Κ23γ: Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα Χειμερινό εξάμηνο 2020-21
2η Προγραμματιστική Εργασία

Εκπαίδευση και αποτίμηση νευρωνικού δικτύου αυτοκωδικοποίησης εικόνων αριθμητικών ψηφίων. Χρήση του κωδικοποιητή για τη δημιουργία νευρωνικού δικτύου κατηγοριοποίησης των εικόνων. Εκπαίδευση και αποτίμηση του νευρωνικού δικτύου ταξινόμησης. Χρησιμοποιείται η γλώσσα Python (3.8) και η προγραμματιστική διεπαφή Keras επί της πλατφόρμας νευρωνικών δικτύων TensorFlow.

Παναγιώτης - Άγγελος Ίσηρης Α.Μ. 1115 2015 00233
Πέτρος - Φώτης Καμπέρη Α.Μ. 1115 2017 00043

Περιγραφή προγράμματος

Α ερώτημα:

Αρχικά για την ανάγνωση των συνόλων δεδομένων, υλοποιήσαμε την κλάση MnistDataloader, η οποία περιέχει συναρτήσεις για την ανάγνωση των δυαδικών αρχείων και την φόρτωση των δεδομένων σε μορφή λίστας.

Έπειτα, ορίζουμε τις συναρτήσεις κωδικοποίησης (encoder) και αποκωδικοποίησης (decoder), βασιζόμενοι στο παράδειγμα που δόθηκε στο πρώτο φροντιστήριο για την δεύτερη εργασία, με την διαφορά ότι ο αριθμός των συνελικτικών στρωμάτων και το μέγεθος των φίλτρων ανα στρώμα δίνονται δυναμικά από τον χρήστη στην main και ακολούθως δίνονται ως ορίσματα στις δύο συναρτήσεις.

Ο encoder είναι σχεδιασμένος να τρέχει για 4+ συνελικτικά στρώματα και δέχεται ως ορίσματα τις διαστάσεις κάθε στοιχείου δεδομένων (input_img, που για το δεδομένο dataset είναι ίσο με (28x28x1)), το πλήθος των συνελικτικών στρωμάτων (num_conv2d) και μια λίστα (num_conv2d_sizes) με στοιχεία το μέγεθος των φίλτρων των συνελικτικών στρωμάτων που δίνονται κατά σειρά ως εξής: 32,64,128, 256, …, $32 \cdot 2^{v}$, όπου v ο αριθμός των συνελικτικών στρωμάτων. Η δομή του encoder είναι η ακόλουθη:

- κάθε συνελικτικό στρώμα ακολουθείται από ένα στρώμα BatchNormalization
- στα δύο πρώτα συνελικτικά στρώματα, το κάθε στρώμα BatchNormalization ακολουθείται από ένα στρώμα MaxPooling2D.
- πριν από κάθε συνελικτικό στρώμα, με εξαίρεση το πρώτο, υπάρχει ένα στρώμα Dropout. Στα δύο πρώτα Dropout στρώματα δίνουμε ως παράμετρο το 0,25 , ενώ στα υπόλοιπα δίνουμε ως παράμετρο το 0,30

Ομοίως, ο decoder είναι σχεδιασμένος να τρέχει για 4+ συνελικτικά στρώματα και δέχεται ως ορίσματα την έξοδο του encoder (encode), το πλήθος των συνελικτικών στρωμάτων του encoder (num_conv2d) και μια λίστα (num_conv2d_sizes) με στοιχεία το μέγεθος των φίλτρων των συνελικτικών στρωμάτων του encoder. Η δομή του decoder είναι η ακόλουθη:

- κάθε συνελικτικό στρώμα με εξαίρεση το τελευταίο ακολουθείται από ένα στρώμα BatchNormalization.
- τα δύο τελευταία συνελικτικά στρώματα ακολουθούνται από ένα στρώμα UpSampling2D
- Επειδή τα συνελικτικά στρώματα στον decoder πρέπει είναι με αντίστροφη σειρά σε σχέση με τον decoder, η συνάρτηση στο σώμα της αντιστρέφει την λίστα με τα μεγέθη των φίλτρων των συνελικτικών στρωμάτων του encoder, ούτως ώστε να ακολουθείται η παραπάνω ιδιότητα.

Στην συνάρτηση main, αρχικά γίνεται φόρτωση του συνόλου δεδομένων, έπειτα χωρίζουμε το σύνολο δεδομένων σε σύνολο εκπαίδευσης και σύνολο ελέγχου με αναλογία 80% προς 20% και στην συνέχεια κάνουμε προεπεξεργασία των δεδομένων. Έπειτα ζητούνται επαναληπτικά από τον χρήστη οι τιμές των υπερπαραμέτρων και γίνεται η εκπαίδευση του μοντέλου που προκύπτει. Μετά το πέρας της εκπαίδευσης του μοντέλου ο χρήστης έχει επιλέγει μια από τις διαθέσιμες επιλογές, όπως αυτές ζητούνται στην εκφώνηση της εργασίας.

Β ερώτημα:

