Κ23γ : Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα

Χειμερινο Εξαμηνο 2020-21

2η Προγραμματιστική Εργασία

Γεωργακλή Έλενα (1115201500023)
Βασιλοπούλου Θωμαΐς (1115201500016)

Ερωτημα Α:

Α1. Δομη Παραδοτεου

Το παραδοτέο για την 2η προγραμματιστική εργασία αποτελείται από τα ακόλουθα αρχεία:

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

Α2. Λειτουργικοτητα Εργασιας

Το εκτελέσιμο τρέχει με την εντολή:

```
python autoencoder.py -d train-images.idx3-ubyte
```

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

```
def autoencoder(dataset, layers, maxFilters, x, y, convFiltSize, batchSize, epochs)
```

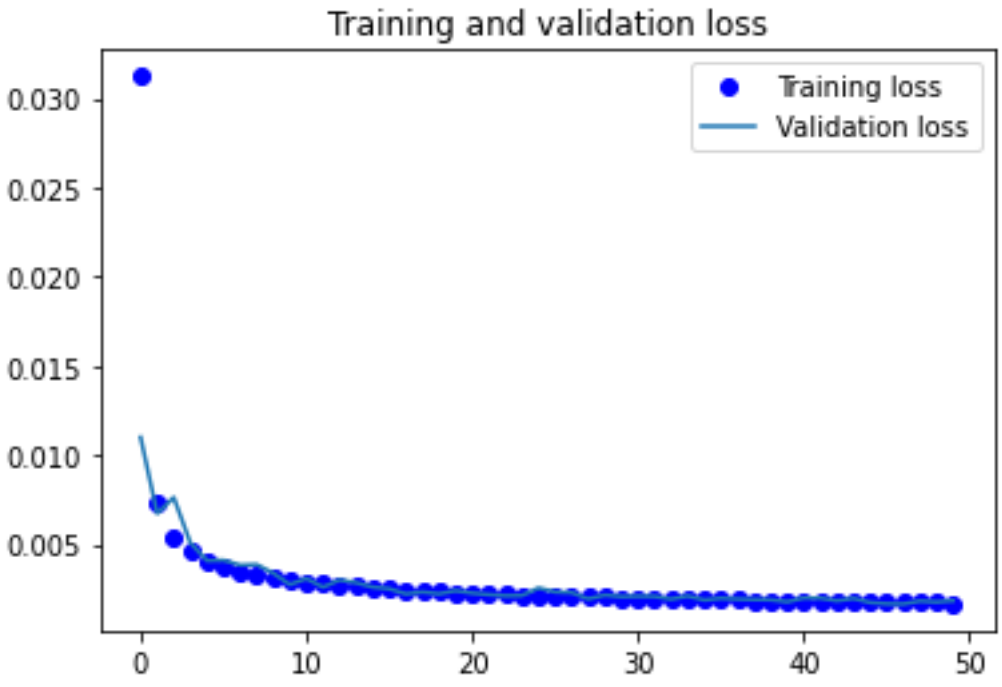
Πιο αναλυτικά, αντλούνται τα metadata από το αρχείο εισόδου (magic number, number of images, dx, dy), διαβάζονται οι εικόνες, μετατρέπει αυτά που διαβάστηκαν σε numpy array των κατάλληλων διαστάσεων και δημιουργεί το κατάλληλο μοντέλο με βάση τις δοθείσες διαστάσεις, με σκοπό ο autoencoder να αξιολογηθεί με μετρική απώλειας το mean_squared_error. Έπειτα δημιουργείται input κατάλληλων διαστάσεων και διαχωρίζεται το training set, ώστε το 80% να είναι για train και το 20% να είναι για validation. Στο μοντέλο του autoencoder ως ground truth ορίζεται το ίδιο το σύνολο των εικόνων και κανονικοποιούνται οι τιμές ώστε να είναι στο διάστημα [0.0, 1.0]. Στην συνέχεια εκπαιδεύεται το παραχθέν μοντέλο με βάσει τις δοθείσες από τον χρήστη υπερπαραμέτρους. Με την ολοκλήρωση της προηγούμενης διαδικασίας ερωτάται ο χρήστης για τον τρόπο με τον οποίο επιθυμεί να συνεχίσει, όπως περιγράφεται στην εκφώνηση.

Ο autoencoder χρησιμοποιεί τα ακόλουθα:

a. `encoder(inputImg, layers, maxFilters = 64, convFiltSize = (3, 3))`: Αποτελείται από layers, που δύναται να είναι είτε Conv2D, είτε MaxPooling2D ή BatchNormalization. Σκοπός του encoder είναι η εικόνα που θα χρησιμοποιηθεί στην συνέχεια από τον decoder να έχει ίδιες διαστάσεις με την αρχική. Για τον λόγο αυτό, το MaxPooling2D λαμβάνει χώρα μετά τα δύο πρώτα BatchNormalization και όχι αργότερα.
b. `decoder(conv, layers, maxFilters, convFiltSize)`: Αποτελείται από layers, που δύναται να είναι είτε Conv2D, είτε UpSampling2D ή BatchNormalization. Ο decoder είναι απόλυτα συμμετρικός με τον encoder, το UpSampling2D λαμβάνει χώρα μετά τα δύο τελευταία BatchNormalization και όχι νωρίτερα.

Α3. Αποτελεσματα Εκτελεσεων

Σημαντική παρατήρηση αποτελεί, ότι μοντέλο που παράγεται δεν κάνει overfit, διότι όπως φαίνεται και στο ακόλουθο διάγραμμα το validation loss φθίνει και, τελικά, κινείται ταυτόχρονα με το training loss.



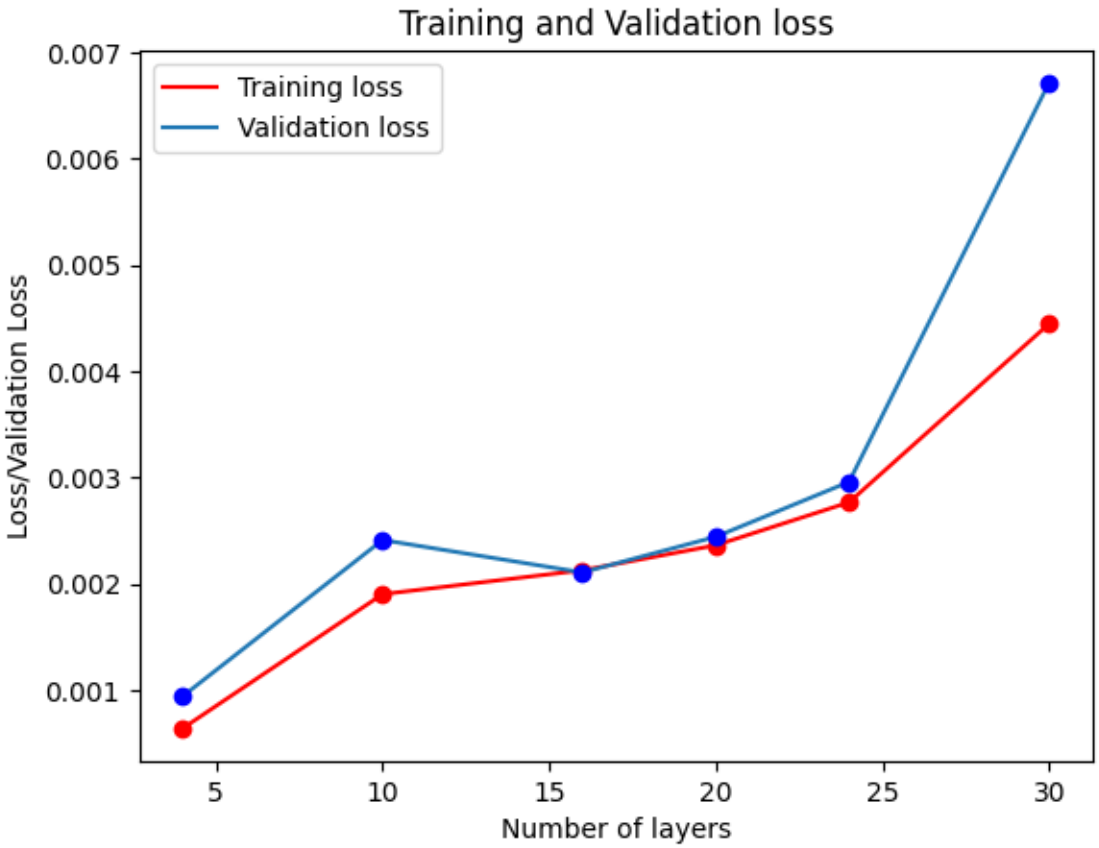
Εικόνα 1: Έλεγχος overfit για τον autoencoder

