Κ23γ: Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
3η Προγραμματιστική Εργασία

ΚΑΝΕΛΛΑΚΗ ΜΑΡΙΑ-ΑΝΝΑ – 1115201400060
ΛΙΤΣΑΣ ΝΙΚΟΛΑΟΣ – 1115201400331

1. README

A. Reduce

Το πρόγραμμα εκτελείται με την εντολή python3 reduce.py -d <dataset path> -q <queryset path> -od <output_dataset_filename> -oq <output queryset_filename. Το πρόγραμμα δεν τρέχει χωρίς τα ορίσματα.

(-d ../Datasets/train-images-idx3-ubyte -q ../Datasets/t10k-images-idx3-ubyte -od dataset_output.bin -oq queryset_output.bin)

Χρησιμοποιήθηκαν τα αρχεία της 2ης εργασίας με τις απαραίτητες αλλαγές.

Τα outputs του ερωτήματος Α δημιουργούνται στον φάκελο A_outputFiles.

Αλλαγές που έγιναν:

- reduce: Προσθέθηκαν τα καινούργια ορίσματα, διαβάζεται πλέον εκτός από το trainset και το testset (το οποίο χρησιμοποιείται ως validation set) και προσθέθηκε στα inputs που ζητούνται από το χρήστη το latent_vector_space. Επίσης δημιουργούνται τα output files με τα latent vectors που παράχθηκαν από τον autoencoder, μετά από κανονικοποίηση (χρησιμοποιήθηκε η MinMaxScaler της sklearn και τα αποτελέσματα μετατράπηκαν σε ints).

- encoder: Προσθέθηκαν στο τέλος δύο νέα layers, ένα Flatten και ένα Dense, που παράγουν τα tent vectors

- decoder: Προσθέθηκε στην αρχή ένα Dense layer και όλα τα convolutional layers μετατράπηκαν σε Convolutional Tranpose.

  Και στα δύο αρχεία του autoencoder προσθέθηκαν Dropout layers για την αντιμετώπιση του overfitting.

- loadDataset: Αποτελείται από 2 συναρτήσεις που διαβάζουν ένα dataset και ένα labelset για τα datasets που μας δίνονται. Δεν έγινε κάποια αλλαγή.

- output: Νέο αρχείο που περιλαμβάνει 1 συνάρτηση για τη δημιουργία ενός outputfile και την εγγραφή των στοιχείων του (magic number, αριθμός εικόνων, γραμμών και στηλών) και των διανυσμάτων αναπαράστασης όλων των εικόνων σε αυτό.

B. Search New Space

Το αρχείο μεταγλωττίζεται με την εντολή make. Το makefile βρίσκεται στον αρχικό φάκελο της εργασίας και μεταγλωττίζει ταυτόχρονα όλα τα εκτελέσιμα που αφορούν τη C++.

Το πρόγραμμα εκτελείται με την εντολή:
./search –d <input file original space> -i <input file new space> –q <query file original space> -s <query file new space> –k <int> -L <int> -o <output file>

(./search -d Datasets/train-images-idx3-ubyte -i A_outputFiles/dataset_output.bin -q Datasets/t10k-images-idx3-ubyte -s A_outputFiles/queryset_output.bin -k 5 -L 10 -o B_output.txt)

Χρησιμοποιήθηκαν τα αρχεία της 1ης εργασίας με τις απαραίτητες αλλαγές.

Αλλαγές που έγιναν:

- lshMain: Στη main της 1ης εργασίας, αλλάχθηκαν οι παράμετροι του προγράμματος, αφαιρέθηκε η εκτέλεση του Range Search, και προσθέθηκε άλλο ένα αντικείμενο LSH, το οποίο κρατάει τις παραμέτρους και τα αρχεία και εκτελεί τις λειτουργίες για τα sets της νέας διάστασης, και άλλο ένα Dataset, όπου αποθηκεύεται το νέο input dataset. Ο approximate neighbour (μέσω LSH) και ο exact neighbour, εκτελούνται πλέον και για τη νέα διάσταση και για την παλιά και τροποποιήθηκε το output αρχείο.

- lshMainUtils: Αποτελείται από συναρτήσεις που χρησιμεύουν για λειτουργίες της main. Τροποποιήθηκαν ορισμένες από τις συναρτήσεις, ώστε να έχουν διαφοροποιημένη συμπεριφορά, ανάλογα με το dataset που τους δόθηκε.

- Τροποποιήθηκε το ImageData αρχείο (κρατάει τις απαραίτητες πληροφορίες που χρειάζονται για μία εικόνα), ώστε πλέον να λειτουργεί με οποιαδήποτε διάσταση εικόνας (στην 1η εργασία δε δούλευε σωστά δυναμικά). Επίσης, όταν δημιουργείται αντικείμενο ImageData, λαμβάνει διπλάσιο μέγεθος εάν πρόκειται για

Δυστυχώς για να λειτουργεί δυναμικά το ImageData χρησιμοποιήθηκαν vectors και το πρόγρμμα έχει γίνει ακόμα πιο αργό.

Όλα τα υπόλοιπα αρχεία του προγράμματος παρέμειναν όπως έχουν.

Παρόλο που ο Cube δεν έχει τροποποιηθεί, περιλαμβάνονται ορισμένα από τα αρχεία του, καθώς χρειάζονται για τη μεταγλώττιση του LSH.

C. EMD Distance

Το πρόγραμμα εκτελείται με την εντολή:

python3 search.py -d Datasets/train-images-idx3-ubyte –q Datasets/t10k-images-idx3-ubyte -l1 Datasets/train-labels-idx1-ubyte -l2 Datasets/t10k-labels-idx1-ubyte -o C_output.txt -EMD

(-d ../Datasets/train-images-idx3-ubyte -q ../Datasets/t10k-images-idx3-ubyte -l1 ../Datasets/train-labels-idx1-ubyte -l2 ../Datasets/t10k-labels-idx1-ubyte -o C_output.txt -EMD)

Το ερώτημα υλοποιήθηκε σε python3.8. Υλοποιήθηκε και η Manhattan σε python, ώστε να συγκριθούν τα αποτελέσματα στην ίδια γλώσσα.

Οι εικόνες κανονικοποιούνται κατα την διάρκεια της ανάγνωσης του dataset και αποθηκεύονται σε 2x2 numpy arrays.

Η υλοποίηση της Manhattan λειτουργεί σωστά, του EMD όμως δεν παράγει σωστά αποτελέσματα.

Αρχεία:

- search.py: Διαβάζει όλα τα datasets που δέχεται το πρόγραμμα και για κάθε query ψάχνει τους 10 κοντινότερους γείτονες με EMD και Manhattan. Εκτυπώνει το C_output.txt με τις συγκρίσεις.
- distance.py: Περιλαμβάνει τις συναρτήσεις για την υλοποίηση των EMD και Manhattan.
- ImageData.py: Class αντίστοιχη με εκείνη στα αρχεία της c++, που κρατάει πληροφορίες για μία εικόνα (αριθμό, διάσταση και το διάνυσμα).
- neighbour.py: Περιλαμβάνει μία συνάρτηση για τον υπολογισμό των κοντινότερων γειτόνων με οποιαδήποτε από τις 2 μεθόδους και μία για την επαλήθευση της ορθότητάς τους.
- LoadDatasetBytes.py: Αλλάχθηκε το loadDataset του ερωτήματος Α, ώστε να διαβάζονται πλέον οι εικόνες σε αντικείμενα ImageData. Πρακτικά μεταφράστηκε το αντίστοιχο διάβασμα από την 1η εργασία.

