Έκθεση πρώτης εργασίας Νευρωνικών Δικτύων - Βαθιάς Μάθησης

Ιωάννης Οικονομίδης

25 Νοεμβρίου 2022

Περιεχόμενα

1 Εισαγωγή

1.1 Γενικά

Ονομάζομαι Ιωάννης Οικονομίδης. Στην πρώτη εργασία επέλεξα να αναλύσω το Mnist dataset. Στο συμπιεσμένο αρχείο που ανέβασα στο e-Learning υπάρχει αυτό το pdf αρχείο που είναι η έκθεσή μου γραμμένη σε Latex και το python script main.py που είναι το python πρόγραμμα που διαβάζει τα δεδομένα εκπαίδευσης και ελέγχου, τρέχει τον κώδικα της ενδιάμεσης εργασίας και εκπαιδεύει και αξιολογεί το νευρωνικό δίκτυό μου που περιέχεται στο κώδικα.

Όσον αφορά το νευρωνικό δίκτυο, υπάρχουν δύο νευρωνικά στο πρόγραμμα. Ένα έτοιμο νευρωνικό δίκτυο βασισμένο στο keras framework που χρησιμοποιούσα για να ελέγχω τα αποτελέσματά μου και χρησιμοποιεί το κώδικα που μας έδειξε ο κύριος Πασσαλής. Στο πρόγραμμα υπάρχει συνάρτηση που τρέχει αυτό το δίκτυο, αλλά την έχω βάλει σε σχόλιο ώστε να μην τρέχει γιατί δεν χρειάζεται με βάση αυτά που ζητάει η εκφώνηση. Συγκεκριμένα, είναι αυτή η γραμμή:

```
# keras_nn(X_train, Y_train, X_test, Y_test)
```

Επιπλέον, υπάρχει το δικό μου νευρωνικό δίκτυο που έφτιαξα και το οποίο θα αναλύσω και χρησιμοποιήσω στη συνέχεια που τρέχει με αυτή την γραμμή:

```
nn = custom_neural_network(X_train, Y_train, X_test, Y_test)
```

2 Ανάλυση κώδικα

2.1 Σημειώσεις

Το κομμάτι της ανάλυσης κώδικα της έκθεσης θα συγκεντρωθεί στην εξήγηση του τι κάνει η κάθε συνάρτηση και κλάση του προγράμματος, χωρίς απαραίτητα να αναλύεται τι κάνει η κάθε γραμμή γιατί αυτό θα έπαιρνε πάρα πολλές σελίδες. Ωστόσο, στο κώδικα του προγράμματος υπάρχουν σχόλια στα αγγλικά πριν από σχεδόν κάθε γραμμή που εξηγούν τι κάνει η αντίστοιχη γραμμή. Για παραπάνω εξηγήσεις από ότι δίνει η έκθεση μπορείτε να δείτε τα σχόλια αυτά.

2.2 Ανάλυση κώδικα συναρτήσεων

2.2.1 read_flatten_mnist

Η συνάρτηση αυτή φορτώνει το Mnist dataset από το keras framework, κάνει flatten τα δείγματα εκπαίδευσης και ελέγχου από δισδιάστατους πίνακες 28 * 28 σε μονοδιάστατους πίνακες 784 στοιχείων και επιστρέφει τα δείγματα εκπαίδευσης και ελέγχου καθώς και τις αντίστοιχες ετικέτες τους.

2.2.2 ncc και knn

Αυτές οι συναρτήσεις αφορούν την ενδιάμεση πρώτη εργασία και είναι ίδιες με αυτές που παρέδωσα στο ενδιάμεσο κομμάτι της εργασίας. Και οι δύο δέχονται ως είσοδο

τα δεδομένα και τις ετικέτες εκπαίδευσης (x_train, y_train) και τα δεδομένα και τις ετικέτες ελέγχου (x_test, y_test). Η knn δέχεται επίσης ως είσοδο την παράμετρο k που είναι ο αριθμός των γειτόνων του κατηγοριοποιητή k κοντινότερων γειτόνων.

Και στις δύο συναρτήσεις, στις πρώτες δύο γραμμές τους αρχικοποιείται και εκπαιδεύεται ο αντίστοιχος κατηγοριοποιητής που δίνεται από την βιβλιοθήκη της python sklearn με τα δεδομένα και τις ετικέτες εκπαίδευσης x_train και y_train αντίστοιχα. Στις επόμενες δύο γραμμές υπολογίζεται η απόδοση του αντίστοιχου κατηγοριοποιητή πρώτα για τα δεδομένα εκπαίδευσης και μετά για τα δεδομένα ελέγχου. Τέλος, επιστρέφεται η ακρίβεια του αντίστοιχου κατηγοριοποιητή για τα δεδομένα εκπαίδευσης και ελέγχου που υπολογίστηκε νωρίτερα.

2.2.3 calculate_accuracy

Παίρνει ως είσοδο την έξοδο του μοντέλου σε μορφή πίνακα πιθανοτήτων για την κάθε κλάση και τις πραγματικές ετικέτες και επιστρέφει την μέση ακρίβεια.

2.2.4 find_wrong_prediction

Παίρνει ως είσοδο ένα αντικείμενο της κλάσης NeuralNetwork που αντιπροσωπεύει ολόκληρο το νευρωνικό δίκτυο, τα δείγματα, τις αντίστοιχες ετικέτες και αν πρέπει να εμφανίζονται τα αποτελέσματα για κάθε δείγμα που ελέγχεται. Ελέγχει κάθε δείγμα για το αν η πρόβλεψη του μοντέλου είναι ίδια με την ετικέτα. Αν είναι ίδια, συνεχίζει στο επόμενο δείγμα, αλλιώς εμφανίζει τα αποτελέσματα για το δείγμα και τις πιθανότητες που έδωσε ως έξοδο το μοντέλο και επιστρέφει το δείκτη του δείγματος.

2.2.5 intermediate_project

Αυτή η συνάρτηση τρέχει τον κώδικα που υπήρχε στο ενδιάμεσο κομμάτι της πρώτης εργασίας. Παίρνει ως είσοδο τα δεδομένα εκπαίδευσης και ελέγχου, τρέχει τους τρεις κατηγοριοποιητές και εμφανίζει τα αποτελέσματά τους ακριβώς όπως γινόταν στην ενδιάμεση εργασία.

Πρώτα χρησιμοποιείται η συνάρτηση ncc που αναφέρθηκε παραπάνω ώστε να εκπαιδευτεί ο κατηγοριοποιητής πλησιέστερου κέντρου κλάσης και να υπολογιστεί η απόδοσή του για το Mnist dataset. Η απόδοσή του για τα δεδομένα εκπαίδευσης και ελέγχου τυπώνεται στην οθόνη και ως δεκαδικός αριθμός και ως ποσοστό με δύο δεκαδικά ψηφία ακρίβειας για το ποσοστό. Επιπλεόν, χρονομετράται σε δευτερόλεπτα ο χρόνος που απαιτεί η εκπαίδευση και ο υπολογισμός της απόδοσης του κατηγοριοποιητή και τυπώνεται στην οθόνη και αυτή η μέτρηση.

Για κάθε τιμή του k που απαιτείται, οι οποίες σε αυτή την περίπτωση είναι οι τιμές 1 και 3 με βάση την εκφώνηση, χρησιμοποιείται η συνάρτηση knn που αναφέρθηκε παραπάνω ώστε να εκπαιδευτεί ο κατηγοριοποιητής πλησιέστερου γείτονα και να υπολογιστεί η απόδοσή του για το Mnist dataset. Η απόδοσή του για τα δεδομένα εκπαίδευσης και ελέγχου τυπώνεται στην οθόνη και ως δεκαδικός αριθμός και ως ποσοστό με δύο δεκαδικά ψηφία ακρίβειας για το ποσοστό. Επιπλεόν,

χρονομετράται σε δευτερόλεπτα ο χρόνος που απαιτεί η εκπαίδευση και ο υπολογισμός της απόδοσης του κατηγοριοποιητή και τυπώνεται στην οθόνη και αυτή η μέτρηση, όπως έγινε και για τον κατηγοριοποιητή πλησιέστερου κέντρου κλάσης.

2.2.6 keras_nn

Αυτή η συνάρτηση δημιουργεί και εκπαιδεύει ένα νευρωνικό δίκτυο χρησιμοποιώντας το keras framework. Επιπλέον αξιολογεί τα δεδομένα εκπαίδευσης και ελέγχου. Ο κώδικάς της είναι κυρίως παρμένος από αυτό το github repo `https://github.com/passalis/keras_meetup/blob/master/1_Getting_Started.ipynb` με το οποίο μας δίδαγε ο κύριος Πασσαλής οπότε δεν χρειάζεται να εξηγηθεί περαιτέρω.

2.2.7 custom_neural_network

Αυτή η συνάρτηση δημιουργεί και εκπαιδεύει ένα νευρωνικό δίκτυο χρησιμοποιώντας τη κλάση NeuralNetwork του προγράμματος και επιστρέφει το δίκτυο αυτό. Επιπλέον εμφανίζει το χρόνο που διήρκησε η εκπαίδευση του δικτύου (χωρίς τον χρόνο αξιολόγησης όλων των δειγμάτων στο τέλος κάθε εποχής) και αξιολογεί τα δεδομένα εκπαίδευσης και ελέγχου. Το δίκτυο που δημιουργεί έχει σχεδόν τα ίδια χαρακτηριστικά και μορφή με το νευρωνικό δίκτυο της προηγούμενης συνάρτησης που χρησιμοποιεί το keras. Τα χαρακτηριστικά και η μορφή του δικτύου θα αναλυθούν περαιτέρω σε επόμενο κομμάτι της έκθεσης.

