Technical Report DL assignment

1.Dataset

Το dataset αποτελείται από 4000 εικόνες και ένα csv με διάφορα annotations απο διαφορετικούς annotators. Διαλέγουμε έναν annotator και μία ερώτηση που αφορά τί συναίσθημα προκαλεί η κάθε φωτογραφία. Το εύρος είναι από 0 έως 9 με το 1 να σημαίνει πολύ αρνητικό το 5 neutral και το 9 πολύ θετικό.Οπότε θα πρέπει να χωρίσουμε τις φωτογραφίες σε διαφορετικούς φακέλους με βάση το csv. Το script που το κάνει αυτό είναι το **script1.py** και έτρεξε αυτόνομα στο laptop και δεν είναι διαθέσιμο στο τελικό .ipynb.

Στην συνέχεια θα χωρίσουμε το dataset σε train/validation/test. Για το validation και το test κρατάμε 50 samples ανά κλάση και για το train όλα τα υπόλοιπα. Να σημειωθεί οτι πρόκειται για imbalanced classification problem καθώς δεν έχουμε τον ίδιο αριθμό δειγμάτων ανα κλάση. Το train set μετά από την δημιουργία των validation/test έχει αριθμό samples από 143 έως 1040 ανα κλάση. Το script που έτρεξε είναι το **script2.py.** Το τρέξαμε δύο φορές μία για να δημιουργήσω το validation και άλλη μία για να δημιουργήσω το test. Οι τρεις φάκελοι που προέκυψαν σώθηκαν σε zip για να ανέβουν στο drive ώστε να είναι διαθέσιμοι στο colab. Αυτό που κάνει το script2.py είναι να δημιοργεί 50 αριθμούς στο εύρος τιμών του κάθε φακέλου από τους 9 που προέκυψαν στο προηγούμενο βήμα και να αντιγράφει την φωτογραφία σε νέο φάκελο και να την διαγράφει απο τον παλιό. Το script αυτό έτρεξε αυτόνομα στο laptop και δεν είναι διαθέσιμο στο τελικό .ipynb.

Στην συνέχεια δοκιμάσαμε να τρέξουμε ένα μοντέλο για να δούμε ότι τρέχει ο κώδικας αλλά έδινε error. Διαπιστώσαμε ότι κάποιες εικόνες ήτανε corrupted οπότε δημιουργήσαμε ένα script για να τις εντοπίσουμε και στην συνέχεια να τις διαγράψουμε. Το script απλά δοκιμάζει να ανοίξει τις εικόνες μία μία και αν υπάρχει πρόβλημα δημιουργεί exception και καταγράφει το όνομα του αρχείου. Στην συνέχεια τις διαγράφουμε. Το script που έτρεξε είναι το **script3.py** και έιναι διαθέσιμο στο τελικό .ipynb.

Στην συνέχεια θα δημιουργήσουμε ένα dictionary με τα μοντέλα που θα δοκιμάσουμε. Θα δοκιμάσουμε αρχιτεκτονική CNN και θα πειραματιστούμε με hyper parameters.Τα hyperparameters με τα οποία θα πειραματιστούμε είναι ο αριθμός των kernels, το μέγεθος τους, το μέγεθος του hidden layer και ο αριθμός των convolutional layers. Ξεκινήσαμε με πολύ μικρή αρχιτεκτονική με 2 και 3 convolutional layers και αριθμό kernels 4 και 8 αλλά είδαμε οτι το training πάει γρήγορα οπότε αποφασίσαμε να πάμε σε πιό πολύπλοκο μοντέλα αλλά όχι πολύ πολύπλοκα δεδομένου ότι έχουμε μόνο 4000 εικόνες. Τα μοντέλα που τελικά θα δοκιμάσουμε ξεκινάνε απο 32 ή 64 kernels για το

πρώτο convolutional διπλασιάζοντας τα kernels ανά layer. Ο αριθμός των layers θα είναι 2, 3 ή 4. Το fully connected θα έχει μέγεθος από 512 έως 1024 και τέλος το kernel size θα έχει μέγεθος 3 ή 4. Το script όπου ορίζουμε τις αρχιτεκτονικές είναι το **script4.py** και είναι διαθέσιμο στο τελικό .ipynb.