Στην συνέχεια παρατίθενται πειραματικές μετρήσεις με διαφοροποίηση μίας εκ των παραμέτρων ανά περίπτωση. Οι προκαθορισμένες τιμές που χρησιμοποιούνται στα πειράματα για τις μεταβλητές που δεν υφίστανται διαφοροποίηση είναι οι ακόλουθες:

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

No 1: ΠΛΗΘΟΣ ΣΥΝΕΛΙΚΤΙΚΩΝ ΣΤΡΩΜΑΤΩΝ

Είναι σημαντικό να αναφερθεί, ότι το πλήθος των συνελικτικών στρωμάτων μοιράζεται ισάξια στον encoder και τον decoder. Για παράδειγμα, αν δοθεί πλήθος layers = 10, τότε ο encoder θα χρησιμοποιήσει τα 5 από αυτά και ο decoder τα υπόλοιπα 5.

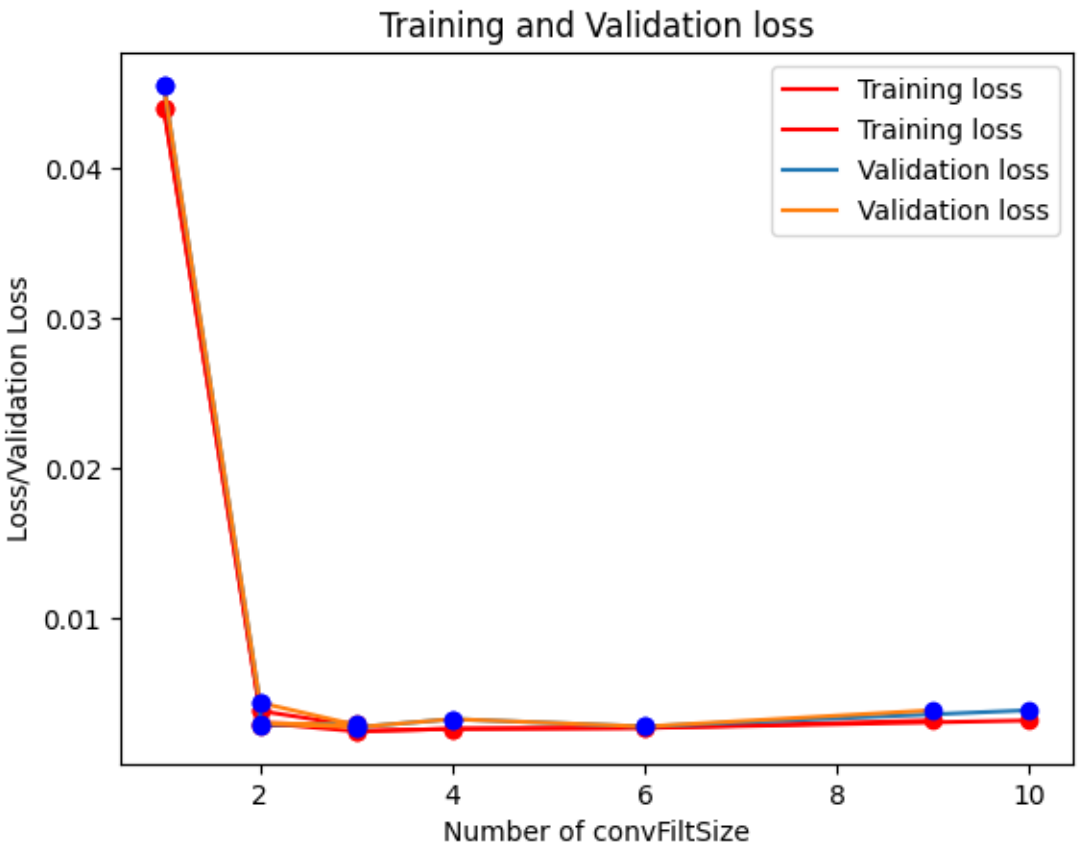


Εικόνα 2: Διαμόρφωση Σφάλματος για διάφορες τιμές συνελικτικών στρωμάτων

Από το παραπάνω διάγραμμα συμπεραίνεται, ότι ο autoencoder είναι πιο αποδοτικός, όταν το πλήθος των συνελικτικών στρωμάτων κυμαίνεται από 15 έως 25, δεδομένου ότι στο διάστημα αυτό το σφάλμα ελαχιστοποιείται.

Μία επιπλέον βασική παραδοχή είναι, ότι τα layers που δίνει ο χρήστης πρέπει να είναι τουλάχιστον 12, ώστε να μοιραστούν 6 από αυτά στον encoder, για να χρησιμοποιηθούν για Conv2D, BatchNormalization και MaxPooling2D, και τα άλλα 6 στον decoder, για να χρησιμοποιηθούν για Conv2D, BatchNormalization και UpSampling2D.

No 2: ΜΕΓΕΘΟΣ ΣΥΝΕΛΙΚΤΙΚΩΝ ΦΙΛΤΡΩΝ



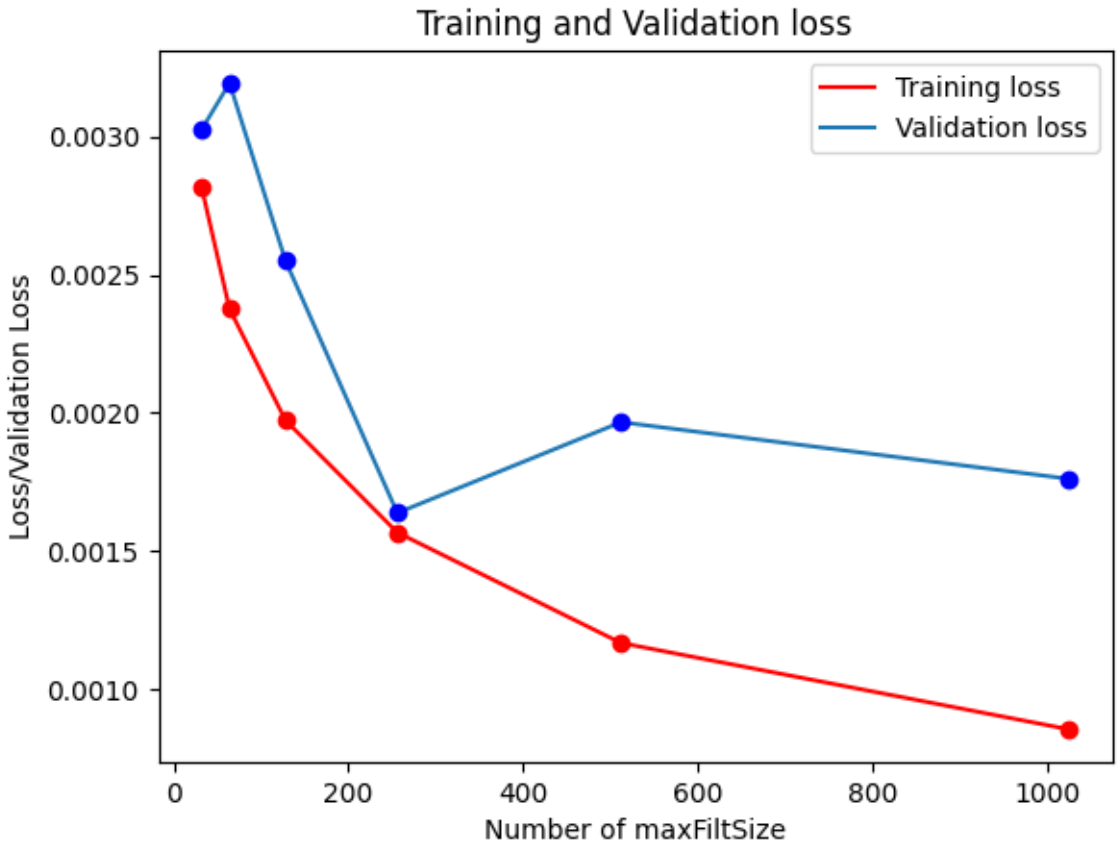
Εικόνα 3: Διαμόρφωση Σφάλματος για διάφορα μεγέθη συνελικτικών φίλτρων