2.3 Ανάλυση κώδικα κλάσεων

2.3.1 Layer

Αυτή η κλάση υλοποιεί ένα στρώμα του νευρωνικού δικτύου. Περιέχει ένα κατασκευαστή και τις μεθόδους forward_pass και back_propagate. Είναι κλάση που πρέπει να κληρονομείται από τα συγκεκριμένα είδη στρωμάτων και όχι που πρέπει να χρησιμοποιείται άμεσα.

Στον κατασκευαστή αρχικοποιούνται όλα τα μέλη της κλάσης. Τα biases αρχικοποιούνται στο 0 ενώ τα βάρη αρχικοποιούνται με τυχαίες τιμές που πολλαπλασιάζονται με το 0.01 ώστε να έχουν μικρές τιμές για να μην βγουν σε περιοχή όπου δεν θα αλλάζει η τιμή τους μετά την αρχικοποίηση. Επίσης αρχικοποιούνται στο None οι είσοδοι και οι έξοδοι του στρώματος αλλά και τα διανύσματα κλίσεων ως προς τις εισόδους, τα biases και τα βάρη. Τέλος, αρχικοποιείται και το momentum κάθε bias και βάρους στο μηδέν ώστε να χρησιμοποιηθεί από τον optimizer του δικτύου. Όλα τα μέλη της κλάσης βρίσκονται σε μορφή πινάκων ώστε να γίνεται γρήγορα η επεξεργασία τους.

Η μέθοδοι forward_pass και back_propagate είναι κενές γιατί παρέχονται από την κλάση που κληρονομεί την κλάση αυτή.

2.3.2 DenseLayer

Αυτή η κλάση υλοποιεί ένα πυκνό στρώμα του νευρωνικού δικτύου. Κληρονομεί την κλάση Layer που αναλύθηκε νωρίτερα. Περιέχει τις ίδιες μεθόδους με την κλάση που κληρονομεί. Στον κατασκευαστή καλείται ο κατασκευαστής της κλάσης Layer την οποία κληρονομεί.

Η μέθοδος forward_pass κάνει ένα forward pass δια μέσω του στρώματος παίρνοντας ως είσοδο τις εισόδος του στρώματος, αποθηκεύοντάς τες για τη χρήση τους στο back propagation και υπολογίζοντας και αποθηκεύοντας την έξοδο του στρώματος με τη χρήση του εσωτερικού γινομένου της numpy.

Η μέθοδος back_propagate κάνει back propagation δια μέσω του στρώματος παίρνοντας ως είσοδο τα διανύσματα κλίσεων ως προς τις εισόδους που επιστρέφει η συνάρτηση ενεργοποίησης του στρώματος. Υπολογίζει τα διανύσματα κλίσεων ως προς τις εισόδους του στρώματος, τα biases και τα βάρη. Κατά τους υπολογισμούς, ο πίνακας των βαρών που χρησιμοποιείται για τον υπολογισμό των διανυσμάτων κλίσεων ως προς τις εισόδους και ο πίνακας των εισόδων που χρησιμοποιείται γαι τον υπολογισμό των διανυσμάτων κλίσεων ως προς τα βάρη πρώτα αναστρέφονται πριν χρησιμοποιηθούν στους υπολογισμούς ώστε να ταιριάζουν οι διαστάσεις των πινάκων κατά το εσωτερικό γινόμενο.

2.3.3 ActivationFunction

Αυτή η κλάση υλοποιεί μια συνάρτηση ενοργοποίησης. Περιέχει ένα κατασκευαστή και τις μεθόδους forward_pass και back_propagate. Είναι κλάση που πρέπει να κληρονομείται από τα συγκεκριμένα είδη συναρτήσεων ενεργοποίησης και όχι που πρέπει να χρησιμοποιείται άμεσα.

Στον κατασκευαστή αρχικοποιούνται όλα τα μέλη της κλάσης. Αρχικοποιούνται στο None οι είσοδοι και οι έξοδοι της συνάρτησης καθώς και τα διανύσματα κλίσεων ως προς τις εισόδους.

Η μέθοδοι forward_pass και back_propagate είναι κενές γιατί παρέχονται από την κλάση που κληρονομεί την κλάση αυτή.

2.3.4 ReLU

Αυτή η κλάση υλοποιεί τη συνάρτηση ενεργοποίησης ReLU. Περιέχει ένα κατασκευαστή και τις μεθόδους forward_pass και back_propagate. Στον κατασκευαστή καλείται ο κατασκευαστής της κλάσης ActivationFunction την οποία κληρονομεί.

Η μέθοδος forward_pass κάνει ένα forward pass δια μέσω της συνάρτησης ενεργοποίησης ReLUκαι παίρνει ως είσοδο τις εισόδους της συνάρτησης ενεργοποίησης, δηλαδή τις εξόδους του στρώματος στο οποίο αντιστοιχεί, αποθηκεύοντάς τες για τη χρήση τους στο back propagation και υπολογίζοντας και αποθηκεύοντας την έξοδο της συνάρτησης ενεργοποίησης με βάση τον τύπο της συνάρτησης ReLU. Συγκεκριμένα, αν η είσοδος είναι μιρότερη από το μηδέν, η έξοδος εγίνεται μηδέν, αλλιώς είναι η ίδια η είσοδος.

Η μέθοδος back_propagate κάνει back propagation δια μέσω της συνάρτησης ενεργοποίησης παίρνοντας ως είσοδο τα διανύσματα κλίσεων ως προς τις εισόδους που επιστρέφει το επόμενο στρώμα του μοντέλου. Υπολογίζει τα διανύσματα

κλίσεων ως προς τις εισόδους της συνάρτησης ενεργοποίησης δημιουργόντας ένα αντίγραφο των διανυσμάτων κλίσεων ως προς τις εισόδους που πήρε ως είσοδο και μηδενίζοντας κάθε τιμή της οποίας η αντίστοιχη είσοδος είναι μικρότερη ή ίση του μηδενός.

2.3.5 SoftMax

Αυτή η κλάση υλοποιεί τη συνάρτηση ενοργοποίησης SoftMax. Περιέχει ένα κατασκευαστή και τις μεθόδους forward_pass και back_propagate. Στον κατασκευαστή καλείται ο κατασκευαστής της κλάσης ActivationFunction την οποία κληρονομεί.

Η μέθοδος forward_pass κάνει ένα forward pass δια μέσω της συνάρτησης ενεργοποίησης SoftMax και παίρνει ως είσοδο τις εισόδους της συνάρτησης ενεργοποίησης, δηλαδή τις εξόδους του στρώματος στο οποίο αντιστοιχεί, αποθηκεύοντάς τες για τη χρήση τους στο back propagation και υπολογίζοντας και αποθηκεύοντας την έξοδο της συνάρτησης ενεργοποίησης με βάση τον τύπο της συνάρτησης SoftMax. Συγκεκριμένα, χρησιμοποιεί τον γνωστό τύπο της συνάρτησης SoftMax, αλλά από όλες τις εισόδους αφαιρεί πρώτα τη μεγαλύτερη είσοδο ώστε όταν ανεβούν στον εκθέτη οι είσοδοι, να μη βγαίνουν υπερβολικά μεγάλες τιμές που θα ξεπαιρνούν το όριο των αριθμών που μπορεί να χειριστεί ο υπολογιστής.

Η μέθοδος back_propagate κάνει back propagation δια μέσω της συνάρτησης ενεργοποίησης παίρνοντας ως είσοδο τα διανύσματα κλίσεων ως προς τις εισόδους που επιστρέφει το επόμενο στρώμα του μοντέλου. Υπολογίζει τα διανύσματα κλίσεων ως προς τις εισόδους της συνάρτησης ενεργοποίησης κάνοντας χρήση πινάκων Τζακόμπι. Δηλαδή, για κάθε έξοδο της συνάρτησης SoftMax και για το κάθε αντίστοιχο διάνυσμα κλίσεων ως προς τις εισόδους που πήρε ως είσοδο, υπολογίζει τον αντίστοιχο πίνακα Τζακόμπι και υπολογίζει το εσωτερικό γινόμενό του με το αντίστοιχο διάνυσμα κλίσεων ως προς τις εισόδους που πήρε ως είσοδο. Αυτό αποθηκεύεται ως το αντίστοιχο διάνυσμα κλίσων ως προς τις εισόδους της συνάρτησης SoftMax για την συγκεκριμένη έξοδο. Οταν αυτό γίνει για την κάθε έξοδο της συνάρτησης SoftMax, αυτά είναι τα διανύσματα κλίσεων ως προς τις εισόδους της συνάρτησης SoftMax.

2.3.6 SgdOptimizer

Αυτή η κλάση υλοποιεί τον Stochastic Gradient Descent optimizer. Περιέχει ένα κατασκευαστή και τις μεθόδους update_learning_rate, update_layer_parameters και increment_iteration_counter.

Στον κατασκευαστή αρχικοποιούνται όλες οι παράμετροι του optimizer. Συγκεκριμένα, το learning rate, το momentum και το decay παίρνουν ως τιμές αυτές που δίνονται ως παράμετροι στον κατασκευαστή. Οι τιμές τους αν δεν δοθούν ως παράμετροι στον κατασκευαστή είναι 0.01 για το learning rate, 0.0 για το momentum και 0.0 για το decay, καθώς αυτές είναι οι τιμές που χρησιμοποιεί ως default για το Stochastic Gradient Descent του keras framework. Επίσης το τωρινό learning rate αρχικοποιείται στη τιμή του αρχικού learning rate ενώ ο μετρητής των επαναλήψεων αρχικοποιείται στο μηδέν.

Η μέθοδος update_learning_rate ενημερώνει το τωρινό learning rate αν το decay είναι διαφορετικό του μηδενός. Συγκεκριμένα, αν το decay δεν είναι μηδέν, το τωρινό learning rate υπολογίζεται από το αρχικό learning rate, το ρυθμό decay και το μετρητή των επαναλήψεων με βάση το τύπο του decay στον Stochastic Gradient Descent optimizer. Αυτή η μέθοδος πρέπει να καλείται πριν από κάθε ενημέρωση που κάνει ο optimizer στις παραμέτρους των στρωμάτων του μοντέλου.

