ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμημα Πληροφορικης και Τηλεπικοινωνιων

Διανυσματική αναπαράσταση εικόνας σε χώρο χαμηλότερης διάστασης με χρήση νευρωνικού δικτύου αυτοκωδικοποίησης

Χατζησπύρου Μιχαήλ – sdi: 1115202000212
Καζάκος Παναγιώτης – sdi: 1115201900067

Κ23Γ: Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα – Χειμερινό '23
Ημερ. Ανακοίνωσης: 15/12
Ημερ. Υποβολής: 12/1 – Ώρα 23:59

Περιεχόμενα

1. Γενική Περιγραφή

Σε αυτήν την εργασία, το κύριο καθήκον μας περιελάμβανε τη δημιουργία ενός αυτόματου κωδικοποιητή για συμπίεση εικόνας, με στόχο να μετατρέψουμε τις εικόνες σε μια νέα και πιο συμπαγή διάσταση. Μόλις ο αυτόματος κωδικοποιητής συμπίεσε με επιτυχία τις εικόνες, η εστίασή μας μετατοπίστηκε στην αξιολόγηση των κατά προσέγγιση αλγορίθμων πλησιέστερου γείτονα, LSH, CUBE, GNNS, MRNG και στον αλγόριθμο BruteForce. Μετά την ανάλυση των αλγορίθμων του πλησιέστερου γείτονα, το επόμενο βήμα μας ήταν να πραγματοποιήσουμε συσταδοποίηση (Clustering) στη νέα διάσταση. Για να αξιολογήσουμε την ποιότητα της ομαδοποίησης, χρησιμοποιήσαμε την μετρική Silhouette, συγκρίνοντας τα αποτελέσματα στη μειωμένη διάσταση με εκείνα της αρχικής διάστασης.

2. Οδηγίες Μεταγλώττισης

Έχει δημιουργηθεί ένα Makefile για την διευκόλυνση της μεταγλώττισης και την εκτέλεση των προγραμμάτων με κάποιες προκαθορισμένες παραμέτρους. Η μεταγλώττιση όλων των αρχείων και γίνεται με την εντολή μακε. Χρησιμοποιείται η τεχνική του separate compilation. Επίσης δίνεται η δυνατότητα στον χρήστη να μεταγλωττίσει το πρόγραμμα με την εντολή:

- make lsh – για την μεταγλώττιση του προγράμματος Lsh
- make cube – για την μεταγλώττιση του προγράμματος Hypercube
- make cluster – για την μεταγλώττιση του προγράμματος Clustering
- make brute – για την μεταγλώττιση του προγράμματος BruteForce
- make graph – για την μεταγλώττιση του προγράμματος Graph

3. Οδηγίες Χρήσης των Προγραμμάτων

Αφού ο χρήστης δημιουργήσει το εκτελέσιμο αρχείο με όποια μέθοδο επιθυμεί μπορεί να το εκτελέσει με την βοήθεια του Makefile χρησιμοποιώντας τις εξής εντολές:

- make run-lsh – για την εκτέλεση του προγράμματος Lsh με τα ορίσματα που δηλώνονται στην μεταβλητή ARGS_LSH
- make run-cube – για την εκτέλεση του προγράμματος Hypercube με τα ορίσματα που δηλώνονται στην μεταβλητή ARGS_CUBE
- make run-cluster – για την εκτέλεση του προγράμματος Clustering με τα ορίσματα που δηλώνονται στην μεταβλητή ARGS_CLUSTER
- make run-brute – για την εκτέλεση του προγράμματος BruteForce με τα ορίσματα που δηλώνονται στην μεταβλητή ARGS_BRUTE
- make run-graph – για την εκτέλεση του προγράμματος Graph με τα ορίσματα που δηλώνονται στην μεταβλητή ARGS_GRAPH

Χατζησπύρου Μιχαήλ – sdi: 1115202000212
Καζάκος Παναγιώτης – sdi: 1115201900067

Οι μεταβλητές ARGS_XXXXX είναι δηλωμένες εντός του Makefile και παίρνουν τα κατάλληλα ορίσματα προκειμένου να εκτελεστούν τα εκτελέσιμα αρχεία.

Επιπλέον δίνεται στον χρήστη η δυνατότητα εκτέλεσης των προγραμμάτων μέσω terminal ως εξής:

- ./bin/brute –d <input file> –q <query file> -N <int>? -o <output file> -sd <number of images to use for train> -sq <number of queries to find> -dinit <input file in original dimension> -qint <query file in original dimension>

  Default arguments: N = 1

- ./bin/graph –d <input file> –q <query file> –k <int>? -E <int>? -R <int>? -N <int>? -l <int, only for Search-on-Graph>? -m <1 for GNNS, 2 for MRNG> -o <output file> -sd <number of images to use for train> -sq <number of queries to find> -dinit <input file in original dimension> -qint <query file in original dimension>

  Default arguments: k = 50, E = 30, R = 1, N = 1, l = 20

- ./bin/cluster -i <input file> -c <configuration file> -o <output file> -complete -m <method: Classic or LSH or Hypercube> -sd <number of images to use> -dinit <input file in original dimension>

  Το configuration file είναι απαραίτητο για να οριστούν τα ορίσματα των αλγορίθμων LSH και Hypercube και είναι στην εξής μορφή:

  1. number_of_clusters: <int> K of KMeans++
  2. number_of_vector_hash_tables: <int>? default: L=3
  3. number_of_vector_hash_functions: <int>? k of LSH for vectors, default: 4
  4. max_number_M_hypercube: <int>? M of Hypercube, default: 10
  5. number_of_hypercube_dimensions: <int>? k of Hypercube, default: 3
  6. number_of_probes: <int>? probes of Hypercube, default:2

Όποια ορίσματα έχουν ερωτηματικό στο τέλος τους είναι προαιρετικά, ο αλγόριθμος θα χρησιμοποιήσει τα default.

Όσον αφορά το πρόγραμμα συμπίεσης των εικόνων ο χρήστης το εκτελεί πληκτρολογόντας σε ένα τερματικό την εξής εντολή:

- python3 reduce.py –d <dataset> -q <queryset> -od <output_dataset_file> -oq <output_query_file>

Στην συνέχεια το πρόγραμμα θα ρωτήσει τον χρήστη σε ποια από τις 3 διαστάσεις (10, 20, 30) θέλει να συμπιέσει τα αρχεία. Αφού ο χρήστης δώσει μια έγκυρη διάσταση τα αρχεία θα συμπιεστούν και θα αποθηκευτούν για μελλοντική χρήση.