Στην συνέχεια επειδή το colab έχει τον περιορισμό των 6 ωρών θα δημιουργήσουμε μία μέθοδο save_state και μία load_state ώστε πρίν κάνει restart το colab να σώζουμε το σημείο της εκπαίδευσης και όταν το ξανατρέξουμε να συνεχίζει από εκεί που σταμάτησε. Το πρώτο που θα σώζει είναι πια απο τις 6 αρχιτεκτονικές δοκιμάζαμε. Το δεύτερο είναι το μοντέλο από την στιγμή που σταμάτησε η εκπαίδευση και το τρίτο είναι πόσες εποχές έχει εκπαιδευτεί το μοντέλο πριν την διακοπή. Αν υπάρχει .state τότε αρχικοποιούνται οι παράμετροι model,architecture και epochs_done από το αρχείο αλλιώς δημιουργείται η πρώτη αρχιτεκτονική και ξεκινάει η εκπαίδευση. Η μέθοδος που το κάνει αυτό είναι η load_state. Ο κώδικας για την save_state/load_sate βρίσκεται στο αρχείο **script5.py** και είναι διαθέσιμος στο τελικό .ipynb.

Στην συνέχεια επειδή η tensorflow τρέχει τις εποχές εσωτερικά στην fit πρέπει να βρούμε έναν τρόπο να σώζουμε το state πρίν το colab κάνει restart όπως περιγράφηκε προηγουμένως. Για τον λόγο αυτό δημιουργούμε μία κλάση callback που παίρνει σαν είσοδο τον χρόνο εκκίνησης και κάνει override την on_epoch_end. Όταν εκτελείται η τελευταία ελέγχει αν έχουν περάσει οι 5 ώρες και αν ναι σώζει το state καλώντας την save_state και τερματίζει την εκπαίδευση. Το script που το κάνει αυτό είναι το **script6.py.**

Στην συνέχεια φτιάχνουμε ένα function που δημιουργεί το μοντέλο. Παίρνει παραμέτρους τα hyper parameters από το προηγούμενο και δημιουργεί το μοντέλο. Το script που δημιουργει το μοντέλο είναι το **script7.py**. Εδώ επίσης αρχικοποιούμε τα βάρη για κάθε επίπεδο. Σύμφωνα με την βιβλιογραφία και αυτά που είπαμε στο μάθημα είναι καλό όλα τα μοντέλα που θα δοκιμάσουμε να έχουν ίδια βάρη στην εκκίνηση της εκπαίδευσης για αυτό χρησιμοποιούμε seed στον weight initializer. Επιλέγουμε να αρχικοποιήσουμε τα βάρη με τιμές με μέση τιμή 0 και std 0.1. Επίσης εφαρμόζουμε L2 regularization και dropout.

Στην συνέχεια θα δημιουργήσουμε τους data generators για train/validation/test. Οι generators αρχικά κάνουν resize και scale τις εικόνες. Στην συνέχεια εφαρμόζουμε data augmentation. Είναι μία τεχνική που εισάγει diversity στο dataset και με αυτόν τον τρόπο βοηθάει ώστε το μοντέλο να γενικεύει πιο αποτελεσματικά και στο μοντέλο που θα εκπαιδεύσουμε θα το εφαρμόσουμε. Αποτελείται από μετασχηματισμούς πάνω στις εικόνες όπως τροποποίηση χρώματα random rotation random scaling και άλλα. Επίσης όπως αναφέραμε στο μάθημα θα μπορούσε να χρησιμοποιηθεί για την δημιουργία νέων

εικόνων με βάση υπάρχουσες και την εισαγωγή τους στην διαδικασία της εκπαίδευσης ώς νέα samples.
Αυτή η δεύτερη χρήση θα μπορούσε να χρησιμοποιηθεί στην περίπτωση που εξετάζουμε, όπου το dataset είναι αρκετά μικρό δημιουργώντας ένα μεγάλο ποσοστό νέων εικόνων 20%-30% και να δούμε αν θα βοηθούσε στην ακρίβεια του μοντέλου. Στην περίπτωση που βλέπαμε καλή απόδοση θα μπορούσαμε να εκπαιδεύσουμε ένα πιο βαθύ δίκτυο. Για την πρώτη χρήση θα χρησιμοποιήσουμε τα layers που προσφέρει η tf, RandomRotation και RandomFlip.Το data augmentation εφαρμόζεται στον train_generator στο **script8.py**.

