22 / 1 / 2025

ΟΡΑΣΗ ΥΠΟΛΟΓΙΣΤΩΝ

3^{Η} ΕΡΓΑΣΙΑ
ΟΡΑΣΗΣ ΥΠΟΛΟΓΙΣΤΩΝ
2024-2025

Ηλίας Ξανθόπουλος 58545

Πίνακας Περιεχομένων

1. Μέρος Α: Ταξινόμηση πολλαπλών κλάσεων (Bag of Visual Words)

1.α. Υλοποίηση

Το *μέρος Α* πραγματεύεται την ταξινόμηση πολλαπλών κλάσεων κάνοντας χρήση του μοντέλου BOVW. Για την επίλυση του συγκεκριμένου προβλήματος έχουν υλοποιηθεί τα παρακάτω αρχεία .py:

- utils.py

Μέσα σε αυτό το αρχείο ορίζονται συναρτήσεις που είναι αναγκαίες για την επίλυση του προβλήματος. Συγκεκριμένα, έχει σχεδιαστεί η συνάρτηση *extract_local_features()* η οποία δέχεται ως είσοδο το directory μίας εικόνας και αφού εξάγει μέσω του ανιχνευτή SIFT τα τοπικά χαρακτηριστικά της και τους αντίστοιχους περιγραφείς τους, επιστρέφει μια λίστα με αυτά τα descriptors.

- part_A_vocabulary_index.py

Το αρχείο αυτό αποσκοπεί στην δημιουργία ενός λεξικού χαρακτηριστικών από τις εικόνες εκπαίδευσης (*γραμμές 22-27*), όπως και στην κωδικοποίηση αυτών εικόνων με τη χρήση του λεξικού (*γραμμές 29-48*). Για την υλοποίηση αυτών των λειτουργιών πρώτα εξάγονται οι περιγραφείς των τοπικών χαρακτηριστικών όλων των εικόνων εκπαίδευσης “caltech-101_5_train” (*γραμμές 9-19*) κάνοντας χρήση της συνάρτησης *extract_local_features()* του αρχείου *utils.py*. Έπειτα γίνεται η δημιουργία του λεξικού, οι “λέξεις” του οποίου προκύπτουν από την ομαδοποίηση των υπολογισμένων descriptors των εικόνων χρησιμοποιώντας τον αλγόριθμο ομαδοποίησης K-means clustering. Το πλήθος των λέξεων που απαρτίζουν το λεξικό είναι ίσο με το K, και ορίζεται από την πρώτη παράμετρο της συνάρτησης της *γραμμής 24*. Το παραγόμενο λεξικό αποθηκεύεται στο αρχείο *vocabulary.npy*, καθώς χρησιμοποιείται αργότερα κατά την ταξινόμηση. Τέλος, η κωδικοποίηση των εικόνων εκπαίδευσης γίνεται στην ουσία μέσω αντιστοίχισης κάθε τοπικών χαρακτηριστικών τους με την κοντινότερη λέξη του λεξικού, οι συχνότητες αντιστοίχισης των οποίων τοποθετούνται σε ένα ιστόγραμμα. Τα ιστογράμματα που εξάγονται αποθηκεύονται στο *index.npy*, όπως αποθηκεύονται στο *paths.py* και τα directories των εικόνων, για ευκολότερη μετέπειτα πρόσβαση.

- part_A_knn.py

Το αρχείο αυτό αξιοποιεί τα προηγουμένως δημιουργημένα αρχεία *vocabulary.npy*, *index.npy* και *paths.py* για την εκπαίδευση ενός ταξινομητή k-Nearest Neighbors (*γραμμές 32-51*) και για το testing του πάνω στο σύνολο εικόνων δοκιμής caltech-101_5_test για την αξιολόγηση της ακρίβειας του συστήματος (*γραμμές 53-279*). Για το training, πρώτα αντιστοιχίζονται labels σε κάθε εικόνα των δεδομένων εκπαίδευσης *(γραμμές 35-48)* ανάλογα με την κλάση στην οποία ανήκει και έπειτα οι ταμπέλες αυτές μαζί με τους περιγραφείς του *index.npy* εκπαιδεύουν τον ταξινομητή k-NN (*γραμμές 50-51*)· ένα “διάγραμμα” με ταξινομημένα σημεία στο οποίο εφαρμόζεται ο

αλγόριθμος k-NN για την εύρεση της κλάσης του άγνωστου σημείου. Κατά το testing, για όλες τις εικόνες δοκιμής κατασκευάζεται από τα τοπικά τους χαρακτηριστικά ένα ιστόγραμμα συχνότητας εμφάνισης λέξεων του *vocabulary.npy* (*γραμμές 57-58*, *72-74* και *133-135*), το οποίο έπειτα εισάγεται στο εκπαιδευμένο k-NN σύστημα για να αποδώσει ένα response, δηλαδή την κατηγορία στην οποία ταξινομείται η άγνωστη εικόνα (*γραμμές 76-77* και *137-138*). Η 2η παράμετρος της συνάρτησης των *γραμμών 76* και *137* αντιστοιχεί στο k του k-NN, δηλαδή στον αριθμό των πλησιέστερων γειτόνων που εξετάζονται. Επιπλέον, κατά την ταξινόμηση γίνεται καταμέτρηση των σωστών ταξινομήσεων (*γραμμές 79-92* και *152-182*), από την οποία υπολογίζεται η ακρίβεια του μοντέλου τόσο ανά κλάση, όσο και συνολικά (*γραμμές 94-96* και *184-227)*. Τέλος, για την διευκόλυνση των αξιολογήσεων των *Κεφαλαίων 1.β.* και *1.γ*, έχει προστεθεί στο αρχείο μια "διεπαφή" (*γραμμές 13-24)* στην οποία ο χρήστης μπορεί να επιλέξει ανάμεσα σε δύο λειτουργίες εκτέλεσης· την "OPTIMAL K-VALUE SEARCHING MODE" (θέτοντας την παράμετρο mode ίση με 1) και την "SINGULAR TESTING MODE" (για τιμή του mode διαφορετική του 1). Η πρώτη λειτουργία επιτρέπει στον χρήστη να εξετάσει αυτόματα την ακρίβεια του μοντέλου για τις διάφορες τιμές του k από 1 εώς και k_max (την οποία ορίζει αυτός μέσα στο μπλοκ κώδικα). Σε αυτήν περίπτωση, προβάλλεται για όλα τα εξετασμένα k μόνο η συνολική ακρίβεια της ταξινόμησης, όπως και η τιμή του k που οδηγεί στα καλύτερα αποτελέσματα (*γραμμές 98-102*). Η δεύτερη λειτουργία επιτρέπει στον χρήστη να αξιολογήσει την ακρίβεια ταξινόμησης του μοντέλου με λεπτομέρεια (συνολική και ανά τάξη ακρίβεια) για την τιμή του k που ορίζει αυτός μέσω της k_sel, καθώς και να προβάλει όλες τις ταξινομημένες φωτογραφίες αν θέσει την μεταβλητή view_imgs ίση με 1.

