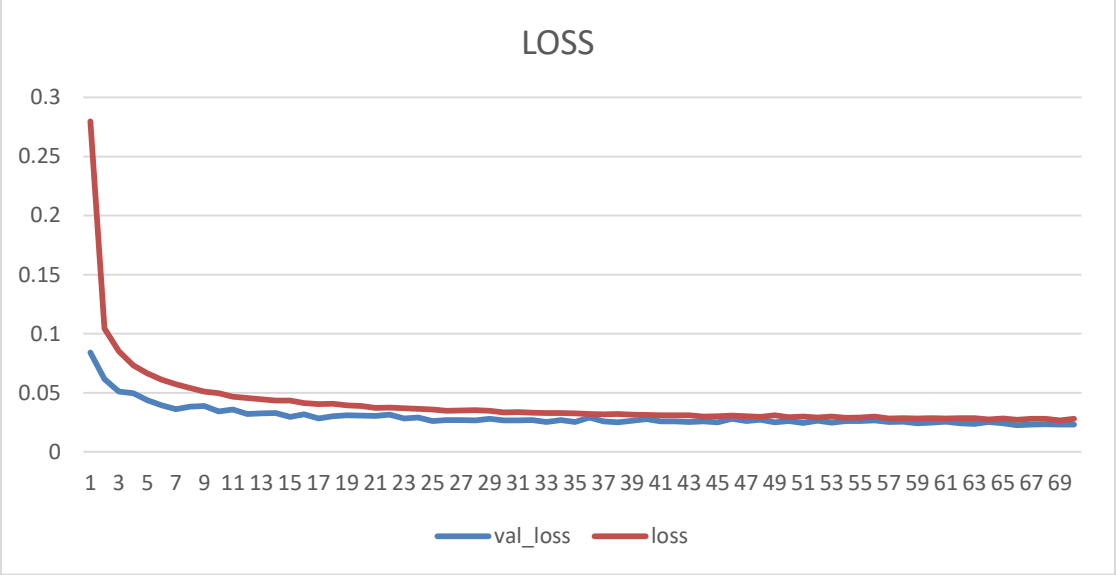


Figure 26 5th Run : 0,02250 Validation loss | 0,02713 Training loss

5.5th Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 1. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs χωρίς overfitting. Το μοντέλο αυτό είναι ίδιο με το 5th Run αλλά εκπαιδεύτηκε για παραπάνω epochs. Επιτυγχάνεται accuracy **99.516%** στο validation set και **99.221%** στο training set.

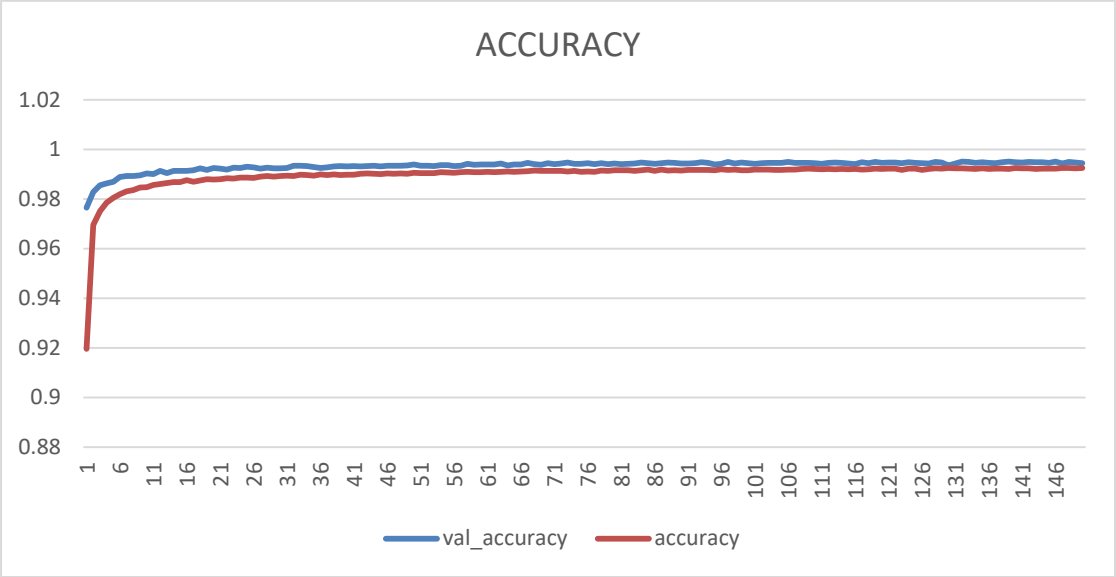


Figure 27 5.5th Run : 0,99516 Validation accuracy | 0,99221 Training accuracy

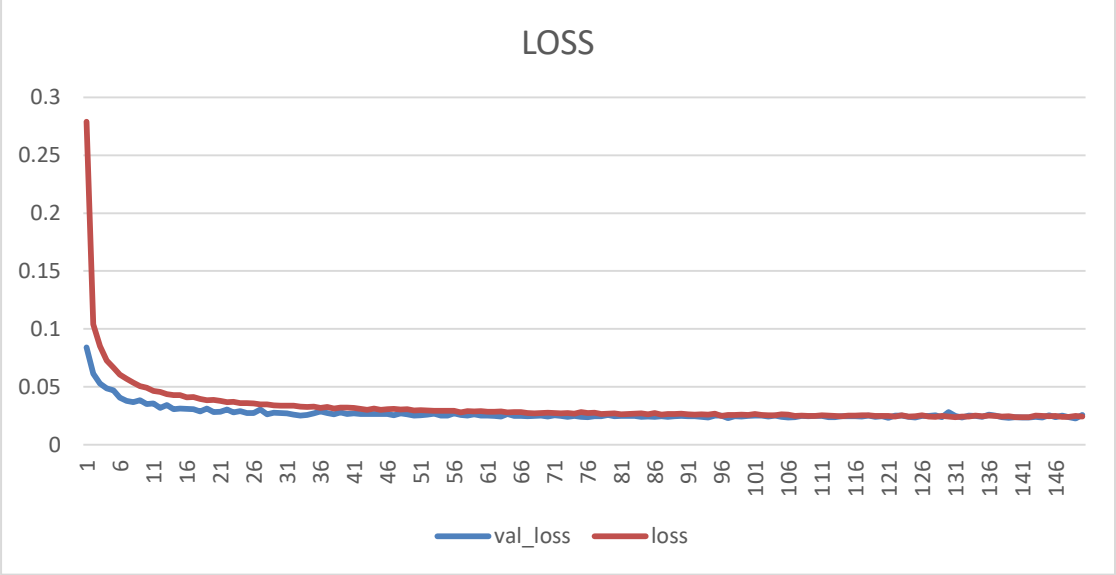


Figure 28 5.5th Run : 0, 02361 Validation loss | 0,02479 Training loss

6th Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 2. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs αλλά προς τα τελευταία epochs αρχίζει να κάνει overfitting (αυξάνεται το validation loss ενώ το training loss συνεχίζει να μειώνεται) και δεν θα ήταν βέλτιστο για παραπάνω εκπαίδευση. Επιτυγχάνεται accuracy **99.742%** στο validation set και **99.858%** στο training set.

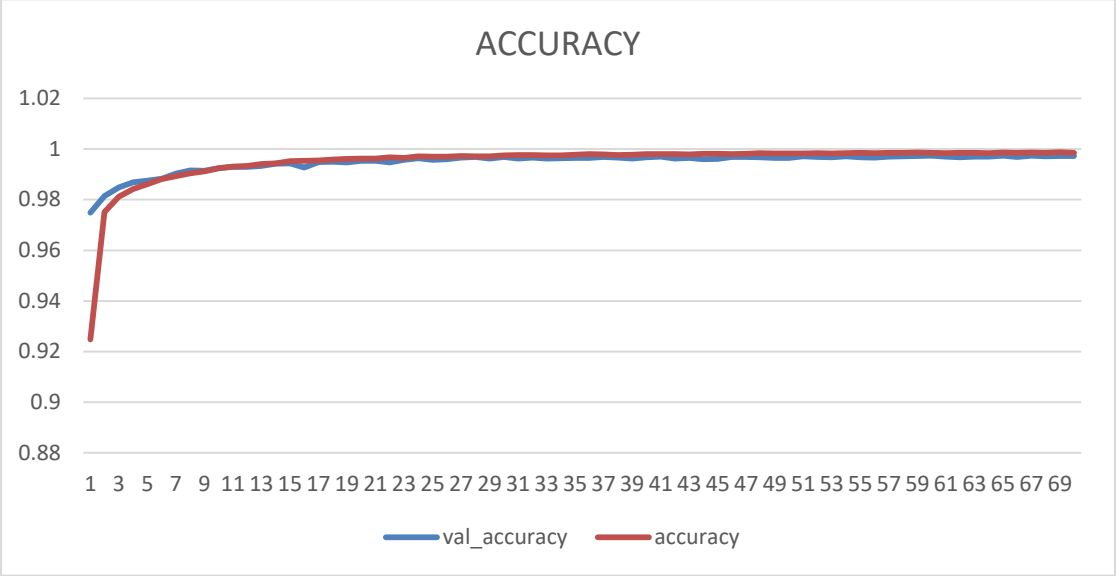


Figure 29 6th Run : 0,99742 Validation accuracy | 0,99858 Training accuracy

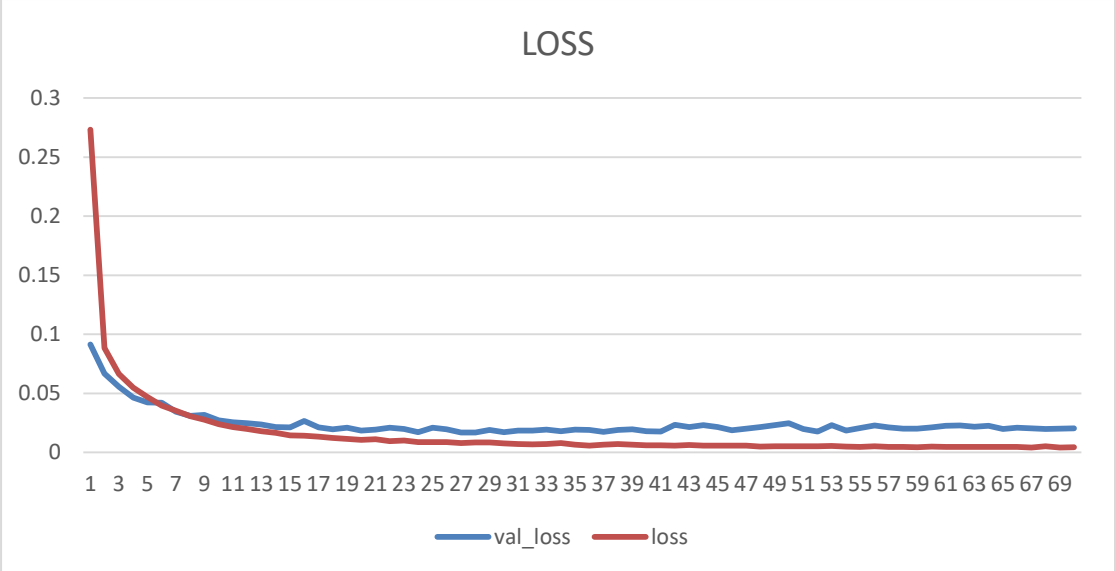


Figure 30 6th Run : 0,01989 Validation loss | 0,00455 Training loss

7^{th} Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 2. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs αλλά προς τα τελευταία epochs αρχίζει να κάνει overfitting (αυξάνεται το validation loss ενώ το training loss συνεχίζει να μειώνεται) και δεν θα ήταν βέλτιστο για παραπάνω εκπαίδευση. Επιτυγχάνεται accuracy **99.750%** στο validation set και **99.783%** στο training set.