Στην συνέχεια θα αντιμετωπίσουμε το class imbalance. Αυτό που κάνουμε είναι να εφαρμόσουμε τον τύπο για τον υπολογισμό των βαρών που χρησιμοποιεί και η sklearn. Αφού δούμε απο τους generators τις αντιστοιχίες των κλάσεων στους αριθμούς 0-8 δημιουργούμε το dictionary με τα βάρη που περιμένει η fit. Το script που υπολογίζει τα βάρη είναι το **script9.py**.

Τέλος έχουμε το κύριο script εκπαίδευσης. Εδώ χρησιμοποιώντας όλα τα παραπάνω πραγματοποιούμε την εκπαίδευση. Αρχικοποιούμε τις μεταβλητές και ελέγχουμε αν συνεχίζεται η εκπαίδευση ή ξεκινάει από την αρχή. Αν συνεχίζεται φορτώνουμε το state όπως περιγράφηκε παραπάνω αλλιως δημιουργούμε το πρώτο μοντέλο. Δημιουργούμε τους data generator. Όταν ολοκληρωθεί η εκπαίδευση ενός μοντέλου πρώτα κάνουμε plot το train/val loss και το train/val accuracy και τα σώζουμε και στην συνέχεια σώζουμε το μοντέλο ώστε να κάνουμε την τελική αξιολόγηση στο test set.

Για το dataset που περιγράψαμε τρέξαμε την εκπαίδευση για 100 εποχές. Παρατηρήσαμε ότι το train/val loss μειωνόταν πολύ απότομα στην αρχή και συνέχιζε μέχρι και την εποχή 100. Το train accuracy όμως δεν ξεπέρασε ποτέ το 0.35 στο train set και στο validation έμενε σταθερά στο 0.9-0.12 δηλαδή το τυχαίο accuracy για 9 κλάσεις. Ελέγξαμε τον κώδικα για λάθη και δοκιμάσαμε να κάνουμε αλλαγές στις παραμέτρους του μοντέλου. Αλλάξαμε και ατομικά και σε συνδυασμό τα activation functions, το μέγεθος του fully connected τον αριθμό των kernels και το μέγεθος τους καθώς και τα convolutional layers απο 2 έως 5. Δοκιμάστηκε μεγάλο εύρος τιμών και δεν υπήρχε αλλαγή. Η επόμενη σκέψη ήταν να ξαναγράψουμε το μοντέλο σε pytorch για να επαληθεύσουμε ότι το μοντέλο δεν λειτουργεί ή να δοκιμάσουμε ένα νέο dataset. Επιλέξαμε την δεύτερη επιλογή.

Επιπλέον με βάση αυτά που είπαμε στην παρουσίαση δοκιμάσαμε τα παρακάτω πειράματα
Πρώτον ενώσαμε τις κλάσεις ώστε απο 9 να έχουμε 3. Τις κλάσεις 1,2,3 τις συγχωνεύσαμε σε μία, δηλαδή απλά Negative. Τις 4,5,6 σε μία Neutral και τις 7,8,9 σε

μία Positive. Τα αποτελέσματα που πήραμε μετά απο 50 εποχές ήταν τα ίδια με τις 9 κλάσεις δηλαδή accuracy 0.33 δηλαδή τυχαίο.
Το δεύτερο πείραμα θα κάνουμε το ίδιο με το προηγούμενο αλλα επίσης εκτός από το να ξαναχωρισουμε το dataset πάλι σε σε 3 κλάσεις αντί για 9 θα πάρουμε majority vote ανάμεσα στους 5 annotators. Το script που το κάνει αυτό είναι το **script1_v2.py** και είναι διαθέσιμο στον φάκελο Dataset1_v2. Επίσης το .ipynb που έτρεξε για αυτό το πείραμα είναι στον ίδιο φάκελο. Εδώ πάλι βλέπουμε ότι δεν υπάρχει καμία βελτίωση στο accuracy, παραμένει στο επίπεδο της τυχαίας επιλογής. Με βάση τα πειράματα και τις κατευθύνσεις που δόθηκαν βλέπουμε ότι στο συγκεκριμένο dataset το annotation δεν δίνει καμία πληροφορία για τις εικόνες.

