



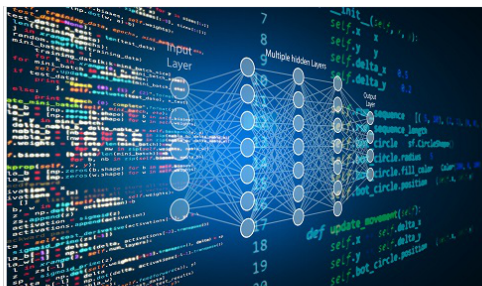
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών
— ΙΔΡΥΘΕΝ ΤΟ 1837 —

ΑΝΑΠΤΥΞΗ ΛΟΓΙΣΜΙΚΟΥ ΓΙΑ ΑΛΓΟΡΙΘΜΙΚΑ ΠΡΟΒΛΗΜΑΤΑ

ΧΕΙΜΕΡΙΝΟ ΕΞΑΜΗΝΟ 2020

3Η ΠΡΟΓΡΑΜΜΑΤΙΣΤΙΚΗ ΕΡΓΑΣΙΑ

ΔΙΑΝΥΣΜΑΤΙΚΗ ΑΝΑΠΑΡΑΣΤΑΣΗ ΕΙΚΟΝΑΣ ΣΕ ΧΩΡΟ ΧΑΜΗΛΟΤΕΡΗΣ ΔΙΑΣΤΑΣΗΣ ΜΕ ΧΡΗΣΗ ΝΕΥΡΩΝΙΚΟΥ ΔΙΚΤΥΟΥ ΑΥΤΟΚΩΔΙΚΟΠΟΙΗΣΗΣ.



Αριθμός Μητρώου(ΑΜ):

1115201700217

1115201700203

Ονοματεπώνυμο:

Ορέστης ΣΤΕΦΑΝΟΥ

Λεωνίδας ΕΦΡΑΙΜ

ΑΚΑΔΗΜΑΪΚΗ ΧΡΟΝΙΑ 2020-2021

ΠΕΡΙΕΧΟΜΕΝΑ

1	ΕΙΣΑΓΩΓΗ	3
2	ΜΕΡΟΣ Α' AUTOENCODER	4
2.1	ΥΛΟΠΟΙΗΣΗ	4
2.2	ΠΕΙΡΑΜΑΤΑ - REPORTS	5
3	ΜΕΡΟΣ Β' LSH	10
3.1	ΥΛΟΠΟΙΗΣΗ	10
3.2	ΠΕΙΡΑΜΑΤΑ - REPORTS	10
4	ΜΕΡΟΣ Γ' Earth Mover's Distance (EMD)	12
4.1	ΥΛΟΠΟΙΗΣΗ	12
4.2	ΠΕΙΡΑΜΑΤΑ - REPORTS	12
5	ΜΕΡΟΣ Δ' CLUSTERING	12
5.1	ΥΛΟΠΟΙΗΣΗ	12
5.2	ΠΕΙΡΑΜΑΤΑ - REPORTS	12

ΕΙΣΑΓΩΓΗ

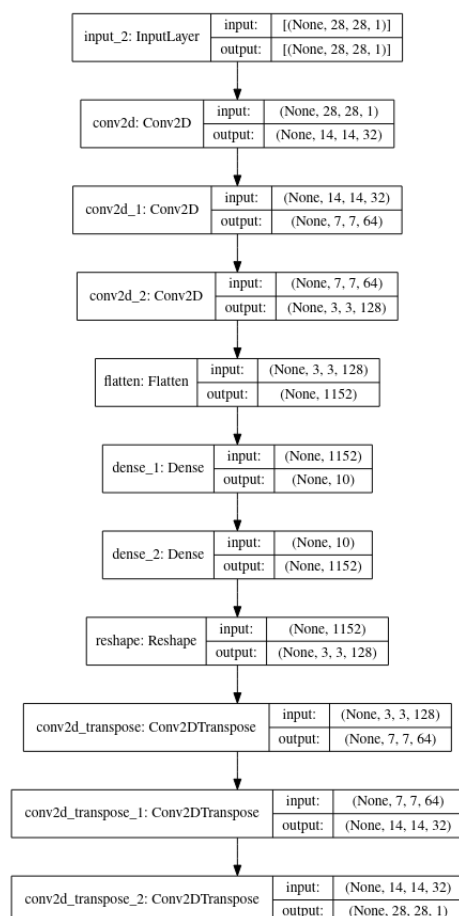
Σε αυτή την εργασία κληθήκαμε να εξάγουμε τα συμπιεσμένα δεδομένα απο το ένα ένα νευρωνικό δίκτυο αυτοκωδικοποίησης ψηφιακών εικόνων. Στην συνέχεια αυτά τα δεδομένα τα χρησιμοποιήσαμε για κάνουμε συσταδοποίηση και να συγκρίνουμε τα αποτελέσματα με τον αρχικό χώρο. Επίσης είχαμε να υλοποιήσουμε τη μετρική Earth Mover's Distance που ανάγεται σε επίλυση προβλήματος Γραμμικού Προγραμματισμού για να ξανακάνουμε συσταδοποίηση και να συγκρίνουμε τον χρόνο και την ορθότητα των αποτελεσμάτων. Τέλος είχαμε να κάνουμε συσταδοποίηση K-medians των εικόνων στον παλίο και τον καινούριο καθώς και σύσταδοποίηση βάση του μοντέλου του clustering της δευτερης εργασίας με σκοπό να τα συγκρίνουμε. Για την υλοποίηση χρησιμοποιήσαμε τη γλώσσα Python με τη βοήθεια των βιβλιοθηκών Keras και Tensorflow. Εκτός από αυτά χρησιμοποιήσαμε και το Google Collab το οποίο μας παρείχε επεξεργαστική ισχύ για τους μεγάλους υπολογισμούς που χρειαστήκαμε μέσω των GPU που μας παρείχε. Στο πρώτο μέρος δημιουργήσαμε τον encoder και τον εκπαιδεύσαμε, ενώ στη συνέχεια στο δεύτερο μέρος υλοποιήσαμε και κατηγοριοποίηση στον encoder μας.

