ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών
ΙΔΡΥΘΕΝ ΤΟ 1837

Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα

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

Ακαδημαικη 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 13

1

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 χρησιμοιποιήσαμε για κάνουμε συσταδοποίση και να συγκρίνουνε τα αποτελέσματα με τον αρχικό χώρο. Επίσης είχαμε να υλοποιήσοτνε τη μετρική Earth Mover's Distance που ανάγεται σε επίλυση προβλήματος Γραμμικού Προγραμματισμού για να ξανακανουμε συστασταδοποιση και να συγκρίνουμε τον χρόνο και την ορθότητα των αποτελέσματων.Τέλος είχαμε να κάνουμε συσταδοποίση K-medians των εικόνων στον παλίο και τον καινουριό καθός και σύσταδοποίση 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.

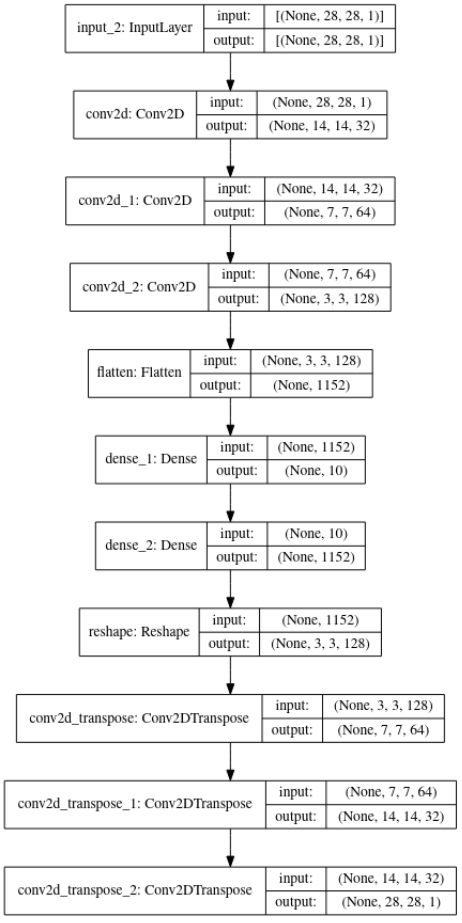
2

ΜΕΡΟΣ Α' AUTOENCODER

2.1 ΥΛΟΠΟΙΗΣΗ

Για την κατασκευή του νευρωνικού δίκτυου αυτοκωδικοποίησης πήραμε σαν βάση τον κώδικα της δευτερης εργασίας τον οποίο τον τροποποοίσαμε καταλληλα έτσι ώστα να προσθέσουμε μία ενδίαμεση αναπάράσταση. Ο autoencoder έχει την παρακάτω μορφή

Out[14]:

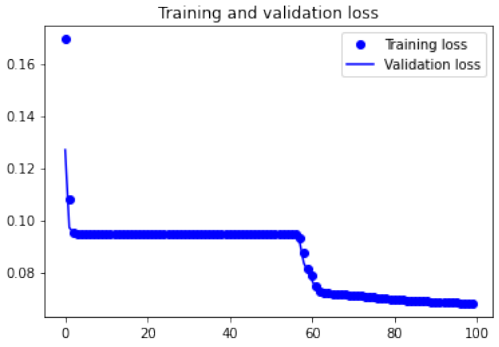


Χρησημοποίσαμε αυτή την μορφή νευρωνικού δίκτυου αυτοκωδικοποίησης γίατί σύμφωνα με τα πειράματα μας είχε το λιγότερο loss. Αρχικά εφορμόσαμε 3 convolution layers στον encoder. Στην συνέχεια περάσαμε ένα flattern layer και ένα dense layer για να φτάσουμε στο bottleneck του autoencoder, όπου έχουμε την μεγαλύτερη συμπίεση πλήροφορίας. Για την αποσυμπίεση περάσαμε απο ενα dense layer, ένα reshape και 3 deconvolution layers. Αφού εκπεδεύσαμε το νευρωνικό μας δικτίο τότε έχουμε την δυνατότα να πάρουμε τα συμπιεσμένα δεδομένα από την ενδιάμεση αναπαράσταση. Τα δεδομένα αυτά ομως πρέπει να κανονικοποιηθουν στο διαστημα 0-25500 με ακεραιες τιμές. Η διαδικασία αυτή γίνετε με την βοήθεια της συνάρτησης normalize που μας μετρατρέπει τις δεκαδικές τιμές σε ακερεες στο διαστημα 0-25500