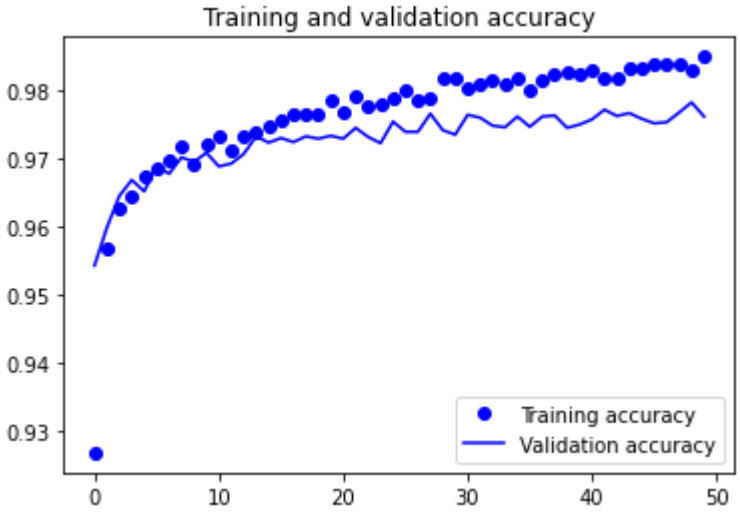
Το δεύτερο ερώτημα περιλαμβάνει το κομμάτι του Α και συνεχίζει με το ίδιο encoder όπως το προηγούμενο ερώτημα συν έναν fc layer.
Η συνάρτηση fc προσθέτει το full connected στρώμα στον autoencoder.
Έχουμε 3 επιλογές: Να κάνει πείραμα με άλλα δεδομένα. 2) Να πάρει τις γραφικές παραστάσεις των αποτελεσμάτων των μέχρι τώρα εκτελεσμένων πειραμάτων και 3) Να δει το classification report του τελευταίου πειράματος βάσει των υπερπαραμέτρων που έχουν δοθεί.(βρίσκεται στο main function) Επιπλέον στο Β ερώτημα, γίνεται train του μοντέλου μία φορά χωρίς τα ήδη trained weights και μία με πλήρες όλο το μοντέλο μαζί με τα pretrained κομμάτια του.

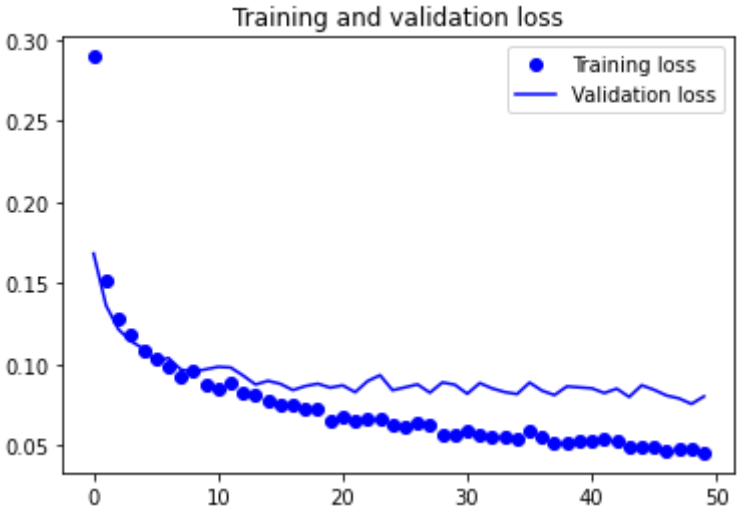
Αρχεία και περιγραφή:

1. autoencoder.py: στο αρχείο αυτό υλοποιείται το νευρωνικό δίκτυο αυτοκωδικοποίησης εικόνων Ν1 και τα ζητούμενα του Α ερωτήματος.
2. classification.py: στο αρχείο αυτό υλοποιείται το νευρωνικό δίκτυο κατηγοριοποίησης εικόνων Ν2 και τα ζητούμενα του Β ερωτήματος.

Σχολιασμός Αποτελεσμάτων:

64 batch size, 50 epochs, 4 convolutional layers with respective sizes: 32, 64, 128, 256





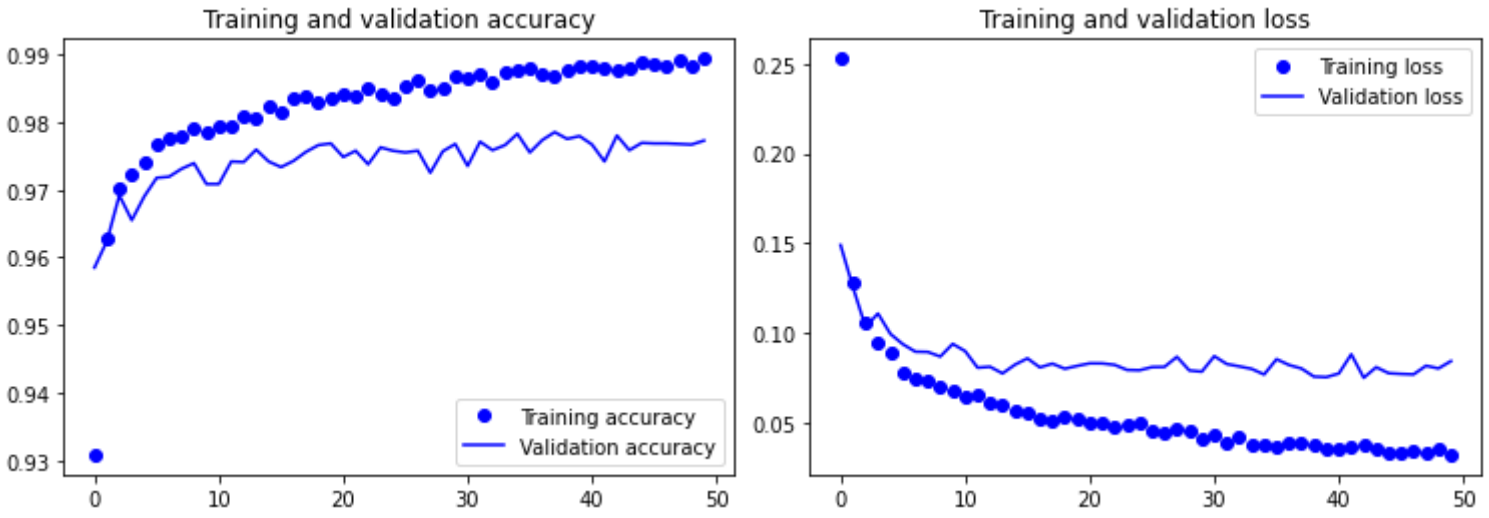
Test accuracy:	0.9858999848365784
Test loss:	0.04312944784760475
Found correct labels	9851
Found incorrect labels	149