- part_A_1vsAll.py

Το αρχείο αυτό αξιοποιεί τα *vocabulary.npy*, *index.npy* και *paths.py* που δημιούργησε το *part_A_vocabulary_index.py* για την εκπαίδευση ενός ταξινομητή του σχήματος One-versus-All (*γραμμές 20-103*) και για το testing του πάνω στο σύνολο εικόνων δοκιμής caltech-101_5_test για την αξιολόγηση της ακρίβειας του συστήματος (*γραμμές 105-294*). Κατά το training ορίζονται πέντε SVM γραμμικού kernel (μία για κάθε κλάση) και πέντε λίστες labels οι οποίες χρησιμοποιούνται μαζί με τους περιγραφείς του *index.npy* για να εκπαιδεύσουν την κάθε SVM. Στην κάθε λίστα σημειώνονται με την ετικέτα 1 οι εικόνες της caltech-101_5_train που ανήκουν σε μία συγκεκριμένη κλάση, ενώ οι υπόλοιπες με την ετικέτα 0 (*γραμμές 50-103)*. Αυτή η δομή είναι το σχήμα One-versus-All και χρησιμοποιείται διότι η SVM από μόνη της μπορεί να κάνει μόνο δυαδική κατηγοριοποίηση, όποτε πέντε SVM (δυαδικά classifications) οδηγούν σε μία ταξινόμηση 5 κλάσεων. Όλες οι SVM αποθηκεύονται και χρησιμοποιούνται για την δοκιμή του συστήματος, στην οποία τα τοπικά χαρακτηριστικά όλων των εικόνων test εισάγονται σε ιστογράμματα συχνότητας εμφάνισης λέξεων του *vocabulary.npy* (*γραμμές 110-111* και *141-143*) και έπειτα αυτά τα ιστογράμματα εισάγονται σε κάθε μια από τις αποθηκευμένες SVM για την παραγωγή πέντε αποκρίσεων, μία για κάθε κλάση (*γραμμές 145-149*). Το αρνητικότερο response αντιστοιχεί στην κλάση στην οποία ταξινομούνται οι εικόνες, επομένως

αυξάνεται κάθε φορά ο counter που αντιστοιχεί σε αυτόν (*γραμμές 167-197*) και με την λήξη της δοκιμής προβάλλεται η ακρίβεια του συστήματος ανά τάξη και συνολικά (*γραμμές 199-242*). Κλείνοντας, όπως και στο προηγούμενο αρχείο που εξετάστηκε, έτσι και σε αυτό, έχει προστεθεί ένα μπλοκ κώδικα (*γραμμές 13-18*) στο οποίο ο χρήστης μπορεί να επιλέξει αν θέλει να προβάλλει τις ταξινομημένες εικόνες θέτοντας την view_imgs ίση με 1.