4. Δομή Φακέλων Κώδικα

root

- bin — Εκτελέσιμα αρχεία από main συναρτήσεις του src και tests
- build — Αντικειμενικά αρχεία που δεν έχουν συνδεθεί
- datasets — Σύνολα δεδομένων, εισόδου και αναζήτησης
- modules — Κομμάτια κώδικα οργανωμένα σε επιμέρους φακέλους, κλάσεις και συναρτήσεις που χρησιμοποιούνται στα προγράμματα για καλύτερη οργάνωση του κώδικα και επαναχρησιμοποίησή του.
  - Cluster — Υλοποίηση της συσταδοποίησης
  - Common — Φάκελος που περιέχει κοινές δομές/υλοποιήσεις για όλα τα προγράμματα
    - BruteForce — Υλοποίηση της εξαντλητικής αναζήτησης
    - FileParser — Υλοποίηση του parser που διαβάζει τα αρχεία εικόνων
    - HashFunction — Υλοποίηση της hash function h
    - ImageDistance — Υλοποίηση γενικευμένης απόστασης
    - Utils — Διάφορα utils που χρησιμοποιούνται από όλους τους αλγορίθμους
  - Cube — Υλοποίηση του κύβου
  - Graphs — Φάκελος που περιέχει τις υλοποιήσεις των γραφοθεωρητικών αλγορίθμων
    - GNNS — Υλοποίηση του GNNS
    - MRNG — Υλοποίηση του MRNG
  - LSH — Υλοποίηση του Lsh
    - HashTable — Υλοποίηση των Hash tables για των αλγόριθμο
- src — Περιέχει τις main συναρτήσεις για την υλοποίηση όλων των αλγορίθμων στη νέα διάσταση καθώς και την υλοποίηση του νευρωνικού αυτοκωδικοποιητή και τα μοντέλα που αποθηκεύσαμε
- tests — Περιέχει τα scripts που χρησιμοποιήθηκαν για τα tests μαζί με τα αποτελέσματα

4.1. Φάκελος Modules

Περιέχει διάφορες συναρτήσεις και κλάσεις που ομαδοποιούν σημαντικές λειτουργίες των προγραμμάτων. Κάθε φάκελος έχει όνομα που ταιριάζει στα τρία διαφορετικά προγράμματα εκτός από τα Common modules που χρησιμοποιούνται σε όλα.

4.2. Φάκελος SRC

Περιέχει τις main συναρτήσεις για τα προγράμματα των αλγορίθμων προσεγγιστικής αναζήτησης, καθώς και για το πρόγραμμα που είναι υπεύθυνο για την συμπίεση των αρχείων στη νέα διάσταση. Η main συνάρτηση είναι υπεύθυνη να λύσει το ζητούμενο πρόβλημα κάθε φορά με αφηρημένο τρόπο διαβάζοντας τα κατάλληλα αρχεία, συνδυάζοντας τα απαραίτητα modules και εκτυπώνοντας τα αποτελέσματα.

4.3. Σημαντικές Δομές, Κλάσεις και Συναρτήσεις

Image: Αναπαριστά ένα σημείο από ένα MNIST σύνολο δεδομένων αποθηκεύοντας ένα αναγνωριστικό (id) και το διάνυσμα (pixels).

Neighbor: Αναπαριστά έναν γείτονα ενός σημείου. Αποθηκεύει το ίδιο το σημείο (image) αλλά και την απόσταση (distance) από το σημείο για το οποίο είναι γείτονας.

DistanceMetric: Enum τύπος που περιέχει τις Manhattan και Ευκλείδια μετρικές.

ImageDistance: Αποθηκεύει ποιά μετρική θα χρησιμοποιηθεί για το κάθε πρόγραμμα. Η αρχικοποίηση γίνεται μία φορα σε κάθε main και μπορεί να χρησιμοποιηθεί δυναμικά οπουδήποτε έχει κληθεί όταν τρέχει η εκάστοτε main.

CmdArgs: Αναλύει τα ορίσματα κάθε main συνάρτησης και τα αποθηκεύει σε μία εύκολα προσβάσιμη δομή. Για κάθε main υπάρχει ξεχωριστή κλάση (**LshCmdArgs**, **CubeCmdArgs**, **GraphCmdArgs**).

FileParser: Αναλύει τα αρχεία εισόδου και αν ταιριάζουν στη μορφή ενός MNIST συνόλου δεδομένων, τότε αποθηκεύει τα μεταδεδομένα και τις εικόνες στην προσωρινή μνήμη του προγραμμάτος.

Utils: Περιέχει διάφορες σύντομες συναρτήσεις που χρησιμοποιούνται σε όλα τα προγράμματα.

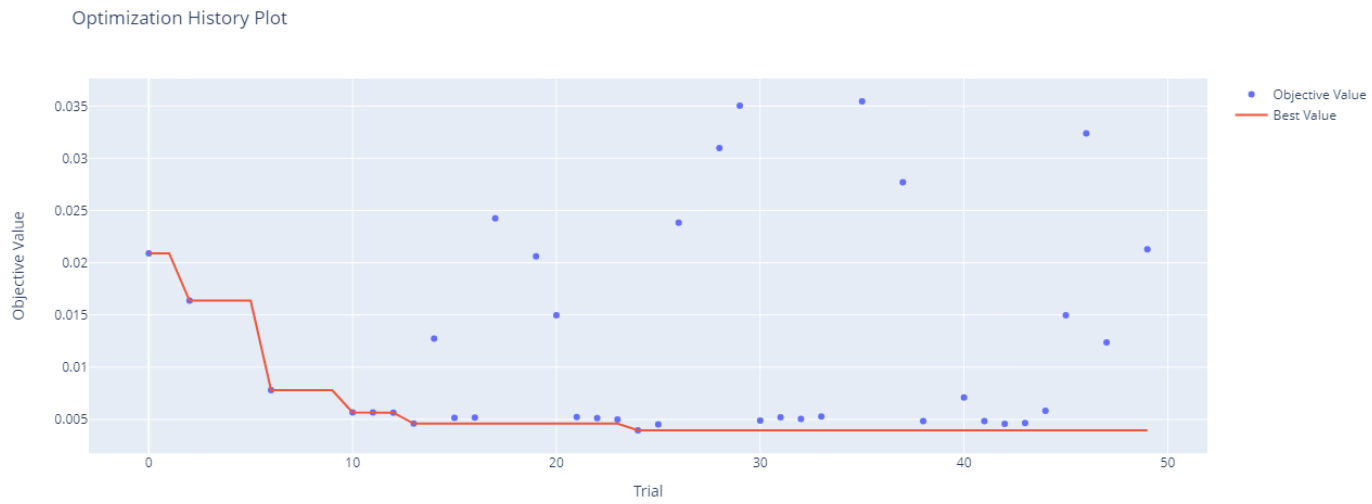
BruteForce: Πραγματοποιεί αναζήτηση Κ πραγματικών πλησιέστερων γειτόνων ελέγχοντας για ένα σημείο αναζήτησης την απόσταση του με όλα τα σημεία.

