Νευρωνικά Δίκτυα

1η Εργασία

Η βάση δεδομένων που χρησιμοποιήθηκε είναι η MNIST και φορτώθηκε από το σύνολο δεδομένων στο API keras του tensorflow στην python. Η MNIST αποτελείται από φωτογραφίες ψηφίων γραμμένων με το χέρι, που ανήκουν σε 10 κλάσεις, από το 0 μέχρι και το 9. Υπάρχουν 60000 φωτογραφίες που αποτελούν το σετ εκπαίδευσης, 60000 ετικέτες που εκπροσωπούν αυτές τις φωτογραφίες και 10000 φωτογραφίες που αποτελούν το τεστ σετ, μαζι με τις 10000 ετικέτες τους αντίστοιχα. Οι φωτογραφίες είναι 28x28 pixel, και περιέχουν μία τιμή φωτεινότητας από το 0 έως το 255.

Αρχικά οι τιμές φωτεινότητας κανονικοποιούνται μεταξύ 0 και 1 έτσι ώστε να δωθεί στα δεδομένα μία κοινή κατάσταση, όλες οι μεταβλητές να είναι δηλαδή κεντραρισμένες παρόμοια. Είναι πολύ σημαντική η διαδικασία αυτή, τόσο για την εκπαίδευση του νευρωνικού δικτύου στη συνέχεια, όσο και για την μειώση διαστάσεων με PCA.

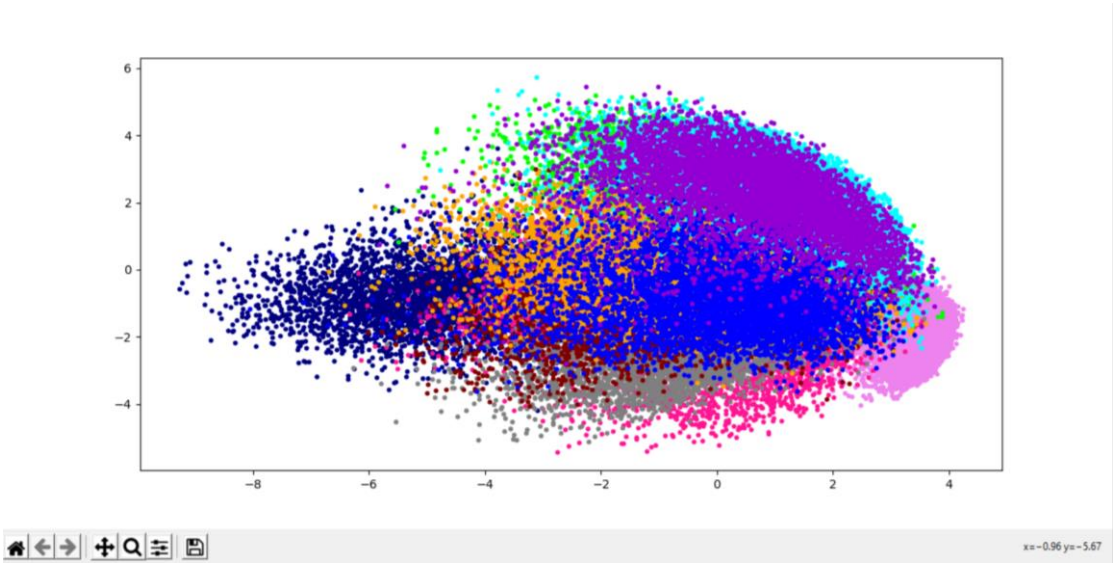
Έπειτα ο πίνακας όπου είναι αποθηκεύμενες αυτές οι τιμές για τα δεδομένα εκπαίδευσης και τεστ, μετατρέπεται απο τρισδιάστατος σε δισδιάστατος, κάνοντας τις διαστάσεις 28x28, 784.

- ### PCA

Στη συνέχεια δοκιμάζουμε να εφαρμόσουμε μείωση διαστάσεων στις φωτογραφίες με τη χρήση της ανάλυσης πρωτευουσών συνιστωσών (Principal Component Analysis). Προσπαθούμε να μειώσουμε τις 784 διαστάσεις. Αρχικά, αφαιρείται το μέσο όλων των γραμμών από κάθε γραμμή, ώστε να κανονικοποιήσουμε τις μεταβλητές. Έπειτα υπολογίζεται ο πίνακας συνδιασποράς, ώστε να κατανοήσουμε πως οι μεταβλητές ποικίλουν η μία με την άλλη από το μέσο και να δούμε αν υπάρχει μεταξύ τους κάποια σχέση. Πολλές φορές επίσης περιέχουν αχρείαστη πληροφορία. Ο πίνακας αυτός είναι συμμετρικός και έχει σαν καταχωρήσεις τις συνδιασπορές που σχετίζονται με όλα τα πιθανά ζευγάρια των αρχικών μεταβλητών. Η κύρια διαγώνιος ουσιαστικά περιέχει τη διασπορά όλων των μεταβλητών, και επειδή οι καταχωρήσεις είναι συμμετρικές ως προς την κύρια διαγώνιο, ο άνω και κάτω τριγωνικός πίνακας είναι ίσοι. Το πρόσημο λοιπόν του πίνακα, μας δείχνει πως σχετίζονται οι μεταβλητές μεταξύ τους. Έπειτα υπολογίζονται οι ιδιοτιμές και τα ιδιοδιανύσματα του πίνακα συνδιασποράς, ώστε να οριστούν οι κύριες συνιστώσες. Τα ιδιοδιανύσματα του πίνακα ουσιαστικά αποτελούν τις κατευθύνσεις των αξόνων όπου υπάρχει η περισσότερη διασπορά, δηλαδή η περισσότερη πληροφορία, και οι ιδιοτιμές είναι οι συντελεστές των διανυσμάτων. Οπότε, ταξινομούμε με φθίνουσα σειρά τα ιδιοδιανύσματα με βάση τις ιδιοτιμές τους, και έτσι παίρνουμε τις κύριες συνιστώσες με βάση τη σημαντικότητα τους. Έτσι απο αυτές κρατάμε όποιες επιθυμούμε απο τις κ πρώτες και αυτό το βήμα αποτελεί τον υπολογισμό του feature vector (πίνακα χαρακτηριστικών). Τέλος πρέπει να προβάλλουμε τα δεδομένα στον άξονα των κυριών συνιστωσών, πολλαπλασιάζοντας τον πίνακα των αρχικών δεδομένων με τον πίνακα χαρακτηριστικών.

Όσον αφορά το τεστ σετ, αφαιρείται από αυτό το μέσο των δεδομένων εκπαίδευσης, και στη συνέχεια προλλαπλασιάζουμε τον πίνακα χαρακτηριστικών, με τον πίνακα του τεστ σετ

Έχουν επιλεχθεί οι 50 κύριες συνιστώσες. Με scatter plot προβάλλονται τα αποτελέσματα του PCA, χρησιμοποιώντας τις 2 διαστάσεις.



Οι κώδικες των παρακάτω βασίστηκαν σε κάποιους κώδικες από το διαδίκτυο.

• K-NN

Ο αλγόριθμος κ κοντινότερων γειτόνων είναι αλγόριθμος ταξινόμησης ο οποίος διαχωρίζει τα σημεία των δεδομένων σε ομάδες, οι οποίες στη συνέχεια παίρνουν μία ετικέτα. Βασίζεται στο όσο πιο κοντά είναι τα σημεία, τόσο πιο όμοια. Δεδομένου κάποιου συγκεκριμένου κ, στην περίπτωσή μας 1 ή 3, υπολογίζονται οι αποστάσεις μεταξύ ενός δοσμένου σημείου και των σημείων του συνόλου δεδομένων, ταξινομούνται με φθίνουσα σειρά και παίρνονται οι ετικέτες των πρώτων κ καταχωρήσεων. Τέλος επιστρέφεται η πρόβλεψη για το σημείο αυτό. Όσον αφορά την απόσταση, χρησιμοποιείται η ευκλείδια.

• Nearest Centroid

Ο αλγόριθμος κοντινότερου κέντρου βασίζεται στο ότι δεδομένου ενός σημείου, του δίνει μία ετικέτα από τα δεδομένα εκπαίδευσης, ανάλογα με το ποιο κέντρο αυτών είναι κοντά του. Το κέντρο κάθε κλάσης υπολογίζεται στην εκπαίδευση. Μετά, δεδομένου ενός σημείου υπολογίζονται οι αποστάσεις από αυτό σε κάθε κέντρο κλάσης. Από όλες τις αποστάσεις επιλέγεται η μικρότερη και η ετικέτα κλάσης αυτού του κέντρου δίνεται στο σημείο.

Παρακάτω φαίνονται οι επιδόσεις των δύο αλγορίθμων ταξινόμησης, πρώτα εφαρμοσμένων στα δεδομένα μετά απο PCA, έχοντας μειώσει τη διάσταση από 784 σε 50, και έχοντας επιλέξει τα 3000 πρώτα δείγματα από το σύνολο της εκπαίδευσης, και 500 από το τεστ. Στον αλγόριθμο κ- κοντινότερων γειτόνων έχει χρησιμοποιηθεί κ=1 και στις τρεις περιπτώσεις.

1.

```
Predictions of k-nn:  [7 2 1 0 9 1 9 9 5 9 0 6 9 0 1 5 9 7 3 4 9 6 6 5 4 0 7 4 0 1 3 1 3 4 7 2 7
 1 2 1 1 7 4 5 5 5 1 8 4 4 6 3 5 5 6 0 4 1 9 5 7 2 9 3 7 4 6 4 3 0 7 0 2 8
 1 7 3 3 9 7 9 6 2 7 8 4 7 3 6 1 3 6 9 3 1 4 1 7 6 9 6 0 5 4 9 9 2 1 9 4 8
 7 3 9 7 9 4 4 9 2 5 4 9 6 7 9 0 5 8 5 6 6 5 7 8 1 0 1 6 4 6 7 3 1 7 1 8 2
 0 2 9 8 5 5 1 5 6 0 3 4 4 6 5 4 6 5 4 5 1 9 4 7 2 3 2 1 1 8 1 8 1 8 5 0 8
 9 2 5 0 1 1 1 0 9 0 3 1 6 4 2 3 6 1 1 1 3 9 5 2 9 4 5 9 3 9 0 3 5 5 3 7 2
 2 7 1 2 8 4 1 7 3 3 8 8 7 9 2 2 4 1 5 8 9 4 2 2 0 6 4 2 4 1 9 5 7 7 2 8 2
 6 8 5 7 7 9 1 0 1 8 0 3 0 1 9 9 4 1 8 2 1 2 9 7 5 9 2 6 4 1 5 8 2 9 2 0 4
 0 0 2 8 1 7 1 2 4 0 2 7 4 3 3 0 0 5 1 9 6 5 2 5 7 7 9 3 0 4 6 0 7 1 1 2 1
 5 3 3 9 7 8 6 5 4 1 3 8 1 0 5 1 9 1 5 5 6 1 8 5 1 4 9 4 6 2 2 5 0 4 5 6 3
 7 2 0 8 8 5 4 1 1 4 0 7 3 7 6 1 6 3 1 9 2 8 6 1 9 5 2 5 4 4 2 8 3 5 2 4 5
 0 3 1 7 7 5 7 9 7 1 9 2 1 4 2 9 2 0 4 9 1 4 8 1 8 9 5 9 7 8 3 7 6 0 0 3 0
 2 0 6 4 9 3 3 3 2 3 9 1 2 6 8 0 5 6 6 6 7 8 8 2 7 5 8 9 6 1 8 4 1 2 6 8 1
 9 7 5 4 0 8 9 9 1 0 5 8 3 7 0 9 4 0 6]
k-NN classification accuracy:  91.60 %
```

```
Predictions of nearest centroid:  [7 5 1 0 4 1 4 9 2 9 0 2 9 0 1 5 4 7 3 4 9 6 4 5 4 0 7 4 0 1 3 1 3 4 7 2 7
 1 3 1 1 7 4 1 3 5 5 2 4 4 6 3 5 5 2 5 4 1 9 1 7 8 9 2 7 9 2 4 3 0 7 0 2 8
 1 7 3 7 1 7 9 6 2 7 8 4 7 5 6 1 3 6 9 3 1 4 1 1 6 9 4 0 5 4 4 9 2 1 9 4 8
 1 3 9 7 9 9 4 9 7 3 6 7 6 4 9 0 5 8 5 6 6 5 7 8 1 0 1 6 4 6 7 3 1 7 1 8 2
 0 1 9 4 5 5 1 5 6 0 3 4 4 6 5 4 4 3 4 5 1 4 4 7 3 3 2 1 1 8 1 8 1 8 5 0 3
 9 3 3 0 1 1 5 0 4 0 1 1 6 4 2 3 2 1 1 1 3 9 5 2 9 4 5 4 1 9 0 3 5 7 5 7 2
 2 7 1 2 8 4 1 7 3 3 8 7 7 9 2 2 4 1 5 3 8 4 1 5 0 2 4 1 9 1 9 5 7 7 1 1 2
 0 8 1 7 7 5 1 8 1 3 0 3 0 1 9 4 4 1 8 2 1 2 9 1 5 9 2 6 4 1 5 4 3 9 2 0 4
 0 0 2 8 1 9 1 2 9 0 2 9 4 3 3 0 0 5 1 9 6 1 3 5 1 1 9 3 5 9 2 0 7 1 1 1 1
 5 3 3 4 7 8 6 6 4 1 3 5 1 5 5 1 9 1 5 0 6 1 8 5 1 7 9 4 6 7 1 5 5 6 5 6 3
 7 2 0 8 8 5 9 1 1 4 5 7 3 7 6 1 6 2 1 9 2 8 6 1 9 5 2 5 4 4 2 8 3 9 2 4 0
 0 3 1 7 1 3 7 9 7 1 9 2 1 4 2 9 2 0 4 9 1 4 4 1 8 4 4 9 8 8 3 7 6 0 0 3 5
 8 0 6 4 8 5 3 3 1 3 4 1 1 5 8 0 9 4 6 6 7 8 8 2 8 5 8 9 6 1 8 4 1 2 5 3 1
 9 7 1 4 0 9 4 9 1 0 5 2 3 7 6 4 9 5 6]
Nearest Centroid classification accuracy:  76.80 %
```