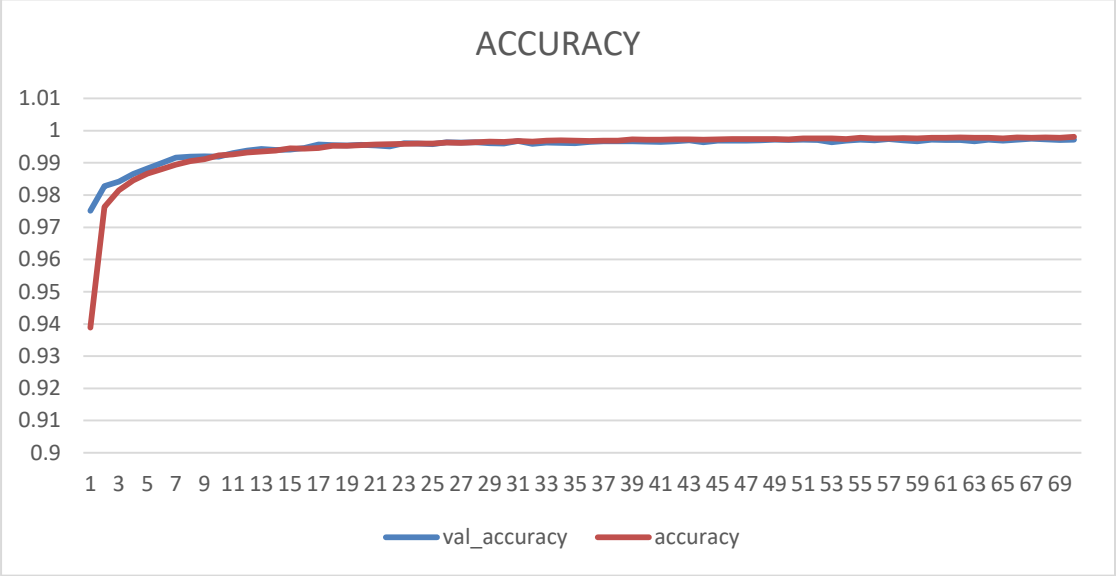


Figure 31 7^{th} Run : 0,99750 Validation accuracy | 0,99783 Training accuracy

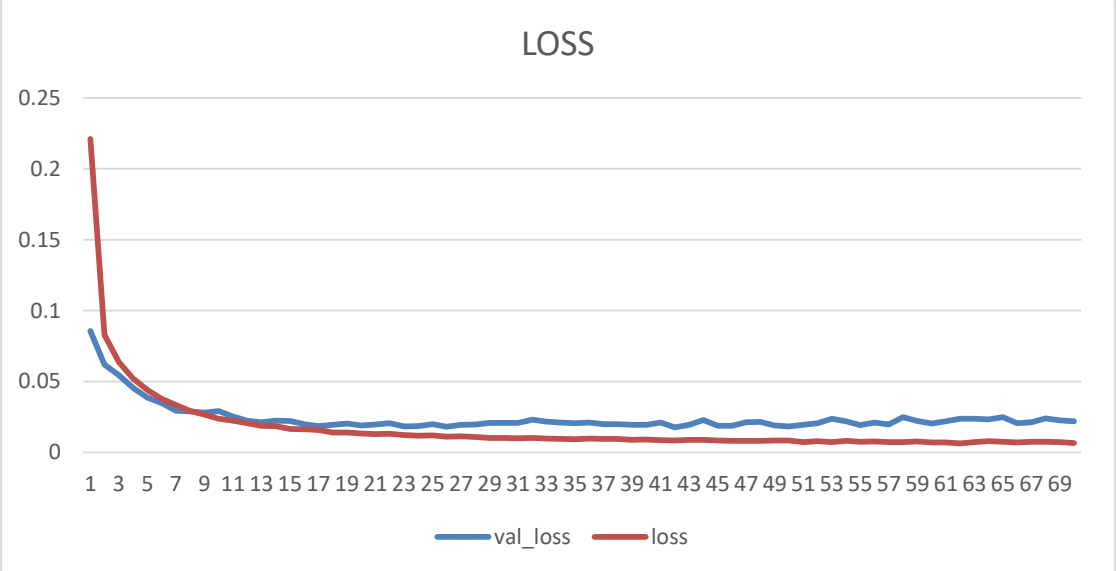


Figure 32 7^{th} Run : 0,02118 Validation loss | 0,00753 Training loss

8th Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 2. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs και φαίνεται ότι θα μπορούσε να εκπαιδευτεί για παραπάνω epochs χωρίς overfitting. Επιτυγχάνεται accuracy **99.646%** στο validation set και **99.261%** στο training set.

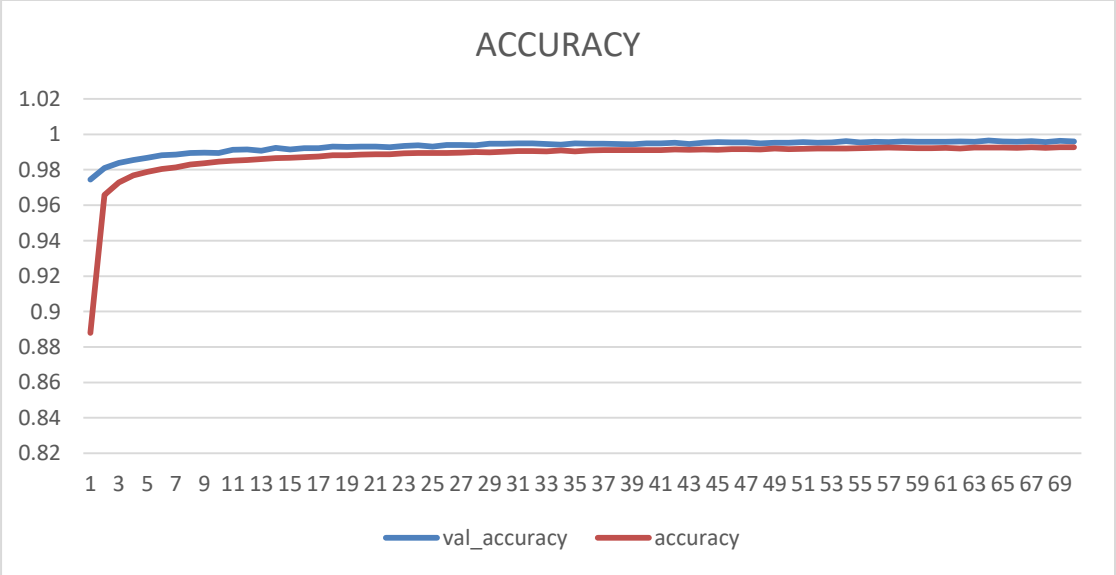


Figure 33 8th Run : 0,99646 Validation accuracy | 0,99261 Training accuracy

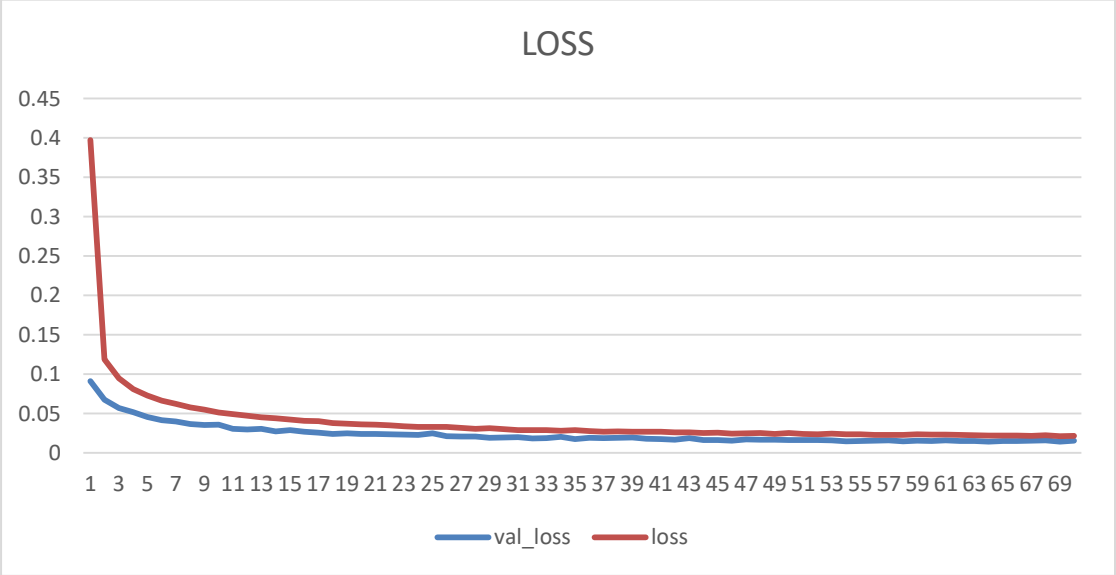


Figure 34 8th Run : 0,01413 Validation loss | 0,02189 Training loss

8.5th Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 2. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs χωρίς overfitting. Το μοντέλο αυτό είναι ίδιο με το 8th Run αλλά εκπαιδεύτηκε για παραπάνω epochs. Επιτυγχάνεται accuracy **99.766%** στο validation set και **99.441%** στο training set.