Η μέθοδος update_layer_parameters δέχεται ως είσοδο ένα στρώμα και ενημερώνει τα biases και τα βάρη του. Συγκεκριμένα, πρώτα υπολογίζονται οι αλλαγές που πρέπει να γίνουν στα biases και τα βάρη με βάση τον τύπο του momentum (αν το momentum δεν είναι ενεγοποιημένο, δηλαδή είναι 0, τότε το αριστερό κομμάτι της αφαίρεσης μηδενίζεται οπότε δεν επηρεάζει τους υπολογισμούς). Στη συνέχεια αν το momentum είναι ενεργοποιημένο ενημερώνεται το αποθηκευμένο momentum για το κάθε bias και το κάθε βάρος. Τέλος, εφαρμόζονται οι αλλαγές στα biases και τα βάρη.

Η μέθοδος increment_iteration_counter αυξάνει κατά ένα το μετρητή επαναλήψεων ώστε να χρησιμοποιείται στο decay. Αυτή η μέθοδος πρέπει να καλείται μετά από κάθε ενημέρωση που κάνει ο optimizer στις παραμέτρους των στρωμάτων του μοντέλου.

2.3.7 Loss

Αυτή η κλάση υλοποιεί μια συνάρτηση απώλειας. Περιέχει ένα κατασκευαστή και τη μέθοδο calculate_loss. Είναι κλάση που πρέπει να κληρονομείται από τα συγκεκριμένα είδη συναρτήσεων απώλειας και όχι που πρέπει να χρησιμοποιείται άμεσα.

Στον κατασκευαστή αρχικοποιούνται τα διανύσματα κλίσης ως προς τις εισόδους της συνάρτησης απώλειας στο None.

Η μέθοδος calculate_loss δέχεται ως είσοδο την έξοδο του μοντέλου και τις αντίστοιχες ετικέτες και επιστρέφει την μέση απώλεια. Κάνει ένα forward pass δια μέσω της συνάρτησης απώλειας (η αντίστοιχη συνάρτηση forward_pass δίνεται από την κλάση που κληρονομεί τη κλάση αυτή) και επιστρέφει τη μέση τιμή της απώλειας.

2.3.8 CategoricalCrossEntropy

Αυτή η κλάση υλοποιεί τη συνάρτηση απώλειας Categorical Cross Entropy. Περιέχει τις μεθόδους forward_pass και back_propagate. Κληρονομεί τη κλάση Loss που αναλύθηκε νωρίτερα.

Η μέθοδος forward_pass κάνει ένα forward pass δια μέσω της συνάρτησης απώλειας Categorical Cross Entropy και παίρνει ως είσοδο τις προβλέψεις του μοντέλου και τις αντίστοιχες ετικέτες. Συγκεκριμένα αποθηκεύει των αριθμό των δειγμάτων για τα οποία υπάρχουν προβλέψεις και περιορίζει τις τιμές των προβλέψεων στο διάστημα [0.0000001, 0.9999999], ώστε να μην υπάρχουν μηδενικές προβλέψεις το οποίο θα έδινε διαίρεση με το μηδέν κατά το back propagation. Στη συνέχεια δημιουργείται ένας μονοδιάστατος πίνακας με τις προβλέψεις του

μοντέλου για τις σωστές ετικέτες και επιστρέφεται ο αρνητικός λογάριθμος αυτού του πίνακα σύμφωνα με τον τύπο της Categorical Cross Entropy.

Η μέθοδος back_propagate κάνει back propagation δια μέσω του της συνάρτησης απώλειας Categorical Cross Entropy παίρνοντας ως είσοδο τα διανύσματα κλίσεων ως προς τις εισόδους που επιστρέφει το επόμενο στρώμα του μοντέλου και τις αντίστοιχες ετικέτες. Αποθηκεύει τον αριθμό των δειγμάτων και των ετικετών και αν οι ετικέτες είναι αποθηκευμένες σε ένα μονοδιάστατο πίνακα, ο πίνακας αυτός μετατρέπεται σε δισδιάστατο με άσσους στην τιμή της ετικέτας που είναι σωστή και μηδέν στις άλλες θέσεις. Τέλος, υπολογίζονται τα διανύσματα κλίσης ως προς τις εισόδους με βάση τον αντίστοιχο τύπο της Categorical Cross Entropy και αφού κανονικοποιηθούν επιστρέφονται.

2.3.9 SoftmaxCategoricalCrossEntropy

Αυτή η κλάση συνδυάζει τη συνάρτηση ενεργοποίησης SoftMax και τη συνάρτηση απώλειας Categorical Cross Entropy σε μία κλάση χρησιμοποιώντας τις αντίστοιχες κλάσεις τους. Αυτή η κλάση χρειάζεται και χρησιμοποιείται αντί των μεμονομένων αντίστοιχων κλάσεων διότι όταν χρησιμοποιούσα τις μεμωνομένες κλάσεις, δημιουργόνταν διαίρεση με το μηδέν κατά το back propagation. Με τη χρήση αυτή της κλάσης, αλλάζει ο τύπος υπολογισμού των διανυσμάτων κλίσης ως προς τις εισόδους, με αποτέλεσμα να μην δημιουργείται διαίρεση με το μηδέν. Περιέχει ένα κατασκευαστή και τις μεθόδους forward_pass και back_propagate.

Στον κατασκευαστή δημιουργούνται αντικείμενα για την κλάση ενεργοποίησης SoftMax και για την κλάση απώλειας Categorical Cross Entropy. Επίσης αρχικοποιείται στο μηδέν η έξοδος της συνάρτησης και τα διανύσματα κλίσης ως προς την είσοδό της.

Η μέθοδος forward_pass κάνει πρώτα ένα forward pass δια μέσω της συνάρτησης ενεργοποίησης Soft Max και μετά κάνει ένα ακόμα forward_pass δια μέσω της συνάρτησης απώλειας Categorical Cross Entropy και παίρνει ως είσοδο τις εισόδους της συνάρτησης ενεργοποίσης και τις ετικέτες των δεδομένων. Επίσης επιστρέφει τη μέση απώλεια. Συγκεκριμένα, πρώτα γίνεται το forward pass δια μέσω της συνάρτησης ενεργοποίησης όπως αναφέρεται στην αντίστοιχη κλάση, μετά αποθηκεύεται η έξοδος της συνάρτησης ενεργοποίησης ως η έξοδος αυτής της κλάσης και επιστρέφεται η μέση απώλεια χρησιμοποιώντας την αντίστοιχη μέθοδο της κλάσης που υλοποιεί την συνάρτηση Categorical Cross Entropy. Κατά τον υπολογισμό της μέσης απώλειας γίνεται ταυτόχρονα και το forward pass δια μέσω της συνάρτησης απώλειας Categorical Cross Entropy όπως αναφέρεται στην αντίστοιχη κλάση.

Η μέθοδος back_propagate κάνει back propagation συνδυάζοντας τις συναρτήσεις SoftMax και Categorical Cross Entropy παίρνοντας ως είσοδο τα διανύσματα κλίσεων ως προς τις εισόδους που επιστρέφει το επόμενο στρώμα του μοντέλου και τις ετικέτες των δεδομένων. Αποθηκεύει τον αριθμό των δειγμάτων και αν οι ετικέτες είναι αποθηκευμένες σε δισδιάστατο πίνακα, τις αποθηκέυει σε μονοδιάστατο πίνακα του οποίου οι τιμές είναι η θέση της σωστής ετικέτας. Μετά δημιουργεί ένα αντίγραφο των διανυσμάτων κλίσεων ως προς τις εισόδους, αφαιρεί τον άσσο από τις θέσεις των σωστών ετικετών, κανονικοποιεί τον πίνακα και

τον αποθηκεύει ως τα διανύσματα κλίσεων ως προς τις εισόδους της συνάρτησης αυτής.

2.3.10 NeuralNetwork

Αυτή η κλάση υλοποιεί το νευρωνικό δίκτυο. Προσπάθησα να κάνω τις συναρτήσεις της να μοιάζουν με τις συναρτήσεις της κλάσης Sequential του keras framework. Περιέχει ένα κατασκευαστή και τις μεθόδους add_layer, fit, evaluate και predict οι οποίες έχουν παρόμοια χρήση με τις αντίστοιχες μεθόδους της κλάσης Sequential του keras framework.

Στον κατασκευαστή αρχικοποιείται οι λιστα των στρωμάτων και η λίστα των συναρτήσεων ενεργοποίησης των αντίστοιχων στρωμάτων του μοντέλου. Επιπλέον, δημιουργείται το αντικείμενο του επιλεγμένου optimizer με τις παραμέτρους που δίνονται ως είσοδο στον κατασκευαστή.

Η μέθοδος add_layer δέχεται ως είσοδο τον αριθμό των εισόδων και νευρόνων του στρώματος, το είδος της συνάρτησης ενεργοποίησής του και το είδος του στρώματος. Δημιουργεί και προσθέτει το αντίστοιχο στρώμα και την αντίστοιχη συνάρτηση ενεργοποίησης στη λίστα στρωμάτων και στη λίστα συναρτήσεων ενεργοποίησης του μοντέλου. Ως συνάρτηση ενεργοποίησης οι επιλογές είναι η ReLU και η SoftMax Categorical Cross Entropy (δηλαδή οι δύο μαζί) καθώς η SoftMax είναι ενσωματωμένη στην τελευταία επιλογή.

Η μέθοδος fit εκπαιδεύει το μοντέλο με τα δείγματα x και τις αντίστοιχες ετικέτες y που δέχεται ως είσοδο και επιστρέφει τον χρόνο που διήρκησε η εκπαίδευση (χωρίς τον χρόνο αξιολόγησης όλων των δειγμάτων στο τέλος κάθε εποχής). Δέχεται επίσης ως είσοδο τον αριθμό των εποχών (default τιμή 1), το κάθε πόσες εποχές εμφανίζεται η πρόοδος (default τιμή 1) και το batch size (default τιμή 32).

