Κ23γ : Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα

Χειμερινο Εξαμηνο 2020-21

3η Προγραμματιστική Εργασία

**Βασιλοπούλου Θωμαΐς (1115201500016)**

**Γεωργακλή Έλενα (1115201500023)**

Ερωτημα Α:

Α1. Δομη Παραδοτεου

Το παραδοτέο για το ερώτημα 3Α αποτελείται από τα ακόλουθα αρχεία:

* reduce.py: το εκτελέσιμο που υλοποιεί το ζητούμενο Α
* results: φάκελος που περιέχει τα γραφικά αποτελέσματα της εκτέλεσης του προηγούμενου αρχείου, καθώς και αρχεία με τα αποτελέσματα των πειραμάτων σε κάθε περίπτωση

A2. Λειτουργικοτητα Εργασιασ

Το εκτελέσιμο τρέχει με την εντολή:

python reduce.py –d train-images.idx3-ubyte -q t10k-images.idx3-ubyte –od outputDataset.txt -oq outputQueryset.txt

Με την εκκίνηση της εκτέλεσης γίνεται έλεγχος για την ορθότητα των ορισμάτων και στην συνέχεια ζητούνται από τον χρήστη οι υπερπαράμετροι για την παρούσα υλοποίηση του autoencoder. Όταν αντληθούν όλες οι απαραίτητες τιμές από το χρήστη, ο έλεγχος μεταβαίνει στην συνάρτηση:

def autoencoderBottleneck(dataset, queryset, layers, maxFilters, x, y, convFiltSize, batchSize, epochs, latentDim)

Πιο αναλυτικά, αντλούνται τα metadata από το αρχείο εισόδου (magic number, number of images, dx, dy), διαβάζονται οι εικόνες, μετατρέπει αυτά που διαβάστηκαν σε numpy array των κατάλληλων διαστάσεων και ορίζει το σχήμα του μοντέλου. Στην συνέχεια δημιουργείται το ζητούμενο μοντέλο με σκοπό να αξιολογηθεί για την απόδοσή του, με μετρική απώλειας mean\_squared\_error. Για την ολοκλήρωση της προηγούμενης διαδικασίας λαμβάνει χώρα η διαδοχική κλήση του encoder και decoder, για τους οποίους ισχύουν τα ακόλουθα:

1. encoder(inputImg, layers, maxFilters = 64, convFiltSize = (3, 3)) ))

Αποτελείται από layers, που δύναται να είναι είτε Conv2D, είτε MaxPooling2D ή BatchNormalization. Σκοπός του encoder είναι η εικόνα που θα χρησιμοποιηθεί στην συνέχεια από τον decoder να έχει ίδιες διαστάσεις με την αρχική. Για τον λόγο αυτό, το MaxPooling2D λαμβάνει χώρα μετά τα δύο πρώτα BatchNormalization και όχι αργότερα.

1. decoder(conv, layers, maxFilters, convFiltSize, dx, dy)

Αποτελέιται από layers, που δύναται να είναι είτε Conv2DTranspose, είτε UpSampling2D ή BatchNormalization. Ο decoder είναι απόλυτα συμμετρικός με τον encoder, το UpSampling2D λαμβάνει χώρα μετά τα δύο τελευταία BatchNormalization και όχι νωρίτερα.

Αξίζει να αναφερθεί ότι τα μοντέλα που παράγονται τόσο από τον encoder, όσο και από τον decoder δεν παρουσιάζουν overfit.

Όταν ο encoder και ο decoder ολοκληρώσουν την εκτέλεσή τους, δημιουργείται input κατάλληλων διαστάσεων και διαχωρίζεται το training set, ώστε το 80% να είναι για train και το 20% να είναι για validation. Στο μοντέλο του autoencoder ως ground truth ορίζεται το ίδιο το σύνολο των εικόνων και κανονικοποιούνται οι τιμές ώστε να είναι στο διάστημα [0.0, 1.0]. Στην συνέχεια εκπαιδεύεται το παραχθέν μοντέλο με βάσει τις δοθείσες από τον χρήστη υπερπαραμέτρους.