1.β. Επίδραση εμπλεκομένων παραμέτρων

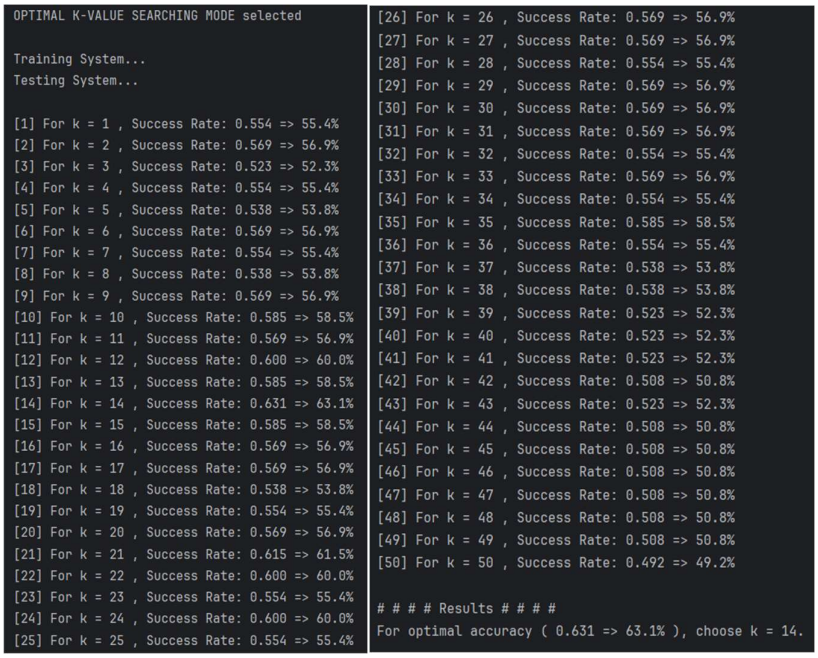
• Μέγεθος οπτικού λεξικού

Όπως αναφέρθηκε στο *Κεφάλαιο 1.α*, το πλήθος των "λέξεων" που αποτελούν το οπτικό λεξικό ορίζεται στο αρχείο *part_A_vocabulary_index.py* στην *γραμμή 24* από την πρώτη παράμετρο της συνάρτηση cv.BOWKMeansTrainer(). Για να φανεί η επίδραση της αλλαγής αυτής της παραμέτρου πρέπει με κάθε αλλαγή να γίνεται εκτέλεση πρώτα του *part_A_vocabulary_index.py* και μετά των *part_A_knn.py* και *part_A_1vsAll.py*. Στο *part_A_knn.py* επιλέγονται οι ρυθμίσεις mode=1, έτσι ώστε να φαίνεται η επίδραση για το καλύτερο δυνατό k (αλλά ταυτόχρονα και για άλλες τιμές του), και k_max=50, το οποίο βέβαια είναι πολύ μεγάλο. Επιπλέον, στο *part_A_1vsAll.py* προτείνεται η ρύθμιση view_imgs=1 για πιο γρήγορη εκτέλεση. Τα μεγέθη οπτικού λεξικού που εξετάζονται είναι στο σύνολο 10 και φαίνονται μαζί με τα αποτελέσματα τους στον *πίνακα 1*. Φωτογραφίες της εξόδου των υλοποιήσεων δεν προβάλλονται για όλες τις τιμές που εξετάζονται για πρακτικούς λόγους, αλλά μόνο για VW=100 (Visual Words) που είναι και η περίπτωση με την μεγαλύτερη συνολικά ακρίβεια (*εικόνες 1* και *2*).

Σχόλιο: Δίνεται παραπάνω η έμφαση στην λέξη "συνολικά" διότι υπάρχουν τιμές VW για τις οποίες επιτυγχάνεται η υψηλότερη ακρίβεια ως προς τον ένα ταξινομητή, αλλά όχι προς τον άλλον. Οπότε ως καλύτερη τιμή VW επιλέγεται αυτή με τον μεγαλύτερο μέσο όρο ακρίβειας των δύο ταξινομητών.

```
Training SVMs...
Testing System...

# # # # Evaluation of SVM One vs All Classification # # # #

=== [1] Accordions ===
- Correctly Classified: 9
- Total Accordions: 11
- Success Rate: 0.818 => 81.8%

=== [2] Electric Guitars ===
- Correctly Classified: 12
- Total Electric Guitars: 18
- Success Rate: 0.667 => 66.7%

=== [3] Grand Pianos ===
- Correctly Classified: 18
- Total Grand Pianos: 21
- Success Rate: 0.857 => 85.7%

=== [4] Mandolins ===
- Correctly Classified: 3
- Total Mandolins: 9
- Success Rate: 0.333 => 33.3%

=== [5] Metronomes ===
- Correctly Classified: 6
- Total Metronomes: 6
- Success Rate: 1.000 => 100.0%

<<====== Total Accuracy ======>>
- Correctly Classified: 48
- Total Objects: 65
- Success Rate: 0.738 => 73.8%
```

Εικόνα 1: Η έξοδος του part_A_1vsAll.py για μέγεθος οπτικού λεξικού ίσο με 100



Εικόνα 2: Η έξοδος του part_A_knn.py για μέγεθος οπτικού λεξικού ίσο με 100

Επίδραση μεγέθους οπτικού λεξικού (VW) στην ακρίβεια			
Για VW = 25	k-NN: 63.1% (k = 8)	Για VW = 105	k-NN: 66.2% (k = 2)
	1vsAll: 46.2%		1vsAll: 67.7%
	M.O.: 54.65%		M.O.: 66.95%
Για VW = 50	k-NN: 61.5% (k = 9)	Για VW = 110	k-NN: 63.1% (k = 9)
	1vsAll: 66.2%%		1vsAll: 66.2%
	M.O.: 63.85%		M.O.: 64.65%
Για VW = 75	k-NN: 63.1% (k = 11)	Για VW = 125	k-NN: 66.2% (k = 20)
	1vsAll: 58.5%		1vsAll: 66.2%
	M.O.: 60.8%		M.O.: 66.2%
Για VW = 95	k-NN: 63.1% (k = 18)	Για VW = 150	k-NN: 61.5% (k = 19)
	1vsAll: 63.1%		1vsAll: 67.7%
	M.O.: 63.1%		M.O.: 64.6%
Για VW = 100	**k-NN: 63.1% (k = 14)**	Για VW = 175	k-NN: 61.5% (k = 19)
	1vsAll: 73.8%		1vsAll: 66.2%
	M.O.: 68.45%		M.O.: 63.85%

Πίνακας 1: Επίδραση μεγέθους οπτικού λεξικού (VW) στην ακρίβεια. Με bold γράμματα είναι η βέλτιστη τιμή VW.

Όπως φαίνεται και στον *Πίνακα 1*, για την επίτευξη της μεγαλύτερης ακρίβειας επιλέγεται μέγεθος οπτικού λεξικού ίσο με 100, καθώς οδηγεί σε ακρίβεια 63.1% για k-NN με k=14, 73.8% για One-versus-All και 68.45% συνολικά.