Αρχικά, υπολογίζεται ο αριθμός των batch που θα χρειαστεί να δημιουργηθούν με βάση τα δοσμένα δείγματα και αρχικοποιείται στο μηδέν ο συνολικός χρόνος εκπαίδευσης. Για κάθε εποχή αποθηκεύεται ο χρόνος αρχής της, μετά για κάθε batch αν είναι το τελευταίο αποθηκεύονται ως τωρινό x και y τα τελευταία δείγματα και οι τελευταίες ετικέτες, αλλιώς αποθηκεύονται τα δείγματα και οι ετικέτες από την αρχή αυτού του batch μέχρι την αρχή του επόμενου μείον ένα.

Παρακάτω, για κάθε στρώμα του μοντέλου γίνεται forward pass της τωρινής εισόδου δια του στρώματος και μετά της εξόδου του στρώματος δια της συνάρτησης ενεργοποίησής του. Αν η συνάρτηση είναι η SoftMax Categorical Cross Entropy, τότε το στρώμα είναι το τελευταίο και η συνάρτηση παίρνει ως παράμετρο και το τωρινό y. Τέλος, ενημερώνεται η τωρινή είσοδος ως η έξοδος της συνάρτησης ενεργοποίησης. Επιστρέφοντας στο βρόχο που περνάει από κάθε batch, αρχικοποιούνται τα τωρινά διανύσματα κλήσης ως προς τις εισόδους που θα χρησιμοποιηθούν κατά το back propagation.

Συνεχίζοντας, γίνεται back propagation, καθώς για κάθε στρώμα του μοντέλου, αρχίζοντας από το τελευταίο και πηγαίνοντας αντίστροφα, γίνεται πρώτα back propagate δια μέσω της συνάρτησης ενεργοποίησης του στρώματος (χρησιμοποιώντας τις εξόδους του μοντέλου ως διανύσματα κλίσης αν η συνάρτηση είναι η SoftMax Categorical Cross Entropy, αλλιώς τα τωρινά διανύσματα κλήσης ως προς τις εισόδους) και μετά δια μέσω του στρώματος χρησιμοποιώντας τα δια-

νύσματα κλίσης ως προς τις εισόδους της αντίστοιχης συνάρτησης ενεργοποίησης. Τέλος, ενημερώνονται τα τωρινά διανύσματα κλήσης ως τα διανύσματα κλήσης ως προς τις εισόδους του στρώματος.

Επιστρέφοντας ξανά στο βρόχο που περνάει από κάθε batch, γίνεται το optimization. Συγκεκριμένα, ενημερώνεται το learning rate του optimizer, κάθε ένα από τα στρώματα του μοντέλου βελτιστοποιείται από τον optimizer και τέλος αυξάνεται ο μετρητής επαναλήψεων του optimizer.

Επιστρέφοντας τώρα στο βρόχο που περνάει από κάθε εποχή, πρώτα αποθηκεύεται ο χρόνος ολοκλήρωσης της εποχής και προστίθεται η διαφορά του με τον χρόνο αρχής της στον συνολικό χρόνο εκπαίδευσης. Αν η τωρινή εποχή συν ένα διαιρείται με μηδενικό υπόλοιπο από το ρυθμό εμφάνισης προόδου ή είναι η τελευταία εποχή, υπολογίζεται η μέση απώλεια και ακρίβεια του μοντέλου για όλα τα δείγματα αξιολογώντας τα με την αντίστοιχη συνάρτηση (αυτός ο χρόνος είναι που δεν μετράται στον συνολικό χρόνο εκπαίδευσης) και εμφανίζονται η απώλεια και η ακρίβεια μαζί με τη πρόοδο της εκπαίδευσης.

Η μέθοδος evaluate δέχεται ως είσοδο δείγματα και τις αντίστοιχες ετικέτες καθώς και αν χρειάζεται να εμφανιστούν η απώλεια και η ακρίβεια, υπολογίζει και αν χρειάζεται εμφανίζει την μέση ακρίβεια και απώλεια και τις επιστρέφει. Συγκεκριμένα, με τον ίδιο τρόπο που γίνεται το forward pass στην μέθοδο fit γίνεται και εδώ, αλλά με όλα τα δείγματα χωρίς χωρισμό σε batches. Η απώλεια υπολογίζεται ως η έξοδος της μεθόδου forward pass της SoftMax Categorical Cross Entropy που συνδυάζει τις δύο αντίστοιχες συναρτήσεις. Η ακρίβεια υπολογίζεται με την αντίστοιχη συνάρτηση που αναλύθηκε νωρίτερα. Τέλος, αν χρειάζεται εμφανίζονται η απώλεια και η ακρίβεια και μετά επιστρέφονται.

Τέλος, η μέθοδος predict δέχεται ως είσοδο ένα δείγμα και επιστρέφει την έξοδο του μοντέλου για το δείγμα αυτό, δηλαδή τη πιθανότητα να έχει την κάθε ετικέτα. Το forward pass γίνεται με τον ίδιο τρόπο που γίνεται στη μέθοδο evaluate, αλλά αν η συνάρτηση ενεργοποίησης είναι η SoftMax Categorical Cross Entropy, τότε γίνεται το forward pass και μέσω αυτής και επιστρέφεται η έξοδός της που είναι η έξοδος του μοντέλου για το δοσμένο δείγμα.

2.4 Ανάλυση κώδικα κυρίου σώματος

Πρώτα αποηκεύονται τα δείγματα και οι ετικέτες του Mnist dataset χρησιμοποιώντας την αντίστοιχη συνάρτηση που αναλύθηκε νωρίτερα. Μετά καλείται η συνάρτηση που τρέχει το κώδικα της ενδιάμεσης εργασίας. Μέσα σε σχόλιο υπάρχει κλήση της συνάρτησης που εκπαιδεύει και τεστάρει το νευρωνικό δίκτυο που είναι βασισμένο στο keras framework (αυτό το έχω βάλει σε σχόλιο ώστε αν θέλετε να το τρέξετε να μπορείτε αλλά να μην τρέχει γενικά γιατί δεν το ζητάει η εκφώνηση). Μετά καλείται η συνάρτηση που εκπαιδεύει και τεστάρει το νευρωνικό δίκτυο που χρησιμοποιεί τις συναρτήσεις και κλάσεις που έγραψα στο πρόγραμμα. Τέλος, μέσα σε σχόλιο υπάρχει και κλήση της συνάρτησης που ψάχνει για μια λάθος πρόβλεψη του νευρωνικού δικτύου του προγράμματος (δηλαδή το custom neural network) και εμφανίζει την έξοδο του μοντέλου που αντιστοιχεί στο δείγμα αυτό. Αν θέλεται να τρέξετε τις συναρτήσεις που είναι σε σχόλια, μπορείτε απλά να αφαιρέσαιτε το # στην αρχή της γραμμή και θα τρέξουν.

3 Ενδιάμεση εργασία

3.1 Απόδοση κατηγοριοποιητών

3.1.1 Απόδοση κατηγοριοποιητή πλησιέστερου κέντρου κλάσης

```
NCC calculations started...
-Train accuracy: 0.8079833333333334 (80.8%).
-Test accuracy: 0.8203 (82.03%).
-Time elapsed for training and scoring train and test data: 0.31 seconds.
```

Σχήμα 1: Αποτελέσματα προγράμματος για ncc

Όπως φαίνεται στο παραπάνω screenshot, για το Mnist dataset ο κατηγοριοποιητής πλησιέστερου κέντρου κλάσης δίνει ακρίβεια 80.8% για τα δεδομένα εκπαίδευσης και 82.03% για τα δεδομένα ελέγχου. Επιπλέον, στον δικό μου υπολογιστή με τους υπολογισμούς να γίνονται χωρίς GPU acceleration, η εκπαίδευση και ο υπολογισμός της απόδοσης του κατηγοριοποιητή για τα δεδομένα εκπαίδευσης και τα δεδομένα ελέγχου παίρνει 0.31 δευτερόλεπτα.

3.1.2 Απόδοση κατηγοριοποιητή πλησιέστερου γείτονα

- Απόδοση για k=1:

```
KNN calculations for k=1 started...
-k=1 train accuracy: 1.0 (100.0%)
-k=1 test accuracy: 0.9691 (96.91%)
-Time elapsed for training and scoring train and test data: 62.69 seconds.
```

Σχήμα 2: Αποτελέσματα προγράμματος για knn (k=1)

Όπως φαίνεται στο παραπάνω screenshot, για το Mnist dataset ο κα τηγοριοποιητής πλησιέστερου γείτονα με k=1 γείτονα δίνει ακρίβεια 100% για τα δεδομένα εκπαίδευσης και 96.91% για τα δεδομένα ελέγχου. Επιπλέον, στον δικό μου υπολογιστή με τους υπολογισμούς να γίνονται χωρίς GPU acceleration, η εκπαίδευση και ο υπολογισμός της απόδοσης του κατηγοριοποιητή για τα δεδομένα εκπαίδευσης και τα δεδομένα ελέγχου παίρνει 62.69 δευτερόλεπτα.

- Απόδοση για k=3:

```
KNN calculations for k=3 started...
-k=3 train accuracy: 0.9867166666666667 (98.67%)
-k=3 test accuracy: 0.9705 (97.05%)
-Time elapsed for training and scoring train and test data: 67.57 seconds.
```

Σχήμα 3: Αποτελέσματα προγράμματος για knn (k=3)

Όπως φαίνεται στο παραπάνω screenshot, για το Mnist dataset ο κατηγοριοποιητής πλησιέστερου γείτονα με k=3 γείτονες δίνει ακρίβεια 98.67% για τα δεδομένα εκπαίδευσης και 97.05% για τα δεδομένα ελέγχου. Επιπλέον, στον δικό μου υπολογιστή με τους υπολογισμούς να γίνονται χωρίς GPU acceleration, η εκπαίδευση και ο υπολογισμός της απόδοσης του κατηγοριοποιητή για τα δεδομένα εκπαίδευσης και τα δεδομένα ελέγχου παίρνει 67.57 δευτερόλεπτα.