2. Παρατηρήσεις και Αποτελέσματα

A. reduce

Έγιναν δοκιμές με διάφορες τιμές των υπερπαραμέτρων.

- Αριθμός layers:
  Δοκιμάστηκαν όλοι οι αριθμοί έως το 7 (για μεγαλύτερους αριθμούς η εκτέλεση σταματούσε κάποια στιγμή γιατι τελείωνε τη μνημη RAM).
  Παρατηρήσαμε τα βέλτιστα αποτελέσματα για τον αριθμό 6, καθώς για να τρέξουμε το πρόγραμμα με 7 έπρεπε να μειωθούν αισθητά οι υπόλοιπες υπερπαραμέτροι.
  Γενικά το loss μειωνόταν όσο αυξάνονταν τα layers.

- Μέγεθος filters:
  Δοκιμάστηκαν διάφοροι αριθμοί.
  Παρατηρήσαμε τα βέλτιστα αποτελέσματα για τον αριθμό 32.
  Γενικά το loss μειωνόταν όσο αυξάνονταν το μέγεθος των filters μέχρι το 32, και μετά άρχισε να αυξάνεται το validation loss.

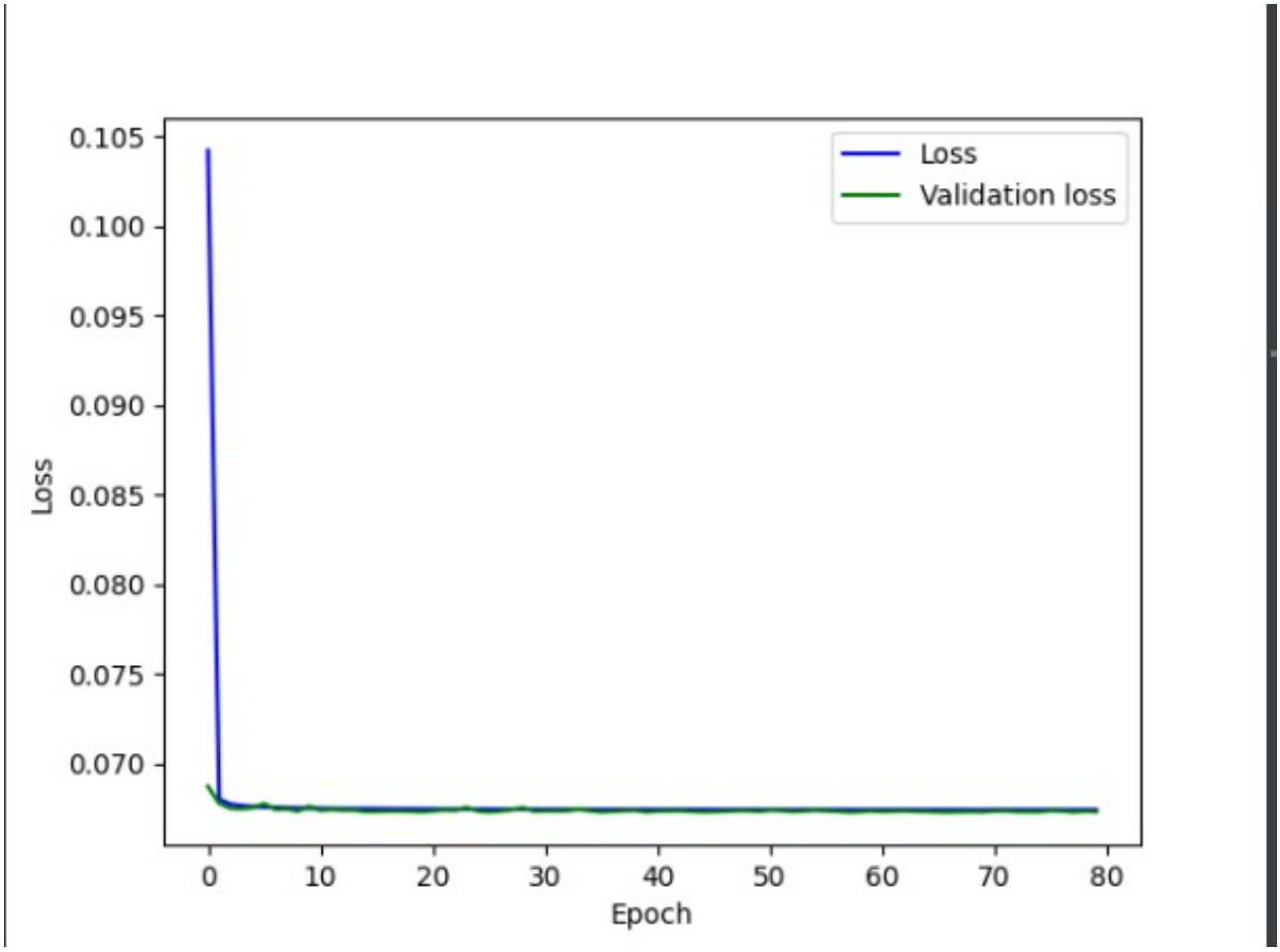
- Αριθμός filters:
  Δοκιμάστηκαν διάφοροι μικροί αριθμοί.
  Παρατηρήσαμε τα βέλτιστα αποτελέσματα για τον αριθμό 7. Γενικώς μέχρι το 7 όσο αυξανόταν το πλήθος των filters τα αποτελέσματα βελτιωνόντουσαν, ενώ για αριθμούς >7, γινόταν overfitting.

- Αριθμός epochs:
  Δοκιμάστηκαν διάφοροι αριθμοί έως το 100. Γενικώς η εναλλαγή του αριθμού των epochs δεν επηρέαζε σημαντικά τα αποτελέσματα. Όσο ο αριθμός αυτός αυξανόταν τα αποτελέσματα βελτιωνόντουσαν ελάχιστα. Τα βέλτιστα τα παρατηρήσαμε για 80. Μετά το 80 τα αποτελέσματα χειροτέρευαν ελάχιστα.