Από το παραπάνω διάγραμμα συμπεραίνεται, ότι ο autoencoder είναι λιγότερο αποδοτικός, όταν δίνεται φίλτρο, στο οποίο τουλάχιστον μία από τις δύο διαστάσεις είναι 1.

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

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

No 3: ΠΛΗΘΟΣ ΦΙΛΤΡΩΝ ΑΝΑ ΣΤΡΩΜΑ



Εικόνα 4: Διαμόρφωση Σφάλματος για διάφορες τιμές συνελικτικών φίλτρων ανά στρώμα

Είναι σημαντικό να αναφερθεί, ότι ο autoencoder λειτουργεί αποδοτικότερα, όταν το πλήθος των φίλτρων του τρέχοντος επιπέδου έχει μία γραμμική σχέση με τα αντίστοιχα πλήθη του προηγούμενου και του επόμενού του. Όπως επισημαίνεται και στις διαφάνειες του μαθήματος, κάθε στρώμα του encoder περιέχει διπλάσιο πλήθος φίλτρων από το προηγούμενο, και αντίστοιχα στρώμα του decoder περιέχει υποδιπλάσιο πλήθος φίλτρων από το προηγούμενο.

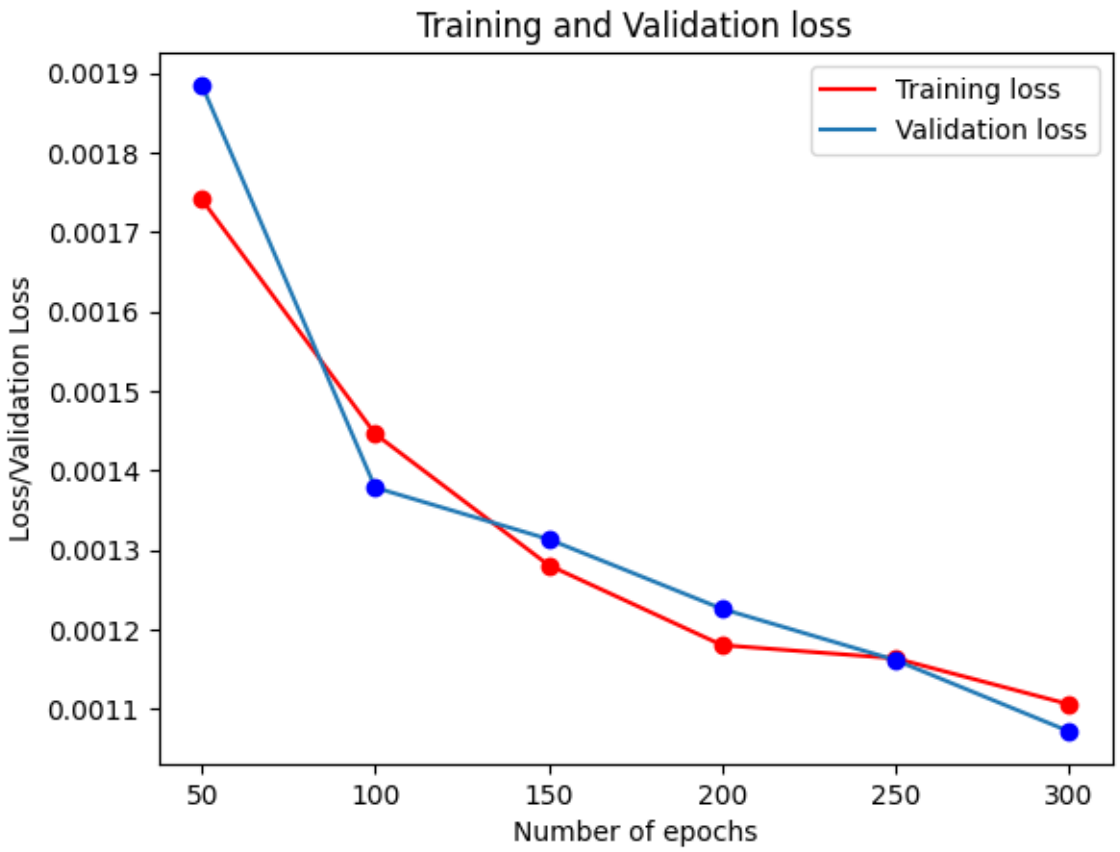
Έτσι, ο χρήστης καλείται να επιλέξει το μέγιστο πλήθος φίλτρων, που θα χρησιμοποιηθούν, και η ανάθεση φίλτρων στο πρώτο στρώμα προσδιορίζεται από την εξίσωση:

```
numOfFilters = (int)(maxFilters/(pow(2, ((layers-6)/2) + 1)))
```

Για παράδειγμα, αν δοθεί από τον χρήστη maxFilters = 64, τότε στο πρώτο στρώμα του encoder θα εφαρμοστούν 8.

Από το παραπάνω διάγραμμα συμπεραίνεται, ότι ο autoencoder γενικά συμπεριφέρεται καλύτερα για μεγαλύτερα φίλτρα, κάτι το οποίο, όμως, στην πράξη δεν προτιμάται.

