Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα

Εργασία 3, Χειμερινό Εξάμηνο 2023-24

Δρακοπούλου Ευγενία, sdi1900054

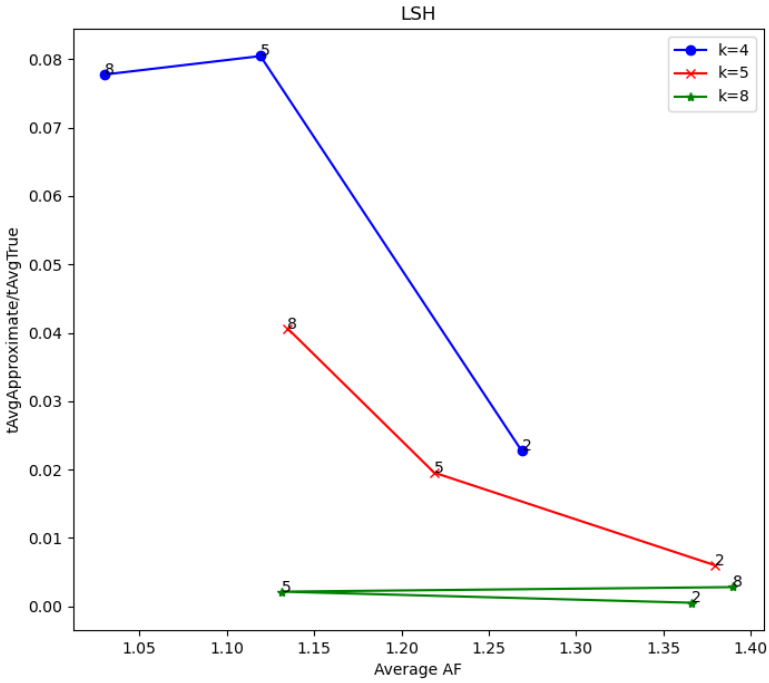
Κοκκινάκη Χριστίνα, sdi2000083

Ερώτημα Β - Αναφορά

Όλα τα πειράματα έχουν γίνει με dataset αυτό των 60000 σημείων και σύνολο αναζήτησης αυτό των 10000 σημείων.

LSH

Για την αξιολόγηση της επίδοσης του αλγορίθμου LSH, πραγματοποιήσαμε διάφορα πειράματα δοκιμάζοντας διαφορετικές τιμές των παραμέτρων k (πλήθος συναρτήσεων h) και L (αριθμός πινάκων κατακερματισμού).



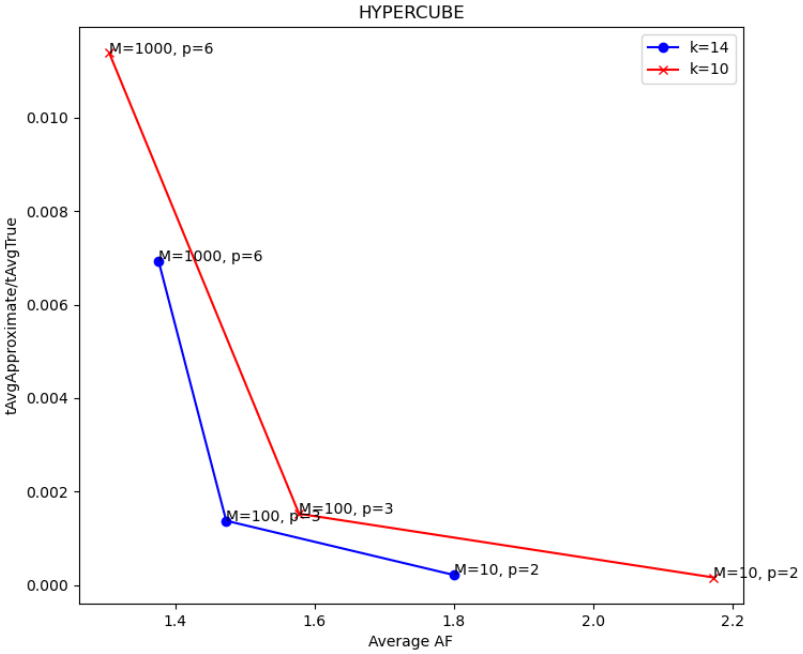
Το N είναι σταθερό σε όλα τα πειράματα και ίσο με 1.

Στο διπλανό διάγραμμα, σε κάθε καμπύλη έχουμε κρατήσει σταθερό το k και μεταβάλλεται η τιμή του L. Οι άξονες αντιπροσωπεύουν τις τιμές Average AF (Average Approximation Factor) και tAverageApproximate/tAverageTrue (χρόνος).

Γενικά παρατηρούμε ότι όσο αυξάνεται το L, το Average AF πλησιάζει όλο και περισσότερο στο 1, με το μειονέκτημα όμως ότι πολλές φορές ο χρόνος αυξάνεται σημαντικά (π.χ. για k=4).

Από το διάγραμμα συμπεραίνουμε ότι οι βέλτιστοι παράμετροι για τον αλγόριθμο LSH είναι οι k=8, L=5 καθώς έχει AAF= 1.13162 (άρα πλησιάζει το 1) και χρόνο≈0.0021 .

Hypercube



Για την αξιολόγηση της επίδοσης του αλγορίθμου Hypercube, πραγματοποιήσαμε διάφορα πειράματα δοκιμάζοντας διαφορετικές τιμές των παραμέτρων k (διάσταση που προβάλλονται τα σημεία), M (πλήθος σημείων που θα ελεγχθούν) και probes (πλήθος κορυφών που θα ελεγχθούν).

Το N είναι σταθερό σε όλα τα πειράματα και ίσο με 1.

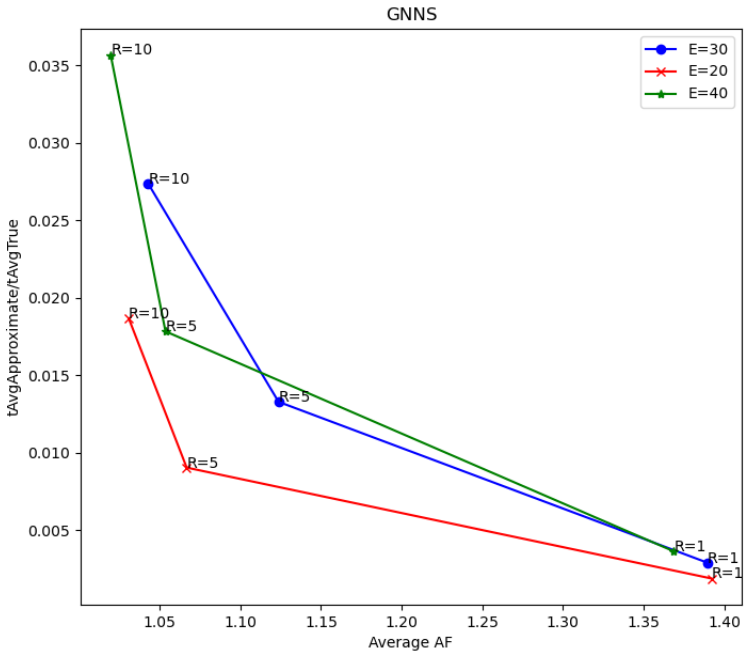
Στο διπλανό διάγραμμα, σε κάθε καμπύλη έχουμε κρατήσει σταθερό το k και μεταβάλλονται οι τιμές των M και probes. Οι άξονες αντιπροσωπεύουν τις τιμές Average AF (Average Approximation Factor) και tAverageApproximate/tAverageTrue (χρόνος).

Γενικά παρατηρούμε ότι όσο αυξάνονται τα σημεία M και οι κορυφές probes που θα ελεγχθούν, το AAF πλησιάζει όλο και περισσότερο στο 1, με το μειονέκτημα όμως ότι πολλές φορές ο χρόνος αυξάνεται σημαντικά.

Από το διάγραμμα συμπεραίνουμε ότι οι βέλτιστοι παράμετροι για τον αλγόριθμο Hypercube είναι οι k=14, M=100, probes=3 καθώς έχει AAF= 1.47338 και χρόνο≈0.0014 . Άλλος ένας συνδυασμός με παρόμοιο AAF είναι για k=14, M=1000, probes=6 (AAF= 1.37643) όμως οι επιπλέον έλεγχοι σημείων/κορυφών επιβαρύνουν την ταχύτητα του αλγορίθμου.

GNNS

Για την αξιολόγηση της επίδοσης του αλγορίθμου GNNS, πραγματοποιήσαμε διάφορα πειράματα δοκιμάζοντας διαφορετικές τιμές των παραμέτρων E (expansions) και R (random restarts).



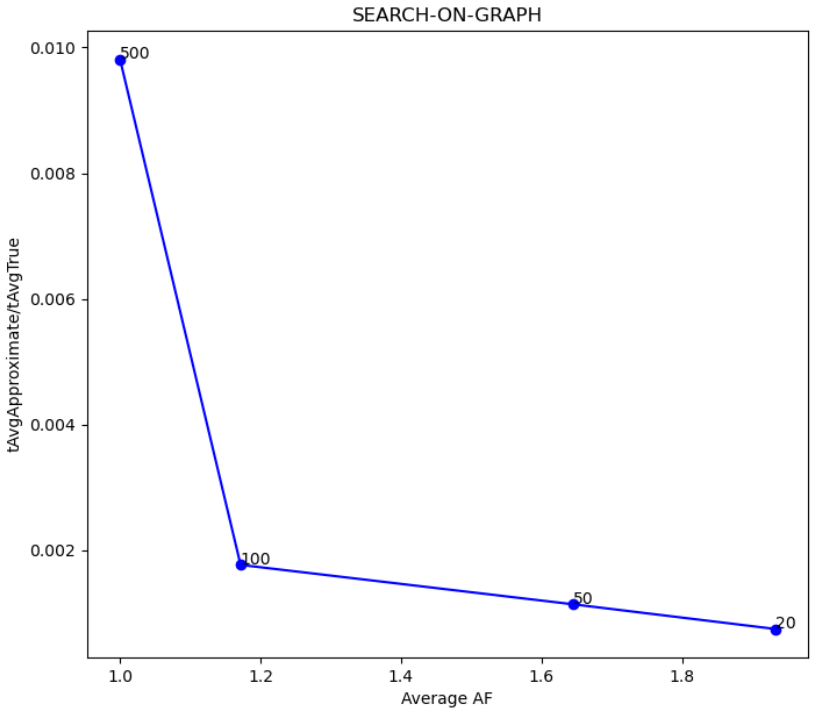
Το N (πλήθος γειτόνων) είναι σταθερό σε όλα τα πειράματα και ίσο με 1. Ομοίως το k (πλήθος πλησιέστερων γειτόνων στο γράφο k-NN) είναι σταθερό και ίσο με 50.

Στο διπλανό διάγραμμα, σε κάθε καμπύλη έχουμε κρατήσει σταθερό το E και μεταβάλλεται η τιμή του R. Οι άξονες αντιπροσωπεύουν τις τιμές Average AF (Average Approximation Factor) και tAverageApproximate/tAverageTrue (χρόνος).

Γενικά παρατηρούμε ότι όσο εκτελούνται περισσότερες επαναλήψεις του αλγορίθμου (λόγω του R), το AAF πλησιάζει όλο και περισσότερο στο 1, με το μειονέκτημα όμως ότι πολλές φορές ο χρόνος αυξάνεται σημαντικά. Για όλους τους συνδυασμούς, ο αλγόριθμος δίνει ικανοποιητικά αποτελέσματα τόσο ως προς το AAF (αφού είναι μικρότερο του 2), όσο και ως προς το χρόνο.

Από το διάγραμμα συμπεραίνουμε ότι οι βέλτιστοι παράμετροι για τον αλγόριθμο GNNS είναι οι E=20, R=5 καθώς έχει AAF= 1.0667 και χρόνο≈0.009 . Για E=20 φαίνεται να υπάρχει μια βελτίωση στο AAF χωρίς να υπάρχει σημαντική αύξηση στο χρόνο όπως γίνεται π.χ. για E=40.

SEARCH-ON-GRAPH (με ευρετήριο MRNG)



Για την αξιολόγηση της επίδοσης του αλγορίθμου SEARCH ON GRAPH με χρήση MRNG ως ευρετήριο, πραγματοποιήσαμε διάφορα πειράματα δοκιμάζοντας διαφορετικές τιμές της παραμέτρου l (πλήθος υποψηφίων).

Το N (πλήθος γειτόνων) είναι σταθερό σε όλα τα πειράματα και ίσο με 1.

Οι άξονες αντιπροσωπεύουν τις τιμές Average AF (Average Approximation Factor) και tAverageApproximate /tAverageTrue (χρόνος).

Γενικά παρατηρούμε ότι από l=20 έως l=100, υπάρχει σημαντική βελτίωση του AAF με ελάχιστη αύξηση του χρόνου, ενώ στο διάστημα από l=100 έως l=500, ο χρόνος αυξάνεται αρκετά.

Από το διάγραμμα συμπεραίνουμε ότι η βέλτιστη παράμετρος για τον αλγόριθμο SEARCH ON GRAPH είναι η l=100 καθώς έχει AAF= 1.17165 και χρόνο≈ 0.0017 . Επιπλέον, για l=500 αν και το AAF=1 (δηλαδή ο αλγόριθμος εντοπίζει επιτυχώς τον πραγματικά πλησιέστερο γείτονα), συγκριτικά με τη βέλτιστη παράμετρο παρουσιάζει αύξηση στο χρόνο εκτέλεσης του.

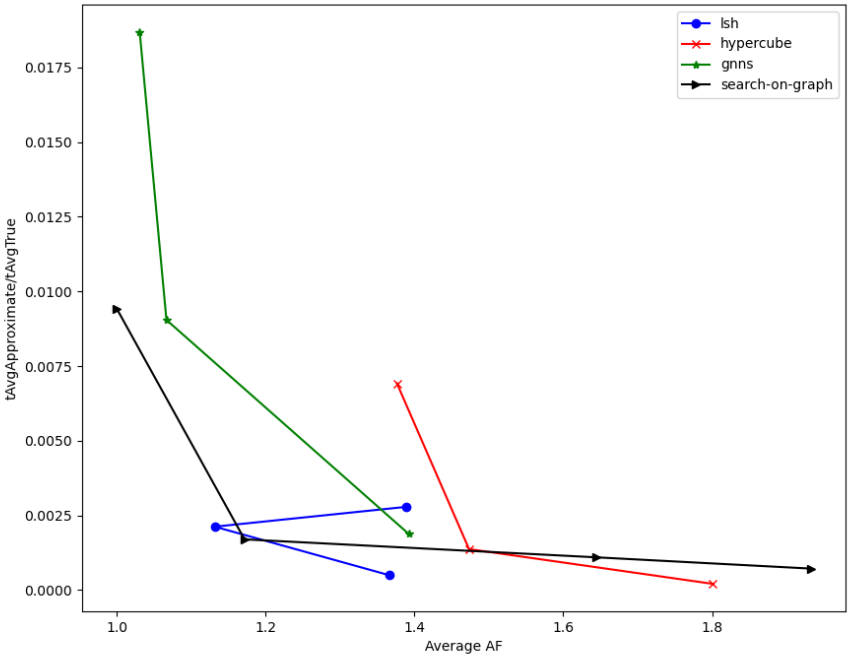
ΣΥΓΚΡΙΣΗ ΑΛΓΟΡΙΘΜΩΝ

Για τη σύγκριση των αλγορίθμων, έχουμε χρησιμοποιήσει τις καμπύλες με τις καλύτερες παραμέτρους.
Συγκεκριμένα:

LSH: k=8

HYPERCUBE: k=14

GNNS: E=20



Από το διπλανό διάγραμμα, παρατηρούμε ότι ο μόνος αλγόριθμος που έχει AAF=1 (βέλτιστη τιμή) είναι ο SEARCH-ON-GRAPH.

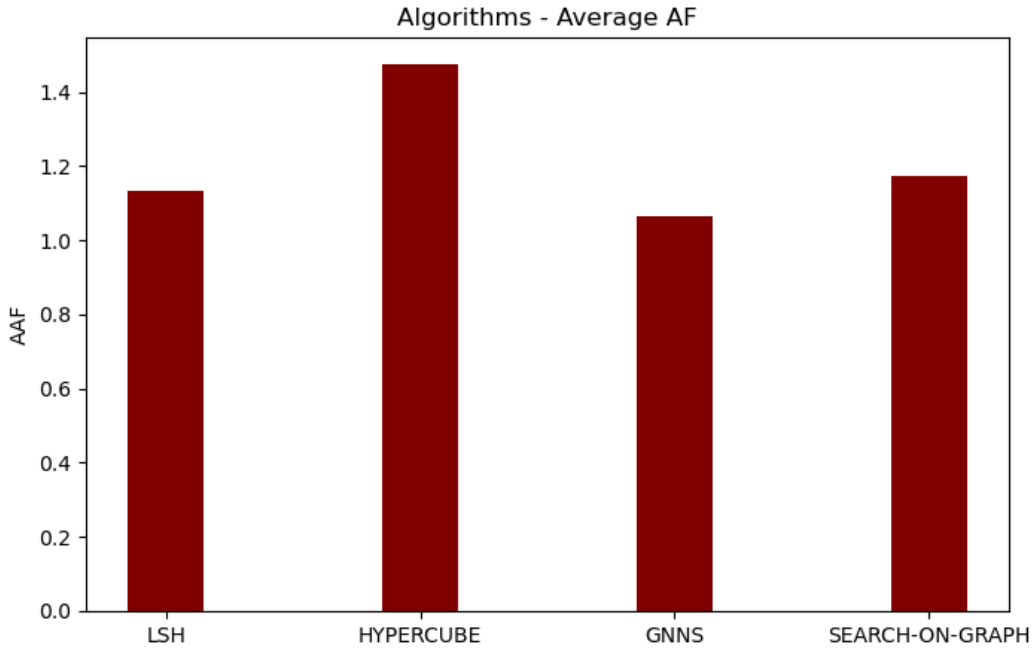
Για τους παραπάνω συνδυασμούς που επιλέχθηκαν, φαίνεται ο LSH να έχει την καλύτερη απόδοση μεταξύ AAF και χρόνου. Συγκεκριμένα, το AAF δεν ξεπερνάει το 1.5 ενώ ο χρόνος του παραμένει σχετικά μικρός.