2.2 ΠΕΙΡΑΜΑΤΑ - REPORTS

- **Πείραμα εκπαίδευσης 1**

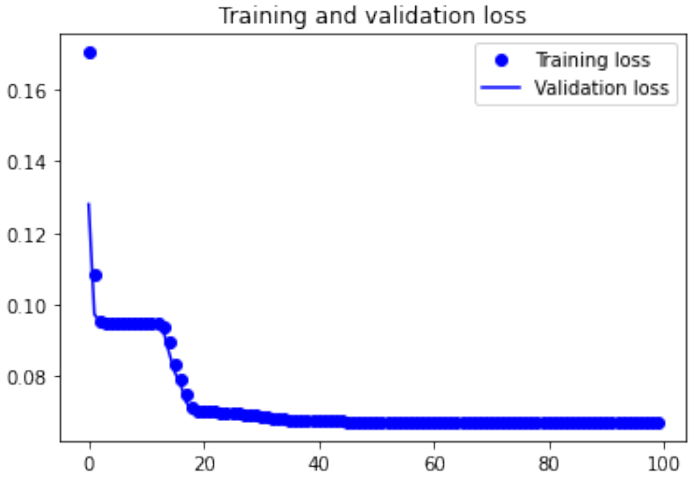
Batch size	128
Epochs	100
Encoder layers	3 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128
Bottleneck size	10
Decoder layers	3 layers 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Softmax και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



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

- **Πείραμα εκπαίδευσης 2**

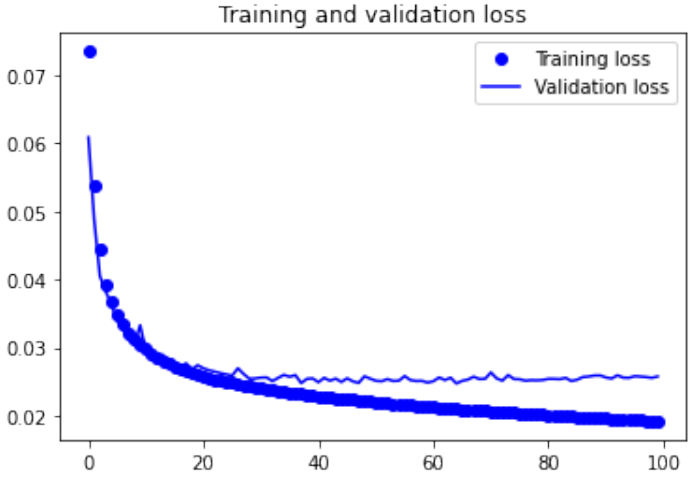
Batch size	256
Epochs	100
Encoder layers	4 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128 1 x 1 x 256
Bottleneck size	10
Decoder layers	4 layers 3 x 3 x 128 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Softmax και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



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

- **Πείραμα εκπαίδευσης 3**

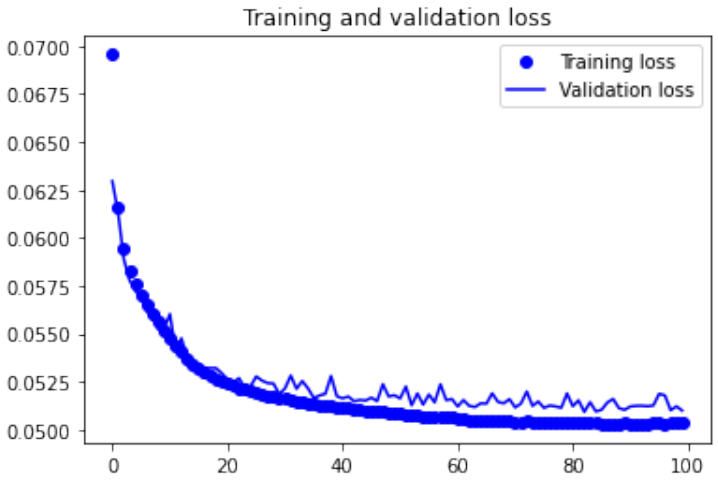
Batch size	256
Epochs	100
Encoder layers	4 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128 1 x 1 x 256
Bottleneck size	10
Decoder layers	4 layers 3 x 3 x 128 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Relu και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