2.Dataset 2

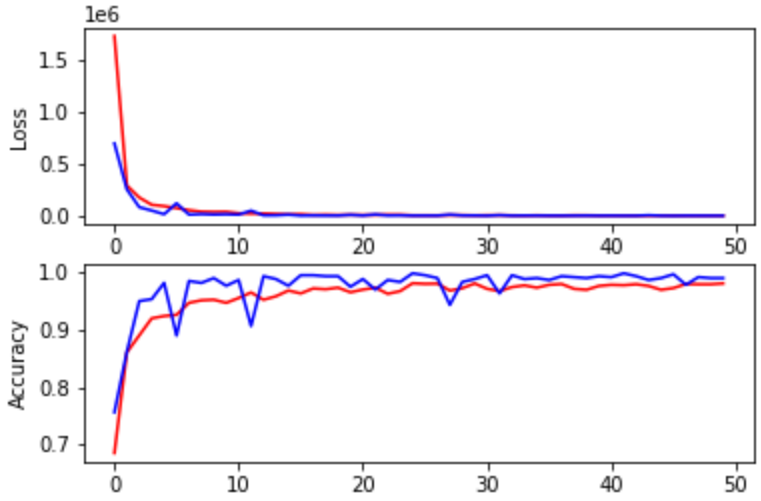
Το δεύτερο dataset αποτελείται από 15 κλάσεις με φωτογραφίες λαχανικών. Έχει 15 κλάσεις και είναι χωρισμένο σε train/validation/test. Οι φωτογραφίες ανα κλάση είναι balanced οπότε δεν χρειάζεται να υπολογίσουμε και να χρησιμοποιήσουμε βάρη στην fit. Για ευκολία κρατήσαμε μόνο 3 κλάσεις. Με μερικές αρχικές δοκιμές το val accuracy του μοντέλου βελτιωνοταν οπότε συμπεράναμε οτι το πρόβλημα στην προηγούμενη περίπτωση ήταν το ίδιο το dataset πιθανόν επειδή οι φωτογραφίες δεν είχαν σχέση μεταξύ τους δεδομένου ότι αποτελούνταν από διάφορες φωτογραφίες πολύ διαφορετικές, με το ερώτημα προς τον annotator τί συναίσθημα σου προκαλεί όταν την βλέπεις. Να σημειωθεί ότι το μοντέλο που χρησιμοποιήσαμε στο δεύτερο dataset ήταν ίδιο με το πρώτο που δεν δούλευε. Δεδομένων αυτών συνεχίσαμε με το δεύτερο dataset.

Για το δεύτερο dataset κρατάμε όλα τα προηγούμενα εκτός από τα class weights που δεν χρειάζονται. Το **script10.py** ανοίγει τα 6 αποθηκευμέναν μοντέλα και κάνει το τελικό evaluation με τον datagenerator του test set και παρουσιάζει τα αποτελέσματα.
Στην συνέχεια παρουσιάζονται τα αποτελέσματα για τα 6 υποψήφια μοντέλα.

3. Αποτελέσματα Dataset 2

Με κόκκινο train με μπλε validation, άξονας x εποχές.
Μοντέλο 1

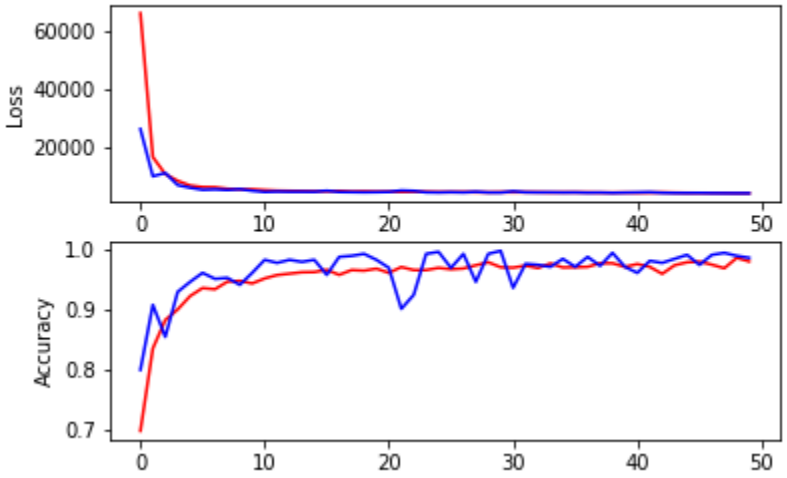
- Αριθμός kernels στο πρώτο επίπεδο 64
- Μέγεθος kernel 3
- Fully connected 1024
- Convolutional layers 4



