



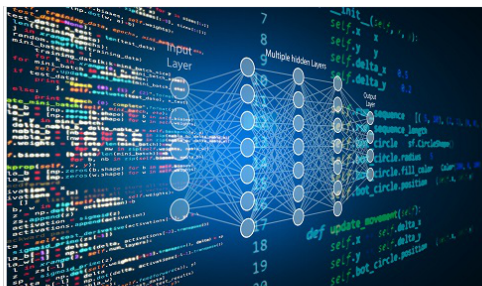
ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών
— ΙΔΡΥΘΕΝ ΤΟ 1837 —

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	13
5.1	ΥΛΟΠΟΙΗΣΗ	13
5.2	ΠΕΙΡΑΜΑΤΑ - REPORTS	14

ΕΙΣΑΓΩΓΗ

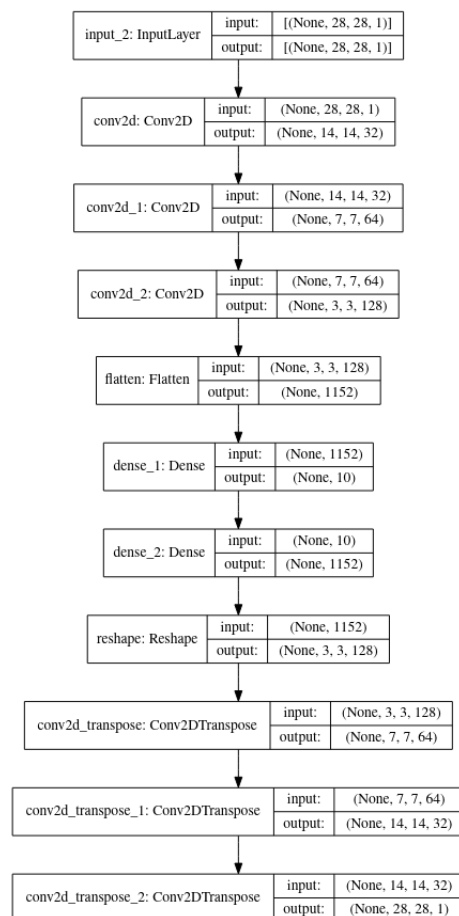
Σε αυτή την εργασία κληθήκαμε να εξάγουμε τα συμπιεσμένα δεδομένα από το ένα ένα νευρωνικό δίκτυο αποκωδικοποίησης ψηφιακών εικόνων. Στην συνέχεια αυτά τα δεδομένα τα χρησιμοποιήσαμε για κάνουμε συσταδοποίηση και να συγκρίνουμε τα αποτελέσματα με τον αρχικό χώρο. Επίσης είχαμε να υλοποιήσουμε τη μετρική Earth Movers 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 δεύτερης 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 αποκωδικοποίησης πήραμε σαν βάση τον κώδικα της δεύτερης εργασίας τον οποίο τον τροποποιήσαμε κατάλληλα έτσι ώστε να προσθέσουμε μία ενδιάμεση αναπαράστασή. Ο autoencoder έχει την παρακάτω μορφή

Out[14]:

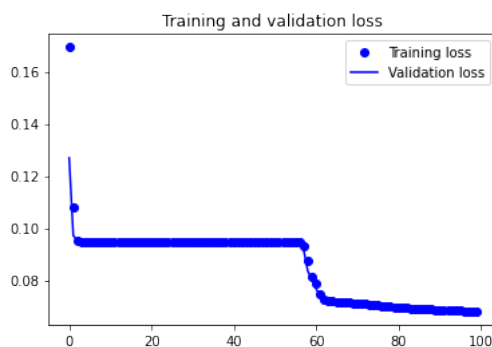


Χρησιμοποιήσαμε αυτή την μορφή νευρωνικού δικτύου αποκωδικοποίησης γιατί σύμφωνα με τα πειράματά μας είχε το λιγότερο loss. Αρχικά εφαρμόσαμε 3 convolution layers στον encoder. Στην συνέχεια περάσαμε ένα flatten layer και ένα dense layer για να φτάσουμε στο bottleneck του autoencoder, όπου έχουμε την μεγαλύτερη συμπίεση πληροφορίας. Για την αποσυμπίεση περάσαμε από ένα dense layer, ένα reshape και 3 deconvolution layers. Αφού εκπαιδύσαμε 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 διάστημα 0-25500 με ακέραιες τιμές. Η διαδικασία αυτή γίνεται με την βοήθεια της συνάρτησης normalize που μας μετατρέπει τις δεκαδικές τιμές σε ακέραιες στο διάστημα 0-25500