ΜΕΡΟΣ Α' AUTOENCODER

2.1 ΥΛΟΠΟΙΗΣΗ

Για την κατασκευή του νευρωνικού δικτύου αυτοκωδικοποίησης πήραμε σαν βάση τον κώδικα της δευτερης εργασίας τον οποίο τον τροποποιήσαμε καταλλήλα έτσι ώστε να προσθέσουμε μία ενδιαμεση αναπάρσταση. Ο autoencoder έχει την παρακάτω μορφή

Out[14]:

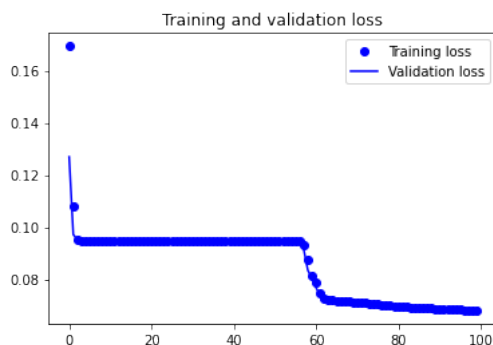


Χρησιμοποιήσαμε αυτή την μορφή νευρωνικού δικτύου αυτοκωδικοποίησης γιατί σύμφωνα με τα πειράματά μας είχε το λιγότερο loss. Αρχικά εφορμόσαμε 3 convolution layers στον encoder. Στην συνέχεια περάσαμε ένα flatten layer και ένα dense layer για να φτάσουμε στο bottleneck του autoencoder, όπου έχουμε την μεγαλύτερη συμπίεση πληροφορίας. Για την αποσυμπίεση περάσαμε από ένα dense layer, ένα reshape και 3 deconvolution layers. Αφού εκπαιδεύσαμε το νευρωνικό μας δίκτυο τότε έχουμε την δυνατότητα να πάρουμε τα συμπιεσμένα δεδομένα από την ενδιάμεση αναπαράσταση. Τα δεδομένα αυτά όμως πρέπει να κανονικοποιηθούν στο διαστήμα 0-25500 με ακέραιες τιμές. Η διαδικασία αυτή γίνεται με την βοήθεια της συνάρτησης normalize που μας μετράει τις δεκαδικές τιμές σε ακέραιες στο διαστήμα 0-25500

2.2 ΠΕΙΡΑΜΑΤΑ - REPORTS

• Πείραμα εκπαίδευσης 1

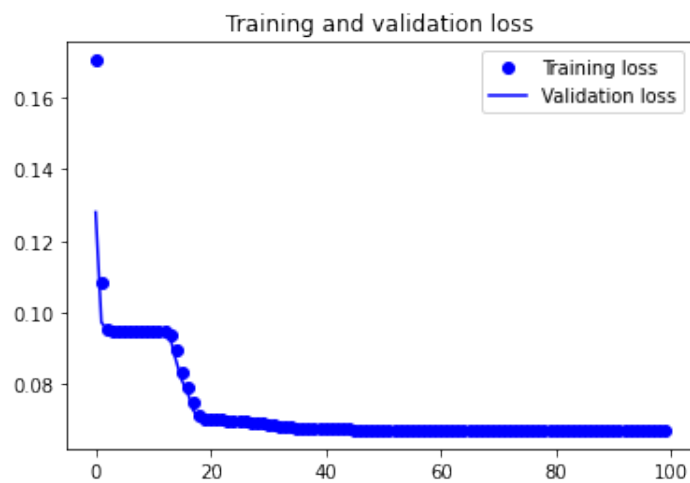
Batch size	128
Epochs	100
Encoder layers	3 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128
Bottleneck size	10
Decoder layers	3 layers 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Softmax και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