Έπειτα εξετάζονται οι επιδόσεις σε όλο το σετ, αφού έχει υποβληθεί σε PCA κρατώντας τις 50 διαστάσεις.

2.

```
Predictions of k-nn:  [7 2 1 ... 4 5 6]
k-NN classification accuracy:  97.34 %
Predictions of nearest centroid:  [7 2 1 ... 4 5 6]
Nearest Centroid classification accuracy:  81.75 %
```

Τέλος εξετάζονται οι επιδόσεις σε όλα τα δεδομένα χωρίς μείωση διάστασης ή επιλογή κάποιων από τα 60000.

3.

```
Predictions of k-nn:  [7 2 1 ... 4 5 6]
k-NN classification accuracy:  96.91 %
Predictions of nearest centroid:  [7 2 1 ... 4 5 6]
Nearest Centroid classification accuracy:  82.03 %
```

Οι ίδιες παρατηρήσεις για κ=3:

1.

```
Predictions of k-nn:  [7 2 1 0 4 1 4 9 5 9 0 6 9 0 1 5 9 7 3 4 9 6 6 5 4 0 7 4 0 1 3 1 3 4 7 2 7
 1 2 1 1 7 4 5 5 5 1 8 4 4 6 3 5 5 6 0 4 1 9 5 7 2 9 3 7 4 6 4 3 0 7 0 2 9
 1 7 3 3 9 7 9 6 2 7 8 4 7 3 6 1 3 6 9 3 1 4 1 7 6 9 6 0 5 4 9 9 2 1 9 4 8
 7 3 9 7 9 4 4 9 2 5 4 7 6 7 9 0 5 8 5 6 6 5 7 8 1 0 1 6 4 6 7 3 1 7 1 8 2
 0 2 9 9 5 5 1 5 6 0 3 4 4 6 5 4 6 5 4 5 1 4 4 7 2 3 2 1 1 8 1 8 1 8 5 0 8
 9 2 5 0 1 1 1 0 4 0 3 1 6 4 2 3 6 1 1 1 3 9 5 2 9 4 5 9 3 9 0 3 5 5 5 7 2
 2 7 1 2 8 4 1 7 3 3 8 7 7 9 2 2 4 1 5 8 9 7 2 2 0 6 4 2 9 1 9 5 7 7 2 6 2
 6 8 5 7 7 9 1 0 1 8 0 3 0 1 9 9 4 1 8 2 1 2 9 7 5 9 2 6 4 1 5 8 2 9 2 0 4
 0 0 2 8 1 7 1 2 4 0 2 7 4 3 3 0 0 5 1 9 6 5 2 5 7 7 9 3 0 4 6 0 7 1 1 2 1
 5 3 3 9 7 8 6 5 4 1 3 8 1 0 5 1 9 1 5 5 6 1 8 5 1 4 9 4 6 2 2 5 0 6 5 6 3
 7 2 0 8 8 5 4 1 1 4 0 7 3 7 6 1 6 2 1 9 2 8 6 1 9 5 2 5 4 4 2 8 3 5 2 4 5
 0 3 1 7 7 5 7 9 7 1 9 2 1 4 2 9 2 0 4 9 1 4 8 1 8 4 5 9 7 8 3 7 6 0 0 3 0
 2 0 6 4 9 3 3 3 2 3 9 1 2 6 8 0 9 6 6 6 7 8 8 2 7 5 8 9 6 1 8 4 1 2 6 8 1
 9 7 5 4 0 8 9 9 1 0 5 8 3 7 0 9 4 0 6]
k-NN classification accuracy:  92.80 %
```

2.

```
Predictions of k-nn:  [7 2 1 ... 4 5 6]
k-NN classification accuracy:  97.60 %
```

3.

```
Predictions of k-nn:  [7 2 1 ... 4 5 6]
k-NN classification accuracy:  97.17 %
```

Παρατηρείται πως ο αλγόριθμος των κ κοντινότερων γειτόνων έχει καλύτερη εφαρμογή από αυτόν των κ πλησιέστερων κέντρων, σε όλες τις περιπτώσεις. Τώρα όσον αφορά την επιλογή των κοντινότερων γειτόνων, ο αλγόριθμος έχει καλύτερα αποτελέσματα για κ=3, και στις 3 περιπτώσεις που εξετάστηκαν. Επιπλέον, και για τα 2 κ, κατά αύξουσα σειρά αποδοτικότητας, έχουμε πρώτα την περίπτωση των 50 διαστάσεων συγκεκριμένων δεδομένων, έπειτα όλων των δεδομένων και τέλος των 50 διαστάσεων όλων των δεδομένων.

Αντίθετα, όσον αφορά τον αλγόριθμο των κοντινότερων μέσων, η αποτελεσματικότητα κατά αύξουσα σειρά είναι πρώτα η περίπτωση των 50 διαστάσεων συγκεκριμένων δεδομένων, μετά των 50 διαστάσεων όλων των δεδομένων και τέλος όλων των δεδομένων.

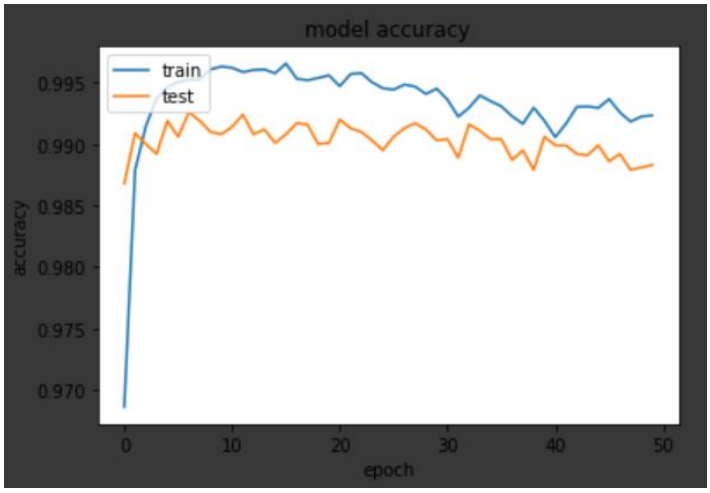
Περνώντας στο κομμάτι των νευρωνικών δικτύων, κάνω ένα συνελικτικό νευρωνικό δίκτυο και έπειτα προσπαθώ να το προ-εκπαιδεύσω με τη χρήση ενός autoencoder, έχοντας για encoder το δίκτυο μου. Αρχικά λοιπόν θα εξετάσω μόνο του το συνελικτικό νευρωνικό δοκιμάζοντας διαφορετικές παραμέτρους και διαφορετικό αριθμό φίλτρων. Γενικά μία καλή και σημαντική προ επεξεργασία δεδομένων πριν την εισαγωγή τους σε ένα νευρωνικό δίκτυο, είναι το augmentation. Δηλαδή να επεξεργαστούμε ελάχιστα τις εικόνες του σετ δεδομένων όπως για παράδειγμα να τις αναποδογυρίσουμε, να τις περιστρέψουμε έτσι ώστε να μεγαλώσουμε το σετ δεδομένων και να βοηθήσουμε το νευρωνικκό να μάθει καλύτερα, εμποδίζοντας το να μάθει αχρείαστα μοτίβα. Ωστόσο στην MNIST αυτή η τεχνική δεν είναι πολύ χρήσιμη καθώς ήδη έχουμε πολλά δεδομένα, αλλά και το τασκ είναι αρκετά εύκολο για το δίκτυο εξαρχής.

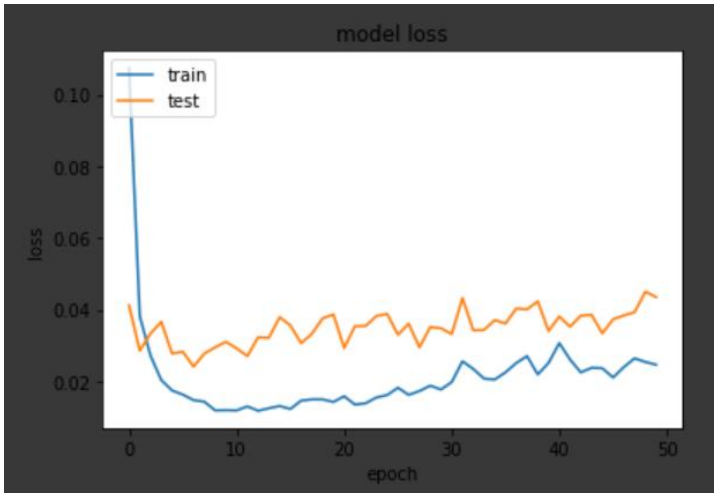
Αρχικά χρησιμοποιώ 3 συνελικτικά επίπεδα, με χρήση πυρήνα 3x3. Στο πρώτο επίπεδο χρησιμοποιούνται 16 φίλτρα, στο δεύτερο 32 και στο τρίτο 64. Έπειτα αφού μετατρέψω την έξοδο του τελευταίου συνελικτικού σε μονοδιάστατη, το περνάω από ένα πλήρως συνδεδεμένο επίπεδο 100 νευρώνων, και έπειτα ένα πλήρως συνδεδεμένο 10 νευρώνων, καθώς οι κλάσεις στις οποίες θα ταξινομηθούν τα δεδομένα μου είναι 10. Εκπαιδεύω το δίκτυο για 50 εποχές και έχω μικρό αριθμό batches ίσο με 8.

```
Epoch 1/50
7500/7500 [==============================] - 20s 3ms/step - loss: 0.1074 - accuracy: 0.9686 - val_loss: 0.0412 - val_accuracy: 0.9868
Epoch 2/50
7500/7500 [==============================] - 20s 3ms/step - loss: 0.0381 - accuracy: 0.9879 - val_loss: 0.0285 - val_accuracy: 0.9909
Epoch 3/50
7500/7500 [==============================] - 20s 3ms/step - loss: 0.0271 - accuracy: 0.9915 - val_loss: 0.0333 - val_accuracy: 0.9900
Epoch 4/50
7500/7500 [==============================] - 20s 3ms/step - loss: 0.0203 - accuracy: 0.9937 - val_loss: 0.0366 - val_accuracy: 0.9892
Epoch 5/50
7500/7500 [==============================] - 20s 3ms/step - loss: 0.0174 - accuracy: 0.9946 - val_loss: 0.0277 - val_accuracy: 0.9919
Epoch 6/50
7500/7500 [==============================] - 20s 3ms/step - loss: 0.0162 - accuracy: 0.9951 - val_loss: 0.0282 - val_accuracy: 0.9906
Epoch 7/50
7500/7500 [==============================] - 20s 3ms/step - loss: 0.0147 - accuracy: 0.9952 - val_loss: 0.0241 - val_accuracy: 0.9926
Epoch 8/50
7500/7500 [==============================] - 20s 3ms/step - loss: 0.0143 - accuracy: 0.9953 - val_loss: 0.0277 - val_accuracy: 0.9919
Epoch 9/50
7500/7500 [==============================] - 20s 3ms/step - loss: 0.0118 - accuracy: 0.9961 - val_loss: 0.0294 - val_accuracy: 0.9910
Epoch 10/50
7500/7500 [==============================] - 20s 3ms/step - loss: 0.0119 - accuracy: 0.9963 - val_loss: 0.0310 - val_accuracy: 0.9908
Epoch 11/50
7500/7500 [==============================] - 20s 3ms/step - loss: 0.0118 - accuracy: 0.9962 - val_loss: 0.0292 - val_accuracy: 0.9914
Epoch 12/50
7500/7500 [==============================] - 20s 3ms/step - loss: 0.0130 - accuracy: 0.9959 - val_loss: 0.0270 - val_accuracy: 0.9924
Epoch 13/50
7500/7500 [==============================] - 20s 3ms/step - loss: 0.0117 - accuracy: 0.9960 - val_loss: 0.0323 - val_accuracy: 0.9908
Epoch 14/50
```