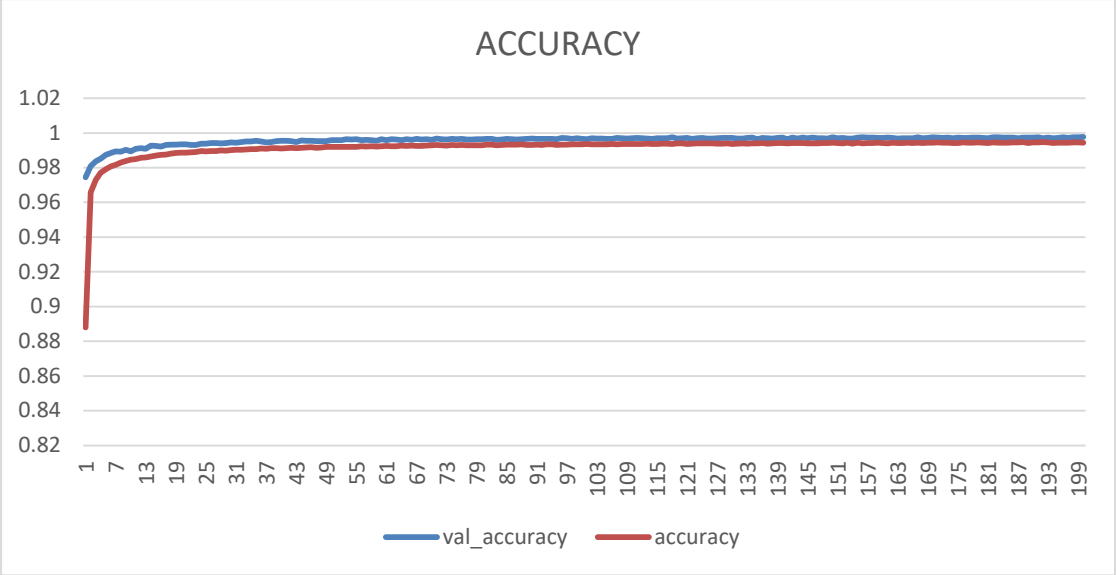


Figure 35 8.5th Run : 0,99766 Validation accuracy | 0,99441 Training accuracy

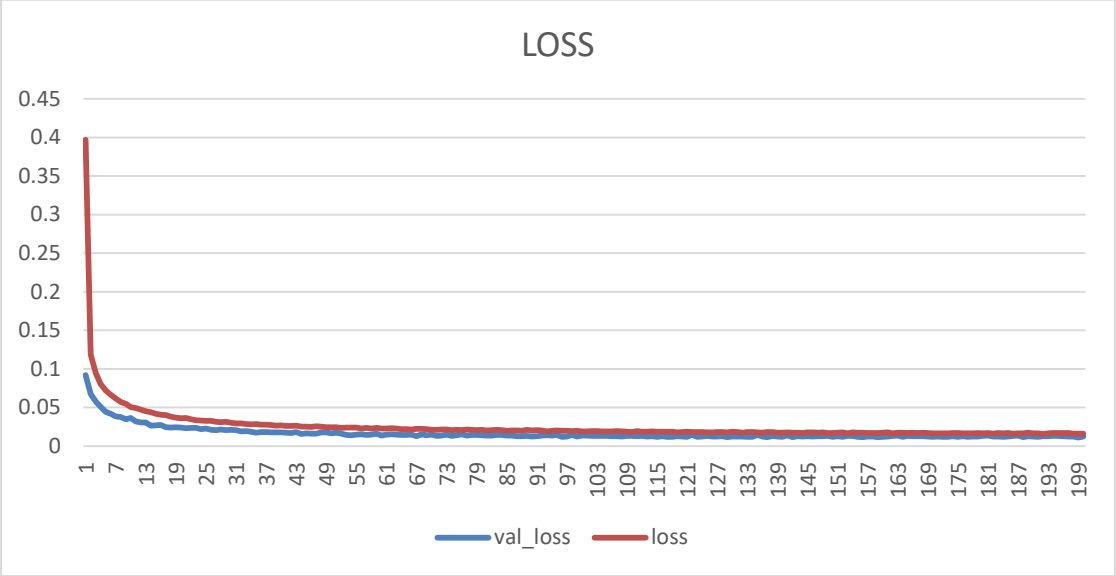


Figure 36 8.5th Run : 0,01222 Validation loss | 0,01591 Training loss

9th Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 2. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs και φαίνεται ότι θα μπορούσε να εκπαιδευτεί για παραπάνω epochs χωρίς overfitting. Επιτυγχάνεται accuracy **99.621%** στο validation set και **99.192%** στο training set.

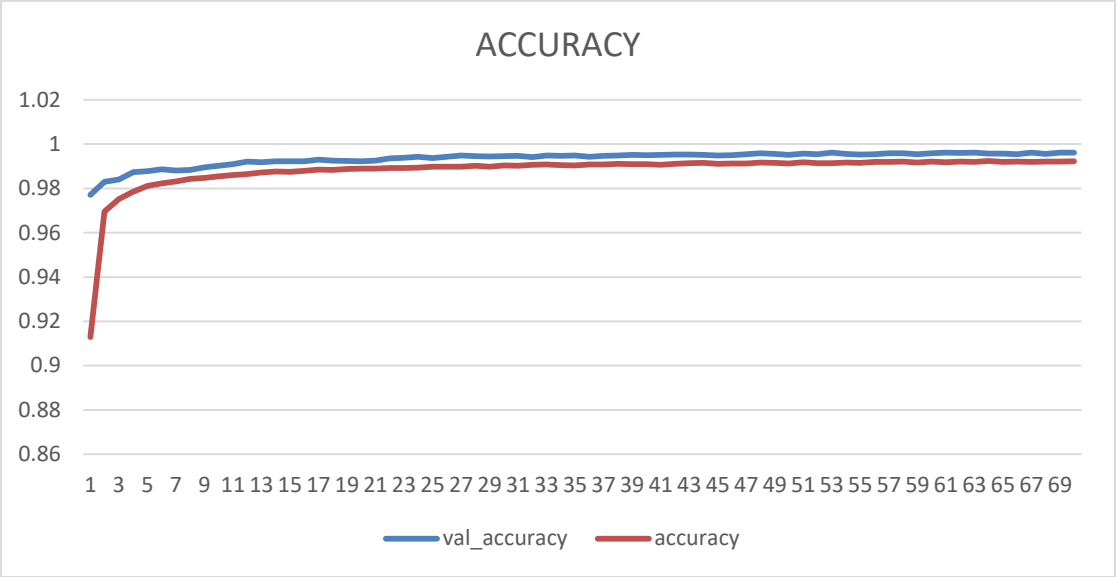


Figure 37 9th Run : 0,99621 Validation accuracy | 0,99192 Training accuracy

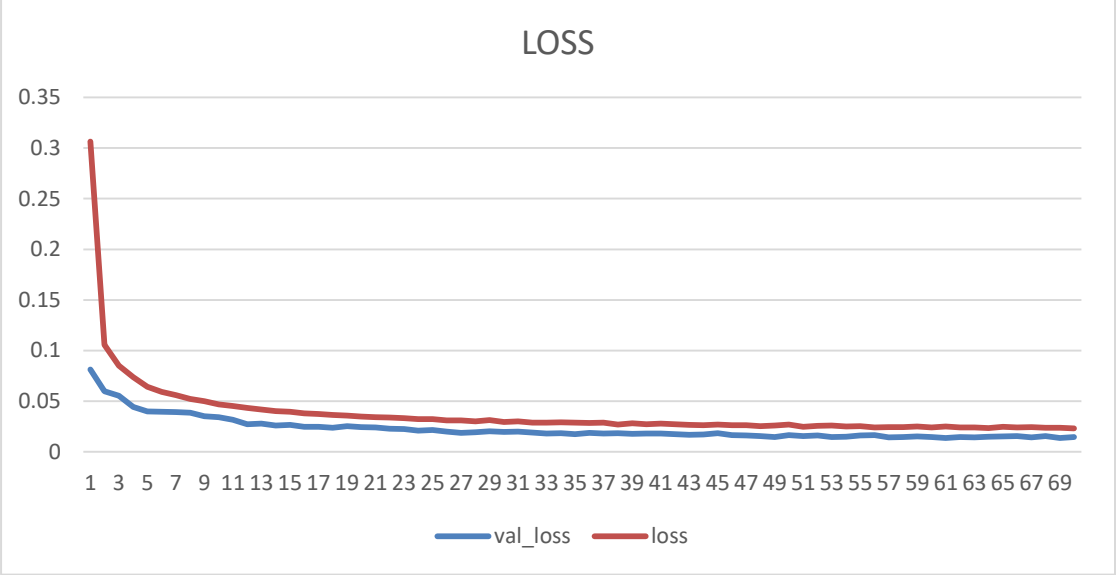


Figure 38 9th Run : 0,01422 Validation loss | 0,02420 Training loss

10th Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 2. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs και φαίνεται ότι θα μπορούσε να εκπαιδευτεί για παραπάνω epochs χωρίς overfitting. Επιτυγχάνεται accuracy **99.557%** στο validation set και **99.124%** στο training set.