Loss: **0.0681** - Validation loss: **0.0682**

- Πείραμα εκπαίδευσης 2

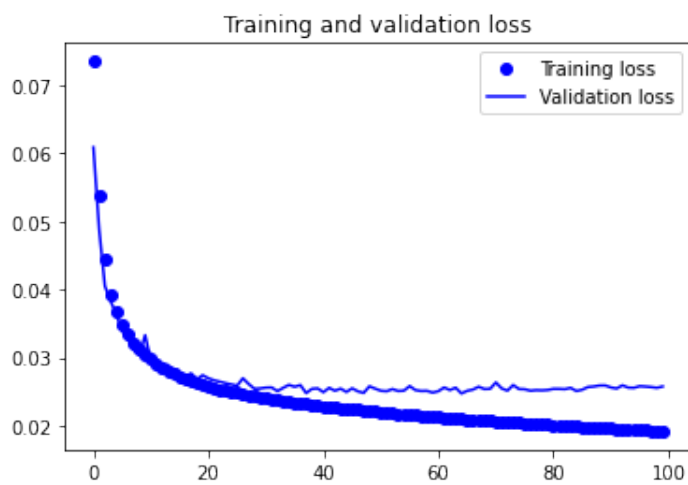
Batch size	256
Epochs	100
Encoder layers	4 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128 1 x 1 x 256
Bottleneck size	10
Decoder layers	4 layers 3 x 3 x 128 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Softmax και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



Loss: **0.0657** - Validation loss: **0.0678**

- Πείραμα εκπαίδευσης 3

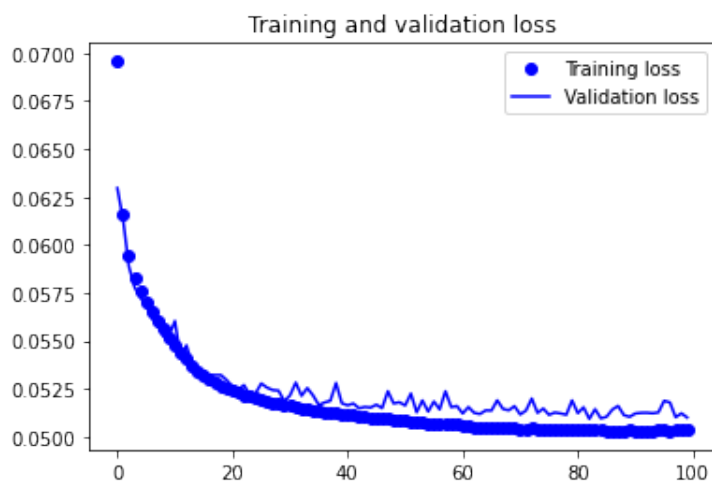
Batch size	256
Epochs	100
Encoder layers	4 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128 1 x 1 x 256
Bottleneck size	10
Decoder layers	4 layers 3 x 3 x 128 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Relu και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



Loss: **0.0190** - Validation loss: **0.0258**

- Πείραμα εκπαίδευσης 4

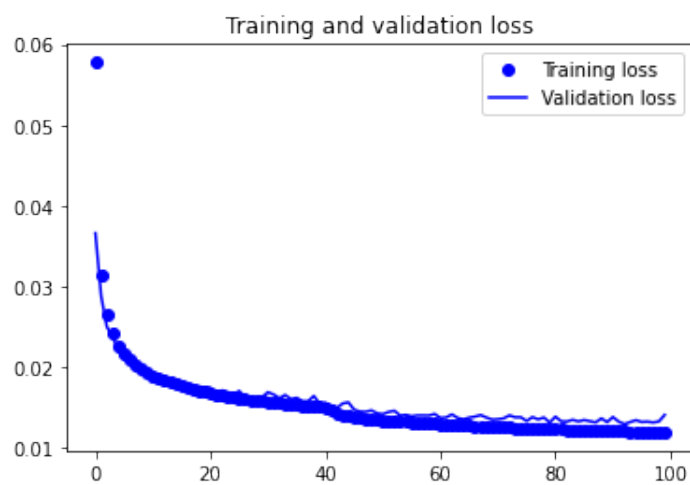
Batch size	128
Epochs	100
Encoder layers	3 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128
Bottleneck size	1
Decoder layers	3 layers 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Relu και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