3.2 Συμπεράσματα απόδοσης κατηγοριοποιητών

3.2.1 Συγκεντρωτικός πίνακας αποδόσεων κατηγοριοποιητών

	NCC	**KNN (K=1)**	**KNN (K=3)**
Train acc	80.80%	100.00%	98.67%
Test acc	82.03%	96.91%	97.05%
Time elapsed	0.31s	62.69s	67.57s

3.2.2 Συμπεράσματα απόδοσης κατηγοριοποιητή πλησιέστερου κέντρου κλάσης

Ο κατηγοριοποιητής πλησιέστερου κέντρου κλάσης δίνει πολύ χαμηλότερη απόδοση από τους κατηγοριοποιητές πλησιέστερου γείτονα για 1 και 3 γείτονες και για τα δεδομένα εκπαίδευσης και για τα δεδομένα ελέγχου, ωστόσο χρειάζεται πολύ λιγότερο χρόνο για να εκπαιδευτεί και να υπολογιστεί η απόδοσή του για τα δεδομένα εκπαίδευσης και ελέγχου.

3.2.3 Συμπεράσματα απόδοσης κατηγοριοποιητή πλησιέστερου γείτονα για k=1

Ο κατηγοριοποιητής πλησιέστερου γείτονα για k=1 γείτονα δίνει 100% ακρίβεια για τα δεδομένα εκπαίδευσης, το οποίο είναι αναμενόμενο αφού λαμβάνει υπόψη τη τιμή του ενός πλησιέστερου γείτονα και εφόσον τα δεδομένα εκπαίδευσης έχουν είδη αποθηκευτεί από τον κατηγοριοποιητή υπάρχει είδη η ετικέτα για κάθε είσοδο και λαμβάνεται μόνο αυτή υπόψη.

3.2.4 Συμπεράσματα απόδοσης κατηγοριοποιητή πλησιέστερου γείτονα για k=3

Ο κατηγοριοποιητής πλησιέστερου γείτονα για k=3 γείτονα δίνει την υψηλότερη ακρίβεια από τους τρεις κατηγοριοποιητές για τα δεδομένα δεδομένα ελέγχου αλλά λίγο χαμηλότερη απόδοση για τα δεδομένα εκπαίδευσης από τον αντίστοιχο κατηγοριοποιητή πλησιέστερου γείτονα για k=1 γείτονα, απαιτώντας όμως και τον περισσότερο χρόνο για την εκπαίδευση και τον υπολογισμό της απόδοσής του.

4 Τελική εργασία

4.1 Σημειώσεις

Σε αυτό το κομμάτι της έκθεσης θα χρησιμοποιήσω νευρωνικό δίκτυο που χρησιμοποιεί τις κλάσεις του προγράμματος. Στα κομμάτια όπου δεν αναφέρονται τα χαρακτηριστικά και οι παράμετροι του νευρωνικού δικτύου, το νευρωνικό δίκτυο έχει την εξής μορφή:

Αποτελείται από τρία στρώματα:

- Το στρώμα εισόδου που δέχεται 784 εισόδους (μία για κάθε τιμή των δειγμάτων του Mnist dataset) και έχει 64 νευρώνες με συνάρτηση ενεργοποίησης την ReLU.
- Το κρυμμένο στρώμα που δέχεται 64 εισόδους (μία για κάθε νευρώνα του στρώματος εισόδου) και έχει 256 νευρώνες με συνάρτηση ενεργοποίησης την ReLU.
- Το στρώμα εξόδου που δέχεται 256 εισόδους (μία για κάθε νευρώνα του κρυμμένου στρώματος) και έχει 10 νευρώνες με συνάρτηση ενεργοποίησης την SoftMax (ένα νευρώνα ανά κλάση του Mnist dataset, δηλάδη ανά ψηφίο).
- Η συνάρτηση απώλειας είναι η Categorical Cross Entropy και υπολογίζεται μαζί με την SoftMax σε μία κλάση που τις ενώνει όπως αναφέρθηκε στο αντίστοιχο κομμάτι της έκθεσης όπου αναλύεται ο κώδικας.

Διάλεξα τη παραπάνω μορφή νευρωνικού δικτύου διότι αυτή τη μορφή χρησιμοποίησε και ο κύριος Πασσαλής στα μαθήματα που μας έκανε δείχνοντάς μας το keras framework. Για να αλλάξετε τη μορφή του νευρωνικού δικτύου μπορείτε να αλλάξετε την έβδομη, ένατη και δωδέκατη γραμμή της συνάρτησης custom_neural_network, αλλάζοντας τα αντίστοιχα ορίσματα:

```
7: custom_nn.add_layer(784, 64, "ReLU", "Dense")
9: custom_nn.add_layer(64, 256, "ReLU", "Dense")
12: custom_nn.add_layer(256, 10, "SoftMaxCategoricalCrossEntropy",
"Dense")
```

Ή μπορείτε να προσθέσετε ή να αφαιρέσετε στρώματα με τη συνάρτηση:

```
custom_nn.add_layer
```

Ο optimizer είναι το Stochastic Gradient Descent και οι παράμετροί του είναι οι εξής:

- Learning rate = 0.01
- Momentum = 0.9
- Decay = 0.01

Διάλεξα τις παραπάνω παραμέτρους διότι μετά από έρευνα για τις συνηθισμένες τιμές των παραμέτρων αυτών, βρήκα πως αυτές είναι οι πιο συχνά χρησιμοποιούμενες τιμές τους. Επιπλέον δουλεύουν καλά με το νευρωνικό δίκτυο του προγράμματός μου και με το συγκεκριμένο dataset. Για να αλλάξετε τις παραμέτρους αυτές μπορείτε να αλλάξετε την τέταρτη γραμμή της συνάρτησης custom_neural_network, αλλάζοντας τα αντίστοιχα ορίσματα:

```
custom_nn = NeuralNetwork(learning_rate=0.01, momentum=0.9, decay
=0.01)
```

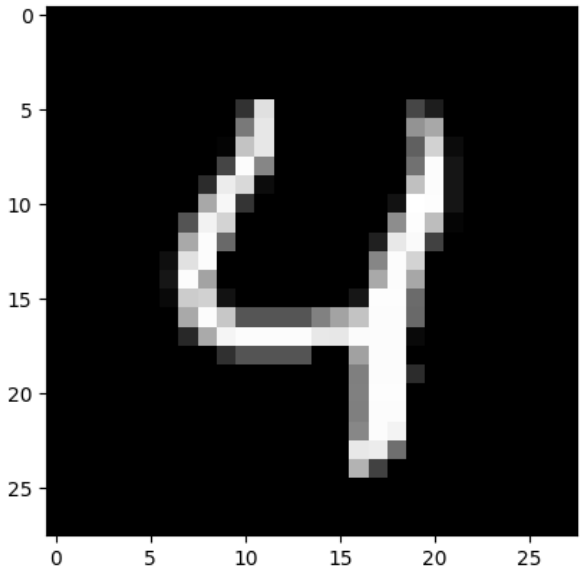
Επιπλέον, χρησιμοποιώ 20 εποχές και batch size 256. Τον αριθμό εποχών αυτό τον χρησιμοποιώ για να φαίνονται οι επιπτώσεις που έχουν οι αλλαγές των διαφόρων παραμέτρων. Το batch size αυτό το χρησιμοποιώ γιατί μου έδινε τα καλύτερα αποτελέσματα και μας το έδειξε και ο κύριος Πασσαλής στα μαθήματα. Για να αλλάξετε τις παραμέτρους αυτές μπορείτε να αλλάξετε την δέκατη πέμπτη γραμμή της συνάρτησης custom_neural_network, αλλάζοντας τα αντίστοιχα ορίσματα:

```
custom_nn.fit(x_train, y_train, 50, batch_size=256)
```

Τέλος, επειδή στο τέλος κάθε εποχής όταν εμφανίζεται η πρόοδος αξιολογούνται όλα τα δείγματα για τον υπολογισμό της απώλειας και της ακρίβειας, προστίθεται αρκετός χρόνος κατά την εκπαίδευση. Οταν υπολογίζεται ο συνολικός χρόνος τις εκπαίδευσης του νευρωνικού δικτύου, δεν μετράται σε αυτόν και ο χρόνος για την αξιολόγηση όλων των δειγμάτων. Οπότε μπορεί για παράδειγμα η εκπαίδευση να πάρει πραγματικά 60 δευτερόλεπτα για να τελειώσει αν εμφανίζει την πρόοδο σε κάθε εποχή, αλλά να λέει πως πήρε συνολικά 40 δευτερόλεπτα. Αυτό συμβαίνει επειδή πήρε 20 δευτερόλεπτα η αξιολόγηση σε κάθε εποχή των δεδομένων και η εμφάνιση των αποτελεσμάτων. Επίσης αυτό σημαίνει πως αν βάλετε μεγαλύτερο epoch_print_rate ώστε να εμφανίζεται σπανιότερα η πρόοδος κατά την εκπαίδευση, θα πάρει λιγότερο πραγματικό χρόνο η εκπαίδευση αλλά δεν θα αλλάξει ο χρόνος που λέει πως διήρκησε στο τέλος.