Classification Report:

	precision	recall	f1-score	support
Class 0	0.95	1.00	0.98	980
Class 1	1.00	0.99	0.99	1135
Class 2	0.98	0.99	0.99	1032
Class 3	0.99	0.99	0.99	1010
Class 4	1.00	0.97	0.98	982
Class 5	1.00	0.99	0.99	892
Class 6	0.99	0.99	0.99	958
Class 7	0.99	0.97	0.98	1028
Class 8	0.99	0.98	0.98	974
Class 9	0.98	0.99	0.99	1009

accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.96	0.99	0.98	10000

128 batch size, 50 epochs, 4 convolutional layers with respective sizes: 32, 64, 128, 256



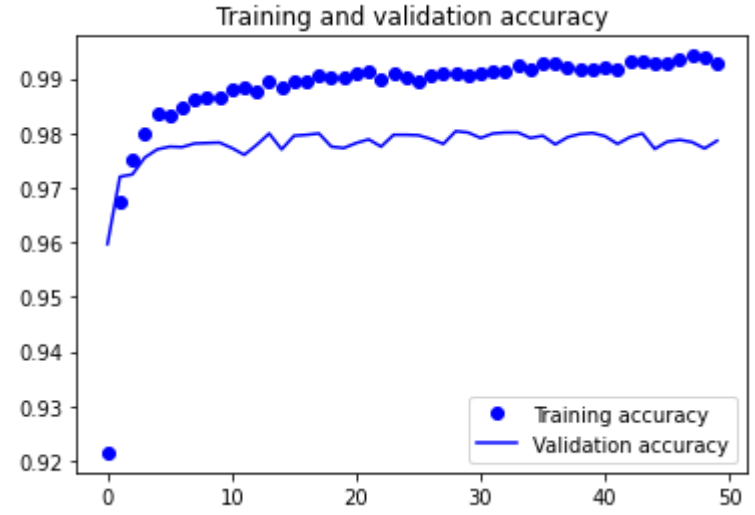
Test accuracy:	0.9904999732971191
Test loss:	0.02873251587152481
Found correct labels	9902
Found incorrect labels	98

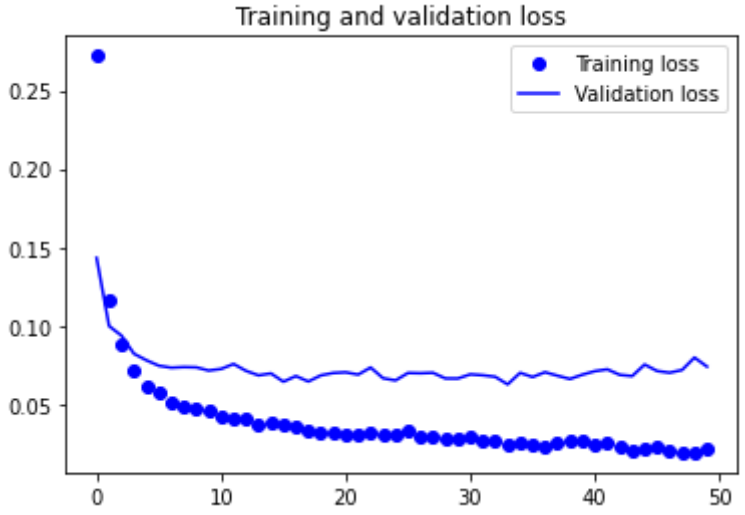
Classification Report:

	precision	**recall**	**f1-score**	**support**
Class 0	0.98	1.00	0.99	980
Class 1	0.99	1.00	0.99	1135
Class 2	0.99	0.99	0.99	1032
Class 3	1.00	0.99	0.99	1010
Class 4	0.99	0.98	0.99	982
Class 5	0.99	0.99	0.99	892
Class 6	0.99	0.99	0.99	958

Class 7	1.00	0.98	0.99	1028
Class 8	0.99	0.99	0.99	974
Class 9	0.98	0.99	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

256 batch size, 50 epochs, 4 convolutional layers with respective sizes: 32, 64, 128, 256





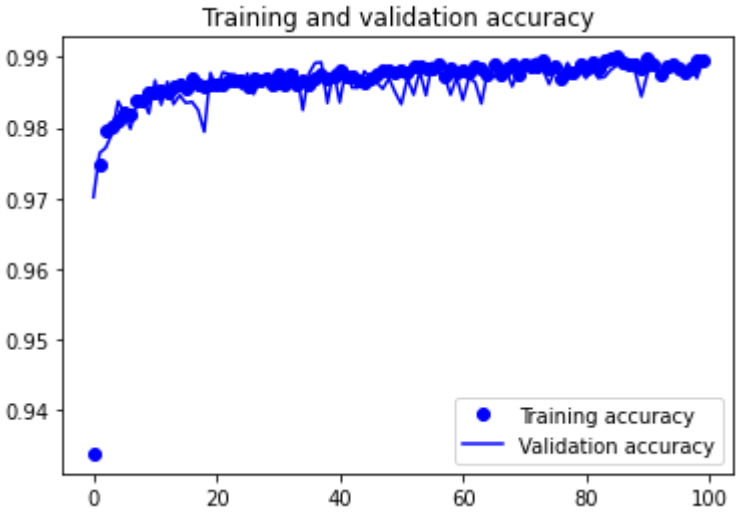
Test accuracy:	0.9925000071525574
Test loss:	0.02468124032020569
Found correct labels	9922
Found incorrect labels	78