Μόλις έχει κατασκευαστεί το επιθυμητό μοντέλο, ξεκινάει η διαδικασία ανάγνωσης του query set από το δοθέν αρχείο. Αρχικά, αντλούνται τα metadata (magic number, number of images, dx, dy), διαβάζονται οι εικόνες και το αποτέλεσμα μετατρέπεται σε numpy array των κατάλληλων διαστάσεων. Τέλος διαχωρίζεται το query set, ώστε το 80% να είναι για train και το 20% να είναι για validation και εκπαιδεύεται το παραχθέν μοντέλο με τις προηγούμενες υπερπαραμέτρους.

Με την ολοκλήρωση του autoencoder, παράγονται οι εικόνες τόσο για το data set, όσο και για το query set στον νέο διανυσματικό χώρο (10x10) από την συνάρτηση:

def reduce(outputDataset, outputQueryset, latentDim, xTrain, xQuery)

A3. Αποτελεσματα Εκτελεσεων

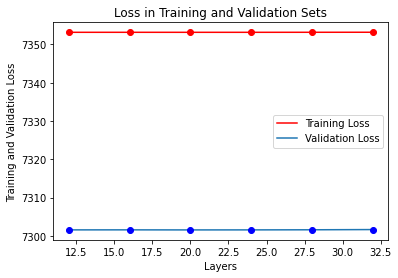
Στην συνέχεια παρατίθενται μετρήσεις κατά τις οποίες, για τον εντοπισμό αλλαγών καθώς διαφοροποιούνται οι υπρεπαράμετροι. Οι προκαθορισμένες τιμές που χρησιμοποιούνται στα πειράματα για τις μεταβλητές είναι οι ακόλουθες:

* *layers = 20*
* *convFiltSize = (3,3)*
* *maxFilterSize = 64*
* *epochs = 50*
* *batchSize = 128*
* *latentDim = 10*

Σε κάθε μία από τις γραφικές παραστάσεις που ακολουθούν γίνεται σύγκριση του σφάλματος που προκύπτει κατά την εκτέλεση του autoencoder με training και validation set. Γενικά είναι σημαντικό να αναφερθεί, ότι το σφάλμα κινούνταν γύρω από το ίδιο εύρος τιμών σε κάθε περίπτωση πειραματισμού για το training set και οι διαταραχές ήταν ορισμένες φορές ευκολότερα ορατές στο σφάλμα του validation set.

Είναι σημαντικό να αναφερθεί, ότι το πλήθος των συνελικτικών στρωμάτων μοιράζεται ισάξια στον encoder και τον decoder. Για παράδειγμα, αν δοθεί πλήθος layers = 12, τότε ο encoder θα χρησιμοποιήσει τα 6 από αυτά και ο decoder τα υπόλοιπα 6. Αξίζει, επίσης, να αναφερθεί, ότι το ελάχιστο πλήθος των συνελικτικών στρωμάτων, για να μην προκύψει υπερχείλιση, είναι 12, δεδομένου, ότι ο encoder και ο decoder χρησιμοποιούν κατ’ ελάχιστο 6 ο κάθε ένας.

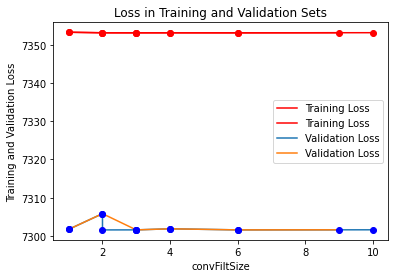
1. Πλήθος Συνελικτικών Στρωμάτων



Εικόνα 1: Διαμόρφωση Σφάλματος για διάφορα πλήθη συνελικτικών στρωμάτων

Παρατηρείται, ότι το πλήθος των συνελικτικών στρωμάτων δεν επηρεάζει σημαντικά την απόδοση του autoencoder, όσον αφορά το σφάλμα. Παρόλα αυτά, όσο αυξάνεται το πλήθος των layer παρατηρείται και μία μικρή αύξηση στο αντίστοιχο σφάλμα και στις δύο περιπτώσεις.