• Αριθμός πλησιέστερων γειτόνων k

Για την εύρεση της επίδρασης του αριθμού πλησιέστερων γειτόνων k που εξετάζει ο k-NN στην ακρίβεια, αρκεί η εκτέλεση πρώτα του *part_A_vocabulary_index.py*, έστω με μέγεθος λεξικού οπτικού λέξεων ίσο με 100, και μετά η εκτέλεση του *part_A_knn.py* με ακριβώς τις ίδιες ρυθμίσεις με εκείνες της προηγούμενης περίπτωσης, δηλαδή με mode=1 και k_max=50. Εκτελώντας τα παραπάνω προκύπτουν τα αποτελέσματα της *εικόνας 2*, από την οποία φαίνεται ότι η βέλτιστη τιμή για το συγκεκριμένο σενάριο είναι το k=14, οδηγώντας σε ακρίβεια ταξινόμησης ίση με 63.1%. Εκτελώντας το *part_A_knn.py* για ακόμη μια φορά, αλλά αυτή την φορά με τις ρυθμίσεις mode=0 και k_sel=14, προκύπτουν πιο λεπτομερή αποτελέσματα για την ταξινόμηση με τιμή k=14, τα οποία φαίνονται στην *εικόνα 3*. Για καλύτερη οπτικοποίηση αποτελεσμάτων συνιστάται η ρύθμιση view_imgs=1.

```
SINGULAR TESTING MODE selected

Training System...
Testing System...

# # # # Evaluation of k-NN Classification # # # #

=== [1] Accordions ===
- Correctly Classified: 9
- Total Accordions: 11
- Success Rate: 0.818 => 81.8%

=== [2] Electric Guitars ===
- Correctly Classified: 6
- Total Electric Guitars: 18
- Success Rate: 0.333 => 33.3%

=== [3] Grand Pianos ===
- Correctly Classified: 20
- Total Grand Pianos: 21
- Success Rate: 0.952 => 95.2%

=== [4] Mandolins ===
- Correctly Classified: 2
- Total Mandolins: 9
- Success Rate: 0.222 => 22.2%

=== [5] Metronomes ===
- Correctly Classified: 4
- Total Metronomes: 6
- Success Rate: 0.667 => 66.7%

<<====== Total Accuracy ======>>
- Correctly Classified: 41
- Total Objects: 65
- Success Rate: 0.631 => 63.1%
```

Εικόνα 3: Η έξοδος του part_A_knn.py για μέγεθος οπτικού λεξικού 100 και k=14

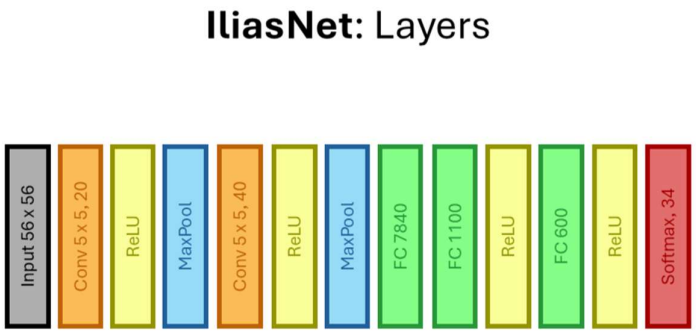
2. Μέρος Β: Ταξινόμηση πολλαπλών κλάσεων (Deep Learning)

2.α. Εισαγωγή

Το *μέρος Β* πραγματεύεται την ταξινόμηση πολλαπλών κλάσεων κάνοντας χρήση συνελικτικών δικτύων. Για την επίλυση αυτού το προβλήματος έχουν υλοποιηθεί δύο εφαρμογές. Η πρώτη αποτελεί το αρχείο *part_B_deep_learning.ipynb* και αξιοποιεί ένα μη προεκπαιδευμένο δίκτυο (το… IliasNet!) για την ταξινόμηση, ενώ η δεύτερη αποτελεί το αρχείο *part_B_pretrained.ipynb* και χρησιμοποιεί ένα προ-εκπαιδευμένο δίκτυο και συγκεκριμένα το ResNet50v2. Για κάθε μια από τις δύο υλοποιήσεις περιγράφεται παρακάτω η αρχιτεκτονική των επιπέδων, αναλύεται ο κώδικας και η διαδικασία εκπαίδευσης και γίνεται μια ποσοτική εκτίμηση της επίδοσης τους στα δεδομένα testing. Κατά την ανάλυση του κώδικα στο παρόν έγγραφο, δεν θα γίνεται επεξήγηση όλων των τμημάτων του, όπως και δεν θα παρέχονται screenshots των εκτελέσεων, καθώς επεξηγούνται οι λειτουργίες των block και παρέχονται τα μηνύματα εξόδου στα Jypyter Notebooks, χωρίς να είναι απαραίτητη η επανεκτέλεση. Θα σχολιάζεται μόνο οτιδήποτε χρήζει περαιτέρω ανάλυση.

2.β. Αρχιτεκτονική μη προ-εκπαιδευμένου δικτύου (IliasNet)

Για την ταξινόμηση από μη εκπαιδευμένο δίκτυο σχεδιάστηκε το IliasNet. Το IliasNet, όπως φαίνεται και από την *εικόνα 4*, αποτελείται από 12 επίπεδα (χωρίς την είσοδο)· δύο δομές {Συνέλιξη, Μη γραμμικότητα (ReLU), Spatial (Max) Pooling}, ένα επίπεδο flattening ακολουθούμενο από δύο δομές {Fully Connected, Μη γραμμικότητα (ReLU)} και στην έξοδο από ένα επίπεδο Softmax.



Εικόνα 4: Τα επίπεδα του συνελικτικού δικτύου IliasNet

Στον *πίνακα 2* φαίνονται τα βάρη στο δίκτυο καθώς και τα χαρακτηριστικά και οι διαστάσεις των εξερχόμενων feature maps του κάθε επιπέδου. Στα επίπεδα συνέλιξης τα K,P και S (υπόμνημα στην λεζάντα του *πίνακα 2*) έχουν οριστεί έτσι ώστε να διατηρούν σταθερές τις διαστάσεις των feature maps της εισόδου και στην έξοδο, ενώ

στα επίπεδα MaxPooling έτσι ώστε να μειώνουν τις διαστάσεις στο μισό. Τα πλήθη των φίλτρων C_{out}, όπως και των νευρώνων στα πλήρως συνδεδεμένα επίπεδα, επιλέχθηκαν πειραματικά. Τέλος, η Softmax έχει 34 εξόδους, αντιστοιχίζοντας έτσι μια πιθανότητα πρόβλεψης για κάθε μία από τις 34 κλάσεις των δεδομένων.