2.2 ΠΕΙΡΑΜΑΤΑ - REPORTS

• Πείραμα εκπαίδευσης 1

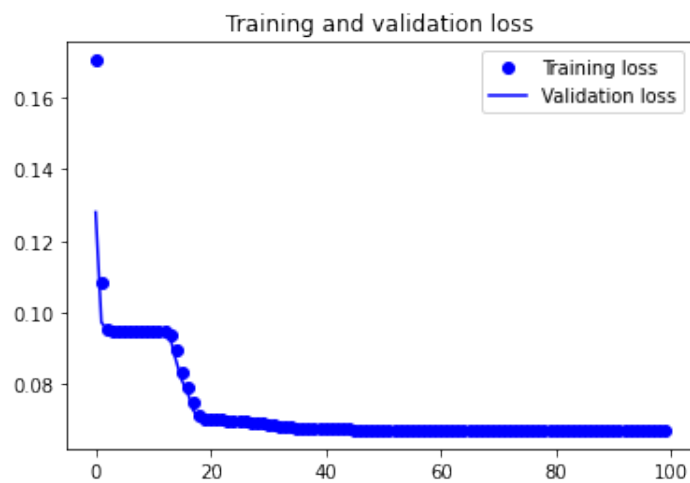
Batch size	128
Epochs	100
Encoder layers	3 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128
Bottleneck size	10
Decoder layers	3 layers 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Softmax και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



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

- Πείραμα εκπαίδευσης 2

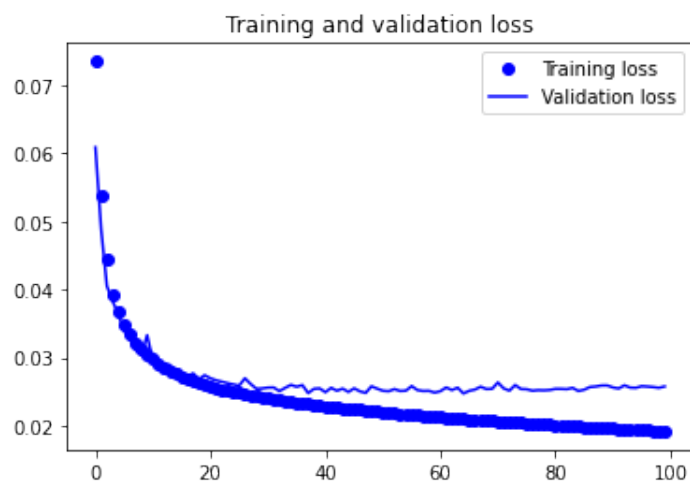
Batch size	256
Epochs	100
Encoder layers	4 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128 1 x 1 x 256
Bottleneck size	10
Decoder layers	4 layers 3 x 3 x 128 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Softmax και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



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

- Πείραμα εκπαίδευσης 3

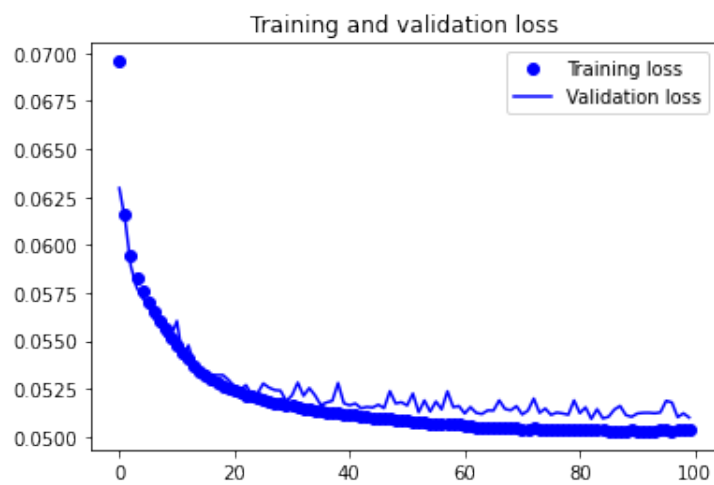
Batch size	256
Epochs	100
Encoder layers	4 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128 1 x 1 x 256
Bottleneck size	10
Decoder layers	4 layers 3 x 3 x 128 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Relu και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



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

- Πείραμα εκπαίδευσης 4

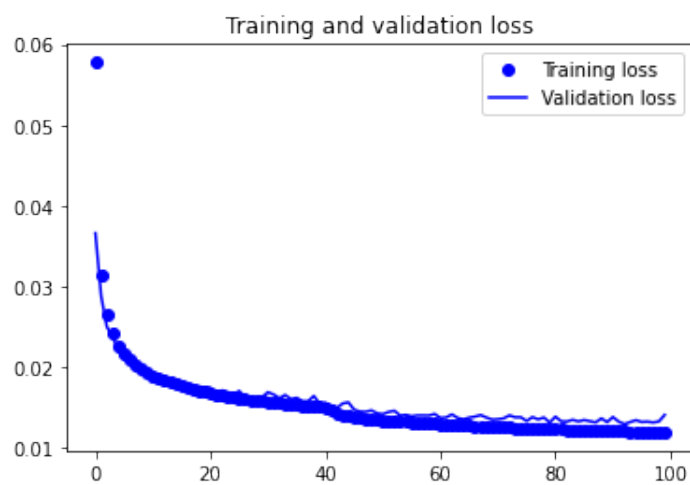
Batch size	128
Epochs	100
Encoder layers	3 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128
Bottleneck size	1
Decoder layers	3 layers 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Relu και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