1. Μεγεθος Συνελικτικών Φιλτρων



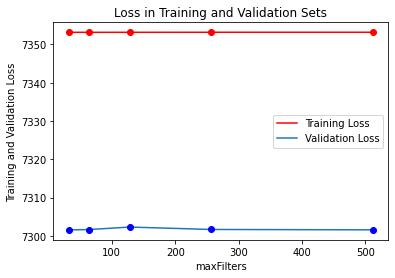
Εικόνα 2: Διαμόρφωση Σφάλματος για διάφορα πλήθη συνελικτικών φίλτρων

Από το παραπάνω διάγραμμα παρατηρείται, ότι η απόδοση του autoencoder δεν επηρεάζεται σημαντικά από το πλήθος των συνελικτικών φίλτρων. Όμως, για φίλτρο διάστασης (2,2) υπάρχει μία αύξηση στο σφάλμα του validation set. Τέλος, αξίζει να αναφερθεί, ότι όσο αυξάνεται η διάσταση του συνελικτικού φίλτρου, τόσο αυξάνεται και ο χρόνος εκτέλεσης του προγράμματος.

Οι διαστάσεις των φίλτρων που χρησιμοποιήθηκαν για την πειραματική μελέτη είναι οι ακόλουθες:

* (1,1)
* (2,2)
* (2,3)
* (3,3)
* (4,4)
* (6,6)
* (10,9)

1. Μέγιστο Πλήθοσ Φίλτρων ανα Στρωμα



Εικόνα 3: Διαμόρφωση Σφάλματος για διάφορες τιμές μέγιστου πλήθους συνελικτικών φίλτρων

Όπως φαίνεται από το παραπάνω διαγράμμα, η αύξηση του μέγιστου πλήθους των συνελικτικών φίλτρων, δεν επηρεάζει την απόδοση του autoencoder ως προς το σφάλμα. Παρόλα αυτά, όσο αυξάνεται το μέγεθος δέσμης, τόσο επιταχύνεται η εκτέλεση του προγράμματος.

1. Μεγεθος Δεσμησ



Εικόνα 4: Διαμόρφωση Σφάλματος για διάφορα μεγέθη δέσμης

Από το παραπάνω διάγραμμα παρατηρείται ότι το σφάλμα του validation set αυξάνεται σημαντικά, όσο αυξάνεται το μέγεθος δέσμης, ενώ το αντίστοιχο σφάλμα για το training set κυμαίνεται γύρω από ένα σταθερό εύρος τιμών.

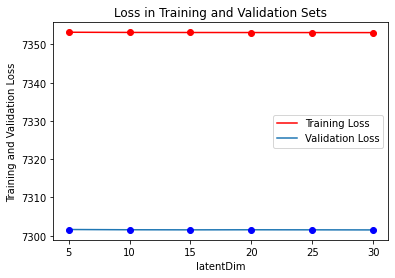
1. Πληθοσ Εποχων Εκπαιδευσησ



Εικόνα 5: Διαμόρφωση Σφάλματος για διάφορες τιμές εποχών εκπαίδευσης

Όπως φαίνεται από το παραπάνω διαγράμμα, η αύξηση του πλήθους των εποχών εκπαίδευσης, δεν επηρεάζει την απόδοση του autoencoder, ως προς το σφάλμα. Παρόλα αυτά, όσο αυξάνεται το πλήθος των εποχών εκπαίδευσης, τόσο καθυστερεί η εκτέλεση του προγράμματος.

1. Μεγεθος Latent Dimension



Εικόνα 6: Διαμόρφωση Σφάλματος για διάφορες τιμές latent dimension

Όπως φαίνεται από το παραπάνω διάγραμμα, η αύξηση του μεγέθους του latent dimension, δεν επηρεάζει την απόδοση του autoencoder, ως προς το σφάλμα.