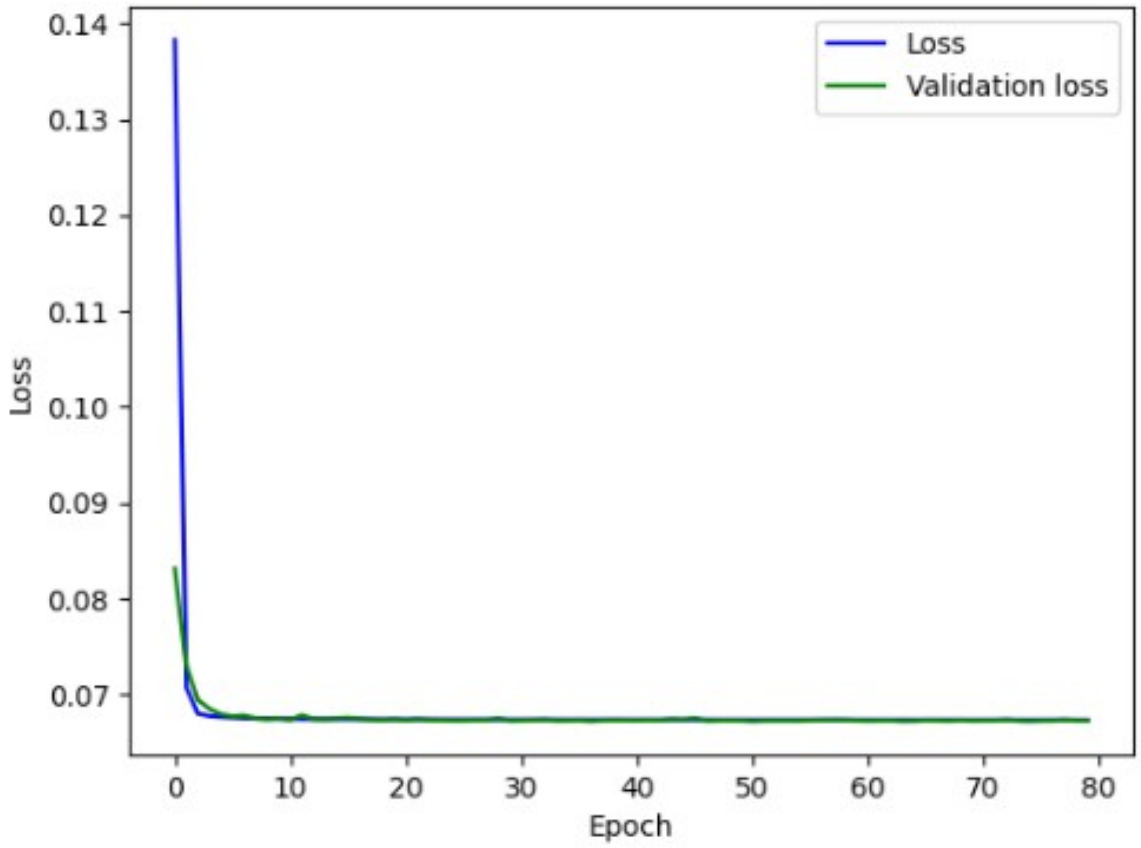
- Batch_size:
  Δοκιμάστηκαν οι αριθμοί {64,128, 256, 512}.
  Παρατηρήσαμε τα βέλτιστα αποτελέσματα για τον αριθμό 64.
  Γενικά όσο αυξάνεται το batch size παρατηρουμε μεγαλύτερη ταχύτητα στην εκτέλεση του προγράμματος αλλα παρουσιάζεται overfitting και αυξημένο loss.
  Με τους αριθμούς 64 και 128 φαίνεται να μην υπαρχει overfitting και έχουμε σταθερά χαμηλά αποτελέσματα σε loss/validation loss.

- Latency:
  Δοκιμάστηκαν διάφοροι αριθμοί στο [1,784] και παρατηρήθηκε το βέλτιστο αποτέλεσμα με διάσταση 5. Ξεκινήσαμε με την default τιμή (10). Κατεβαίνοντας μέχρι το 5 τα αποτελέσματα βελτιωνόντουσαν ένω τα πιο χαμηλά νούμερα έβγαζαν χειρότερα αποτελέσματα. Για τιμές μεγαλύτερες του 10 τα αποτελέσματα ποικίλουν. Δεν υπήρχε κάποια σταθερή βελτίωση ή μείωση, οπότε δεν μπορούσαμε να βγάλουμε κάποιο γενικευμένο συμπέρασμα.

Στη συνέχεια περιλαμβάνονται ενδεικτικά διαγράμματα απο τα πειράματα που εκτελέσαμε. Τα πρωτα διαγράμματα παρουσιάζουν τα καλύτερα αποτελέσματα που επιτεύχθηκαν.

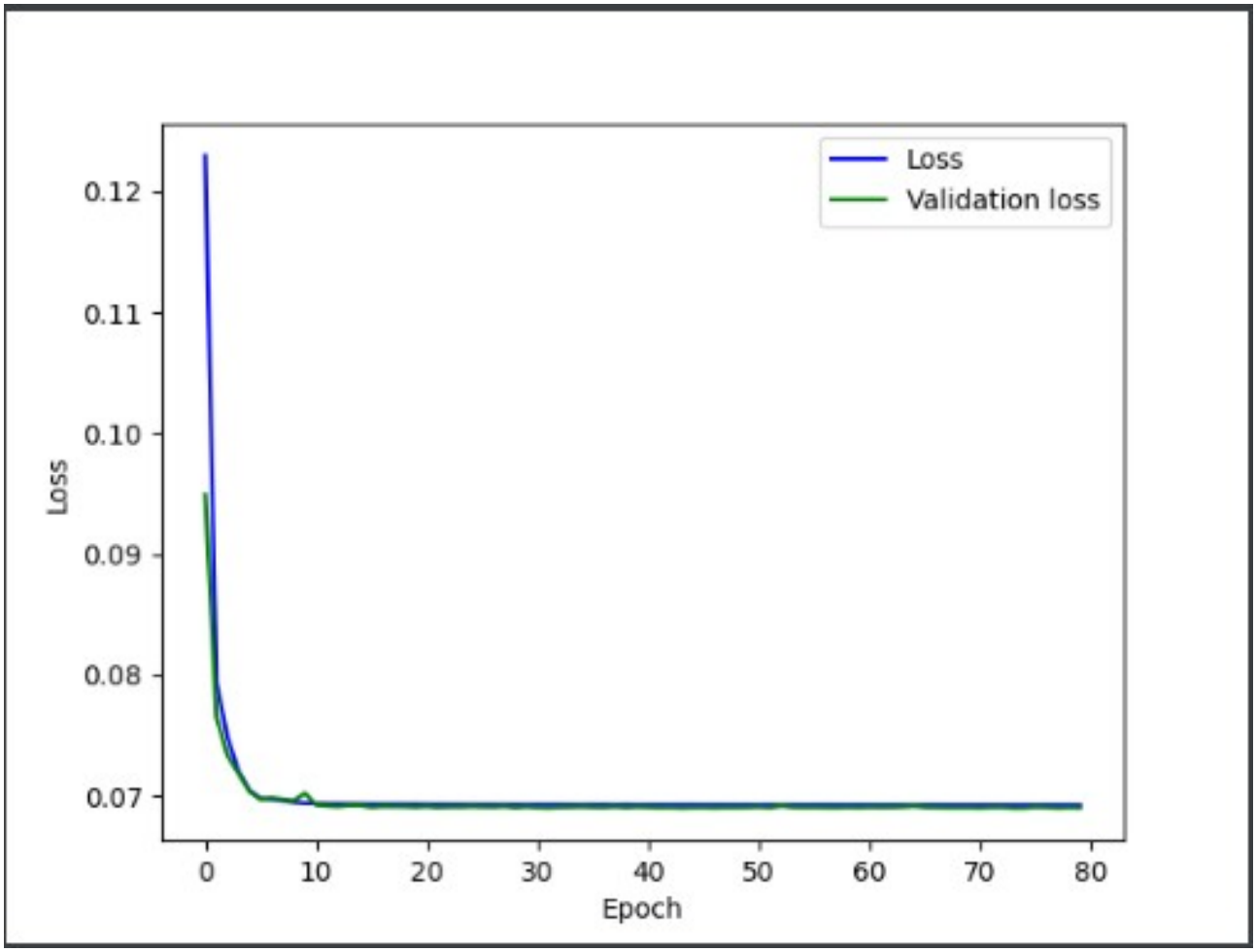


layer_num = 6
filter_size = 32
filter_num = 7
epochs = 80
batch_size = 64
latent_space = 5

