3η εργασία Νευρωνικών Δικτύων 2024-2025

Αβραμίδου Αφροδίτη, ΑΕΜ: 10329, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Ι. ΠΡΟΛΟΓΟΣ

Από τις διαφορετικές προτάσεις για την παρούσα εργασία επιλέχθηκε η εξής:
Υλοποίηση ενός autoencoder που εκπαιδεύεται για να επιλύει το εξής πρόβλημα δημιουργίας δεδομένων: Για την βάση δεδομένων MNIST το δίκτυο κατασκευάζει το επόμενο ψηφίο από αυτό που δίνω ως είσοδο. Στην αναφορά περιλαμβάνονται η περιγραφή του αλγορίθμου, ποσοστά επιτυχίας στα στάδια εκπαίδευσης και ελέγχου για διαφορετικούς αριθμούς νευρώνων και διαφορετικές παραμέτρους, παραδείγματα ορθής και λανθασμένης ανακατασκευής. Τέλος γίνεται έλεγχος κατά πόσο μπορούν τα ψηφία των ανακατασκευασμένων εικόνων να αναγνωριστούν από ένα νευρωνικό αναγνώρισης ψηφίων, η υλοποίηση του οποίου πάρθηκε έτοιμη.

ΙΙ. ΑΝΑΛΥΣΗ ΚΩΔΙΚΑ

Εισαγωγή βιβλιοθηκών:
numpy για αριθμητικούς υπολογισμούς, matplotlib.pyplot για γραφική αναπαράσταση και σχεδίαση, time για παρακολούθηση του χρόνου εκτέλεσης κάθε epoch και tensorflow για την κατασκευή και την εκπαίδευση του νευρωνικού δικτύου. Η βάση δεδομένων MNIST φορτώνεται μέσω της Keras.

Κλάση **OneHotEncoder:**
Η κλάση αυτή χρησιμοποιείται για να μετατρέψει τις ετικέτες σε κωδικοποιημένα με κωδικοποίηση one-hot διανύσματα. Αυτό γίνεται με την μέθοδο **encode()**.

Κλάση **Encoder:**
Εδώ υλοποιείται ο encoder του μοντέλου autoencoder. Παίρνει ως είσοδο μια εικόνα και μία ετικέτα, τα κρυφά επίπεδα σταδιακά μειώνουν την διάσταση της εισόδου, καταγράφοντας σημαντικά χαρακτηριστικά και μοτίβα. Αυτά τα επίπεδα αποτελούν τον κωδικοποιητή. Το επίπεδο του στενότερου σημείου (Dense layer) είναι η τελευταία κρυφή στρώση, όπου η διάσταση μειώνεται σημαντικά. Αυτή η στρώση αντιπροσωπεύει την συμπιεσμένη κωδικοποίηση των δεδομένων εισόδου.
Η μέθοδος call() κάνει flatten την εικόνα, την συνενώνει με την κωδικοποιημένη με one-hot ετικέτα και την περνάει από ενδιάμεσα, πλήρως συνδεδεμένα επίπεδα (Dense layers) με συνάρτηση ενεργοποίησης την ReLU. Τελικά επιστρέφει την λανθάνουσα αναπαράσταση της εικόνας (latent) , δηλαδή μια αναπαράστασή της σε χαμηλότερες διαστάσεις από την αρχική, που είναι και ο σκοπός του encoder. Το ενδιαμέσο αυτό επίπεδο όπου έχει μειωθεί η αρχική διάσταση της εικόνας λέγεται “bolttleneck” (στον κώδικα υπάρχει ως παράμετρος **latent_layer**) και έχει οριστεί να αποτελείται από 64 νευρώνες (latent_dim=64)

Κλάση **Decoder:**
Υλοποιείται ο decoder του μοντέλου του autoencoder. Παίρνει ως είσοδο την κωδικοποιημένη αναπαράσταση, δηλαδή την έξοδο του encoder και προσπαθεί να ανακατασκευάσει την πρωτότυπη εικόνα.
Με την μέθοδο call() αυτής της κλάσης, το διάνυσμα της αναπαράστασης σε χαμηλότερες διαστάσεις που προέκυψε ως έξοδος του encoder, συνενώνεται με την ετικέτα και στην συνέχεια περνάει από τα ενδιάμεσα επίπεδα (Dense layers) και αναδιαμορφώνεται στο μέγεθος της επιθυμητής εικόνας. Το τελικό επίπεδο εξόδου χρησιμοποιεί σιγμοειδή συνάρτηση ενεργοποίησης για να κανονικοποιήσει τις τιμές των πίξελ μεταξύ 0 και 1. Έτσι παράγεται η ανακατασκευασμένη έξοδος, η οποία θα πρέπει να είναι όσο το δυνατόν πιο κοντά στα δεδομένα εισόδου.