4.2 Χαρακτηριστικά παραδείγματα ορθής κατηγοριοποίησης

4.2.1 Πρώτο παράδειγμα



Σχήμα 4: Απεικόνιση 5ου δείγματος ελέγχου

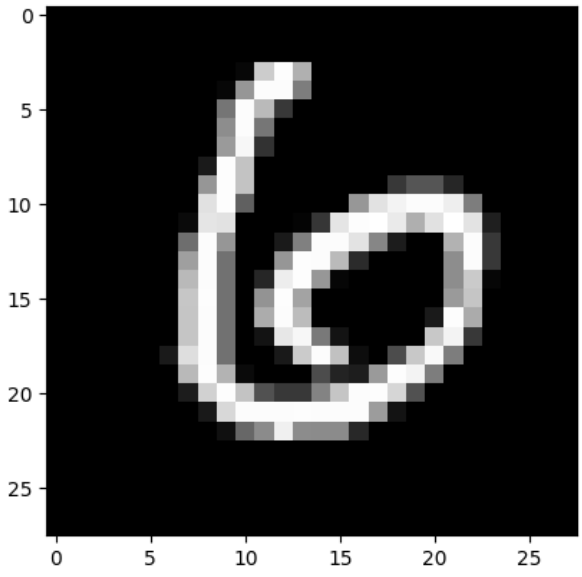
Για το 5ο δείγμα ελέγχου (αρχίζοντας από το ένα) που απεικονίζεται παραπάνω, το νευρωνικό δίκτυό μου προβλέπει σωστά την ετικέτα 4, συγκεκριμένα η έξοδός του είναι η ακόλουθη:

```
Sample 5: prediction is 4, label is 4.
Here is the model's output for sample 5:
[[8.30258803e-10 6.76076758e-12 2.53613214e-08 1.78552424e-10
  9.99838585e-01 9.26802530e-11 1.41042988e-09 3.76623003e-07
  1.40315867e-09 1.61009225e-04]]
```

Σχήμα 5: Έξοδος μοντέλου για το 5ο δείγμα ελέγχου

Όπως βλέπουμε δίνει 99.9838585% πιθανότητα να είναι 4 το δείγμα που είναι και η μεγαλύτερη από τις πιθανότητες.

4.2.2 Δεύτερο παράδειγμα



Σχήμα 6: Απεικόνιση 12ου δείγματος ελέγχου

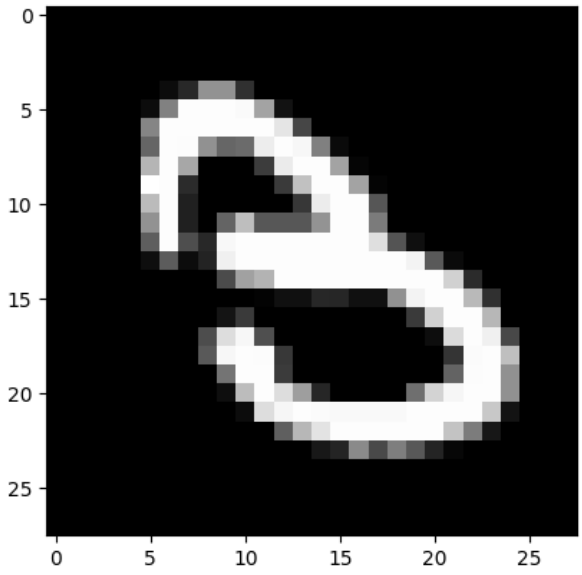
Για το 12ο δείγμα ελέγχου (αρχίζοντας από το ένα) που απεικονίζεται παραπάνω, το νευρωνικό δίκτυό μου προβλέπει σωστά την ετικέτα 6, συγκεκριμένα η έξοδός του είναι η ακόλουθη:

```
Sample 12: prediction is 6, label is 6.
Here is the model's output for sample 12:
[[2.96578965e-06 2.24196130e-09 5.65083639e-06 1.63482015e-06
  9.28714872e-08 2.13454960e-06 9.99376211e-01 5.91074631e-10
  6.09966995e-04 1.33999056e-06]]
```

Σχήμα 7: Έξοδος μοντέλου για το 12ο δείγμα ελέγχου

Όπως βλέπουμε δίνει 99.376211% πιθανότητα να είναι 6 το δείγμα που είναι και η μεγαλύτερη από τις πιθανότητες.

4.2.3 Τρίτο παράδειγμα



Σχήμα 8: Απεικόνιση 19ου δείγματος ελέγχου

Για το 19ο δείγμα ελέγχου (αρχίζοντας από το ένα) που απεικονίζεται παραπάνω, το νευρωνικό δίκτυό μου προβλέπει σωστά την ετικέτα 3, συγκεκριμένα η έξοδός του είναι η ακόλουθη:

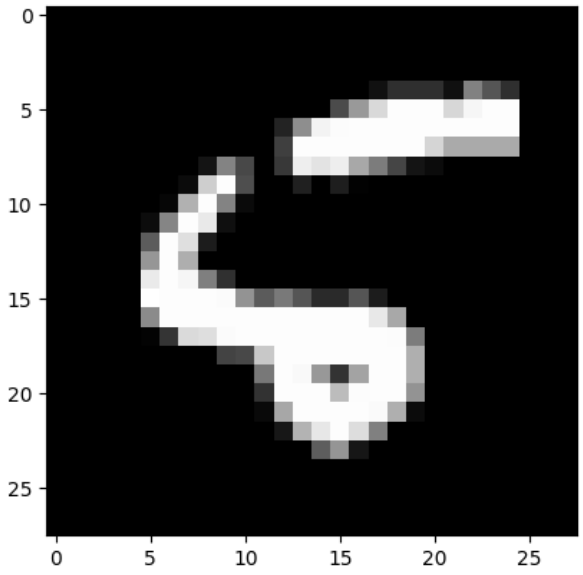
```
Sample 19: prediction is 3, label is 3.
Here is the model's output for sample 19:
[[3.99357410e-05 9.36690113e-07 1.63100230e-04 6.51661874e-01
  2.54208606e-07 7.97307267e-06 6.35581987e-07 2.55551935e-05
  3.45813496e-01 2.28623963e-03]]
```

Σχήμα 9: Έξοδος μοντέλου για το 19ο δείγμα ελέγχου

Όπως βλέπουμε δίνει 65.1661874% πιθανότητα να είναι 3 το δείγμα που είναι και η μεγαλύτερη από τις πιθανότητες. Ωστόσο για αυτό το δείγμα μερικές φορές το βγάζει ως 8 αντί για 3 επειδή όπως βλέπουμε μοιάζει με 8 του οποίου οι κύκλοι δεν είναι κλειστοί.

4.3 Χαρακτηριστικά παραδείγματα εσφαλμένης κατηγοριοποίησης

4.3.1 Πρώτο παράδειγμα



Σχήμα 10: Απεικόνιση 9ου δείγματος ελέγχου

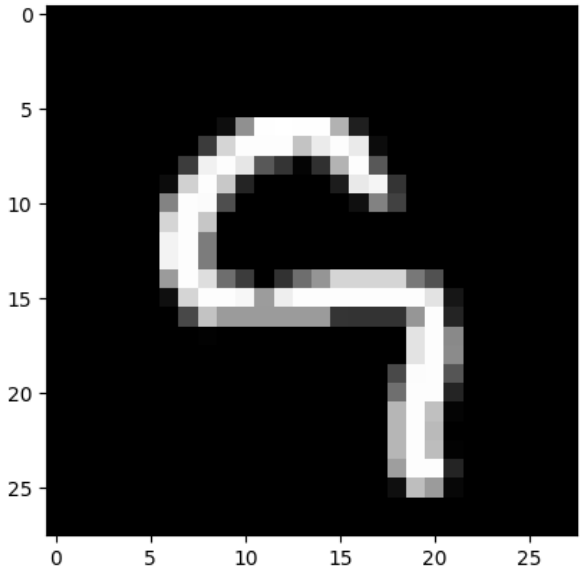
Για το 9ο δείγμα ελέγχου (αρχίζοντας από το ένα) που απεικονίζεται παραπάνω, το νευρωνικό δίκτυό μου προβλέπει εσφαλμένα την ετικέτα 4 ενώ η σωστή ετικέτα είναι το 5. Όπως βλέπουμε στην απεικόνιση του δείγματος, είναι δύσκολο ακόμη και για εμάς να αποφασίσουμε αν είναι 4 ή 5 καθώς για να είναι 4 θα έπρεπε να έχει κάθετη γραμμή στη βάση αντί για κύκλο και για να είναι 5 θα έπρεπε να μην κλείνει ο κύκλος της βάσης του, συγκεκριμένα η έξοδός του είναι η ακόλουθη:

```
Sample 9: prediction is 4, label is 5.
Here is the model's output for sample 9:
[[5.93377561e-05 1.82196483e-07 2.65860968e-05 4.36995542e-04
  5.51993229e-01 4.25247650e-01 6.94161154e-03 4.62256880e-05
  8.35824964e-03 6.88993252e-03]]
```

Σχήμα 11: Έξοδος μοντέλου για το 9ο δείγμα ελέγχου

Όπως βλέπουμε στην έξοδο, η πιθανότητες να είναι 4 ή 5 είναι οι δύο μεγαλύτερες, με την πιθανότητα να είναι 4 να είναι 12.6745579% μεγαλύτερη από την πιθανότητα να είναι 5.