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

- **Πείραμα εκπαίδευσης 4**

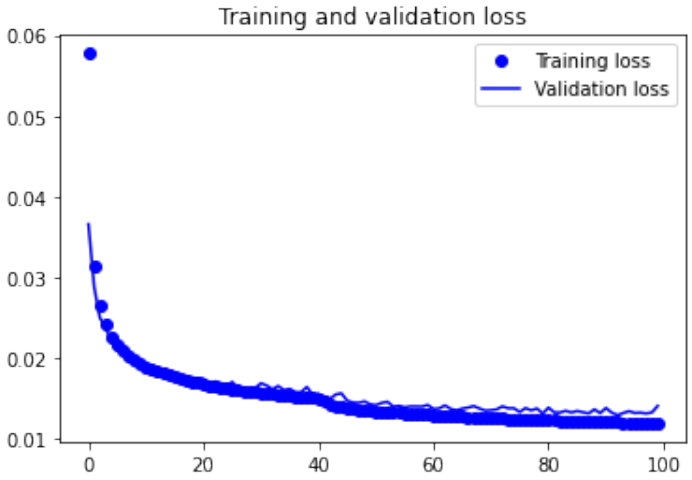
Batch size	128
Epochs	100
Encoder layers	3 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128
Bottleneck size	1
Decoder layers	3 layers 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Relu και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



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**

3

ΜΕΡΟΣ Β' LSH

3.1 ΥΛΟΠΟΙΗΣΗ

Στο μέρος Β' της εργασίας είχαμε να τροποιποίσουμε το παραδοτέο της πρώτης άσκσης με σκοπό να δέχετε 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 ασκηση διαβαζε τα δεδομένα. Η διαφορά της καινούριας συνάρτησης είναι ότι διαβάζει τα δεδομένα με την καινουρια μοφρή του που μας δίνει ο autoencoder του πρώτου ερωτήματος. Εκτός από αυτή την συνάρτηση προσθέσαμε και το κλάσμα προσεγγισης (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 άσκσης κάναμε διαφορα πειράματα και μετρήσεις για να αντιληφθούμε τις διαφορές που προκύπτουν με την συμπιέση των δεδομένων. Παρόλο που η άσκηση ζητούσε να προσαρμόσουμε τον αριθμό πλησιέστερων γειτόνων που αναζητούνται N σε 1, κάναμε πειράμετα με μεγαλήτερα 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 παρατηρήσμαε ότι ο χρόνος εκτέλεσης δεν επηρεάζετε αισθήτα απο το πληθος του N. Στην συνεχεία παρατηρήσαμε τα αποτέλεσμα στο κλάσμα προσέγγίσης και βρήκαμε τα εξής.

- **Approximation Factor for Reduced:** 1.2818
- **Approximation Factor for LSH:** 1.1197

Το αποτέλεσμα του Reduced πιστεύουμε είναι χείροτερο επειδή έχουμε απόλεια πληροφορίας για τον λόγο ότι μειώθεικαν η διαστάσεις και έχοουμε χάσει αρκίβεια

4

ΜΕΡΟΣ Γ' Earth Mover's Distance (EMD)

4.1 ΥΛΟΠΟΙΗΣΗ

Στο Γ' μέρος της εργασίας είχαμε να υλοποίσομε την μετρικη Earth Mover's Distance που ανάγεται σε επίλυση προβλήματος Γραμμικού Προγραμματισμού. Αρχικά έπρεπε να δημιουργήσουμε clusters υποδιερόντας τις εικόνες σε μικρότερα κομματια. 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 είκώνων ενώ το αλλο 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 των αριθμό των clusters θα αυξηθει η ακριβεια αλλα παραλληλα και η πολυπλοκότητα οποτε προτιμίσαμε 4x4 clusters.

4.2 ΠΕΙΡΑΜΑΤΑ - REPORTS

Τρέχοντας το πρόγραμα πήραμε τα παρακάτω αποτελέμσατα σε dataset με 100 εικόνες για train και 10 είκόνες για query.

Average Correct Search Results EMD:0.4

Average Correct Search Results MANHATTAN:0.47

Επίσης κάναμε πειράματε 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 χρειάζετε για να τρέξει ο EMD είναι πολυ περισσοτερος από τον χρόνο που χρειάζετε ο αλγόριθμος με μετρική Manhattan. Επίσης όσο μεγαλόνουμε τον αριθμό των clusters τόσο μεγαλώνει και ο χρόνος εκτέλεσης. Αυτό συμβένει γιατί αυξάνετε το πλήθος των υπολογίσμων. Όμως έχουμε περισσοτέρη ακρίβεια για τον λόγο ότι χάνετε λιγότερη πληροφορία

5

ΜΕΡΟΣ Δ' CLUSTERING

5.1 ΥΛΟΠΟΙΗΣΗ

5.2 ΠΕΙΡΑΜΑΤΑ - REPORTS