3η Εργασία

Λαμπρόπουλος Κωνσταντίνος sdi1800092,
Γκέργκη Δημήτρης sdi1800029

Περίληψη

Στην εργασία αυτή μας ζητήθηκε η δημιουργία και εκπαίδευση ενός Autoencoder όπως και η χρήση των εικόνων στη νέα διάσταση για εύρεση πλησιέστερου γείτονα και συσταδοποίηση.Ειδικότερα,πειραματιστήκαμε με διάφορες παραμμέτρους του νευρωνικού δικτύου,με σκοπό την ελαχιστοποίηση του σφάλματος και ταυτόχρονα την δημιουργία εικόνων σε μία διάσταση αρκετά μικρή ώστε να βελτιστοποιηθεί ο χρόνος εκτέλεσης των αλγορίθμων εύρεσης πλησιέστερου γείτονα και συσταδοποίησης.Αρχικά στην ενότητα 1,θα αναλύσουμε περαιτέρω την διαδικασία εκπαίδευσης και μοντελοποίησης του νευρωνικού δικτύου,παρέχοντας και Learning Curves για να διακρίνουμε αν το μοντέλο μας έχει καλό generalization και αν έχουμε φαινόμενα overfitting.

Η 2η ενότητα προβαίνει σε μια πλήρη παρουσίαση των αποτελεσμάτων που προέκυψαν από τη χρήση των προσεγγιστικών αλγορίθμων, καθώς και της εξαντλητικής αναζήτησης, χρησιμοποιώντας το νέο διανυσματικό χώρο που παράχθηκε από το νευρωνικό δίκτυο της προηγούμενης ενότητας. Επιπλέον, αξιολογείται η απόδοσή τους όσον αφορά την απόσταση του κοντινότερου γείτονα, τον χρόνο, καθώς και τη μέση απόσταση των προσεγγιστικών αλγορίθμων σε σύγκριση με τον πραγματικό και προσσεγιστικό πλησιέστερο γείτονα στον αρχικό χώρο. Η παρουσίαση αυτών των αποτελεσμάτων αποσκοπεί επίσης στην παραγωγή συμπερασμάτων ως προς την αξία δημιουργίας και χρήσης ενός τέτοιου νέου διανυσματικού χώρου μειωμένης διάστασης,συγκριτικά με τον αρχικό διανυσματικό χώρο.

Η 3η και τελευταία ενότητα,αποσκοπεί στην σύγκριση και ανάλυση των αποτελεσμάτων συσταδοποίησης στον νέο διανυσματικό χώρο και στον αρχικό.Πιο συγκεκριμένα,θα συγκρίνουμε τις τιμές Silhouette κάθε συστάδας, καθώς και της μέσης τιμής Silhouette όλων των συστάδων,του αρχικού και του νέου διανυσματικού χώρου,θα συγκρίνουμε τους χρόνους συσταδοποίησης στους 2 χώρους αυτούς και την τιμή της συνάρτησης αποτίμησης στόχου (Objective Function).Τέλος θα αποφανθούμε για την αξία χρήσης ενός νέου διανυσματικού χώρου μειωμένης διάστασης,σε φαινόμενα συσταδοποίησης.

Περιεχόμενα

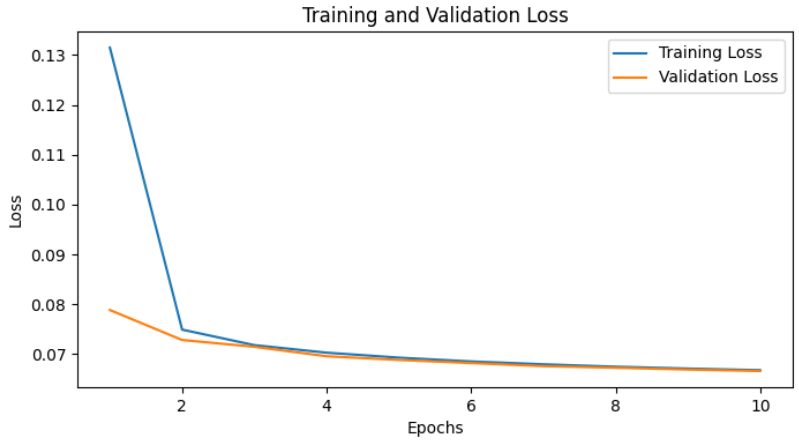
1 Autoencoder

1.1 Δημιουργία

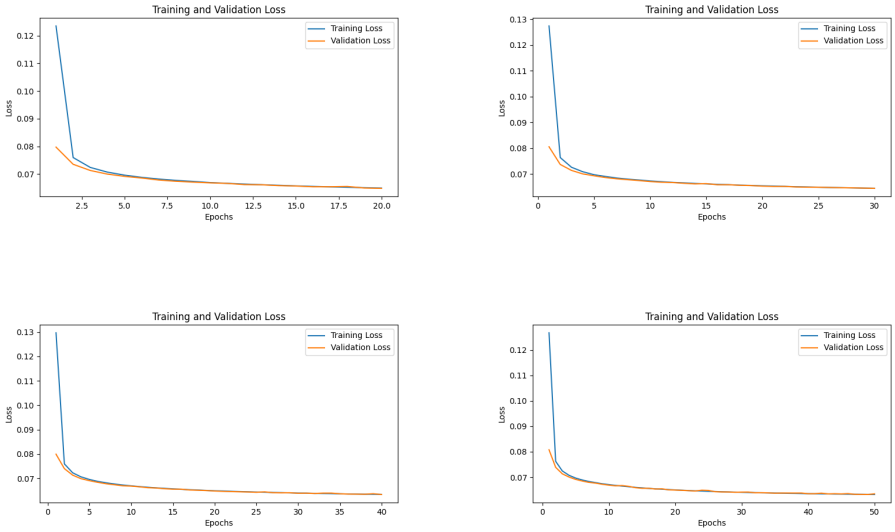
Ένα Autoencoder Νευρωνικό Δίκτυο αποτελείται από 2 σημεία: Το Encoder και το Decoder.Ειδικότερα,ο Encoder κωδικοποιεί την εικόνα,μειώνοντας την διάσταση της προσπαθώντας παράλληλα να κρατήσει όση περισσότερη πληροφορία,ενώ ο Decoder έχοντας την κωδικοποιημένη είκονα,προσπαθεί να ανακατασκευάσει την αρχική.Σκοπός μας είναι η μείωση του Loss.Αρχικά έχουμε 3 encoding layers και 3 decoding μαζί με 2 downsampling και 2 upsampling και μέχρι την αλλαγή των convolutional layers,χρησιμοιείται αυτό το Νευρωνικό Δίκτυο.

1.2 Βελτιστοποίηση (Tuning)

Αρχικά η πρώτη παράμμετρος για την βελτιστοποίηση του νευρωνικού δικτύου,την οποία εξετάσαμε είναι οι εποχές της εκπαίδευσης (Epochs).Ξεκινήσαμε με 10 epochs :



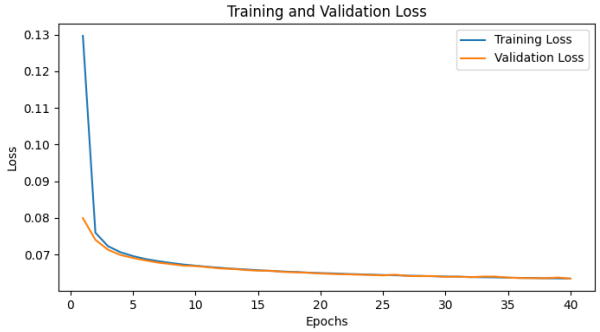
Παρατηρούμε ότι το μοντέλο μας έχει καλό generalization και δεν δείχνει σημάδια overfitting μιάς και στο validation set και στο training set αποδίδει εξίσου καλά.Όμως παρατηρούμε πως δεν σταθεροποιούνται οι καμπύλες,που σημαίνει πως πιθανότατα να μπορεί να μειωθεί περαιτέρω το loss αυξάνοντας τις εποχές.Παρακάτω είναι οι υπόλοιποι πειραματισμοί με τα epochs:



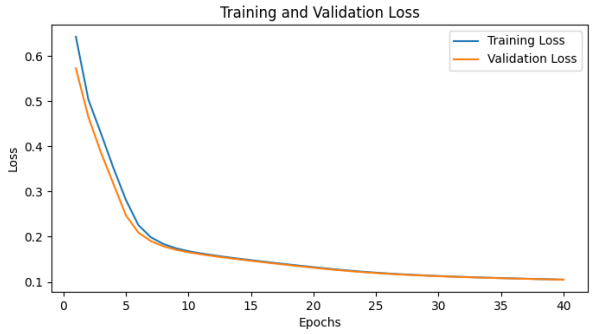
Παρατηρούμε ότι ενώ από τις 20 εποχές στις 30 και στις 40 ,το loss μειώνεται αρκετά,από τις 40 στις 50 εποχές υπάρχει ελάχιστη μείωση.Αν λάβουμε υπόψη πως αυξάνοντας τις εποχές αυξάνεται και ο χρόνος εκπαίδευσης,η μείωση δεν είναι σημαντική για να την λάβουμε υπόψιν.Συνεπώς επιλέξαμε τις 40 εποχές ως τη βέλτιστη επιλογή.Παρακάτω είναι ο πίνακας των πειραμάτων:

Epochs	Loss
10	0.0668
20	0.0649
30	0.0644
40	0.0634
50	0.0632

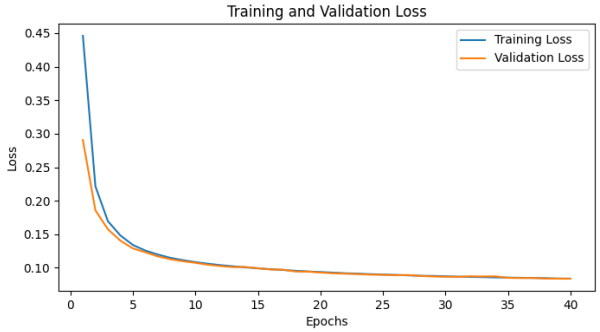
Έπειτα δοκιμάστηκαν διαφορετικοί optimizers.Ο default ήταν ο Adam, το learning curve του οποίου φαίνεται παρακάτω:



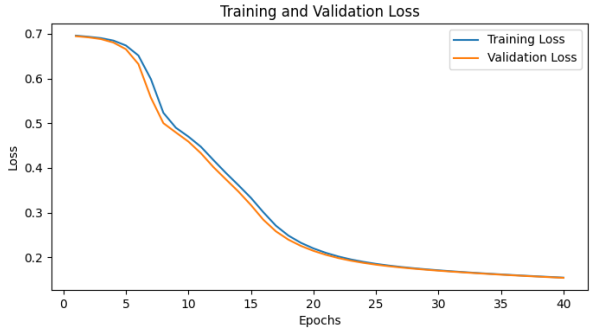
Παρατηρούμε ότι ο Adam παράγει αρκετά καλά αποτελέσματα.Επόμενος optimizer είναι ο Adagrad:



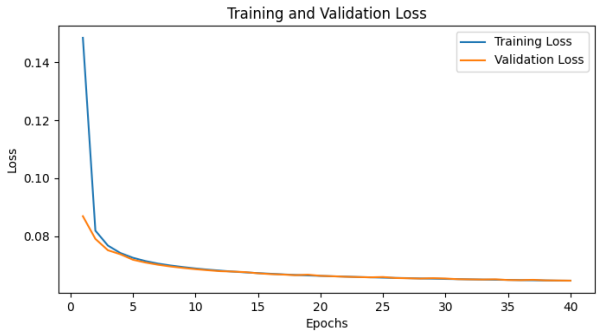
Φαίνεται πως ο Adagrad δεν ελαχιστοποιεί το loss όσο ο Adam.Προχωρώντας έχουμε τον Stohastic Gradient Descent (SGD):



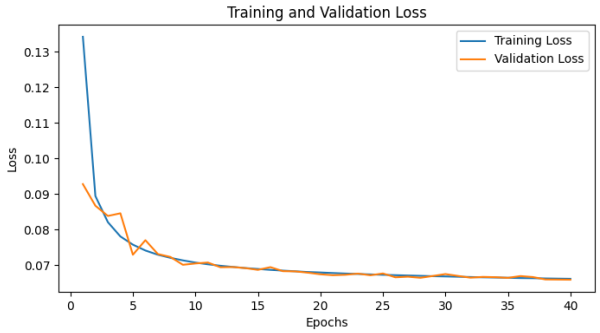
Ο SGD πετυχαίνει καλύτερη ελαχιστοποίηση του loss από τον Adagrad αλλά ακόμα χειρότερη από τον Adam.Συνεχίζουμε με τον Adadelta :



Ο Adadelta εμφανίζει το μεγαλύτερο loss μέχρι στιγμής. Τέλος μας μένουν οι Adamax και RMS-prop όπου και οι δύο ελαχιστοποιούν αρκετά καλά το loss αλλά όχι όσο ο Adam.Για τον Adamax:



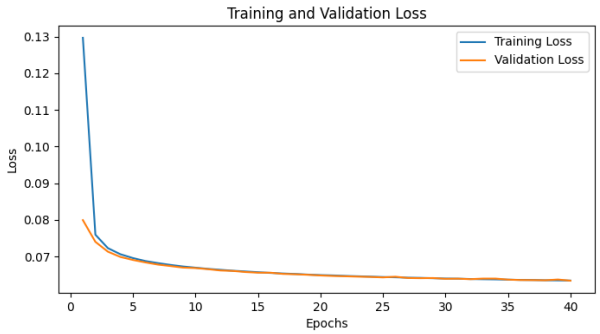
και για τον RMS-prop:



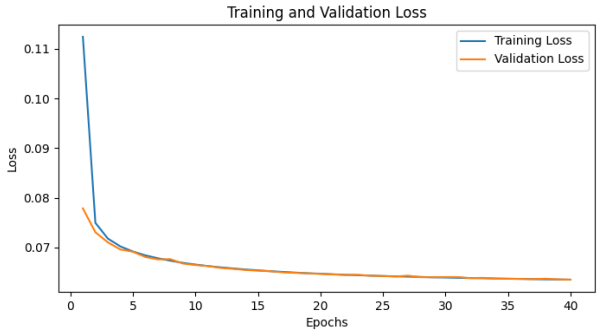
Τελικά παρατηρούμε ότι ενώ όλοι οι optimizers δεν προκαλούν overfitting και αποδίδουν αρκετά καλά,ο βέλτιστος είναι ο Adam. Παρακάτω είναι ο πίνακας με τα πειράματα:

Optimizer	Loss
Adam	0.0634
Adagrad	0.1050
SGD	0.0861
Adadelta	0.1549
Adamax	0.0646
RMS-prop	0.0662

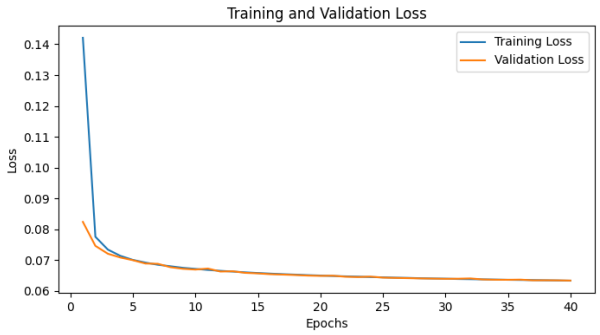
Στη συνέχεια εξετάσαμε κάποια activation functions.Αρχικά ο default μας ήταν ο ReLu ο οποίος μας έδωσε το εξής learning curve:



Τα αποτελέσματα είναι πολύ καλά,αλλά χρειάζεται να πειραματιστούμε και με άλλους activators.Ο επόμενος activator με τον οποίο πειραματιστήκαμε, ήταν ο Elu:



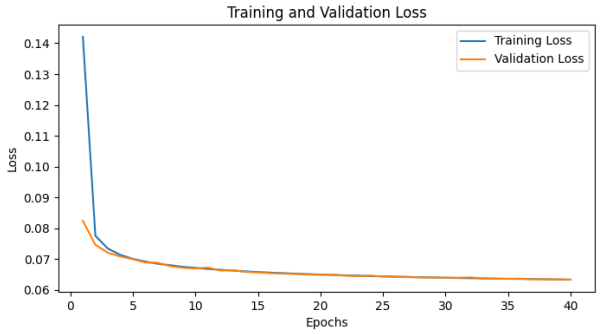
Τα αποτελέσματα είναι σχεδόν παρόμοια,με διαφορά της τάξης του 0.0001,αλλά ήταν και λίγο πιο αργός.Τέλος έχουμε τον GeLu :



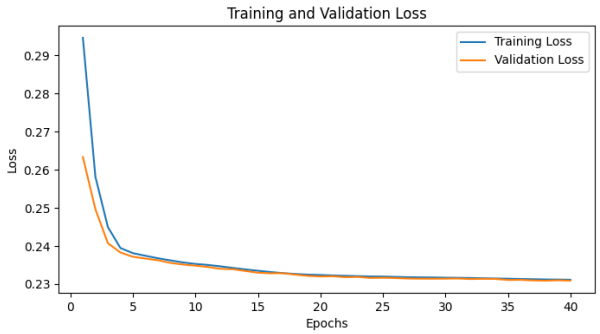
ο οποίος έστω και για ελάχιστο,ήταν καλύτερος και γι αυτό προτιμήθηκε.Παρακάτω είναι ο πίνακας των πειραμάτων:

Activators	Loss
Elu	0.0635
ReLu	0.0634
GeLu	0.0633

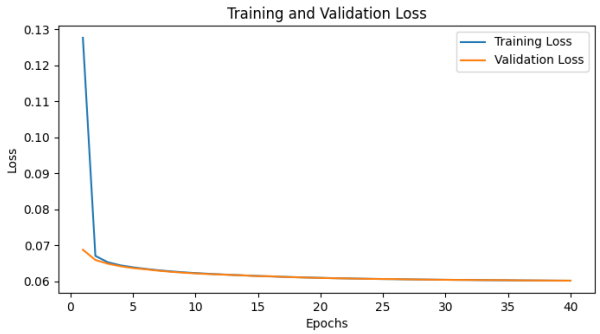
Επειτα,πειραματιστήκαμε με τον αριθμό των Convolutional Layers του νευρωνικού μας δικτύου.Ειδικότερα αρχίσαμε με 3 στρώματα encoding 3 decoding,όπου κάθε στρώμα encoding(πέραν του τελευταίου) έχει και ένα downsampling στρώμα,και κάθε στρώμα decoding(πέραν πάλι του τελευταίου) έχει ένα στρώμα upsampling. Χρησιμοποιώντας 3 συνελικτικά στρώματα σε καθε phase(encoding,decoding έχουμε:



Χρησιμοποιώντας 4 στρώματα σε κάθε επίπεδο έχουμε :