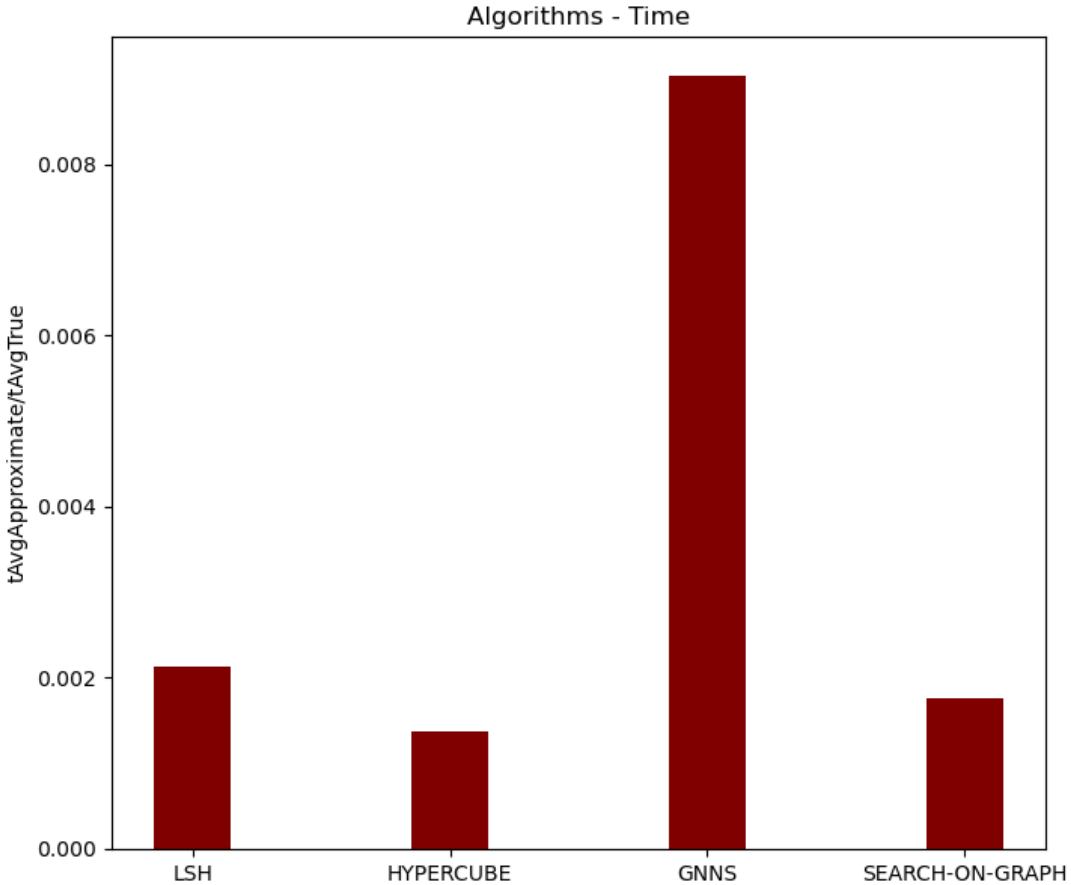
Παρατηρούμε ότι πιο αργός αλγόριθμος είναι ο GNNS ο οποίος όμως έχει αρκετά καλό AAF (μικρότερο ή ίσο του 1.4).

Όλοι οι αλγόριθμοι έχουν παρόμοια σχέση μεταξύ χρόνου και AAF, καθώς όσο αυξάνεται ο χρόνος (με αύξηση των τιμών των παραμέτρων) βελτιώνεται το AAF.

Γενικά, συμπεραίνουμε ότι ως προς την ταχύτητα, οι αλγόριθμοι LSH, SEARCH-ON-GRAPH είναι καλύτεροι, με τον τελευταίο μάλιστα να παρουσιάζει βέλτιστες τιμές στο AAF για μεγαλύτερες τιμές της παραμέτρου l. Ως προς την ακρίβεια των αποτελεσμάτων, καλύτεροι φαίνονται οι αλγόριθμοι LSH, GNNS οι οποίοι έχουν κυρίως AAF μικρότερο του 1.4.

Στα παρακάτω διαγράμματα φαίνονται συγκριτικά τα αποτελέσματα όλων των αλγορίθμων για τις βέλτιστες παραμέτρους τους.



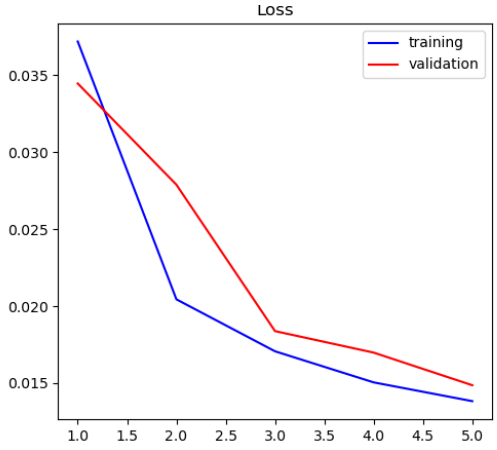


Τα πειράματα που ακολουθούν έχουν γίνει με τη χρήση των αρχείων με μειωμένη διάσταση που δημιουργούνται από το παρακάτω νευρωνικό δίκτυο με αλλαγές στις διάφορες παραμέτρους του.

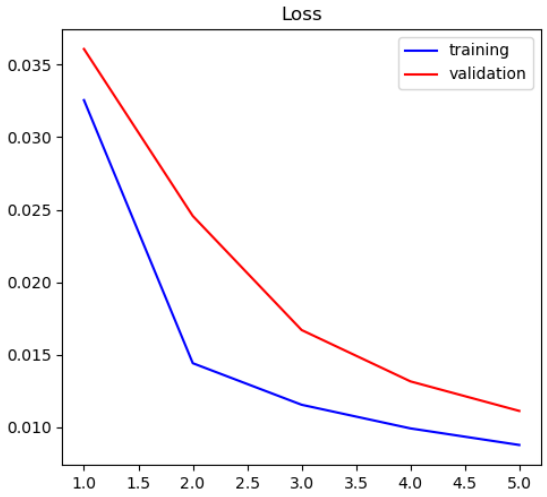
```
#create autoencoder
def encoder(input_img,latent_dim):     #input = 28 x 28 x 1
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img) #28 x 28 x 32
    x = BatchNormalization()(x)
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(2, 2))(x) #14 x 14 x 32
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(x) #14 x 14 x 64
    x = BatchNormalization()(x)
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(2, 2))(x) #7 x 7 x 64
    x = Conv2D(128, (3, 3), activation='relu', padding='same')(x) #7 x 7 x 128
    x = BatchNormalization()(x)
    x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = Conv2D(256, (3, 3), activation='relu', padding='same')(x) #7 x 7 x 256
    x = BatchNormalization()(x)
    x = Conv2D(256, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    flat = Flatten()(x)
    latent_output = Dense(latent_dim, activation='relu')(flat)  # Latent dimension
    return latent_output

def decoder(latent_output):
    x = Dense(6272, activation='relu')(latent_output)
    x = Reshape((7, 7, 128))(x)
    x = Conv2D(128, (3, 3), activation='relu', padding='same')(x) #7 x 7 x 128
    x = BatchNormalization()(x)
    x = Conv2D(128, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(x) #7 x 7 x 64
    x = BatchNormalization()(x)
    x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = UpSampling2D((2,2))(x) #14 x 14 x 64
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(x) # 14 x 14 x 32
    x = BatchNormalization()(x)
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)
    x = BatchNormalization()(x)
    x = UpSampling2D((2,2))(x) # 28 x 28 x 32
    decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x) # 28 x 28 x 1
    return decoded
```

Ενδεικτικά κάποια από τα διαγράμματα για το loss και το accuracy των πειραμάτων με το παραπάνω νευρωνικό.

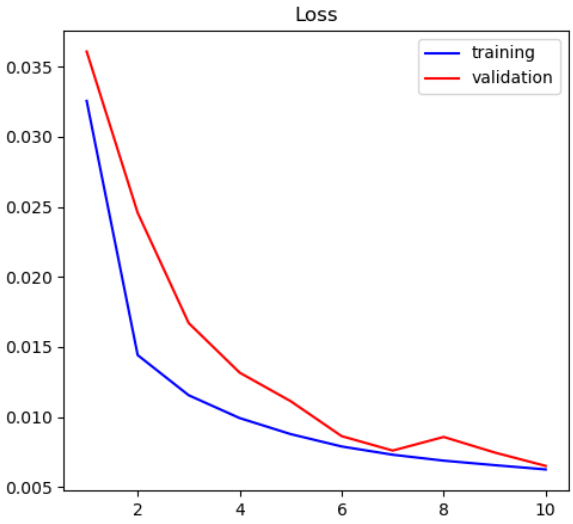
1. latent dimension=10, batch size=64, epochs=5



2. latent dimension=20, batch size=64, epochs=5

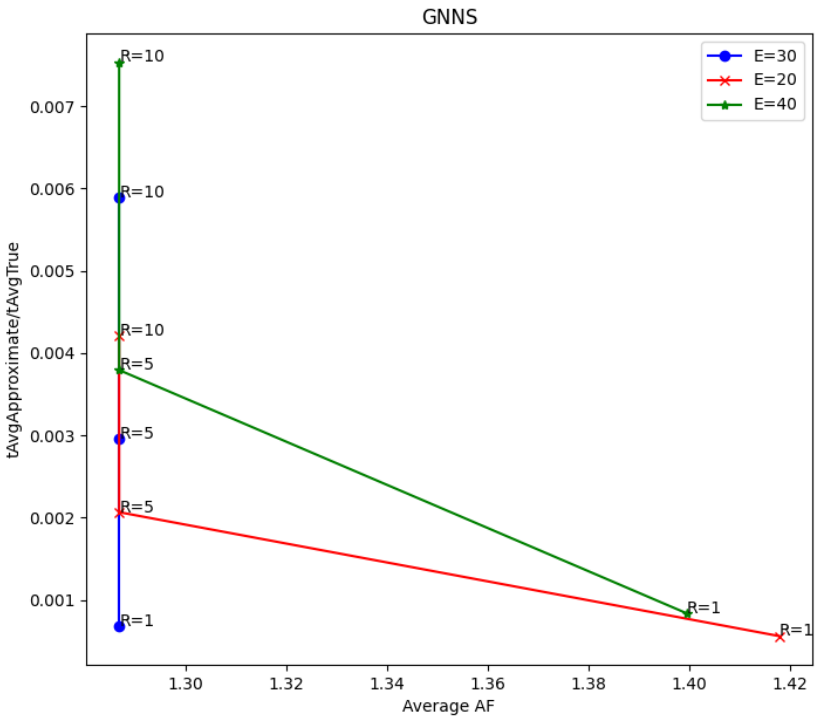


3. Latent dimension=20, batch size=64, epochs=10



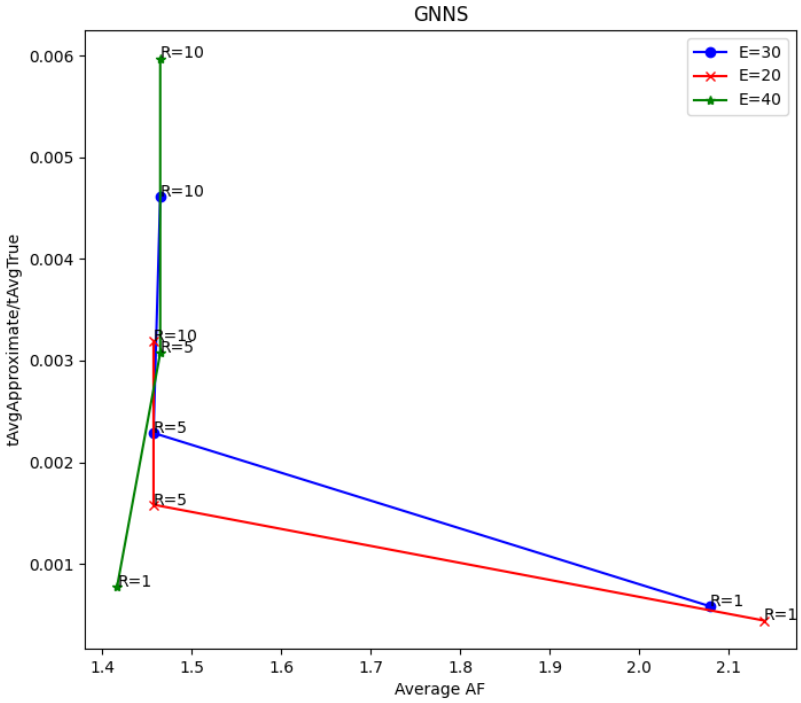
GNNS

- Latent dimension=10, batch size=128, epochs=5



Από το διπλανό διάγραμμα παρατηρούμε ότι γενικά για όλους τους συνδυασμούς, το AAF παραμένει μικρότερο του 1.5 .
Επιπλέον, για E=30, ανεξάρτητα του πλήθους των επαναλήψεων, το AAF παραμένει σταθερό στο 1.2868 . Το ίδιο παραμένει για E=20 και E=40, όμως με R=5 και R=10.
Συμπεραίνουμε ότι οι βέλτιστοι παράμετροι είναι E=30 , R=1 όπου με μία επανάληψη πετυχαίνει το βέλτιστο δυνατό AAF.

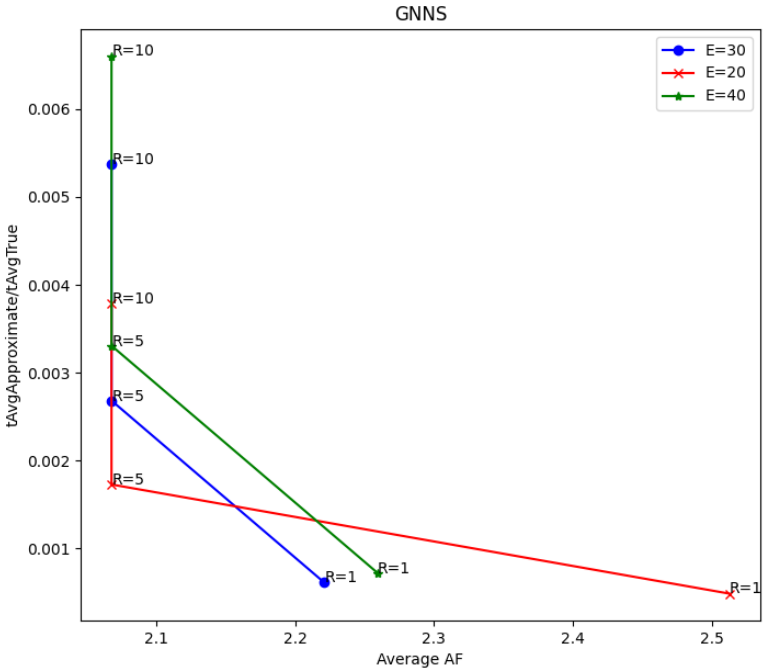
- Latent dimension=20, batch size=128, epochs=5



Από το διπλανό διάγραμμα παρατηρούμε ότι για E=40 το AAF είναι μικρότερο ή ίσο του 1.5 ενώ για E=20 και E=30 ξεκινάει με τιμή μεγαλύτερη του 2.
Βέβαια για E=20, E=30 με την αύξηση των επαναλήψεων R, το AAF βελτιώνεται σημαντικά με μια μικρή αύξηση του χρόνου όμως.
Συμπεραίνουμε ότι οι βέλτιστοι παράμετροι είναι οι E=40, R=1 όπου με μία επανάληψη πετυχαίνει το βέλτιστο δυνατό AAF.

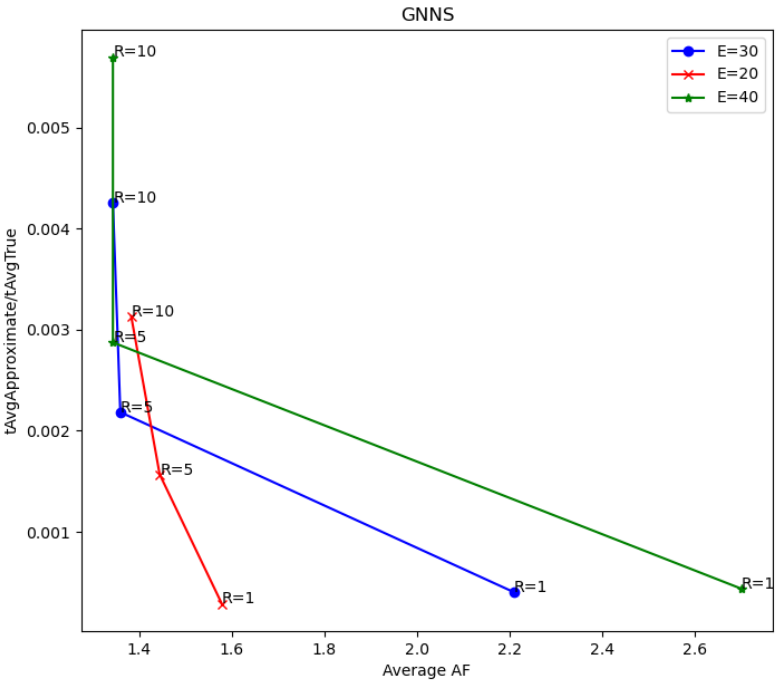
- Latent dimension=10, batch size=64, epochs=5

  Από το διπλανό διάγραμμα παρατηρούμε ότι για όλους τους συνδυασμούς το AAF παραμένει μεγαλύτερο του 2. Για R μεγαλύτερο ή ίσο του 5, για οποιοδήποτε E, το AAF παραμένει σταθερό στο 2.06763 .
  Συμπεραίνουμε ότι οι βέλτιστοι παράμετροι είναι οι E=20, R=5.



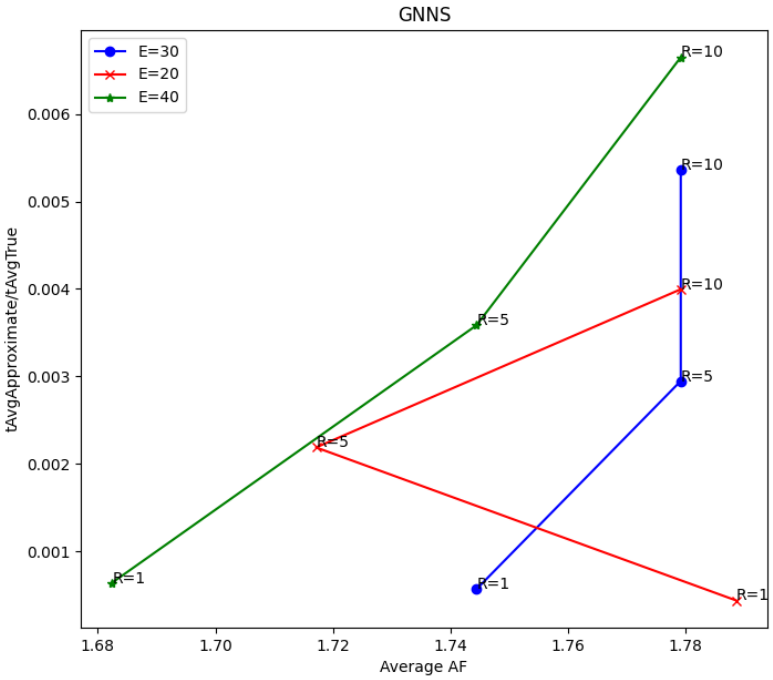
- Latent dimension=20, batch size=64, epochs=5

  Από το διπλανό διάγραμμα ότι για E=20, το AAF ξεκινάει με πολύ μικρότερες τιμές σε σχέση με E=30, E=40, ενώ για R=10 ο χρόνος που κάνει είναι αρκετά μικρότερος συγκριτικά με το αντίστοιχο R στις άλλες καμπύλες.
  Για E=30, E=40 ενώ αυξάνεται ο χρόνος σημαντικά, δεν υπάρχει μεγάλη βελτίωση στο AAF.
  Συμπεραίνουμε ότι οι βέλτιστοι παράμετροι είναι οι E=20, R=5.



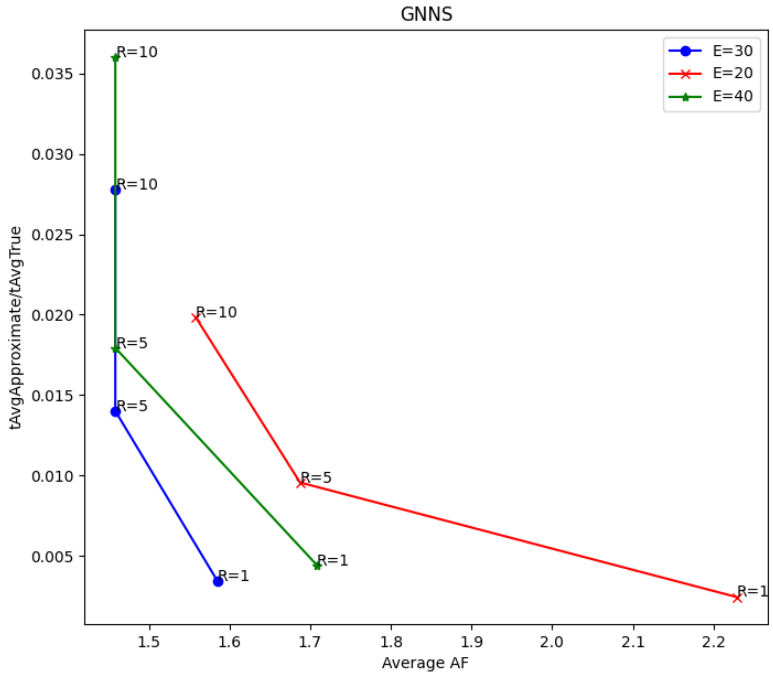
- Latent dimension=10, batch size=128, epochs=10

  Από το διπλανό διάγραμμα παρατηρούμε ότι όσο αυξάνεται το πλήθος των επαναλήψεων R, αυξάνεται και η τιμή AAF. Αυτό οφείλεται στην αναγωγή στον αρχικό χώρο, καθώς ο προσεγγιστικά πλησιέστερος γείτονας μπορεί να απέχει αρκετά από τον πραγματικά πλησιέστερο.
  Οι βέλτιστοι παράμετροι είναι οι E=40, R=1.