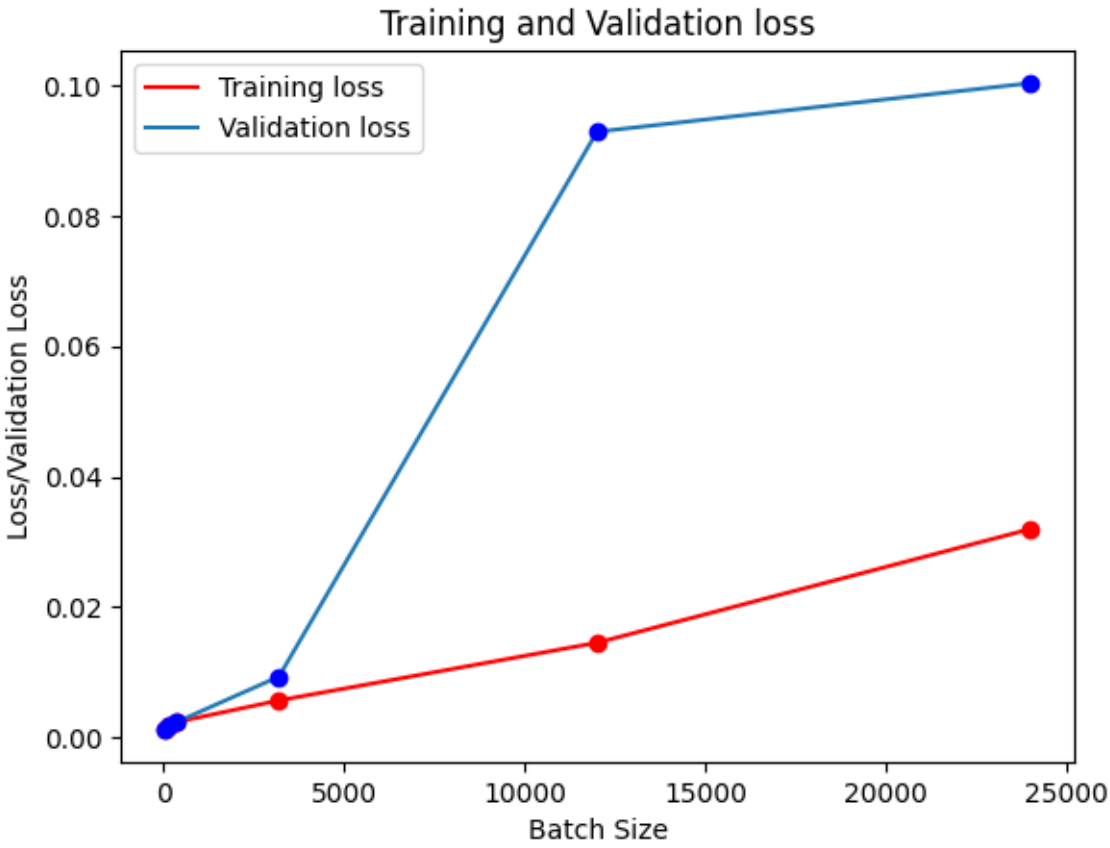
No 4: ΠΛΗΘΟΣ ΕΠΟΧΩΝ ΕΚΠΑΙΔΕΥΣΗΣ



Εικόνα 5: Διαμόρφωση Σφάλματος για διάφορες τιμές εποχών εκπαίδευσης ανά εκτέλεση

Από το παραπάνω διάγραμμα συμπεραίνεται, ότι όσο αυξάνεται το πλήθος των εποχών εκπαίδευσης, το σφάλμα μειώνεται, αλλά ο χρόνος εκτέλεσης του προγράμματος αυξάνεται δραματικά.

No 5: ΜΕΓΕΘΟΣ ΔΕΣΜΗΣ



Εικόνα 6: Διαμόρφωση Σφάλματος για διάφορες τιμές μεγέθους δέσμης ανά εκτέλεση

Από το παραπάνω διάγραμμα συμπεραίνεται, ότι όσο μειώνεται το πλήθος των δεσμών, το σφάλμα ελαχιστοποιείται και ο χρόνος εκτέλεσης του προγράμματος μειώνεται σημαντικά.

Ερωτημα Β:

Β1. Δομη Παραδοτεου

Το παραδοτέο για την 2η προγραμματιστική εργασία αποτελείται από τα ακόλουθα αρχεία:

- classifier.py: το εκτελέσιμο που υλοποιεί το ζητούμενο Β
- graphs: φάκελος που περιέχει τα γραφικά αποτελέσματα της εκτέλεσης του προηγούμενου αρχείου

Β2. Λειτουργικοτητα Εργασιας

Το εκτελέσιμο τρέχει με την εντολή:

```
python classifier.py -d .\train-images.idx3-ubyte -dl .\train-labels.idx1-ubyte -t
   .\t10k-images.idx3-ubyte -tl .\t10k-labels.idx1-ubyte -model .\autoencoder.h5
```

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

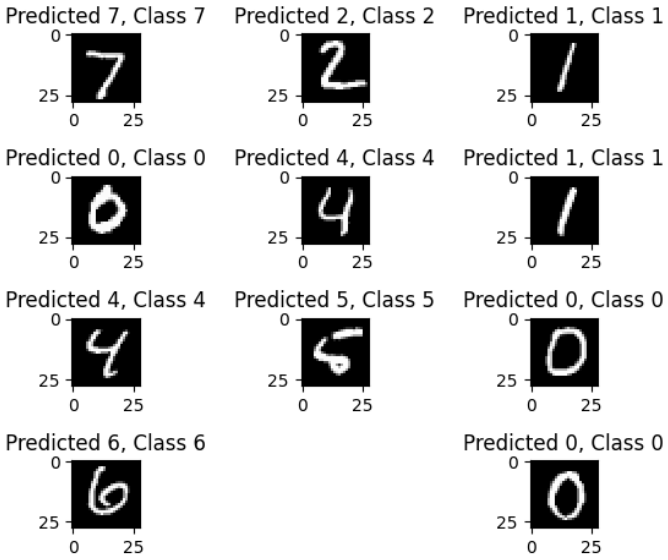
```
def classification(trainSet, trainLabelsSet, testSet, testLabelsSet, autoencoderPath,
 layers, fcNodes, batchSize, epochs)
```

Πιο αναλυτικά, αντλούνται τα metadata από το αρχείο εισόδου (magic number, number of images, dx, dy), διαβάζονται οι εικόνες και τα labels και μετατρέπει αυτά που διαβάστηκαν σε numpy array των κατάλληλων διαστάσεων. Όσον αφορά τα labels, η τιμή του καθενός μετατρέπεται σε ένα vector με τιμή 1 στη αντίστοιχη θέση και 0 στις υπόλοιπες. Έπειτα δημιουργείται input κατάλληλων διαστάσεων και διαχωρίζεται το training set, ώστε το 80% να είναι για train και το 20% να είναι για validation. Στην περίπτωση του classifier ως ground truth ορίζεται το σύνολο των labels. Στην συνέχεια, φορτώνεται το μοντέλο του autoencoder που προσδιορίζει ο χρήστης από το command line, δημιουργείται ένα καινούργιο μοντέλο, που αντί για decoder χρησιμοποιεί ένα πλήθος από fully connected layers. Στο μοντέλο αυτό αντιγράφονται τα βάρη των layers του encoder του autoencoder. Σε πρώτη φάση εκπαιδεύεται το παραχθέν μοντέλο μόνο ως προς τα fully connected layers. Σε επόμενη φάση η εκπαίδευση γίνεται για το συνολικό παραχθέν μοντέλο. Στην συνέχεια το μοντέλο αξιολογείται και λαμβάνουν χώρα προβλέψεις πάνω στο test set και ερωτάται ο χρήστης για τον τρόπο με τον οποίο επιθυμεί να συνεχίσει, όπως περιγράφεται στην εκφώνηση.

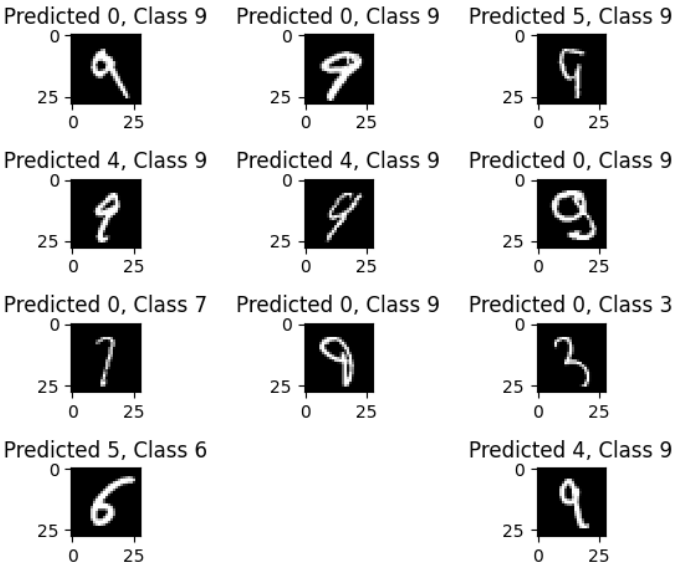
Ο classifier χρησιμοποιεί τα ακόλουθα:

a. `encoder(inputImg, layers, maxFilters = 64, convFiltSize = (3, 3))`: Ίδιος με αυτόν του autoencoder.
b. `fullyConnected(enco, fcNodes)`: Αρχικά κάνει flatten το αποτέλεσμα που παράγει ο encoder, περνάει το vector από όσα fully connected layers έχουν οριστεί από το χρήστη και εφαρμόζει το activation function. Λόγω παρατήρησης overfitting στο μοντέλο εφαρμόζεται ένα dropout της τάξης του 0.2, προκειμένου να αποφευχθεί η προηγούμενη αστοχία.