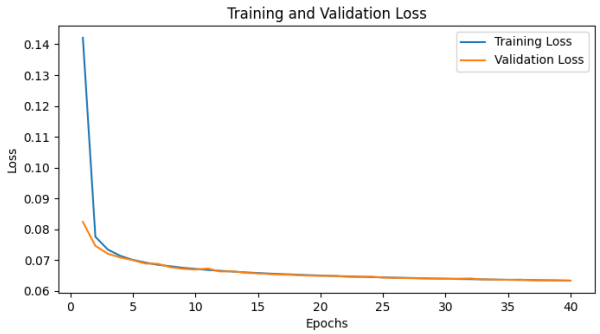
και με 2 στρώματα:



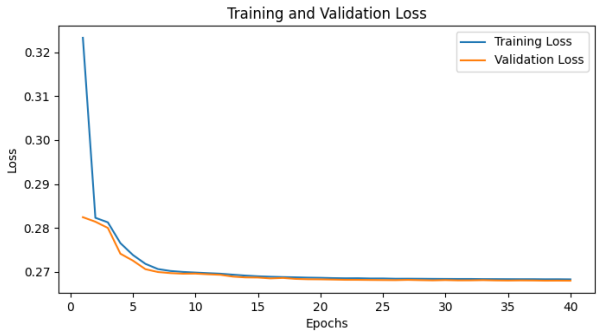
Παρατηρούμε ότι όσο μειώνουμε τα συνελικτικά στρώματα του νευρωνικού δικτύου,αυξάνεται η απόδοση του.Αυτό είναι αναμενόμενο αφού καθώς αυξάνουμε το μέγεθος του δικτύου μας,έχουμε και μεγαλύτερη συμπίεση της εικόνας,αλλά αυτό σημαίνει πως θα χάνεται περισσότερη πληροφορία.Για παράδειγμα στο πείραμα με 2 συνελτικά στρώματα η εικόνα στο latent dimension έχει διάστασεις (14x14x10) ,ενώ στο πείραμα με 3 συνελικτικά στρώματα έχουμε διάσταση 7x7x10). Παρατηρούμε ότι έχοντας 2 συνελικτικά στρώματα παράγουμε μια εικόνα μεγαλύτερης διάστασης από την αρχική.Επίσης στα 4 στρώματα έχουμε συνειδητά μεγαλύτερο loss,οπότε προτιμήθηκαν 3 συνελτικικά στρώματα σε κάθε φάση.

Convolutional Layers	Loss
2 Encoding 2 Decoding	0.0602
3 Encoding 3 Decoding	0.0633
4 Encoding 4 Decoding	0.2312

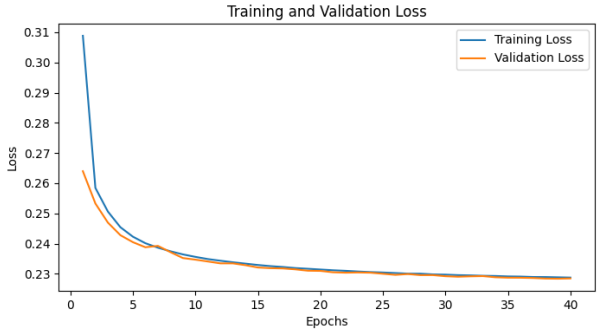
Έπειτα δοκιμάσαμε διαφορετικά downsampling values.Το αρχικό ήταν 2x2 σε κάθε στρώμα το οποίο έβγαζε:



Η παραγόμενη εικόνα είναι δίαστασης (7x7x10) που είναι αρκετά μικρότερη της αρχικής.Έπειτα δοκιμάσαμε να αυξήσουμε το downsampling και δοκιμάσαμε στο πρώτο στρώμα downsampling 4x4 και στο δεύτερο 7x7:



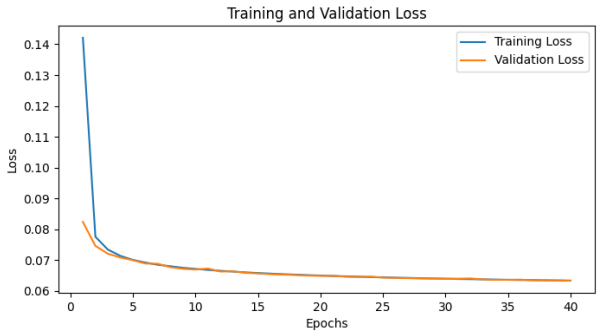
Η παραγόμενη εικόνα εδώ,έχει διάσταση (1x1x10) αλλά το τελικό loss είναι αρκετά μεγαλύτερο.Τέλος δοκιμάσαμε downsampling 2x2 στο πρώτο στρώμα και 7x7 στο δεύτερο:



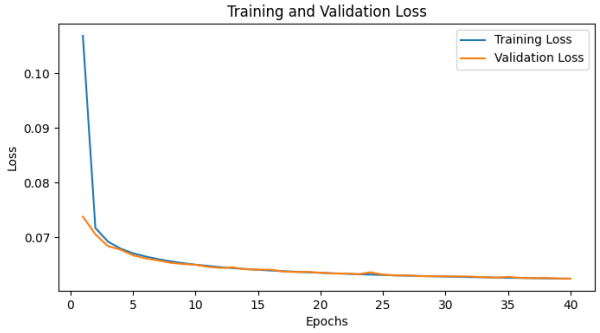
Εδώ η παραγόμενη εικόνα έχει διάσταση (2x2x10) αλλά το loss ενώ καλύτερο από πριν,είναι αρκετά μεγάλο.Συνεπώς προτιμήθηκε downsampling 2x2 σε κάθε στρώμα.Παρακάτω είναι ο πίνακας των πειραμάτων:

Sampling	Loss
2x2,2x2	0.0633
4x4,7x7	0.2683
2x2,7x7	0.2287

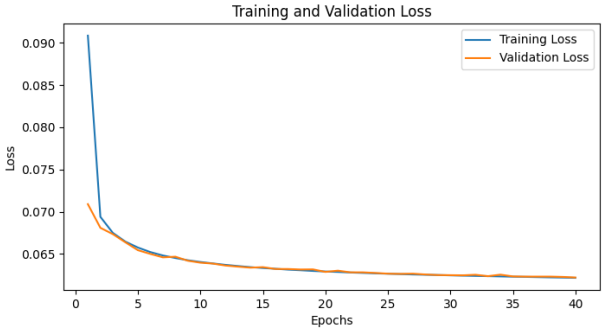
Στη συνέχεια πειραματιστήκαμε με το Batch Size.Συνήθως όσο μικρότερο το batch size,τόσο πιο συχνή η ενημέρωση των βαρών του νευρωνικού δικτύου,όμως υπάρχει επίσης και σοβαρή αύξηση του χρόνου εκπαίδευσης.Μέχρι στιγμής, όλα τα πειράματα έχουν γίνει με batch size 128,το οποίο παράγει το εξής learning curve:



Τα αποτελέσματα είναι αρκετά ικανοποιητικά και σε αρκετά καλούς χρόνους (3 seconds per epoch).Μειώνοντας το batch size σε 64 έχουμε:



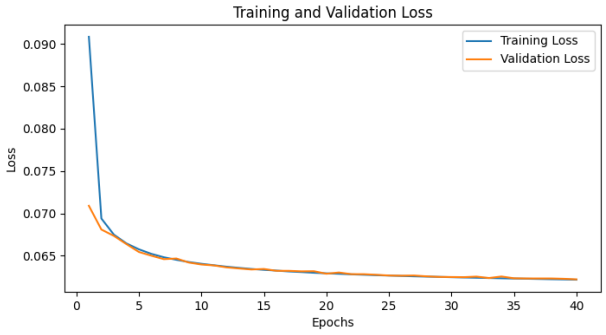
Παρατηρούμε μια καλή βελτίωση αλλά σε διπλασιασμό στον χρόνο κάθε epoch.Τέλος έχουμε για batch size 32:



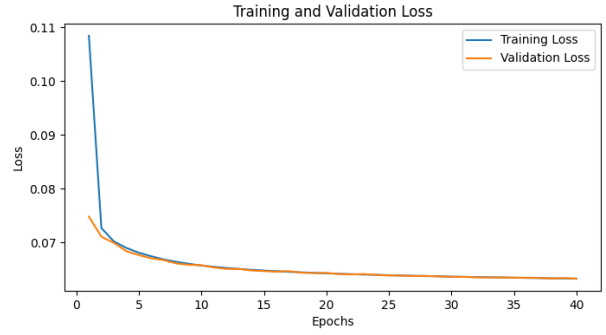
Παρατηρούμε πως η διαφορά σε σχέση με 64 είναι πολύ μικρή (στο loss) αλλά ο χρόνος διπλασιάζεται στα 12 seconds per epoch.Τελικά καταλήγουμε πως το βέλτιστο batch size ,από θέμα χρόνου και loss,είναι το 64.Παρακάτω είναι ο πίνακας των πειραμάτων:

Batch Size	Loss
128	0.0633
64	0.0624
32	0.0622

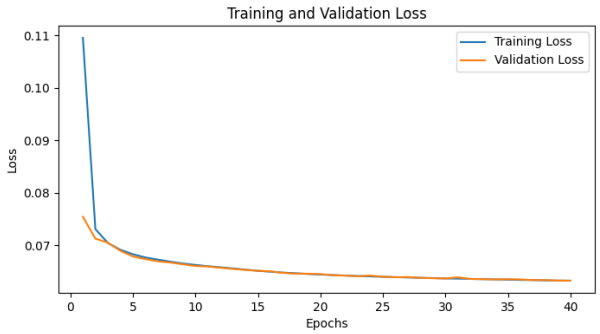
Η τελευταία παράμμετρος με την οποία πειραματιστήκαμε είναι το Latent Dimension,το οποίο αντιπροσωπεύει τον νέο διανυσματικό χώρο στον οποίο θα αναπαρασταθούν οι εικόνες.Μέχρι στιγμής,όλα τα πειράματα είχαν γίνει σε latent dimension 10,με learning curve:



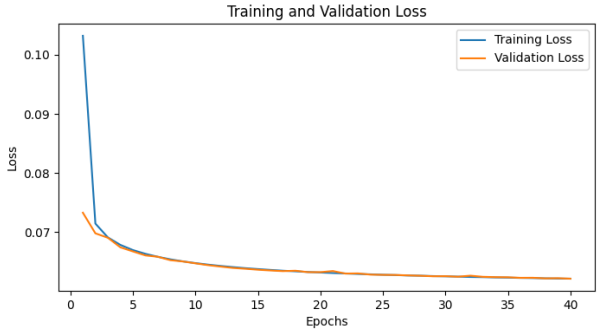
Η τελική διάσταση της εικόνας είναι (7x7x10) με latent dimension 10.Μειώνοντας το latent dimension αρχικά στο 5,έχουμε:



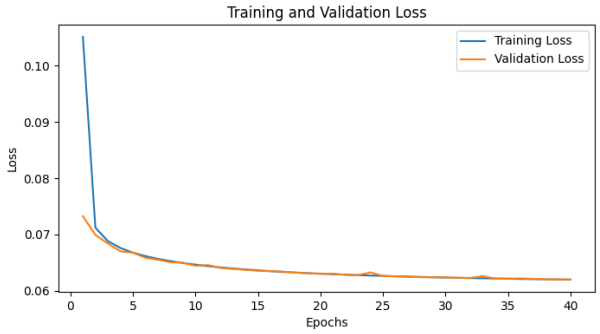
Παρατηρούμε πως αυξάνεται το loss αν και μειώνεται δραματικά το μέγεθος της εικόνας.Συνεχίζουμε με latent dimension 7:



Εδώ δεν παρατηρούμε σημαντική βελτίωση από το latent dimension 5.Έπειτα ακολουθούν latent dimension τιμές μεγαλύτερες του 10,οι οποίες δεν θα χρησιμέψουν ιδιαίτερα,διότι αυξάνεται πολύ το μέγεθος της εικόνας στον νέο διανυσματικό χώρο,και ουσιαστικά δεν πετυγχαίνουμε τον σκοπό μας για την μείωση των διαστάσεων.Για latent dimension 20 έχουμε:



και για 30:

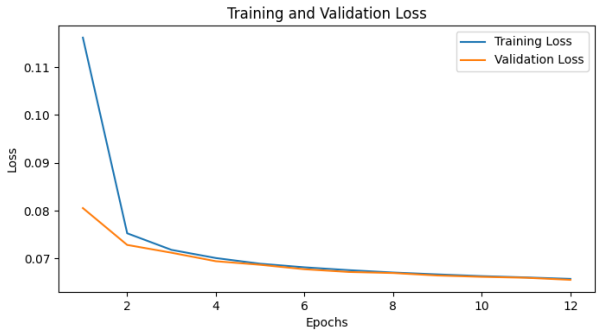


Παρατηρούμε πως όλα τα αποτελέσματα παράγουν αποδεκτά αποτελέσματα,μόνο που πρέπει εκτός από το loss,να λάβουμε υπόψιν και το μέγεθος της εικόνας στον νέο διανυσματικό χώρο.Οι διανυσματικοί χώροι 20,30 δεν αξίζουν γιατί παράγουν

πολύ μεγάλες εικόνες.Η επιλογή του διανυσματικού χώρου δίαστασης 7 δεν αξίζει γιατί έχει 0.0001 λιγότερο loss από το latent dimension 5 και παράγει εικόνα μεγέθους (7x7x7) ενάντι του (7x7x5).Τέλος το latent dimension 10 είναι μία καλή επιλογή,αλλά παράγει εικόνα μεγέθους (7x7x10).Εφόσον ο σκοπός μας είναι η σημαντική μείωση της διάστασης της εικόνας και ταυτόχρονα η ελαχιστοποίηση του σφάλαματος ανακατασκευής(άρα και καλή κωδικοποίηση της εικόνας) προτιμήθηκε latent dimension 5.Παρακάτω είναι ο πίνακας των πειραμάτων:

Latent Dimension	Loss
5	0.0633
7	0.0632
10	0.0624
20	0.0621
30	0.0620

Εκτός από τους απλούς πειραματισμούς των παραμέτρων, έγιναν και κάποιες συνδυαστικές επιλογές για την εύρεση ελαχιστοποίησης του λοσς. Κρατήθηκε ο optimizer Adam για το NN, μειώθηκαν τα εποχης σε 12 και αυξήσαμε το latent dimension σε 20 ώστε να δούμε τη σχέση μεταξύ διαστάσεων και εποχης αντιστρόφως ανάλογα. Παρόλα αυτά, όπως προαναφέραμε, το μεγάλο latent dimension δεν είναι αποτελεσματικό διότι παράγει μεγάλα μεγέθη εικόνας. Όπως φαίνεται στο παρακάτω διάγραμμα:

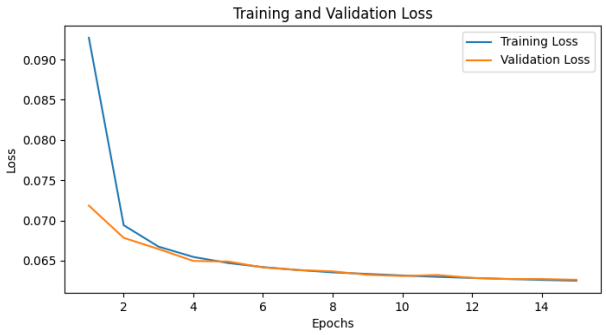


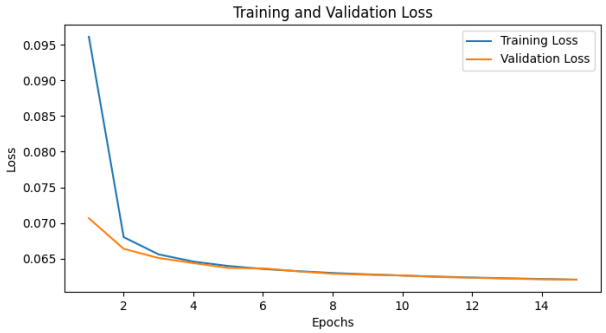
Epochs	Latent Dimension	Loss
12	20	0.0655

Παρατηρούμε ότι, με βάση τα αποτελέσματα, επιβεβαιώνεται ότι το λοσς αυξήθηκε ελάχιστα από τα προηγούμενα καλύτερα αποτελέσματα, παρά τη σημαντική αύξηση της παραμέτρου latent dimension. Αυτό δείχνει και τη βαρύτητα της παραμέτρου epochs στο νευρωνικό δίκτυο μας.

Στη συνέχεια, εστιάζοντας στην παράμετρο batch size, την δοκιμάσαμε με συνδυασμούς άλλων προαναφερθέντων activators. Όπως παρατηρήθηκε νωρίτερα,

η μείωση της παραμέτρου batch size σε 64 και 32 οδήγησε σε σημαντική μείωση του Loss. Έτσι, δοκιμάστηκαν σε Adam optimizer με latent dimension να παραμένει στα 5, ως τιμές που θεωρήθηκαν βέλτιστες από τα προηγούμενα τεστ με batch size 32 και activator elu, ενώ δοκιμάστηκε και batch size 16 με activator gelu.





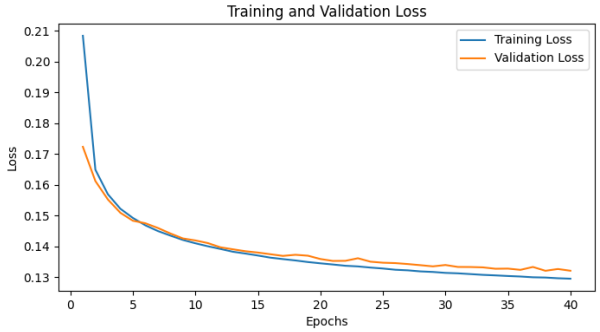
Τα αποτελέσματα ήταν ακόμα πιο βέλτιστα, ειδικά όταν μειώθηκε περαιτέρω σε 16 το batch size (με μικρή βέβαια απόκλιση από την τιμή 32), ενώ επιβεβαιώθηκε για δεύτερη φορά ότι ο gelu φαίνεται να ελαχιστοποιεί ελαφρώς περισσότερο το loss σε σχέση με το elu και νωρίτερες δοκιμές σε relu. Έτσι, καταλήξαμε σε loss 0.062 με καλύτερες παραμέτρους για activator την gelu και batch size 16 (όμως και η 32 είναι μια πολύ καλή τιμή επίσης με ελάχιστη απόκλιση).