Αποτέλεσμα accuracy στο test set: `0.9950000047683716`

Μοντέλο 2

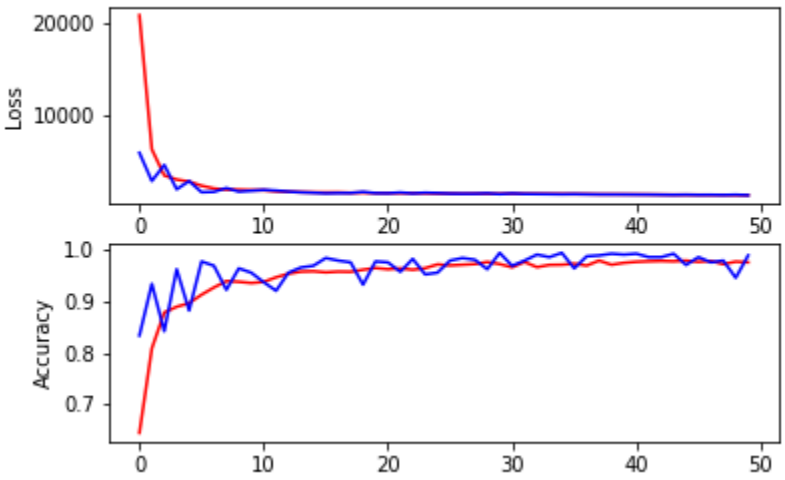
- Αριθμός kernels στο πρώτο επίπεδο 64
- Μέγεθος kernel 3
- Fully connected 1024
- Convolutional layers 3



Αποτέλεσμα accuracy στο test set:`0.9816666841506958`

Μοντέλο 3

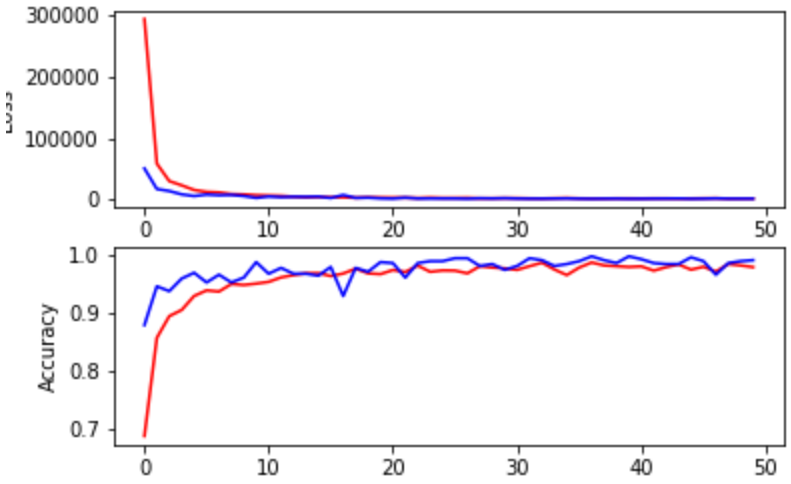
- Αριθμός kernels στο πρώτο επίπεδο 32
- Μέγεθος kernel 3
- Fully connected 768
- Convolutional layers 3



Αποτέλεσμα accuracy στο test set: `0.9933333396911621`

Μοντέλο 4

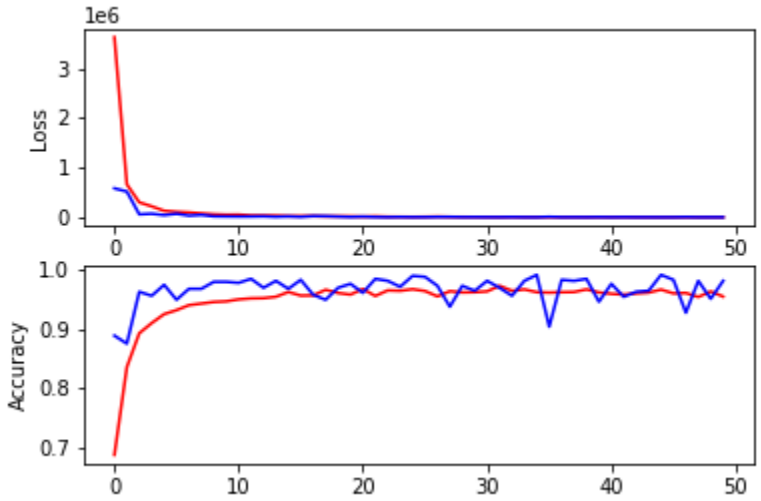
- Αριθμός kernels στο πρώτο επίπεδο 32
- Μέγεθος kernel 3
- Fully connected 1024
- Convolutional layers 4