Αν ο χρήστης επιθυμεί να λάβουν χώρα και προβλέψεις στο test set, τότε το μοντέλο πρέπει να αξιολογηθεί και ως προς την ορθότητα αυτών. Αρχικά τυπώνεται το πλήθος των σωστών και των λανθασμένων προβλέψεων και στην συνέχεια εμφανίζονται κάποιες εικόνες των ψηφίων του συνόλου ελέγχου και η κατηγορία στην οποία εντάχθηκαν.



Εικόνα 7: Σώστες προβλέψεις για το δοθέν test set



Εικόνα 8: Λανθασμένες προβλέψεις για το δοθέν test set

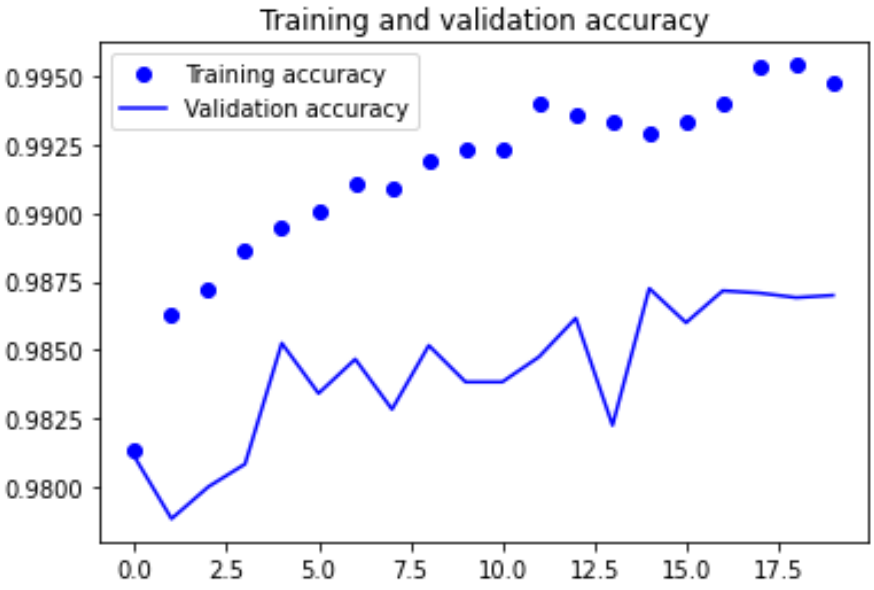
Όπως καθίσταται σαφές από τα παραπάνω διαγράμματα, το νευρωνικό δίκτυο τείνει να συγχέει το ψηφίο 9 κατά κύριο λόγο με το 0 και το 4.

Τελικά εκτυπώνεται το classification report με βάση τους δείκτες αξιολόγησης:

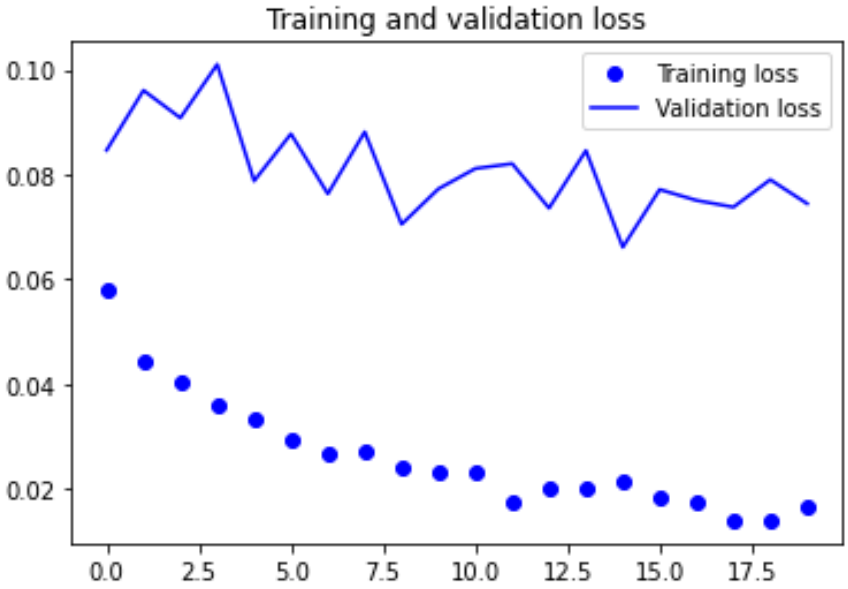
- Ορθότητα (Accuracy)
- Σφάλμα (Loss)
- Ακρίβεια (Precision)
- Ανάκληση (Recall)
- F Score

Β3. Αποτελεσματα Εκτελεσεων

Σημαντική παρατήρηση αποτελεί, ότι μοντέλο που παράγεται κάνει overfit, διότι, όπως φαίνεται και στο ακόλουθο διάγραμμα, τόσο η ακρίβεια, όσο και το σφάλμα μεταξύ validation και training set απέχει (δεν υπάρχουν σημεία που οι δύο τιμές να ταυτίζονται). Έτσι οδηγούμαστε στη χρήση dropout στο fully connected layer, με σκοπό το μοντέλο να μην κάνει τελικά overfit.



Εικόνα 9: Έλεγχος overfit για τον classifier

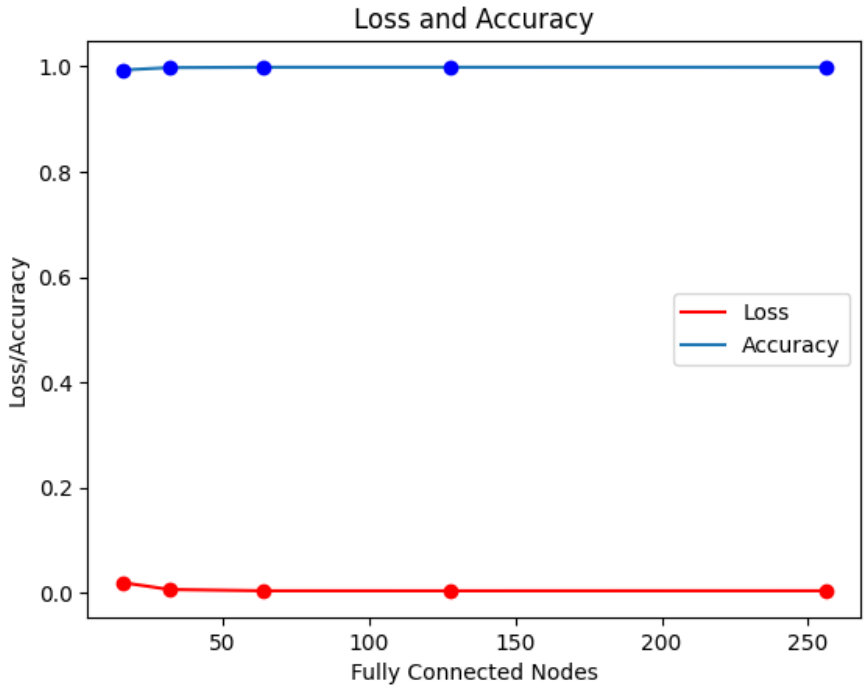


Εικόνα 10: Έλεγχος overfit για τον classifier

Στην συνέχεια παρατίθενται πειραματικές μετρήσεις με διαφοροποίηση μίας εκ των παραμέτρων ανά περίπτωση. Οι προκαθορισμένες τιμές που χρησιμοποιούνται στα πειράματα για τις μεταβλητές που δεν υφίστανται διαφοροποίηση είναι οι ακόλουθες:

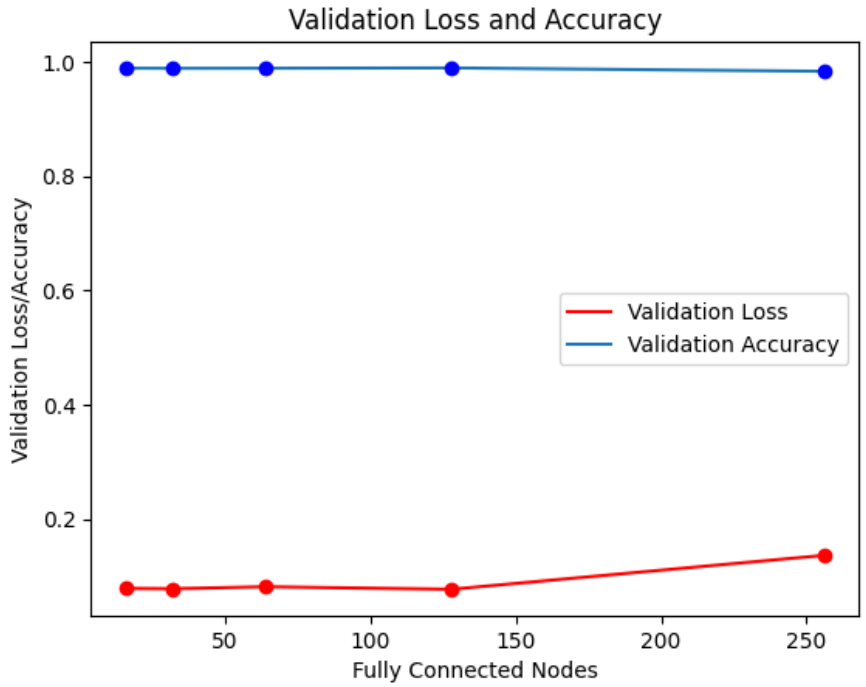
- *fcnodes = 32*
- *epochs = 50*
- *batchSize = 128*

No 1: ΠΛΗΘΟΣ ΠΛΗΡΩΣ ΣΥΝΔΕΔΕΜΕΝΩΝ ΚΟΜΒΩΝ



Εικόνα 11: Διαμόρφωση Ακρίβειας/Σφάλματος για διαφορετικά πλήθη πλήρως συνδεδεμένων κόμβων ανά εκτέλεση για το training set

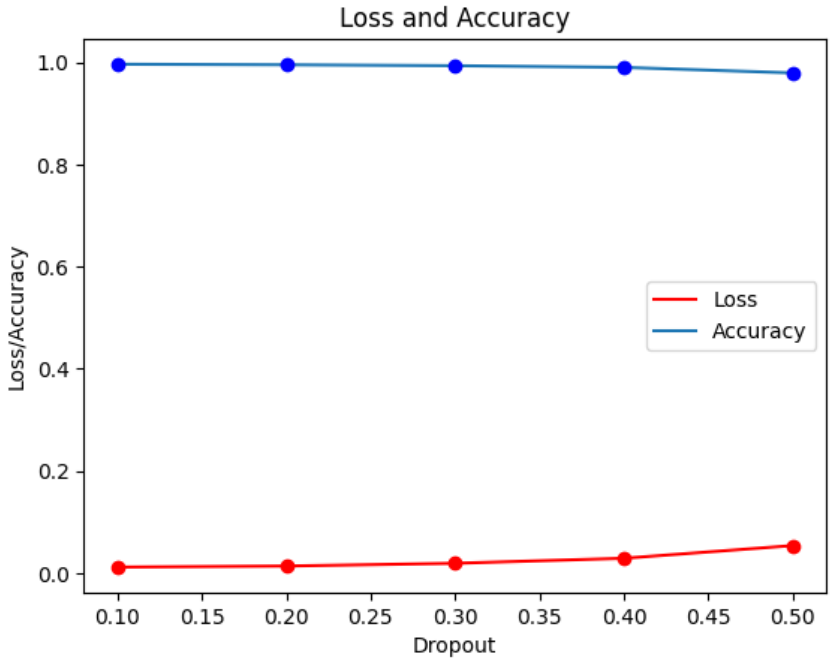
Από το παραπάνω διάγραμμα συμπεραίνεται, ότι το πλήθος των fully connected layers δεν επηρεάζει την απόδοση του classifier, ούτε ως προς το σφάλμα, ούτε ως προς την ακρίβεια. Παρόλα αυτά παρατηρήθηκε, ότι επιδρά στον χρόνο εκτέλεσης του προγράμματος.



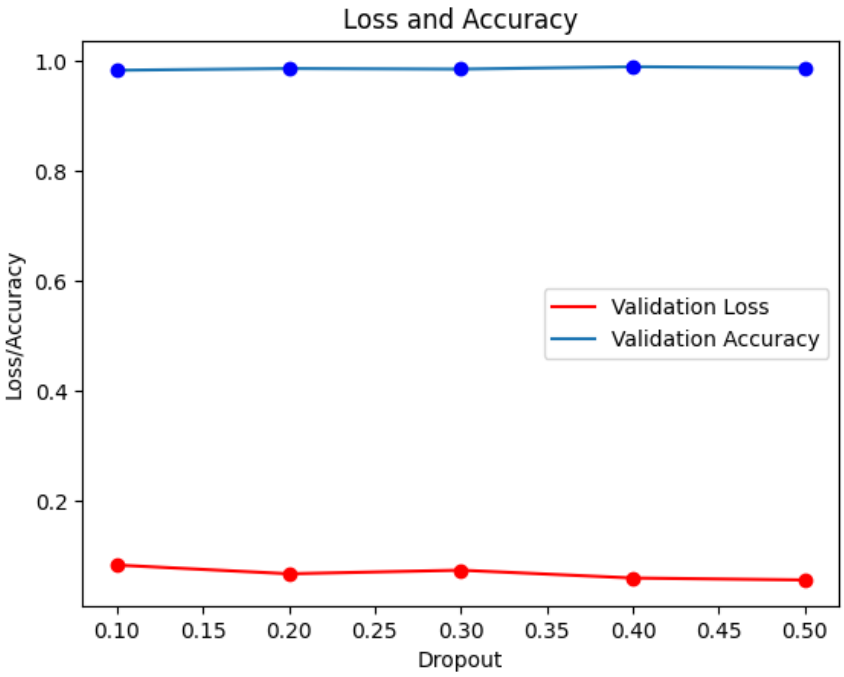
Εικόνα 12: Διαμόρφωση Ακρίβειας/Σφάλματος για διαφορετικά πλήθη πλήρως συνδεδεμένων κόμβων ανά εκτέλεση για το validation set

Από το παραπάνω διάγραμμα συμπεραίνεται, ότι το πλήθος των fully connected layers δεν επηρεάζει την απόδοση του classifier, ως προς την ακρίβεια, όμως το σφάλμα αυξάνεται ελαφρώς. Παρόλα αυτά παρατηρήθηκε, ότι επιδρά στον χρόνο εκτέλεσης του προγράμματος.