Activators	Batch size	Loss
elu	32	0.0626
gelu	16	0.0620

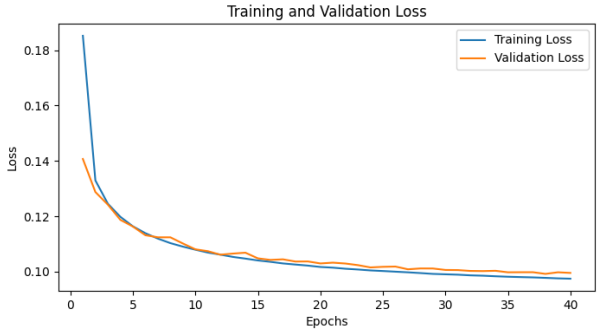
Επομένως επιτεύχθηκε το καλύτερο Loss με τιμή 0.062 χρησιμποιώντας μικρό latent dimension (5) αυτή την φορά, σε αντίθεση με προηγούμενα πειράματα όπου επιτεύχθηκε με τιμή 20. Αυτό καθιστά το νευρωνικό μας δίκτυο πιο αποδοτικό με τις συγκεκριμένες παραμέτρους.

1.3 Τελική Κατασκευή Νευρωνικού Δικτύου

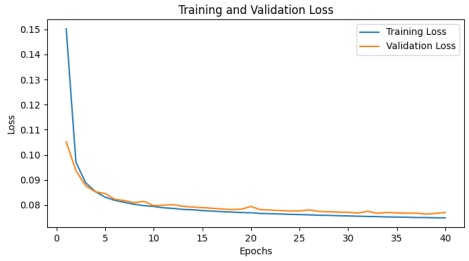
Παρατηρήσαμε πως αν προσθέσουμε ένα dense Layer στη θέση του τελευταίου παράλληλα κρατώντας το τελευταίο Convolutional Layer αυξάνοντας τον αριθμό των φίλτρων σε 128,η είκονα στον νέο χώρο έχει διάσταση Latent Dim.Αρχικά πειραματιστήκαμε με το επιλεγμένο από πριν Latent Dim 5, το οποίο έβγαλε τα εξής αποτελέσματα:



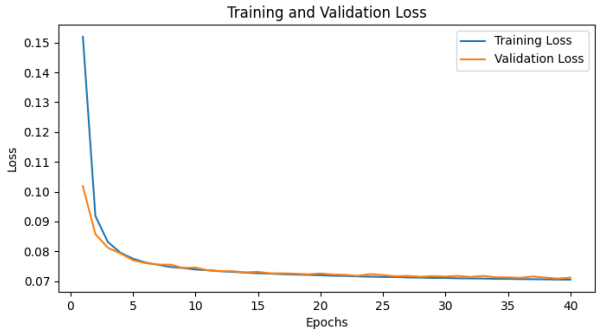
Παρατηρούμε πως συγκριτικά με τα προηγούμενα πειράματά μας,το Loss είναι πολύ μεγάλο.Επομένως αυξάνουμε το Latent Dimension σε 10:



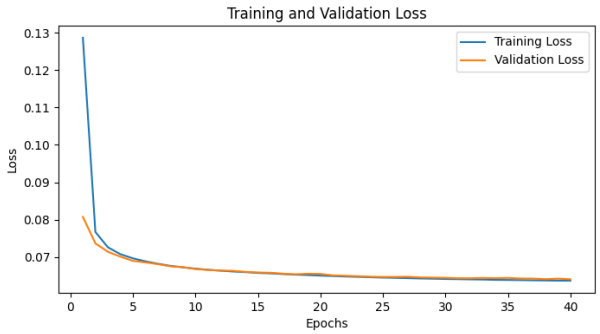
Φαίνεται μία σημαντική μείωση στο Loss αλλά ακόμα είναι πολύ παραπάνω από τα προηγούμενα.Με Latent Dimension 20 έχουμε:



Εδώ βλέπουμε πως το Loss μειώθηκε αρκετά αλλα ακόμα είναι στο 0.0749.Προχωράμε με Latent Dimension 30:



Παρατηρούμε πως τα αποτελέσματα εδώ είναι ιδανικά,μιάς και ενώ σε σχέση με τα πειράματα χωρίς το Dense Layer είχαμε Loss 0.0633,πλέον η διάσταση του νέου χώρου είναι αντί για 245,μόλις 30.Συνεπώς όπως θα φανεί και παρακάτω ,ενώ έχουμε μία πολύ μικρή απόκλιηση,θα έχουμε πολύ μεγαλύτερο κέρδος χρονικά. Συνεπώς επιλέχθηκε ως βέλτιστη διάσταση για τον νέο χώρο.Τέλος,δοκιμάσαμε μία πολύ μεγαλύτερη διάσταση για να δούμε τα αποτελέσματά της.Για Latent Dimension 100 έχουμε:



Φαίνεται ξεκάθαρα πλέον πως πάνω από δίασταση 100,δεν υπάρχει σημαντική βελτίωση στην κωδικοποίηση της εικόνας.Συνεπώς αν δεν μας ένοιαζε το κριτήριο του χρόνου,θα επιλέγαμε Latent Dimension 100.Όμως μιάς και μας νοιάζει ο χρόνος,καταλήξαμε σε Latent Dimension 30.Παρακάτω είναι ο πίνακας των πειραμάτων:

Latent Dimension	Loss
5	0.1295
10	0.0975
20	0.0749
30	0.0705
100	0.0636

Latent Space	Normal Space
GNNS	GNNS
MRNG	MRNG
BRUTE	BRUTE
MRNG	LSH
MRNG	HYPERCUBE

2 Ιδιότητες αρχικού και νέου διανυσματικού χώρου

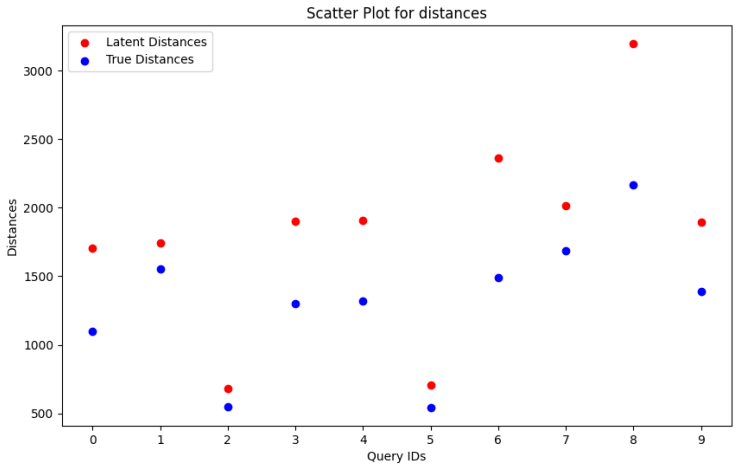
Η ενότητα αυτή έχει ως στόχο την πλήρη παρουσίαση των αποτελεσμάτων των προσεγγιστικών αλγορίθμων και της εξαντλητικής αναζήτησης, με τη χρήση του νέου διανυσματικού χώρου που παρήγαγε το νευρωνικό δίκτυο της προηγούμενης ενότητας. Επιπλέον, αξιολογείται η απόδοσή τους ως προς την απόσταση κοντινότερου γείτονα, τον χρόνο και τη μέση απόσταση των προσεγγιστικών αλγορίθμων σε σχέση με τον πραγματικό πλησιέστερο γείτονα στον αρχικό χώρο. Η παράθεση αυτών των αποτελεσμάτων έχει επίσης σκοπό τον σχολιασμό και την εξαγωγή συμπερασμάτων.

2.1 Διαφορές των αλγορίθμων στους δύο χώρους

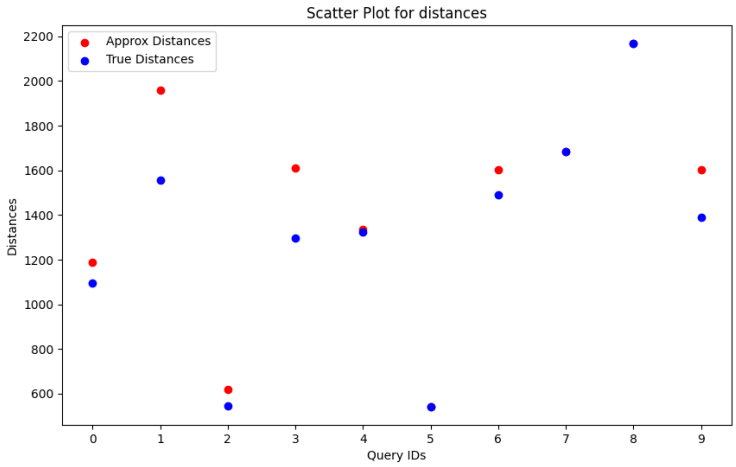
Σε αυτή την υποενότητα, θα μελετήσουμε τις συσχετίσεις που έχουν οι προσεγγιστικοί αλγόριθμοι αλλά και ο εξαντλητικός όσον αφορά τους διανυσματικούς χώρους. Θα εξετάσουμε την απόδοσή τους τόσο σε διαστασιακή κοντινότερου γείτονα όσο και ως προς τον χρόνο που έκαναν συγκριτικά με τον αρχικό και το νέο χώρο.

2.1.1 GNNS

Η περίπτωση αυτή έχει την εξέταση της απόδοσης του GNNS στο νέο διανυσματικό χώρο. Αυτό επιτυγχάνεται συγκρίνοντας την απόσταση του πλησιέστερου γείτονα που βρήκε προσεγγιστικά ο GNNS στον σμικρυσμένο διανυσματικό χώρο με την απόσταση του πλησιεστέρου γείτονα που βρήκε ο αλγόριθμος του Lloyd για το ίδιο query point στον αρχικό χώρο. Στο παρακάτω διάγραμμα παρατηρείται η συσχέτισή τους, όπου ο GNNS είναι χείριστος στις αποστάσεις, αλλά όχι κατά πολύ.