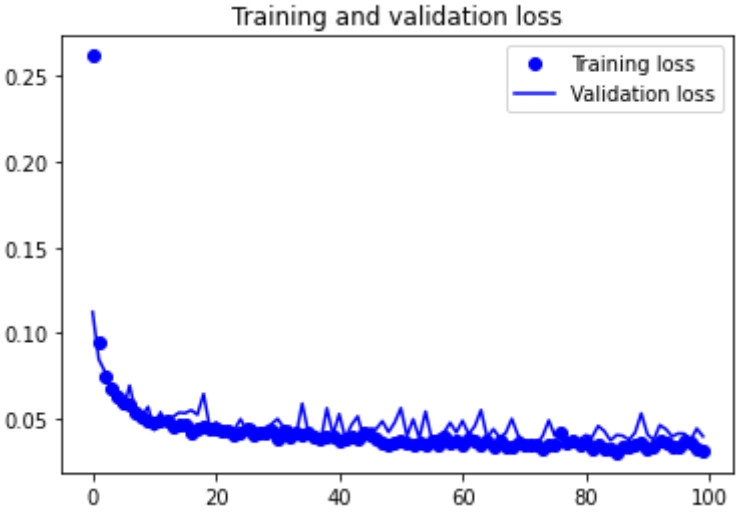
Classification Report:

	precision	**recall**	**f1-score**	**support**
Class 0	0.98	1.00	0.99	980
Class 1	1.00	1.00	1.00	1135
Class 2	0.99	1.00	1.00	1032

Class 3	0.99	0.99	0.99	1010
Class 4	1.00	0.97	0.99	982
Class 5	1.00	0.99	0.99	892
Class 6	1.00	0.99	0.99	958
Class 7	1.00	0.99	0.99	1028
Class 8	0.99	1.00	0.99	974
Class 9	0.98	1.00	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

64 batch size, 100 epochs, 4 convolutional layers with respective sizes: 32, 64, 128, 256



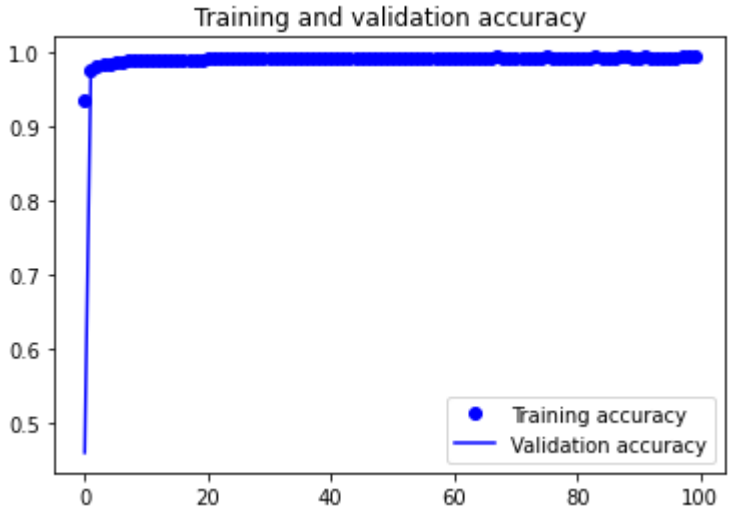


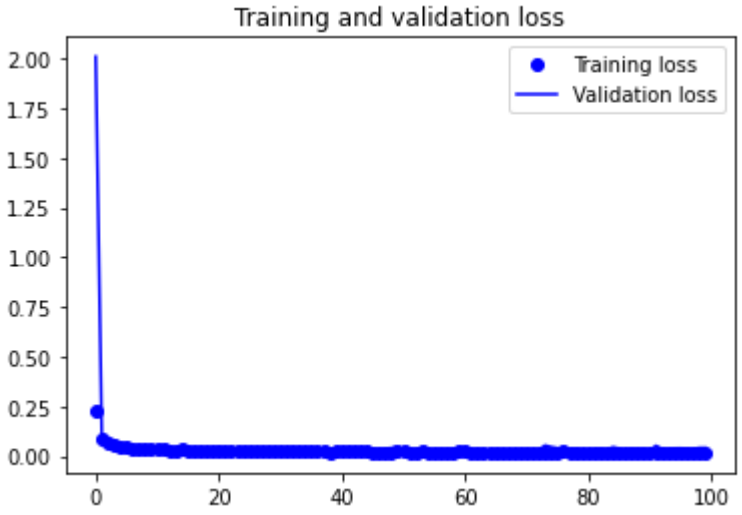
Test accuracy:	0.9904999732971191
Test loss:	0.03152688220143318
Found correct labels	9902
Found incorrect labels	98

Classification Report:

	precision	recall	f1-score	support
Class 0	0.98	0.99	0.99	980
Class 1	1.00	1.00	1.00	1135
Class 2	0.99	1.00	0.99	1032
Class 3	0.99	0.99	0.99	1010
Class 4	0.99	0.98	0.99	982
Class 5	0.99	0.99	0.99	892
Class 6	0.99	0.99	0.99	958
Class 7	1.00	0.98	0.99	1028
Class 8	0.99	0.99	0.99	974
Class 9	0.98	0.99	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

128 batch size, 100 epochs, 4 convolutional layers with respective sizes: 32, 64, 128, 256