Κλάση **Autoencoder:**
Συνδυάζει τον encoder και τον decoder για την υλοποίηση του autoencoder. Αντιστοιχεί την είσοδο (εικόνες και ετικέτες) στον χώρο μειωμένων διαστάσεων και έπειτα ανακατασκευάζει την αρχική εικόνα εισόδου.
Με την μέθοδο call() παίρνει ως είσοδο την εικόνα και την ετικέτα, κωδικοποιεί την εικόνα με την βοήθεια του encoder και αποκωδικοποιεί την χαμηλότερων διαστάσεων αναπαράστασή της χρησιμοποιώντας τον decoder.

Κλάση **AutoencoderTrainer:**
Σε αυτήν γίνεται η εκπαίδευση του μοντέλου του autoencoder, προσαρμόζοντας τις παραμέτρους του με τέτοιοντρόπο ώστε να ελαχιστοποιηθεί το σφάλμα ανακατασκευής. Η εκπαίδευση γίνεται χρησιμοποιώντας forward propagation και την συνάρτηση κόστους (loss function).
Οι υπερπαράμετροι που δέχεται είναι: batch_size δηλαδή ο αριθμός των δειγμάτων που χρησιμοποιείται σε κάθε βήμα εκπαίδευσης, epochs, optimizer, ο οποίος ενημερώνει τα βάρη κατά την εκπαίδευση και για τον οποίο επιλέχθηκε ως προκαθορισμένη τιμή ο adam και loss_fn όπου επιλέχθηκε ως προκαθορισμένη η συνάρτηση απωλειών binary_crossentropy, η οποία υπολογίζει τον μέσο όρο των διαφορών ανάμεσα στην προβλεπόμενη τιμή των πίξελ των δειγμάτων και στην πραγματική τους τιμή.
Με την μέθοδο train() ανατρέχεται όλο το σύνολο δεδομένων εκπαίδευσης και ενημερώνονται τα βάρη του μοντέλου με βάση τις απώλειες, δηλαδή υπολογίζονται οι διαφορές μεταξύ των πραγματικών εικόνων και των ανακατασκευασμένων από το

μοντέλο.

Κλάση **AutoencoderGenerator:**
Δημιουργεί τις εικόνες χρησιμοποιώντας το εκπαιδευμένο μοντέλο του autoencoder. Η μέθοδος generate_next_digit() έχει ως εισόδους της μια εικόνα κάποιου ψηφίου της MNIST η οποία είναι προεπεξεργασμένη έτσι ώστε ταιριάζει με την μορφή εισόδου που …. και την ετικέτα του επόμενου ψηφίου από αυτό που αναπαρίσταται στην εικόνα εισόδου. Δημιουργεί την εικόνα του ζητούμενου ψηφίου με χρήση του autoencoder (η παράμετρος model είναι ένα στιγμιότυπο του εκπαιδευμένου autoncoder) ο οποίος παράγει την εικόνα του επόμενου ψηφίου. Η δημιουργία της εικόνας του ψηφίου γίνεται ως εξής:
Ο autoencoder χρησιμοποιεί αρχικά τον encoder για να μετατρέψει την εικόνα και την ετικέτα εισόδου σε έναν χώρο αμηλότερων διαστάσεων (latent space). Έπειτα ο decoder ανακατασκευάζει την εικόνα που αντιστοιχεί στον χώρο αυτόν. Η έξοδος του autoencoder τυπικά είναι ένα μονοδιάστατο διάνυσμα (784x1) αντί για εικόνα 28x28, οπότε χρησιμοποιείται η συνάρτηση reshape για την μετατροπή σε δισδιάστατο πίνακα.