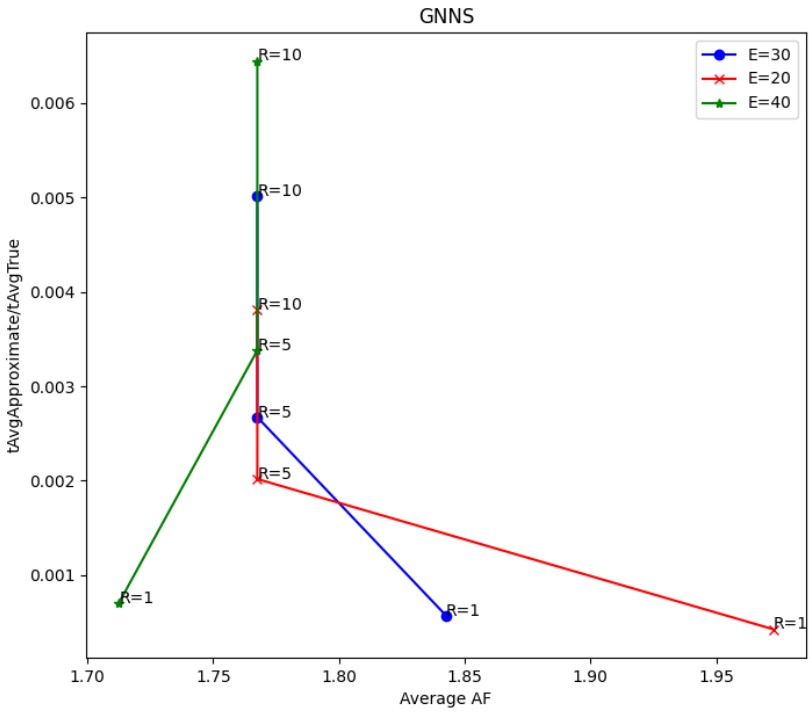
- Latent dimension=20, batch size=128, epochs=10

  Από το διπλανό διάγραμμα παρατηρούμε ότι για E=30, E=40 το AAF ξεκινάει με μικρότερη τιμή από ότι για E=20. Όμως για E=20, αυξάνοντας το R, βελτιώνεται αρκετά το AAF σε λίγο χρόνο συγκριτικά με τις άλλες καμπύλες όπου το AAF παραμένει σταθερό για R=5, R=10.
  Οι βέλτιστοι παράμετροι είναι E=30, R=5.



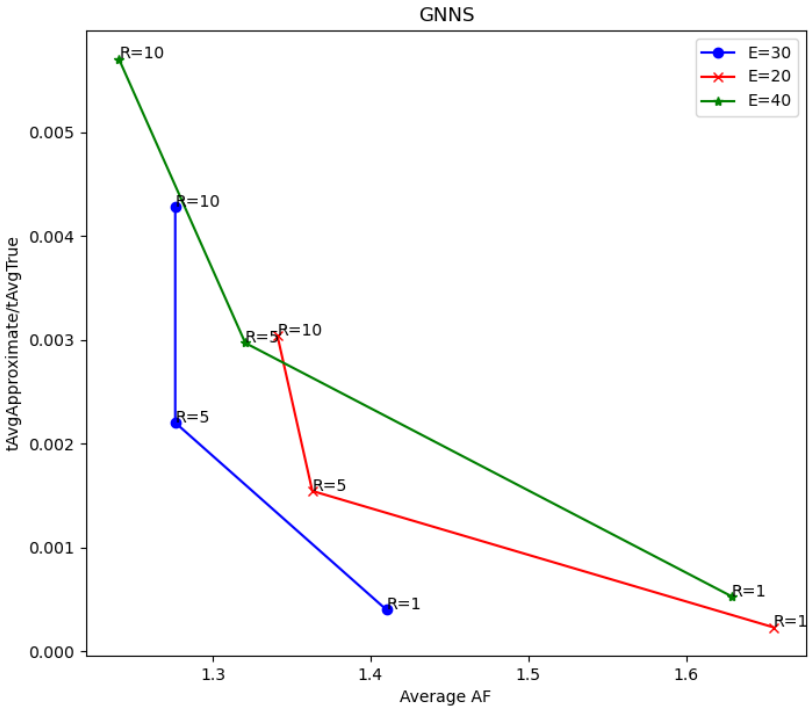
- Latent dimension=10, batch size=64, epochs=10

  Από το διπλανό διάγραμμα παρατηρούμε ότι για E=20, E=30 (σε αντίθεση με E=40) καθώς αυξάνονται οι τιμές των παραμέτρων, οι τιμές των AAF βελτιώνονται. Συγκεκριμένα για E=20, το AAF έχει τη μεγαλύτερη βελτίωση με την αύξηση των επαναλήψεων.
  Οι βέλτιστοι παράμετροι είναι E=40, R=1.



- Latent dimension=20, batch size=64, epochs=10

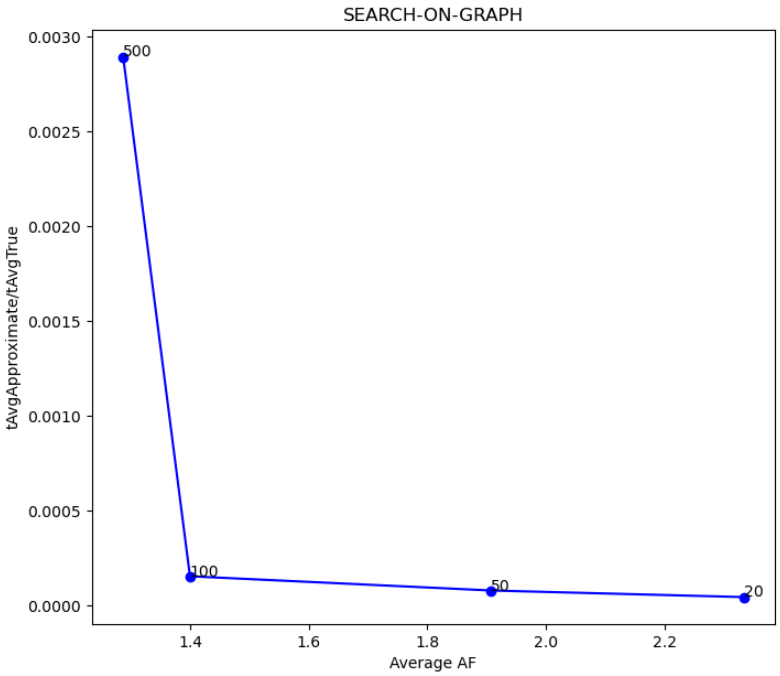
  Από το διπλανό διάγραμμα παρατηρούμε ότι για όλους τους συνδυασμούς το AAF είναι μικρότερο του 1.7, ενώ για E=30, το AAF ξεκινάει κοντά στο 1.4 .
  Για E=40, R=10 το AAF πλησιάζει το 1 που είναι το βέλτιστο, όμως έχει σημαντική αύξηση του χρόνου.
  Οι βέλτιστοι παράμετροι είναι E=30, R=5.



SEARCH-ON-GRAPH (με ευρετήριο MRNG)

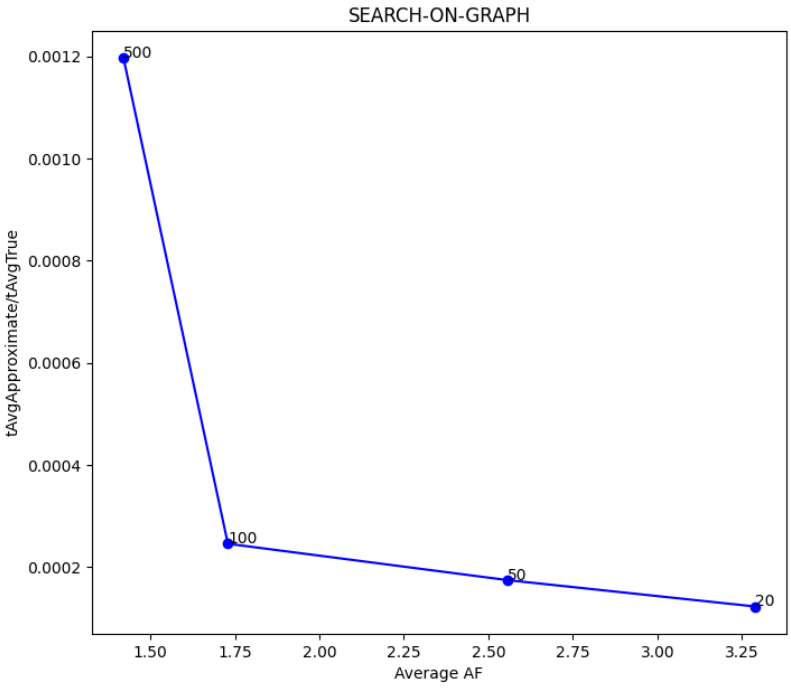
- Latent dimension=10, batch size=128, epochs=5

  Από το διπλανό διάγραμμα παρατηρούμε ότι για l=20 έως l=100, αμελητέα αύξηση του χρόνου, όμως σημαντική βελτίωση του AAF. Αντίθετα, από l=100 μέχρι l=500, η αύξηση του χρόνου είναι συγκριτικά πολύ μεγαλύτερη από τη βελτίωση του AAF.
  Η βέλτιστη παράμετρος είναι l=100 καθώς σε ελάχιστο χρόνο έχει AAF= 1.39955 .



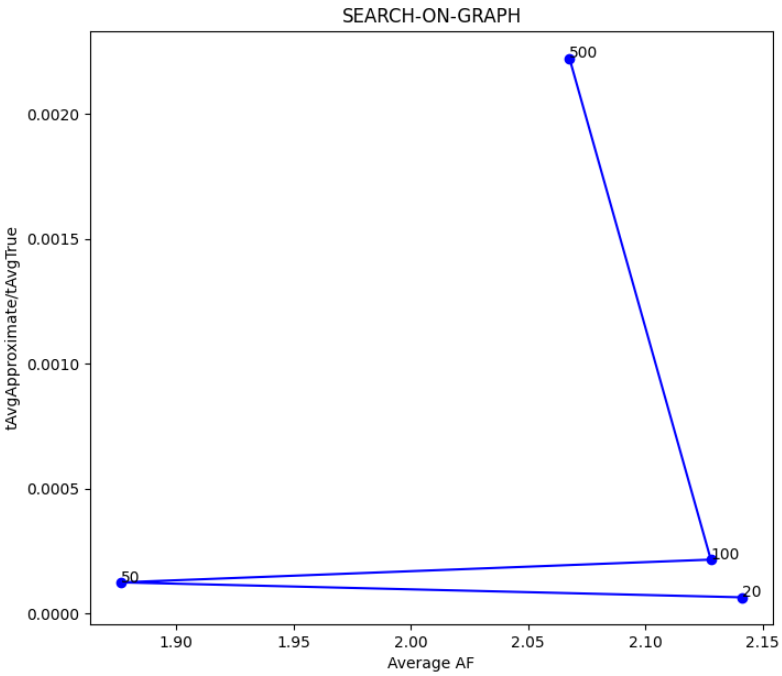
- Latent dimension=20, batch size =128, epochs=5

  Από το διπλανό διάγραμμα παρατηρούμε ότι για l=20, l=50 το AAF είναι αρκετά μεγάλο (>2.5). Όμως για l=100 το AAF βελτιώνεται σημαντικά με μικρή αύξηση του χρόνου. Για l=500, το AAF παρουσιάζει μικρή βελτίωση με μεγάλη αύξηση του χρόνου.
  Από τα παραπάνω, η βέλτιστη παράμετρος φαίνεται να είναι η l=100.



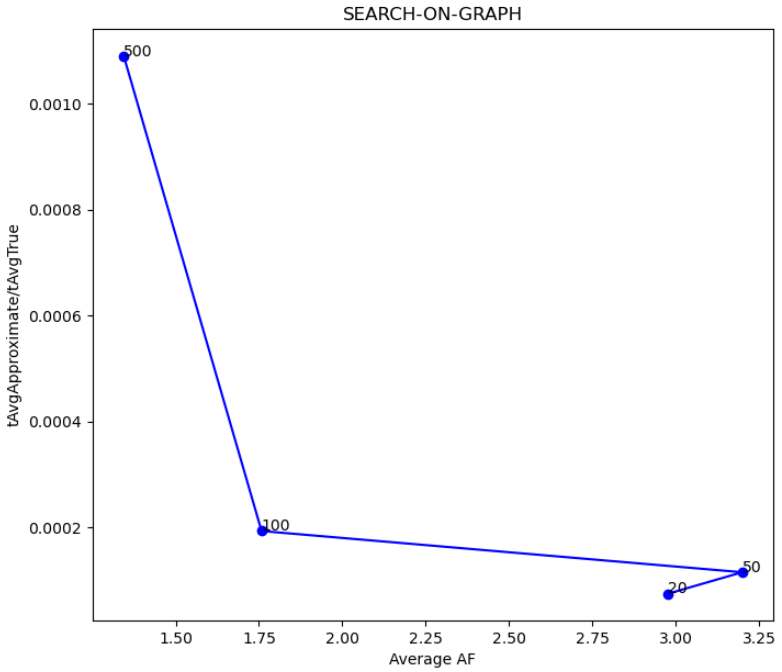
- Latent dimension=10, batch size=64, epochs=5

  Από το διπλανό διάγραμμα παρατηρούμε ότι για l=100, το AAF χειροτερεύει συγκριτικά με l=50. Αυτό οφείλεται στην αναγωγή στον αρχικό χώρο, καθώς ο προσεγγιστικά πλησιέστερος γείτονας μπορεί να απέχει αρκετά από τον πραγματικά πλησιέστερο.
  Η βέλτιστη παράμετρος είναι η l=50 καθώς έχει το βέλτιστο δυνατό AAF σε ελάχιστο χρόνο.



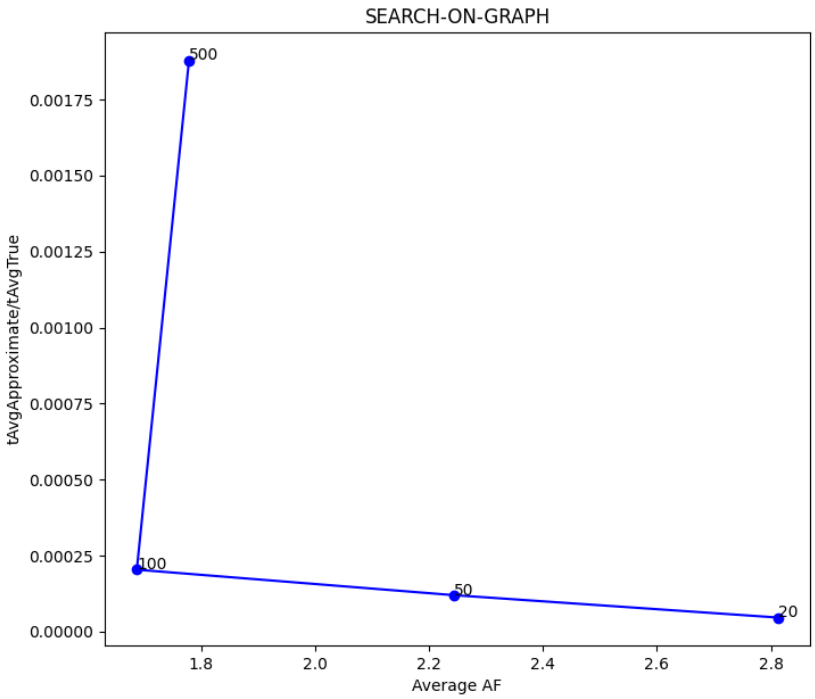
- Latent dimension=20, batch size=64, epochs=5

  Από το διπλανό διάγραμμα παρατηρούμε ότι για l=20, l=50 , το AAF έχει μεγάλες τιμές (> 3). Από l=50 έως l=100, το AAF σχεδόν υποδιπλασιάζεται με ελάχιστη αύξηση του χρόνου και φτάνει το 1.75 .
  Η βέλτιστη παράμετρος είναι η l=100.



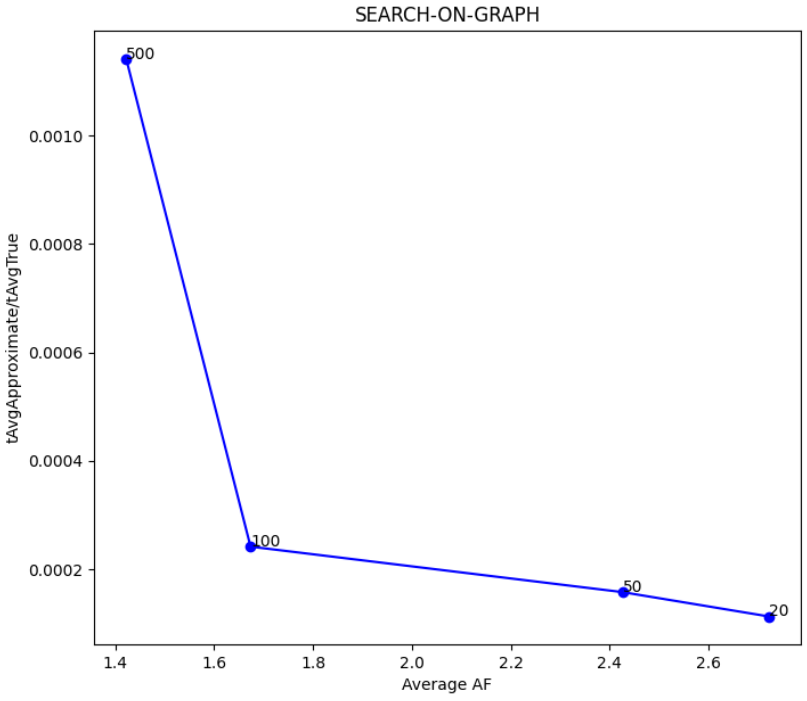
- Latent dimension=10, batch size=128, epochs=10

  Από το διπλανό διάγραμμα παρατηρούμε ότι από l=20 έως l=100, το AAF εμφανίζει σημαντική βελτίωση με αμελητέα αύξηση του χρόνου.
  Η βέλτιστη παράμετρος είναι η l=100.