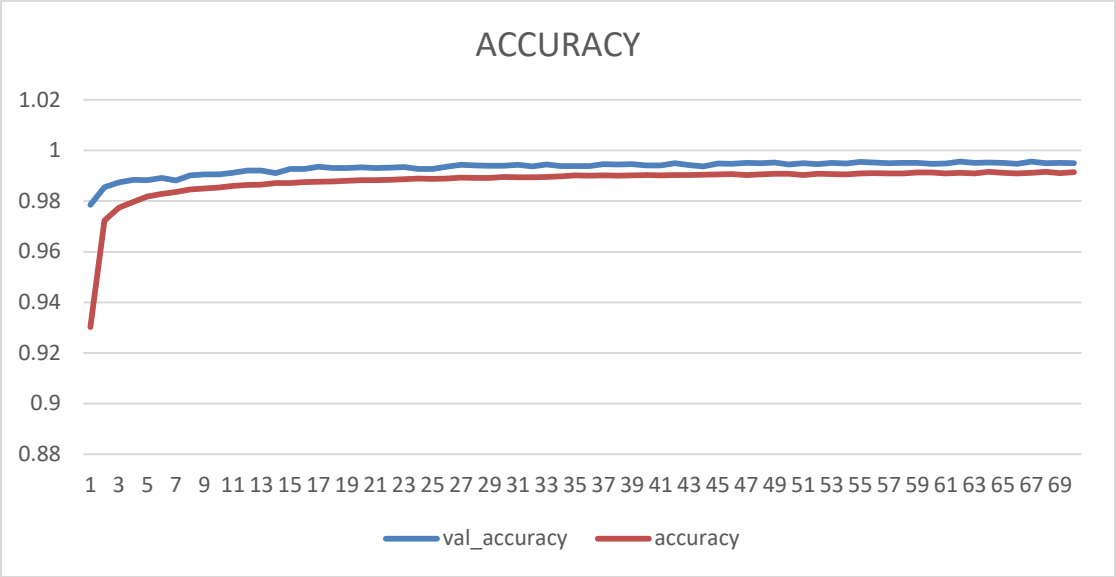


Figure 39 10th Run : 0,99557 Validation accuracy | 0,99124 Training accuracy

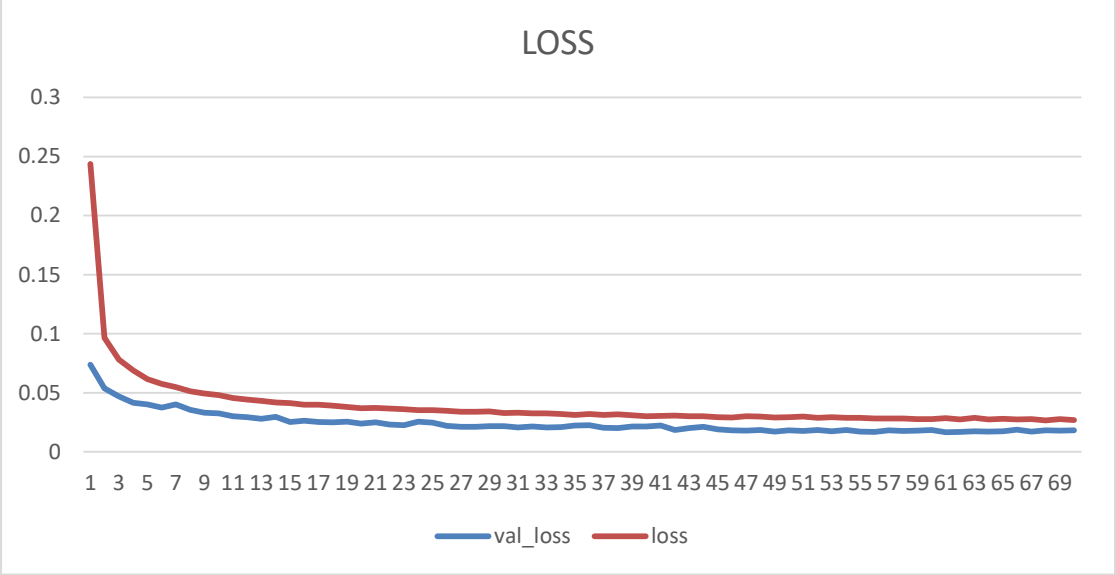


Figure 40 10th Run : 0,01676 Validation loss | 0,02744 Training loss

10.5th Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 2. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs χωρίς overfitting. Το μοντέλο αυτό είναι ίδιο με το 10th Run αλλά εκπαιδεύτηκε για παραπάνω epochs. Επιτυγχάνεται accuracy **99.834%** στο validation set και **99.504%** στο training set. Το μοντέλο αυτό είχε την μεγαλύτερη ακρίβεια χωρίς overfitting.

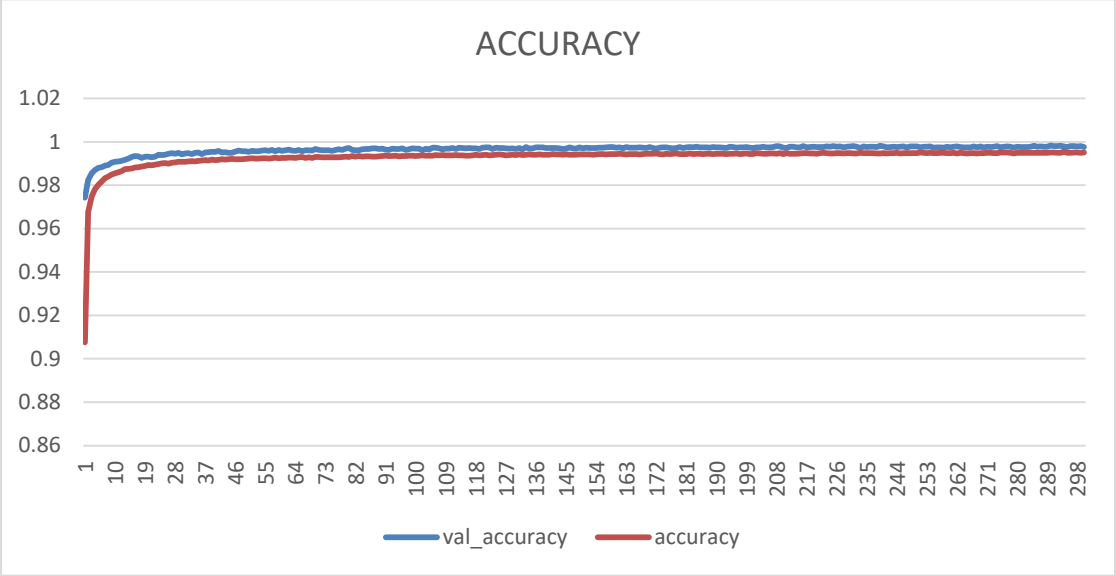


Figure 41 10.5th Run : 0,99834 Validation accuracy | 0,99504 Training accuracy

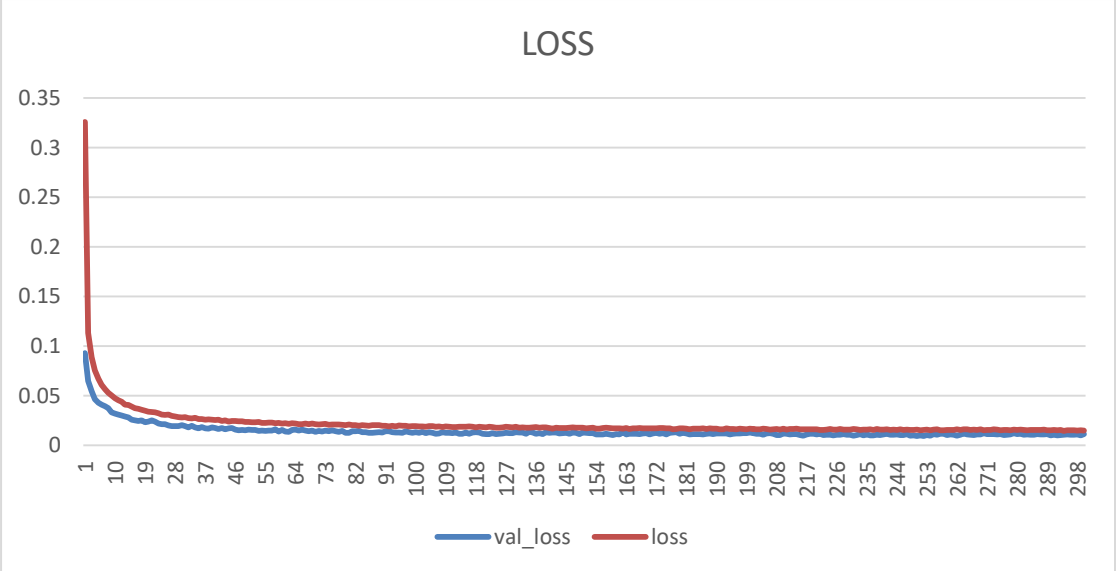


Figure 42 10.5th Run : 0,00981 Validation loss | 0,01498 Training loss

6.2 Αποτελέσματα προ-επεξεργασίας και αναγνώρισης κειμένου

Στο στάδιο αυτό, εφόσον έχει καθοριστεί το εκπαιδευμένο μοντέλο, πρέπει να γίνει κάποια προ-επεξεργασία της εικόνας εισόδου ώστε να τροφοδοτηθεί κατάλληλα το μοντέλο και να προβλέψει επιτυχώς τα γράμματα που αναπαριστώνται στην εικόνα.

Οι κυριότερες παραμετροποιήσεις αφορούν την εντόπιση του κειμένου, όπου όμως είναι σε σύγκριση με το σύνολο δεδομένων που είναι διαθέσιμο, και την μορφή λήψης της εικόνας από τον χρήστη.

Ιδανικό θα ήταν να μην γίνεται η λήψη από πολύ μακριά και η φωτεινότητα να είναι ικανοποιητική. Επίσης, λόγω του ότι το σύνολο δεδομένων που χρησιμοποιήθηκε για την εκπαίδευση του μοντέλου CNN ήταν περιορισμένο, κάποια γράμματα όπως το I, πρέπει να γραφτούν στην μορφή που είναι και στο σύνολο δεδομένων όπως παρακάτω:

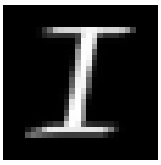


Figure 43

Επίσης το σύνολο δεδομένων είναι τα **ΚΕΦΑΛΑΊΑ** γράμματα της **αγγλικής** αλφαβήτου, χωρίς σημεία στίξης. Παρακάτω, θα παρουσιαστεί ένα παράδειγμα αποτελεσμάτων.