Loss: **0.0502** - Validation loss: **0.0510**

- Πείραμα εκπαίδευσης 5

Batch size	128
Epochs	100
Encoder layers	3 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128
Bottleneck size	10
Decoder layers	3 layers 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Relu και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



Loss: **0.0119** - Validation loss: **0.0141**

ΜΕΡΟΣ Β' LSH

3.1 ΥΛΟΠΟΙΗΣΗ

Στο μέρος Β' της εργασίας είχαμε να τροποποιήσουμε το παραδοτέο της πρώτης άσκησης με σκοπό να δέχεται τα συμπιεσμένα δεδομένα για να κάνει αναζήτηση στον νέο διανυσματικό χώρο με σκοπό να βρεί τον πλησιέστερο γείτονα. Αρχικά το δημιουργήσαμε την συνάρτηση `ReadDataCompressed` η οποία είναι παρόμοια με την συνάρτηση `ReadData` η οποία στην πρώτη άσκηση διαβαζε τα δεδομένα. Η διαφορά της καινούριας συνάρτησης είναι ότι διαβάζει τα δεδομένα με την καινούρια μορφή του που μας δίνει ο autoencoder του πρώτου ερωτήματος. Εκτός από αυτή την συνάρτηση προσθήσαμε και το κλάσμα προσεγγίσης (Approximation Factor) που μας τη μέση απόσταση Manhattan προσεγγιστικού δια του πραγματικού πλησιέστερου γείτονα από το διάνυσμα επερώτησης στον αρχικό χώρο.

3.2 ΠΕΙΡΑΜΑΤΑ - REPORTS

Αφού κάναμε τις αλλαγές στο παραδοτέο της πρώτης άσκησης κάναμε διαφορα πειράματα και μετρήσεις για να αντιληφθούμε τις διαφορές που προκύπτουν με την συμπίεση των δεδομένων. Παρόλο που η άσκηση ζητούσε να προσαρμόσουμε τον αριθμό πλησιέστερων γειτόνων που αναζητούνται N σε 1, κάναμε πειράματα με μεγαλύτερα N για να δούμε αν έχει αισθητή διαφορά στον χρόνο εκτέλεσης του αλγορίθμου.

- **Για $N = 100$**

- Χρόνος LSH με συμπιεσμένα δεδομένα:0.56058
- Χρόνος brute force με συμπιεσμένα δεδομένα:0.59
- Χρόνος LSH με αρχικά δεδομένα:6.8085
- Χρόνος brute force με αρχικά δεδομένα:7.9102

- **Για $N = 10$**

- Χρόνος LSH με συμπιεσμένα δεδομένα:0.56502
- Χρόνος brute force με συμπιεσμένα δεδομένα:0.60
- Χρόνος LSH με αρχικά δεδομένα:7.1085
- Χρόνος brute force με αρχικά δεδομένα:8.1102

- **Για $N = 1$**

- Χρόνος LSH με συμπιεσμένα δεδομένα:0.55244
- Χρόνος brute force με συμπιεσμένα δεδομένα:0.59909
- Χρόνος LSH με αρχικά δεδομένα:6.0255
- Χρόνος brute force με αρχικά δεδομένα:7.771

Με παραπάνο αποτελέσματα παρατηρήσαμε ότι ο χρόνος εκτέλεσης δεν επηρεάζεται αισθητά από το πλήθος του N . Στην συνέχεια παρατηρήσαμε τα αποτελέσματα στο κλάσμα προσέγγισης και βρήκαμε τα εξής.

- **Approximation Factor for Reduced: ???**

- **Approximation Factor for LSH: ???**

Το αποτέλεσμα του Reduced πιστεύουμε είναι χειρότερο επειδή έχουμε απώλεια πληροφορίας για τον λόγο ότι μειώθηκαν η διαστάσεις και έχουμε χάσει ακρίβεια

4

ΜΕΡΟΣ Γ' EARTH MOVER'S DISTANCE (EMD)

4.1 ΥΛΟΠΟΙΗΣΗ

4.2 ΠΕΙΡΑΜΑΤΑ - REPORTS

5

ΜΕΡΟΣ Δ' CLUSTERING

5.1 ΥΛΟΠΟΙΗΣΗ

5.2 ΠΕΙΡΑΜΑΤΑ - REPORTS