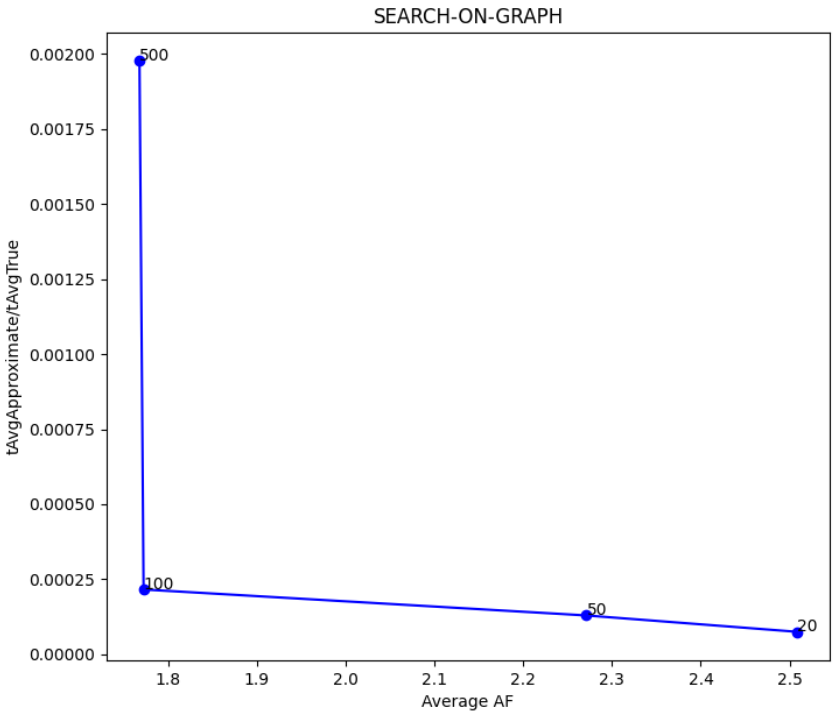
- Latent dimension=20, batch size=128, epochs=10

  Από το διπλανό διάγραμμα παρατηρούμε ότι όσο αυξάνεται η παράμετρος l, βελτιώνεται το AAF, με μικρή σχετικά αύξηση του χρόνου (με εξαίρεση το διάστημα από l=100 έως l=500).
  Η βέλτιστη παράμετρος είναι η l=100.



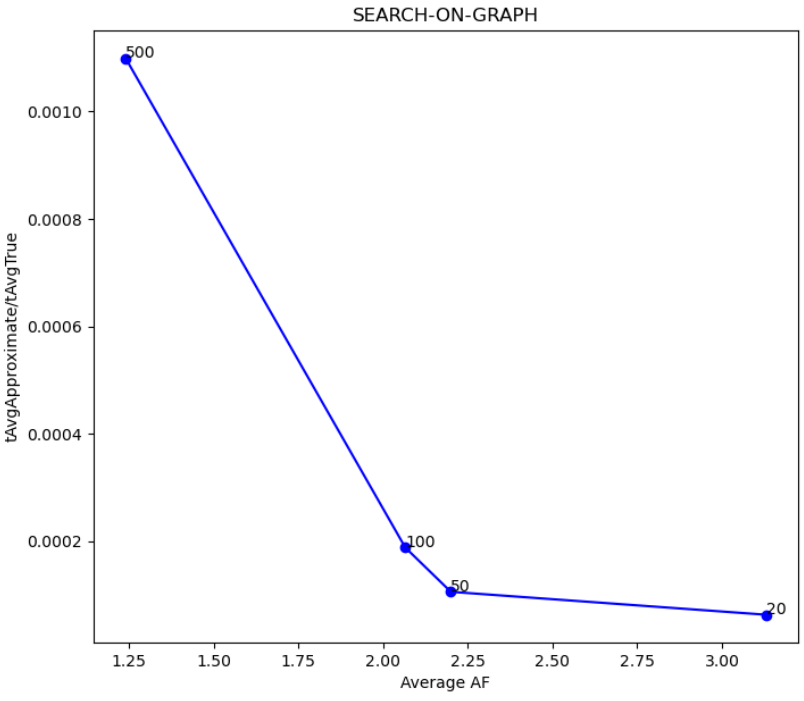
- Latent dimension=10,
  batch size=64, epochs=10

  Από το διπλανό διάγραμμα παρατηρούμε ότι από l=20 έως l=100, το AAF βελτιώνεται αρκετά με ελάχιστη αύξηση του χρόνου, ενώ από l=100 έως l=500, το AAF παραμένει σταθερό με το χρόνο να συνεχίζει να αυξάνεται.
  Η βέλτιστη παράμετρος είναι η l=100.



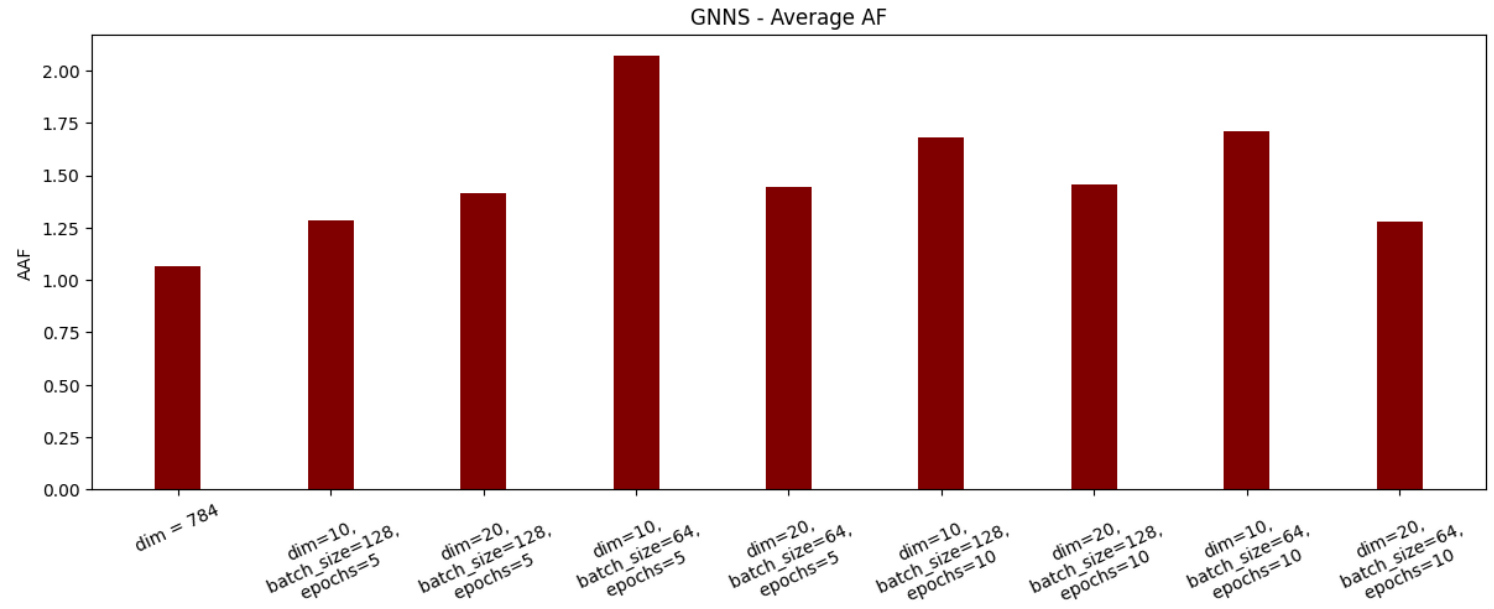
- Latent dimension=20,
  batch size=64, epochs=10

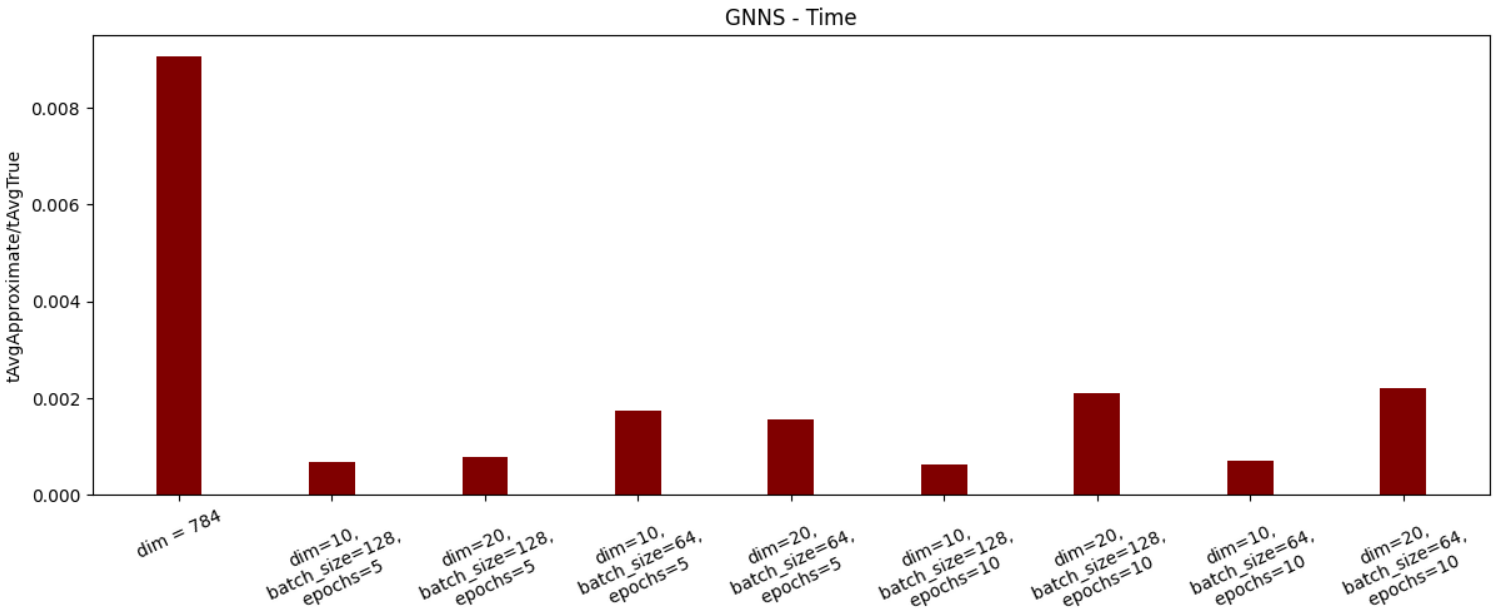
  Από το διπλανό διάγραμμα παρατηρούμε ότι από l=20 έως l=50 βελτιώνεται το AAF με μικρή αύξηση του χρόνου, από l=50 έως l=100 η βελτίωση του AAF δεν είναι ανάλογη της αύξησης του χρόνου. Από l=100 έως l=500 βελτιώνεται το AAF σημαντικά αφού φτάνει το 1.25 αλλά ο χρόνος αυξάνεται σε μεγάλο βαθμό.
  Η βέλτιστη παράμετρος όσον αφορά το χρόνο είναι η l=100, ενώ όσον αφορά το AAF είναι η l=500.



ΣΥΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

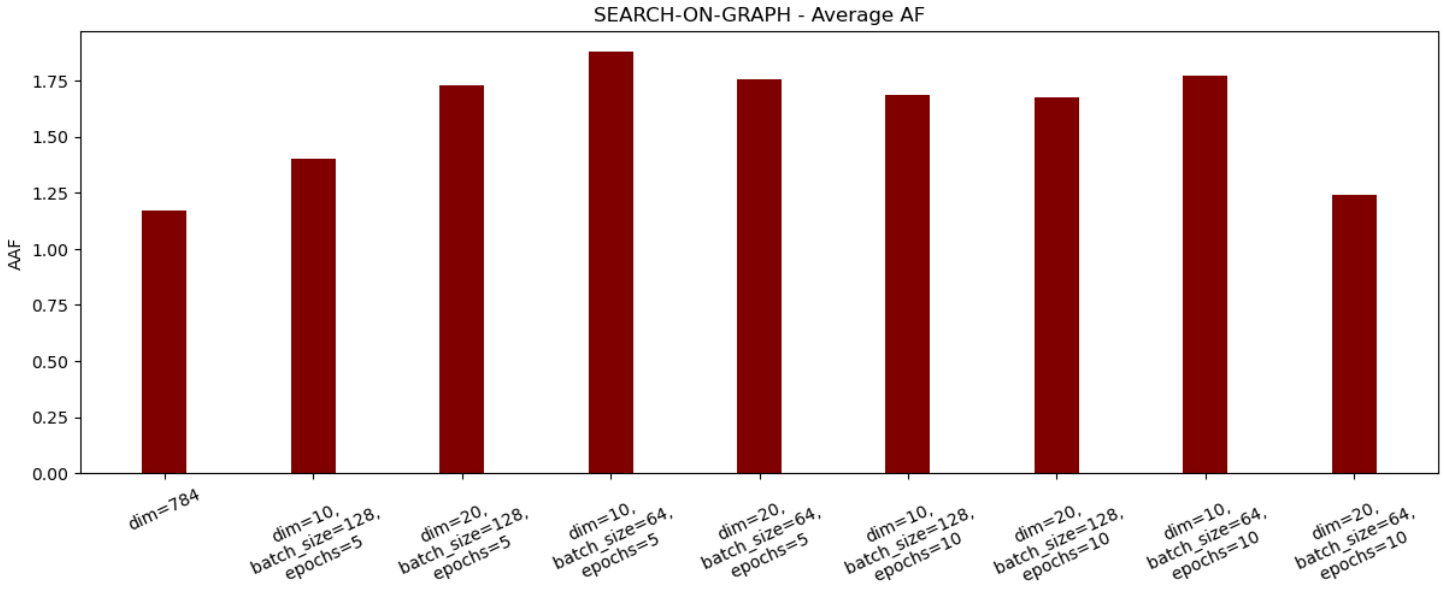
Τα διαγράμματα δημιουργήθηκαν με τις βέλτιστες τιμές παραμέτρων που επιλέχθηκαν παραπάνω.

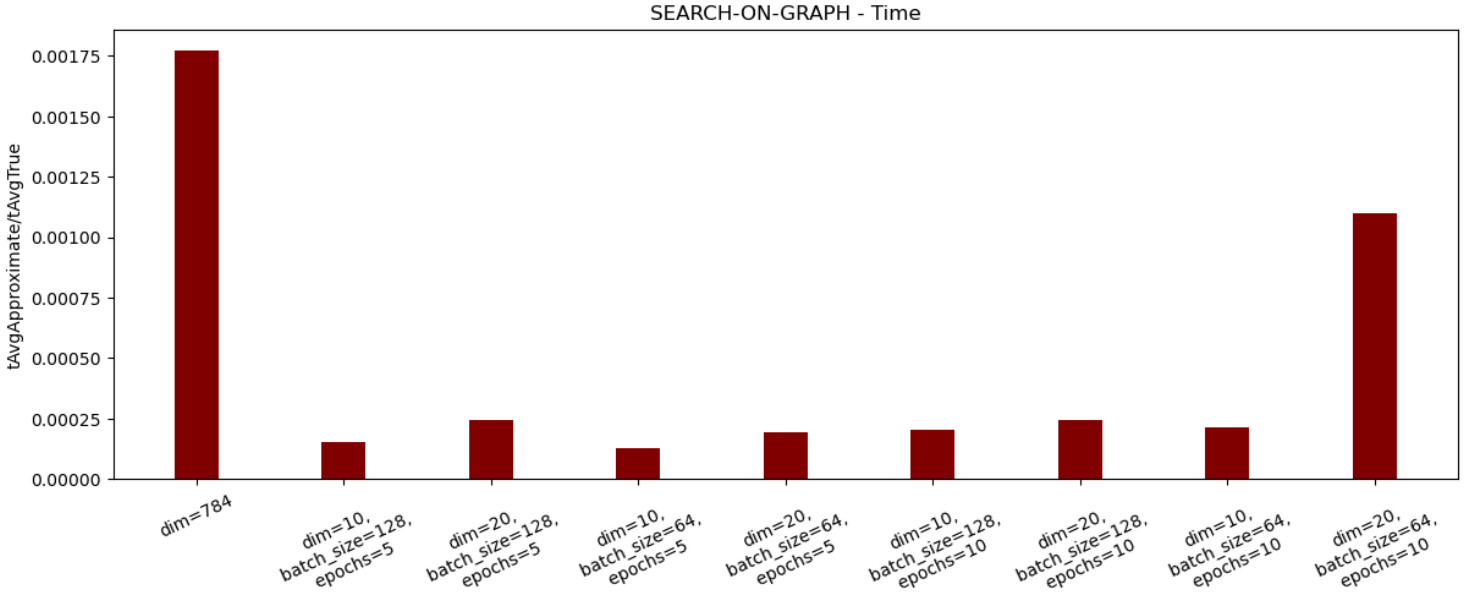




Παρατηρούμε ότι όσον αφορά το AAF, η είσοδος με την αρχική διάσταση (784) έχει τιμή που πλησιάζει περισσότερο το 1, ενώ τα υπόλοιπα κυμαίνονται από 1.25 έως 1.75, με εξαίρεση ένα που φτάνει στο 2.

Όσον αφορά το χρόνο, φαίνεται ότι με μειωμένη διάσταση ως είσοδο, ο χρόνος μειώνεται σημαντικά σε σύγκριση με την αρχική διάσταση.





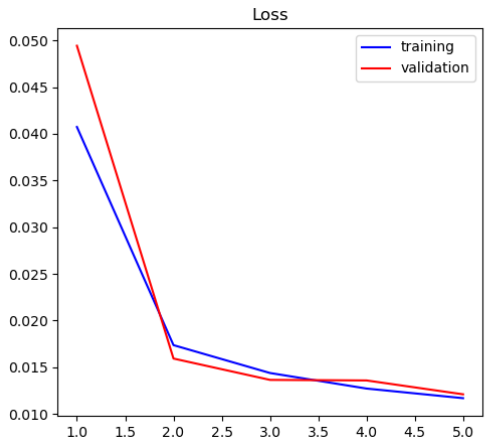
Παρατηρούμε ότι όσον αφορά το AAF, κανένας συνδυασμός δεν ξεπερνάει το 2, ενώ η είσοδος με την αρχική διάσταση (784) και αυτή που δημιουργήθηκε από το νευρωνικό με παραμέτρους 20-64-10, έχουν AAF που πλησιάζει το 1.

Όσον αφορά το χρόνο, φαίνεται ότι με μειωμένη διάσταση ως είσοδο, ο χρόνος μειώνεται σημαντικά σε σύγκριση με την αρχική διάσταση, με εξαίρεση το συνδυασμό 20-64-10, που αν και ο χρόνος του είναι χαμηλότερος από αυτόν της αρχικής διάστασης, παραμένει σχετικά υψηλός.

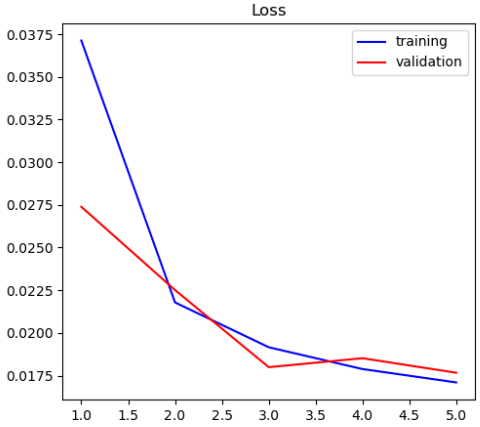
Μετά από πειράματα στη δομή του προηγούμενου νευρωνικού, καταλήξαμε στο παρακάτω. Ο κώδικας βρίσκεται στο αρχείο reduce.py .

```python
#create autoencoder
def encoder(input_img,latent_dim):     #input = 28 x 28 x 1 (wide and thin)
    x = Conv2D(32, (3, 3), activation='relu', padding='same')(input_img)    # 28 x 28 x 32
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)                                   # 14 x 14 x 32

    x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)            # 14 x 14 x 64
    x = BatchNormalization()(x)
    x = MaxPooling2D(pool_size=(2, 2))(x)                                   # 7 x 7 x 64

    flat = Flatten()(x)
    latent_output = Dense(latent_dim, activation='relu')(flat)  # Latent dimension
    return latent_output

def decoder(latent_output):
    x = Dense(3136, activation='relu')(latent_output)
    x = Reshape((7, 7, 64))(x)

    x = Conv2D(64, (3, 3), activation='relu', padding='same')(x)            # 7 x 7 x 64
    x = BatchNormalization()(x)
    x = UpSampling2D((2,2))(x)                                              # 14 x 14 x 64

    x = Conv2D(32, (3, 3), activation='relu', padding='same')(x)            # 14 x 14 x 32
    x = BatchNormalization()(x)
    x = UpSampling2D((2,2))(x)                                              # 28 x 28 x 32

    decoded = Conv2D(1, (3, 3), activation='sigmoid', padding='same')(x)    # 28 x 28 x 1
    return decoded
```

Ενδεικτικά κάποια από τα διαγράμματα για το loss και το accuracy των πειραμάτων.