Παράδειγμα:

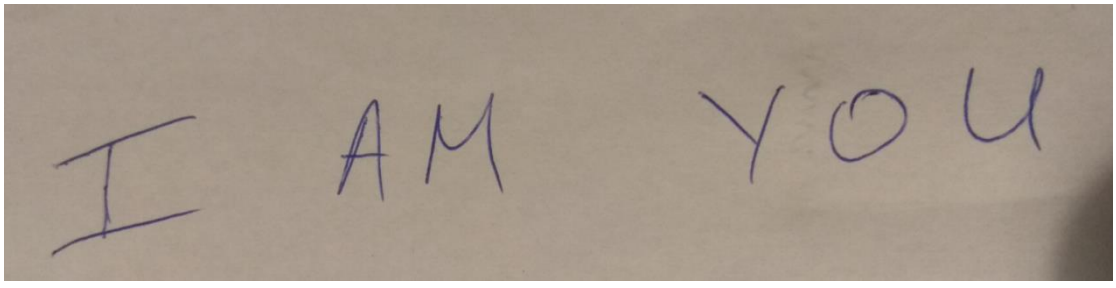


Figure 44 Input picture

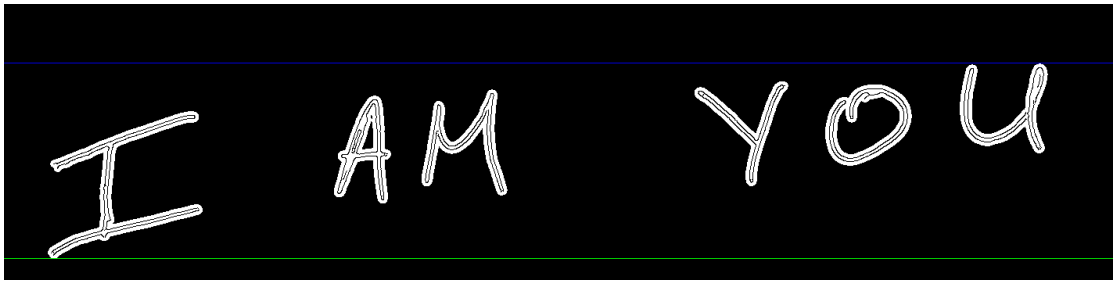


Figure 45 Line segmentation by blue and green boundaries

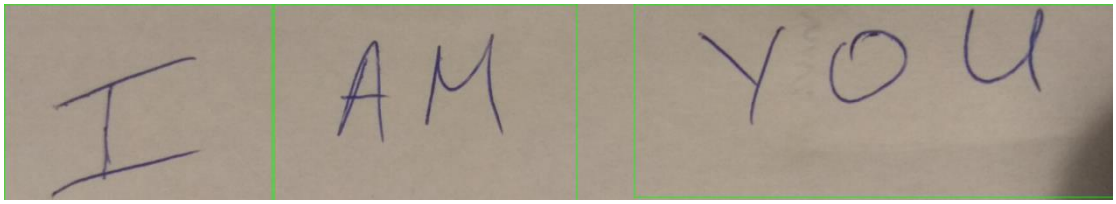
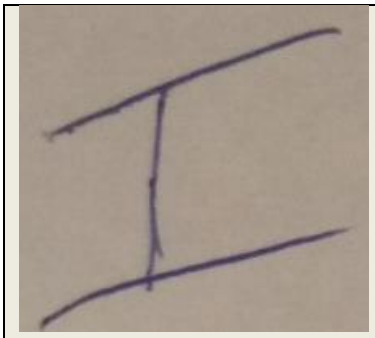
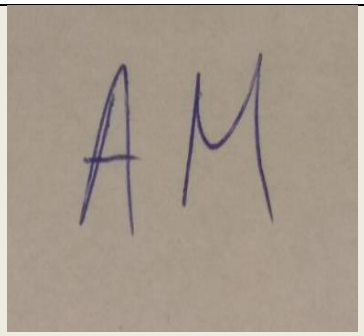


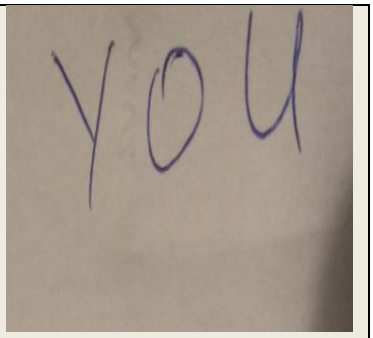
Figure 46 Word segmentation



Word “I”



Word “AM”



Word “YOU”

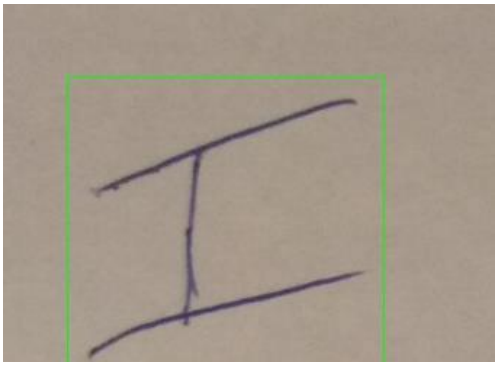


Figure 47 Character segmentation

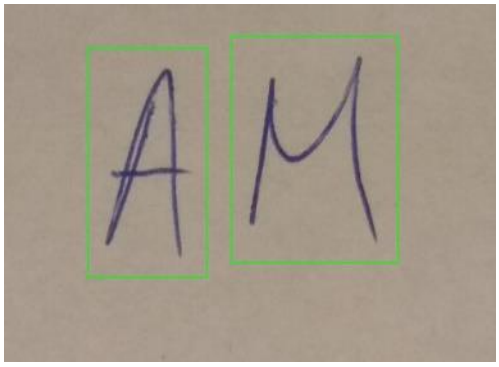


Figure 48 Character segmentation

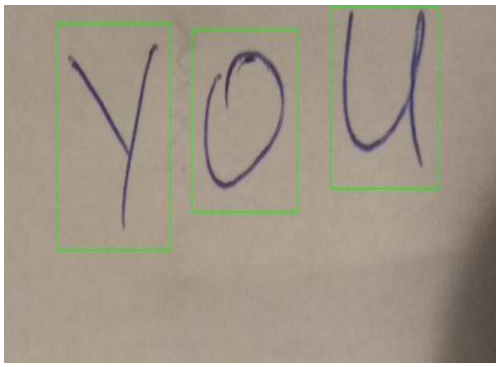


Figure 49 Character segmentation

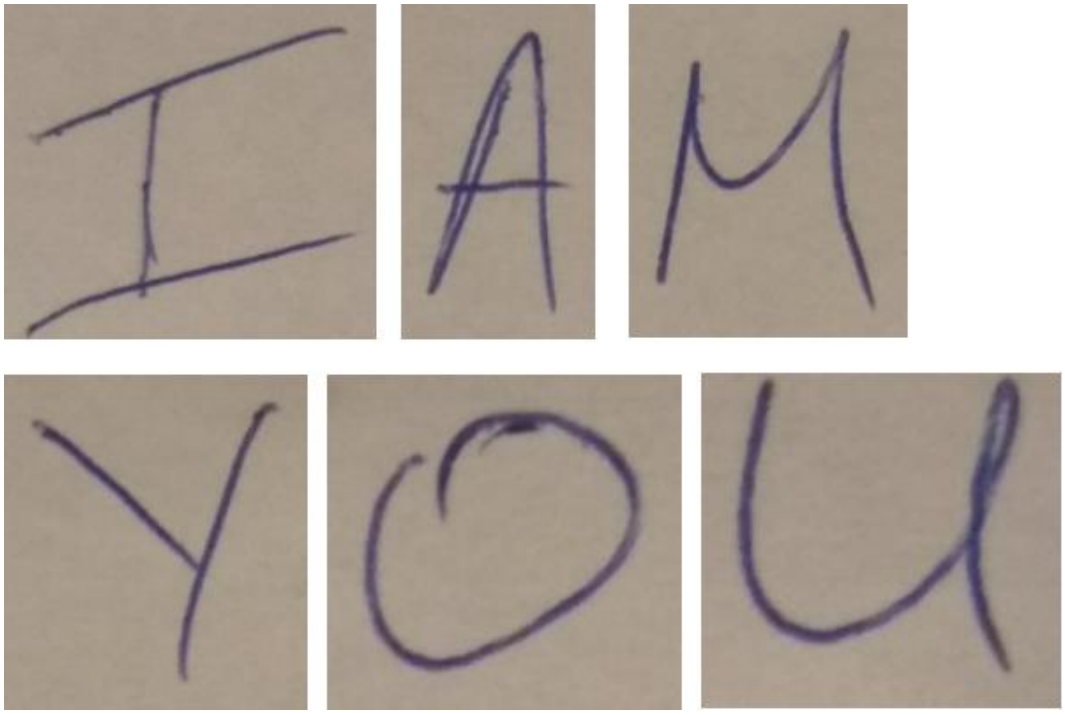


Figure 50 Cropped characters

Figure 51 Characters regulated to MNIST dataset and ready to feed the CNN model

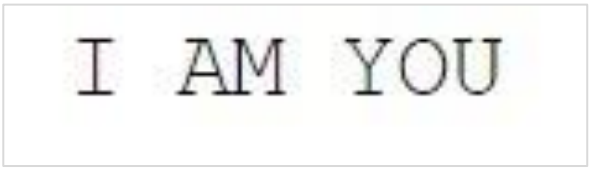


Figure 52 Python result after running

6.3 Αποτελέσματα LSTM

Στο μοντέλο που αναπτύχθηκε δοκιμάστηκαν κάποιοι παράμετροι για την καλύτερη ακρίβεια του μοντέλου. Η λογική είναι να τρέξουν τα διάφορα μοντέλα με τις διάφορες παραμέτρους και να διαλέξουμε το μοντέλο με την μεγαλύτερη ακρίβεια χωρίς να έχουμε κάνει overfitting. Μερικά αποτελέσματα είναι οι παρακάτω εικόνες και στην περιγραφή τους οι διάφοροι παράμετροι.

	Data split 90-10							
	Layer	Batch size	Learning Rate	Dropout	Val.Acc	Val.loss	Speed/Epoch	Epochs
1st Run	2	32	0.0005	0.5	0,14751	5,99743	450s	20
2nd Run	2	32	0.0005	0.2	0,13717	6,05898	450s	11
		Best Validation Accuracy			0,14751	Training	150 min	

Table 3 LSTM Πίνακας Data split 90-10

1st Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 3. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs αλλά προς τα τελευταία epochs αρχίζει να κάνει overfitting (αυξάνεται το validation loss ενώ το training loss συνεχίζει να μειώνεται) και δεν θα ήταν βέλτιστο για παραπάνω εκπαίδευση. Επιτυγχάνεται accuracy **14.751%** στο validation set και **17.003%** στο training set.