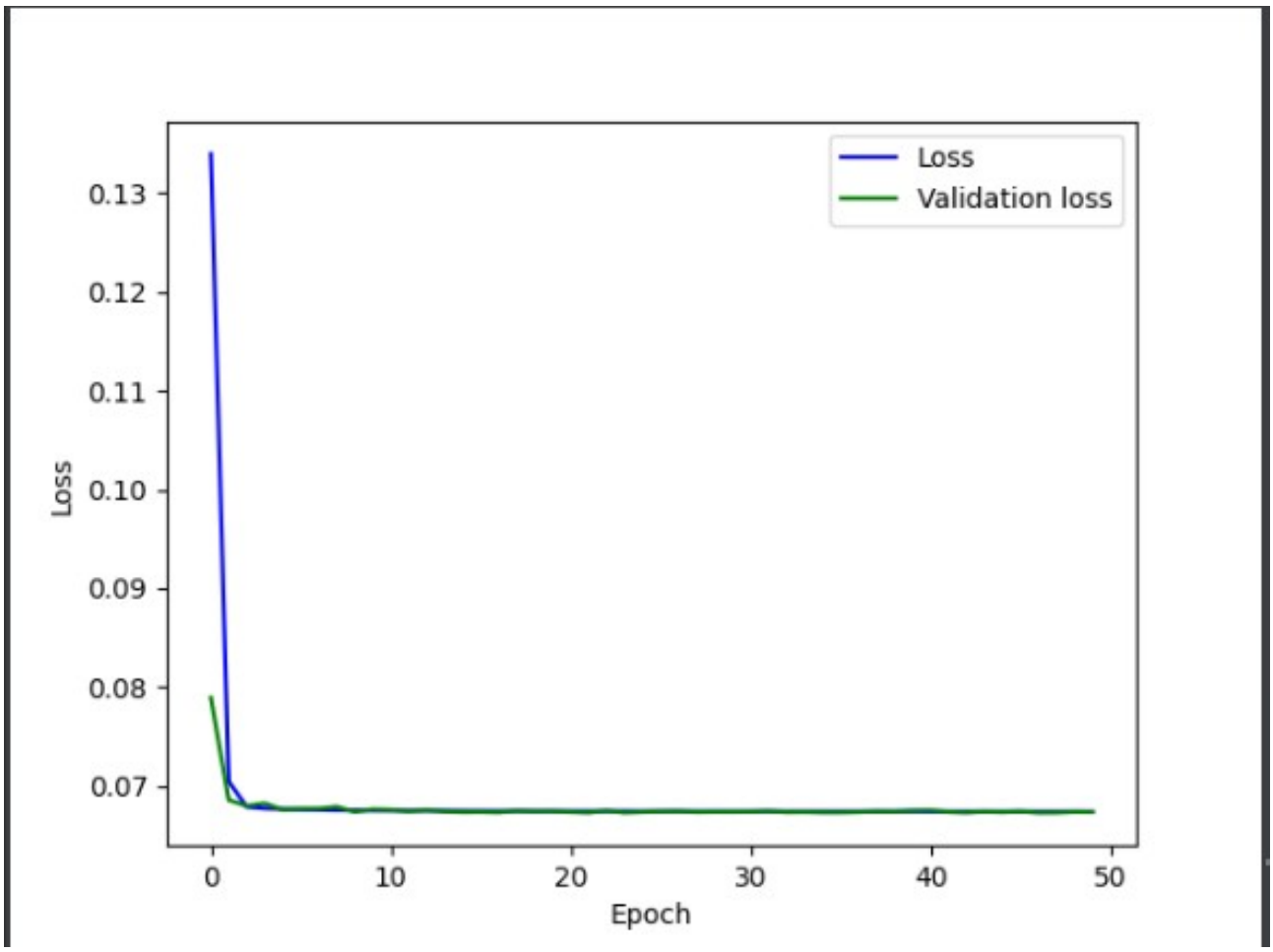


layer_num = 6
filter_size = 32
filter_num = 7
epochs = 80
batch_size = 128
latent_space = 10

Έδω παρατηρούμε το αποτέλεσμα των παραμέτρων που προαναφέρθηκαν. Απο τα πρώτα 5 epoch βλέπουμε σταθερά βέλτιστα αποτελέσματα.

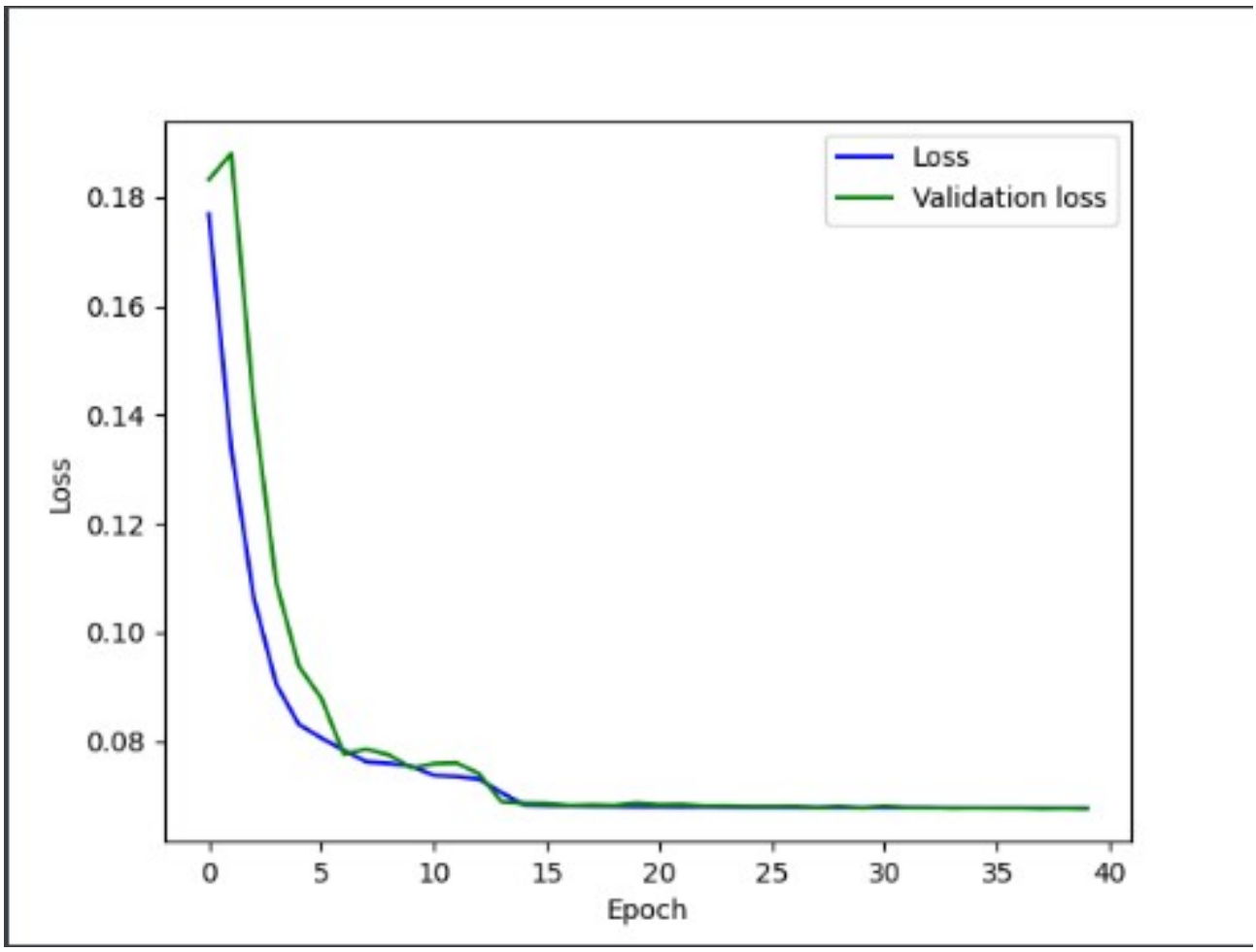


Η φωτογραφία αυτή δείχνει το αποτέλεσμα του προγράμματος με τις ίδιες υπερπαραμέτρους, είναι παρόμοιο με το βέλτιστο, αλλά εδώ βλέπουμε ότι στην αρχή το validation loss ήταν λιγο μεγαλύτερο.