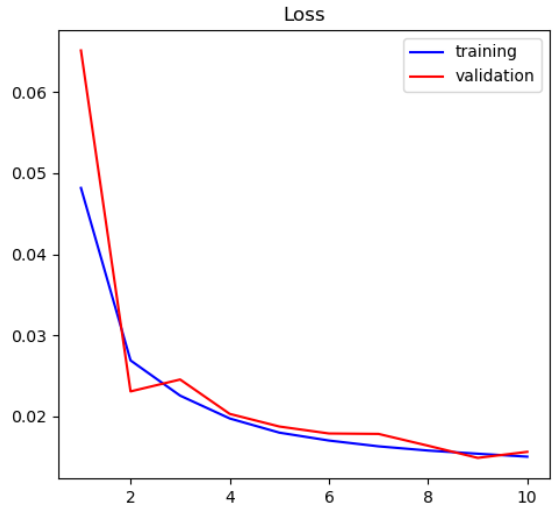


1. Latent dimension=20, batch size=128, epochs=5

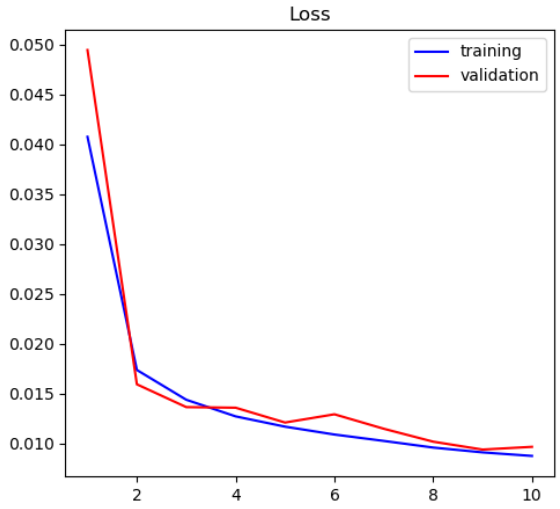


2. Latent dimension=10, batch size=64, epochs=5

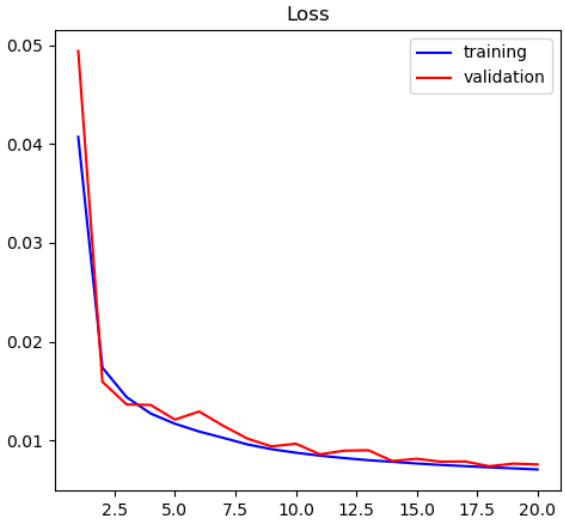
3. Latent dimension= 10, batch size=128, epochs=10



4. Latent dimension=20, batch size=128, epochs=10



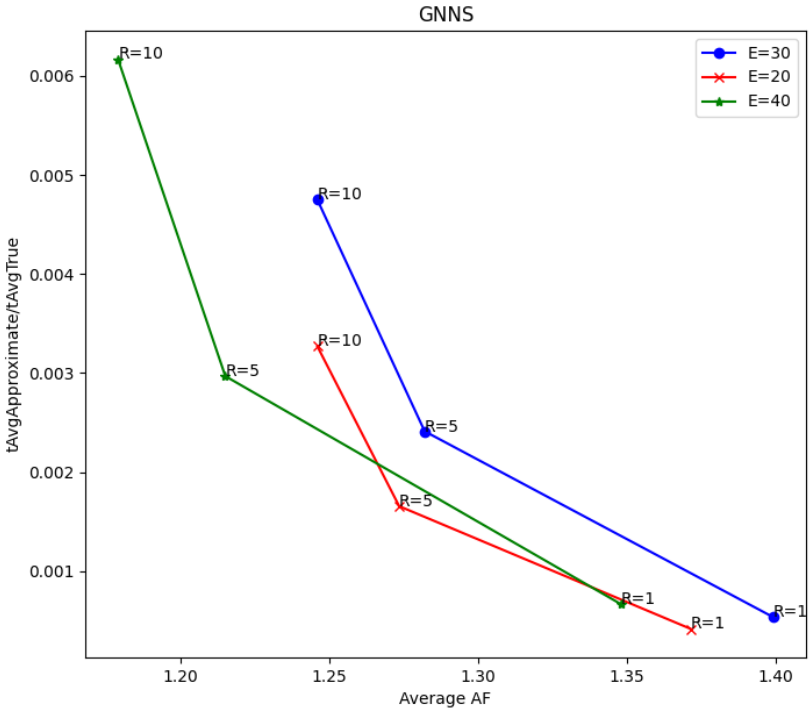
5. Latent dimension=20, batch size=128, epochs=20



Τα πειράματα που ακολουθούν έχουν γίνει με αυτό το νευρωνικό με αλλαγές στις διάφορες παραμέτρους του.

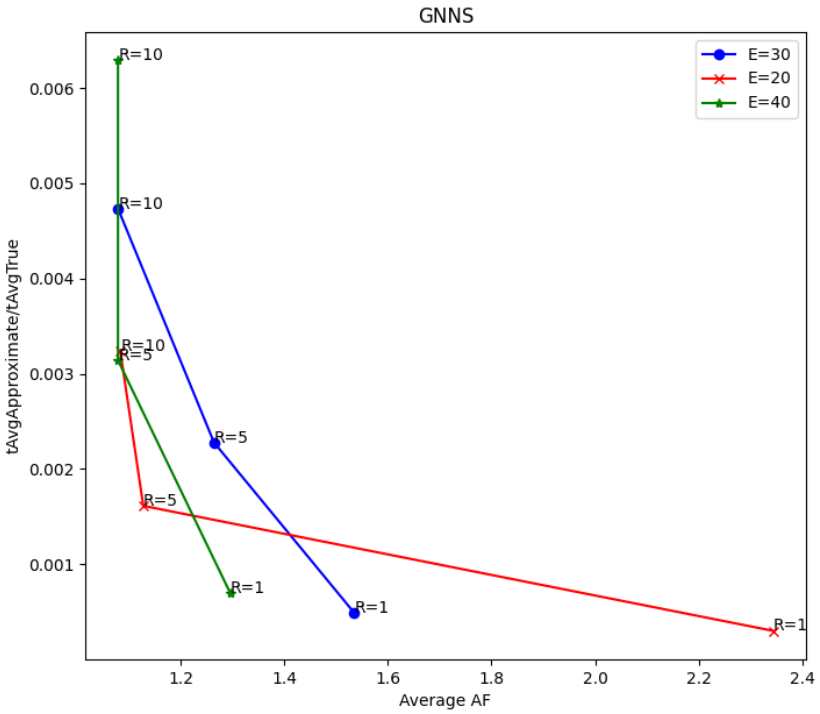
GNNS

- Latent dimension=10, batch size=128, epochs=5



Από το διπλανό διάγραμμα παρατηρούμε ότι σε όλους τους συνδυασμούς, το AAF ξεκινάει με χαμηλή τιμή (μικρότερη του 1.4) ενώ όσο αυξάνονται οι επαναλήψεις, άρα και ο χρόνος, βελτιώνεται και το AAF.
Οι βέλτιστοι παράμετροι είναι οι E=20, R=5.

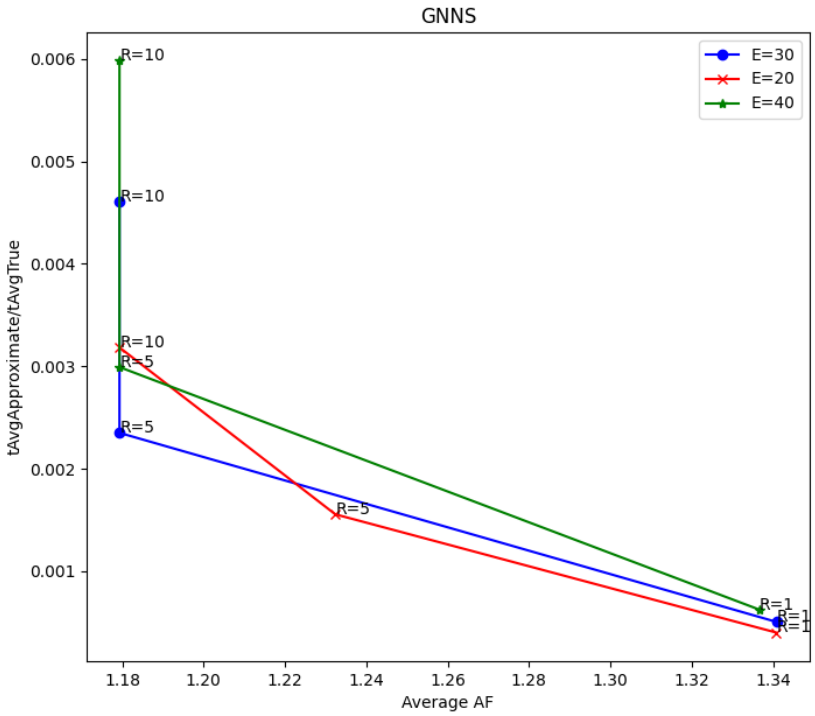
- Latent dimension=20, batch size=128, epochs=5



Από το διπλανό διάγραμμα παρατηρούμε ότι η μόνη καμπύλη που ξεκινάει με AAF μεγαλύτερο του 2, είναι η E=20 η οποία όμως κάνει ελάχιστο χρόνο.
Οι άλλες 2 καμπύλες σε παρόμοιο χρόνο, σε μια επανάληψη ξεκινάνε με καλύτερο AAF (μικρότερο του 1.6).
Οι βέλτιστοι παράμετροι είναι οι E=20, R=5. Μια ακόμα καλή επιλογή είναι οι E=40, R=1 η οποία πολύ μικρότερο χρόνο αλλά μεγαλύτερη τιμή AAF.

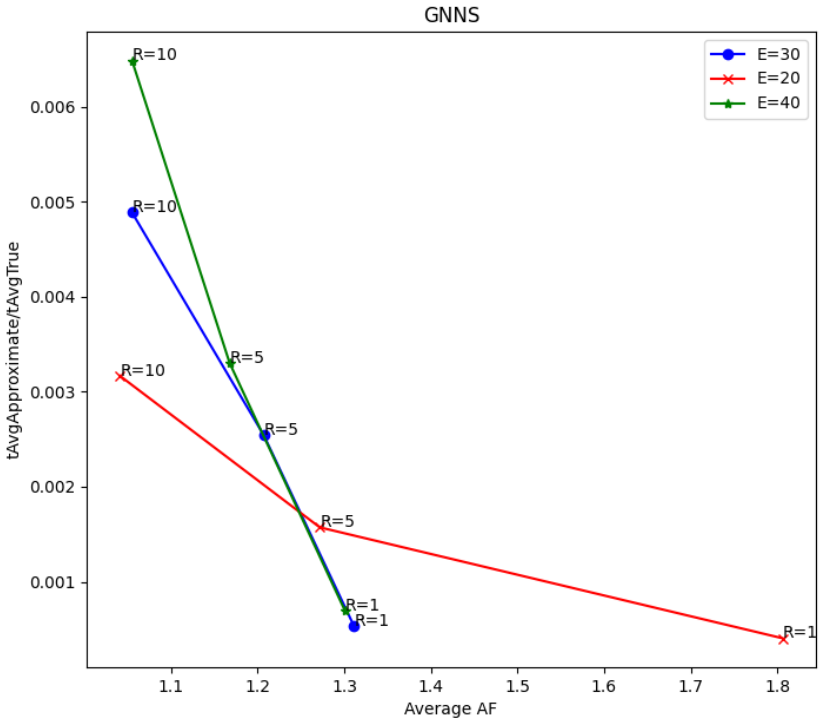
- Latent dimension=10, batch size=64, epochs=5

  Από το διπλανό διάγραμμα παρατηρούμε ότι γενικά όλες οι καμπύλες έχουν παρόμοια αποτελέσματα στο AAF (με εξαίρεση το E=20, R=5).
  Οι βέλτιστοι παράμετροι είναι οι E=30, R=5.



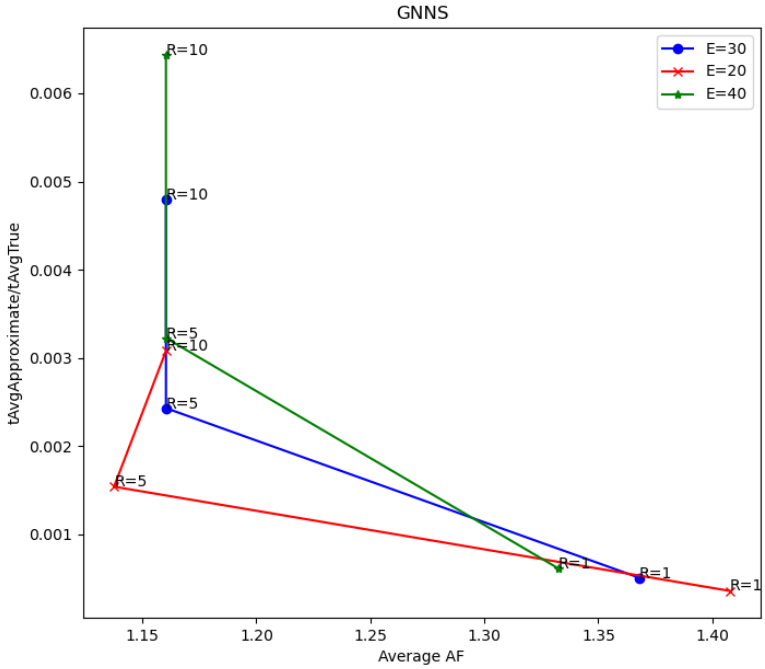
- Latent dimension=20, batch size=64, epochs=5

  Από το διπλανό διάγραμμα παρατηρούμε ότι οι καμπύλες E=30, E=40 έχουν σχετικά παρόμοια αποτελέσματα. Για E=20, R=1 αν και έχει παρόμοιο χρόνο με τις άλλες 2 καμπύλες, ξεκινάει με μεγαλύτερο AAF.
  Οι βέλτιστοι παράμετροι E=30, R=1 και E=40, R=1.



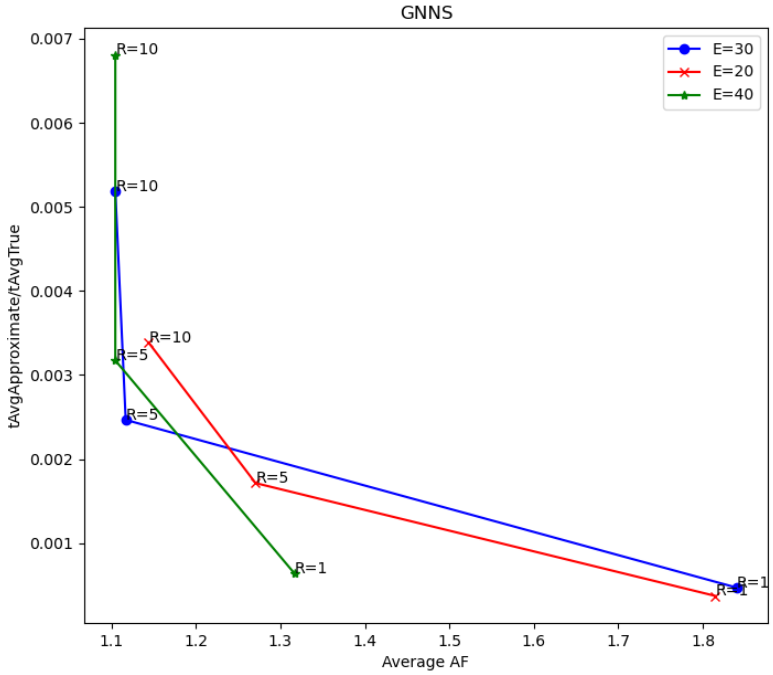
- Latent dimension=10, batch size=128, epochs=10

  Από το διπλανό διάγραμμα παρατηρούμε ότι όλοι οι συνδυασμοί ξεκινάνε με AAF μικρότερο του 1.5, ενώ μετά από ένα σημείο τείνει να σταθεροποιηθεί σε όλες τις καμπύλες.
  Οι βέλτιστοι παράμετροι είναι οι E=20, R=5.



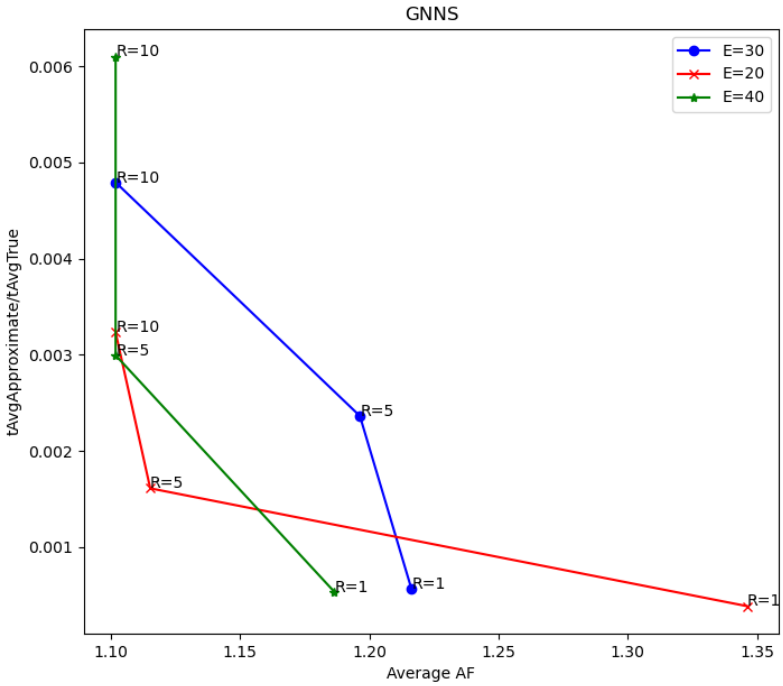
- Latent dimension=20, batch size=128, epochs=10

  Από το διπλανό διάγραμμα παρατηρούμε ότι η καμπύλη που ξεκινάει με το καλύτερο AAF είναι η E=40 κάνοντας αντίστοιχο χρόνο με τις άλλες 2.
  Τη μεγαλύτερη βελτίωση στο AAF έχει η E=30 μεταξύ R=1 και R=5 με ανάλογη αύξηση του χρόνου όμως.
  Οι βέλτιστοι παράμετροι είναι οι E=40, R=1.