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:** 1.2818

- **Approximation Factor for LSH:** 1.1197

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

ΜΕΡΟΣ Γ' EARTH MOVER'S DISTANCE (EMD)

4.1 ΥΛΟΠΟΙΗΣΗ

Στο Γ' μέρος της εργασίας είχαμε να υλοποιήσουμε την μετρική Earth Mover's Distance που ανάγεται σε επίλυση προβλήματος Γραμμικού Προγραμματισμού. Αρχικά έπρεπε να δημιουργήσουμε clusters υποδιερόντας τις εικόνες σε μικρότερα κομμάτια. Αυτό το κάναμε με την βοήθεια της συνάρτησης **getImgsCluster** οπού πέρνει σαν όρισμα ένα vector απο vectors με όλες τις εικόνες και στην συνέχεια τις σπάζει σε μικρότερα clusters των 7x7 pixels για να επιστέψει δυο vectors. Το ένα περιέχει τα βάρη των εικόνων ενώ το άλλο τα κεντρικά σημεία του κάθε cluster της εικόνας. Το βάρος ενός cluster ορίζετε σαν το αθροισμα όλων των pixels τις εικόνας. Ενώ το κεντρικό σημείο είναι το κεντρικό pixel του κάθε cluster. Επιλέξαμε για αριθμός των clusters 4x4 επειδή αν μεγαλώσουμε τον αριθμό των clusters θα αυξηθεί η ακριβεία αλλά παραλληλα και η πολυπλοκότητα οποτε προτιμήσαμε 4x4 clusters.

4.2 ΠΕΙΡΑΜΑΤΑ - REPORTS

Τρέχοντας το πρόγραμμα πήραμε τα παρακάτω αποτελέσματα σε dataset με 100 εικόνες για train και 10 εικόνες για query.

Average Correct Search Results EMD:0.4

Average Correct Search Results MANHATTAN:0.47

Επίσης κάναμε πειράματα και με διαφορετικά μεγέθη cluster

- **Για cluster 7 x 7**

- Χρόνος EMD:100.734

- Χρόνος MANHATTAN:4.345

- **Για cluster 4 x 4**

- Χρόνος EMD:20.145

- Χρόνος MANHATTAN:0.43

Αυτό που παρατηρήσαμε είναι ο χρόνος που χρειάζεστε για να τρέξει ο EMD είναι πολυ περισσότερος από τον χρόνο που χρειάζεστε ο αλγόριθμος με μετρική Manhattan. Επίσης όσο μεγαλώνουμε τον αριθμό των clusters τόσο μεγαλώνει και ο χρόνος εκτέλεσης. Αυτό συμβέει γιατί αυξάνετε το πλήθος των υπολογίσεων. Όμως έχουμε περισσότερη ακρίβεια για τον λόγο ότι χάνετε λιγότερη πληροφορία

5

ΜΕΡΟΣ Δ' CLUSTERING

5.1 ΥΛΟΠΟΙΗΣΗ

Για το τελευταίο κομμάτι της εργασίας είχαμε να κάνουμε συσταδοποίηση των εικόνων του συνόλου εισόδου στον νέο χώρο, στον αρχικό χώρο και σε ένα χώρο μετά την συσταδοποίηση απο το παραδοτέο της δεύτερης εργασίας. Οπότε τροποποιήσαμε το

παραδοτέο της πρώτης εργασίας με σκοπό να δέχετε τα συμπιεσμένα δεδομένα των εικόνων.

5.2 ΠΕΙΡΑΜΑΤΑ - REPORTS

- **Για cluster με 1000 εικόνες**

NEW SPACE Clustering time:0.136282

NEW SPACE Silhouette:[-0.313122 0.178616 0.389752 0.366613 0.557391
0.316433 0.211924 0.0513069 0.398375 0.162362 2.82425]

ORIGINAL SPACE Clustering time:1.57707

ORIGINAL SPACE Silhouette:[-0.134824 -0.106549 -0.0460078 0.101237
0.686879 0.303492 0.539355 0.305419 0.431233 0.744012 2.82425]

- **Για cluster με 2000 εικόνες**

NEW SPACE Clustering time:0.409637

NEW SPACE Silhouette:[0.195591 -0.144514 0.230702 0.109205 0.264016
0.325896 0.30479 0.0777447 0.184329 0.147548 3.52675]

ORIGINAL SPACE Clustering time:2.78626

ORIGINAL SPACE Silhouette:[0.774161 0.326452 0.523682 0.299679
0.0597106 0.606904 -0.575428 0.266267 0.44984 0.795483 3.52675]

Παρατηρούμε ότι clustering στον καινούριο έχει μικροτερο χρόνο εκτέλεσης αλλα όχι τόσα καλά αποτελέσματα όσο ο αρχικός χώρος. Αυτό συμβένει γιατί έχουμε απώλεια πληροφορία στον νέο χώρο απο συμπίεση