4.3.2 Δεύτερο παράδειγμα



Σχήμα 12: Απεικόνιση 105ου δείγματος ελέγχου

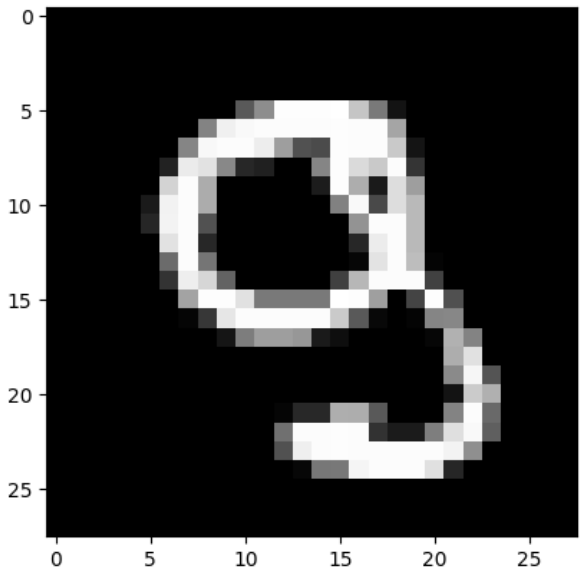
Για το 105ο δείγμα ελέγχου (αρχίζοντας από το ένα) που απεικονίζεται παραπάνω, το νευρωνικό δίκτυό μου προβλέπει εσφαλμένα την ετικέτα 5 ενώ η σωστή ετικέτα είναι το 9. Όπως βλέπουμε στην απεικόνιση του δείγματος, είναι δύσκολο ακόμη και για εμάς να αποφασίσουμε πιο ψηφίο είναι γενικά. Η βάση του δεν έχει αρκετή καμπύλη για είναι 5 αλλά ούτε ο κύκλος της κορυφής είναι κλειστός για να είναι 9. Συγκεκριμένα η έξοδος του δικτύου είναι η ακόλουθη:

```
Sample 105: prediction is 5, label is 9.
Here is the model's output for sample 105:
[[2.56002026e-06 6.28790179e-08 3.09676436e-08 3.35190813e-03
  8.79192237e-06 6.74372184e-01 2.49725284e-07 1.89949558e-05
  8.27770827e-04 3.21417447e-01]]
```

Σχήμα 13: Έξοδος μοντέλου για το 105ο δείγμα ελέγχου

Όπως βλέπουμε στην έξοδο, η πιθανότητες να είναι 5 ή 9 είναι οι δύο μεγαλύτερες, με την πιθανότητα να είναι 5 να είναι 35.2954737% μεγαλύτερη από την πιθανότητα να είναι 9.

4.3.3 Τρίτο παράδειγμα



Σχήμα 14: Απεικόνιση 152ου δείγματος ελέγχου

Για το 152ο δείγμα ελέγχου (αρχίζοντας από το ένα) που απεικονίζεται παραπάνω, το νευρωνικό δίκτυό μου προβλέπει εσφαλμένα την ετικέτα 3 ενώ η σωστή ετικέτα είναι το 9. Όπως βλέπουμε στην απεικόνιση του δείγματος, για εμάς είναι σχετικά σίγουρο πως είναι 9. Ωστόσο, το μοντέλο θεωρεί πιθανό να είναι 3, 5 ή 9. Συγκεκριμένα η έξοδος του δικτύου είναι η ακόλουθη:

```
Sample 152: prediction is 3, label is 9.
Here is the model's output for sample 152:
[[1.54047178e-02 1.88860841e-07 1.07780184e-05 3.44525761e-01
  4.64853472e-05 1.71210198e-01 4.18847072e-05 2.04544568e-03
  3.28371627e-01 1.38342913e-01]]
```

Σχήμα 15: Έξοδος μοντέλου για το 152ο δείγμα ελέγχου

Όπως βλέπουμε στην έξοδο, η πιθανότητες να είναι 3, 5 ή 9 είναι οι τρεις μεγαλύτερες, με την πιθανότητα να είναι 3 να είναι 17.3315563% μεγαλύτερη από την πιθανότητα να είναι 5 και 20.6182848% μεγαλύτερη από την πιθανότητα να είναι 9.

ς

4.4 Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου

```
Train data results:
loss: 0.0405 - accuracy: 0.9888
Test data results:
loss: 0.0974 - accuracy: 0.9727
Total time elapsed for training: 37.38 seconds.
```

Σχήμα 16: Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου

Αυτά τα αποτελέσματα είναι με 50 εποχές, ώστε να βελτιωθεί η απώλεια και η ακρίβεια του μοντέλου. Όπως φαίνεται στο παραπάνω screenshot, παρατηρούμε σχεδόν διπλασιασμό της απώλειας και μικρή μείωση της ακρίβειας πηγαίνοντας από τα δείγματα εκπαίδευσης στα δείγματα ελέγχου. Αυτό είναι λογικό καθώς τα δείγματα ελέγχου δεν έχουν χρησιμοποιηθεί για την εκπαίδευση του μοντέλου.

4.5 Χρόνος εκπαίδευσης και ποσοστά επιτυχίας για διαφορετικούς αριθμούς νευρώνων στο κρυφό επίπεδο

4.5.1 Με 64-256-10 νευρώνες

```
Train data results:
loss: 0.0497 - accuracy: 0.9853
Test data results:
loss: 0.1007 - accuracy: 0.9689
Total time elapsed for training: 16.78 seconds.
```

Σχήμα 17: Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου με 64-256-10 νευρώνες

Με τη μορφή του νευρωνικού δικτύου που αναφέρθηκε στις σημειώσεις της τελικής εργασίας, ο χρόνος εκπαίδευσής του δικτύου είναι 16.78 δευτερόλεπτα, όπως βλέπουμε στο παραπάνω screenshot. Αυτά τα νούμερα θα χρησιμοποιηθούν για σύγκριση με διαφορετικούς αριθμούς νευρώνων στο κρυφό στρώμα καθώς και με τα αποτελέσματα των αλλαγών στις παραμέτρους του δικτύου.

4.5.2 Με 32-64-128-64-10 νευρώνες

```
Train data results:
loss: 0.1951 - accuracy: 0.9449
Test data results:
loss: 0.2376 - accuracy: 0.9357
Total time elapsed for training: 16.19 seconds.
```

Σχήμα 18: Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου με 32-64-128-64-10 νευρώνες

Με στρώμα εισόδου 32 νευρώνων και 3 κρυφά στρώματα 64, 128 και 64 νευρώνων αντίστοιχα καθώς και στρώμα εξόδου 10 νευρώνων, 20 εποχές, batch size 256 και τις παραμέτρους του optimizer που χρησιμοποιήσαμε πριν, ο χρόνος εκπαίδευσής του δικτύου είναι 16.19 δευτερόλεπτα, όπως βλέπουμε στο παραπάνω screenshot.

Όπως βλέπουμε εδώ, με περισσότερα κρυφά στρώματα, αργεί περισσότερο να μειωθεί η απώλεια και να αυξηθεί η ακρίβεια. Στις 20 εποχές είναι χειρότερο από το προηγούμενο μοντέλο. Ωστόσο, ο χρόνος εκπαίδευσης δεν αλλάζει.

4.5.3 Με 256-512-128-96-10 νευρώνες

```
Train data results:
loss: 0.0282 - accuracy: 0.9936
Test data results:
loss: 0.0818 - accuracy: 0.9751
Total time elapsed for training: 53.84 seconds.
```

Σχήμα 19: Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου με 256-512-128-96-10 νευρώνες

Με στρώμα εισόδου 256 νευρώνων και 3 κρυφά στρώματα 512, 128 και 96 νευρώνων αντίστοιχα καθώς και στρώμα εξόδου 10 νευρώνων, 20 εποχές, batch size 256 και τις παραμέτρους του optimizer που χρησιμοποιήσαμε πριν, ο χρόνος εκπαίδευσής του δικτύου είναι 53.84 δευτερόλεπτα, όπως βλέπουμε στο παραπάνω screenshot.

Όπως βλέπουμε εδώ, με ίδιο αριθμό κρυφών στρωμάτων αλλά πολύ περισσότερους νευρώνες ανά στρώμα, βελτιώνεται αισθητά ή απώλεια και η ακρίβεια από όταν είχαμε τον ίδιο αριθμό στρωμάτων αλλά λιγότερους νευρώνες ανά στρώμα. Η απώλεια και η ακρίβεια είναι καλύτερες από όταν είχαμε λιγότερα στρώματα και λιγότερους νευρώνες ανά στρώμα. Επίσης πάλι η απώλεια και η ακρίβεια είναι στην αρχή χειρότερες από την πρώτη περίπτωση των τριών στρωμάτων αλλά όσο περνάνε οι εποχές γίνονται καλύτερες από τη περίπτωση των τριών στρωμάτων. Τέλος, ο χρόνος εκπαίδευσης τριπλασιάζεται περίπου από τις άλλες δύο περιπτώσεις που εξετάσαμε.

4.6 Χρόνος εκπαίδευσης και ποσοστά επιτυχίας για διαφορετικές τιμές των παραμέτρων εκπαίδευσης

4.6.1 Σημειώσεις

Εφόσον είδη δοκιμάσαμε το νευρωνικό δίκτυο για 20 και για 50 εποχές, δεν θα κάνουμε έξτρα δοκιμές σε αυτό το κομμάτι με περισσότερες εποχές, αντιθέτως θα χρησιμοποιήσουμε 20 εποχές ώστε να φαίνεται περισσότερο το αποτέλεσμα της αλλαγής των άλλων παραμέτρων στην απώλεια και την ακρίβεια του μοντέλου.

Τα αποτελέσματα, δηλαδή η απώλεια, η ακρίβεια και ο χρόνος εκπαίδευσης που δίνει το κάθε μοντέλο θα συγκρίνονται με τα αποτελέσματα της παραγράφου 4.5.1

και φυσικά μεταξύ των μοντέλων με διαφορετικές τιμές στις ίδιες παραμέτρους τους. Οπότε όταν δεν αναφέρεται από ποιό μοντέλο είναι οι τιμές σύγκρισης, εννοείται το μοντέλο της παραγράφου 4.5.1.

Στις παραγράφους όπου αλλάζουν οι παράμετροι του optimizer μόνο, δεν αλλάζει ο χρόνος εκπαίδευσης, οπότε δεν θα αναφέρεται ο χρόνος αυτός στα συμπεράσματα.

4.6.2 Αποτελέσματα για batch size=32

```
Train data results:
loss: 0.1900 - accuracy: 0.9410
Test data results:
loss: 0.2106 - accuracy: 0.9375
Total time elapsed for training: 36.5 seconds.
```

Σχήμα 20: Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου με batch size=32

Με batch size 32 ο χρόνος εκπαίδευσής του δικτύου είναι 36.5 δευτερόλεπτα, όπως βλέπουμε στο παραπάνω screenshot. Με πολύ μικρό batch size αυξάνεται ο χρόνος εκπαίδευσης (εδώ διπλασιάζεται) και η απώλεια και η ακρίβεια χειροτερεύουν. Αυτό ίσως συμβαίνει επειδή με το μικρό batch size το μοντέλο μαθαίνει πολύ καλά λίγα λίγα τα δείγματα και φτάνει δυσκολότερα στον μέσο όρο.