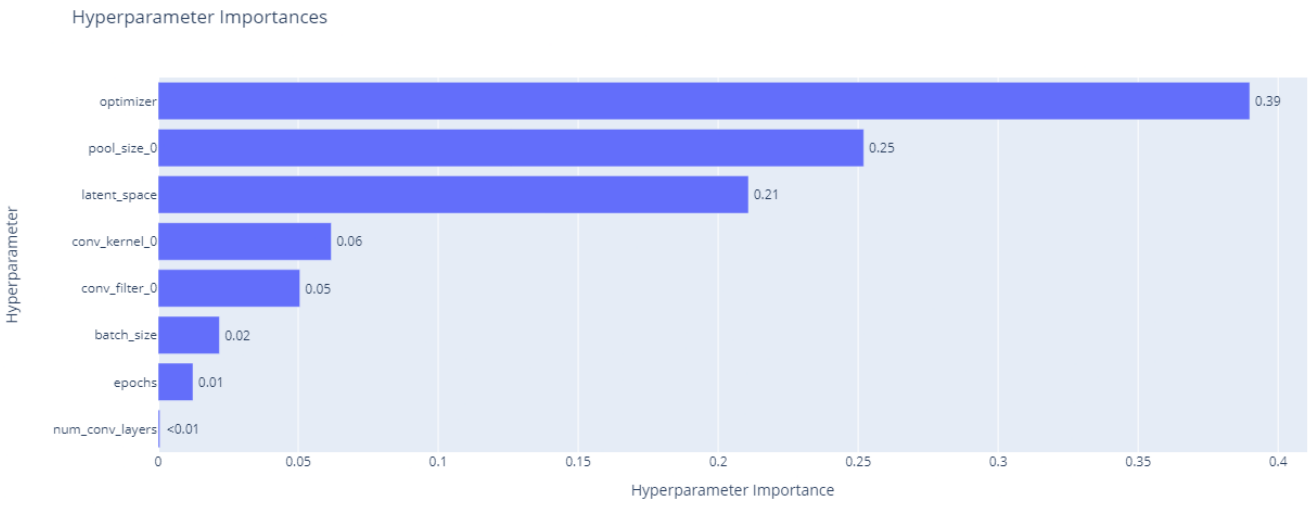
5. Autoencoder

Ένας συνελικτικός αυτόματος κωδικοποιητής είναι ένας τύπος αρχιτεκτονικής νευρωνικών δικτύων που έχει σχεδιαστεί για μάθηση χωρίς επίβλεψη και μείωση διαστάσεων των δεδομένων εισόδου. Αυτό το μοντέλο συνδυάζει συνελικτικά επίπεδα, που χρησιμοποιούνται συνήθως στην αναγνώριση εικόνας, με τις αρχές των αυτόματων κωδικοποιητών. Ο κωδικοποιητής χρησιμοποιεί συνελικτικά επίπεδα για να εξάγει ιεραρχικά χαρακτηριστικά από τα

δεδομένα εισόδου, καταγράφοντας χωρικές εξαρτήσεις και μοτίβα. Στη συνέχεια, ο αποκωδικοποιητής αναδομεί την αρχική είσοδο από την κωδικοποιημένη αναπαράσταση, επιτρέποντας στο μοντέλο να μάθει μια συμπιεσμένη και κατατοπιστική αναπαράσταση των δεδομένων.

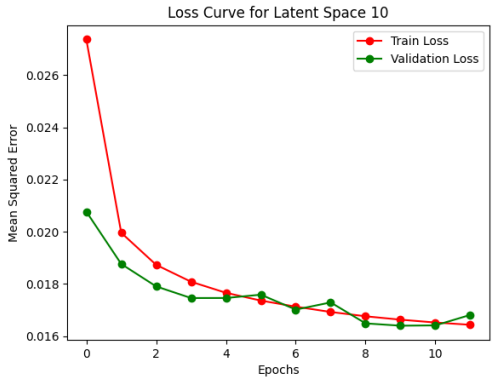
Σε αυτή την εργασία, ο στόχος μας ήταν να αναπτύξουμε έναν συνελικτικό αυτόματο κωδικοποιητή προσαρμοσμένο για τη μείωση διαστάσεων των συνόλων δεδομένων εικόνας MNIST. Αξιοποιώντας τον ισχυρό συνδυασμό TensorFlow και Keras, δημιουργήσαμε μια αρχιτεκτονική νευρωνικών δικτύων που ενσωματώνει απρόσκοπτα συνελικτικά επίπεδα, απαραίτητα για την καταγραφή χωρικών εξαρτήσεων στα δεδομένα εικόνας. Η πειραματική διαδικασία απλοποιήθηκε και εμπλουτίστηκε μέσω της ενσωμάτωσης του framework Optuna, διευκολύνοντας την εξερεύνηση των υπερπαραμέτρων. Μεταβάλλοντας συστηματικά the number of convolutional layers, filters, kernel sizes, pooling sizes, epochs, batch sizes, optimizers, and latent space dimensions, πραγματοποιήσαμε μια ολοκληρωμένη σειρά δοκιμών. Για κάθε latent space (10, 20, 30) προσδιορίστηκε η βέλτιστη διαμόρφωση, διασφαλίζοντας την εξαγωγή του πιο αποτελεσματικού μοντέλου για τη μείωση διαστάσεων στα σύνολα εικόνων MNIST. Αυτή η προσέγγιση όχι μόνο έδωσε πολλά υποσχόμενα αποτελέσματα, αλλά παρείχε επίσης πολύτιμες γνώσεις σχετικά με την αλληλεπίδραση των υπερπαραμέτρων και την απόδοση του μοντέλου.