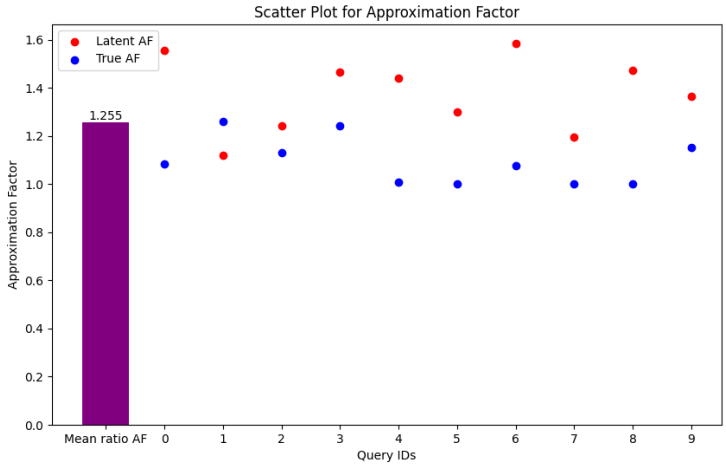


Το τελευταίο πόρισμα αποτυπώνεται καλύτερα παίρνοντας τον λόγο των δύο αυτών αποστάσεων, δημιουργώντας έτσι το μέσο κλάσμα προσέγγισης (AF) του νέου διανυσματικού χώρου για κάθε query point ID του εξαγόμενου διανυσματικού χώρου. Επιπλέον, χρησιμοποιώντας τις αποστάσεις του πλησιέστερου γείτονα που βρήκε προσεγγιστικά ο GNNS στον κανονικό χώρο με την απόσταση του πλησιεστέρου γείτονα που βρήκε ο αλγόριθμος του Lloyd στον αρχικό χώρο (βλέπε επόμενο πινακά).

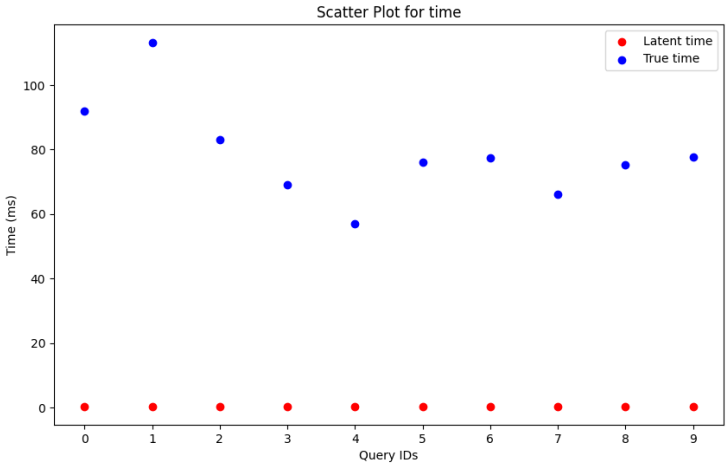


Από τον λόγο των δύο αποστάσεων αυτών προκύπτουν και τα AF για τον κανονικό χώρο. Γνωρίζοντας κάθε query point και για τους δύο διανυσματικούς χώρους, μπορούμε, με το κλάσμα των μέσων τιμών τους, να βρούμε το μεσο κλάσμα προσέγγισης του προσεγγιστικού αλγορίθμου ως προς τον Lloyd's στον

αρχικό διανυσματικό χώρο. Τα στοιχεία αυτά αποτυπώνονται στον επόμενο πίνακα, όπου για τον GNNS είχαμε μέση τιμή των AF στο 1.255.



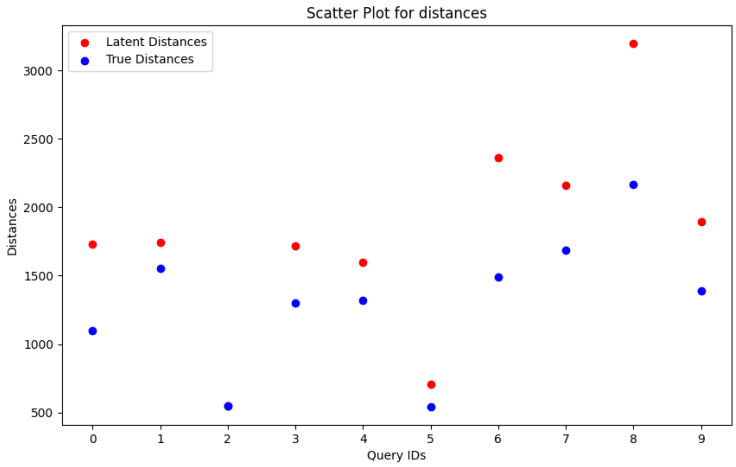
Όσον αφορά τον χρόνο στο νέο διανυσματικό χώρο, ο GNNS ήταν κατα πολυ πιο γρήγορος, όπου κατά μέσο όρο κάθε query point ID έπαιρνε 0.3 ms, ενώ στον κανονικό χώρο έπαιρνε τιμές από 2 ms ως 6 ms. Επίσης, ήταν αρκετά γρηγορότερος επίσης από την εξαντλητική αναζήτηση στον κανονικό χώρο, όπως θα δούμε και στον επόμενο πίνακα.



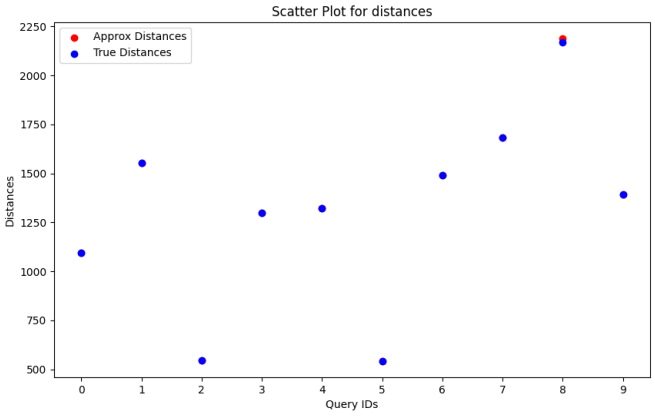
2.1.2 MRNG

Σε αυτήν την υποενότητα , εξετάζουμε πόσο καλά προσεγγίζει ο αλγόριθμος MRNG τον πλησιέστερο γείτονα στον νέο , μικρότερης διάστασης διανυσματικό

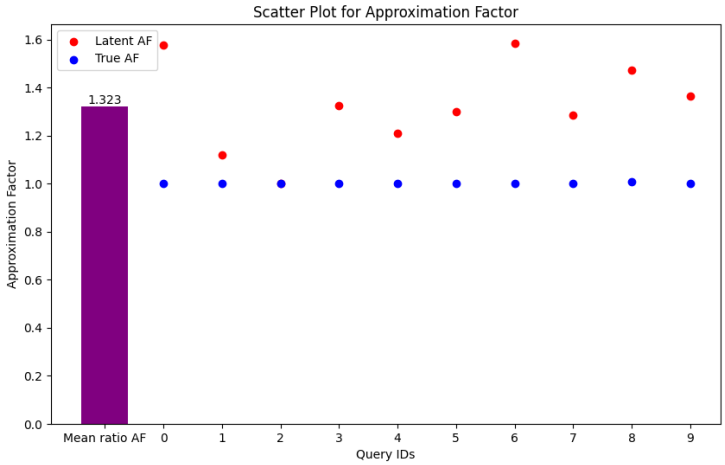
χώρο, σε σχέση με τον αλγόριθμο του Bruteforce στον αρχικό χώρο. Αναλύουμε τις αποστάσεις που προσεγγίζονται από τους δύο αλγορίθμους και παρατηρούμε ότι πέρα από οτι ο MRNG δεν είναι ο καλύτερος στις αποστάσεις, υστερεί πιο πολύ απο τον GNNS.



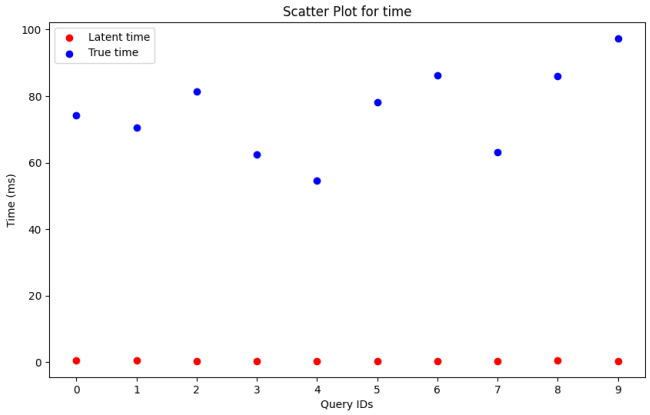
Το συμπέρασμα αυτό αναδεικνύεται καλύτερα μέσω του λόγου των δύο αποστάσεων, δημιουργώντας ένα μέσο κλάσμα προσέγγισης (AF) για κάθε query point ID στον νέο διανυσματικό χώρο. Μελετώντας επίσης τον ιδιό λόγο, δηλαδή τις αποστάσεις του πλησιέστερου γείτονα που προσεγγίζει προσεγγιστικά ο αλγόριθμος MRNG στον κανονικό χώρο με τις αποστάσεις του πλησιεστέρου γείτονα που προσεγγίζει ο αλγόριθμος του Lloyd στον αρχικό χώρο (βλ. επόμενο πίνακα), βρίσκουμε την απόδοση του MRNG στους δύο αυτούς χώρους.Ο επόμενος πίνακας δείχνει την εξαιρετική ακριβεία του MRNG στον αρχικό χώρο.