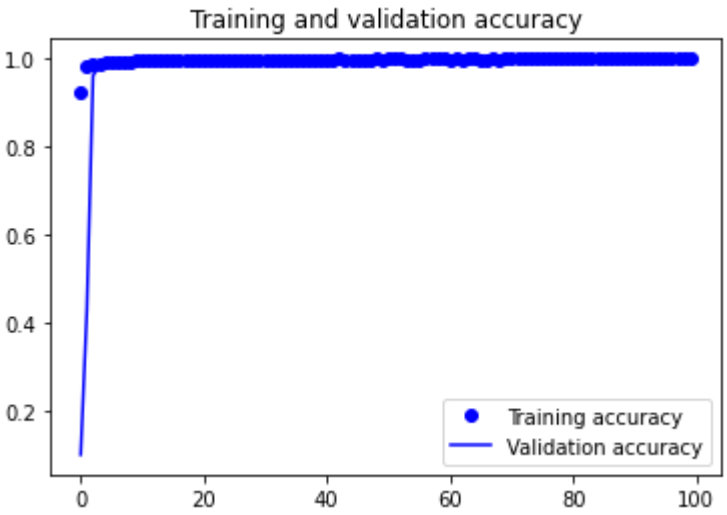


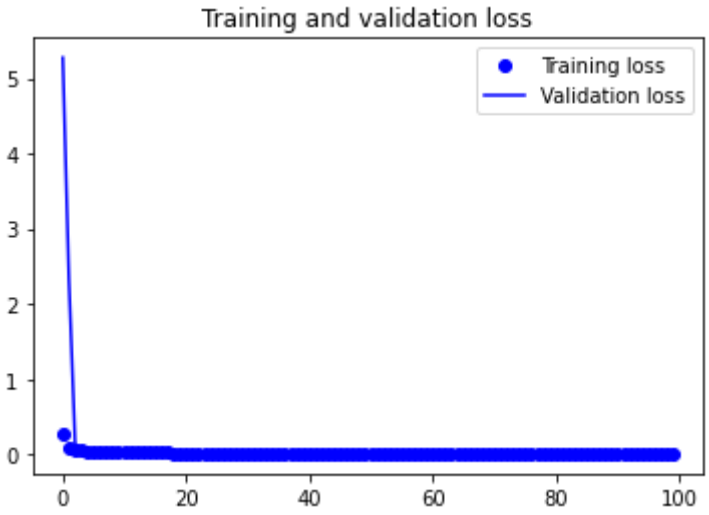
Test accuracy:	0.9904000163078308
Test loss:	0.03340807184576988
Found correct labels	9903
Found incorrect labels	97

Classification Report:

	precision	recall	f1-score	support
Class 0	0.97	1.00	0.98	980
Class 1	1.00	0.99	0.99	1135
Class 2	0.99	1.00	0.99	1032
Class 3	1.00	0.99	0.99	1010
Class 4	0.99	0.99	0.99	982
Class 5	0.99	0.99	0.99	892
Class 6	0.99	0.99	0.99	958
Class 7	1.00	0.98	0.99	1028
Class 8	0.99	0.99	0.99	974
Class 9	0.99	0.99	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

256 batch size, 100 epochs, 4 convolutional layers with respective sizes: 32, 64, 128, 256





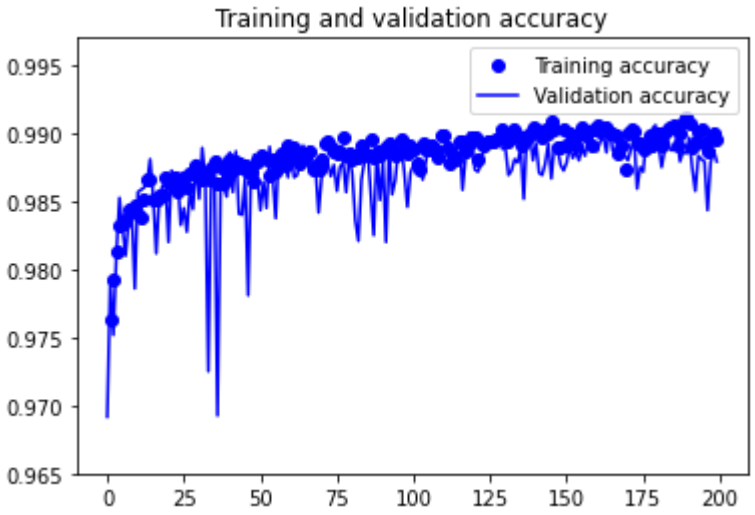
Test accuracy:	0.9925000071525574
Test loss:	0.02420792169868946
Found correct labels	9923
Found incorrect labels	77

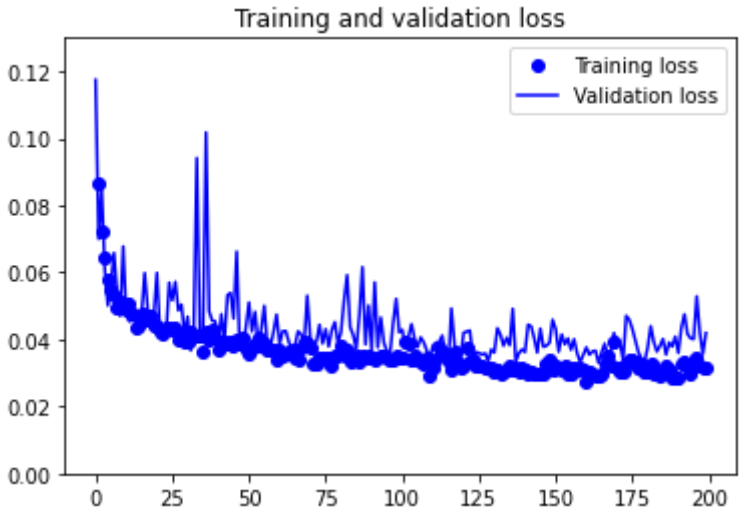
Classification Report:

	precision	recall	f1-score	support
Class 0	0.99	1.00	0.99	980
Class 1	0.99	0.99	0.99	1135
Class 2	0.99	1.00	0.99	1032
Class 3	1.00	0.99	0.99	1010
Class 4	1.00	0.99	0.99	982
Class 5	0.99	1.00	0.99	892
Class 6	0.99	0.99	0.99	958
Class 7	1.00	0.98	0.99	1028
Class 8	0.99	0.99	0.99	974
Class 9	0.98	0.99	0.99	1009

accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

64 batch size, 200 epochs, 4 convolutional layers with respective sizes: 32, 64, 128, 256