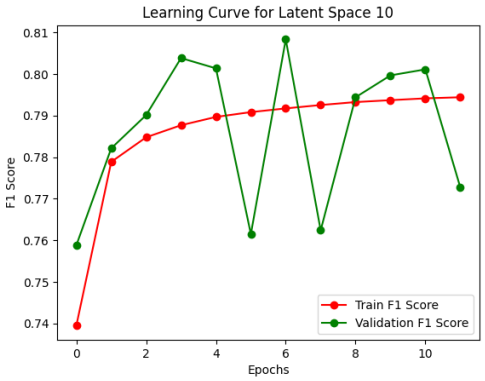




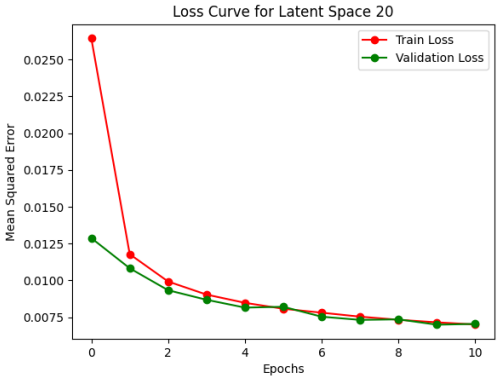
Καθ΄ όλη τη διάρκεια των δοκιμών μας, εμφανίστηκε μια ευδιάκριτη τάση καθώς η συνάρτηση απώλειας μειώθηκε σταθερά μέχρι περίπου το 0,005, ένα φαινόμενο που συνδέεται με την επιλεγμένη latent space. Αξιοσημείωτα, έγινε προφανές ότι οι μεγαλύτερες latent space οδήγησαν σε μια πιο αποτελεσματική μείωση της απώλειας, τονίζοντας τον κρίσιμο ρόλο που διαδραματίζει το latent space στην απόδοση του συνελικτικού αυτόματου κωδικοποιητή. Επιπλέον, εμφανίστηκε μια αξιοσημείωτη παρατήρηση - ο optimizer αποδείχθηκε ότι ήταν η πιο σημαντική υπερπαράμετρος μεταξύ όλων. Η επιλογή του βελτιστοποιητή επέδειξε βαθύ αντίκτυπο στην ικανότητα του μοντέλου να συγκλίνει αποτελεσματικά και να μειώσει αποτελεσματικά τις απώλειες κατά την προπόνηση. Αυτό υπογραμμίζει τη σημασία της προσεκτικής επιλογής βελτιστοποιητή στη διαδικασία λεπτομέρειας, καθώς μπορεί να επηρεάσει σημαντικά τη συνολική απόδοση και την επιτυχία του συνελικτικού αυτόματου κωδικοποιητή στη μείωση διαστάσεων για σύνολα εικόνων MNIST.

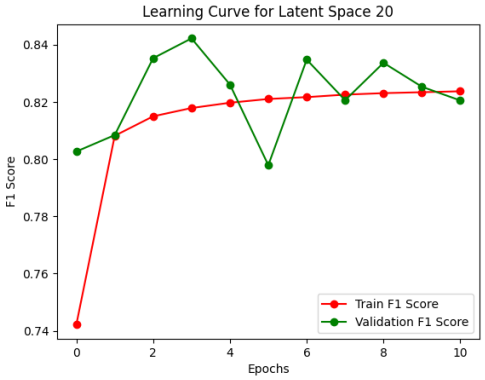
5.1. Latent Space 10



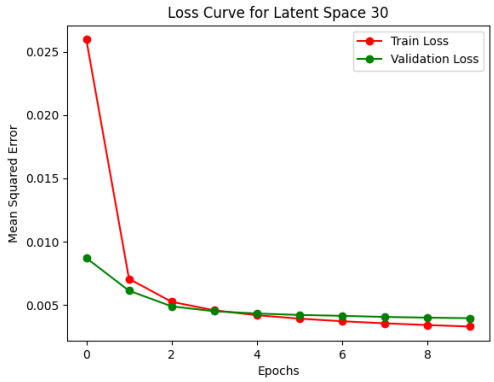


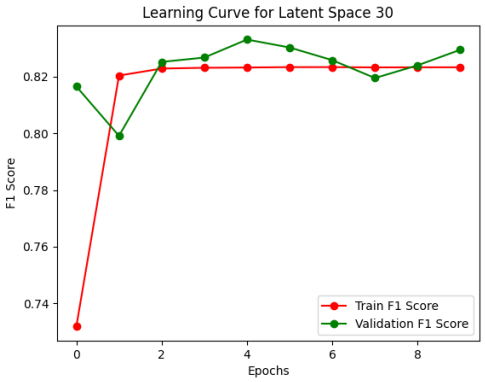
5.2. Latent Space 20





5.3. Latent Space 30





Η εξέταση των καμπυλών εκμάθησης, που περιλαμβάνει τόσο την απώλεια όσο και τη βαθμολογία F1, επιβεβαιώνει το προηγούμενο συμπέρασμα σχετικά με τη σχέση μεταξύ του latent space και της ελαχιστοποίησης των απωλειών. Με συνέπεια, τα δεδομένα υποδεικνύουν ότι οι μεγαλύτεροι λανθάνοντες χώροι αντιστοιχούν σε μικρότερες απώλειες, υπογραμμίζοντας τη σημασία αυτής της παραμέτρου στη διαμόρφωση της απόδοσης του συνελικτικού αυτόματου κωδικοποιητή. Επιπλέον, η ανάλυση μας παρουσιάζει την άψογη ισορροπία που παρατηρείται σε όλα τα μοντέλα, όπως αποδεικνύεται από την απουσία τάσεων υπερπροσαρμογής ή υποπροσαρμογής. Αν και εμφανίζονται ευδιάκριτες αιχμές στα διαγράμματα βαθμολογίας F1, υποδηλώνοντας πιθανή αστάθεια, είναι σημαντικό να υπογραμμιστεί η ελάχιστη απόκλιση μεταξύ των τιμών του άξονα. Η εγγύτητα αυτών των τιμών δείχνει ότι, παρά τις περιστασιακές διακυμάνσεις, τα μοντέλα διατηρούν μια αξιοσημείωτη σταθερότητα, τεκμηριώνοντας την στιβαρότητά τους στον χειρισμό των περιπλοκών των συνόλων εικόνων MNIST και ενισχύοντας την αξιοπιστία του συνελικτικού αυτόματου κωδικοποιητή στην επίτευξη ισορροπημένης και αποτελεσματικής μείωσης διαστάσεων.