Η ακρίβεια του μοντέλου ξεκινάει από καλό σημείο τόσο για την εκπαίδευση όσο και για το τεστ. Η μέση ακρίβεια του τεστ είναι 98.8%.

Οι γραφικές παραστάσεις:





Φαίνεται πως απο την 30η εποχή και μετά, και οι δύο ακρίβειες αρχίζουν να πεφτούν και αντίστοιχα οι απώλειες να ανεβαίνουν. Αυτό θα μπορούσε να αποτελεί ένδειξη πως δεν χρειάζεται να έχουμε τόσες πολλές εποχές στο μοντέλο, αφού έχει ήδη εκπαιδευτεί ικανοποιητικά.

Ωστόσο έπειτα, αφήνοντας ίδιο τον αριθμό των επιπέδων, και αλλάζοντας τον αριθμό των φίλτρων τους σε 32, 64 και 256 για κάθε συνελικτικό επίπεδο αντίστοιχα, προσπαθώ να εκπαιδεύσω ακόμα περισσότερο το δίκτυο ώστε να δω αν προκληθεί overfitting. Έτσι χρησιμοποιώ 70 εποχές και batch size 4.

```
Epoch 1/70
938/938 [==============================] - 5s 5ms/step - loss: 0.1520 - accuracy: 0.9570 - val_loss: 0.0429 - val_accuracy: 0.9869
Epoch 2/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0389 - accuracy: 0.9886 - val_loss: 0.0344 - val_accuracy: 0.9901
Epoch 3/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0252 - accuracy: 0.9923 - val_loss: 0.0288 - val_accuracy: 0.9912
Epoch 4/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0185 - accuracy: 0.9944 - val_loss: 0.0286 - val_accuracy: 0.9902
Epoch 5/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0141 - accuracy: 0.9961 - val_loss: 0.0259 - val_accuracy: 0.9916
Epoch 6/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0097 - accuracy: 0.9971 - val_loss: 0.0219 - val_accuracy: 0.9934
Epoch 7/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0072 - accuracy: 0.9980 - val_loss: 0.0288 - val_accuracy: 0.9921
Epoch 8/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0082 - accuracy: 0.9978 - val_loss: 0.0236 - val_accuracy: 0.9925
Epoch 9/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0069 - accuracy: 0.9980 - val_loss: 0.0268 - val_accuracy: 0.9920
Epoch 10/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0055 - accuracy: 0.9983 - val_loss: 0.0213 - val_accuracy: 0.9935
```

Η εκπαίδευση ξεκινά πάλι απο πολύ καλό σημείο, ακόμα καλύτερο απο την προηγούμενη δοκιμή.

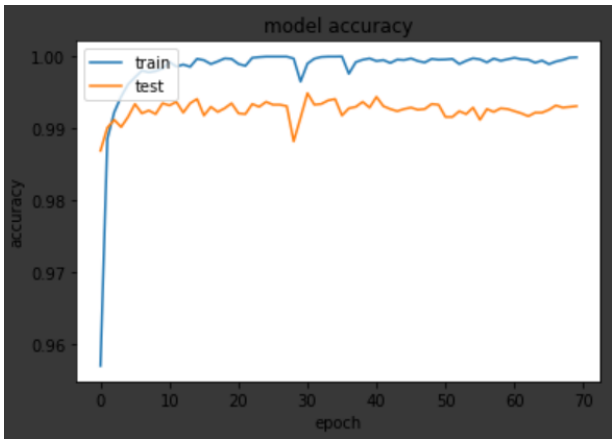
Ωστόσο όπως φαίνεται παρακάτω, σε καποιές εποχές η ακρίβεια της εκπαίδευσης γίνεται 100%, το οποίο συχνά σημαίνει πως το μοντέλο παρουσιάζει overfitting. Παρόλο που η ακρίβεια του τεστ σε αυτές τις εποχές παρουσιάζει κάποια ελάχιστη μείωση, βλέπουμε πως έπειτα ανεβαίνει πάλι, ώντας σε πολύ καλο επίπεδο, πάνω απο 99%! Οπότε μάλλον το μοντέλο δεν έχει υπερ-εκπαιδευτεί.

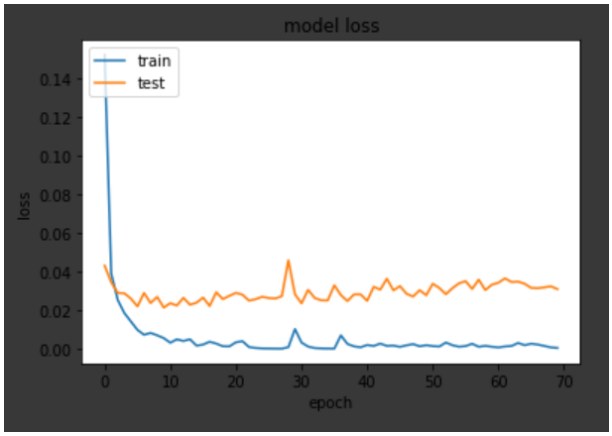
```
Epoch 18/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0027 - accuracy: 0.9993 - val_loss: 0.0292 - val_accuracy: 0.9923
Epoch 19/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0013 - accuracy: 0.9997 - val_loss: 0.0257 - val_accuracy: 0.9928
Epoch 20/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0012 - accuracy: 0.9997 - val_loss: 0.0273 - val_accuracy: 0.9935
Epoch 21/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0033 - accuracy: 0.9990 - val_loss: 0.0289 - val_accuracy: 0.9921
Epoch 22/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0039 - accuracy: 0.9987 - val_loss: 0.0279 - val_accuracy: 0.9920
Epoch 23/70
938/938 [==============================] - 4s 5ms/step - loss: 8.7961e-04 - accuracy: 0.9998 - val_loss: 0.0249 - val_accuracy: 0.9934
Epoch 24/70
938/938 [==============================] - 4s 5ms/step - loss: 3.9839e-04 - accuracy: 0.9999 - val_loss: 0.0256 - val_accuracy: 0.9930
Epoch 25/70
938/938 [==============================] - 4s 5ms/step - loss: 1.8173e-04 - accuracy: 1.0000 - val_loss: 0.0269 - val_accuracy: 0.9937
Epoch 26/70
938/938 [==============================] - 4s 5ms/step - loss: 1.1078e-04 - accuracy: 1.0000 - val_loss: 0.0262 - val_accuracy: 0.9933
Epoch 27/70
938/938 [==============================] - 4s 5ms/step - loss: 6.2855e-05 - accuracy: 1.0000 - val_loss: 0.0260 - val_accuracy: 0.9933
Epoch 28/70
938/938 [==============================] - 4s 5ms/step - loss: 3.9239e-05 - accuracy: 1.0000 - val_loss: 0.0271 - val_accuracy: 0.9931
Epoch 29/70
938/938 [==============================] - 4s 5ms/step - loss: 8.2095e-04 - accuracy: 0.9997 - val_loss: 0.0457 - val_accuracy: 0.9882
```

```
Epoch 65/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0018 - accuracy: 0.9994 - val_loss: 0.0336 - val_accuracy: 0.9922
Epoch 66/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0026 - accuracy: 0.9989 - val_loss: 0.0314 - val_accuracy: 0.9926
Epoch 67/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0022 - accuracy: 0.9993 - val_loss: 0.0313 - val_accuracy: 0.9932
Epoch 68/70
938/938 [==============================] - 4s 5ms/step - loss: 0.0014 - accuracy: 0.9995 - val_loss: 0.0317 - val_accuracy: 0.9929
Epoch 69/70
938/938 [==============================] - 4s 5ms/step - loss: 6.9252e-04 - accuracy: 0.9998 - val_loss: 0.0322 - val_accuracy: 0.9930
Epoch 70/70
938/938 [==============================] - 5s 5ms/step - loss: 4.1120e-04 - accuracy: 0.9999 - val_loss: 0.0308 - val_accuracy: 0.9931
```

Η μέση ακρίβεια που επιτεύχθηκε είναι 99.3 %.

Οι γραφικές παραστάσεις:





Βλέπουμε πως από τις αρχικές κιόλας εποχές το δίκτυο έχει εκπαιδευτεί, οπότε οι τιμές τις ακρίβειας και της απώλειας δεν αλλάζουν πολύ. Ωστόσο υπάρχουν πολλες παλινδρομήσεις πάνω κάτω και έπειτα από ένα σημείο κοντά στο 60, κυρίως στην απώλεια, φαίνεται πως η απώλεια του τεστ αυξάνεται.

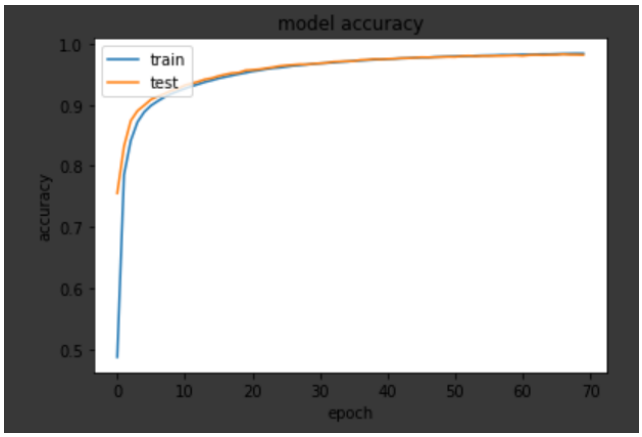
Στη συνέχεια κρατώντας τα πάντα ακριβώς ίδια, αλλάζω μόνο το learning rate, κανοντάς το από 10^{-3}, που είναι η default τιμή του βελτιστοποιητή adam, 10^{-6}. Το learning rate αποτελεί το βήμα αλλαγής βαρών και το κάνω μικρότερο ώστε να δυσκολέψω λίγο το δίκτυο και να εμποδίσω το overfitting.