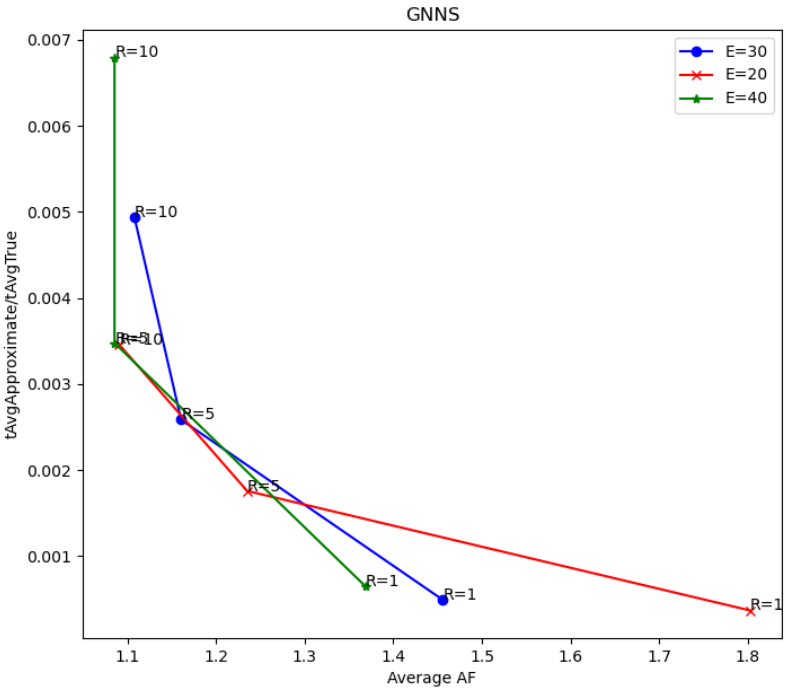
- Latent dimension=10, batch size=64, epochs=10

  Από το διπλανό διάγραμμα παρατηρούμε ότι για E=30, μεταξύ R=1 και R=5 υπάρχει μεγάλη αύξηση του χρόνου με ελάχιστη βελτίωση του AAF. Αυτό δεν παρατηρείται στις άλλες καμπύλες όπου υπάρχει ανάλογη αύξηση του χρόνου και βελτίωση στο AAF.
  Οι βέλτιστοι παράμετροι είναι οι E=20, R=5.



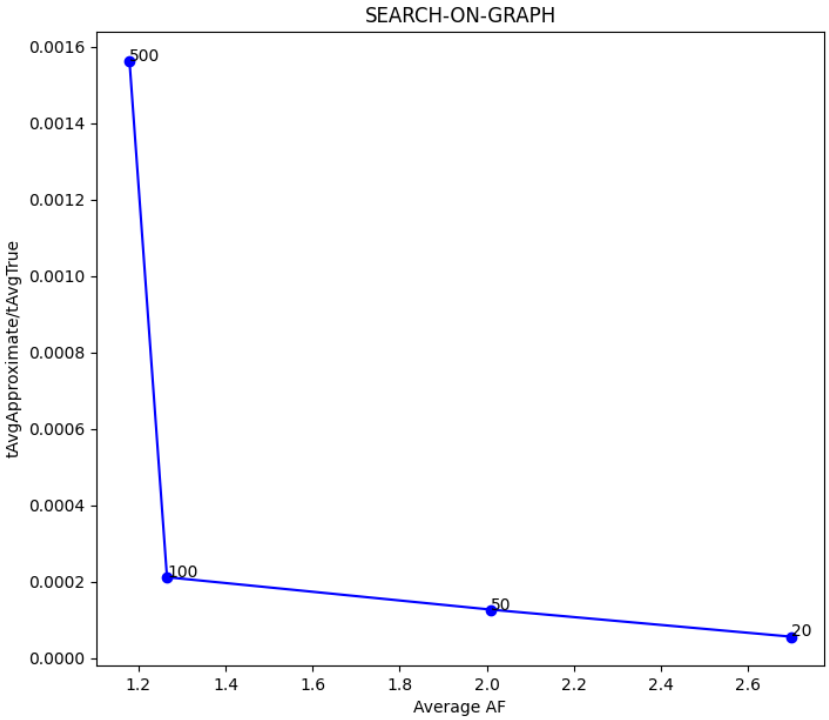
- Latent dimension=20, batch size=64, epochs=10

  Από το διπλανό διάγραμμα παρατηρούμε ότι η καμπύλη E=20 ξεκινάει με μεγαλύτερο AAF σε σχέση με τις υπόλοιπες. Το χειρότερο χρόνο κάνει ο συνδυασμός E=40, R=10 χωρίς να προσφέρει καλύτερο AAF, αφού ίδια τιμή εμφανίζουν οι E=40, R=5 και E=20, R=10 σε μικρότερο χρόνο.
  Οι βέλτιστοι παράμετροι είναι οι E=20, R=5.



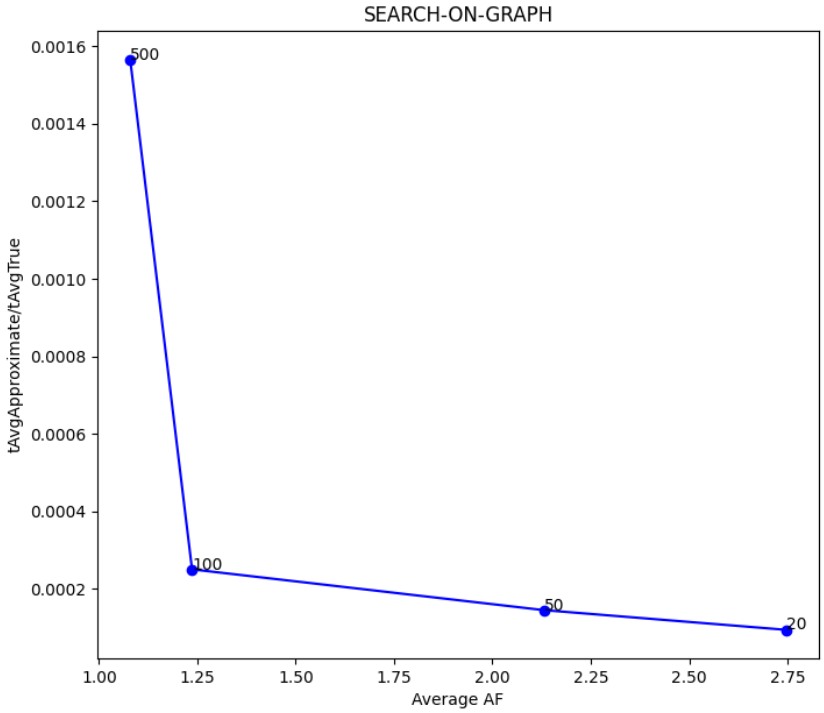
SEARCH-ON-GRAPH (με ευρετήριο MRNG)

- Latent dimension=10, batch size=128, epochs=5



Από το διπλανό διάγραμμα παρατηρούμε ότι αν και για l=20 ξεκινάει με AAF μεγαλύτερο του 2.6, υπάρχει μεγάλη βελτίωση του AAF μέχρι l=100 με ελάχιστη αύξηση του χρόνου.
Από l=100 έως l=500, η αύξηση του χρόνου δεν είναι ανάλογη με τη βελτίωση του AAF.
Η βέλτιστη παράμετρος είναι η l=100.

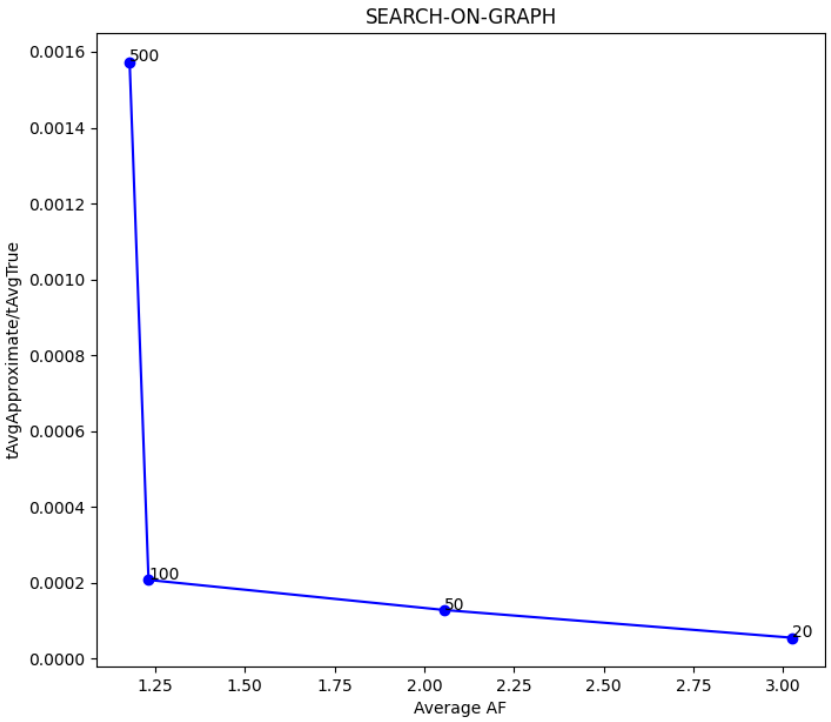
- Latent dimension=20, batch size=128, epochs=5



Από το διπλανό διάγραμμα παρατηρούμε ότι αν και για l=20 ξεκινάει με AAF ίσο με 2.75, υπάρχει μεγάλη βελτίωση του AAF μέχρι l=100 με ελάχιστη αύξηση του χρόνου.
Από l=100 έως l=500, η αύξηση του χρόνου δεν είναι ανάλογη με τη βελτίωση του AAF.
Η βέλτιστη παράμετρος είναι η l=100.

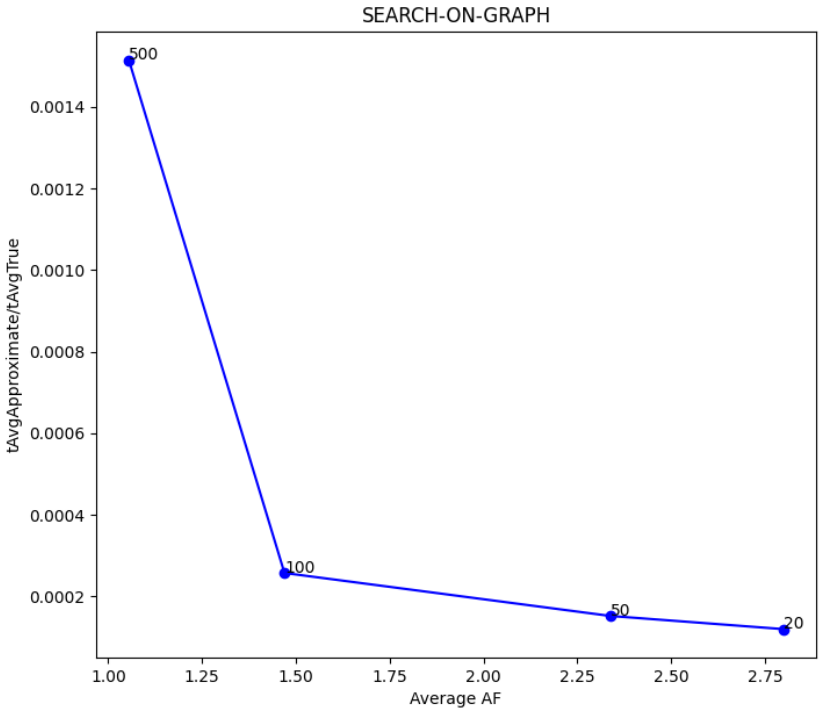
- Latent dimension=10, batch size=64, epochs=5

  Από το διπλανό διάγραμμα παρατηρούμε ότι το AAF ξεκινάει για l=20 με τιμή ίση με 3, που είναι πολύ μακριά από τη βέλτιστη (δηλαδή το 1). Βέβαια για l=100 πλησιάζει αρκετά καθώς έχει τιμή μικρότερη του 1.25 .
  Η βέλτιστη παράμετρος είναι η l=100.



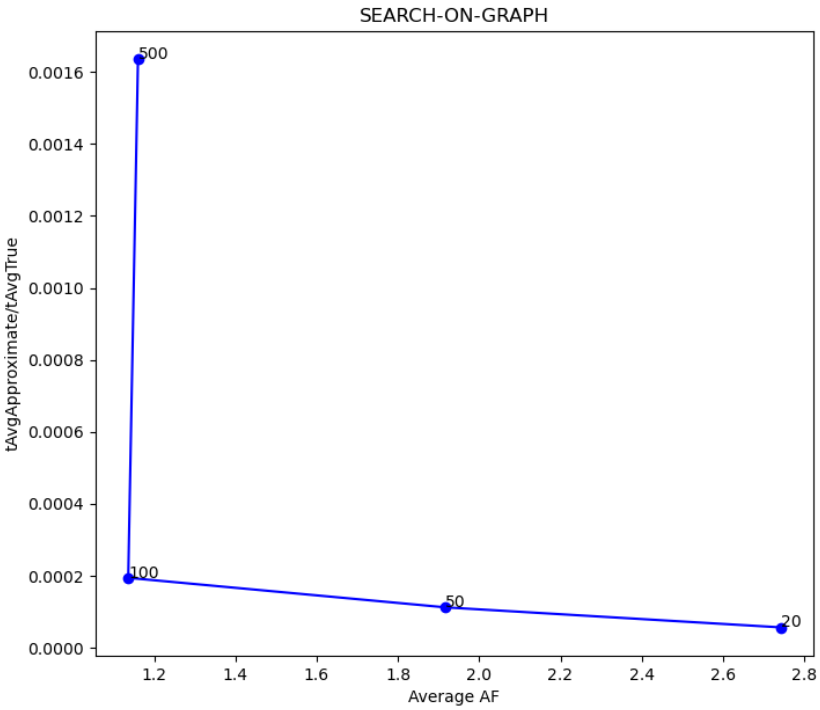
- Latent dimension=20, batch size=64, epochs=5

  Από το διπλανό διάγραμμα παρατηρούμε ότι για l=500, το AAF πλησιάζει αρκετά το 1 με μεγάλη αύξηση του χρόνου όμως.
  Η βέλτιστη παράμετρος είναι η l=100.



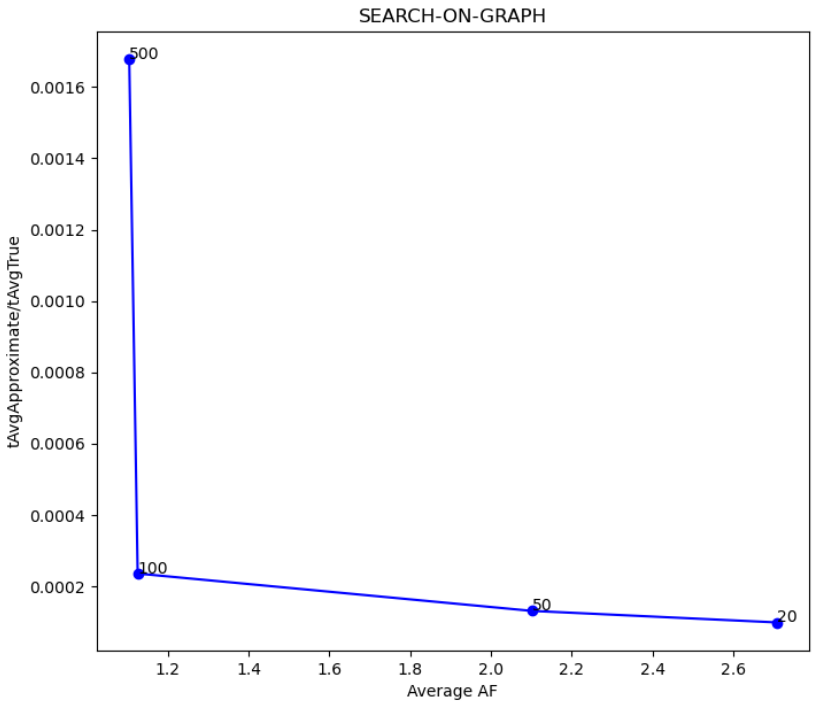
- Latent dimension=10,
  batch size=128, epochs=10

  Από το διπλανό διάγραμμα παρατηρούμε ότι για l=100 και l=500, το AAF πλησιάζει τη βέλτιστη τιμή 1. Για l=20 και l=50, οι τιμές του AAF δεν είναι ικανοποιητικές καθώς πλησιάζουν το 2.8 και το 2 αντίστοιχα.
  Η βέλτιστη παράμετρος είναι η l=100.



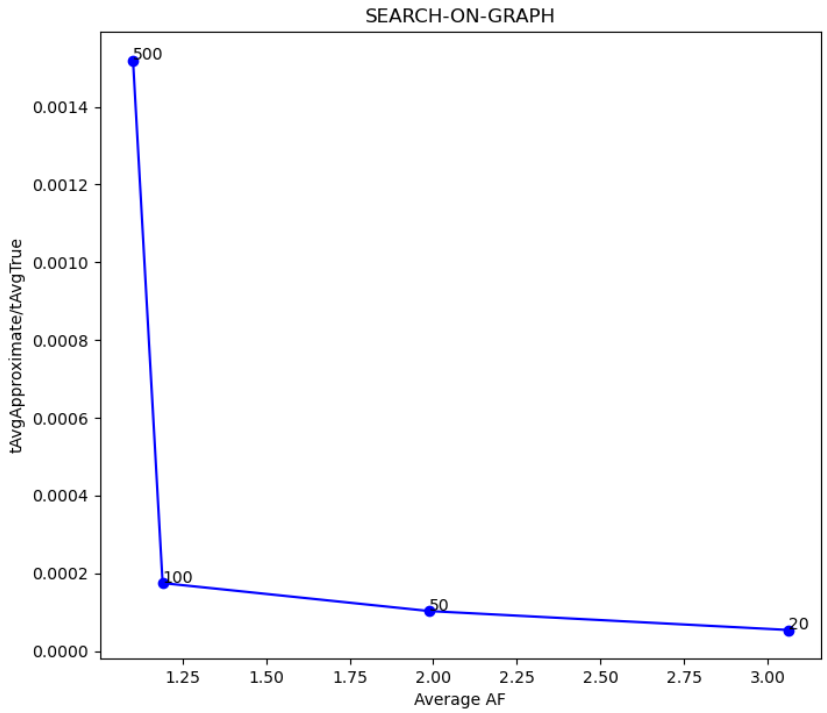
- Latent dimension=20,
  batch size=128, epochs=10

  Από το διπλανό διάγραμμα παρατηρούμε ότι για l=100 και l=500, το AAF πλησιάζει τη βέλτιστη τιμή 1. Για l=20 και l=50, οι τιμές του AAF δεν είναι ικανοποιητικές καθώς είναι μεγαλύτερες του 2.
  Η βέλτιστη παράμετρος είναι η l=100.



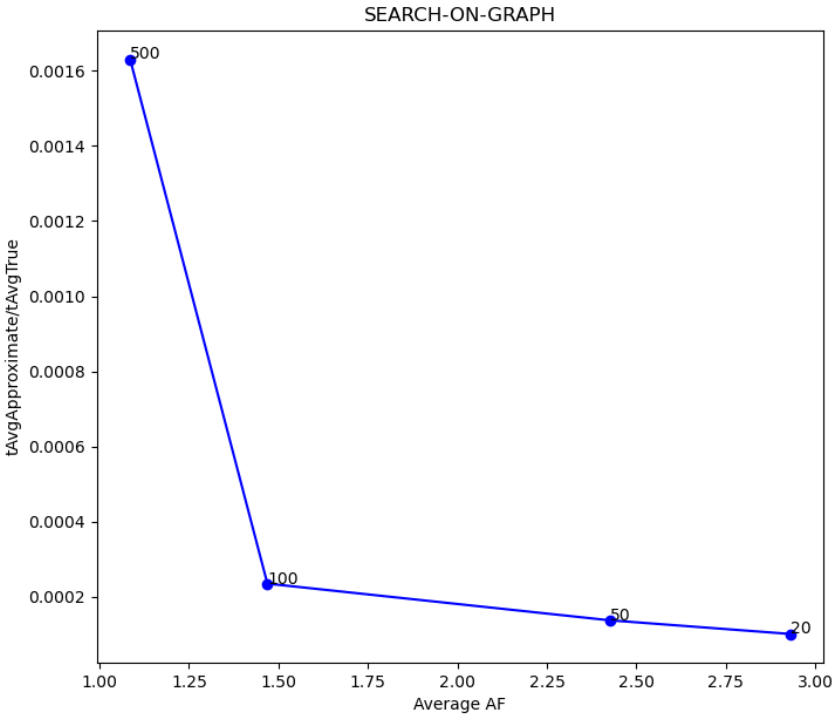
- Latent dimension=10, batch size=64, epochs=10

  Από το διπλανό διάγραμμα παρατηρούμε ότι το AAF αν και ξεκινάει για l=20 με τιμή μεγαλύτερη του 3, για l=100 φτάνει να είναι μικρότερη του 1.25 με ελάχιστη αύξηση του χρόνου.
  Η βέλτιστη παράμετρος είναι η l=100.