6. Αξιολόγηση

Για την αξιολόγηση, χρησιμοποιήθηκαν οι εξής μετρικές:

For Nearest Neighbors:

- Time Taken – Μέσος χρόνος αναζήτησης
- AAF – Average Approximation Factor
- MAF – Maximum Approximation Factor

For Clustering:

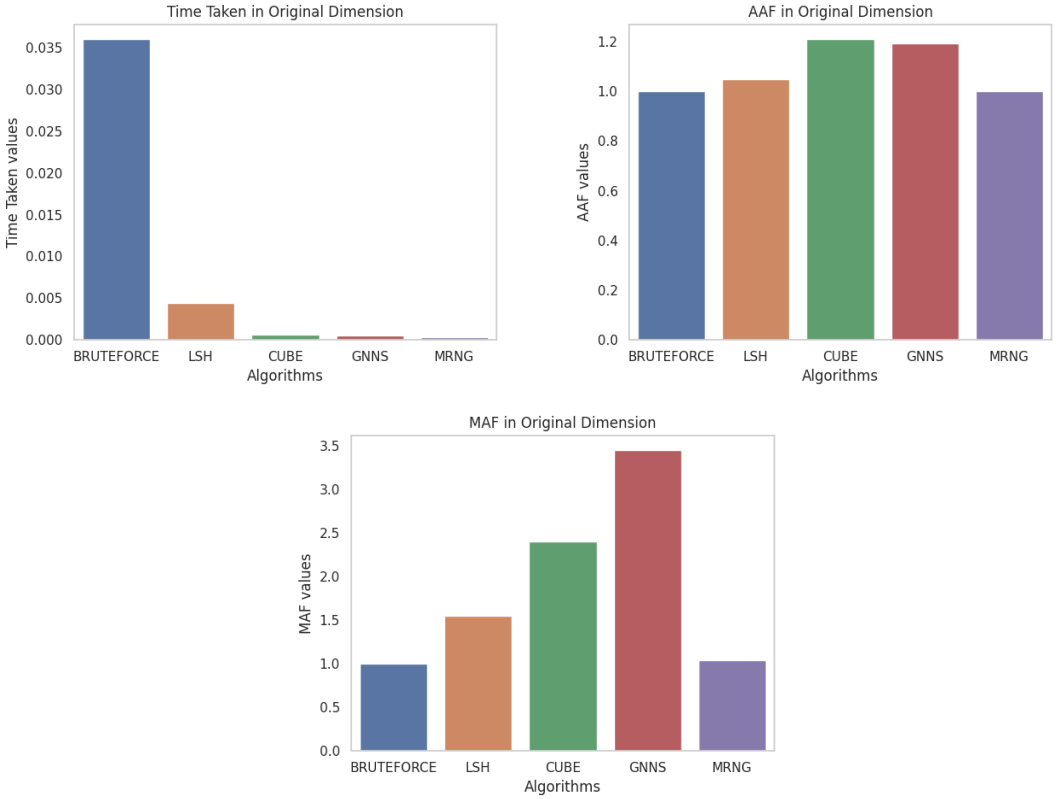
- Clustering Time – Χρόνος Συσταδοποίησης
- Average Silhouette – Μέση Σιλουέτα
- Objective Function – Αντικειμενική Συνάρτηση

Η επιλογή αυτών των μετρήσεων καθοδηγήθηκε από τη δέσμευση για συνέπεια σε προηγούμενες εργασίες, διασφαλίζοντας μια τυποποιημένη προσέγγιση για την αξιολόγηση της

απόδοσης. Συγκεκριμένα, στο πλαίσιο των λανθάνοντων διαστάσεων, υπολογίστηκαν συγκεκριμένες μετρήσεις όπως το AAF, MAF, Average Silhouette και η Objective Function μετά την προβολή των εικόνων πίσω στον αρχικό χώρο. Αυτή η διαφοροποιημένη προσέγγιση παρέχει μια ολοκληρωμένη κατανόηση της ικανότητας του μοντέλου να ανακατασκευάζει με ακρίβεια και να αναπαριστά τα δεδομένα εισόδου εντός των επιλεγμένων λανθάνοντων διαστάσεων. Με την τήρηση ενός συνεπούς μετρικού πλαισίου, στοχεύαμε να δημιουργήσουμε μια ισχυρή βάση για σύγκριση και ανάλυση, επιτρέποντας μια αξιόπιστη αξιολόγηση της αποτελεσματικότητας του συνελικτικού αυτόματου κωδικοποιητή στην επίτευξη μείωσης διαστάσεων για σύνολα εικόνων MNIST.