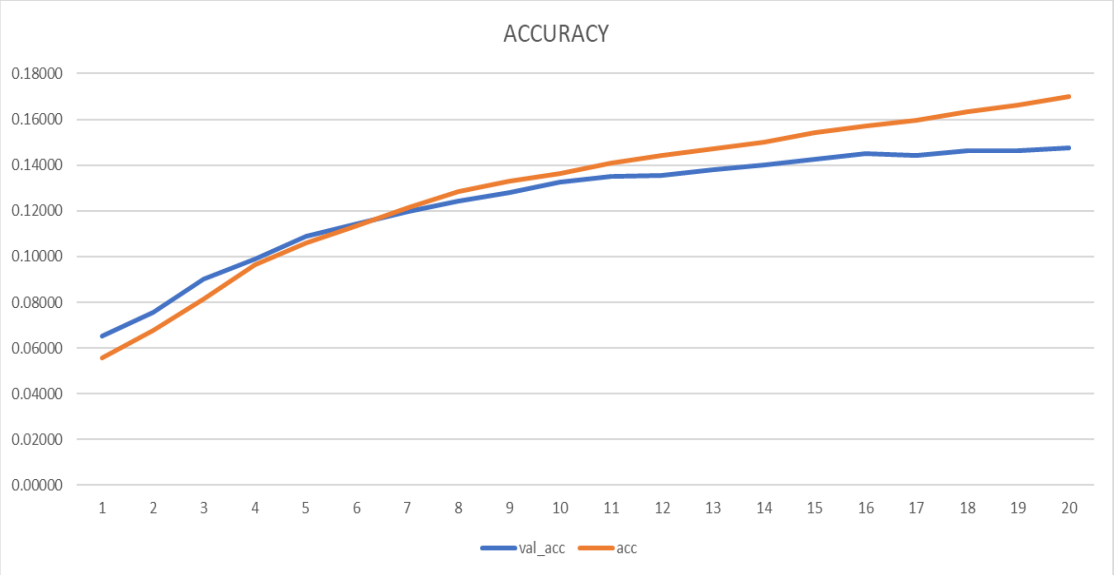


Figure 53 1st Run : 0,14751 Validation accuracy | 0,17003 Training accuracy

Figure 54 1st Run: 5,99743 Validation loss | 4,80891 Training loss

2nd Run πείραμα με τις παραμέτρους που αναδεικνύονται στο Table 3. Παρατηρούμε ότι τα validation και training losses είναι αρκετά κοντά όσο αυξάνονται τα epochs αλλά προς τα τελευταία epochs αρχίζει να κάνει overfitting (αυξάνεται το validation loss ενώ το training loss συνεχίζει να μειώνεται) και δεν θα ήταν βέλτιστο για παραπάνω εκπαίδευση. Επιτυγχάνεται accuracy **13.717%** στο validation set και **15.622%** στο training set.

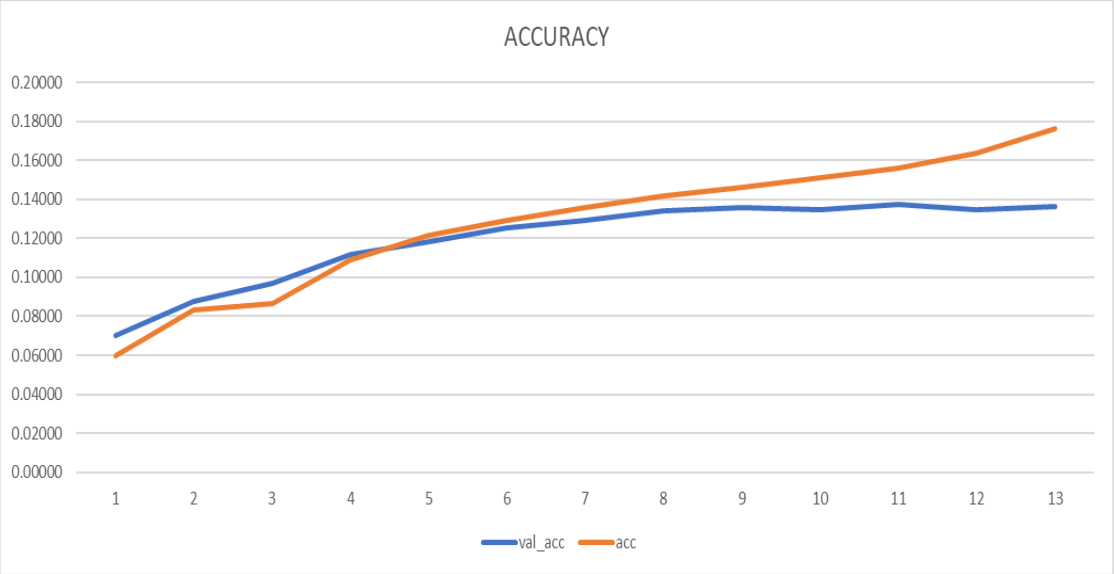


Figure 55 2nd Run : 0,13717 Validation accuracy | 0,15622 Training accuracy

Figure 56 2nd Run: 6,05898 Validation loss | 4,69262 Training loss

Εάν θεωρηθεί σαν είσοδος seed το αποτέλεσμα του 6.2, δηλαδή το ‘I am you’, και δηλωθεί σαν παράμετρος ο αριθμός επόμενων λέξεων σε 1, τότε κάποια από τα αποτελέσματα του μοντέλου φαίνονται παρακάτω:

```
Model prediction:
i am you have
```

Εάν για παράδειγμα δηλωθεί σαν παράμετρος ο αριθμός επόμενων λέξεων σε 15 τότε κάποιο αποτελέσματα είναι:

```
Model prediction:
i am you have said a little man and i know that i cant be in love with
```

6.4 Αποτελέσματα Markov model

Εάν θεωρηθεί σαν είσοδος seed το αποτέλεσμα του 6.2, δηλαδή το ‘I am you’, και δηλωθεί σαν παράμετρος ο αριθμός επόμενων λέξεων σε 1, τότε κάποια από τα αποτελέσματα του μοντέλου φαίνονται παρακάτω:

```
Model prediction:
i am you are

Model prediction:
i am you dont
```

Ο λόγος για τα δύο διαφορετικά αποτελέσματα σε δύο διαφορετικές εκτελέσεις του script, είναι διότι έχει προστεθεί η λειτουργία ώστε όταν κάνει την πρόβλεψη της επόμενης λέξης να διαλέγει στην τύχη ανάμεσα στις τρεις καλύτερες υποψήφιες λέξεις ώστε να υπάρχει ευελιξία.

Εάν για παράδειγμα δηλωθεί σαν παράμετρος ο αριθμός επόμενων λέξεων σε 15 τότε κάποια αποτελέσματα είναι:

```
Model prediction:
i am you are the old men who had been in the princess was the same thing to

Model prediction:
i am you dont want to be a smile of his face in the old men not go
```

7 *Συμπεράσματα*

Σε αυτήν την εργασία, περιγράφηκε ο τρόπος με τον οποίο μπορεί κάποιος να αναγνωρίσει το χειρόγραφο κείμενο μίας εικόνας και να προβλέψει κάποιον αριθμό επόμενων λέξεων δεδομένου του κειμένου της εικόνας. Παρατηρήθηκε ότι στην εκμάθηση του μοντέλου CNN τα 2 layers αποφέρουν καλύτερη ακρίβεια σε σχέση με το 1 layer. Επίσης, ο αριθμός του batch size φαίνεται να επηρεάζει την ακρίβεια του μοντέλου και να είναι ανάλογος και με τον χρόνο εκτέλεσης (όσο πιο μικρό batch size, τόσο μεγαλύτερος χρόνος εκτέλεσης).

Σε συνδυασμό με την αποτελεσματική προ-επεξεργασία της εικόνας, αυτό αποσκοπεί στην καλή αναγνώριση του κειμένου της εικόνας. Πιο συγκεκριμένα, επιτεύχθηκε ακρίβεια **99.834%** σε **120 λεπτά** του CNN μοντέλου και με την κατάλληλη προ-επεξεργασία το μοντέλο προβλέπει καλύτερα τους χαρακτήρες της εικόνας.

Επιπλέον, η πρόβλεψη επόμενων λέξεων, επιτεύχθηκε με την χρήση αρχικά ενός μοντέλου Markov και έπειτα ενός μοντέλου LSTM. Παρατηρήθηκε ότι στην εκμάθηση του μοντέλου LSTM, η τιμή του dropout από 0.2 σε 0.5 βοήθησε το μοντέλο λόγω του overfitting και συνεπώς στην αύξηση της ακρίβειας με την παραπάνω εκπαίδευση του (από 11 epochs το μοντέλο κατάφερε να τρέξει στα 20 epochs χωρίς να κάνει overfitting).

Το Markov μοντέλο, το οποίο είναι πολύ πιο γρήγορο στην εκμάθηση του (λίγα δευτερόλεπτα) από το LSTM, παρουσιάζει αξιοπρεπή αποτελέσματα τα οποία βγάζουν νόημα στην αγγλική γλώσσα αλλά δεν παρουσιάζουν κάποια ιδιαίτερη λογική. Αντιθέτως, το LSTM παρόλο που είναι πολύ πιο αργό στην εκμάθηση του (ακρίβεια **14.751%** σε **150 λεπτά**), παρουσιάζει αποτελέσματα τα οποία έχουν

κάποια συντακτική και γραμματική λογική που θα μπορούσε να είναι πραγματικό κείμενο.

Κάποιες μελλοντικές βελτιώσεις θα ήταν μεγαλύτερα σύνολα δεδομένων για την εκμάθηση των μοντέλων CNN και LSTM (καθώς τα μοντέλα παρουσιάζουν overfitting μετά από κάποιες επαναλήψεις και ιδιαίτερα το LSTM) και αναγνώριση κειμένου με σημεία στίξης και τόνους.

8 *Βιβλιογραφία*

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks, 5*, 157–166.

Bishop, C. M. (2006). *Pattern recognition and machine learning.* springer.

Bishop, C. M., & others. (1995). *Neural networks for pattern recognition.* Oxford university press.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the fifth annual workshop on Computational learning theory*, (pp. 144–152).

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine learning, 20*, 273–297.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., & Wei, Y. (2017). Deformable convolutional networks. *Proceedings of the IEEE international conference on computer vision*, (pp. 764–773).

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, *1*, pp. 886–893.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics, 7*, 179–188.

Girshick, R. (2015). Fast r-cnn. *Proceedings of the IEEE international conference on computer vision*, (pp. 1440–1448).

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 580–587).

Goodfellow, I. J., Koenig, N., Muja, M., Pantofaru, C., Sorokin, A., & Takayama, L. (2010). Help me help you: Interfaces for personal robots. *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, (pp. 187–188).

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* MIT press.

Graves, A. (2012). Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks* (pp. 5–13). Springer.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A.-r., Jaitly, N., . . . others. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine, 29*, 82–97.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation, 9*, 1735–1780.