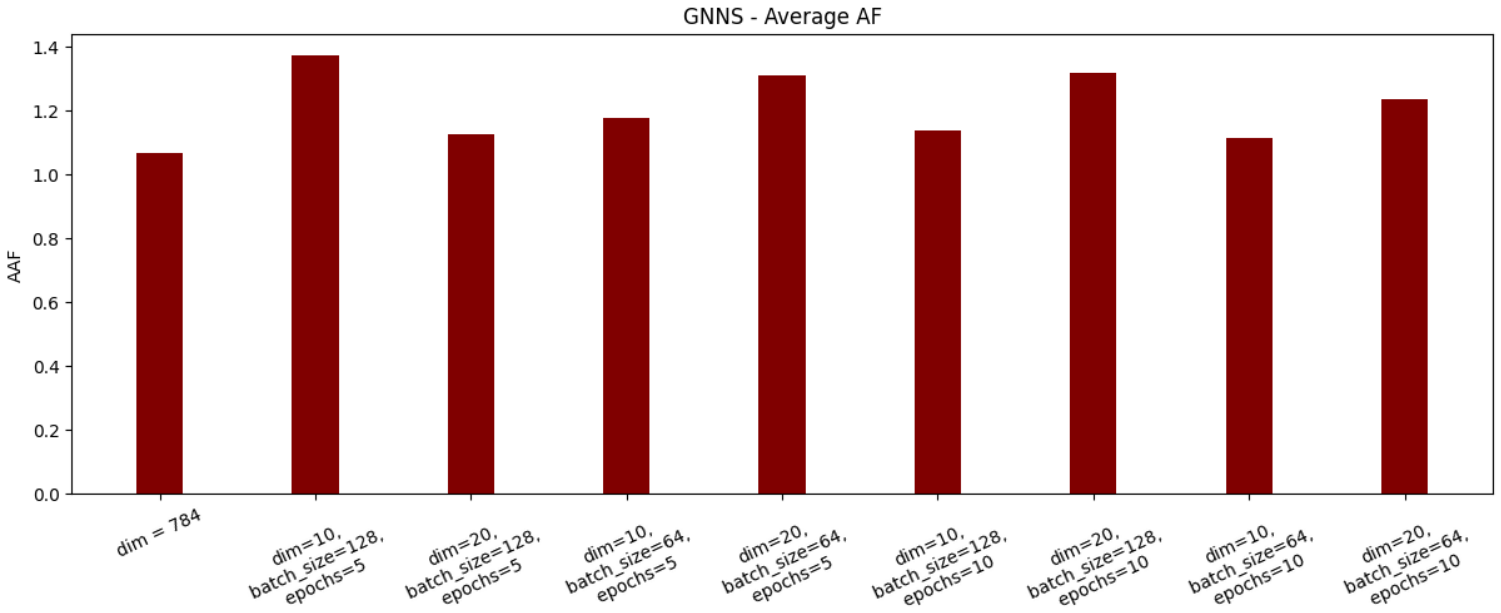
- Latent dimension=20, batch size=64, epochs=10

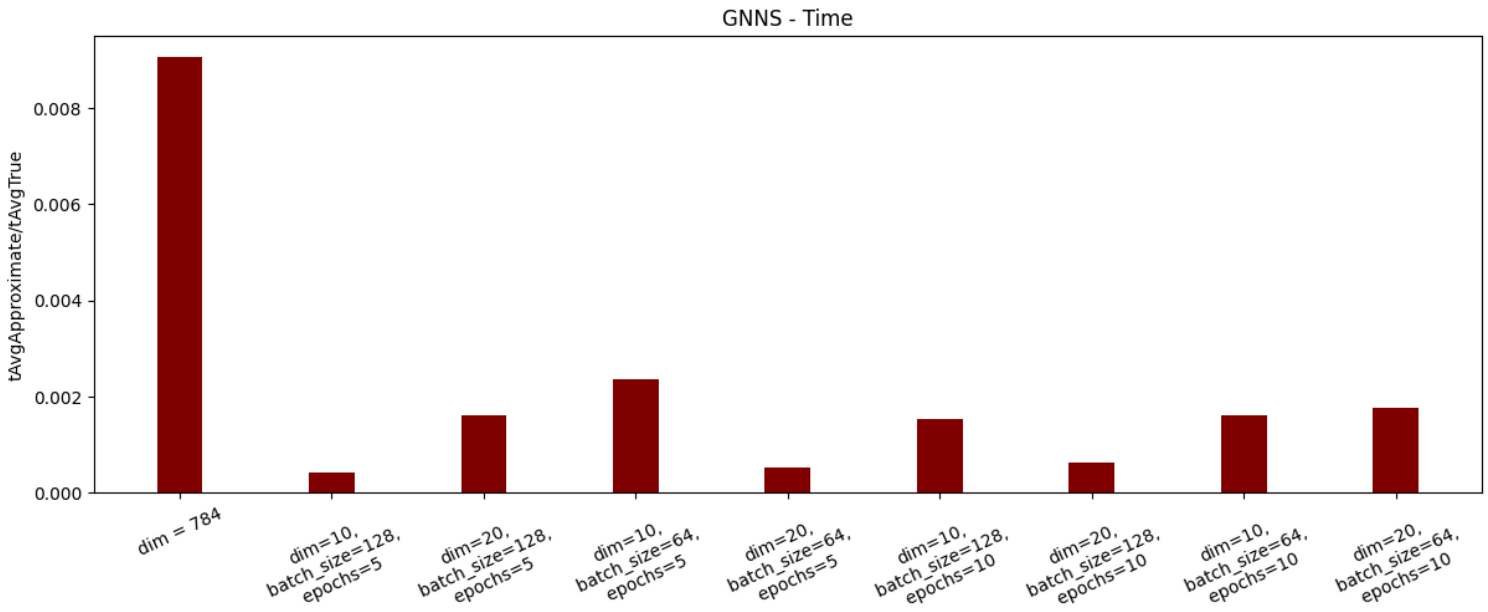
  Από το διπλανό διάγραμμα παρατηρούμε ότι η μόνη τιμή που πλησιάζει το AAF=1 είναι η l=500 με το μειονέκτημα τον αυξημένο χρόνο. Για l=100, το AAF είναι κοντά στο 1.5 με αρκετά μικρότερο χρόνο από l=500.
  Η βέλτιστη παράμετρος είναι η l=100.



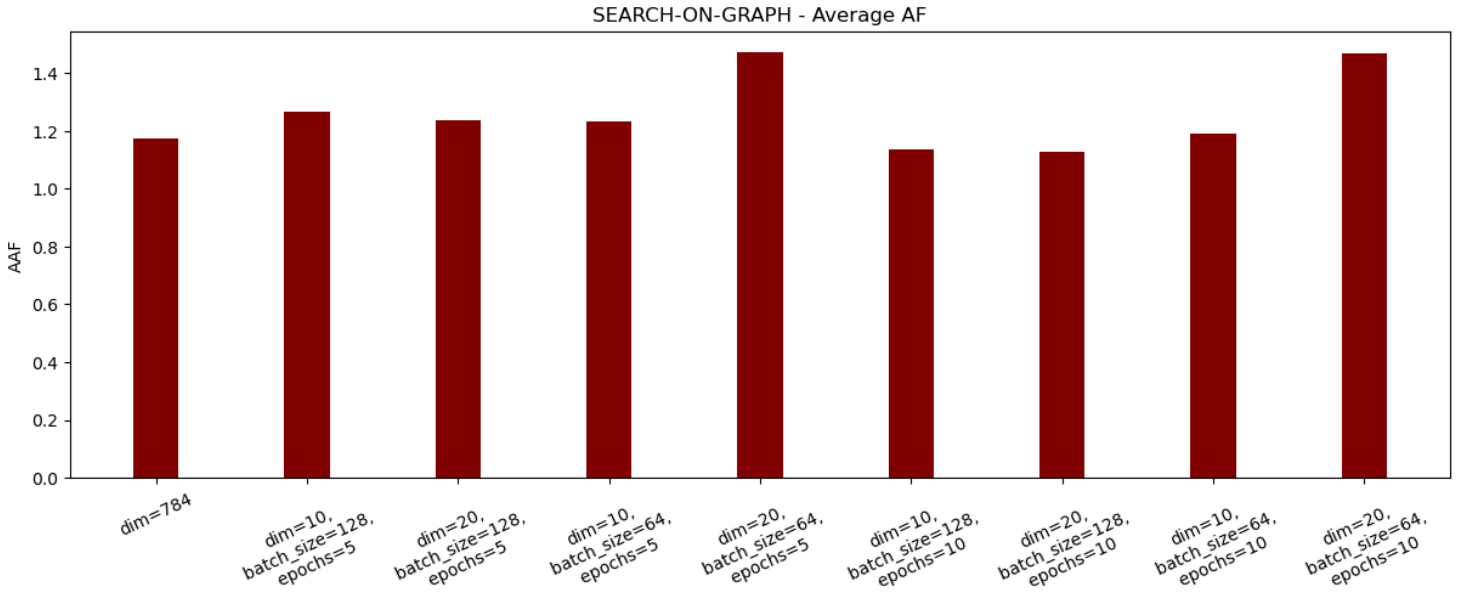
ΣΥΓΚΡΙΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

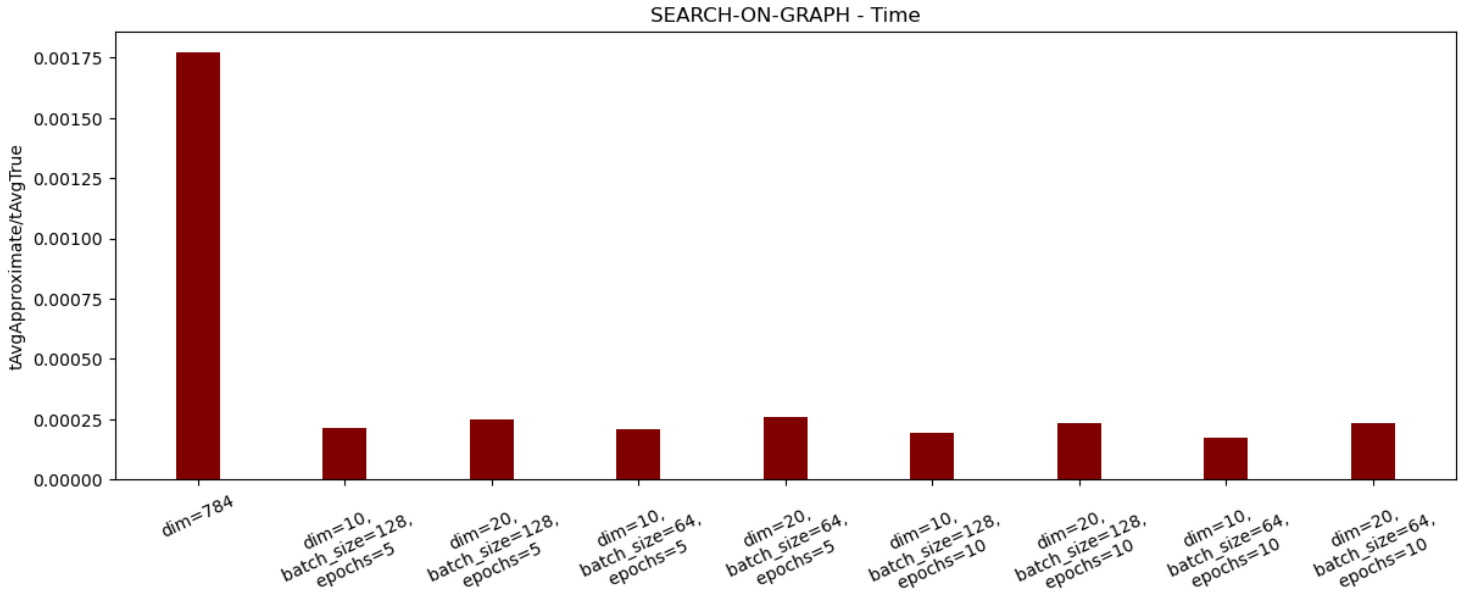
Τα διαγράμματα δημιουργήθηκαν με τις βέλτιστες παραμέτρους που επιλέχθηκαν παραπάνω.





Παρατηρούμε ότι η είσοδος με την αρχική διάσταση (784) πλησιάζει το AAF=1 όμως φαίνεται και ορισμένοι συνδυασμοί με μειωμένη διάσταση το πλησιάζουν εξίσου σε πολύ μικρότερο χρόνο, όπως φαίνεται και από το σχετικό διάγραμμα.





Παρατηρούμε ότι οι συνδυασμοί 10-128-10 και 20-128-10 έχουν καλύτερο AAF συγκριτικά με την αρχική διάσταση σε πολύ λιγότερο χρόνο. Γενικά αρκετοί συνδυασμοί φαίνεται να έχουν AAF μικρότερο ή ίσο του 1.2 .

Από τα παραπάνω διαγράμματα συμπεραίνουμε ότι αρκετοί συνδυασμοί που ελέγχθηκαν με μειωμένη διάσταση φαίνεται να ευνοούν τους συγκεκριμένους αλγορίθμους καθώς δίνουν τιμές AAF κοντινές στη βέλτιστη σε ελάχιστο χρόνο.

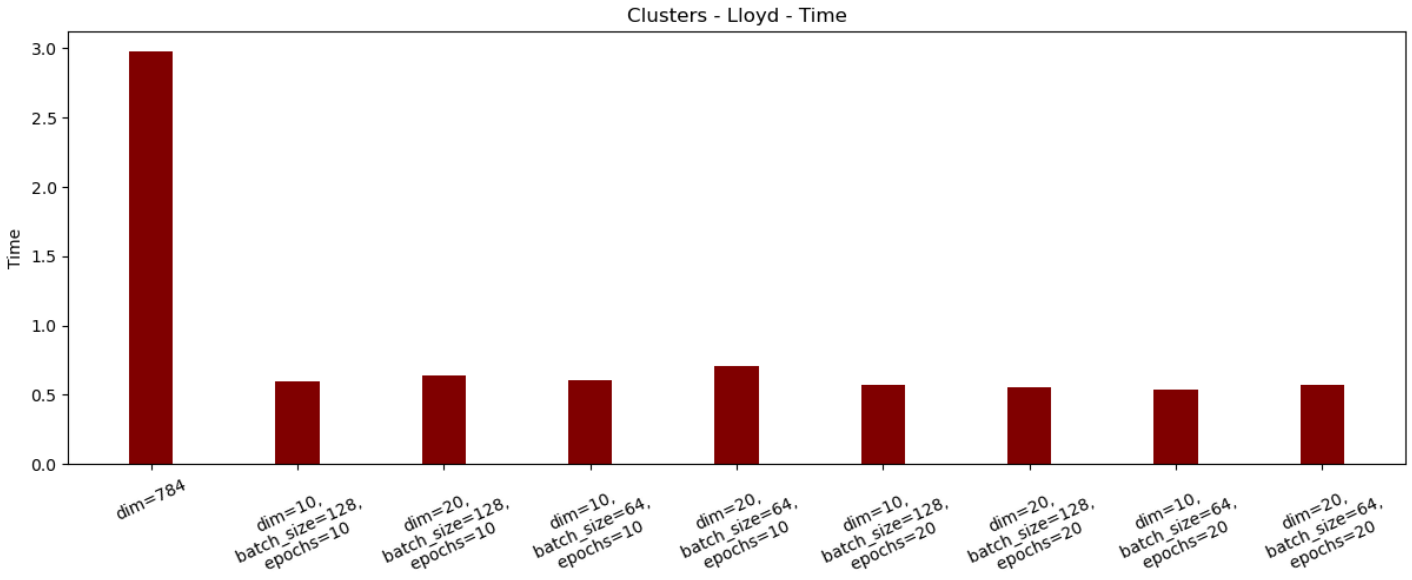
Συνολικά από τα αποτελέσματα των πειραμάτων, ο βέλτιστος συνδυασμός φαίνεται να είναι ο 10-128-10, ο οποίος έχει στο GNNS (AAF= 1.13754, χρόνος≈ 0,0015) και στο MRNG(AAF= 1.13633, χρόνος≈0,00019).

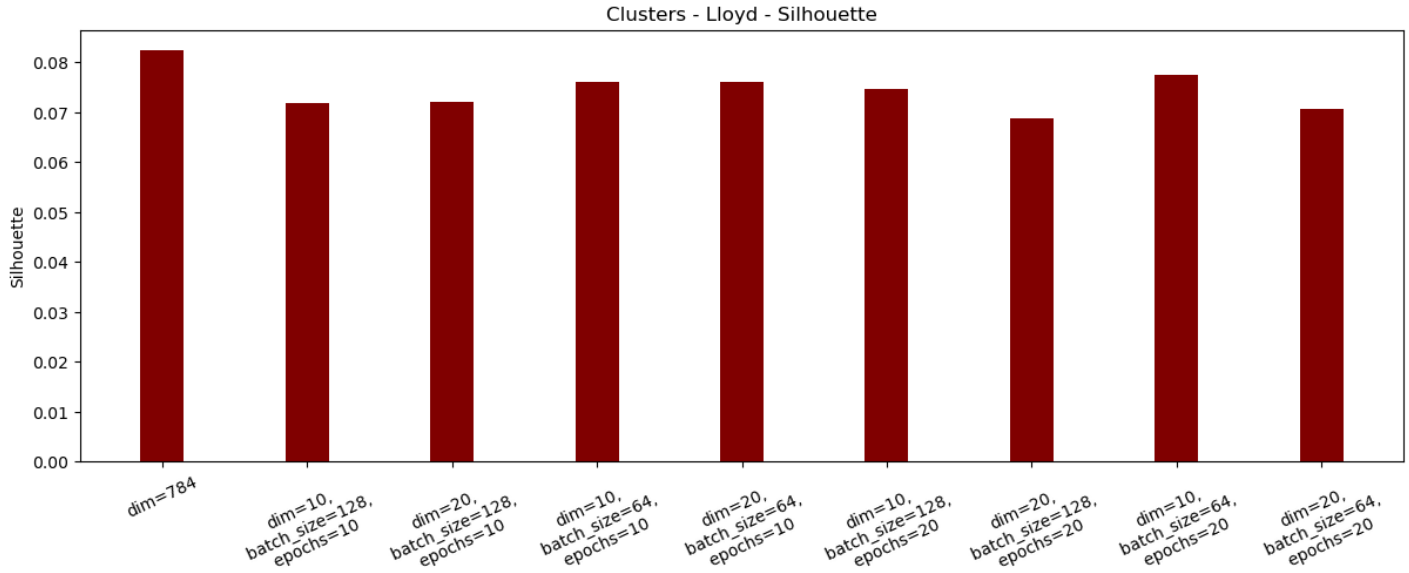
Ερώτημα Γ - Αναφορά

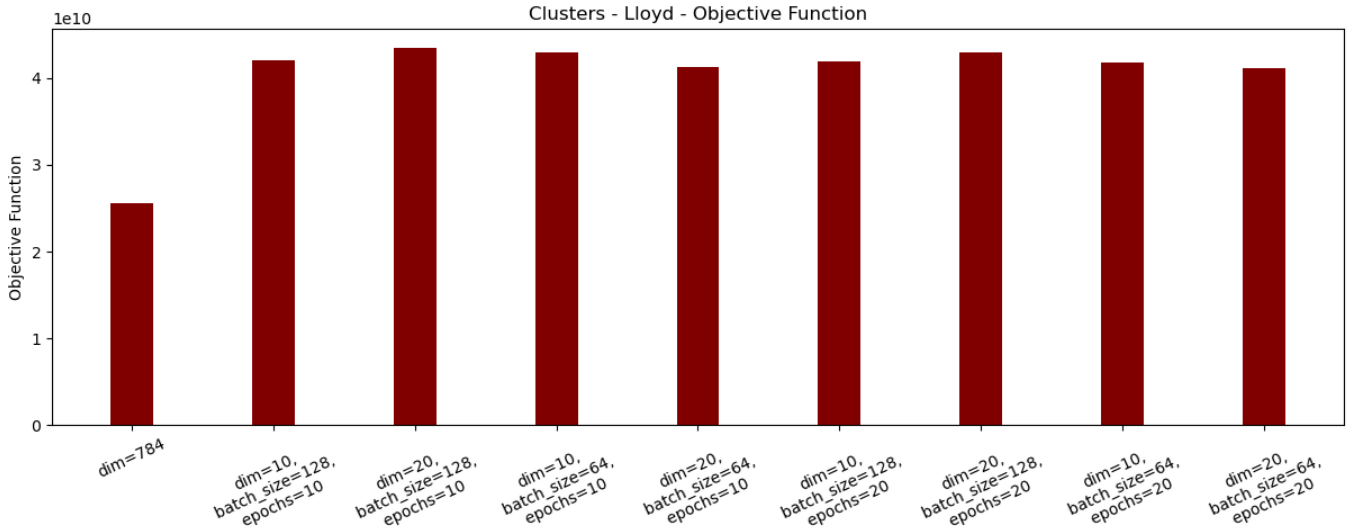
Όλα τα πειράματα έχουν γίνει με dataset αυτό των 10000 σημείων και με τη χρήση του flag -O2 κατά τη μεταγλώττιση. Τα αρχεία εισόδου μειωμένης διάστασης έχουν δημιουργηθεί με το νευρωνικό που βρίσκεται στο reduce.py . Για τις παραμέτρους των αλγορίθμων χρησιμοποιήθηκε το αρχείο cluster.conf με τις default τιμές.