Layer	Output Size	Weight Size
Input	1×56×56	
Conv (Cout = 20, K=5, P=2, S=1)	20×56×56	20×1×5×5
ReLU	20×56×56	
MaxPool (K=2, S=2)	20×28×28	
Conv (Cout = 40, K=5, P=2, S=1)	40×28×28	40×20×5×5
ReLU	40×28×28	
MaxPool (K=2, S=2)	40×14×14	
Flatten	7840	
Linear (7840 =>1100)	1100	7840×1100
ReLU	1100	
Linear (1100 =>600)	600	1100×600
ReLU	600	
SoftMax (600=>34)	34	600×34

Πίνακας 2: Τα βάρη, τα μεγέθη των εξόδων και τα χαρακτηριστικά των επιπέδων του συνελικτικού δικτύου IliasNet. Cout: πλήθος φίλτρων, K: διάσταση kernel, P: padding, S: stride

2.γ. Ανάλυση κώδικα / εκπαίδευση μη προ-εκπαιδευμένου δικτύου (IliasNet)

Όπως προαναφέρθηκε στο *κεφάλαιο 2.α*, η εκπαίδευση και η αξιολόγηση της ταξινόμησης του δικτύου IliasNet πραγματοποιείται στο αρχείο *part_B_deep_learning.ipynb*. Τα δεδομένα της συγκεκριμένης εφαρμογής είναι εκείνα του αρχείου *imagedb_btsd.zip*, το οποίο περιλαμβάνει τις εικόνες imagedb για training και imagedb_test για testing.

Σημαντικό βήμα πριν την εκπαίδευση ενός δικτύου είναι η κατανόηση των χαρακτηριστικών των εικόνων της βάσης και συγκεκριμένα των διαστάσεων τους. Οι διαστάσεις όπως και το aspect ratio είναι πολύτιμες πληροφορίες, διότι για να εισαχθούν εικόνες στο συνελικτικό δίκτυο πρέπει πρώτα να υποστούν ένα μετασχηματισμό των διαστάσεων τους. Αυτός ο μετασχηματισμός, μπορεί να οδηγήσει λόγω παραμόρφωσης σε απώλεια χαρακτηριστικών σημείων/πληροφοριών που διακρίνουν τις διάφορες τάξεις μεταξύ τους. Για την διευκόλυνση λοιπόν της επιλογής διαστάσεων εισόδου σχεδιάστηκε το *μπλοκ 2* το οποίο σαρώνει όλες τις εικόνες imagedb και επιστρέφει την μέση max διάσταση και το μέσο aspect ratio. Εκτελώντας αυτό το block προκύπτει ότι η μέση μέγιστη διάσταση είναι 132 pixels, ενώ το μέσο aspect ratio 0.94. Παρόλο που, με βάση αυτά τα αποτελέσματα, το μέγεθος αυτό δεν είναι το ιδανικό για την ελαχιστοποίηση των παραμορφώσεων, επιλέγεται μέγεθος εισόδου 56×56, διότι αυτό οδήγησε σε καλύτερη ακρίβεια.

Πέρα από μετασχηματισμό των διαστάσεων των δεδομένων, υπάρχουν και άλλες προ-επεξεργασίες στις οποίες μπορεί να υποβληθεί η βάση εικόνων. Όλες αυτές οι διεργασίες πραγματοποιούνται στο *μπλοκ 4*, δύο από τις οποίες στην συγκεκριμένη περίπτωση είναι οι μετατροπή των φωτογραφιών σε grayscale και ο διαχωρισμός των δεδομένων εκπαίδευσης σε δύο μέρη. Ο διαχωρισμός είναι αναγκαίος, γιατί η βάση από μόνη της δεν παρέχει δεδομένα για validation, οπότε διαχωρίζεται το training set σε 90% για εκπαίδευση και 10% για επικύρωση. Εξίσου αναγκαία είναι και η οργάνωση των εικόνων (οποιουδήποτε phase) σε batches, δηλαδή σε ομάδες οι οποίες εισάγονται μαζί στο δίκτυο. Το batch_size τόσο του training, όσο και του validation επιλέγεται να είναι ίσο με 50.

Πέρα από τις προαναφερθείσες, υπάρχουν και άλλες δύο επεξεργασίες οι οποίες, αν και κανονικά οδηγούν σε καλύτερη εκπαίδευση, δεν εφαρμόζονται εν τέλει, διότι χειροτερεύουν την ακρίβεια του μοντέλου. Η πρώτη είναι η επαύξηση δεδομένων μέσω της συνάρτησης ImageDataGenerator(), δηλαδή η τυχαία εφαρμογή μετασχηματισμών, όπως flipping, rotating, zooming, στις διαθέσιμες εικόνες για την αναπαραγωγή επιπλέον δεδομένων και άρα καλύτερη εκπαίδευση. Ο λόγος που δεν οδήγησε σε αποτελεσματικότερο μοντέλο είναι μάλλον επειδή οι διαθέσιμες εικόνες είναι τέτοιες ώστε με τους μετασχηματισμούς να χάνονται τα χαρακτηριστικά τους σημεία και άρα η δυνατότητα ταξινόμησής τους. Η δεύτερη διεργασία είναι το ανακάτεμα των εικόνων προτού εισαχθούν στο δίκτυο, έτσι ώστε να εισάγεται τυχαιότητα στο σύστημα. Αν και η τυχαιότητα είναι επιθυμητή σε τέτοιου είδους εφαρμογές, εδώ μάλλον τα δεδομένα γίνονται τόσο τυχαία, ώστε πλέον να μη είναι αποδοτική η εκπαίδευση.

Ο ορισμός της αρχιτεκτονικής του δικτύου γίνεται στο *μπλοκ 5*, στο οποίο επιλέγονται τα κατάλληλα επίπεδα και παράμετροι για την σύνθεση της δομής του IliasNet.