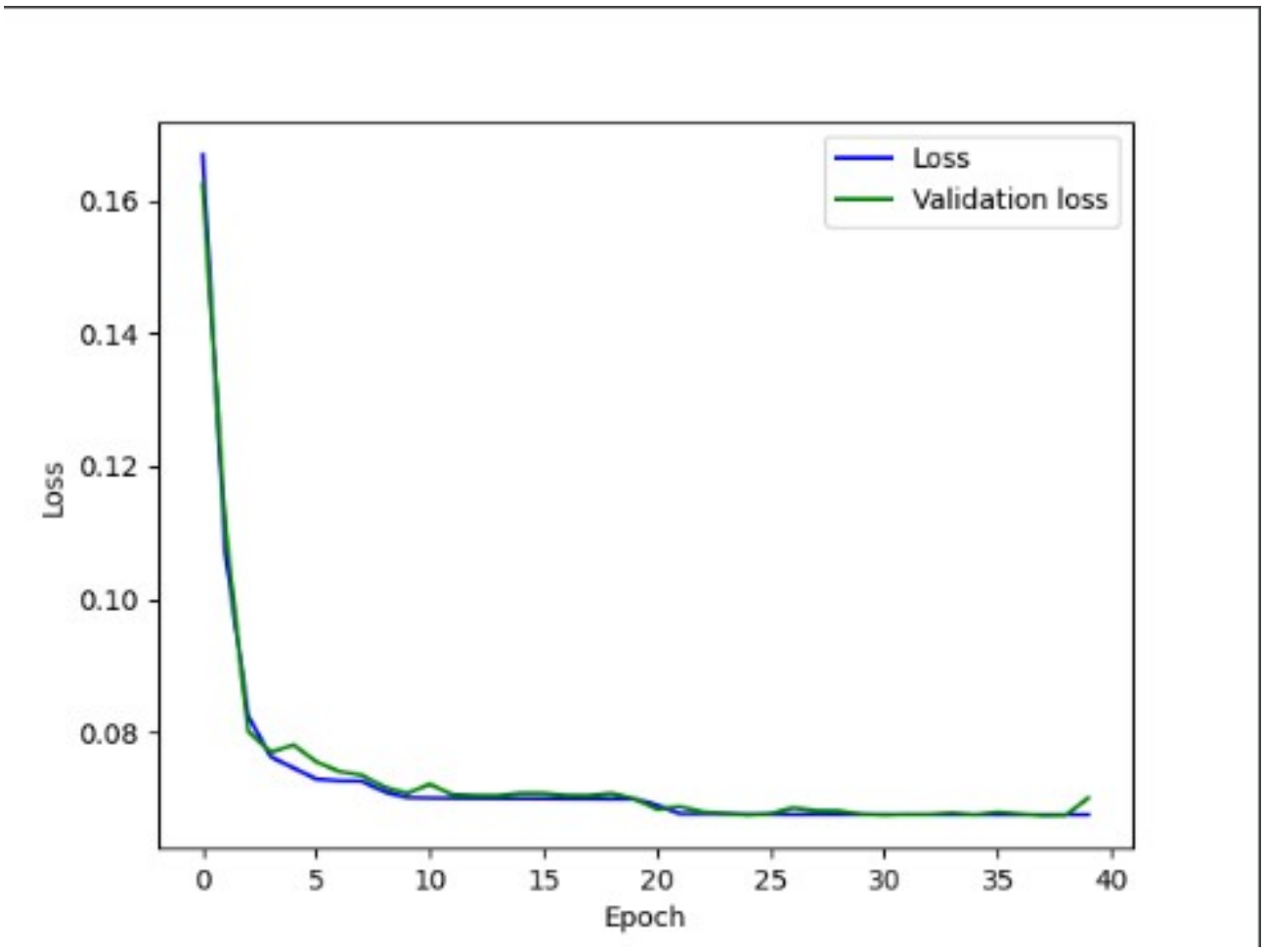
Έδω παραθέτουμε ένα ακόμα βέλτιστο αποτέλεσμα με το χαμηλότερο validation loss που καταφέραμε να επιτύχουμε.

layer_num = 6
filter_size = 32
filter_num = 7
epochs = 50
batch_size = 128
latent_space = 5



Σε αυτή τη φωτογραφία βλέπουμε την επιρροή του μεγαλύτερου batch size στο πρόγραμμα και το πόσο αυξάνεται το validation loss, αν και φαίνεται να σταθεροποιείται κατα τη διάρκεια του προγράμματος. Οι παράμετροι για αυτό το πείραμα ήταν:

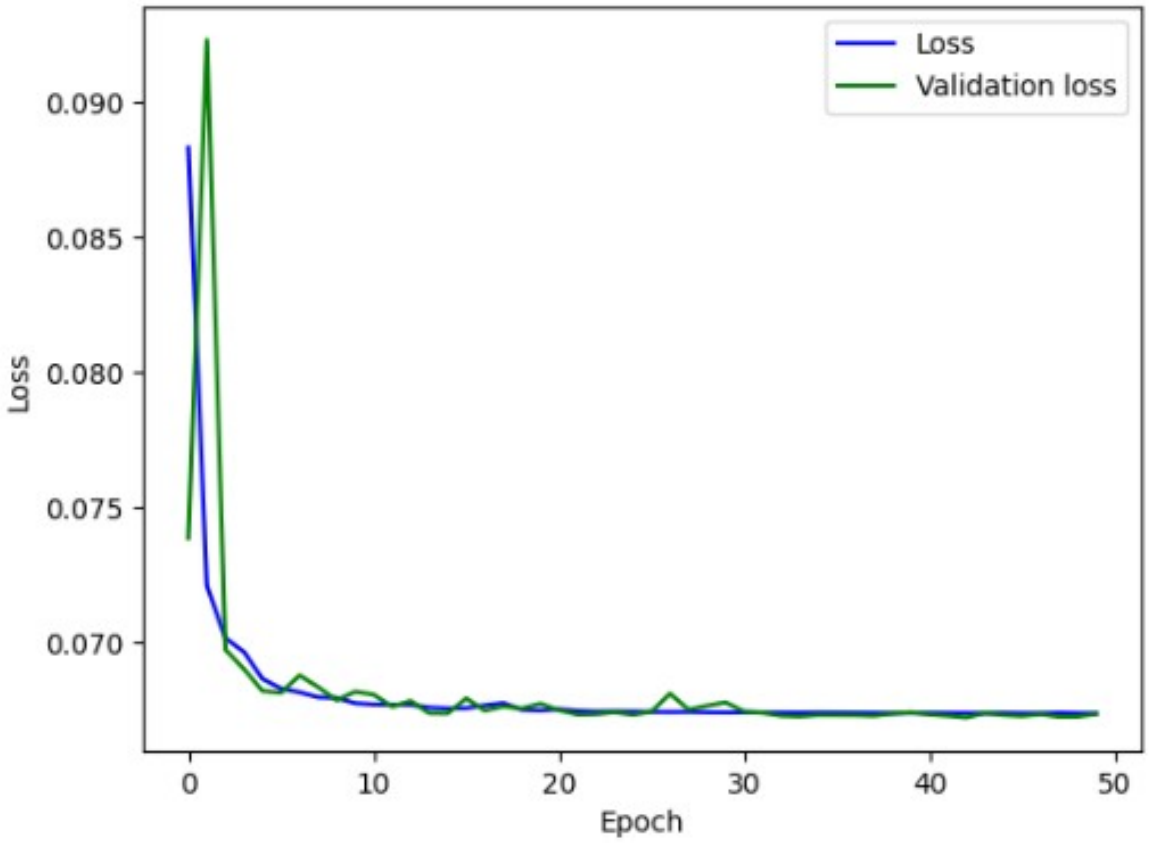
layer_num = 6
filter_size = 32
filter_num = 5
epochs = 40
batch_size = 512
latent_space = 10