Τα παρακάτω διαγράμματα παρουσιάζουν το χρόνο εκτέλεσης των διαφορετικών μεθόδων συσταδοποίησης, το δείκτη εσωτερικής αξιολόγησης silhouette (stotal) και την τιμή της συνάρτησης στόχου.

- **Μέθοδος Lloyd**

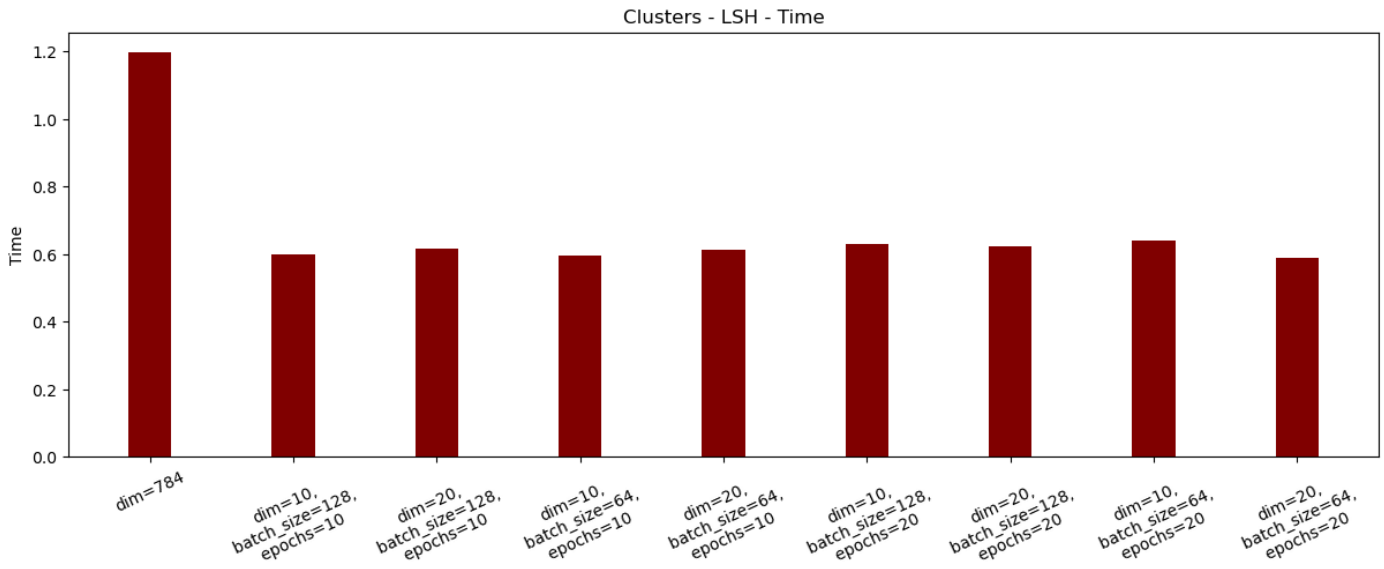


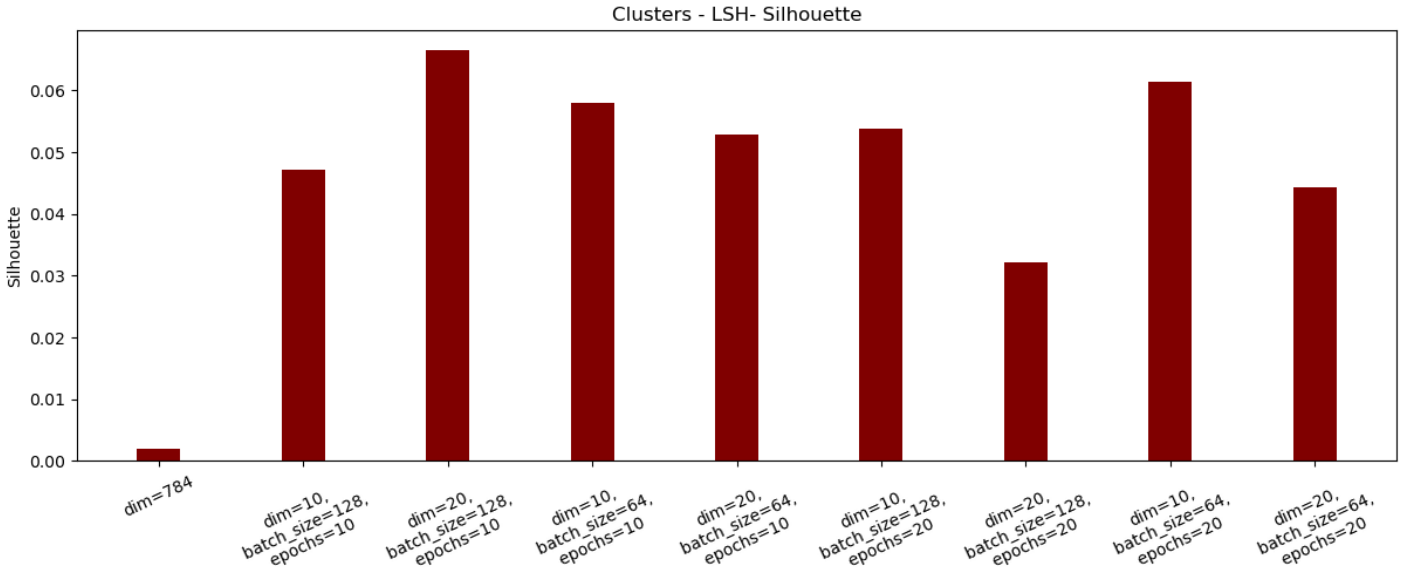


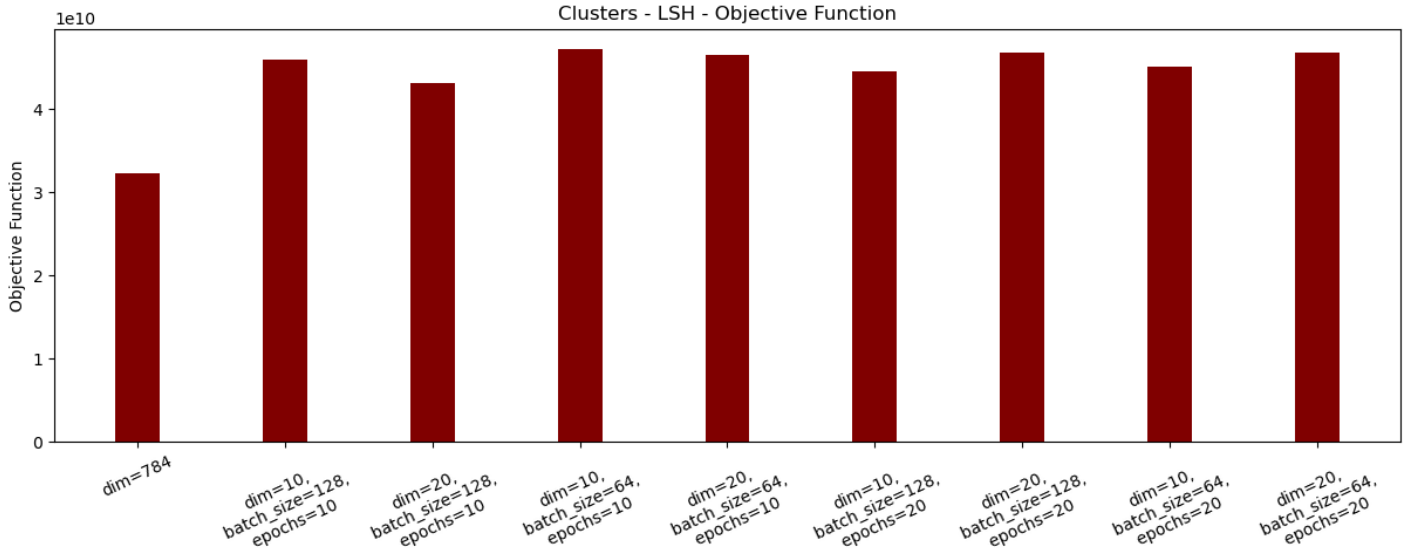


Από τα παραπάνω διαγράμματα, όσον αφορά τη μέθοδο Lloyd, παρατηρούμε ότι για τις εισόδους μειωμένης διάστασης, ο χρόνος εκτέλεσης της συσταδοποίησης είναι πολύ μικρότερος συγκριτικά με αυτόν της αρχικής διάστασης (784). Φαίνεται ότι η αρχική διάσταση έχει τις καλύτερες τιμές στο δείκτη silhouette και στη συνάρτηση στόχου. Γενικά, όμως ο δείκτης silhouette κυμαίνεται σε παρόμοιες τιμές σε όλα τα πειράματα.

- **Μέθοδος Reverse Search με LSH**

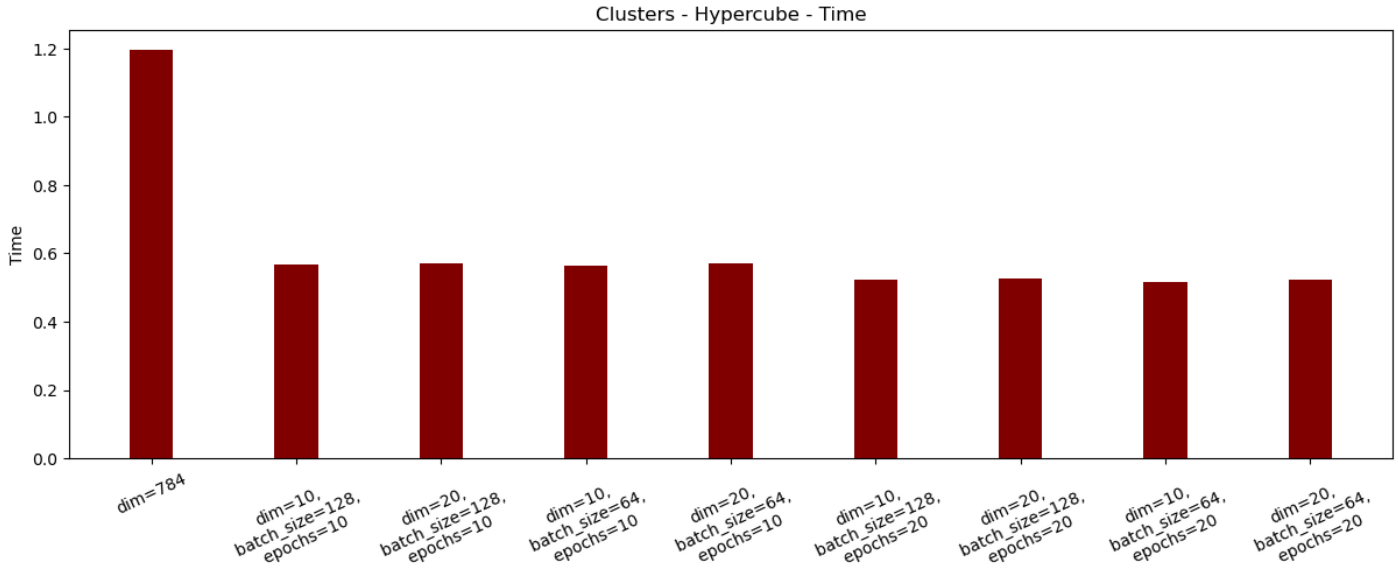


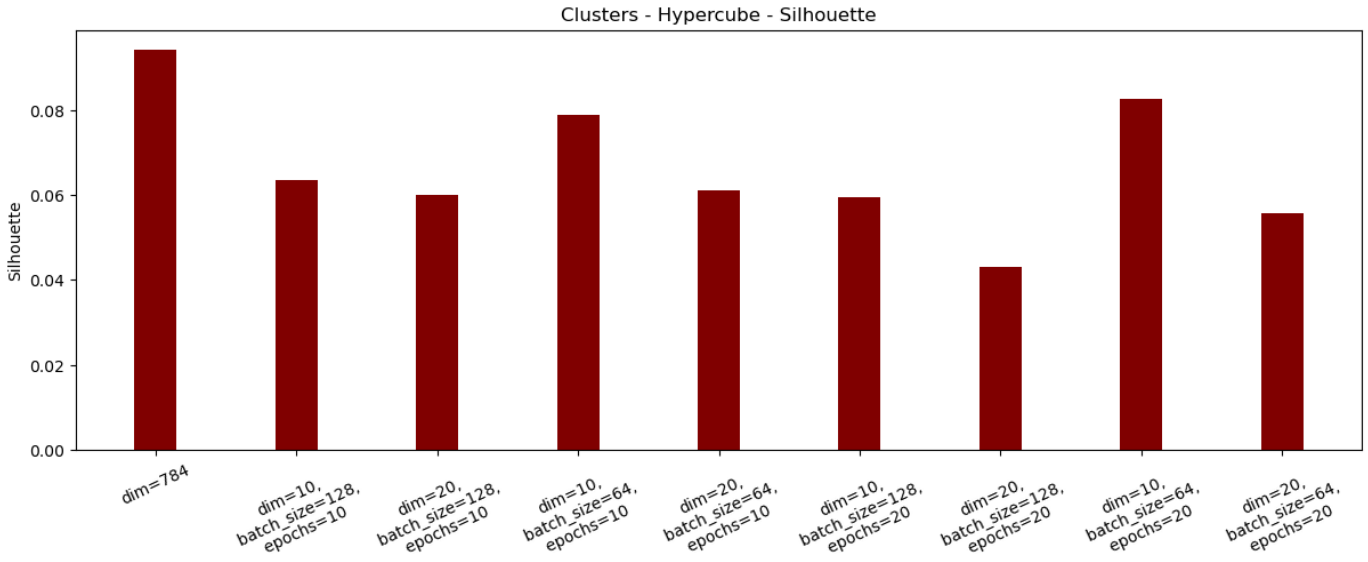


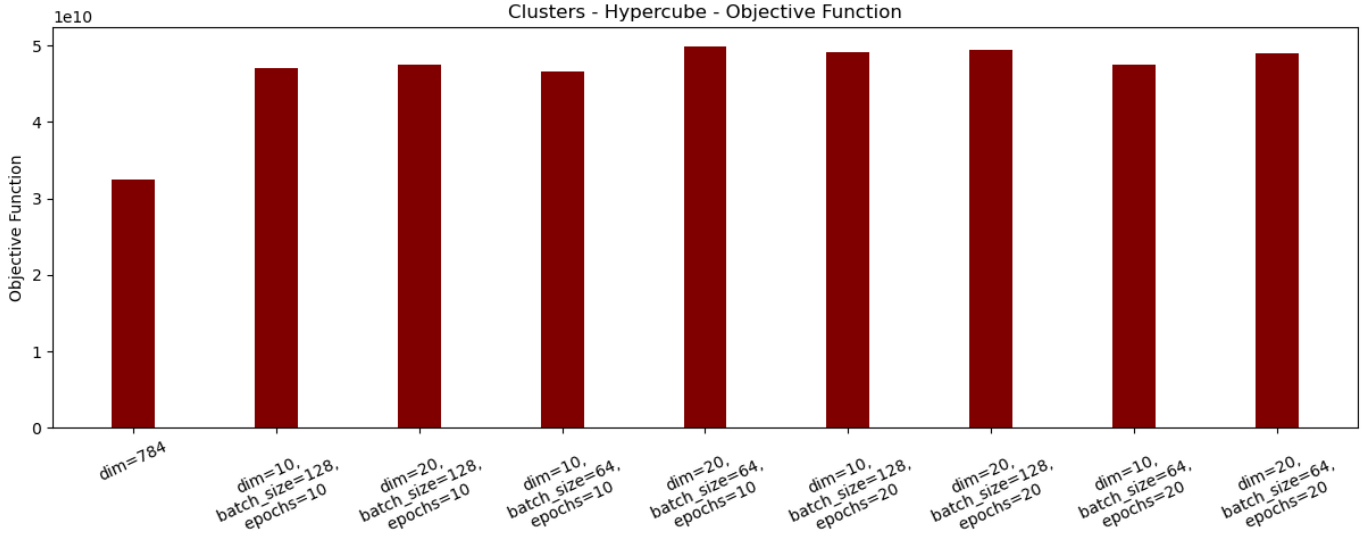


Από τα παραπάνω διαγράμματα, όσον αφορά τη μέθοδο Reverse Search με LSH, παρατηρούμε ότι για τις εισόδους μειωμένης διάστασης, ο χρόνος εκτέλεσης της συσταδοποίησης έχει βελτιωθεί συγκριτικά με αυτόν της αρχικής διάστασης (784) και ο δείκτης silhouette δίνει καλύτερες τιμές. Όσον αφορά τη συνάρτηση στόχου, η είσοδος με την αρχική διάσταση φαίνεται να μειώνει περισσότερο την τιμή της συνάρτησης.

- **Μέθοδος Reverse Search με Hypercube**







Από τα παραπάνω διαγράμματα, όσον αφορά τη μέθοδο Reverse Search με Hypercube, παρατηρούμε ότι για τις εισόδους μειωμένης διάστασης, ο χρόνος εκτέλεσης της συσταδοποίησης έχει βελτιωθεί συγκριτικά με αυτόν της αρχικής διάστασης (784). Ο δείκτης silhouette είναι καλύτερος στην αρχική διάσταση, ενώ φαίνεται ορισμένοι συνδυασμοί να πλησιάζουν αρκετά και άλλοι να έχουν μεγαλύτερη απόκλιση. Όσον αφορά τη συνάρτηση στόχου, η είσοδος με την αρχική διάσταση φαίνεται να μειώνει περισσότερο την τιμή της συνάρτησης.