Επιπλέον, αξίζει να σχολιαστεί η συμπεριφορά του cliassiefier για διαφοροποίηση του dropout και του activation function, που χρησιμοποιείται για τον υπολογισμό των fully connected layers. Για τον λόγο αυτό, έχουν παραχθεί τα ακόλουθα διαγράμματα:

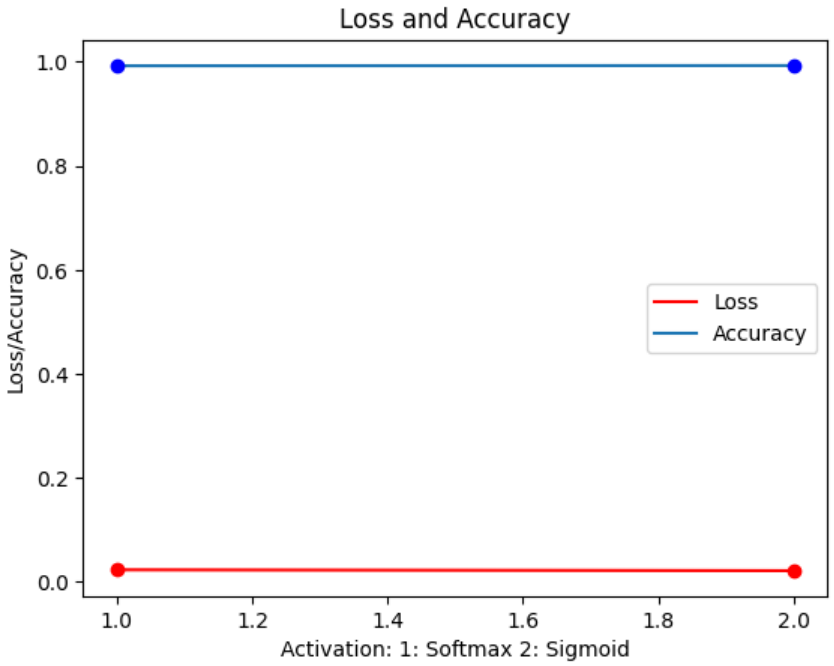


Εικόνα 13: Διαμόρφωση Ακρίβειας/Σφάλματος για διαφορετικές τιμές dropout ανά εκτέλεση για το training set

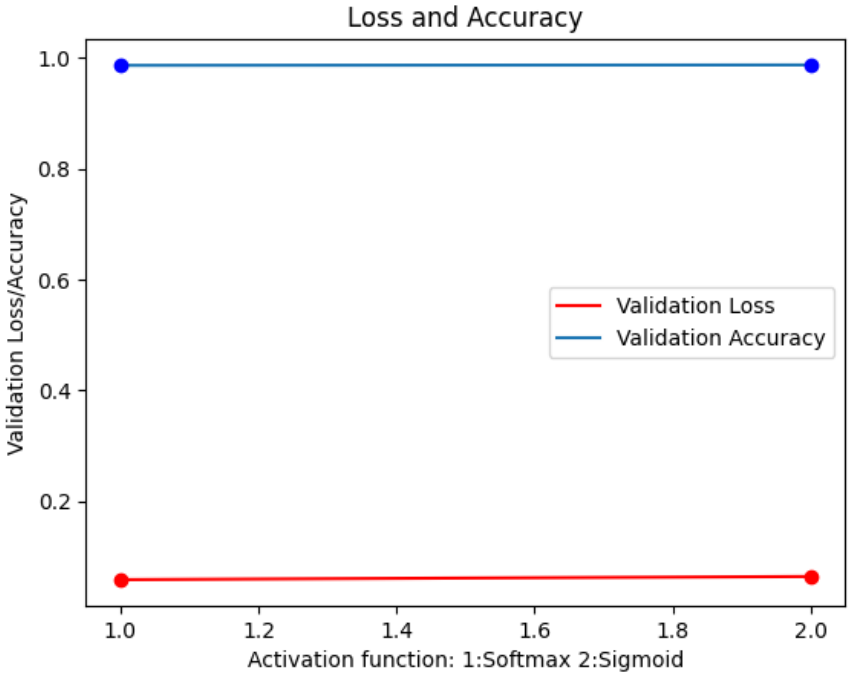


Εικόνα 14: Διαμόρφωση Ακρίβειας/Σφάλματος για διαφορετικές τιμές dropout ανά εκτέλεση για το validation set

Όπως φαίνεται από τα παραπάνω διαγράμματα, η αλλαγή στο dropout, δεν επηρεάζει την απόδοση του classifier, ούτε τον χρόνο εκτέλεσής του. Το dropout λαμβάνει τιμές στο διάστημα [0.1, 0.5]. Τιμές μεγαλύτερες από 0.6 οδηγούν στην απώλεια μεγάλου ποσοστού των δεδομένων και δεν κρίνεται σκόπιμο να ελεγχθούν.



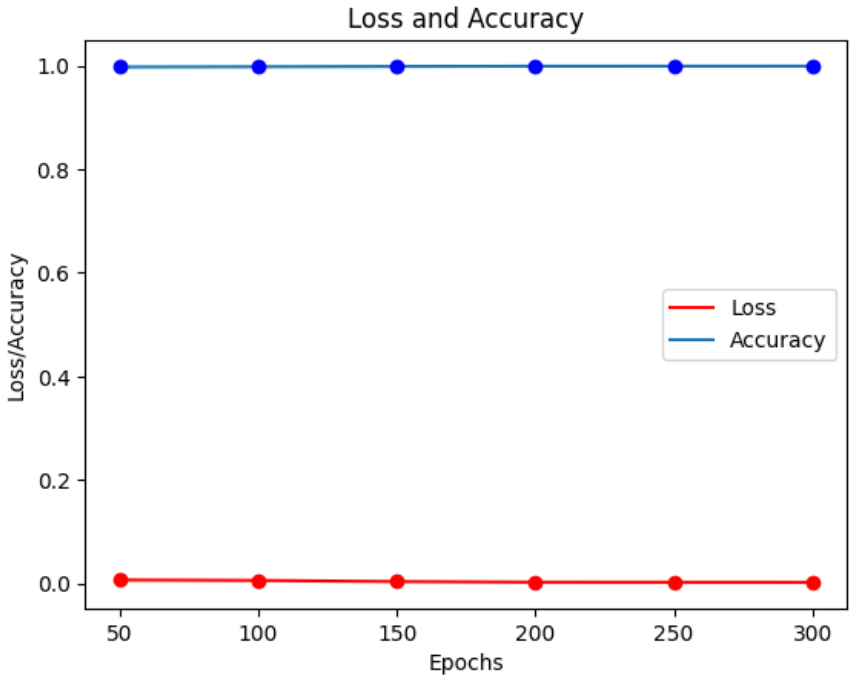
Εικόνα 15: Διαμόρφωση Ακρίβειας/Σφάλματος για διαφορετικό activation function ανά εκτέλεση για το validation set



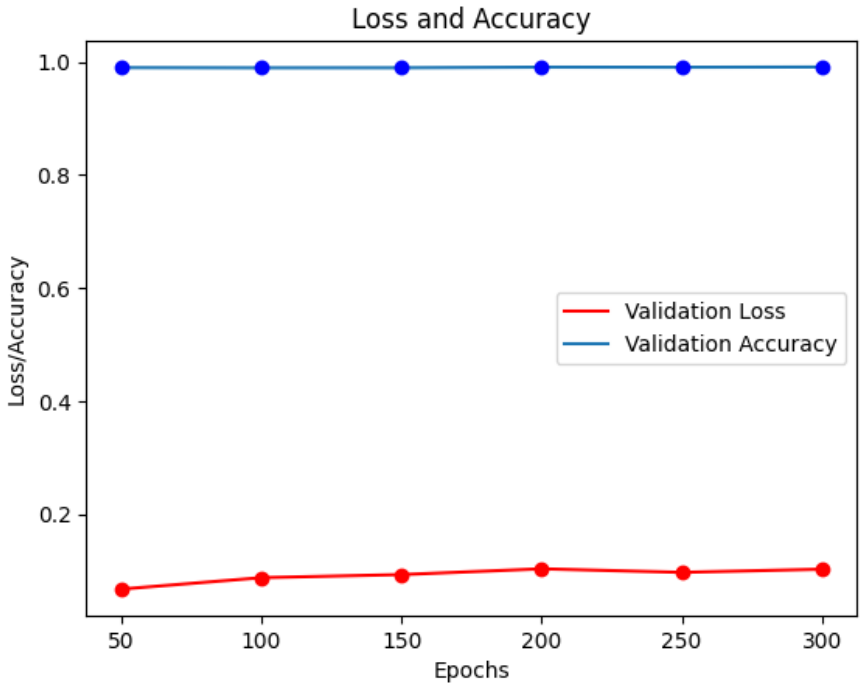
Εικόνα 16: Διαμόρφωση Ακρίβειας/Σφάλματος για διαφορετικό activation function ανά εκτέλεση για το validation set