Ερωτημα B:

B1. Δομη Παραδοτεου

Το παραδοτέο για το ερώτημα 3Β αποτελείται από τα ακόλουθα αρχεία:

* search.cpp : το εκτελέσιμο αρχείο που περιλαμβάνει τη main
* classFuncs.cpp: αρχείο πηγαίου κώδικα που περιλαμβάνει τις υλοποιήσεις όλων των συναρτήσεων που αφορούν τις κλάσεις που χρησιμοποιούνται
* funcHeader.hpp: αρχεία που περιλαμβάνουν δηλώσεις των βοηθητικών συναρτήσεων που χρησιμοποιούνται για την υλοποίηση του ζητούμενου της εργασίας
* genericFuncs.cpp: αρχείο που περιλαμβάνει την υλοποίηση όλων των βοηθητικών συναρτήσεων που χρησιμοποιούνται για την υλοποίηση του ζητούμενου
* header.hpp: αρχείο με τις δηλώσεις όλων των κλάσεων που χρησιμοποιούνται
* makefile: αρχείο με τις εντολές μεταγλώττισης όλων των εκτελέσιμων της εργασίας

Στο παραδοτέο εκτός από τα προαναφερθέντα περιέχονται και δύο αρχεία, το outputDataset.txt και outputQuery.txt για την κλήση του προγράμματος.

Όλα τα εκτελέσιμα αρχεία που αναφέρονται παραπάνω μεταγλωττίζονται με την χρήση της εντολής:

make

Το εκτελέσιμο τρέχει με την εντολή:

./search -d train-images.idx3-ubyte -i outputDataset.txt -q t10k-images.idx3-ubyte -s outputQueryset.txt -k 4 -L 3 -o output.txt

Με την εκκίνηση της εκτέλεσης γίνεται έλεγχος για την ορθότητα των ορισμάτων και αποδίδονται τιμές στις αντίστοιχες μεταβλητές. Στην συνέχεια, αντλούνται τα metadata, από το αρχείο που περιλαμβάνει τις εικόνες στον αρχικό χώρο, διαβάζονται μία προς μία οι εικόνες και εκτελείται ο αλγόριθμος LSH για το αρχικό σύνολο. Μόλις ολοκληρωθεί ο αλγόριθμος, διαβάζονται τα metadata και οι εικόνες του νέου διανυσματικού χώρου, καθώς και τα metadata από τα αρχεία που περιλαμβάνουν τα query sets.

Για κάθε μία από τις εικόνες λαμβάνουν χώρα οι ακόλουθοι υπολογισμοί:

* Απόσταση πλησιέστερου γείτονα στο αρχικό σύνολο και ο χρόνος εύρεσής της.
* Απόσταση πλησιέστερου γείτονα μέσω ANN από το σύνολο που παράχθηκε μέσω LSH και ο χρόνος εύρεσής της.
* Απόσταση πλησιέστερου γείτονα στον νέο περιορισμένης διάστασης διανυσματικό χώρο και ο χρόνος εύρεσής της.
* Συνολικό άθροισμα αποστάσεων που βρέθηκαν μέσω LSH.
* Συνολικό άθροισμα αποστάσεων που βρέθηκαν μέσω NN.
* Συνολικό άθροισμα πραγματικών αποστάσεων.

Σε κάθε επανάληψη τυπώνονται ο αύξοντας αριθμός του query image, οι τρεις διαφορετικές αποστάσεις καθώς και οι χρόνοι υπολογισμού τους.

Μετά την ολοκλήρωση της παραπάνω επαναληπτικής διαδικασίας υπολογίζεται η μέση προσεγγιστική απόσταση ανά περίπτωση και η μέση πραγματική απόσταση από τον πλησιέστερο γείτονα. Το πηλίκο των δύο παραπάνω τελικά συνθέτει το ζητούμενο κλάσμα προσέγγισης. Τα αποτελέσματα της εκτέλεσης, τυπώνονται στο αρχείο εξόδου, όπως υποδεικνύεται στην εκφώνηση.