4.6.3 Αποτελέσματα για batch size=4096

```
Train data results:
loss: 0.1263 - accuracy: 0.9620
Test data results:
loss: 0.1480 - accuracy: 0.9547
Total time elapsed for training: 15.12 seconds.
```

Σχήμα 21: Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου με batch size=4096

Με batch size 4096 ο χρόνος εκπαίδευσής του δικτύου είναι 15.12 δευτερόλεπτα, όπως βλέπουμε στο παραπάνω screenshot. Με πολύ μεγαλύτερο batch size

δεν αλλάζει ο χρόνος εκπαίδευσης (εδώ μειώθηκε ελάχιστα) και η απώλεια και η ακρίβεια χειροτερεύουν. Όπως φαίνεται το batch size 256 που χρησιμοποιούσαμε νωρίτερα είναι μια καλή τιμή για το batch size σε σχέση με το 32 και το 4096.

4.6.4 Αποτελέσματα για batch size=60000

```
Train data results:
loss: 0.9693 - accuracy: 0.6462
Test data results:
loss: 0.9555 - accuracy: 0.6490
Total time elapsed for training: 20.05 seconds.
```

Σχήμα 22: Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου με batch size=60000

Με batch size 60000 ο χρόνος εκπαίδευσής του δικτύου είναι 20.05 δευτερόλεπτα, όπως βλέπουμε στο παραπάνω screenshot. Με ουσιαστικά ένα batch, δηλαδή εκπαιδεύοντας το μοντέλο με όλα τα δείγματα με ένα πέρασμα σε κάθε εποχή, αυξάνεται αρκετά ο χρόνος εκπαίδευσης και η απώλεια και η ακρίβεια χειροτερεύουν πάρα πολύ.

4.6.5 Αποτελέσματα για learning rate=0.1

```
Train data results:
loss: 2.3012 - accuracy: 0.1124
Test data results:
loss: 2.3010 - accuracy: 0.1135
Total time elapsed for training: 16.08 seconds.
```

Σχήμα 23: Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου με learning rate=0.1

Με μεγαλύτερο learning rate 0.1 το μοντέλο κολλάει στο σημείο που βλέπουμε στο σχήμα από το τέλος της δεύτερης εποχής κιόλας και δεν βελτιώνεται μετά. Επομένως η απώλεια και η ακρίβεια χειροτερεύουν καταστροφικά και για τα δεδομένα εκπαίδευσης και για τα δεδομένα ελέγχου. Αυτό συμβαίνει πιθανώς γιατί το μοντέλο καταλήγει σε ένα τοπικό ελάχιστο λόγω του πολύ μεγάλου learning rate και δεν μπορεί να ξεφύγει από αυτό.

4.6.6 Αποτελέσματα για learning rate=0.001

```
Train data results:
loss: 0.1506 - accuracy: 0.9571
Test data results:
loss: 0.1556 - accuracy: 0.9525
Total time elapsed for training: 17.78 seconds.
```

Σχήμα 24: Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου με learning rate=0.001

Με μικρότερο learning rate 0.001 η απώλεια και η ακρίβεια και στα δεδομένα εκπαίδευσης και στα δεδομένα ελέγχου είναι λίγο χειρότερες από τις αντίστοιχες τιμές του μοντέλου της παραγράφου 4.5.1. Επομένως το μικρότερο learning rate λειτουργεί κανονικά αλλά θα μπορούσε να αυξηθεί για να γίνει γρηγορότερα η εκπαίδευση.

4.6.7 Αποτελέσματα για momentum=0.5

```
Train data results:
loss: 0.0849 - accuracy: 0.9756
Test data results:
loss: 0.1034 - accuracy: 0.9666
Total time elapsed for training: 17.67 seconds.
```

Σχήμα 25: Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου με momentum=0.5

Δεν θα εξετάσουμε μεγαλύτερο momentum από το 0.9 γιατί ήδη είναι μεγάλο το 0.9. Με λίγο μικρότερο momentum 0.5 η απώλεια και η ακρίβεια και στα δεδομένα εκπαίδευσης και στα δεδομένα ελέγχου είναι πάλι λίγο χειρότερες από τις αντίστοιχες τιμές του μοντέλου της παραγράφου 4.5.1. Επομένως το λίγο μικρότερο momentum λειτουργεί κανονικά αλλά θα μπορούσε να αυξηθεί για να γίνει γρηγορότερα η εκπαίδευση.

4.6.8 Αποτελέσματα για momentum=0.0

```
Train data results:
loss: 0.1433 - accuracy: 0.9585
Test data results:
loss: 0.1502 - accuracy: 0.9554
Total time elapsed for training: 17.6 seconds.
```

Σχήμα 26: Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου με momentum=0.0

Με μηδενικό momentum, δηλαδή χωρίς momentum, η απώλεια και η ακρίβεια και στα δεδομένα εκπαίδευσης και στα δεδομένα ελέγχου είναι πάλι λίγο χειρότερες από τις αντίστοιχες τιμές του μοντέλου της παραγράφου 4.5.1. Είναι επίσης χειρότερες και από όταν είχαμε momentum=0.5. Επομένως το μηδενικό momentum λειτουργεί κανονικά αλλά θα μπορούσε να ενεργοποιηθεί για να γίνει γρηγορότερα η εκπαίδευση.

4.6.9 Αποτελέσματα για decay=0.1

```
Train data results:
loss: 0.1373 - accuracy: 0.9587
Test data results:
loss: 0.1428 - accuracy: 0.9557
Total time elapsed for training: 16.95 seconds.
```

Σχήμα 27: Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου με decay=0.1

Με μεγαλύτερο decay 0.1 η απώλεια και η ακρίβεια και στα δεδομένα εκπαίδευσης και στα δεδομένα ελέγχου είναι πάλι λίγο χειρότερες από τις αντίστοιχες τιμές του μοντέλου της παραγράφου 4.5.1. Το μεγαλύτερο decay έχει ως αποτέλεσμα να μικραίνει πιο γρήγορα το learning rate με αποτέλεσμα να βελτιώνεται με όλο και αργότερο ρυθμό το μοντέλο όσο περνάνε οι εποχές.

4.6.10 Αποτελέσματα για decay=0.001

```
Train data results:
loss: 0.0284 - accuracy: 0.9906
Test data results:
loss: 0.1401 - accuracy: 0.9677
Total time elapsed for training: 16.54 seconds.
```

Σχήμα 28: Ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου με decay=0.001

Με μικρότερο decay 0.001 η απώλεια και η ακρίβεια στα δεδομένα εκπαίδευσης αυξάνονται ελάχιστα από τις αντίστοιχες τιμές του μοντέλου της παραγράφου 4.5.1. Ωστόσο, το αντιθετο συμβαίνει με την απώλεια και την ακρίβεια στα δεδομένα ελέγχου.

4.7 Σύγκριση απόδοσης νευρωνικού δικτύου με τους κατηγοριοποιητές

4.7.1 Συγκεντρωτικός πίνακας αποδόσεων κατηγοριοποιητών

	NCC	**KNN (K=1)**	**KNN (K=3)**	**Network**
Train acc	80.80%	100.00%	98.67%	98.53%
Test acc	82.03%	96.91%	97.05%	96.89%
Time elapsed	0.31s	62.69s	67.57s	16.78s

4.7.2 Συμπεράσματα

- Τα συμπεράσματα ισχύουν για το μοντέλο που περιγράφεται στη παράγραφο 4.5.1 και χρησιμοποιούν τα αντίστοιχα αποτελέσματα της παραγράφου αυτής. Οπότε επηρεάζονται από την συγκεκριμένη τυχαία αρχικοποίηση των βαρών του μοντέλου.

- Το νευρωνικό δίκτυο δίνει καλύτερη ακρίβεια από τον κατηγοριοποιητή nearest class centroid και για τα δεδομένα εκπαίδευσης και για τα δεδομένα ελέγχου αλλά διαρκεί πολύ περισσότερο η εκπαίδευσή του από την εκπαίδευση του κατηγοριοποιητή ncc.

- Το νευρωνικό δίκτυο δίνει ελάχιστα χειρότερη ακρίβεια από τον κατηγοριοποιητή k nearest neighbors και για τα δεδομένα εκπαίδευσης και για τα δεδομένα ελέγχου αλλά διαρκεί πολύ λιγότερο η εκπαίδευσή του από την εκπαίδευση του κατηγοριοποιητή ncc. Αυτά ισχύουν και για k=1 και για k=3.

- Πρέπει να σημειωθεί πως αν αυξήσουμε τον αριθμό των εποχών από 20 σε 50 ή παραπάνω, πολυ πιθανώς θα ξεπεράσουμε την ακρίβεια του κατηγοριοποιητή knn για τα δεδομένα ελέγχου. Για k=1 μπορούμε στη καλύτερη περίπτωση να φτάσουμε την ακρίβεια του κατηγοριοποιητή knn για τα δεδομένα εκπαίδευσης αφού ουσιαστικά αποθηκεύει τις ετικέτες που αντιστοιχούν σε κάθε δείγμα εκπαίδεσης. Για k μεγαλύτερο του 1 μπορούμε να ξεπεράσουμε την ακρίβεια του κατηγοριοποιητή knn για τα δεδομένα εκπαίδευσης. Η ακρίβεια που θα φτάσουμε και για τα δεδομένα εκπαίδευσης και για τα δεδομένα ελέγχου εξαρτάται από το πόσο χρόνο έχουμε διαθέσιμο για την εκπαίδευση του μοντέλου καθώς και πότε θα αρχίσει να "παπαγαλίζει" τα δεδομένα.