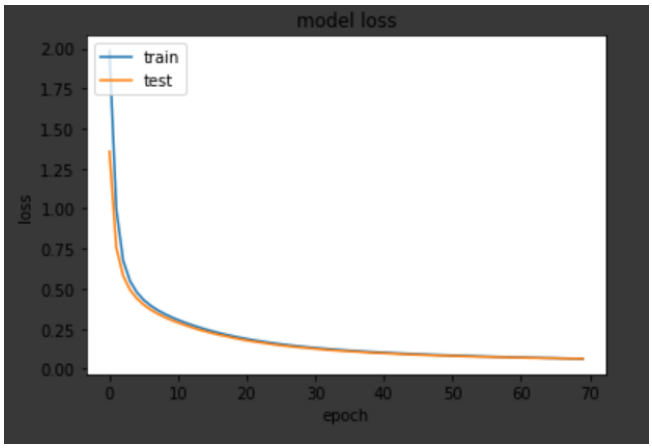
```
Epoch 1/70
15000/15000 [==============================] - 48s 3ms/step - loss: 1.9834 - accuracy: 0.4869 - val_loss: 1.3562 - val_accuracy: 0.7554
Epoch 2/70
15000/15000 [==============================] - 46s 3ms/step - loss: 0.9979 - accuracy: 0.7853 - val_loss: 0.7562 - val_accuracy: 0.8322
Epoch 3/70
15000/15000 [==============================] - 46s 3ms/step - loss: 0.6759 - accuracy: 0.8415 - val_loss: 0.5815 - val_accuracy: 0.8741
Epoch 4/70
15000/15000 [==============================] - 45s 3ms/step - loss: 0.5488 - accuracy: 0.8721 - val_loss: 0.4931 - val_accuracy: 0.8904
Epoch 5/70
15000/15000 [==============================] - 44s 3ms/step - loss: 0.4775 - accuracy: 0.8885 - val_loss: 0.4392 - val_accuracy: 0.8994
Epoch 6/70
15000/15000 [==============================] - 43s 3ms/step - loss: 0.4300 - accuracy: 0.8990 - val_loss: 0.3997 - val_accuracy: 0.9089
Epoch 7/70
15000/15000 [==============================] - 43s 3ms/step - loss: 0.3949 - accuracy: 0.9063 - val_loss: 0.3697 - val_accuracy: 0.9140
Epoch 8/70
15000/15000 [==============================] - 44s 3ms/step - loss: 0.3670 - accuracy: 0.9129 - val_loss: 0.3451 - val_accuracy: 0.9184
Epoch 9/70
15000/15000 [==============================] - 42s 3ms/step - loss: 0.3438 - accuracy: 0.9182 - val_loss: 0.3235 - val_accuracy: 0.9235
Epoch 10/70
15000/15000 [==============================] - 42s 3ms/step - loss: 0.3233 - accuracy: 0.9226 - val_loss: 0.3050 - val_accuracy: 0.9267
Epoch 11/70
15000/15000 [==============================] - 43s 3ms/step - loss: 0.3054 - accuracy: 0.9267 - val_loss: 0.2883 - val_accuracy: 0.9322
Epoch 12/70
15000/15000 [==============================] - 42s 3ms/step - loss: 0.2889 - accuracy: 0.9306 - val_loss: 0.2740 - val_accuracy: 0.9351
Epoch 13/70
15000/15000 [==============================] - 42s 3ms/step - loss: 0.2739 - accuracy: 0.9340 - val_loss: 0.2587 - val_accuracy: 0.9382
```

Σε σχέση με πριν, η ακρίβεια του δικτύου ξεκινά από πολύ χειρότερο σημείο. Επίσης η διαδικασία της εκπαίδευσης είναι αρκετά πιο χρονοβόρα. Και τα δυό οφείλονται στο πολύ μικρό βήμα αλλαγής βαρών.

```
Epoch 60/70
15000/15000 [==============================] - 45s 3ms/step - loss: 0.0728 - accuracy: 0.9822 - val_loss: 0.0706 - val_accuracy: 0.9812
Epoch 61/70
15000/15000 [==============================] - 45s 3ms/step - loss: 0.0717 - accuracy: 0.9826 - val_loss: 0.0711 - val_accuracy: 0.9805
Epoch 62/70
15000/15000 [==============================] - 45s 3ms/step - loss: 0.0709 - accuracy: 0.9825 - val_loss: 0.0685 - val_accuracy: 0.9816
Epoch 63/70
15000/15000 [==============================] - 45s 3ms/step - loss: 0.0697 - accuracy: 0.9830 - val_loss: 0.0687 - val_accuracy: 0.9821
Epoch 64/70
15000/15000 [==============================] - 45s 3ms/step - loss: 0.0687 - accuracy: 0.9832 - val_loss: 0.0674 - val_accuracy: 0.9816
Epoch 65/70
15000/15000 [==============================] - 45s 3ms/step - loss: 0.0678 - accuracy: 0.9832 - val_loss: 0.0667 - val_accuracy: 0.9820
Epoch 66/70
15000/15000 [==============================] - 45s 3ms/step - loss: 0.0670 - accuracy: 0.9836 - val_loss: 0.0665 - val_accuracy: 0.9821
Epoch 67/70
15000/15000 [==============================] - 43s 3ms/step - loss: 0.0660 - accuracy: 0.9839 - val_loss: 0.0649 - val_accuracy: 0.9830
Epoch 68/70
15000/15000 [==============================] - 44s 3ms/step - loss: 0.0652 - accuracy: 0.9841 - val_loss: 0.0642 - val_accuracy: 0.9823
Epoch 69/70
15000/15000 [==============================] - 43s 3ms/step - loss: 0.0643 - accuracy: 0.9843 - val_loss: 0.0643 - val_accuracy: 0.9822
Epoch 70/70
15000/15000 [==============================] - 43s 3ms/step - loss: 0.0634 - accuracy: 0.9842 - val_loss: 0.0631 - val_accuracy: 0.9819
```

Ωστόσο με το τέλος της εκπαίδευσης επιτυγχάνονται πολύ υψηλές ακρίβειες, ωστόσο χαμηλότερες από πριν. Παρατηρώντας όμως τις γραφικές παραστάσεις, βλέπουμε πως είναι περισσότερο ομαλοποιημένες απο πριν. Η μέση ακρίβεια που επιτυγχάνεται είναι 98.2%.



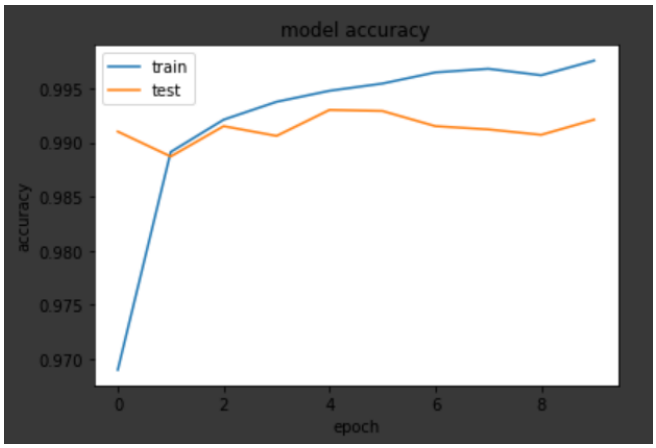


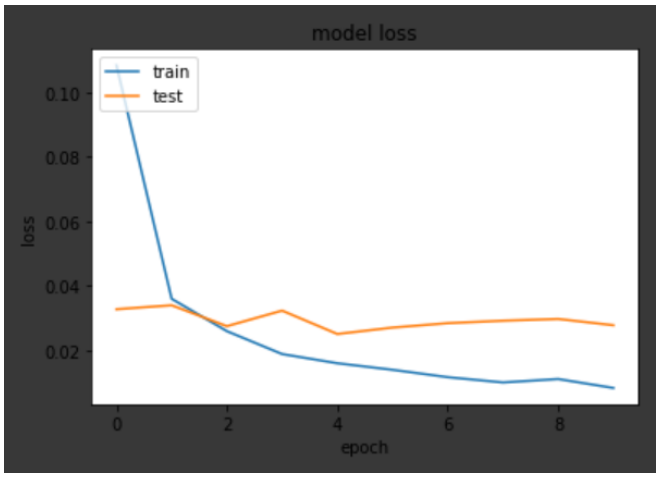
Παρότι πολύ κοντά η μία στην άλλη, βλέπουμε πως κατά μέσο όρο η ακρίβεια του τεστ είναι υψηλότερη από αυτήν της εκπαίδευσης, και αντίστοιχα η απώλεια μικρότερη. Αυτό βγάζει νόημα καθώς το μικρό learning rate δυσκολεύει την εκπαίδευση.

Έπειτα θα κάνω κάποιες δοκιμές κρατώντας τους τελικούς αριθμούς φίλτρων στα συνελικτικά επίπεδα, που θα είναι 32, 64 και 128 αντίστοιχα. Αρχικά θα προσθέσω αρχικοποίηση βαρών, glorot normal. Αφορά τα βάρη με τα οποία ξεκινά το δίκτυο και αναφέρεται ουσιαστικά στην στατιστική κατανομή ή συνάρτηση που θα χρησιμοποιηθεί για την αρχικοποίηση των βαρών. Χρησιμοποιούνται 10 εποχές και batch size 16.

```
Epoch 1/10
3750/3750 [==============================] - 11s 3ms/step - loss: 0.1083 - accuracy: 0.9690 - val_loss: 0.0328 - val_accuracy: 0.9910
Epoch 2/10
3750/3750 [==============================] - 11s 3ms/step - loss: 0.0360 - accuracy: 0.9891 - val_loss: 0.0340 - val_accuracy: 0.9887
Epoch 3/10
3750/3750 [==============================] - 11s 3ms/step - loss: 0.0260 - accuracy: 0.9921 - val_loss: 0.0276 - val_accuracy: 0.9915
Epoch 4/10
3750/3750 [==============================] - 10s 3ms/step - loss: 0.0189 - accuracy: 0.9937 - val_loss: 0.0324 - val_accuracy: 0.9906
Epoch 5/10
3750/3750 [==============================] - 11s 3ms/step - loss: 0.0161 - accuracy: 0.9948 - val_loss: 0.0251 - val_accuracy: 0.9930
Epoch 6/10
3750/3750 [==============================] - 11s 3ms/step - loss: 0.0141 - accuracy: 0.9954 - val_loss: 0.0271 - val_accuracy: 0.9929
Epoch 7/10
3750/3750 [==============================] - 11s 3ms/step - loss: 0.0118 - accuracy: 0.9965 - val_loss: 0.0285 - val_accuracy: 0.9915
Epoch 8/10
3750/3750 [==============================] - 11s 3ms/step - loss: 0.0102 - accuracy: 0.9968 - val_loss: 0.0293 - val_accuracy: 0.9912
Epoch 9/10
3750/3750 [==============================] - 10s 3ms/step - loss: 0.0112 - accuracy: 0.9962 - val_loss: 0.0298 - val_accuracy: 0.9907
Epoch 10/10
3750/3750 [==============================] - 11s 3ms/step - loss: 0.0085 - accuracy: 0.9976 - val_loss: 0.0279 - val_accuracy: 0.9921
```

Η μέση ακρίβεια είναι 99.2% και οι γραφικές παραστάσεις:





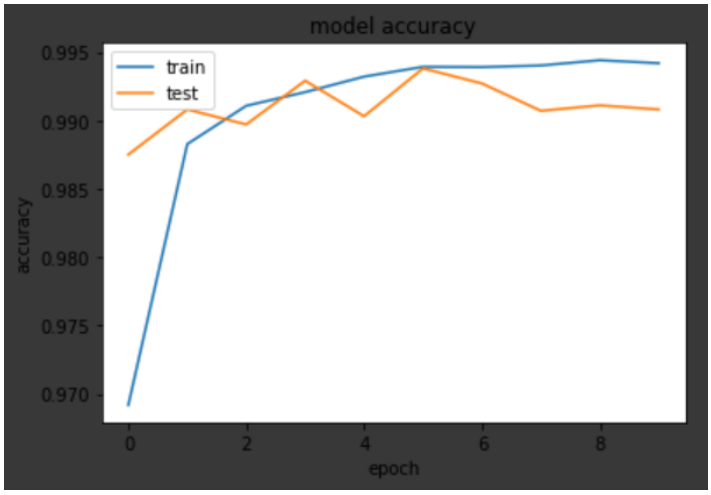
Παρατηρείται πως η ακρίβεια και η απώλεια για το τεστ παραμένουν σταθερά.