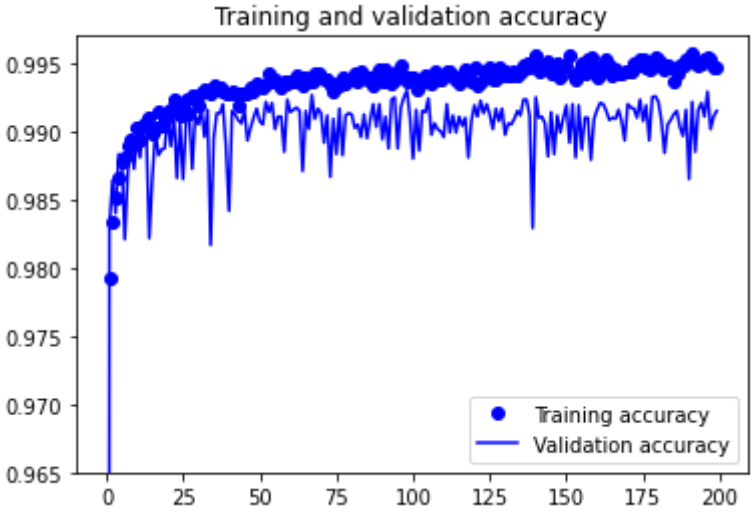
Test accuracy:	0.9894000291824341
Test loss:	0.03264113888144493
Found correct labels	9982
Found incorrect labels	107

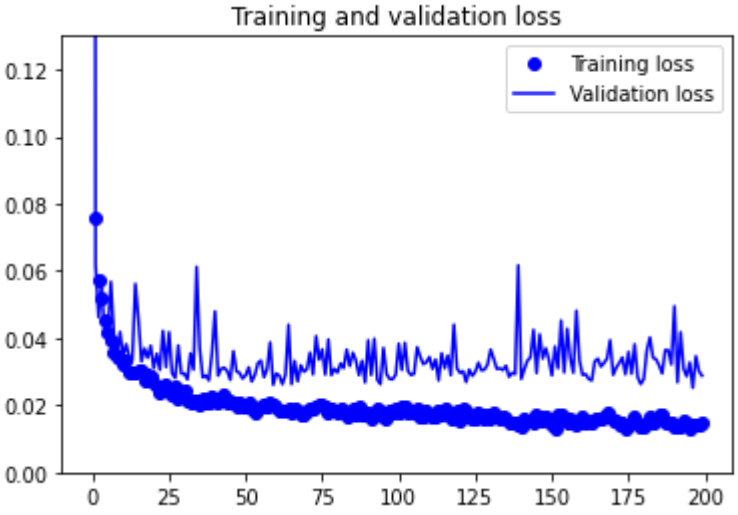
Classification Report:

	precision	**recall**	**f1-score**	**support**
Class 0	0.98	1.00	0.99	980
Class 1	1.00	0.99	1.00	1135
Class 2	0.98	0.99	0.99	1032
Class 3	0.99	0.99	0.99	1010
Class 4	1.00	0.98	0.99	982
Class 5	0.99	0.99	0.99	892
Class 6	0.99	0.99	0.99	958
Class 7	0.99	0.98	0.99	1028

Class 8	0.98	0.99	0.99	974
Class 9	0.99	0.98	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

128 batch size, 200 epochs, 4 convolutional layers with respective sizes: 32, 64, 128, 256 best





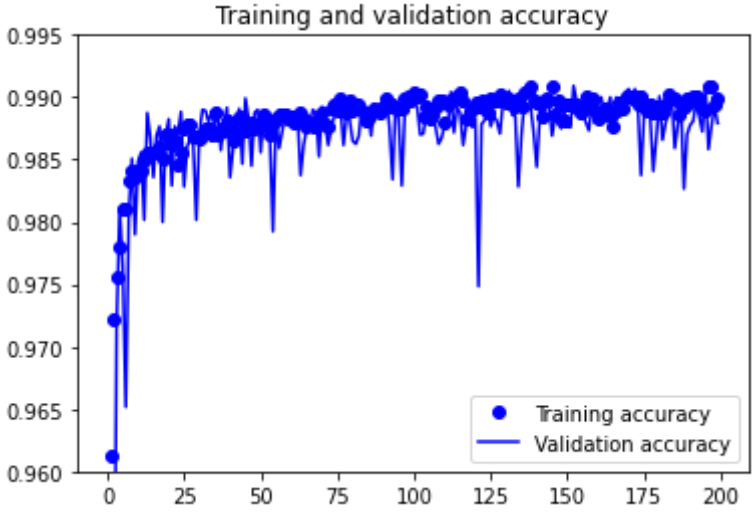
Test accuracy:	0.9926999807357788
Test loss:	0.02394816279411316
Found correct labels	9925
Found incorrect labels	75

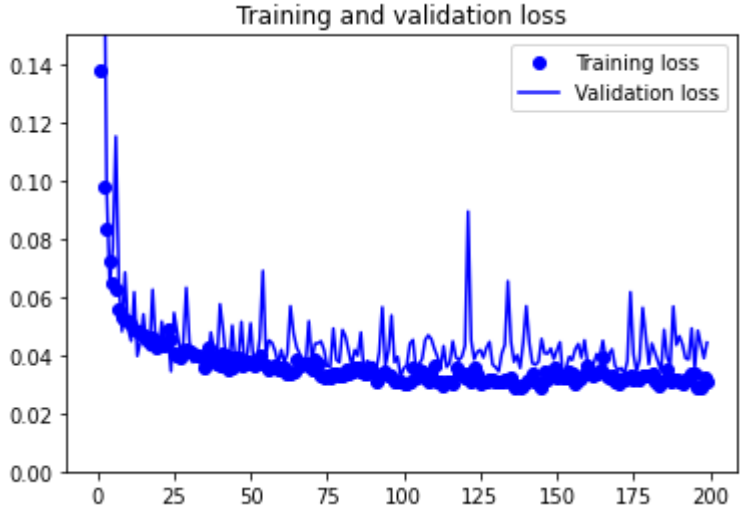
Classification Report:

	precision	**recall**	**f1-score**	**support**
Class 0	0.98	1.00	0.99	980
Class 1	0.99	1.00	1.00	1135
Class 2	0.98	0.99	0.99	1032