Ioffe, S., & Szegedy, C. (2015, 2 11). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, (pp. 1097–1105).

LeCun, Y., & others. (1989). Generalization and network design strategies. *Connectionism in perspective, 19*, 143–155.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*, 2278–2324.

Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017). Focal loss for dense object detection. *Proceedings of the IEEE international conference on computer vision*, (pp. 2980–2988).

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. *European conference on computer vision*, (pp. 21–37).

Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the seventh IEEE international conference on computer vision*, *2*, pp. 1150–1157.

Mitchell, T. M. (1997). *Machine Learning* (1 ed.). USA: McGraw-Hill, Inc.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective.*

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. *Proceedings of the 27th international conference on machine learning (ICML-10)*, (pp. 807–814).

Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W., & Lin, D. (2019). Libra r-cnn: Towards balanced learning for object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 821–830).

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *International conference on machine learning*, (pp. 1310–1318).

Pulli, K., Baksheev, A., Kornyakov, K., & Eruhimov, V. (2012). Real-time computer vision with OpenCV. *Communications of the ACM, 55*, 61–69.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 779–788).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, (pp. 91–99).

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *nature, 323*, 533–536.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal, 27*, 379–423.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research, 15*, 1929–1958.

Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the Gap to Human-Level Performance in Face Verification. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, (pp. 1701-1708).

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, *1*, pp. I–I.

Viola, P., Jones, M., & others. (2001). Robust real-time object detection. *International journal of computer vision, 4*, 4.

Zhang, S., Wen, L., Bian, X., Lei, Z., & Li, S. Z. (2018). Single-shot refinement neural network for object detection. *Proceedings of the IEEE conference on computer vision and pattern recognition*, (pp. 4203–4212).

9 *Παράρτημα*

Script για το LSTM μοντέλο

```
1.  from keras.preprocessing.sequence import pad_sequences
2.  from keras.layers import Embedding, Dense, Dropout, CuDNNLSTM
3.  from keras.preprocessing.text import Tokenizer
4.  from keras.callbacks import EarlyStopping
5.  from keras.models import Sequential
6.  from keras.optimizers import Adam
7.  import keras.utils as ku
8.  import numpy as np
9.  import keras
10. from keras.optimizers import Adam
11. from keras.callbacks import ModelCheckpoint
12.
13. tokenizer = Tokenizer()
14. optimizer = keras.optimizers.Adam(lr=0.0005)
15. def dataset_preparation(data):
16.     # basic cleanup
17.     corpus = data.lower().split("\n")
18.     # tokenization
19.     tokenizer.fit_on_texts(corpus)
20.     total_words = len(tokenizer.word_index) + 1
21.     # create input sequences using list of tokens
22.     input_sequences = []
23.     for line in corpus:
24.         token_list = tokenizer.texts_to_sequences([line])[0]
25.         for i in range(1, len(token_list)):
26.             n_gram_sequence = token_list[:i+1]
27.             input_sequences.append(n_gram_sequence)
28.     # pad sequences
29.     max_sequence_len = max([len(x) for x in input_sequences])
30.     input_sequences = np.array(pad_sequences(input_sequences, maxlen=max_seq
    uence_len, padding='pre'))
31.     # create predictors and label
32.     predictors, label = input_sequences[:,:-1],input_sequences[:,-1]
33.     label = ku.to_categorical(label, num_classes=total_words)
34.     return predictors, label, max_sequence_len, total_words
35. def create_model(predictors, label, max_sequence_len, total_words):
36.     model = Sequential()
37.     model.add(Embedding(total_words, 10, input_length=max_sequence_len-1))
38.     model.add(CuDNNLSTM(1024, return_sequences = True))
39.     model.add(Dropout(0.5))
40.     model.add(CuDNNLSTM(1024))
41.     model.add(Dropout(0.5))
42.     model.add(Dense(total_words, activation='softmax'))
43.     checkpoint = ModelCheckpoint('run_1_5_model.h5', verbose=1, monitor='val
    _acc',save_best_only=True, mode='max')
```

```
    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['acc'])
    model.fit(predictors, label, callbacks=[checkpoint], validation_split=0.1, batch_size=32, epochs=25, verbose=1)
    model.summary()
    return model
def generate_text(seed_text, next_words, max_sequence_len):
    for _ in range(next_words):
        token_list = tokenizer.texts_to_sequences([seed_text])[0]
        token_list = pad_sequences([token_list], maxlen=max_sequence_len-1, padding='pre')
        predicted = model.predict_classes(token_list, verbose=0)
        output_word = ""
        for word, index in tokenizer.word_index.items():
            if index == predicted:
                output_word = word
                break
        seed_text += " " + output_word
    return seed_text
data = open('data.txt').read()
predictors, label, max_sequence_len, total_words = dataset_preparation(data)

model = create_model(predictors, label, max_sequence_len, total_words)
```