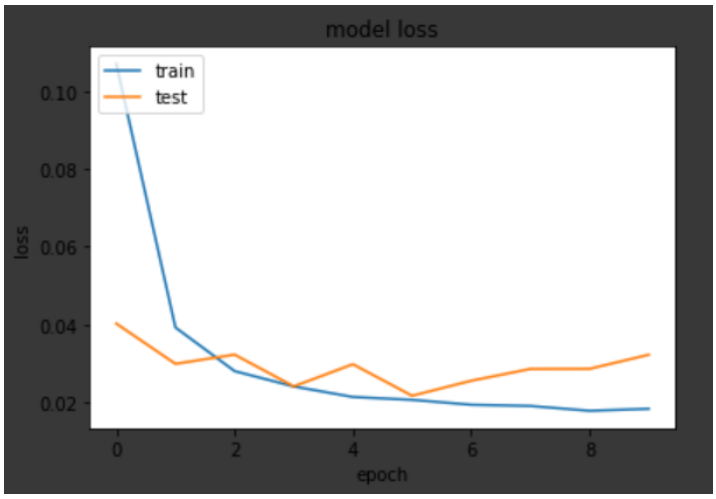
Έπειτα προσθέτω ένα ακόμα επίπεδο μετά από τα 2 πρώτα συνελικτικά, το Drop out layer με παράμετρο αρχικά 0.2, το οποίο θέτει τυχαία τις μονάδες εισόδου σε 0 με μία συχνότητα σε κάθε βήμα του training και βοηθάει στην αποφυγή του overfitting.

Τα αποτελέσματα:

```
Epoch 1/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.1070 - accuracy: 0.9692 - val_loss: 0.0402 - val_accuracy: 0.9875
Epoch 2/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.0392 - accuracy: 0.9883 - val_loss: 0.0298 - val_accuracy: 0.9908
Epoch 3/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.0280 - accuracy: 0.9911 - val_loss: 0.0322 - val_accuracy: 0.9897
Epoch 4/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.0240 - accuracy: 0.9921 - val_loss: 0.0240 - val_accuracy: 0.9929
Epoch 5/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.0213 - accuracy: 0.9932 - val_loss: 0.0297 - val_accuracy: 0.9903
Epoch 6/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.0206 - accuracy: 0.9939 - val_loss: 0.0216 - val_accuracy: 0.9938
Epoch 7/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.0193 - accuracy: 0.9939 - val_loss: 0.0255 - val_accuracy: 0.9927
Epoch 8/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.0190 - accuracy: 0.9940 - val_loss: 0.0285 - val_accuracy: 0.9907
Epoch 9/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.0178 - accuracy: 0.9944 - val_loss: 0.0286 - val_accuracy: 0.9911
Epoch 10/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.0183 - accuracy: 0.9942 - val_loss: 0.0322 - val_accuracy: 0.9908
```

Η μέση ακρίβεια είναι 99%.

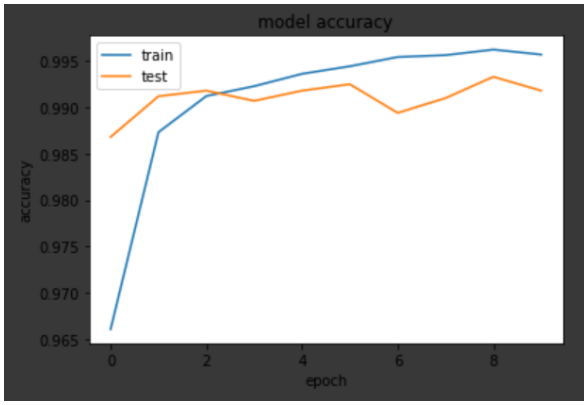


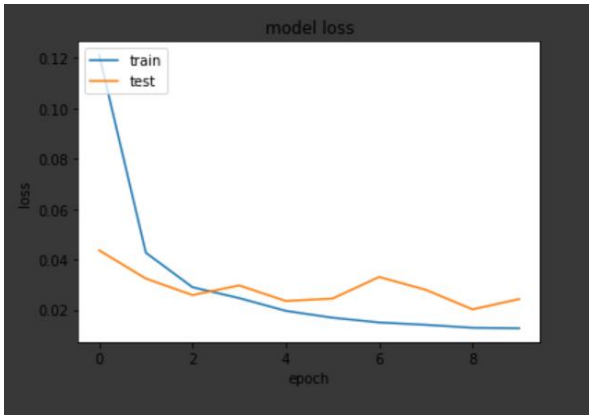


Τώρα δοκιμάζω να μεγαλώσω το dropout σε 0.4.

```
Epoch 1/10
1667/1667 [==============================] - 7s 4ms/step - loss: 0.1212 - accuracy: 0.9661 - val_loss: 0.0436 - val_accuracy: 0.9868
Epoch 2/10
1667/1667 [==============================] - 7s 4ms/step - loss: 0.0427 - accuracy: 0.9873 - val_loss: 0.0324 - val_accuracy: 0.9912
Epoch 3/10
1667/1667 [==============================] - 7s 4ms/step - loss: 0.0290 - accuracy: 0.9912 - val_loss: 0.0258 - val_accuracy: 0.9918
Epoch 4/10
1667/1667 [==============================] - 7s 4ms/step - loss: 0.0246 - accuracy: 0.9923 - val_loss: 0.0297 - val_accuracy: 0.9907
Epoch 5/10
1667/1667 [==============================] - 7s 4ms/step - loss: 0.0196 - accuracy: 0.9936 - val_loss: 0.0235 - val_accuracy: 0.9918
Epoch 6/10
1667/1667 [==============================] - 7s 4ms/step - loss: 0.0168 - accuracy: 0.9944 - val_loss: 0.0244 - val_accuracy: 0.9925
Epoch 7/10
1667/1667 [==============================] - 7s 4ms/step - loss: 0.0150 - accuracy: 0.9954 - val_loss: 0.0330 - val_accuracy: 0.9894
Epoch 8/10
1667/1667 [==============================] - 7s 4ms/step - loss: 0.0140 - accuracy: 0.9956 - val_loss: 0.0279 - val_accuracy: 0.9910
Epoch 9/10
1667/1667 [==============================] - 7s 4ms/step - loss: 0.0129 - accuracy: 0.9962 - val_loss: 0.0202 - val_accuracy: 0.9933
Epoch 10/10
1667/1667 [==============================] - 7s 4ms/step - loss: 0.0127 - accuracy: 0.9957 - val_loss: 0.0242 - val_accuracy: 0.9918
```

Η μέση ακρίβεια είναι λίγο βελτιωμένη, στο 99.2%

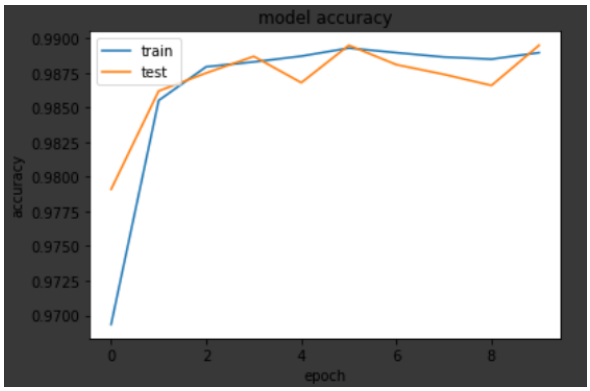


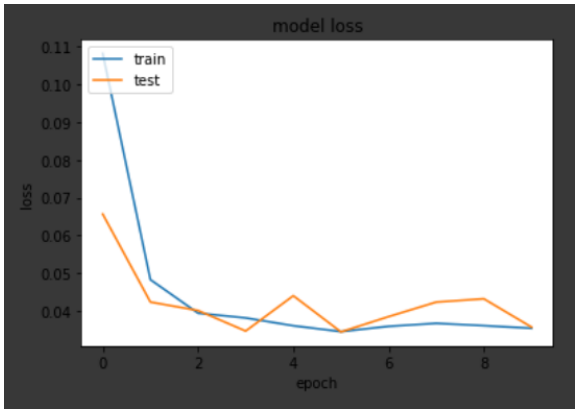


Τέλος προσθέτω ένα Batch Normalization επίπεδο, μετά από τα δύο Drop out, το οποίο κανονικοποιεί την είσοδο του με μέσο κοντά στο 0 και τυπική απόκλιση κοντά στο 1.

```
Epoch 1/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.1082 - accuracy: 0.9693 - val_loss: 0.0657 - val_accuracy: 0.9791
Epoch 2/10
3750/3750 [==============================] - 11s 3ms/step - loss: 0.0483 - accuracy: 0.9855 - val_loss: 0.0423 - val_accuracy: 0.9862
Epoch 3/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.0394 - accuracy: 0.9880 - val_loss: 0.0401 - val_accuracy: 0.9875
Epoch 4/10
3750/3750 [==============================] - 13s 3ms/step - loss: 0.0381 - accuracy: 0.9883 - val_loss: 0.0347 - val_accuracy: 0.9887
Epoch 5/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.0361 - accuracy: 0.9887 - val_loss: 0.0440 - val_accuracy: 0.9868
Epoch 6/10
3750/3750 [==============================] - 13s 3ms/step - loss: 0.0345 - accuracy: 0.9893 - val_loss: 0.0344 - val_accuracy: 0.9895
Epoch 7/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.0359 - accuracy: 0.9890 - val_loss: 0.0385 - val_accuracy: 0.9881
Epoch 8/10
3750/3750 [==============================] - 12s 3ms/step - loss: 0.0367 - accuracy: 0.9887 - val_loss: 0.0423 - val_accuracy: 0.9874
Epoch 9/10
3750/3750 [==============================] - 11s 3ms/step - loss: 0.0361 - accuracy: 0.9885 - val_loss: 0.0432 - val_accuracy: 0.9866
Epoch 10/10
3750/3750 [==============================] - 11s 3ms/step - loss: 0.0354 - accuracy: 0.9890 - val_loss: 0.0357 - val_accuracy: 0.9895
```

Μέση επίδοση: 98.9%





Περνώντας στο κομμάτι του autoencoder, προσπαθώ να προ-εκπαιδεύσω το δίκτυο, εκπαιδεύοντας έναν autoencoder, έχοντας για encoder το δίκτυο μου.

Ο autoencoder αποτελείται από τον encoder και τον decoder. Στον encoder ρίχνεται η ανάλυση της εικόνας και ουσιαστικά η εικόνα συμπιέζεται, χρησιμοποιώντας συνελικτικά layers με μεγαλύτερο αριθμό φίλτρων κάθε φορά, ακολουθούμενα απο maxpooling layers τα οποία βοηθάν στο down sampling. Ο encoder δηλαδή αποτελεί κωδικοποίηση της εικόνας, μικρότερων διαστάσεων. Ο decoder, προσπαθεί να επανακτήσει την εικόνα χρησιμοποιώντας σαν είσοδο μόνο την αναπαράσταση του encoder.

Ο autoencoder μπορεί να θεωρηθεί ένα είδος μη γραμμικής γενίκευσης του PCA.

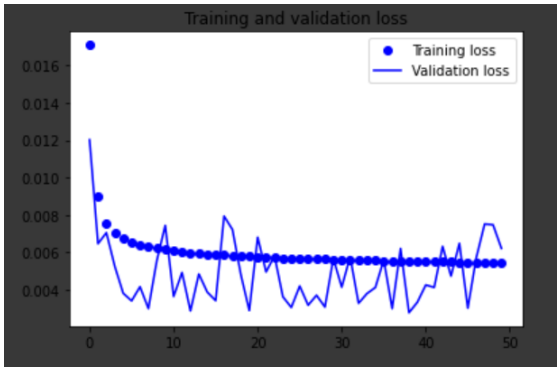
Αναλύοντας λίγο περισσότερο τον autoencoder, στον encoder υπάρχουν 3 convolutional layers, από τα οποία τα δύο πρώτα ακολοθούνται από ένα dropout, ένα batch normalization, και ένα max pooling layer. Τα φίλτρα σε κάθε συνελικτικό layer είναι 32, 64 και 128 αντίστοιχα, πράγμα που σημαίνει ότι στην έξοδο του encoder η εικόνα μας έχει διαστάσεις 7x7x128.

Ανίστοιχα στον decoder, όπου προσπαθούμε να ανεβάσουμε την ανάλυση της εικόνας υπάρχουν τα ίδια στρώματα, με την διαφορά ότι τα φίλτρα στα συνελικτικά είναι 64, 32, και τέλος 1, ενώ αντί για max pooling χρησιμοποιείται up sampling.