Ένα τελευταίο παράδειγμα με το batch size (=256) και τα αρνητικά αποτελέσματα που δημιουργεί.

layer_num = 6
filter_size = 32
filter_num = 5
epochs = 40
batch_size = 256
latent_space = 10

Στη συνέχεια θα παρουσιάσουμε λίγα διαγράμματα με την επιρροή του filter size.

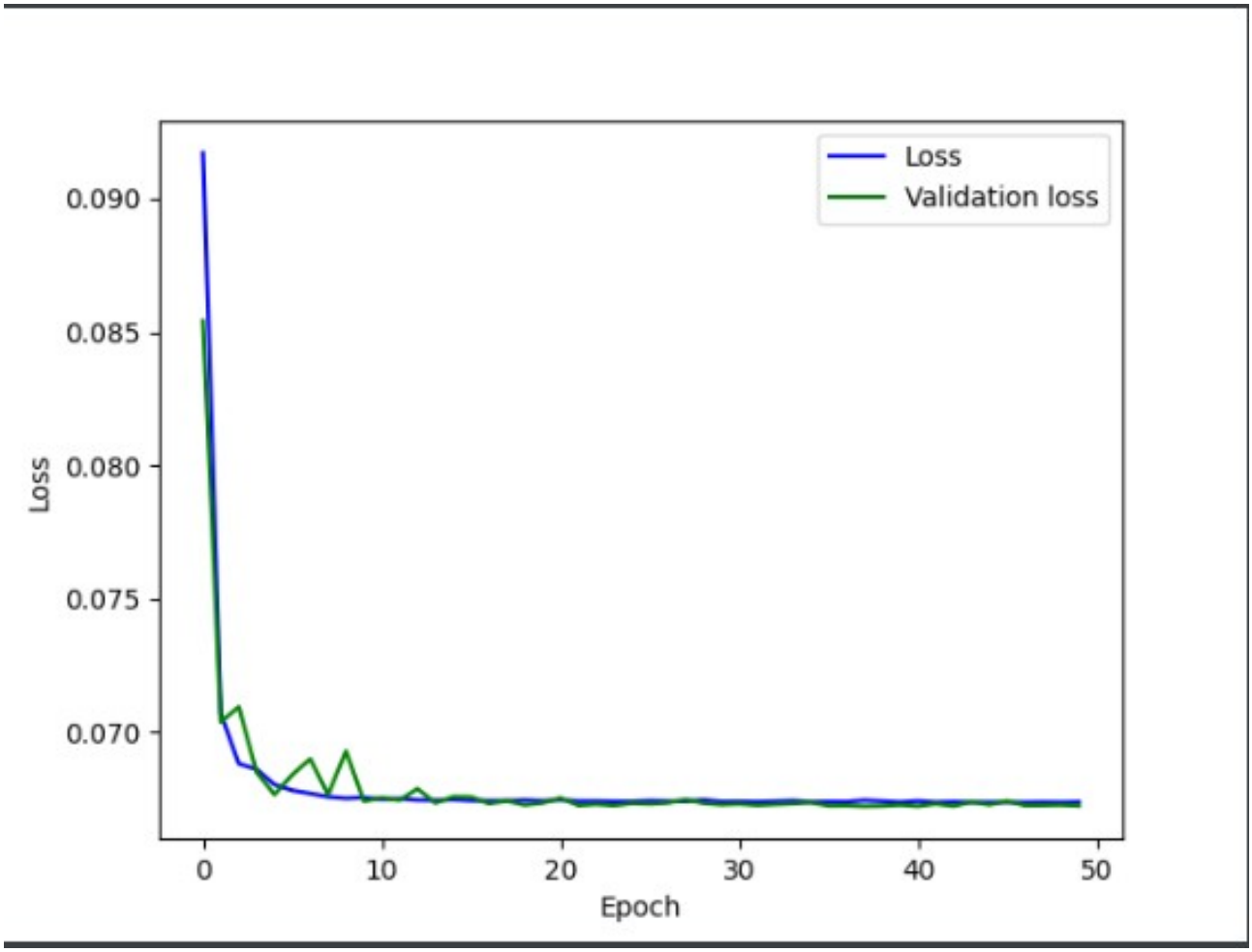


Οι παράμετροι που χρησιμοποιήθηκαν ήταν:

layer_num = 5
filter_size = 128
filter_num = 5
epochs = 50
batch_size = 128
latent_space = 10

Μειώθηκαν τα layers λόγω έλλειψης μνήμης ώστε να μπορέσουμε να δοκιμάσουμε μεγαλύτερο filter size. Παρατηρούμε ότι τα αποτελέσματα αλλοιώθηκαν με την μεγάλη αύξηση του filter_size.