6.1. Nearest Neighbors

6.1.1. Original Dimension:



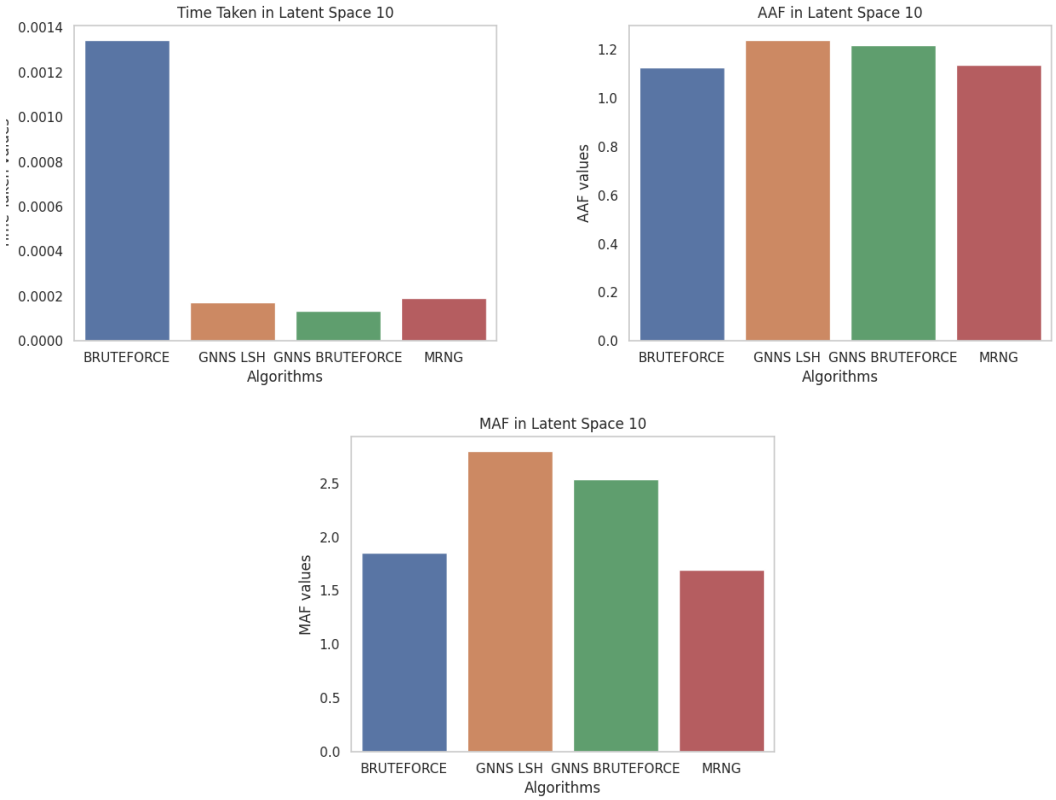
Σχήμα 5: Original Dimension Plots

Πίνακας 1: Original Dimension Table

Original Dimension	Time Taken	AAF	MAF
BRUTEFORCE	0.036007000	1	1
LSH	0.004449070	1.04879	1.54656
CUBE	0.000615765	1.20910	2.40257
GNNS	0.000496924	1.19411	3.44551
MRNG	0.000326701	1.00002	1.04169

Από τις γραφικές παραστάσεις (5) και τον πίνακα (1), μπορούμε να παρατηρήσουμε ότι στην αρχική διάσταση, ο αλγόριθμος ΒρυτεΦορςε ξεχωρίζει ως ο πιο αργός ως προς την υπολογιστική πολυπλοκότητα. Ωστόσο, η υπολογιστική πολυπλοκότητα του έχει ως συνέπεια την άψογη αποτελεσματικότητα του, η οποία προέρχεται από την εξαντλητική προσέγγιση δοκιμής κάθε συνδυασμού, καθιστώντας τον τον πιο ακριβή. Ωστόσο, για εκείνους που αναζητούν έναν σημαντικά ταχύτερο αλγόριθμο χωρίς συμβιβασμούς στην ακρίβεια, ο αλγόριθμος MRNG φαίνεται να είναι μια πολλά υποσχόμενη επιλογή, δεδομένου του γρήγορου χρόνου αναζήτησής του και της εξαιρετικής του ακρίβειας 1,00002. Από την άλλη πλευρά, η μέθοδος LSH προσφέρει μια αξιοπρεπή ισορροπία όσον αφορά την ακρίβεια, αλλά είναι σημαντικό να ληφθεί υπόψη ο μεγαλύτερος χρόνος αναζήτησής της σε σύγκριση με το MRNG. Όσον αφορά τους αλγόριθμους Cube και GNNS, φαίνεται να ταιριάζουν στενά τόσο σε χρόνο αναζήτησης όσο και σε ακρίβεια, καθιστώντας οποιονδήποτε από αυτούς βιώσιμες επιλογές ανάλογα με συγκεκριμένες απαιτήσεις ή προτιμήσεις.

6.1.2. Latent Space 10:



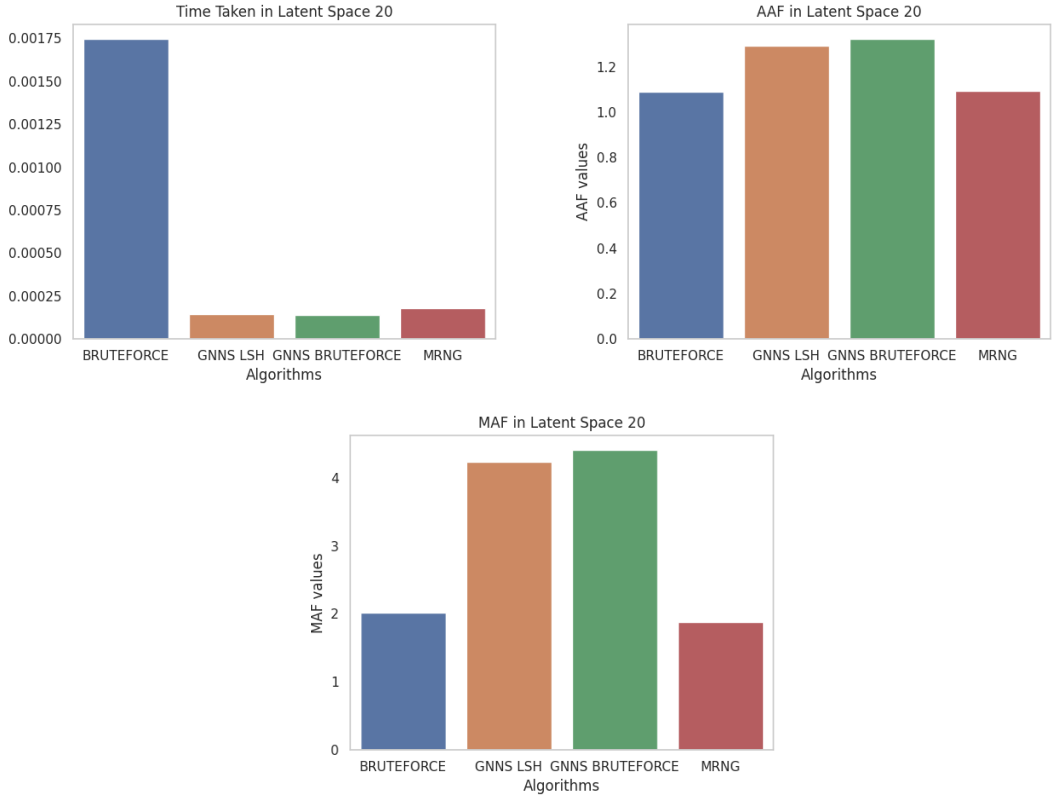
Σχήμα 6: Latent Space 10 Plots

Πίνακας 2: Latent Space 10 Table

Latent Space = 10	Time Taken	AAF	MAF
BRUTEFORCE	0.001343520	1.12587	1.85081
GNNS LSH	0.000171030	1.23841	2.79808
GNNS BRUTEFORCE	0.000133263	1.21754	2.53537
MRNG	0.000192239	1.13878	1.69740