Αποτέλεσμα accuracy στο test set: `0.9950000047683716`

Μοντέλο 5

- Αριθμός kernels στο πρώτο επίπεδο 64
- Μέγεθος kernel 4
- Fully connected 512
- Convolutional layers 4

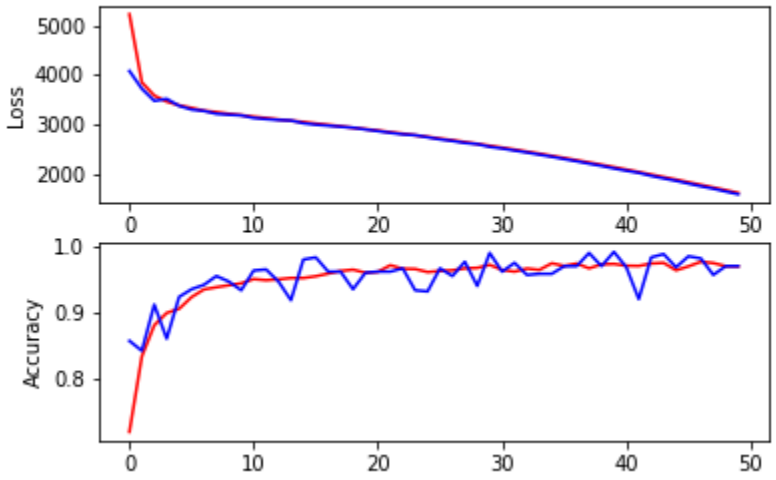


Αποτέλεσμα accuracy στο test set: `0.9733333587646484`

Μοντέλο 6

- Αριθμός kernels στο πρώτο επίπεδο 32
- Μέγεθος kernel 4

- Fully connected 768
- Convolutional layers 2



Αποτέλεσμα accuracy στο test set: `0.9833333492279053`

Conclusions

Βλέπουμε ότι όλα τα μοντέλα τα πάνε εξίσου καλά. Τα regularizations για το overfitting αποδίδουν, δεν βλέπουμε κανένα σημάδι. Στο monitor ίσως θα έπρεπε να παρακολουθούμε το val_accuracy αντί του loss. Βλέπουμε ότι το τελευταίο μοντέλο που είναι και το πιό απλό με μόνο δύο convolution layers τα πάει εξίσου καλά με τα άλλα πράγμα που μας κάνει να συμπεραίνουμε ότι μπορούμε να δοκιμάσουμε πιο μικρά μοντέλα και να μειώσουμε επίσης το fully connected για να μειώσουμε δραστικά το μέγεθος του μοντέλου. Φαίνεται ότι το πρόβλημα classification που λύσαμε είναι πολύ απλό. Επίσης στον αριθμό των εποχών ξεκινήσαμε από τις 100 και μετά απο πειραματισμό κατεβήκαμε στις 50 αλλά βλέπουμε ότι το μοντέλο μαθαίνει πολύ γρήγορα στην αρχή και πετυχαίνει πολύ καλή απόδοση με λίγες εποχές. Θα μπορούσαμε να δοκιμάσουμε χωρίς regularization για να δούμε πως θα συμπεριφερθεί το μοντέλο. Σχετικά με το πρώτο dataset εκτός του οτι δοκιμάσαμε ακριβώς τον ίδιο κώδικα με το δεύτερο αλλάζοντας μόνο τα filenames και τον αριθμό των κλάσεων στο output layer είδαμε αν οι generators αντιστοιχούν του ίδιους φακέλους στις κλάσεις, αυτό επίσης δεν έδινε κάποιο σφάλμα. Η μόνη υπόθεση που έμεινε είναι αυτή που αναφέραμε στην αρχή αν και το οτι το αποτέλεσμα είναι απόλυτα τυχαίο είναι περίεργο, θα περιμέναμε κάποια απόκλιση από το εντελώς τυχαίο. Άλλη μία σκέψη είναι ότι ίσως φταίει το ότι οι εικόνες είναι υψηλής ανάλυσης και κάνοντας resize σε 150 χάνεται η πληροφορία .