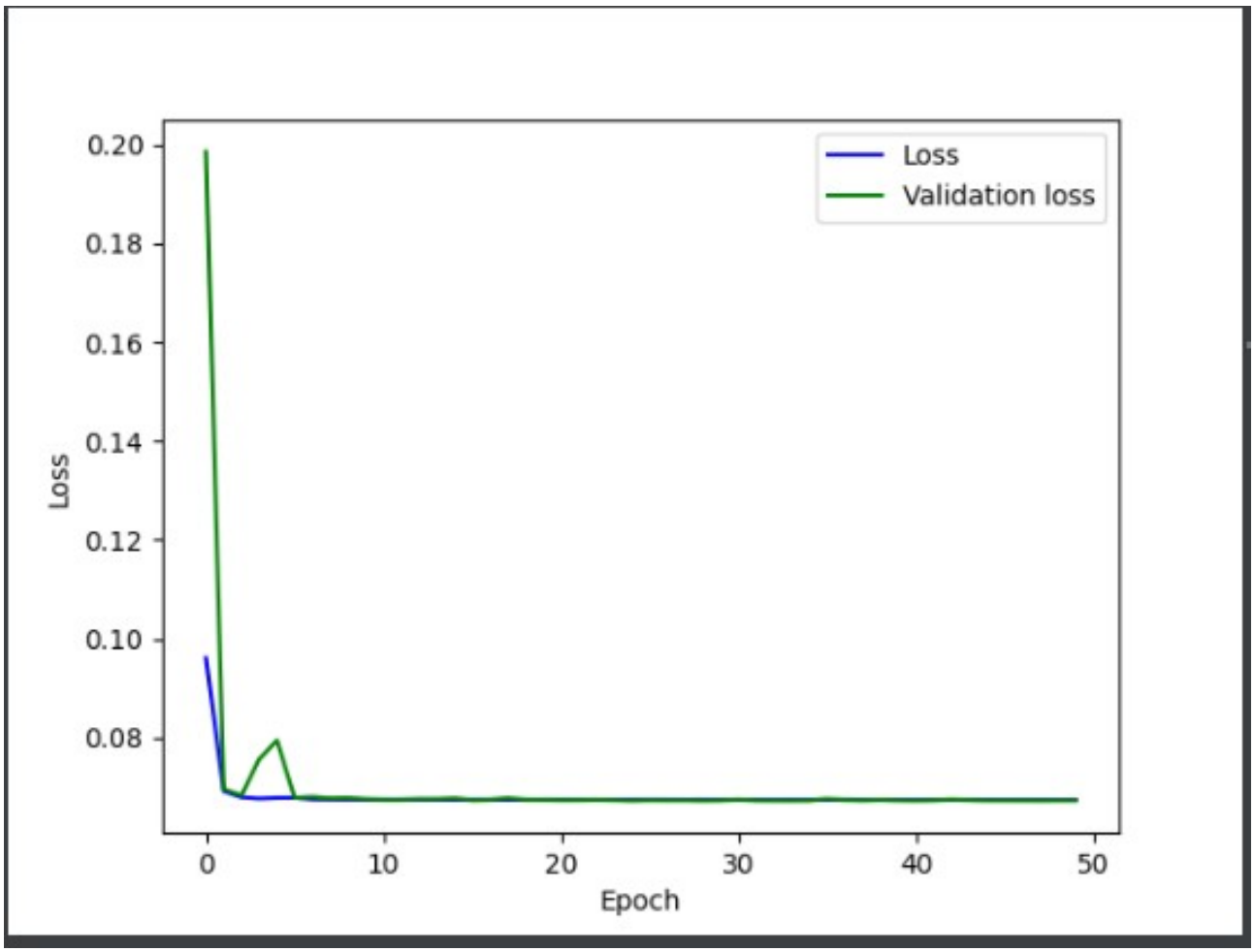
Στη συνέχεια θα προσθεθούν δυο παραδείγματα με αυξημένο filter size (=64) σε συνδυασμό αυξησης του αριθμού των φίλτρων.



Παρατηρούμε οτι το τελικό validation loss φτάνει σε πολύ χαμηλές τιμές, όμως στα πρώτα epochs υπάρχει overfitting.

Οι παράμετροι που χρησιμοποιήθηκαν ήταν:

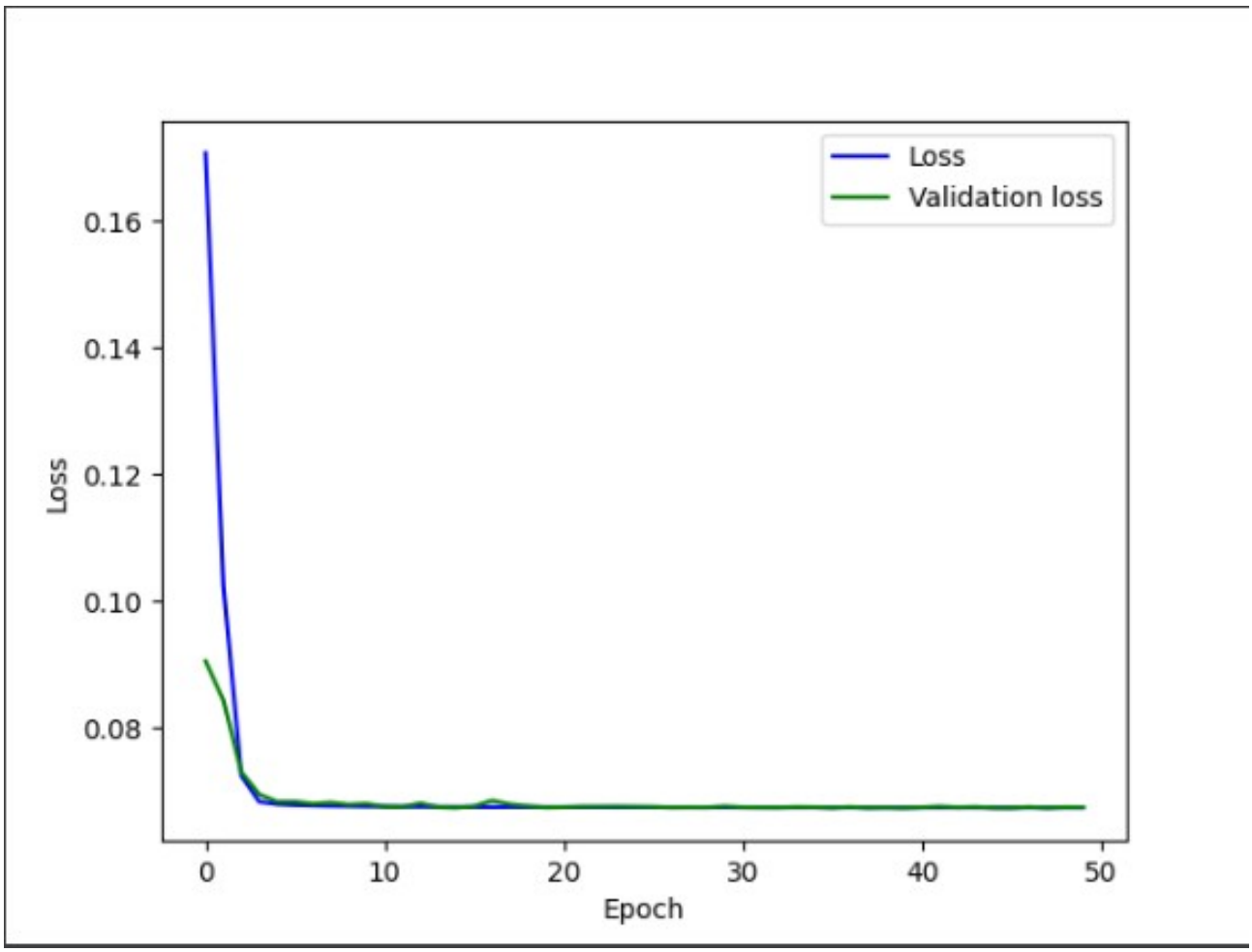
layer_num = 5
filter_size = 128
filter_num = 9
epochs = 50
batch_size = 128
latent_space = 10



Μειώνοντας το filter_size απο το προηγούμενο πείραμα παρατηρούμε ότι το overfitting μειώθηκε αλλά δεν εξαφανίσθηκε.

