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

Σε αυτή την εργασία κληθήκαμε να εξάγουμε τα συμπιεσμένα δεδομένα από το ένα νευρωνικό δίκτυο αποκωδικοποίησης ψηφιακών εικόνων. Στη συνέχεια αυτά τα δεδομένα τα χρησιμοποιήσαμε για να 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 παλιό και τον καινούριο χώρο ,καθώς και σύσταδοποίηση βάσει 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.

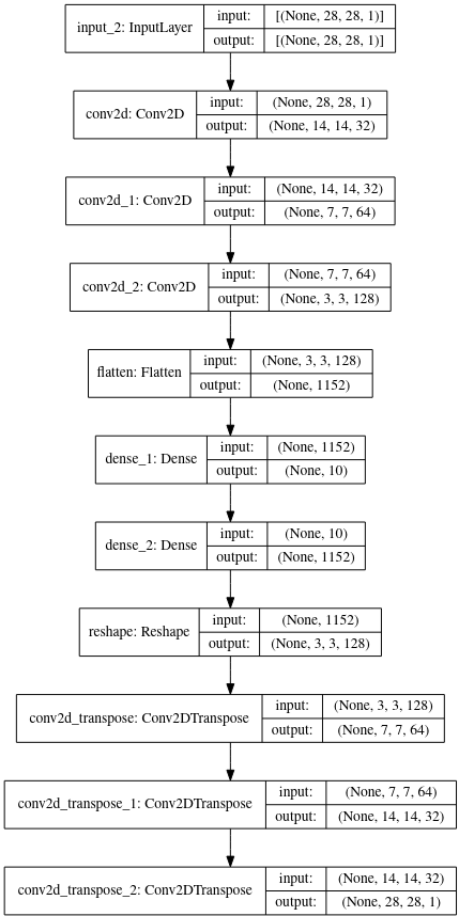
2

ΜΕΡΟΣ Α' AUTOENCODER

2.1 ΥΛΟΠΟΙΗΣΗ

Για την κατασκευή του νευρωνικού δικτύου αποκωδικοποίησης πήραμε ως βάση τον κώδικα της δεύτερης εργασίας τον οποίο τον τροποποιήσαμε κατάλληλα ,έτσι ώστε να προσθέσουμε μία ενδιάμεση αναπαράσταση. Ο autoencoder έχει την παρακάτω μορφή

Out[14]:

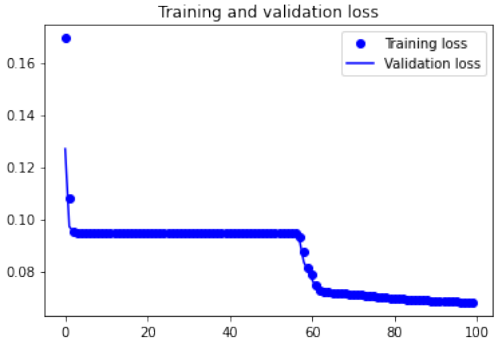


Χρησημοποιήσαμε αυτή τη μορφή νευρωνικού δικτύου αποκωδικοποίησης γιατί σύμφωνα με τα πειράματά μας είχε το λιγότερο loss. Αρχικά ,εφορμόσαμε 3 convolution layers στον encoder. Στη συνέχεια περάσαμε ένα flattern layer και ένα dense layer για να φτάσουμε στο bottleneck του autoencoder, όπου έχουμε τη 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 τη δυνατότα 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 τη βοήθεια της συνάρτησης normalize που μας μετατρέπει τις δεκαδικές τιμές σε ακέραιες στο διάστημα 0-25500

2.2 ΠΕΙΡΑΜΑΤΑ - REPORTS

- **Πείραμα εκπαίδευσης 1**

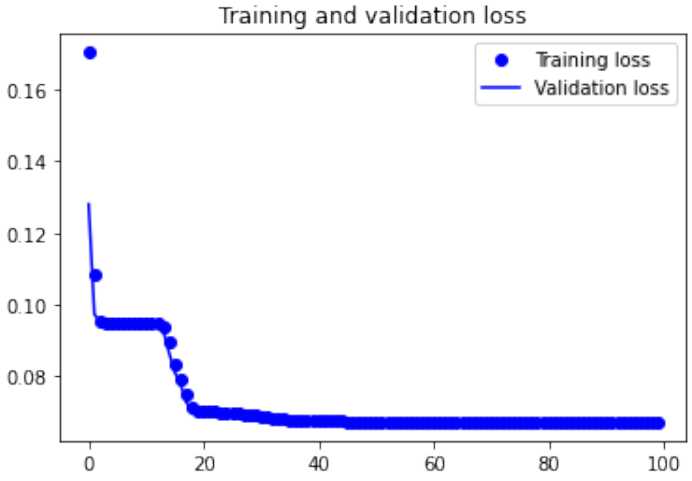
Batch size	128
Epochs	100
Encoder layers	3 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128
Bottleneck size	10
Decoder layers	3 layers 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Softmax και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