Λαμβάνοντας τις αποστάσεις αυτές για κάθε query point στους δύο διανυσματικούς χώρους, μπορούμε να υπολογίσουμε το μέσο κλάσμα προσέγγισης του προσεγγιστικού αλγορίθμου MRNG σε σχέση με τον αλγόριθμο του Lloyd στον αρχικό διανυσματικό χώρο. Τα αποτελέσματα αυτά παρουσιάζονται στον παρακάτω πίνακα, όπου η μέση τιμή των AF για τον MRNG ήταν 1.323 γεγονός που αποδεικνυεί την καλύτερη απόδοση του αλγόριθμου στον αρχικό χώρο σε σημαντικό βαθμό.



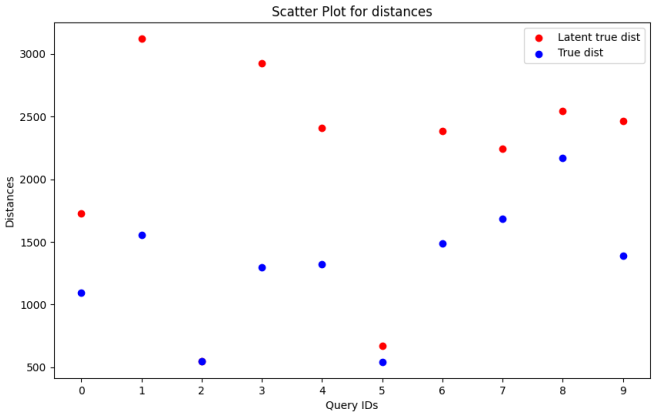
Σχετικά με το χρόνο, που αποτελεί το μεγάλο μειονέκτημα της MRNG παρά την απώλεια ακρίβειας στο νέο διανυσματικό χώρο, παρατηρείται σημαντική μείωση στον χρόνο αναζήτησης του πλησιέστερου γείτονα, παρόμοια με τον GNNS. Συγκεκριμένα, κατά μέσο όρο, για κάθε query point, απαιτείται μόλις 0.4 ms στον νέο διανυσματικό χώρο, σε σύγκριση με τον αρχικό χώρο που απαιτούσε μεταξύ 2 ms και 5 ms.



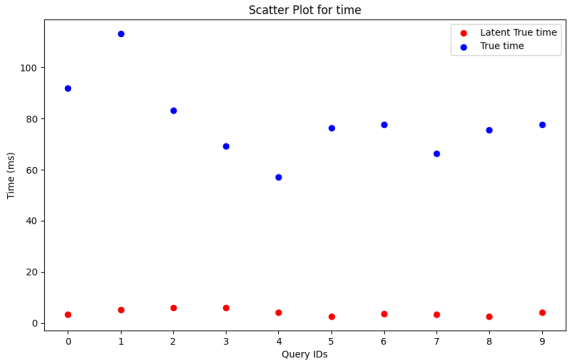
2.1.3 Εξαντλητικής αναζήτησης (Bruteforce)

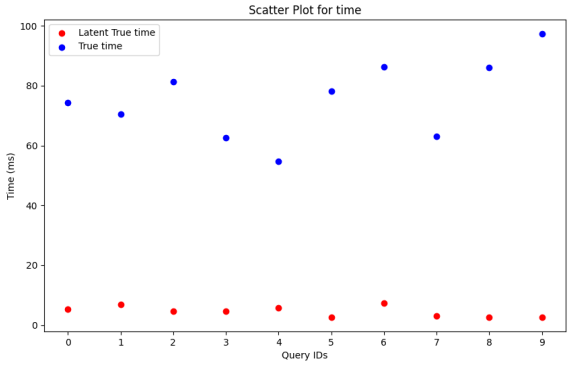
Σειρά έχει η σύγκριση της εξαντλητικής αναζήτησης στο πόσο αποτελεσματικά προσεγγίζει τον πλησιέστερο γείτονα στον νέο χώρο σε σύγκριση με τον κανονικό. Πέρα από τις αποστάσεις στους δύο χώρους, θα εξετάσουμε επίσης τους χρόνους εκτέλεσης του αλγορίθμου στους δύο γνωστούς χώρους

Ο αλγόριθμος bruteforce εξήγαγε αποτελέσματα με χαμηλότερη ακρίβεια στο νέο διανυσματικό χώρο, όπως αποδεικνύεται σαφέστατα στο επόμενο διάγραμμα. Για κάθε query point, παρατηρούμε αυξημένες τιμές, από 20% έως περιπού 100%, διπλάσιες δηλαδή αποστάσεις απο τον κανονικό.



Η σημαντική μείωση των χρονικών αποδόσεων αποτελεί ένα σημαντικό πλεονέκτημα του νέου διανυσματικού χώρου. Και με τη χρήση του GNNS και του MRNG στο νέο χώρο, παρατηρούμε ότι, ενώ στον κανονικό χώρο οι χρόνοι εκτέλεσης είναι της τάξεως των 40 ms έως 100+ ms, στο νέο χώρο κυμαίνονται από 2 ms έως 8 ms. Επιπλέον, όπως φαίνεται στις παρακάτω δύο εικόνες, τα χρονικά δεδομένα του αλγορίθμου Lloyd's είναι παρόμοια είτε εκτελείται με τον GNNS είτε με τον MRNG προσεγγιστικό αλγόριθμο για το νεο διανυσματικό χώρο.



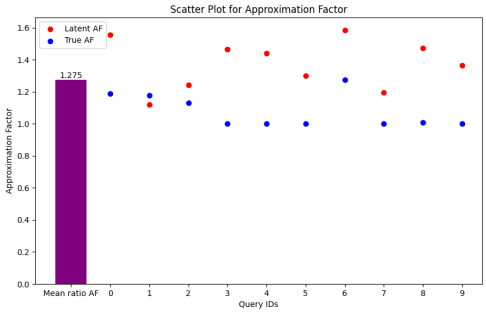


2.2 Διαφορές των προσεγγιστικων στο νεο χώρο σε σχέση με LSH & Hypercube

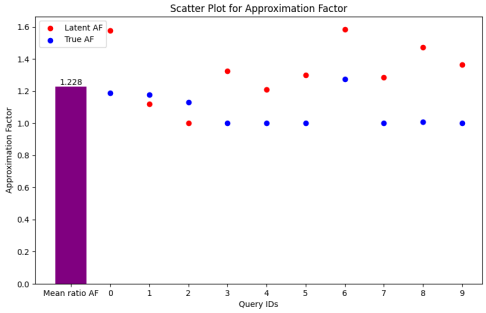
Εκτός από τους προσεγγιστικούς αλγόριθμους MRNG, GNNS και την εξαντλητική αναζήτηση, υπάρχουν επίσης οι αλγόριθμοι LSH και Hypercube. Στη συνέχεια, θα εκτελέσουμε τους προσεγγιστικούς αλγόριθμους στο νέο διανυσματικό χώρο και θα τους συγκρίνουμε με τους αλγορίθμους LSH και Hypercube στον κανονικό χώρο ως προς την απόδοση, χρησιμοποιώντας το μέσο κλάσμα προσέγγισης.

2.2.1 LSH

Θα αναλύσουμε πρώτα την LSH χρησιμοποιώντας τον GNNS και υπολογίζοντας το μέσο όρο των λόγων των αποστάσεων του GNNS στο νέο χώρο σε σχέση με τις αποστάσεις της εξαντλητικής αναζήτησης στον κανονικό χώρο. Αντίστοιχα, υπολογίζουμε το μέσο όρο των λόγων των αποστάσεων της LSH και των αποστάσεων του Lloyds στον κανονικό χώρο. Έτσι, προκύπτουν τα δύο (average AF). Καθώς παίρνουμε το κλάσμα αυτών των δύο τιμών, προκύπτει η απόδοση: 1.275. Αυτό φαίνεται και στην παρακάτω εικόνα. Από αυτό τον αριθμό συμπεραίνουμε ότι ο GNNS στο νέο χώρο παράγει μεγαλύτερες αποστάσεις κατά μέσο όρο κατά 27,5% σε σύγκριση με τον LSH.

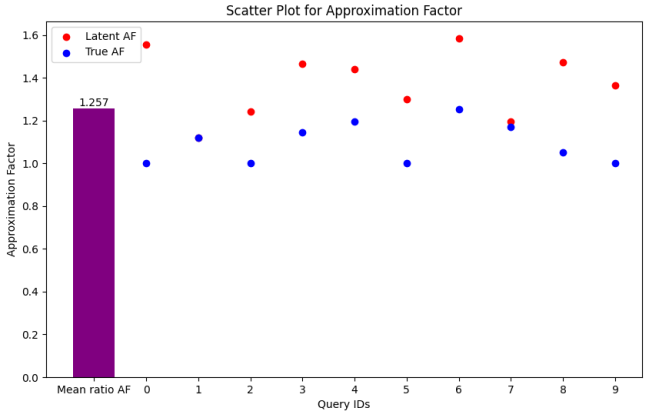


Αντίστοιχα, υπολογίζουμε τον μέσο όρο των (AF) για την MRNG, όπως και με τον GNNS, και τον συγκρίνουμε με τον μέσο όρο των AF της LSH στον κανονικό χώρο. Η τιμή που προκύπτει είναι 1.228, γεγονός που δείχνει ότι η MRNG στο νέο χώρο είναι κατά μέσο όρο 22,8% χειρότερη από την LSH. Τουλάχιστον, είναι λιγότερο χειρότερη από την LSH σε σύγκριση με τον GNNS.