Όπως φαίνεται από τα παραπάνω διαγράμματα, η αλλαγή στο activation function, δεν επηρεάζει την συνολική απόδοση του classifier. Παρόλα αυτά, στα τελικά αποτελέσματα για σωστές προβλέψεις η sigmoid αποδίδει πολύ χειρότερα από τη softmax. Με ενδεικτικές σωστές προβλέψεις για τη sigmoid της τάξης του 35%, έναντι του 95% της softmax.

No 2: ΠΛΗΘΟΣ ΕΠΟΧΩΝ ΕΚΠΑΙΔΕΥΣΗΣ



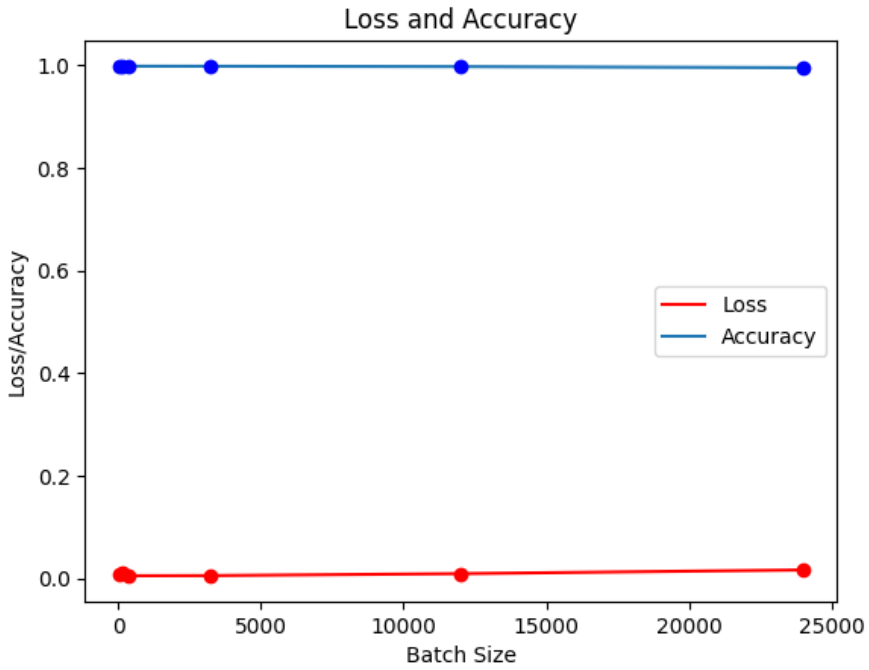
Εικόνα 17: Διαμόρφωση Σφάλματος για διάφορες τιμές εποχών εκπαίδευσης ανά εκτέλεση για το training set



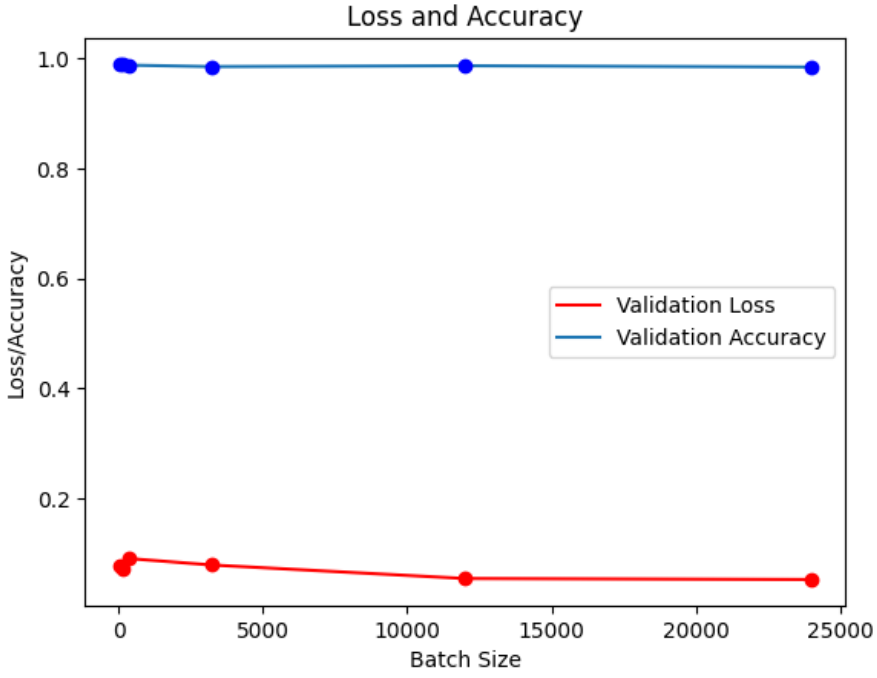
Εικόνα 18: Διαμόρφωση Σφάλματος για διάφορες τιμές εποχών εκπαίδευσης ανά εκτέλεση για το validation set

Όπως φαίνεται από τα παραπάνω διαγράμματα, η αύξηση του πλήθους των εποχών εκπαίδευσης, δεν επηρεάζει την απόδοση του classifier, ούτε ως προς το σφάλμα ούτε ως προς την ακρίβεια, σε αντίθεση με τα αντίστοιχα πειράματα που αφορούν τον autoencoder. Παρόλα αυτά και στην περίπτωση του classifier, όσο αυξάνεται το πλήθος των εποχών εκπαίδευσης, τόσο καθυστερεί η εκτέλεση του προγράμματος.

No 3: ΜΕΓΕΘΟΣ ΔΕΣΜΗΣ



Εικόνα 19: Διαμόρφωση Σφάλματος για διάφορες τιμές μεγέθους δέσμης ανά εκτέλεση για το training set



Εικόνα 20: Διαμόρφωση Σφάλματος για διάφορες τιμές μεγέθους δέσμης ανά εκτέλεση για το validation set

Όπως φαίνεται από τα παραπάνω διαγράμματα, η αύξηση του μεγέθους, δεν επηρεάζει την απόδοση του classifier, ούτε ως προς το σφάλμα ούτε ως προς την ακρίβεια, σε αντίθεση με τα αντίστοιχα πειράματα που αφορούν τον autoencoder. Παρόλα αυτά και στην περίπτωση του classifier, όσο αυξάνεται το μέγεθος δέσμης, τόσο επιταχύνεται η εκτέλεση του προγράμματος.

Βιβλιογραφια/ Πηγες:

Η ανάπτυξη των αλγορίθμων για την υλοποίηση των ερωτημάτων της εκφώνησης βασίστηκε κυρίως στις διαφάνειες του μαθήματος. Συμπληρωματικό υλικό βρέθηκε στις ακόλουθες πηγές:

(1) https://www.datacamp.com/community/tutorials/autoencoder-keras-tutorial
(2) https://blog.keras.io/building-autoencoders-in-keras.html
(3) https://www.tensorflow.org/guide/keras/save_and_serialize
(4) https://www.datacamp.com/community/tutorials/autoencoder-classifier-python
(5) https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python
(6) https://towardsdatascience.com/dont-overfit-how-to-prevent-overfitting-in-your-deep-learning-models-63274e552323