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

- **Πείραμα εκπαίδευσης 2**

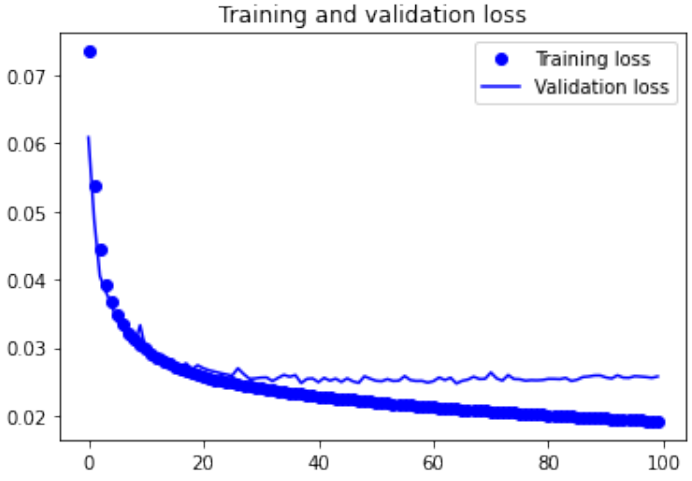
Batch size	256
Epochs	100
Encoder layers	4 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128 1 x 1 x 256
Bottleneck size	10
Decoder layers	4 layers 3 x 3 x 128 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Softmax και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



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

- **Πείραμα εκπαίδευσης 3**

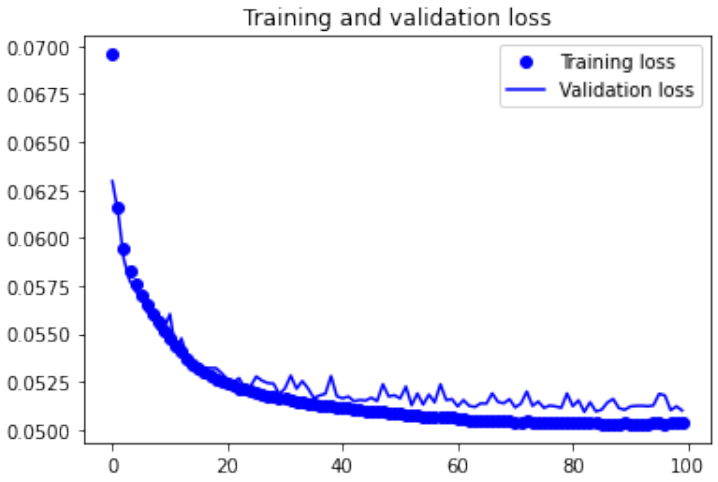
Batch size	256
Epochs	100
Encoder layers	4 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128 1 x 1 x 256
Bottleneck size	10
Decoder layers	4 layers 3 x 3 x 128 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Relu και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



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

- **Πείραμα εκπαίδευσης 4**

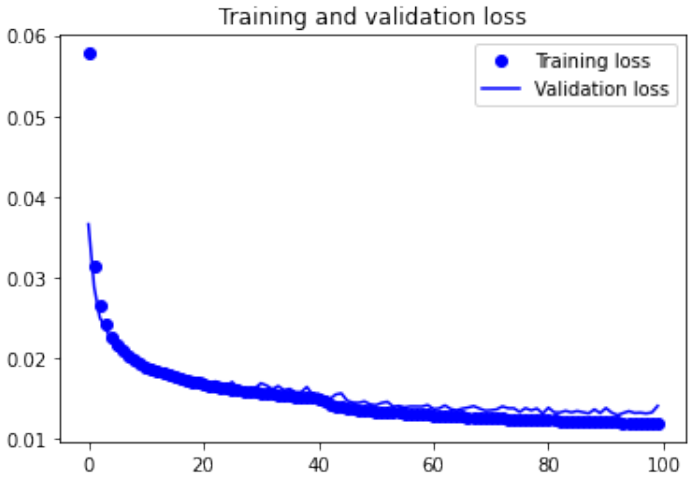
Batch size	128
Epochs	100
Encoder layers	3 layers 14 x 14 x 32 7 x 7 x 64 3 x 3 x 128
Bottleneck size	1
Decoder layers	3 layers 7 x 7 x 64 14 x 14 x 32 28 x 28 x 1
Activation Function	Relu και sigmoid στο εξωτερικό layer
Loss function	Mean squared error
Optimizer	RMSprop