Class 3	0.99	0.99	0.99	1010
Class 4	0.99	0.99	0.99	982
Class 5	0.99	0.99	0.99	892
Class 6	1.00	0.99	0.99	958
Class 7	0.99	0.99	0.99	1028
Class 8	1.00	0.99	0.99	974
Class 9	0.99	0.99	0.99	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

256 batch size, 200 epochs, 4 convolutional layers with respective sizes: 32, 64, 128, 256



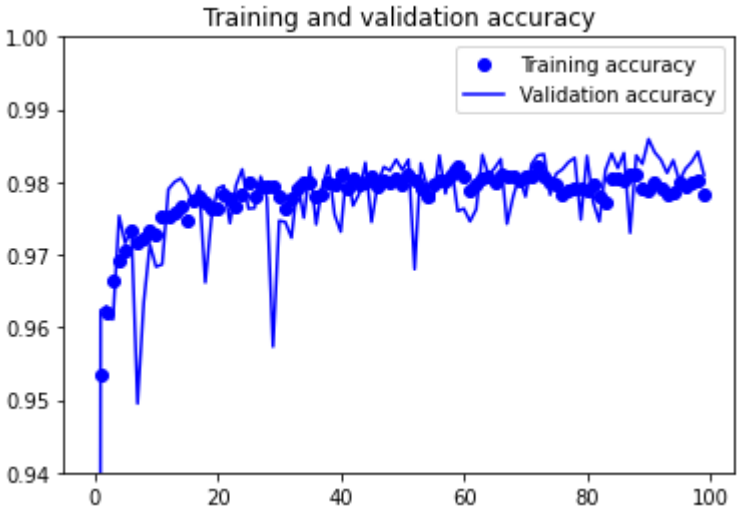


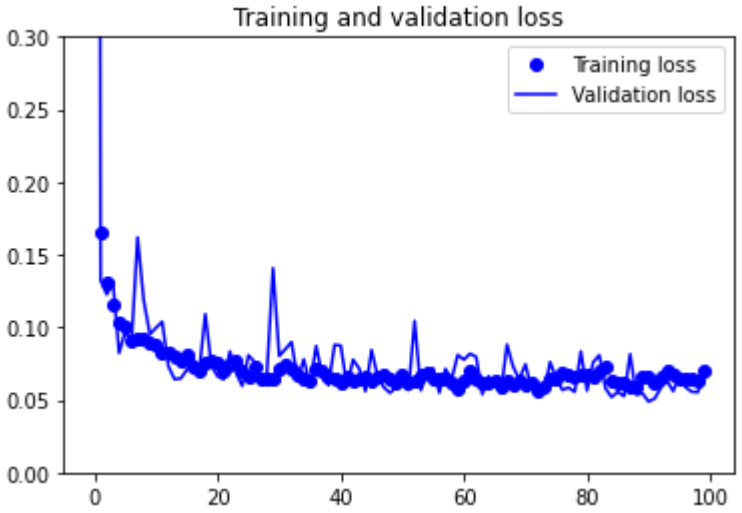
Test accuracy:	0.989300012588501
Test loss:	0.030454842373728752
Found correct labels	9890
Found incorrect labels	110

Classification Report:

	precision	recall	f1-score	support
Class 0	0.98	1.00	0.99	980
Class 1	0.99	0.99	0.99	1135
Class 2	0.98	1.00	0.99	1032
Class 3	0.99	0.99	0.99	1010
Class 4	1.00	0.99	0.99	982
Class 5	0.99	0.99	0.99	892
Class 6	1.00	0.99	0.99	958
Class 7	1.00	0.98	0.99	1028
Class 8	0.98	0.99	0.99	974
Class 9	0.98	0.98	0.98	1009
accuracy			0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000

128 batch size, 100 epochs, 5 convolutional layers with respective sizes: 32, 64, 128, 256, 512





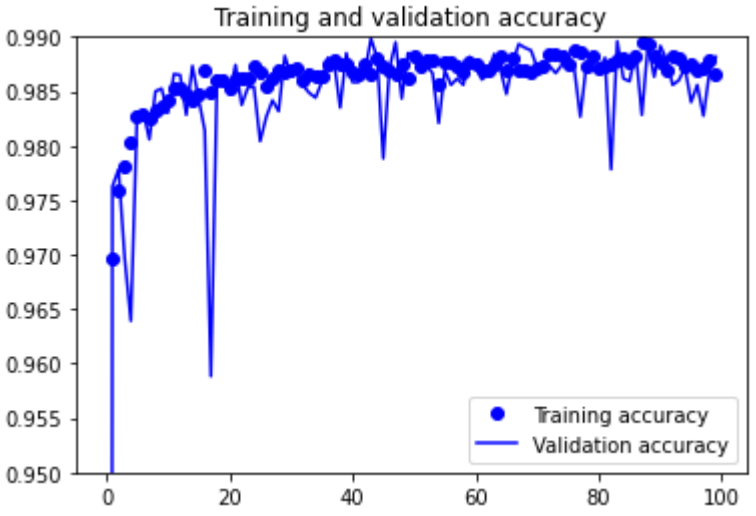
Test accuracy:	0.9815000295639038
Test loss:	0.05675699934363365
Found correct labels	9808
Found incorrect labels	198