Κλάση **DigitRecognizer**:
Πάρθηκε μία έτοιμη κλάση που υλοποιεί ένα νευρωνικό δίκτυο αναγνώρισης ψηφίων, με σκοπό να χρησιμοποιηθεί για ελεγχθεί κατά πόσο το ανακατασκευασμένο ψηφίο μπορεί να αναγνωριστεί από το νευρωνικό αυτό.

Συνάρτηση **main:**
Αρχικά φορτώνεται η βάση δεδομένων MNIST και στην συνέχεια επεξεργάζεται ως εξής: πραγματοποιείται κανονικοποίηση των δεδομένων της έτσι ώστε οι τιμές των πίξελ των δειγμάτων να ανήκουν στο διάστημα [0,1]. Στην MNIST υπάρχουν 60.000 δείγματα εκπαίδευσης και 10.000 δείγματα δοκιμής, καθένα από τα οποία είναι μία εικόνα 28x28 πίξελ, δηλαδή 784 χαρακτηριστικών. Επίσης προστίθεται μία ακόμη διάσταση στα δεδομένα, η διάσταση καναλιού, και από (28, 28) γίνονται (28, 28, 1) για συμβατότητα με την είσοδο που δέχεται ο encoder και ο decoder. Έπειτα γίνεται υπολογισμός της ετικέτας του ζητούμενου επόμενου ψηφίου από αυτό που αναπαριστά η εικόνα εισόδου. Στην συνέχεια γίνεται η εξής αντιστοίχιση: Για κάθε ψηφίο από 0 έως 9 βρίσκει τους δείκτες όλων των αποτελεσμάτων του ψηφίου δοκιμής και αντιστοιχεί κάθε ψηφίο με το επόμενό του. Έτσι για κάθε ψηφίο στο X_train ή στο X_test, το αντίστοιχο επόμενο ψηφίο να είναι στο X_train_next ή στο X_test_next.

Δημιουργούνται στιγμιότυπα των κλάσεων encoder και decoder για να σχηματίσουν έναν autoencoder. Αυτός εκπαιδεύεται για 10 epochs χρησιμοποιώντας την κλάση AutoencoderTrainer, έτσι ώστε να μάθει πώς να παράγει το επόμενο ψηφίο από αυτό που δόθηκε στην είσοδο. Ο εκπαιδευμένος autoencoder χρησιμοποιείται ως είσοδος στην κλάση AutoencoderGenerator με την οποία παράγει το επόμενο ψηφίο για τυχαία επιλεγμένο σύνολο 10 εικόνων δοκιμής και τα αποτελέσματα προβάλλονται οτπικοποιημένα. Ακολουθεί η εκπαίδευση του Digit Recognizer, ο οποίος στην συνέχεια χρησιμοποιείται για να προβλέψει τις ετικέτες των παραγόμενων ψηφίων από τον encoder, που λειτουργούν ως σύνολο δοκιμής.

III. ΕΞΕΤΑΣΗ ΠΑΡΑΜΕΤΡΩΝ

Έγινε δοκιμή διαφορετικών αριθμών νευρώνων για το ενδιάμεσο επίπεδο (bottleneck) μεταξύ των encoder και decoder, με σκοπό να παρατηρηθεί το αν αυτοί επηρεάζουν την ακρίβεια ανακατασκευής των εικόνων των ψηφίων και σε ποιον βαθμό. Οι διαφορές που υπάρχουν με αλλαγή του πλήθους τον νευρώνων είναι αρκετά μικρές ώστε να μην μπορούν να παρατηρηθούν οπτικά, ωστόσο μπορούν να ανιχνευτούν παρατηρώντας τις τιμές του σφάλματος ανακατασκευής, δηλαδή της διαφοράς μεταξύ της αρχικής εικόνας εισόδου και της ανακατασκευασμένης

Number of “bottleneck” layers	Reconstruction error (loss)
2	0.2214
32	0.2214
64	0.2214
128	0.2216
256	0.2218
1000	0.2217