Script για την προ-επεξεργασία της εικόνας εισόδου και αναγνώρισης του κειμένου

```
import cv2
import numpy as np
import pickle
import math
from scipy import ndimage
def getBestShift(img):
    cy,cx = ndimage.measurements.center_of_mass(img)
    rows,cols = img.shape
    shiftx = np.round(cols/2.0-cx).astype(int)
    shifty = np.round(rows/2.0-cy).astype(int)
    return shiftx,shifty
def shift(img,sx,sy):
    rows,cols = img.shape
    M = np.float32([[1,0,sx],[0,1,sy]])
    shifted = cv2.warpAffine(img,M,(cols,rows))
    return shifted
def numberToLetter(number):
    number = number + 1
    if number==1:
        letter = 'A'
    elif number==2:
        letter = 'B'
    elif number==3:
        letter = 'C'
    elif number==4:
        letter = 'D'
    elif number==5:
        letter = 'E'
    elif number==6:
```

```
        letter = 'F'
    elif number==7:
        letter = 'G'
    elif number==8:
        letter = 'H'
    elif number==9:
        letter = 'I'
    elif number==10:
        letter = 'J'
    elif number==11:
        letter = 'K'
    elif number==12:
        letter = 'L'
    elif number==13:
        letter = 'M'
    elif number==14:
        letter = 'N'
    elif number==15:
        letter = 'O'
    elif number==16:
        letter = 'P'
    elif number==17:
        letter = 'Q'
    elif number==18:
        letter = 'R'
    elif number==19:
        letter = 'S'
    elif number==20:
        letter = 'T'
    elif number==21:
        letter = 'U'
    elif number==22:
        letter = 'V'
    elif number==23:
        letter = 'W'
    elif number==24:
        letter = 'X'
    elif number==25:
        letter = 'Y'
    else:
        letter = 'Z'
    return letter
#Load the trained model
loaded_model = pickle.load(open('1finalized_model_Keras_Char_Rec.sav', 'rb')
)
# Load the image
img = cv2.imread('IAMYOU.jpg')
# convert to grayscale
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
gray = cv2.medianBlur(gray,5)
v = np.median(gray)
s = 0.33
if v > 191:
    lower = int(max(0, (1 - 2*s) * (255 - v)))
    upper = int(min(85, (1 + 2*s) * (255 - v)))
elif v > 127:
    lower = int(max(0, (1 - s) * (255 - v)))
    upper = int(min(255, (1 + s) * (255 - v)))
elif v < 63:
    lower = int(max(0, (1 - 2*s) * v))
    upper = int(min(85, (1 + 2*s) * v))
else:
    lower = int(max(0, (1 - s) * v))
```

```
    upper = int(min(85, (1 + s) * v))
# apply automatic Canny edge detection using the computed median
gray = cv2.Canny(gray, lower, upper)
# Apply adaptive threshold
thresh = cv2.adaptiveThreshold(gray,255,1,1,11,2)
## find and draw the upper and lower boundary of each lines
hist = cv2.reduce(thresh,1, cv2.REDUCE_AVG).reshape(-1)
th = 0
          H,W = img.shape[:2]
          uppers = [y for y in range(H-1) if hist[y]<=th and hist[y+1]>th]
          lowers = [y for y in range(H-1) if hist[y]>th and hist[y+1]<=th]
          thresh = cv2.cvtColor(thresh, cv2.COLOR_GRAY2BGR)
          for y in uppers:
                  cv2.line(thresh, (0,y), (W, y), (255,0,0), 1)
          for y in lowers:
                  cv2.line(thresh, (0,y), (W, y), (0,255,0), 1)
          H,W = img.shape[:2]
          output = ""
          for K in range(0,len(uppers)):
                  cropped_line = img[uppers[K]:lowers[K]]
                  # convert to grayscale
                  gray = cv2.cvtColor(cropped_line,cv2.COLOR_BGR2GRAY)

                  gray = cv2.medianBlur(gray,5)
                  v2 = np.median(gray)
                  s = 0.33
                  if v2 > 191:
                      lower = int(max(0, (1 - 2*s) * (255 - v2)))
                      upper = int(min(85, (1 + 2*s) * (255 - v2)))
                  elif v2 > 127:
                      lower = int(max(0, (1 - s) * (255 - v2)))
                      upper = int(min(255, (1 + s) * (255 - v2)))
                  elif v2 < 63:
                      lower = int(max(0, (1 - 2*s) * v2))
                      upper = int(min(85, (1 + 2*s) * v2))
                  else:
                      lower = int(max(0, (1 - s) * v2))
                      upper = int(min(85, (1 + s) * v2))
                  # apply automatic Canny edge detection using the computed
    median
                  gray = cv2.Canny(gray, lower, upper)
                  # Apply adaptive threshold
                  thresh = cv2.adaptiveThreshold(gray,255,1,1,11,2)
                  # apply some dilation and erosion to join the gaps
                  kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (16,
     16))
                  #an i eikona einai sxetika mikrh
                  if H < 850 and W < 850:
                      thresh = cv2.dilate(thresh,kernel,iterations = 2)
                  else:
                      thresh = cv2.dilate(thresh,kernel,iterations = 11)
                  (height,width,_) = cropped_line.shape
                  # Find the contours for WORDS
                  _,contours,hierarchy = cv2.findContours(thresh,cv2.RETR_EX
    TERNAL,cv2.CHAIN_APPROX_NONE)
                  img_temp = cropped_line.copy()
                  #vector initializations
                  texts = []
                  counter = 0
                  #For each contour, find the bounding rectangle and draw it

                  for cnt in contours:
                      x,y,w,h = cv2.boundingRect(cnt)
```

```
                        # Don't plot small false positives that aren't text
                        if H < 850 and W < 850:
                            if w < 80 and h < 80:
                                print(3)
                                continue
                        else:
                            if w < 180 and h < 180:
                                print(3)
                                continue
                        cv2.rectangle(img_temp,(x,y),(x+w,y+h),(0,255,0),1)
                        cv2.rectangle(thresh,(x,y),(x+w,y+h),(0,255,0),1)
                        boundingBoxesVertical = [cv2.boundingRect(c) for c in
    contours]
                        (cntsVertical, boundingBoxesVertical) = zip(*sorted(zi
    p(contours, boundingBoxesVertical),key=lambda b:b[1][0], reverse=False))

                        #do same procedure for each word now
                    #take each word as an image to precess it seperately
                    for i in range(0,len(cntsVertical)):
                        x2, y2, w2, h2 = cv2.boundingRect(cntsVertical[i])
                        cropped_img = img_temp[y2+2:y2+h2-1,x2+2:x2+w2-1]
                        # convert to grayscale
                        gray2 = cv2.cvtColor(cropped_img,cv2.COLOR_BGR2GRAY)
                        gray2 = cv2.medianBlur(gray2,5)
                        v3 = np.median(gray2)
                        s = 0.33
                        if v3 > 191:
                            lower = int(max(0, (1 - 2*s) * (255 - v3)))
                            upper = int(min(85, (1 + 2*s) * (255 - v3)))
                        elif v3 > 127:
                            lower = int(max(0, (1 - s) * (255 - v3)))
                            upper = int(min(255, (1 + s) * (255 - v3)))
                        elif v3 < 63:
                            lower = int(max(0, (1 - 2*s) * v3))
                            upper = int(min(85, (1 + 2*s) * v3))
                        else:
                            lower = int(max(0, (1 - s) * v3))
                            upper = int(min(85, (1 + s) * v3))
                        # apply automatic Canny edge detection using the compu
    ted median
                        gray2 = cv2.Canny(gray2, lower, upper)
                        # Apply adaptive threshold
                        thresh2 = cv2.adaptiveThreshold(gray2,255,1,1,11,2)
                        thresh2_color = cv2.cvtColor(thresh2,cv2.COLOR_GRAY2BG
    R)
                        if H < 850 and W < 850:
                            thresh2 = cv2.dilate(thresh2,None,iterations = 2)

                        else:
                            thresh2 = cv2.dilate(thresh2,None,iterations = 10)

                        _,contours2,hierarchy = cv2.findContours(thresh2,cv2.R
    ETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)
                        boundingBoxes2 = [cv2.boundingRect(c) for c in contour
    s2]
                        (cnts2, BoundingBoxes) = zip(*sorted(zip(contours2, bo
    undingBoxes2),key=lambda b:b[1][0], reverse=False))
                        for cn in cnts2:
                            x3,y3,w3,h3 = cv2.boundingRect(cn)
                            if H < 850 and W < 850:
                                if (w3 < 40 and h3 < 40):
                                    print(3)
                                    continue
```

```
                            else:
                                if (w3 < 80 and h3 < 80):
                                    print(3)
                                    continue
                            cv2.rectangle(cropped_img,(x3,y3),(x3+w3,y3+h3),(0
,255,0),1)
                            cv2.rectangle(thresh2_color,(x3,y3),(x3+w3,y3+h3),
(0,255,0),1)
                        for l in range(0,len(cnts2)):
                            x4,y4,w4,h4 = cv2.boundingRect(cnts2[l])

                            if H < 850 and W < 850:
                                if (w3 < 20 and h3 < 20):
                                    print(3)
                                    continue
                            else:
                                if (w3 < 50 and h3 < 50):
                                    print(3)
                                    continue
                            cropped_img2 = cropped_img[y4+2:y4+h4-
1,x4+2:x4+w4-1]
                            cv2.imwrite('test_images_for_deletion/extracted_ch
aracters%d%d%d.jpg' %(i,l,K),cropped_img2)
                        #REGULATE TO MNIST DATASET STANDARDS
                            gray = cv2.imread("test_images_for_deletion/extrac
ted_characters%d%d%d.jpg" %(i,l,K))
                            gray = cv2.cvtColor(gray,cv2.COLOR_BGR2GRAY)

                            gray = cv2.GaussianBlur(gray,(3,3),0)

                            (thresh, gray) = cv2.threshold(gray, 127, 255, cv2
.THRESH_BINARY | cv2.THRESH_OTSU)
                            gray = cv2.resize(255-
gray, (28, 28))
                            while np.sum(gray[0]) == 0:
                                gray = gray[1:]
                            while np.sum(gray[:,0]) == 0:
                                gray = np.delete(gray,0,1)
                            while np.sum(gray[-1]) == 0:
                                gray = gray[:-1]
                            while np.sum(gray[:,-1]) == 0:
                                gray = np.delete(gray,-1,1)
                            rows,cols = gray.shape
                            if rows > cols:
                                factor = 20.0/rows
                                rows = 20
                                cols = int(round(cols*factor))
                                gray = cv2.resize(gray, (cols,rows))
                            else:
                                factor = 20.0/cols
                                cols = 20
                                rows = int(round(rows*factor))
                                gray = cv2.resize(gray, (cols, rows))
                            colsPadding = (int(math.ceil((28-
cols)/2.0)),int(math.floor((28-cols)/2.0)))
                            rowsPadding = (int(math.ceil((28-
rows)/2.0)),int(math.floor((28-rows)/2.0)))
                            gray = np.lib.pad(gray,(rowsPadding,colsPadding),'
constant')
                            shiftx,shifty = getBestShift(gray)
                            shifted = shift(gray,shiftx,shifty)
                            gray = shifted
                            texts.append(gray)
```

```
                        #PREDICTION
                            texts[counter] = texts[counter].reshape(-
    1, 28, 28, 1).astype('float32')
                            texts[counter] = texts[counter] / 255
                            pred = numberToLetter(loaded_model.predict_classes
    (texts[counter]))
                            output += pred
                            counter = counter + 1
                        output += " "

            print(output)
            #filelist=glob.glob("test_images_for_deletion/*.jpg")
            #for file in filelist:
            #    os.remove(file)
```

Script για την εκμάθηση του μοντέλου CNN

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
from sklearn.model_selection import train_test_split
import numpy as np
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint

# seed for reproducing same results
seed = 785
np.random.seed(seed)
# load dataset
dataset = np.loadtxt('A_Z Handwritten Data.csv', delimiter=',')
# split into input and output variables
X = dataset[:,1:785]
Y = dataset[:,0]
# split the data into training (50%) and testing (50%)
(X_train, X_test, Y_train, Y_test) = train_test_split(X, Y, test_size=0.1, r
    andom_state=seed)
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')
randomize = np.arange(len(X_test))
np.random.shuffle(randomize)
X_test = X_test[randomize]
Y_test = Y_test[randomize]
randomize2 = np.arange(len(X_train))
np.random.shuffle(randomize2)
X_train = X_train[randomize2]
Y_train = Y_train[randomize2]
X_train = X_train / 255
X_test = X_test / 255
# one hot encode outputs
Y_train = np_utils.to_categorical(Y_train)
Y_test = np_utils.to_categorical(Y_test)
num_classes = Y_test.shape[1]
# create model
model = Sequential()
```

```
model.add(Conv2D(64, (5, 5), input_shape=(28, 28, 1), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))
model.add(Conv2D(32, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.4))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(50, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
# Compile model
checkpoint = ModelCheckpoint('model_1.h5', verbose=1, monitor='val_accuracy'
,save_best_only=True, mode='max')
model.compile(loss='categorical_crossentropy', optimizer=Adam(lr=0.0005), me
trics=['accuracy'])
model.fit(X_train, Y_train,callbacks=[checkpoint], validation_data=(X_test,
Y_test), epochs=300, batch_size=128, verbose=1)
```

Script για το Markov model

```
import string
import numpy as np
from heapq import nlargest
from random import randint

training_data_file = 'anna.txt'
def remove_punctuation(sentence):
    return sentence.translate(str.maketrans('','', string.punctuation))
def add2dict(dictionary, key, value):
    if key not in dictionary:
        dictionary[key] = []
    dictionary[key].append(value)
def list2probabilitydict(given_list):
    probability_dict = {}
    given_list_length = len(given_list)
    for item in given_list:
        probability_dict[item] = probability_dict.get(item, 0) + 1
    for key, value in probability_dict.items():
        probability_dict[key] = value / given_list_length
    return probability_dict
initial_word = {}
second_word = {}
transitions = {}
# Trains a Markov model based on the data in training_data_file
def train_markov_model():
    for line in open(training_data_file):
        tokens = remove_punctuation(line.rstrip().lower()).split()
        tokens_length = len(tokens)
        for i in range(tokens_length):
            token = tokens[i]
            if i == 0:
                initial_word[token] = initial_word.get(token, 0) + 1
            else:
                prev_token = tokens[i - 1]
                if i == tokens_length - 1:
                    add2dict(transitions, (prev_token, token), 'END')
                if i == 1:
                    add2dict(second_word, prev_token, token)
```

```
                else:
                    prev_prev_token = tokens[i - 2]
                    add2dict(transitions, (prev_prev_token, prev_token), tok
en)

    # Normalize the distributions
    initial_word_total = sum(initial_word.values())
    for key, value in initial_word.items():
        initial_word[key] = value / initial_word_total

    for prev_word, next_word_list in second_word.items():
        second_word[prev_word] = list2probabilitydict(next_word_list)

    for word_pair, next_word_list in transitions.items():
        transitions[word_pair] = list2probabilitydict(next_word_list)

    print('Training successful.')
train_markov_model()
def sample_word(dictionary):
    return nlargest(3,dictionary, key=dictionary.get)
    assert(False)
number_of_sentences = 1
# Function to generate sample text
def generate(input):
    output = {}
    for i in range(number_of_sentences):
        num_words = len(input.split())
        if ( num_words == [1]):
            sentence = []
            # Initial word
            word0 = input
            sentence.append(word0)
            # Second word
            word2 = sample_word(second_word[word0])
            sentence.append(word2[randint(0, 2)]) #random from 1 to 3 from b
iggest candidates
            print(' '.join(sentence))
        else:
            sentence = []
            for items in range(0,len(input.split())):
                # Initial word
                word= {}
                word = input.split()[items]

                sentence.append(word)
            word2 = sample_word(second_word[word])
            sentence.append(word2[randint(0, 2)])
    output = ' '.join(sentence)
    return output
```