Προχωρώντας στον πρώτο latent space (6, 2), εμφανίζεται μια αξιοσημείωτη αλλαγή στην απόδοση του αλγόριθμου BruteForce, ο οποίος μετατρέπεται σε έναν κατά προσέγγιση αλγόριθμο. Αυτή η αλλαγή μπορεί να αποδοθεί στη συμπίεση των διαστάσεων, με αποτέλεσμα την απώλεια ορισμένων πληροφοριών που καθιστούν τον αλγόριθμο ανίκανο να προσδιορίσει με ακρίβεια τον αληθινό πλησιέστερο γείτονα. Παρά αυτή την προσέγγιση, ο BruteForce παραμένει ο πιο αργός αλγόριθμος σε αυτό το πλαίσιο. Από την άλλη πλευρά, ο MRNG διατηρεί τη θέση του ως ο δεύτερος καλύτερος αλγόριθμος, επιδεικνύοντας σημαντικά μειωμένο χρόνο αναζήτησης με σχεδόν ίδια ακρίβεια με αυτή του BruteForce. Παραδόξως, ο GNNS, με τις δύο παραλλαγές του LSH και BruteForce, παρουσιάζει μια εντυπωσιακή ομοιότητα στην ακρίβεια. Αρχικά, υπήρχε η υπόθεση ότι το LSH, ως ένας κατά προσέγγιση αλγόριθμος, μπορεί να δυσκολευτεί στον latent space, αλλά αυτή η υπόθεση αποδείχθηκε ότι ήταν μόνο εν μέρει ακριβής.

6.1.3. Latent Space 20:



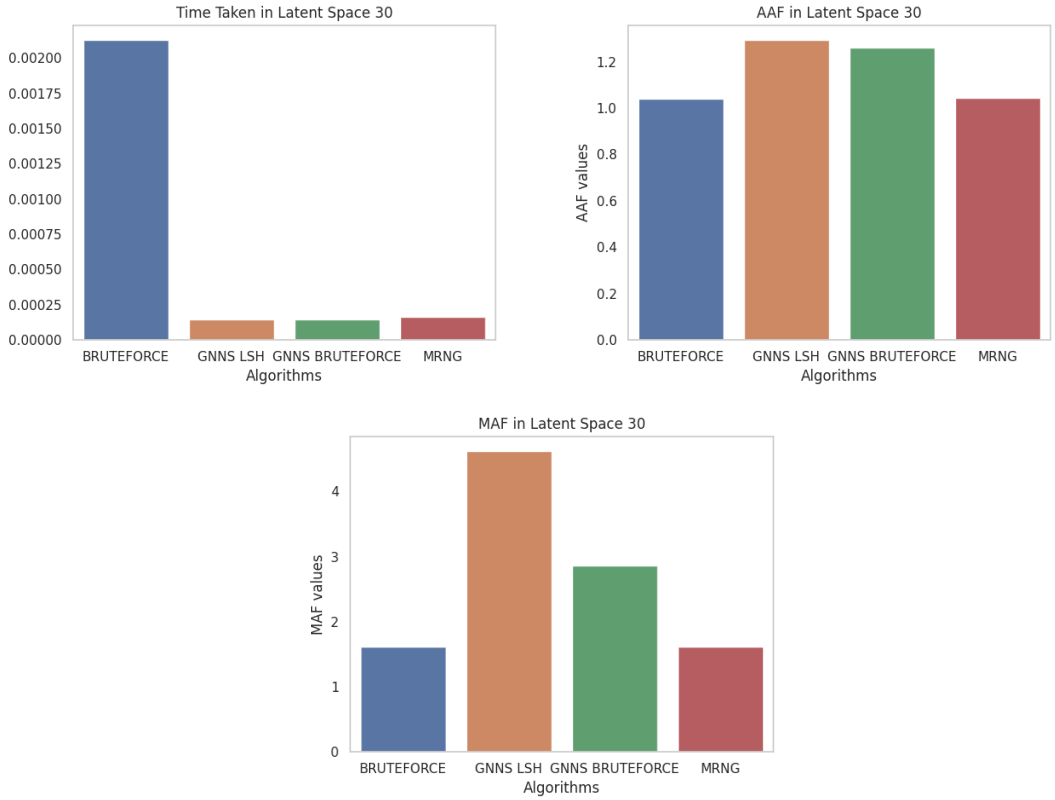
Σχήμα 7: Latent Space 12 Plots

Πίνακας 3: Latent Space 20 Table

Latent Space = 20	Time Taken	AAF	MAF
BRUTEFORCE	0.001746020	1.08886	2.01859
GNNS LSH	0.000142500	1.29320	4.23699
GNNS BRUTEFORCE	0.000138986	1.32244	4.40786
MRNG	0.000179752	1.09369	1.88183

Στην ανάλυση του δεύτερου latent space (7, 3), ξεχωρίζουν δύο διακριτές συγκρίσεις, ιδιαίτερα μεταξύ των αλγορίθμων BruteForce και MRNG. Όπως αναμενόταν, ο χρόνος που απαιτείται για τον υπολογισμό είναι μικρότερος από ό,τι στην αρχική διάσταση και η ακρίβεια συνεχίζει να βελτιώνεται με τη μεγαλύτερη συμπιεσμένη διάσταση, υποδηλώνοντας λιγότερα χαμένα κομμάτια πληροφοριών. Ωστόσο, μια αξιοσημείωτη παρατήρηση είναι η μεγαλύτερη αιχμή στο MAF για το BruteForce στο 2,01, σε σύγκριση με το 1,88 του MRNG. Αυτή η απόκλιση υποδεικνύει ότι το MRNG μπορεί να είναι πιο κατάλληλη επιλογή σε διάφορα σενάρια λόγω του χαμηλότερου ποσοστού σφάλματος. Από την άλλη, οι υποψίες για το GNNS επικυρώνονται, όπως αποδεικνύεται από τον AAF, ο οποίος επιδεινώνεται παρά την αύξηση του λανθάνοντος χώρου.