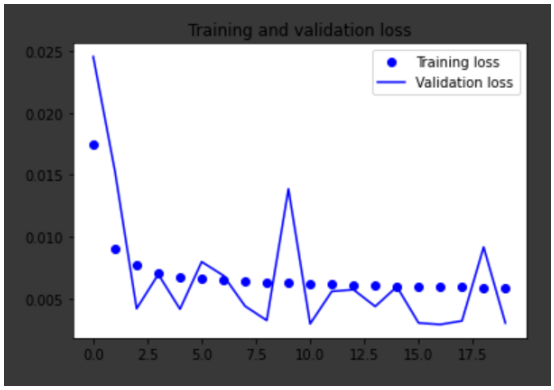
Το χαρακτηριστικό του autoencoder είναι πως στην εκπαίδευση δεν χρειάζεται τις ετικέτες του σετ, αλλά βασίζεται μόνο στο στις εικόνες.

Αφού λοιπόν εκπαιδευτεί ο autoencoder, χτίζω το classifier δίκτυο, κρατώντας τα βάρη του εκπαιδευμένου encoder. Εκτυπώνω τα βάρη του encoder από τον autoencoder ώστε να σιγουρέψω πως είναι τα ίδια με αυτά που φόρτωσα στον classifier. Θα δοκιμάσω δύο πράγματα. Αρχικά, θα περάσω τον encoder από 2 dense (πλήρως συνδεδεμένα) layers, και θα εκπαιδεύσω αυτό το δίκτυο όλο μαζί, εκπαιδεύοντας τον encoder ξανά. Το δεύτερο πράγμα που θα δοκιμάσω είναι να εκπαιδεύσω μόνο τα dense layers, κρατώντας τα βάρη του encoder.

Στον autoencoder, αρχικά χρησιμοποίησα 50 epochs με batch size 64. Ωστόσο παρατηρώ πως δεν χρειάζονται τόσα πολλά epochs αφού από το 20^{ο} και μετά το loss της εκπαίδευσης μένει σταθερό.



Αλλάζω λοιπόν τα epochs. Έχω δοκιμάσει πολλά διαφορετικά batch sizes ωστόσο επέλεξα να μείνω στα 64.



Εκτυπώνω κάποια ψηφία από το σετ του τεστ ώστε να τα συγκρίνω με αυτά που ξανά έχτισε ο autoencoder.

Βλέπουμε πως η ομοιότητα είναι μεγάλη οπότε ο autoencoder κάνει τη δουλειά του πολύ σωστά.

Έπειτα, εκτυπώνουμε τα βάρη, πρώτα του autoencoder και έπειτα του classifier, όπως αναφέρθηκε πριν ώστε να σιγουρευτούμε πως είναι τα ίδια.

```
array([[[ 0.13979228,  0.00241466, -0.11483953,  0.04132303,
         -0.14781803,  0.01146205, -0.13223667, -0.15630694,
         -0.2216639 ,  0.1542979 ,  0.04934234, -0.03900288,
          0.16051276,  0.266065  ,  0.1161904 ,  0.09086831,
         -0.32296368,  0.07836763,  0.1306458 ,  0.02582418,
         -0.02910794,  0.05294368,  0.17757918,  0.06251585,
          0.20578052,  0.3196018 ,  0.24397203,  0.12763537,
          0.1784079 , -0.05336333, -0.0551049 , -0.22452451]],

       [[ 0.20921437,  0.12486466, -0.26103777,  0.04291965,
         -0.05488269,  0.41765168, -0.24372075,  0.321273  ,
          0.16954312,  0.13645445, -0.20291781,  0.22395378,
         -0.17808956,  0.2830734 ,  0.09051228,  0.07913894,
          0.1238046 , -0.29357156,  0.22023968,  0.32303572,
         -0.2309487 ,  0.36000586,  0.09762176,  0.14407364,
          0.0765526 ,  0.37354705,  0.40153968, -0.04024968,
          0.10856169,  0.41470844, -0.20402828,  0.06827083]],

       [[-0.04696924,  0.23093106,  0.01168966,  0.07022101,
          0.09690997, -0.08670298,  0.01034522, -0.1263135 ,
          0.16260646, -0.03556449,  0.12143409,  0.09403189,
         -0.3356727 ,  0.22568643,  0.10160554, -0.265916  ,
          0.03322339, -0.13899511,  0.20277111, -0.02261603,
         -0.14852415, -0.00707453,  0.17431387,  0.09956068,
          0.15119076,  0.25238416,  0.09951743, -0.04005602,
```

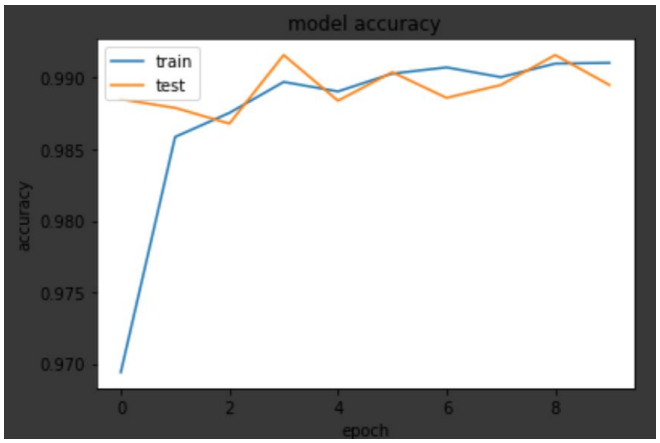
```
([[[ 0.13979228,  0.00241466, -0.11483953,  0.04132303,
    -0.14781803,  0.01146205, -0.13223667, -0.15630694,
    -0.2216639 ,  0.1542979 ,  0.04934234, -0.03900288,
     0.16051276,  0.266065  ,  0.1161904 ,  0.09086831,
    -0.32296368,  0.07836763,  0.1306458 ,  0.02582418,
    -0.02910794,  0.05294368,  0.17757918,  0.06251585,
     0.20578052,  0.3196018 ,  0.24397203,  0.12763537,
     0.1784079 , -0.05336333, -0.0551049 , -0.22452451]],

  [[ 0.20921437,  0.12486466, -0.26103777,  0.04291965,
    -0.05488269,  0.41765168, -0.24372075,  0.321273  ,
     0.16954312,  0.13645445, -0.20291781,  0.22395378,
    -0.17808956,  0.2830734 ,  0.09051228,  0.07913894,
     0.1238046 , -0.29357156,  0.22023968,  0.32303572,
    -0.2309487 ,  0.36000586,  0.09762176,  0.14407364,
     0.0765526 ,  0.37354705,  0.40153968, -0.04024968,
     0.10856169,  0.41470844, -0.20402828,  0.06827083]],

  [[-0.04696924,  0.23093106,  0.01168966,  0.07022101,
     0.09690997, -0.08670298,  0.01034522, -0.1263135 ,
     0.16260646, -0.03556449,  0.12143409,  0.09403189,
    -0.3356727 ,  0.22568643,  0.10160554, -0.265916  ,
     0.03322339, -0.13899511,  0.20277111, -0.02261603,
    -0.14852415, -0.00707453,  0.17431387,  0.09956068,
     0.15119076,  0.25238416,  0.09951743, -0.04005602,
    -0.29647848,  0.05475332, -0.00608599,  0.20629875]]],
```

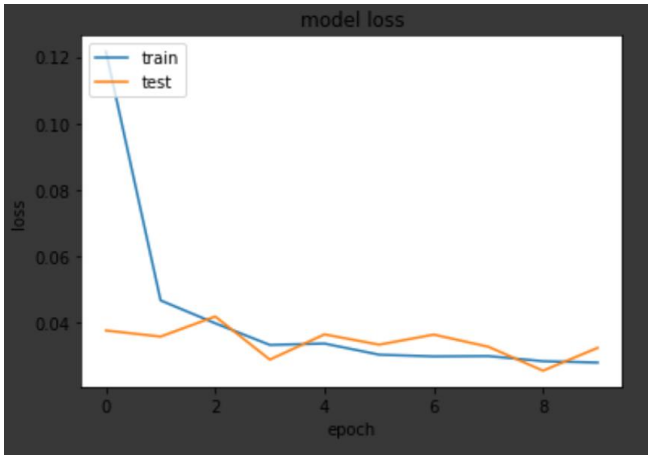
Εκπαιδεύουμε αρχικά όλο τον classifier, δηλαδή μαζί με τον encoder. Διαλέγω να χρησιμοποιήσω 10 epochs καθώς αντιλαμβάνομαι πως είναι αρκετά και σχετικά μικρό batch size, δηλαδή 16. Για optimizer χρησιμοποιείται ο ‘adam’ ενώ για συνάρτηση κόστους η categorical cross entropy η οποία είναι κατάλληλη για προβήματα ταξινόμησης πολλών κλάσεων. Τα αποτελέσματα:

```
Epoch 1/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.1216 - accuracy: 0.9694 - val_loss: 0.0378 - val_accuracy: 0.9885
Epoch 2/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0468 - accuracy: 0.9859 - val_loss: 0.0359 - val_accuracy: 0.9879
Epoch 3/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0399 - accuracy: 0.9876 - val_loss: 0.0420 - val_accuracy: 0.9868
Epoch 4/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0334 - accuracy: 0.9897 - val_loss: 0.0290 - val_accuracy: 0.9916
Epoch 5/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0338 - accuracy: 0.9890 - val_loss: 0.0366 - val_accuracy: 0.9884
Epoch 6/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0305 - accuracy: 0.9903 - val_loss: 0.0335 - val_accuracy: 0.9904
Epoch 7/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0300 - accuracy: 0.9907 - val_loss: 0.0365 - val_accuracy: 0.9886
Epoch 8/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0301 - accuracy: 0.9901 - val_loss: 0.0329 - val_accuracy: 0.9895
Epoch 9/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0285 - accuracy: 0.9910 - val_loss: 0.0257 - val_accuracy: 0.9916
Epoch 10/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0281 - accuracy: 0.9911 - val_loss: 0.0326 - val_accuracy: 0.9895
```

Βλέπουμε πως από την πρώτη κιόλας εποχή η ακρίβεια τόσο της εκπαίδευσης όσο και του τεστ ξεκινάει από πολύ καλό σημείο.

Οι αντίστοιχες γραφικές παραστάσεις, πρώτα της ακρίβειας και μετά του loss:





Κάτι που φαίνεται ίσως παράδοξο είναι πως σε ορισμένα σημεία το τεστ τα πηγαίνει καλύτερα από την εκπαίδευση. Αυτό όμως συμβαίνει λόγω του drop out επιπέδου που χρησιμοποιήθηκε στην εκπαίδευση, προς αποφυγή του overfitting.

Η μέση ακρίβεια που πήραμε είναι 98,95%.

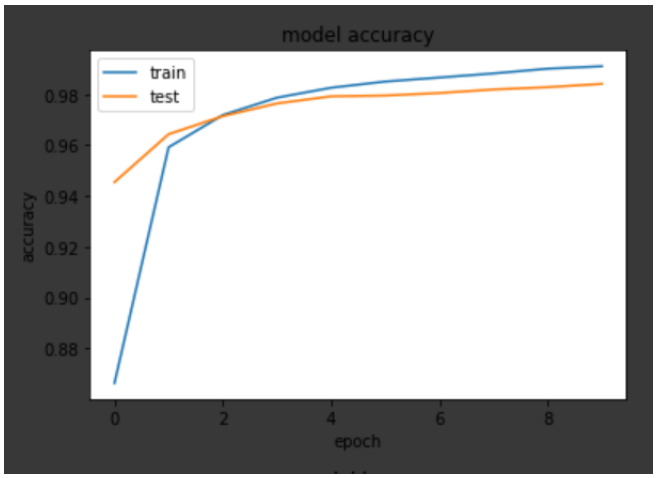
Έπειτα δοκιμάζω να άλλαξω το learning rate. Ενώ το default είναι της τάξης του 10^{-3}, το κάνω 10^{-5}.