Στο *block 6* ορίζονται οι αντίστοιχοι παράμετροι και εκτελείται το training και validation του μοντέλου. Πέρα από το να διατυπωθούν φυσικά ποιες είναι οι εικόνες-δεδομένα, ορίζεται και η διάρκεια εκπαίδευσης για maximum 100 εποχές, ένα πλήθος session υπέρ αρκετό για την επίτευξη του καλύτερου δυνατού μοντέλου για ένα συγκεκριμένο κάθε φορά set παραμέτρων και δεδομένων. Τα βήματα ανά εποχή, δηλαδή το πόσα training batches διοχετεύονται στο δίκτυο ανά εποχή, θέτονται να είναι ίσα με το πηλίκο του πλήθους training εικόνων προς το batch size, κάτι που εξασφαλίζει ότι όλο το σύνολο φωτογραφιών εκπαίδευσης θα εισάγεται στο μοντέλο (τουλάχιστον) μια φορά ανά εποχή. Αν και δεν ορίζεται ευθέως κάτι τέτοιο και για το validation, αυτό ισχύει ως default αν δεν διατυπωθεί διαφορετικά. Τέλος, ορίζονται και δύο callbacks κατά την εκπαίδευση, δηλαδή ανακαλέσεις σε προηγούμενες εποχές· το save_best_callback και το early_stop_callback. Το πρώτο στην ουσία, μέσω της συνάρτησης tf.keras.callbacks.ModelCheckpoint(), αποθηκεύει στο αρχείο best_weights.keras τα τρεχούμενα βάρη του νευρωνικού στο τέλος κάθε εποχής, με την προϋπόθεση ότι οδηγούν στην χαμηλότερη τιμή validation loss μέχρι εκείνη την στιγμή. Αυτό το callback είναι αναγκαίο ώστε μέχρι το τέλος της διαδικασίας εκπαίδευσης να μην χαθούν τα καλύτερα δυνατά βάρη του μοντέλου και εν τέλει να εφαρμοσθούν σε αυτό. Το δεύτερο callback επιτυγχάνεται μέσω της συνάρτησης

tf.keras.callbacks.EarlyStopping(), η οποία μετά από ένα συγκεκριμένο αριθμό διαδοχικών εποχών που δεν βελτιώνεται το val_loss, τερματίζει άμεσα την εκπαίδευση και φορτώνει τα βέλτιστα βάρη, έτσι ώστε να μην υπάρχει άσκοπη κατανάλωση πόρων. Το πλήθος των διαδοχικών εποχών στο οποίο δεν πρέπει να υπάρχει βελτίωση ορίζεται από την μεταβλητή patience και θέτεται ίση με 20, ένα νούμερο αρκετά μεγάλο ώστε να εξασφαλίζεται η επίτευξη του καλύτερου δυνατού μοντέλου.

Κλείνοντας το κομμάτι της εκπαίδευσης, μετά την εκτέλεση του *μπλοκ 6*, δημιουργούνται μέσω του *μπλοκ 7* δύο διαγράμματα που απεικονίζουν την τιμή της ακρίβειας και του σφάλματος αντίστοιχα, κατά την διάρκεια όλων των εποχών, τόσο για το training, όσο και για το validation.

2.δ. Αξιολόγηση μη προ-εκπαιδευμένου δικτύου (IliasNet)

Η ποσοτική εκτίμηση της επίδοσης του δικτύου στο σύνολο των δεδομένων imaged_test γίνεται στο *μπλοκ 8*. Πρώτα εκτελείται η προ-επεξεργασία των εικόνων δοκιμής η οποία υλοποιείται με τον ίδιο τρόπο με εκείνον των δεδομένων training και validation, μόνο που σε αυτή την περίπτωση λαμβάνουν χώρα μόνο τρεις διεργασίες· ο μετασχηματισμός του μεγέθους σε 56×56 και η μετατροπή σε grayscale, διότι πάνω σε αυτές τις ρυθμίσεις έχει εκπαιδευτεί το δίκτυο, και η οργάνωση σε batches, το μέγεθος των οποίων δεν παίζει κανένα ρόλο στο αποτέλεσμα, καθώς το μοντέλο δοκιμάζεται σε όλο το σύνολο του imaged_test, ανεξάρτητα από το πλήθος και το μέγεθος των batches.

Με την ολοκλήρωση του testing, το μοντέλο αξιολογείται ως προς την ακρίβεια και ως προς το σφάλμα, τα οποία στην συγκεκριμένη εκτέλεση είναι 92.7% και 36.3% αντίστοιχα. Ποιοτικά αυτό το accuracy σημαίνει ότι μοντέλο είναι αρκετά καλό στην επίλυση του προβλήματος του classification, αφού από τις 289 εικόνες που εισάχθηκαν στο δίκτυο, οι 268 ταξινομήθηκαν στην σωστή κλάση, ένα ποσοστό αρκετά καλό δεδομένου της απλότητας του συστήματος και του μικρού μεγέθους της βάσης δεδομένων. Από την άλλη, αν και στις περισσότερες περιπτώσεις το σύστημα κάνει σωστές προβλέψεις, το σχετικά μεγάλο ποσοστό loss σημαίνει ότι το μοντέλο δεν είναι τόσο σίγουρο στα predictions του. Δηλαδή μπορεί η μεγαλύτερη πιθανότητα από τις 34 παραγόμενες να αντιστοιχεί στην σωστή κλάση, αλλά αυτή να είναι σχετικά μικρή και οι υπόλοιπες σχετικά μεγάλες, κάτι το οποίο ιδανικά δεν θα θέλαμε να υπάρχει.

Για την οπτικοποίηση και την καλύτερη κατανόηση των αποτελεσμάτων δοκιμής έχει σχεδιαστεί το *block 9*, το οποίο αφού φορτώσει στο δίκτυο το μοντέλο που αξιολογήθηκε, εισάγει σε αυτό το πρώτο test στοιχείο από κάθε κλάση (κάνοντάς τες ταυτόχρονα plot) και τυπώνει όλες τις πιθανότητες εξόδου του συστήματος, όπως και την τελική πρόβλεψη, για κάθε μια από αυτές. Έπειτα, ανάλογα με τις προβλέψεις του συστήματος, τυπώνει το ποσοστό επιτυχών ταξινομήσεων του συγκεκριμένου συνόλου δεδομένων. Στην συγκεκριμένη περίπτωση, όπως φαίνεται και στο output του αντίστοιχου *block*, το ποσοστό επιτυχίας είναι 85.3% (29 στα 34), το οποίο

αναμενόμενα δεν είναι ίσο με 92.7%, αφού εξετάστηκε μόνο ένα πολύ μικρό μέρος του imagedb_test.