Οι παράμετροι που χρησιμοποιήθηκαν ήταν:

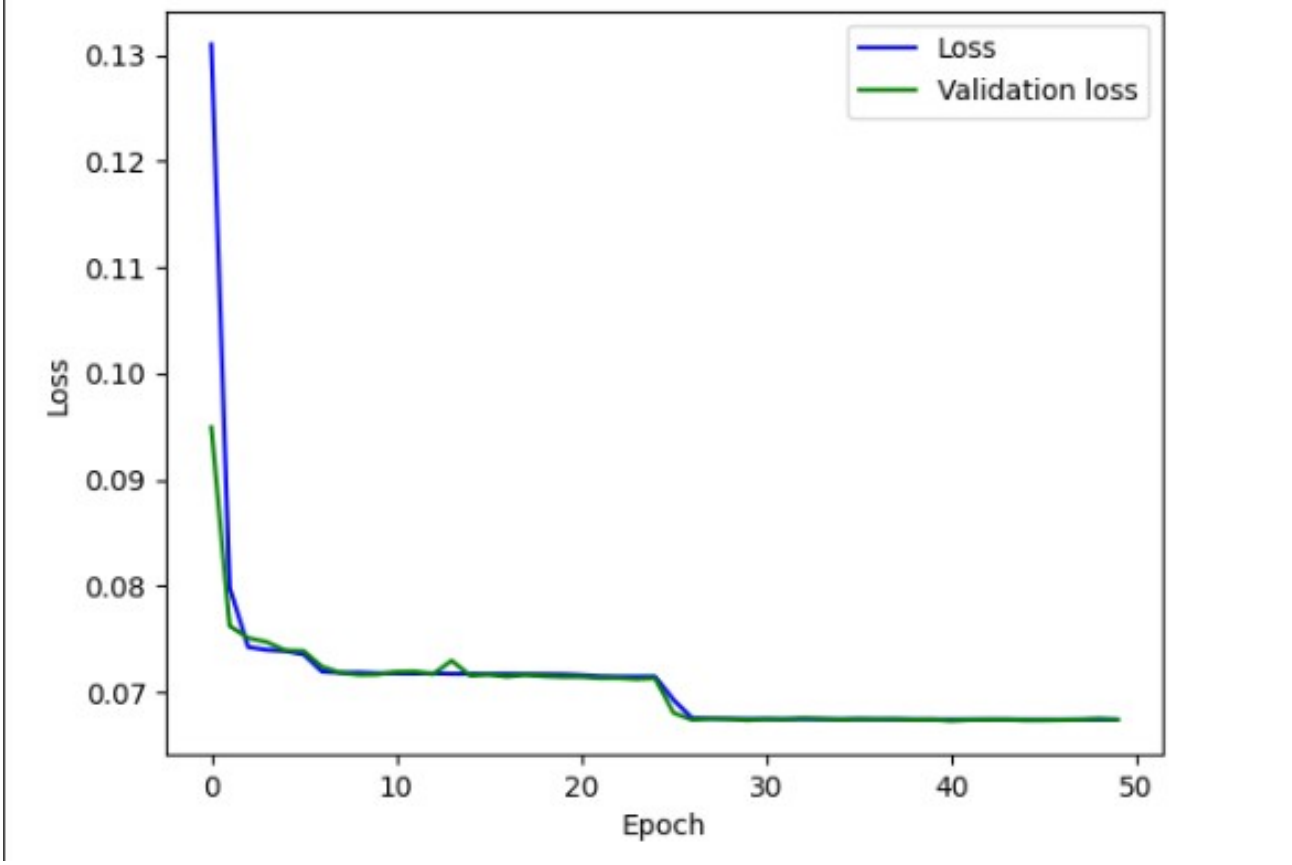
layer_num = 5
filter_size = 64
filter_num = 9
epochs = 50
batch_size = 128
latent_space = 10



Παρουσιάζουμε ένα ακόμα απο τα παραδείγματα με διάσταση 5.

Οι παράμετροι που χρησιμοποιήθηκαν ήταν:

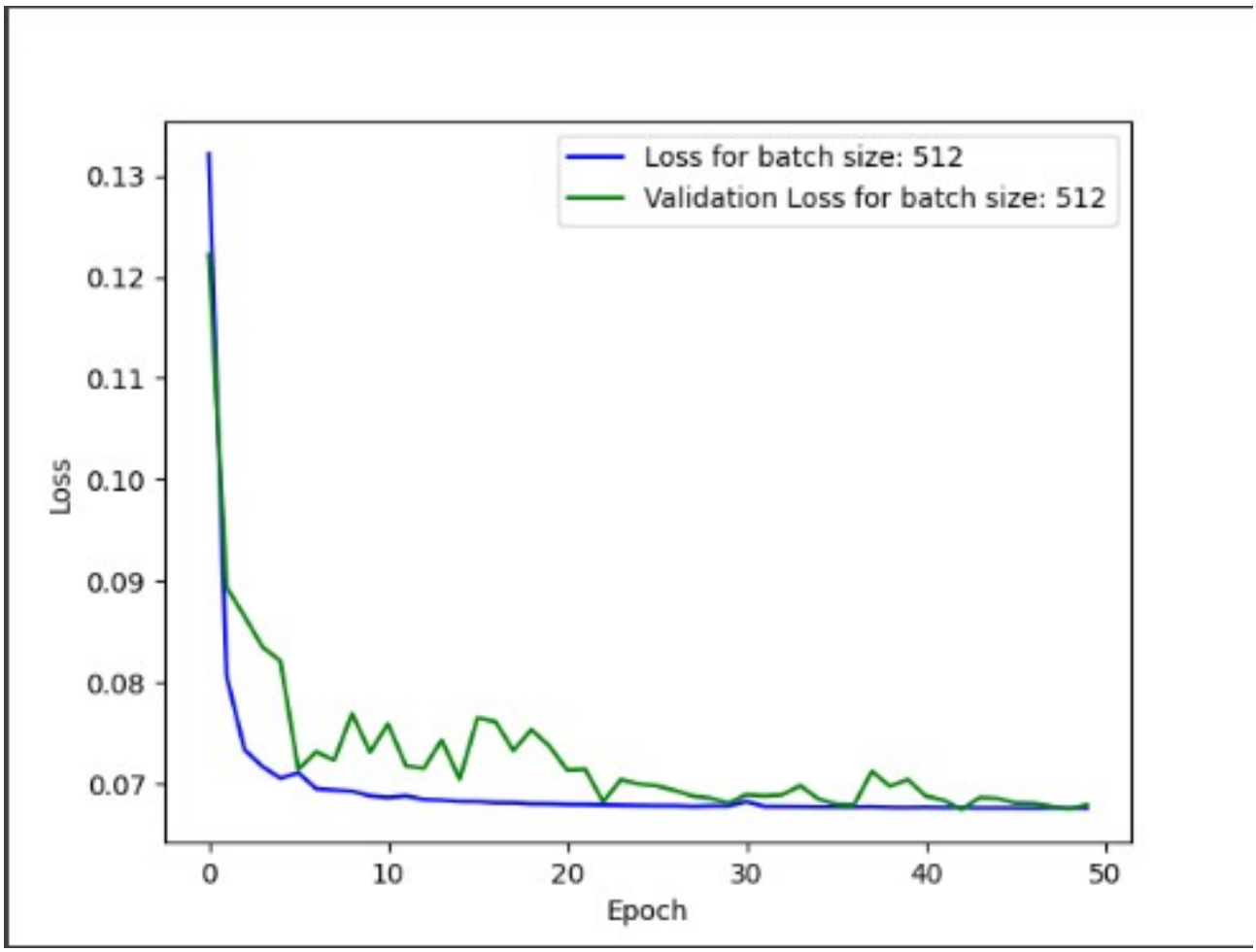
layer_num = 6
filter_size = 32
filter_num = 7
epochs = 50
batch_size = 256
latent_space = 5



Παραθέτουμε ένα παράδειγμα με τον πειραματισμό στον αριθμό των διαστάσεων με όχι τόσο καλά αποτελέσματα.

Οι παράμετροι που χρησιμοποιήθηκαν ήταν:

layer_num = 6
filter_size = 32
filter_num = 7
epochs = 50
batch_size = 128
latent_space = 100



Τέλος θα παρουσίασουμε ένα απο τα αρχικά αποτελέσματα του προγράμματος πριν αντιμετωπιθεί το πρόβλημα του Overfitting, το οποίο διορθώθηκε με την χρήση των Dropout layers.