6.1.4. Latent Space 30:



Σχήμα 8: Latent Space 30 Plots

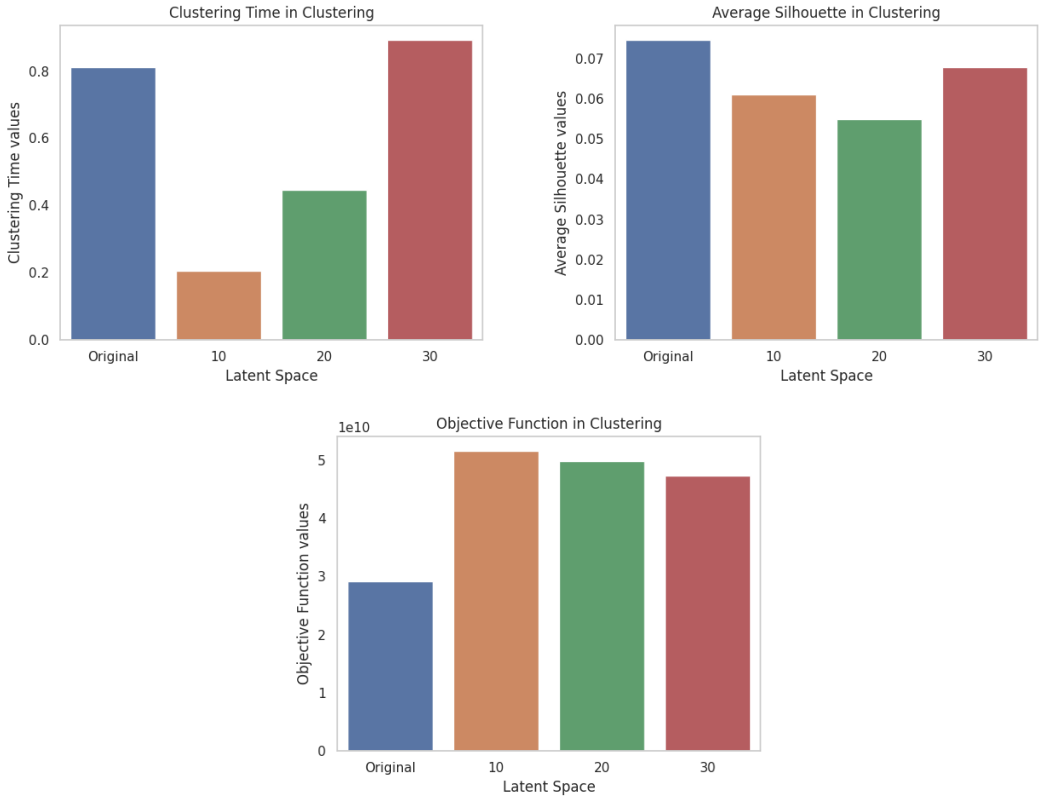
Πίνακας 4: Latent Space 30 Table

Latent Space = 30	Time Taken	AAF	MAF
BRUTEFORCE	0.002126670	1.04199	1.61952
GNNS LSH	0.000145430	1.29336	4.62074
GNNS BRUTEFORCE	0.000145230	1.25933	2.85942
MRNG	0.000166726	1.04451	1.61952

Στην εξέταση του τελευταίου latent space (8, 4), παρόμοιες τάσεις επιμένουν, επιβεβαιώνοντας τα μοτίβα που παρατηρήθηκαν στις προηγούμενες αναλύσεις. Ο αλγόριθμος

BruteForce παραμένει ο πιο αργός αλλά επιδεικνύει απαράμιλλη ακρίβεια. Το MRNG, από την άλλη πλευρά, διατηρεί ένα αξιέπαινο επίπεδο ακρίβειας, σχεδόν στο ίδιο επίπεδο με το BruteForce, αλλά πολύ πιο γρήγορο από άποψη χρόνου αναζήτησης. Είναι ενδιαφέρον ότι σε αυτήν την περίπτωση, το MAF είναι πανομοιότυπο τόσο για το BruteForce όσο και για το MRNG, δίνοντας έμφαση στην αποτελεσματικότητα του MRNG χωρίς να διακυβεύεται η ακρίβεια. Όσο για το GNNS, ενώ η ακρίβεια τόσο στις παραλλαγές LSH όσο και στις παραλλαγές BruteForce βελτιώνεται από τον προηγούμενο latent space, δεν φτάνει τα υψηλά επίπεδα που παρατηρούνται στον latent space 10.

6.2. Clustering Lloyd's



Σχήμα 9: Latent Space 30 Plots

Πίνακας 5: Clustering Table

Latent Space	Clustering Time	Average Silhouette	Objective Function
Original	0.811292	0.0745650	2.91229e+10
10	0.205660	0.0610482	5.15093e+10
20	0.446624	0.0549118	4.97412e+10
30	0.892129	0.0677806	4.73113e+10

Στο Clustering (9, 5), αναμενόταν ότι το clustering time θα αυξανόταν αναλογικά με την ανάπτυξη του latent space. Παραδόξως, ο latent space των 30 χρειάστηκε λίγο περισσότερο χρόνο από τον αρχικό, ενώ οι latent spaces των 10 και 20 παρουσίασαν σημαντικά μειωμένους χρόνους επεξεργασίας. Έγινε μια ενδιαφέρουσα παρατήρηση όσον αφορά τις

Silhouette. Αναμενόταν ότι η μέτρηση θα βελτιωνόταν με μια αύξηση στον latent space, κάτι που επιβεβαιώθηκε για τους latent space 20 και 30. Ωστόσο, ο latent space 10 έδειξε μεγαλύτερη ακρίβεια στη ομαδοποίηση από 20, σε αντίθεση με την προσδοκία ότι οι μικρότεροι latent space ενδέχεται να έχουν ως αποτέλεσμα μεγαλύτερη απώλεια δεδομένων και μικρότερη Silhouette. Τέλος, λαμβάνοντας υπόψη την αντικειμενική συνάρτηση, η οποία στοχεύει στη μέτρηση της αποτελεσματικότητας της ομαδοποίησης ελαχιστοποιώντας το άθροισμα των τετραγώνων αποστάσεων μεταξύ των σημείων δεδομένων και των εκχωρημένων κεντροειδών συστάδων τους, προέκυψε μια αξιοσημείωτη τάση. Καθώς η διάσταση μειώθηκε, η αντικειμενική συνάρτηση έγινε μικρότερη, υποδεικνύοντας ότι η ομαδοποίηση γινόταν σταδιακά πιο ακριβής όπως και είχαμε προβλέψει.