2.ε. Αρχιτεκτονική προ-εκπαιδευμένου δικτύου (ResNet50v2)

Για την επίλυση του προβλήματος της πολλαπλής ταξινόμησης μέσω προ-εκπαιδευμένου δικτύου επιλέγεται το ResNet50v2 προ-εκπαιδευμένο στο ImageNet. Το ResNet50v2 είναι ένα δίκτυο το οποίο αποτελείται από 192 layers (μαζί με το επίπεδο εισόδου), δέχεται δεδομένα εισόδου με διαστάσεις 224×224×3 και έχει εκπαιδευτεί για classification της τάξεως των 1000 κατηγοριών (όσες είναι και οι κλάσεις του ImageNet dataset). Στο μήνυμα εξόδου του *μπλοκ 5* του αρχείου *part_B_pretrained.ipynb* φαίνονται οι τύποι όλων των layers του δικτύου πλην των 2 τελευταίων, των οποίων οι τύποι είναι AveragePooling και Softmax αντίστοιχα. Κανονικά, με βάση τον κώδικα του *μπλοκ*, θα έπρεπε να είναι τυπωμένο και το πλήθος των layers του δικτύου, αλλά τα outputs των εκτελέσεων που προβάλλονται στο Jupyter Notebook αρχείο αντιστοιχούν στην αμέσως προηγούμενη έκδοση που δεν εξάγει αυτόν τον αριθμό. Το νέα μηνύματα που θα έπρεπε να τυπώνονται στην νεότερη έκδοση είναι αυτά της *εικόνας 5*.

```
# Freeze the layers except the last 4 layers
for layer in ResNet50V2_conv.layers[:-4]:
    layer.trainable = False

layers_ResNet50v2 = ResNet50V2_conv.layers
print("ResNet50v2 is made of %d layers (2 top layers not included)." % len(layers_ResNet50v2))
print("The layers are:")
print()

# Check the trainable status of the individual layers
for layer in layers_ResNet50v2:
    print(layer, layer.trainable)
```

```
ResNet50v2 is made of 190 layers (2 top layers not included).
The layers are:

<InputLayer name=input_layer, built=True> False
<ZeroPadding2D name=conv1_pad, built=True> False
<Conv2D name=conv1_conv, built=True> False
```

Εικόνα 5: Οι τυπωμένες γραμμές που λείπουν από την έξοδο του block 5 του part_B_pretrained.ipynb

Στην συγκεκριμένη εφαρμογή δεν χρησιμοποιείται το προ-εκπαιδευμένο ResNet50v2 ακριβώς έτσι όπως είναι, αλλά έχουν "παγώσει" τα (προ-εκπαιδευμένα) βάρη των πρώτων 190 layers, δηλαδή παραμένουν σταθερά, και έχουν αντικατασταθεί τα δύο τελευταία επίπεδα από άλλα πέντε, τα οποία με την σειρά είναι:

- Flatten επίπεδο
- FC με 1024 νευρώνες
- ReLU
- Dropout 50%
- Softmax

Το Dropout είναι ένα επίπεδο που χρησιμοποιείται για την αποφυγή του overfitting μηδενίζοντας τυχαία το βάρος των νευρώνων με μία πιθανότητα (50% σε αυτήν την

περίπτωση) για τον καθένα. Ως προς τον ορισμό των παραμέτρων, τόσο το πλήθος των νευρώνων στο πλήρως συνδεδεμένο δίκτυο, όσο και η πιθανότητα απόρριψης του Dropout, επιλέχθηκαν πειραματικά.

Κλείνοντας, οι προαναφερθείσες αλλαγές στο δίκτυο γίνονται με σκοπό την παροχή δυνατότητας fine-tuning του συστήματος, που στην ουσία είναι η περαιτέρω (μερική) εκπαίδευση του μοντέλου, προκειμένου να γίνει πιο ειδικό.

2.στ. Ανάλυση κώδικα / εκπαίδευση προ-εκπαιδευμένου δικτύου (ResNet50v2)

Σχόλιο: Σε αυτό το *κεφάλαιο* υπάρχουν έννοιες και υλοποιήσεις οι οποίες αναλύθηκαν στο *κεφάλαιο 2.γ*. Οπότε, παρόλο που ξανασυναντιούνται και εδώ, δεν επαναλμβάνεται η περιγραφή τους.

Προτού ξεκινήσει η διαδικασία της εκπαίδευσης, πρέπει πρώτα να κατασκευαστεί το δίκτυο του προηγούμενου *κεφαλαίου*. Τα *μπλοκ* που το επιτυγχάνουν αυτό είναι τα *4*, *5* και *6*. Το πρώτο από αυτά φορτώνει το προ-εκπαιδευμένο στο ImageNet ResNet50v2, αφαιρεί τα δύο top επίπεδα και θέτει τις διαστάσεις εισόδου 256×256×3. Τόσο το 256, όσο και τα 3 κανάλια (RGB) επιλέχθηκαν, γιατί οδηγούν σε καλύτερα αποτελέσματα. Όπως αναφέρθηκε στο προηγούμενο *κεφάλαιο*, το *μπλοκ 5*, αφού βέβαια παγώσει τα βάρη όλων των επιπέδων πλην των τεσσάρων τελευταίων, τυπώνει το πλήθος, το είδος και το status όλων των layers του δικτύου. Τέλος, με το με το *μπλοκ 6*, ολοκληρώνεται η δομή του συστήματος προς εκπαίδευση προσθέτοντας τα 5 επιπλέον layers που αναλύθηκαν προηγουμένως.