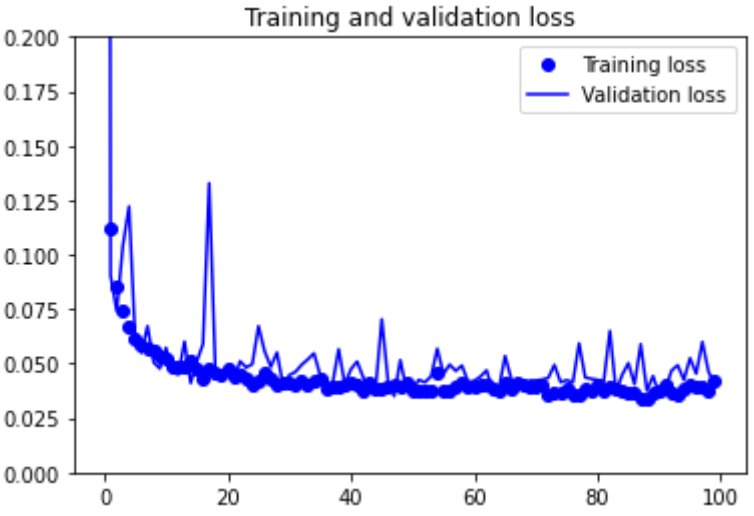
Classification Report:

	precision	recall	f1-score	support
Class 0	0.96	0.99	0.98	980
Class 1	1.00	0.99	0.99	1135
Class 2	0.98	0.97	0.98	1032
Class 3	0.98	0.98	0.98	1010
Class 4	0.99	0.98	0.98	982
Class 5	0.97	0.98	0.97	892
Class 6	0.99	0.98	0.98	958
Class 7	0.98	0.97	0.98	1028
Class 8	0.97	0.97	0.97	974
Class 9	0.98	0.98	0.98	1009

accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

128 batch size, 100 epochs, 6 convolutional layers with respective sizes: 32, 64, 128, 256, 512, 1028





Test accuracy:	0.9889000058174133
Test loss:	0.032552555203437805
Found correct labels	9808
Found incorrect labels	198

Classification Report:

	precision	**recall**	**f1-score**	**support**
Class 0	0.94	0.99	0.97	980
Class 1	0.99	0.99	0.99	1135
Class 2	0.98	0.98	0.98	1032

Class 3	1.00	0.98	0.99	1010
Class 4	0.99	0.96	0.98	982
Class 5	0.99	0.99	0.99	892
Class 6	0.99	0.98	0.98	958
Class 7	1.00	0.97	0.98	1028
Class 8	0.96	0.98	0.97	974
Class 9	0.97	0.97	0.97	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

Με βάση τα παραπάνω πειραματικά αποτελέσματα μπορούμε να συμπεράνουμε τα εξης:

- αύξηση του πλήθους των συνελικτικών στρωμάτων έχει ως αποτέλεσμα αύξηση του σφάλματος.
- αύξηση των epochs οδηγεί σε μικρότερες τιμές σφάλματος. Ειδικότερα, όταν το batch size δεν διαφέρει πολύ από την τιμή των epochs (πχ batch size 128 και epochs 100 ή batch size 256 και epochs 200) παρατηρούμε ότι έχουμε τα καλύτερα αποτελέσματα.
- Παρατηρούμε ότι το batch size 128 μας δίνει αρκετά καλά αποτελέσματα εκτός από τις περιπτώσεις με πολύ μικρό αριθμό epoch. Επίσης παρατηρείται μια μικρή απόκλιση που τείνει προς underfitting στο πείραμα με batch size 128 και epochs 200 πράγμα που επιβεβαιώνει την προηγούμενη μας παρατήρηση.
- σε άλλα πειράματα που κάναμε και δεν συμπεριλάβαμε στο παρόν README είδαμε ότι μεγάλη τιμή batch size (πχ 2048) ή μικρή τιμή epochs (πχ 10) οδηγεί σε μεγάλη αύξηση του σφάλματος, παρόλο που η ταχύτητα εκτέλεσης ήταν γρηγορότερη.
- Επίσης, παρατηρείται ότι, για καποιες κλάσεις το μοντέλο έχει μεγαλύτερη ακρίβεια από κάποιες άλλες πχ.(η κλάση 1).

Μεταγλώττιση και εκτέλεση προγραμμάτων:

1. python autoencoder.py -d train-images-idx3-ubyte
2. python classification.py -d train-images-idx3-ubyte -dl train-labels-idx1-ubyte -t t10k-images-idx3-ubyte -tl t10k-labels-idx1-ubyte -model autoencoder.h5

Πηγές από το διαδίκτυο:

- Για την ανάγνωση των δυαδικων αρχείων των συνόλων δεδομένων: https://www.kaggle.com/hojjatk/read-mnist-dataset
- Για την υλοποίηση των νευρωνικών δικτύων: https://keras.io/guides/functional_api/, https://www.datacamp.com/community/tutorials

Σχόλια - διευκρινίσεις:

1. Το πρόγραμμα τρέχει για το λιγότερο 4 επίπεδα φίλτρων. Η υλοποίηση έγινε σε google colab λόγω έλλειψης hardware (έλλειψη gpu). Τα υπόλοιπα μέρη του προγράμματος όπως τα ορίσματα από την γραμμή εντολών έγιναν στον υπολογιστή μας και το πρόγραμμα δοκιμάστηκε και εκεί με το tensorflow για cpu. (Δεν υπάρχει κάποια διαφορά στην υλοποίηση, απλά διαφέρει η ταχύτητα εκτέλεσης του προγράμματος χωρίς την χρήση gpu)
2. Το μέγεθος των φίλτρων πρέπει να δίνεται με την εξής σειρά: 1ο -> 32, 2ο -> 64, 3ο -> 128, 4ο -> 256 και ούτω καθεξής δήλαδή αυξάνεται το μέγεθος με βάση τις δυνάμεις του 2 επί 32.
3. Στον παραδοτέο φάκελο έχουμε συμπεριλάβει επίσης ένα ήδη εκπαιδευμένο μοντέλο που βρίσκεται στο αρχείο autoencoder.h5. Το μοντέλο αυτό είχε τις εξής τιμές υπερπαραμέτρων: batch_size = 128, epochs = 100, αριθμός convolutional layer = 4 και αντίστοιχα μεγέθη κάθε layer [32, 64, 128, 256].