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**

3

ΜΕΡΟΣ Β' LSH

3.1 ΥΛΟΠΟΙΗΣΗ

Στο μέρος Β' της εργασίας είχαμε να τροποιποιήσουμε το παραδοτέο της πρώτης άσκησης με σκοπό να δέχεται τα συμπιεσμένα δεδομένα για να κάνει αναζήτηση στον νέο διανυσματικό χώρο με σκοπό να βρει τον πλησιέστερο γείτονα. Αρχικά, δημιουργήσαμε τη συνάρτηση ReadDataCompressed η οποία είναι παρόμοια με τη συνάρτηση ReadData που στην πρώτη άσκηση διάβαζε τα δεδομένα. Η διαφορά της καινούριας συνάρτησης είναι ότι διαβάζει τα δεδομένα με την καινούρια μορφή που μας δίνει ο autoencoder του πρώτου ερωτήματος. Εκτός από αυτή τη συνάρτηση προσθέσαμε και το κλάσμα προσέγγισης (Approximation Factor) που ορίζει τη μέση απόσταση Manhattan προσεγγιστικού διά του πραγματικού πλησιέστερου γείτονα από το διάνυσμα επερώτησης στον αρχικό χώρο.
Η μεταγλώττιση γίνετε με την εντολή **make search**

3.2 ΠΕΙΡΑΜΑΤΑ - REPORTS

Αφού κάναμε τις αλλαγές στο παραδοτέο της πρώτης άσκησης ,κάναμε διάφορα πειράματα και μετρήσεις για να αντιληφθούμε τις διαφορές που προκύπτουν με τη 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 τα πιο πάνω αποτελέσματα παρατηρήσαμε ότι ο χρόνος εκτέλεσης δεν επηρεάζεται αισθητά από το πλήθος του N. Στη 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 δεν είναι τόσο καλό, όσο το αποτέλεσμα του LSH, επειδή υπάρχει απώλεια πληροφορίας λόγω του ότι μειώθηκαν οι διαστάσεις και έχουμε χάσει αρκίβεια στον καινούριο χώρο

4

ΜΕΡΟΣ Γ' Earth Mover's Distance (EMD)

4.1 ΥΛΟΠΟΙΗΣΗ

Στο Γ' μέρος της εργασίας είχαμε να υλοποιήομε τη μετρική Earth Mover's Distance που ανάγεται σε επίλυση προβλήματος Γραμμικού Προγραμματισμού. Αρχικά έπρεπε να δημιουργήσουμε clusters υποδιαιρώντας τις εικόνες σε μικρότερα κομμάτια. Αυτό το κάναμε με τη βοήθεια της συνάρτησης **getImgsCluster** όπου παίρνει ως όρισμα ένα vector από vectors με όλες τις εικόνες και στη 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 ορίζεται σαν το άθροισμα όλων των pixels της εικόνας. Ενώ το κεντρικό σημείο ,είναι το κεντρικό pixel του κάθε cluster. Επιλέξαμε για αριθμούς των clusters 4x4, επειδή αν μεγαλώσουμε τον αριθμό των clusters, θα αυξηθεί η ακρίβεια αλλά παράλληλα και η πολυπλοκότητα, οπότε προτιμίσαμε 4x4 clusters.
Η μεταγλώττιση γίνετε με την εντολή **make search**

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 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 χρειάζεται για να τρέξει ο EMD είναι πολύ περισσότερος από τον χρόνο που χρειάζεται ο αλγόριθμος με μετρική Manhattan. Επίσης όσο μεγαλώνουμε τον αριθμό των clusters τόσο μεγαλώνει και ο χρόνος εκτέλεσης. Αυτό συμβαίνει γιατί αυξάνεται το πλήθος των υπολογισμών. Όμως έχουμε περισσοτέρη ακρίβεια για τον λόγο ότι χάνεται 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 τη συσταδοποίηση από το παραδοτέο της δεύτερης εργασίας. Οπότε τροποποιήσαμε το παραδοτέο της πρώτης εργασίας με σκοπό να δέχεται τα συμπιεσμένα δεδομένα των εικόνων.Η μεταγλώττιση γίνετε με την εντολή **make cluster**

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 το clustering στον καινούριο χώρο , έχει μικρότερο χρόνο εκτέλεσης αλλά όχι τόσα καλά αποτελέσματα όσο ο αρχικός χώρος. Αυτό συμβαίνει γιατί έχουμε απώλεια πληροφορίας στον νέο χώρο από συμπίεση