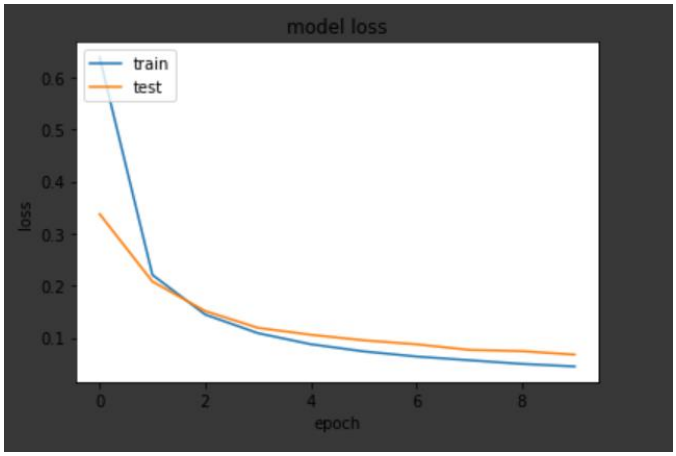
Τα αποτελέσματα είναι τα εξής:

```
Epoch 1/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.6377 - accuracy: 0.8663 - val_loss: 0.3373 - val_accuracy: 0.9454
Epoch 2/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.2205 - accuracy: 0.9592 - val_loss: 0.2078 - val_accuracy: 0.9643
Epoch 3/10
3750/3750 [==============================] - 26s 7ms/step - loss: 0.1445 - accuracy: 0.9718 - val_loss: 0.1513 - val_accuracy: 0.9714
Epoch 4/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.1087 - accuracy: 0.9787 - val_loss: 0.1190 - val_accuracy: 0.9764
Epoch 5/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0876 - accuracy: 0.9826 - val_loss: 0.1057 - val_accuracy: 0.9792
Epoch 6/10
3750/3750 [==============================] - 26s 7ms/step - loss: 0.0738 - accuracy: 0.9850 - val_loss: 0.0951 - val_accuracy: 0.9795
Epoch 7/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0640 - accuracy: 0.9866 - val_loss: 0.0875 - val_accuracy: 0.9805
Epoch 8/10
3750/3750 [==============================] - 26s 7ms/step - loss: 0.0569 - accuracy: 0.9882 - val_loss: 0.0768 - val_accuracy: 0.9819
Epoch 9/10
3750/3750 [==============================] - 26s 7ms/step - loss: 0.0498 - accuracy: 0.9901 - val_loss: 0.0743 - val_accuracy: 0.9828
Epoch 10/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0450 - accuracy: 0.9910 - val_loss: 0.0676 - val_accuracy: 0.9841
```

Είναι προφανές πως η ακρίβεια ξεκινάει απο πιο χαμηλή τιμή αυτή τη φορά, και η μέση τιμή της τελικά είναι 98,41%. Δεν υπαρχεί τεράστια διαφορά δηλαδή.

Οι γραφικές παραστάσεις:





Φαίνεται να έχουν εξομαλυνθεί σε σχέση με την προηγούμενη περίπτωση.

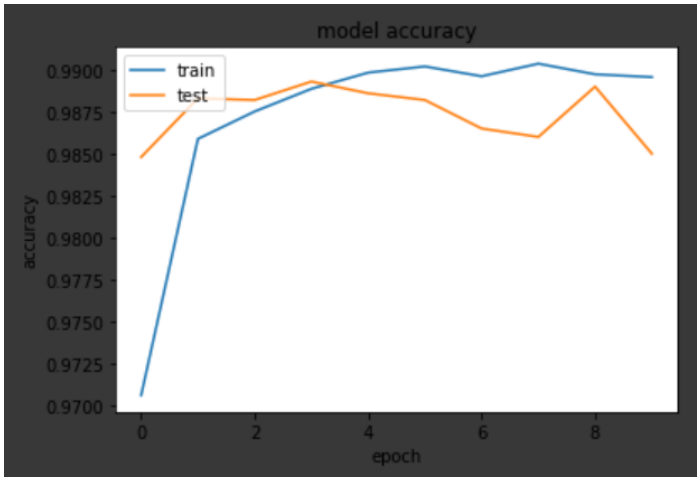
Έπειτα δοκιμάζω να εκπαιδεύσω μόνο τα dense layers, χωρίς τον encoder πάλι. Θα δοκιμάσω πρώτα με το default learning rate και έπειτα με το μειωμένο.

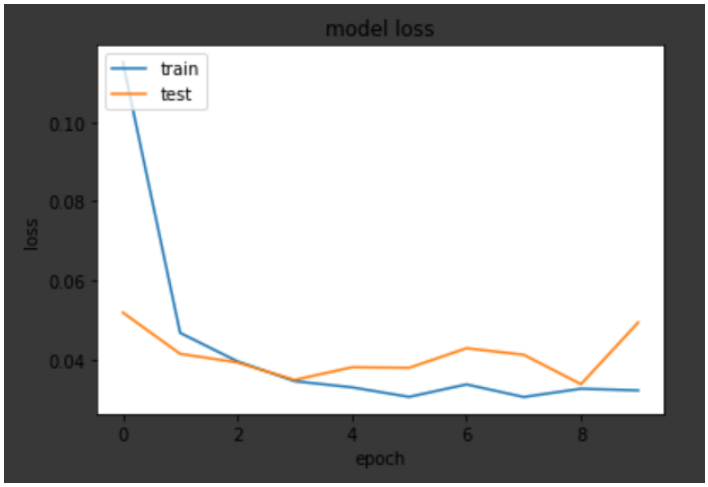
Τα αποτελέσματα:

```
Epoch 1/10
3750/3750 [==============================] - 22s 6ms/step - loss: 0.1152 - accuracy: 0.9706 - val_loss: 0.0519 - val_accuracy: 0.9848
Epoch 2/10
3750/3750 [==============================] - 26s 7ms/step - loss: 0.0468 - accuracy: 0.9859 - val_loss: 0.0415 - val_accuracy: 0.9883
Epoch 3/10
3750/3750 [==============================] - 26s 7ms/step - loss: 0.0396 - accuracy: 0.9875 - val_loss: 0.0393 - val_accuracy: 0.9882
Epoch 4/10
3750/3750 [==============================] - 26s 7ms/step - loss: 0.0346 - accuracy: 0.9889 - val_loss: 0.0349 - val_accuracy: 0.9893
Epoch 5/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0330 - accuracy: 0.9898 - val_loss: 0.0381 - val_accuracy: 0.9886
Epoch 6/10
3750/3750 [==============================] - 26s 7ms/step - loss: 0.0306 - accuracy: 0.9902 - val_loss: 0.0380 - val_accuracy: 0.9882
Epoch 7/10
3750/3750 [==============================] - 26s 7ms/step - loss: 0.0338 - accuracy: 0.9896 - val_loss: 0.0429 - val_accuracy: 0.9865
Epoch 8/10
3750/3750 [==============================] - 27s 7ms/step - loss: 0.0306 - accuracy: 0.9904 - val_loss: 0.0413 - val_accuracy: 0.9860
Epoch 9/10
3750/3750 [==============================] - 26s 7ms/step - loss: 0.0327 - accuracy: 0.9897 - val_loss: 0.0338 - val_accuracy: 0.9890
Epoch 10/10
3750/3750 [==============================] - 26s 7ms/step - loss: 0.0323 - accuracy: 0.9896 - val_loss: 0.0494 - val_accuracy: 0.9850
```

Η μέση ακρίβεια: 98,5%

Οι γραφικές παραστάσεις:



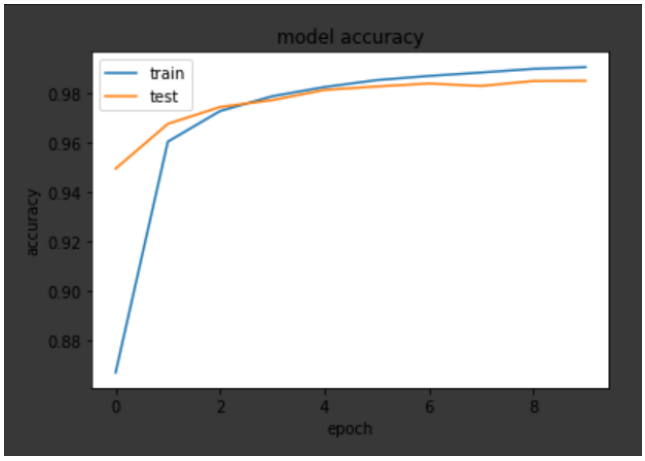


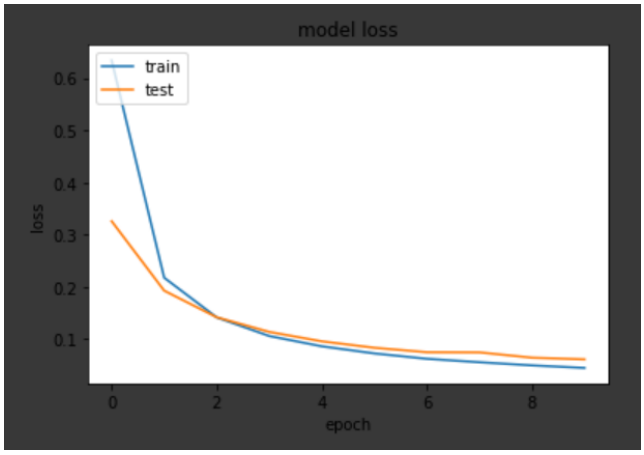
Τέλος, τα αποτελέσματα με μικρότερο learning rate:

```
Epoch 1/10
3750/3750 [==============================] - 24s 6ms/step - loss: 0.6338 - accuracy: 0.8669 - val_loss: 0.3259 - val_accuracy: 0.9493
Epoch 2/10
3750/3750 [==============================] - 25s 7ms/step - loss: 0.2176 - accuracy: 0.9602 - val_loss: 0.1928 - val_accuracy: 0.9674
Epoch 3/10
3750/3750 [==============================] - 25s 7ms/step - loss: 0.1415 - accuracy: 0.9726 - val_loss: 0.1418 - val_accuracy: 0.9742
Epoch 4/10
3750/3750 [==============================] - 25s 7ms/step - loss: 0.1059 - accuracy: 0.9786 - val_loss: 0.1138 - val_accuracy: 0.9770
Epoch 5/10
3750/3750 [==============================] - 25s 7ms/step - loss: 0.0862 - accuracy: 0.9823 - val_loss: 0.0957 - val_accuracy: 0.9811
Epoch 6/10
3750/3750 [==============================] - 25s 7ms/step - loss: 0.0725 - accuracy: 0.9851 - val_loss: 0.0834 - val_accuracy: 0.9825
Epoch 7/10
3750/3750 [==============================] - 25s 7ms/step - loss: 0.0623 - accuracy: 0.9868 - val_loss: 0.0749 - val_accuracy: 0.9837
Epoch 8/10
3750/3750 [==============================] - 25s 7ms/step - loss: 0.0557 - accuracy: 0.9881 - val_loss: 0.0745 - val_accuracy: 0.9827
Epoch 9/10
3750/3750 [==============================] - 25s 7ms/step - loss: 0.0497 - accuracy: 0.9896 - val_loss: 0.0644 - val_accuracy: 0.9847
Epoch 10/10
3750/3750 [==============================] - 25s 7ms/step - loss: 0.0448 - accuracy: 0.9903 - val_loss: 0.0612 - val_accuracy: 0.9848
```

Η μέση ακρίβεια: 98,48%

Οι γραφικές παραστάσεις:





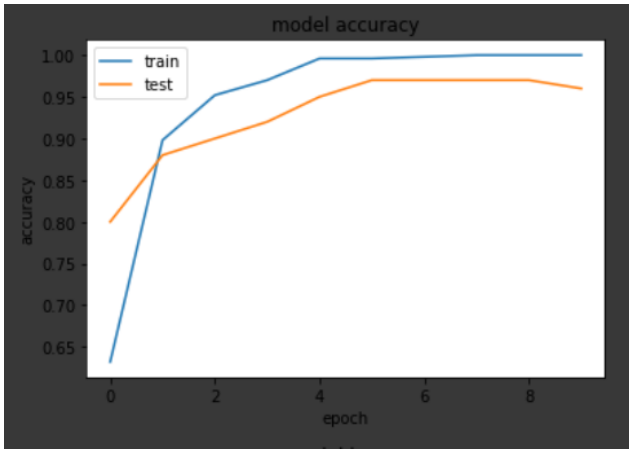
Παρατηρώ ξανά πως είναι περισσότερο εξομαλυμένες από αυτές με μεγαλύτερο learning rate.