Πίνακας 1

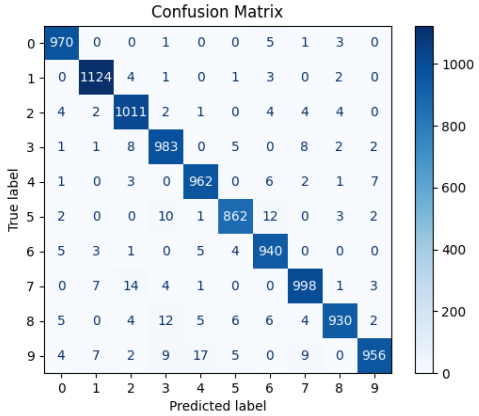
Στην περίπτωση ωστόσο του παρόντος προβλήματος δεν γίνεται ανακατασκευή του ψηφίου του δείγματος της MNIST που δίνεται στην είσοδο, αλλά ενός δείγματος που επιλέγεται από την βάση δεδομένων ως αντιπροσωπευτικό μετέπειτα, αφού γίνει η αντιστοίχιση του ποιο είναι το επόμενο ψηφίο από αυτό που απεικονίζεται στην εικόνα εισόδου. Έτσι το σφάλμα ανακατασκευής που υπολογίζεται συγκρίνοντας την είσοδο με την έξοδο. Επομένως ίσως δεν είναι τόσο αντιπροσωπευτικό ώστε να το λάβουμε υπόψιν μας, αφού εδώ αξιολογεί την ικανότητα του autoencoder να δημιουργήσει το "επόμενο ψηφίο" (όπως αντιπροσωπεύεται από το X_batch_next), όχι να ανακατασκευάσει την είσοδο (X_batch).
Ένας σωστότερος τρόπος θα ήταν να χρησιμοποιήσω τον Digit Recognizer που έχει συμπεριληφθεί στον κώδικα, ο οποίος ταξινομεί τις παραγόμενες εικόνες συγκρίνοντας τις προβλεπόμενες ετικέτες των παραγόμενων εικόνων τις αληθινές ετικέτες για το επόμενο ψηφίο (y_train_next ή y_test_next). Η υψηλή ακρίβεια υποδηλώνει ότι οι εικόνες που δημιουργούνται είναι κοντά στο πραγματικό "επόμενο ψηφίο". Επομένως παρατηρώντας την ακρίβεια που μου δίνει για διαφορετικές τιμές νευρώνων του ενδιάμεσο επιπέδου του autoencoder, μπορώ να καταλάβω σε ποιες περιπτώσεις είχα καλύτερη ανακατασκευή των ψηφίων.

Number of bottleneck’s neurons	Test Accuracy of digit recognizer (%)

2	97.14
32	97.21
64	97.21
128	97.23
256	97.08
1000	96.95

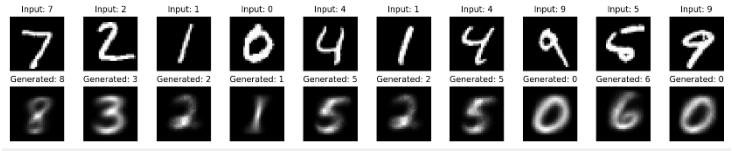
Πίνακας 2

Παρότι οι διαφορές είναι και πάλι αρκετά μικρές, παρατηρώ ότι η καλύτερη ακρίβεια παρουσιάζεται για αριθμό νευρώνων του ενδιάμεσου επιπέδου του autoencoder ίσων με 32 και 64. Για αυτούς τους αριθμούς επομένως πραγματοποιείται ελαφρώς καλύτερη ανακατασκευή των ψηφίων από τον autoencoder.

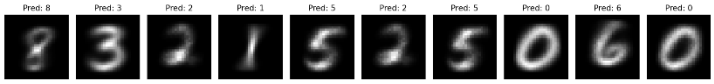


Ενδεικτικός Confusion Matrix για οπτική αναπαράσταση της ακρίβειας του Digit Recognizer

IV. ΠΑΡΑΔΕΙΓΜΑΤΑ ΕΞΟΔΟΥ ΤΟΥ ΚΩΔΙΚΑ



Δημιουργία ψηφίων από τον autoencoder



Κατηγοριοποίηση των δημιουργούμενων ψηφίων από τον Digit Recognizer