Στο *μπλοκ 7* πραγματοποιείται η προ-επεξεργασία των δεδομένων προτού εισαχθούν στο σύστημα. Οι διεργασίες που εφαρμόζονται είναι παρόμοιες με εκείνες του IliasNet· διαχωρισμός σε 90% training και 10% validation data, μεταβολή διαστάσεων εικόνων σε target_size=(256,256) και οργάνωση σε batches μεγέθους 20, όλα επειδή οδηγούν σε μεγαλύτερη ακρίβεια. Οι διαφορές αυτής της περίπτωσης με τις προηγούμενης είναι ότι δεν μετατρέπονται σε grayscale οι φωτογραφίες, αφού έχει οριστεί το δίκτυο να δέχεται εικόνες σε RGB, και ότι εφαρμόζεται σε όλο το train και validation database η συνάρτηση προ-επεξεργασίας του ResNet50v2 process_input.

Κλείνοντας, η εκπαίδευση του συστήματος εκτελείται στο *block 8* και έχει ακριβώς τις ίδιες παραμέτρους και callbacks με εκείνες του μη προ-εκπαιδευμένου δικτύου, για ακριβώς τους ίδιους λόγους. Τα μόνα διαφορετικά μεγέθη είναι το patience του tf.keras.callbacks.EarlyStopping() callback, το οποίο θέτεται ίσο με 10, επειδή φτάνει πολύ πιο γρήγορα το σύστημα στην καλύτερη του κατάσταση, και το learning rate, το οποίο σε αυτήν την περίπτωση ορίζεται άμεσα ίσο με 0.0004. Το παραγμένο μοντέλο αποθηκεύεται ως my_model_pre.keras για την μελλοντική χρήση του.

2.στ. Αξιολόγηση προ-εκπαιδευμένου δικτύου (ResNet50v2)

Η διαδικασία της ποσοτικής εκτίμησης της επίδοσης του προ-εκπαιδευμένου δικτύου γίνεται ακριβώς με τον ίδιο τρόπο με εκείνη του μη προ-εκπαιδευμένου. Πρώτα αξιολογείται η ακρίβεια και το σφάλμα του μοντέλου σε όλο το σύνολο της βάσης testing (*μπλοκ 10)* και μετά οπτικοποιείται η επίδοση του, εισάγοντας μέσα σε αυτό το πρώτο στοιχείο από κάθε κλάση του testing (*μπλοκ 11*). Στην πρώτη περίπτωση η ακρίβεια και το σφάλμα του μοντέλου προκύπτουν ίσα με 96% και 16.6% αντίστοιχα, μεγέθη σαφώς πολύ καλύτερα συγκριτικά με εκείνα του IliasNet. Επομένως, όχι μόνο το ResNet50v2 κάνει ορθότερη ταξινόμηση των testing εικόνων (277 στις 289), αλλά είναι αρκετά πιο σίγουρο στην πρόβλεψη του και ικανότερο στην διαφοροποίηση των κλάσεων μεταξύ τους. Στην δεύτερη αξιολόγηση, το σύστημα απέδωσε το ίδιο καλά με το IliasNet, καταφέρνοντας να ταξινομήσει σωστά 30 από τις 34 φωτογραφίες (καλύτερο από το μη προ-εκπαιδευμένο κατά ένα classification) επιτυγχάνοντας έτσι ένα ποσοστό επιτυχίας της τάξεως του 88.2%.

Σχόλιο: Παρόλο που τα ποσοστά των εξόδων της Softmax έχουν οριστεί να προβάλλονται με ακρίβεια τριών δεκαδικών, στα εκτυπωμένα μηνύματα φαίνεται τα αντίστοιχα μεγέθη να έχουν ακρίβεια ενός δεκαδικού ψηφίου. Αυτό συμβαίνει γιατί τα αποτελέσματα που προβάλλονται αντιστοιχούν στην αμέσως προηγούμενη έκδοση. Ποιοτικά δεν αλλάζει κάτι, απλώς η στρογγυλοποίηση στην ακρίβεια ενός δεκαδικού δυσχεραίνει λίγο την οπτικοποίηση του σφάλματος.

3. Μέρος Γ: Ανίχνευση αντικειμένων

Για το τρίτο μέρος, δεν υπάρχει ανάλυση, λόγω της ανεπιτυχής επίλυσης του προβλήματος. Οι κώδικες της εφαρμογής αυτής έχουν γραφθεί (στο μεγαλύτερο μέρος τους) και μάλιστα είναι τα επισυνημμένα αρχεία *part_C_f_RCNN.ipynb* και *part_C_YOLOv3*, αλλά δεν καταφέρνουν να επιλύσουν το πρόβλημα της ανίχνευσης αντικειμένων.

Στην περίπτωση του Faster-RCNN, υπάρχει εκτέλεση χωρίς μηνύματα σφάλματος και γίνεται η εκπαίδευση κανονικά, όμως δεν υπάρχει πρόοδος. Οποιαδήποτε τιμή και να έχουν οι παράμετροι της εκπαίδευσης, το μοντέλο καταφέρνει να ανιχνεύει 0 αντικείμενα καθόλη την διάρκεια της εκπαίδευσης. Η συμπεριφορά αυτή, ή οφείλεται στον λάθος ορισμό των labels για την βάση δεδομένων TrainIJCNN2013, ή οφείλεται στα αυστηρά κριτήρια object detection, π.χ. μεγάλο confidence threshold, δεδομένου ότι σε μία μόνο υλοποίηση υπήρχαν κάποια στιγμή συνολικά 2 ανιχνεύσεις καθόλη την διάρκεια του training.

Στην περίπτωση του YOLOv3, όσα μικρές και να είναι οι παράμετροι της εκπαίδευσης, η εκτέλεση επιστρέφει ότι έχει τελειώσει η μνήμη της CUDA. Υπάρχει μια πιθανότητα να ευθύνεται πάλι ο ορισμός ετικετών ή/και το αρχείο data.yaml που χρησιμοποιείται, λαμβάνοντας υπόψιν κάποια συγκεκριμένα μηνύματα σφάλματος που τυπώνονται στην πρώτη κάθε φορά εκτέλεση του *μπλοκ εκπαίδευσης.*