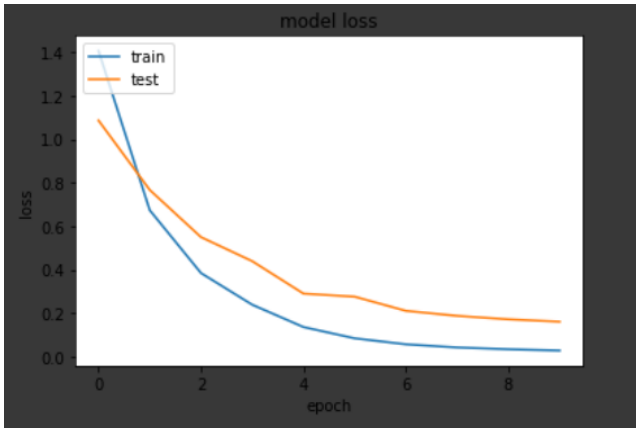
Τώρα θέλω να συγκρίνω τον προ-εκπαιδευμένο ταξινομητή με αυτόν που εκπαιδεύται μόνος. Αν και δεν φαίνεται κάποια ουσιαστική διαφορά όταν χρησιμοποιώ ολόκληρο το σετ δεδομένων, γιατί το τασκ είναι αρκετά εύκολο και οι εικονίτσες επεξεργασμένες κατάλληλα, με αποτέλεσμα το νευρωνικό και στις 2 περιπτώσεις να τα πηγαίνει εξίσου καλά. Οπότε, ωστέ να καταλάβω τη βοήθεια του autoencoder επιλέγω να χρησιμοποιήσω πολύ λιγα δεδομένα σαν είσοδο στον ταξινομητή, δηλαδή μόνο 20 από κάθε κλάση και όλα τα υπόλοιπα να τα περάσω σαν είσοδο στον autoencoder. Δηλαδή να εκπαιδεύσω τον ταξινομητή με δεδομένα που δεν έχει δει ο autoencoder.

Αφού λοιπόν εκπαιδεύσω τον autoencoder (χωρίς αλλαγές σε σχέση με το προηγούμενο μοντέλο) με τα υπόλοιπα δεδομένα, σαν είσοδο περνώ στον classifier μόνο 20 δεδομένα από κάθε κλάση και αντίστοιχα για τεστ χρησιμοποιώ τα πρώτα 100 δεδομένα του τεστ σετ, επιλέγω να εκπαιδεύσω μόνο τα dense επίπεδα για να έχω πιο καθαρή εικόνα της δουλειάς που έκανε ο autoencoder. Όσον αφορά τις εποχές, τα batches την συνάρτηση κόστους και τον βελτιστοποιητή, χρησιμοποιώ τα ίδια με τα πειράματα λίγο πιο πάνω. Τα αποτελέσματα που παίρνω είναι τα εξής:

```
Epoch 1/10
32/32 [==============================] - 0s 11ms/step - loss: 1.4077 - accuracy: 0.6320 - val_loss: 1.0868 - val_accuracy: 0.8000
Epoch 2/10
32/32 [==============================] - 0s 5ms/step - loss: 0.6734 - accuracy: 0.8980 - val_loss: 0.7667 - val_accuracy: 0.8800
Epoch 3/10
32/32 [==============================] - 0s 4ms/step - loss: 0.3839 - accuracy: 0.9520 - val_loss: 0.5494 - val_accuracy: 0.9000
Epoch 4/10
32/32 [==============================] - 0s 5ms/step - loss: 0.2381 - accuracy: 0.9700 - val_loss: 0.4386 - val_accuracy: 0.9200
Epoch 5/10
32/32 [==============================] - 0s 4ms/step - loss: 0.1352 - accuracy: 0.9960 - val_loss: 0.2892 - val_accuracy: 0.9500
Epoch 6/10
32/32 [==============================] - 0s 5ms/step - loss: 0.0832 - accuracy: 0.9960 - val_loss: 0.2751 - val_accuracy: 0.9700
Epoch 7/10
32/32 [==============================] - 0s 4ms/step - loss: 0.0557 - accuracy: 0.9980 - val_loss: 0.2098 - val_accuracy: 0.9700
Epoch 8/10
32/32 [==============================] - 0s 5ms/step - loss: 0.0411 - accuracy: 1.0000 - val_loss: 0.1867 - val_accuracy: 0.9700
Epoch 9/10
32/32 [==============================] - 0s 5ms/step - loss: 0.0330 - accuracy: 1.0000 - val_loss: 0.1709 - val_accuracy: 0.9700
Epoch 10/10
32/32 [==============================] - 0s 4ms/step - loss: 0.0269 - accuracy: 1.0000 - val_loss: 0.1599 - val_accuracy: 0.9600
```

Το δίκτυο τα πηγαίνει περίφημα παρόλο που έχει σαν είσοδο τόσο μικρό dataset! Αν και ξεκινάει απο πιο χαμηλή επίδοση και αντίστοιχα πιο υψηλή απώλεια, στο τέλος πετυχαίνει 100% στην εκπαίδευση και μέση ακρίβεια του τεστ 91.9%.

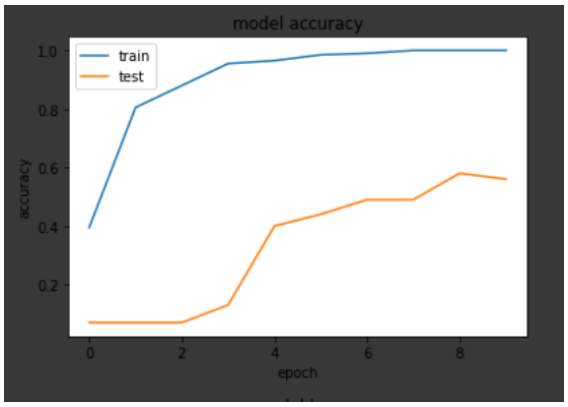


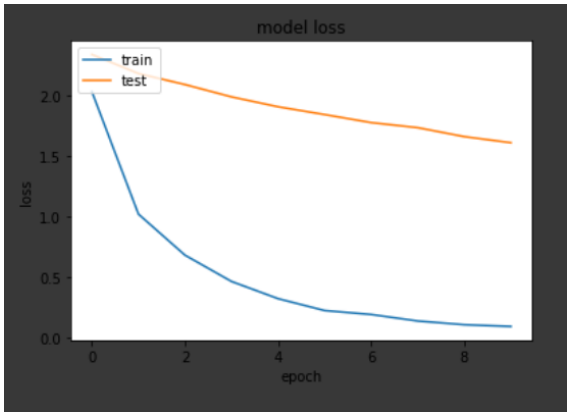


Έχοντας πάλι ακριβώς ίδιο δίκτυο με αυτό του encoder για ταξινομητή, χωρίς να έχω προ εκπαιδεύσει το δίκτυο, δοκιμάζω να περάσω σαν είσοδο πάλι 20 δεδομένα από κάθε κλάση σε αυτόν.

```
Epoch 1/10
13/13 [==============================] - 0s 32ms/step - loss: 2.0286 - accuracy: 0.3950 - val_loss: 2.3365 - val_accuracy: 0.0700
Epoch 2/10
13/13 [==============================] - 0s 6ms/step - loss: 1.0166 - accuracy: 0.8050 - val_loss: 2.1788 - val_accuracy: 0.0700
Epoch 3/10
13/13 [==============================] - 0s 5ms/step - loss: 0.6773 - accuracy: 0.8800 - val_loss: 2.0878 - val_accuracy: 0.0700
Epoch 4/10
13/13 [==============================] - 0s 5ms/step - loss: 0.4600 - accuracy: 0.9550 - val_loss: 1.9859 - val_accuracy: 0.1300
Epoch 5/10
13/13 [==============================] - 0s 5ms/step - loss: 0.3179 - accuracy: 0.9650 - val_loss: 1.9042 - val_accuracy: 0.4000
Epoch 6/10
13/13 [==============================] - 0s 5ms/step - loss: 0.2195 - accuracy: 0.9850 - val_loss: 1.8396 - val_accuracy: 0.4400
Epoch 7/10
13/13 [==============================] - 0s 5ms/step - loss: 0.1870 - accuracy: 0.9900 - val_loss: 1.7732 - val_accuracy: 0.4900
Epoch 8/10
13/13 [==============================] - 0s 5ms/step - loss: 0.1331 - accuracy: 1.0000 - val_loss: 1.7313 - val_accuracy: 0.4900
Epoch 9/10
13/13 [==============================] - 0s 5ms/step - loss: 0.1025 - accuracy: 1.0000 - val_loss: 1.6573 - val_accuracy: 0.5800
Epoch 10/10
13/13 [==============================] - 0s 5ms/step - loss: 0.0882 - accuracy: 1.0000 - val_loss: 1.6071 - val_accuracy: 0.5600
```

Παρατηρώ πως ενώ και πάλι το δίκτυο εκπαιδεύεται, καθώς στην επίδοση της εκπαίδευσης πετυχαίνει πολύ καλά ποσοστά, στο τεστ δεν τα πηγαίνει τόσο καλά όσο πριν. Η μέση επίδοση είναι 49%.





Έτσι λοιπόν καταλαβαίνουμε πως ο autoencoder διαδραμάτισε σημαντικό ρόλο στην εκπαίδευση του ταξινομητή, ενώ με τη χρήση ολόκληρου του σετ δεδομένων αυτό δεν ήταν προφανές καθώς και στις δύο περιπτώσεις οι επιδόσεις ήταν πολύ υψηλές.

Όπως γίνεται αντιληπτό, τα νευρωνικά μοντέλα παρουσιάζουν καλύτερες επιδόσεις από τους δύο αλγόριθμους ταξινόμησης που παρουσιάστηκαν καθώς οι περισσότερες είναι της τάξης του 99%, ενώ των κατηγοροποιητών της τάξης του 90%. Βέβαια και οι δύο αυτές επιδόσεις είναι αρκετά ικανοποιητικές.

Επισυνάπτοντας, αναφορικά με τους χρόνους εκτέλεσης υπάρχουν διαφορετικές μετρήσεις για τους αλγορίθμους και τα νευρωνικά που υλοποιήθηκαν. Για τους αλγορίθμους των πλησιέστερων γειτόνων και πλησιέστρου κέντρου, στην περίπτωση που χρησιμοποιείται όλο το σετ δεδομένων χωρίς μείωση διαστάσεων, ο χρόνος εκτέλεσης είναι 1 ώρα και 46

λεπτά. Αν χρησιμοποιηθεί όλο το σετ αλλά με μειωμένες διαστάσεις, όπως συζητηθήκε επάνω, ο χρόνος είναι 1 ώρα και 20 λεπτά. Τέλος όταν η είσοδος είναι επιλεγμένα δεδομένα από το data set ο χρόνος είναι 40 λεπτά.

Όσον αφορά τα νευρωνικά δίκτυα, για περισσότερη ταχύτητα εκτελέστηκαν σε χρόνο GPU και για κάθε μοντέλο και περίπτωση οι χρόνοι φαίνονται στα στιγμότυπα πιο πάνω και είναι φυσικά κατά πολύ μικρότεροι από αυτούς των αλγορίθμων ταξινόμησης. Δηλαδή λίγα δευτερόλεπτα για κάθε εποχή. Ωστόσο δοκιμάζοντας να τρέξουμε ένα μοντέλο σε χρόνο CPU, τα αποτελέσματα δείχνουν πως ο χρόνος είναι της τάξης των 3 λεπτών ανά εποχή. Ανάλογα λοιπόν τον αριθμό των εποχών και των batches του μοντέλου υπολογίζεται ο χρόνος που χρειάζεται όλο το μοντέλο να εκπαιδευτεί, που στην περίπτωση μας είναι μισή ώρα, άρα λίγο πιο γρήγορα από τους αλγορίθμους.

Οι κώδικες έχουν γραφτεί σε notebook του Google Colab, οπότε στέλνω τα αρχεία .ipynb, αλλά και τα λινκ που είναι:

Knn-Nc: https://colab.research.google.com/drive/1PbZ7A2rBNIhzmhbv9pgsw2txqJE4G7_e

Classifier: https://colab.research.google.com/drive/1V4LYrH3VAJ-iFBCxmzWm-3CNywvzB7Xd

Autoencoder: https://colab.research.google.com/drive/1t8j82Z8gOo0-1OAJHbmiU24bu7Rbr7NS

Κώστογλου Σοφία 3114