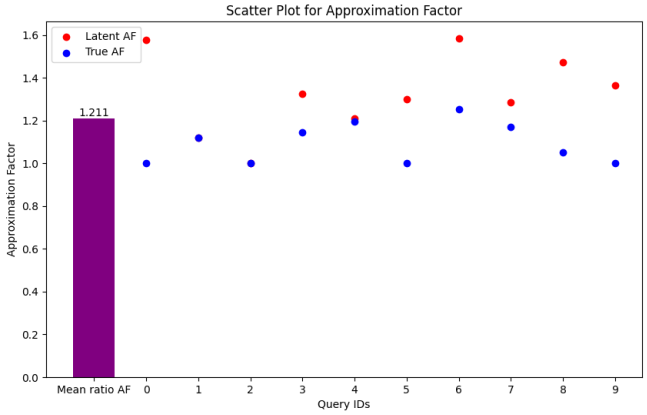


2.2.2 Hypercube

Σχετικά με τον Hypercube αλγόριθμο , ακολουθούμαι την ίδια διαδικασία για την σύγκριση με τον GNNS . Υπολογίζουμε το μέσο όρο των λόγων των αποστάσεων του GNNS στο νέο χώρο σε σχέση με τις αποστάσεις της εξαντλητικής αναζήτησης στον κανονικό χώρο. Στη συνέχεια, υπολογίζουμε τον μέσο όρο των λόγων των αποστάσεων της Hypercube και των αποστάσεων του Bruteforce στον κανονικό χώρο. Με βάση αυτά, προκύπτουν τα δύο (average AF). Κατά τη διάρκεια της ανάλυσης, παίρνουμε το κλάσμα αυτών των δύο τιμών, προκειμένου να προκύψει η συγκεκριμένη απόδοση, η οποία είναι 1.257 (βλ. επόμενο πίνακα). Αυτό αποδεικνύει ότι ο GNNS στο νέο χώρο δημιουργεί μεγαλύτερες αποστάσεις κατά μέσο όρο περίπου 25,7%, σε σύγκριση με την Hypercube.



Αντίστοιχα, εξετάζοντας τον μέσο όρο των AF της MRNG στην latent dimension και αντίστοιχα των Lloyd's στον αρχικό, καταλήγουμε στο μέσο κλάσμα προσέγγισης 1.211. Αυτό σημαίνει ότι ο MRNG βρίσκει τους πλησιέστερους γείτονες κατά μέσο όρο με 21,1% μεγαλύτερες αποστάσεις σε σύγκριση με τον Hypercube στον αρχικό χώρο. Ενδιαφέρον παρουσιάζει το γεγονός ότι ο MRNG, πάλι σε σύγκριση με τον GNNS, δεν έχει τόσο σημαντική χαμηλότερη απόδοση απέναντι στον αλγόριθμο του Hypercube.



Υποσημείωση: Όλα τα παραπάνω αναφερόμενα αποτελέσματα έχουν προκύψει χρησιμοποιώντας δεδομένα εισόδου με μέγεθος 2000 εικόνων, για λόγους εξοικονόμησης χρόνου (ιδιαιτέρως για τον MRNG αλγόριθμο στον αρχικό δινυσματικό χώρο).

3 Συσταδοποίηση στους 2 χώρους

Σε αυτό το κεφάλαιο θα ασχοληθούμε με την απόδοση του αλγορίθμου συσταδοποίησης,στους 2 διανυσματικούς χώρους.Ειδικότερα θα συγκρίνουμε την απόδοση ως προς την Silhouette,την Objective Function και τον χρόνο συσταδοποίησης.Οι συσταδοποιήσεις και στις 2 περιπτώσεις γίνανε χρησιμοποιώντας τον αλγόριθμο της εξαντλητικής αναζήτησης σε 10,000 εικόνες και οι εικόνες συσταδοποιήθηκαν σε 10 συστάδες.

3.1 Συσταδοποίηση στον νέο χώρο

Στην συσταδοποίηση στον νέο χώρο,ο χρόνος συσταδοποίησης είναι : 0.655015 seconds.Ο χρόνος αυτός είναι πάρα πολύ μικρός και είναι αρκετά αποδοτικός. Επίσης η συνάρτηση αποτίμησης στόχου (Objective Function) έχει τιμή: 4.10524e+10.Είναι πολύ μεγάλη τιμή και όπως θα δούμε συγκριτικά με την συσταδοποίηση στον αρχικό χώρο,είναι αρκετά μεγαλύτερη από αυτήν του αρχικού χώρου.Τέλος η μέση τιμή της Silhouette είναι: 0.0461508 που υποδεικνύει μία μέτρια συσταδοποίηση,

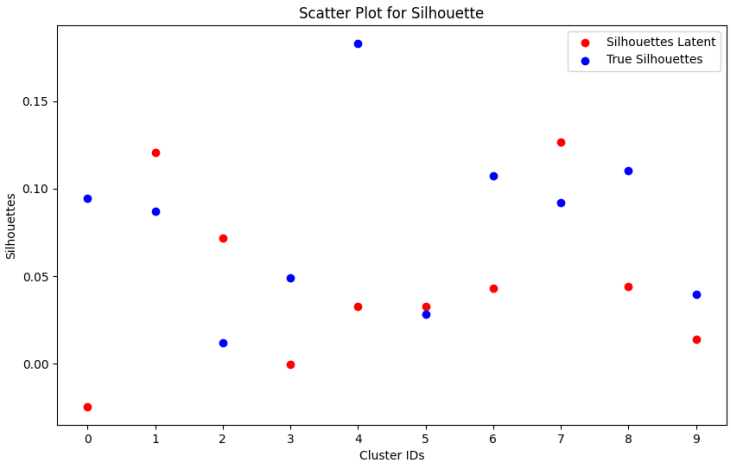
που κάποια σημεία μπορεί να μπορούσαν να ανατεθούν σε άλλη συστάδα,αλλά δεν αξίζει από θέμα κόστους.

3.2 Συσταδοποίηση στον αρχικό χώρο

Η συσταδοποίηση στον αρχικό χώρο εκτελέστηκε σε : 21.9788 seconds.Είναι σημαντικά μεγαλύτερος χρόνος από αυτόν του νέου χώρου.Η τιμή της συνάρτησης αποτίμησης στόχου (Objective Function) είναι: 159244.Τέλος η μέση τιμή της Silhouette είναι : 0.0803209,το οποίο πάλι υποδεικνύει μία μέτρια συσταδοποίηση,χωρίς όμως να αξίζει να γίνουν αλλαγές.

3.3 Σύγκριση Συσταδοποιήσεων

Αρχικά παρατηρούμε ότι η συσταδοποίηση στον αρχικό μας χώρο είναι καλύτερη σε θέμα ανάθεσης σημείων στο σωστό κέντρο,μιάς και η τιμή της Silhouette της είναι σχεδόν διπλάσια, σε σχέση με τον νέο διανυσματικό μας χώρο.Επίσης η συνάρτηση αποτίμησης στόχου (Objective Function) είναι δραματικά μικρότερη (159244 ενώ στον νέο χώρο είναι 4.10524e+10).Τέλος, ο χρόνος συσταδοποίησης στον αρχικό χώρο είναι αρκετά μεγαλύτερος,κατά 3,355 % που είναι μία σημαντική διαφορά. Τα συμπεράσματα από αυτά είναι πως σε μεγάλο πλήθος δεδομένων όπου αυξάνεται σημαντικά ο χρόνος,η χρήση ενός διανυσματικού χώρου μειωμένης διάστασης, είναι χρήσιμη,ενώ σε μεσαίο και μικρό πλήθος δεδομένων ,είναι προτιμητέα η χρήση του αρχικού διανυσματικού χώρου.Παρακάτω είναι και η σύγκριση των τιμών Silhouette για τους 2 χώρους:



Παρατηρούμε πως η Silhouette σε πολλές συστάδες στον νέο διανυσματικό χώρο έχει παρόμοιες ή και καλύτερες τιμές, αλλά συνολικά η Silhouette στον αρχικό χώρο έχει καλύτερα αποτελέσματα,πράγμα το οποίο φαίνεται και από τη μέση τιμής της,η οποία αναφέρθηκε παραπάνω.Παρακάτω είναι ο πίνακας συγκρίσεων:

Cluster ID	Silhouette	Silhouette Latent
0	0.0944434	-0.0244527
1	0.0870551	0.120654
2	0.0121154	0.0718743
3	0.0491916	-0.00016935
4	0.182871	0.0327459
5	0.0282508	0.0328352
6	0.107397	0.0431995
7	0.0921531	0.126622
8	0.110062	0.0440974
9	0.0396697	0.0141014

Φαίνεται ξεκάθαρα από τον παραπάνω πίνακα πως η ποιότητα συσταδοποίησης στον αρχικό χώρο είναι καλύτερη,μιας και υπάρχουν περιπτώσεις που η τιμή της Silhouette στον νέο διανυσματικό χώρο είναι αρνητική,γεγονός που υποδεικνύει πως υπάρχει καλύτερη συστάδα από αυτήν στην οποία βρίσκεται. Βέβαια όταν οι τιμές Silhouette είναι πολύ κοντά στο 0,σημαίνει πως μπορεί να υπάρχει καλύτερη συστάδα για να ενταχθεί το συγκεκριμένο σημείο,αλλά δεν αξίζει από θέμα κόστους να γίνει η αλλαγή.