Άσκηση 1
Θέματα Όρασης Υπολογιστών

Περιεχόμενα

Ο κώδικας μπορεί να βρεθεί εδώ:
https://github.com/GrigorisTzortzakis/Computer-Vision/tree/main/Exercise%201

Οι κυρίες υλοποιήσεις για τα ερωτήματα 1-3 έχουν γίνει σε matlab όπως και ζητείτε, ενώ το ερώτημα 4 είναι σε python σε περιβάλλον google collab. Τα ερωτήματα 1-3 έχουν γίνει επίσης και με python, αλλά με διαφορετικό πυρήνα από ότι ζητείτε.

Διαδικασία 1

Αρχικά, πρώτο βήμα αποτελεί η κατασκευή της Γκαουσιανής πυραμίδας της μάσκας m1. Η μάσκα αυτή ορίζει σε ποιες περιοχές κυριαρχεί η πρώτη εικόνα και σε ποιες η δεύτερη. Επειδή μας ενδιαφέρει η σταδιακή μετάβαση μεταξύ των δύο εικόνων, θέλουμε η μάσκα να γίνει λιγότερο απότομη σε υψηλές κλίμακες. Για τον σκοπό αυτόν, εφαρμόζουμε έναν χαμηλοπερατό πυρήνα (Gaussian) και στη συνέχεια υποδειγματοληψία (downsampling) επαναληπτικά, ώστε να δημιουργήσουμε τα L+1 επίπεδα της πυραμίδας {g0(n),g1(n),...,gL(n)}. Σε κάθε επίπεδο, η ανάλυση ελαττώνεται και η μάσκα θολώνει περισσότερο, το οποίο διευκολύνει σημαντικά την ομαλή ένωση των εικόνων.

Συνεχίζοντας, δημιουργούμε τις Λαπλασιανές πυραμίδες για καθεμιά από τις δύο αρχικές εικόνες, I1 και I2. Ξεκινάμε κατασκευάζοντας αρχικά τη Γκαουσιανή πυραμίδα κάθε εικόνας με τα ίδια επίπεδα L+1. Στη συνέχεια, στο επίπεδο iii (όπου i=0,...,L−1), υπολογίζουμε τη Λαπλασιανή συνιστώσα αφαιρώντας από τη Γκαουσιανή εκδοχή του ίδιου επιπέδου τη μεγεθυμένη (upsampling) εκδοχή του επιπέδου i+1. Οι διαφορές αυτές περιέχουν τις υψηλές συχνότητες της αρχικής εικόνας, δηλαδή τις λεπτομέρειες, τις ακμές και τις γρήγορες μεταβολές φωτεινότητας και χρώματος. Έτσι, η πυραμίδα κάθε εικόνας περιλαμβάνει L διαφορικές εικόνες και ένα τελικό Γκαουσιανό επίπεδο.

Παρακάτω, δημιουργούμε την πυραμίδα B που θα εμπεριέχει το μείγμα των δύο εικόνων σε κάθε κλίμακα. Για τα επίπεδα i=0,...,L−1 ορίζουμε $b_i(n) = g_i(n)l_{1,i}(n) + \big(1 - g_i(n)\big)l_{2,i}(n)$, όπου $g_i(n)$ είναι η τιμή της μάσκας στη συγκεκριμένη κλίμακα (επίπεδο i). Έτσι, σε κάθε κλίμακα, όπου η μάσκα είναι κοντά στο 1, υπερισχύει η Λαπλασιανή συνιστώσα της πρώτης εικόνας ενώ όπου η μάσκα τείνει στο 0, κυριαρχεί εκείνη της δεύτερης. Στην ουσία, αυτή η διαδικασία σε κάθε επίπεδο της πυραμίδας μάς δίνει μια συνεχόμενη μετάβαση, αποφεύγοντας απότομες διαφοροποιήσεις φωτεινότητας ή χρώματος.

Τέλος, κάνουμε ανακατασκευή της τελικής εικόνας. Έχουμε πλέον την πυραμίδα B, άρα κάνουμε την αντίστροφη πυραμίδα, δηλαδή τη διαδοχική μεγέθυνση (upsampling) κάθε επιπέδου και πρόσθεσή του με το αμέσως ανώτερο.

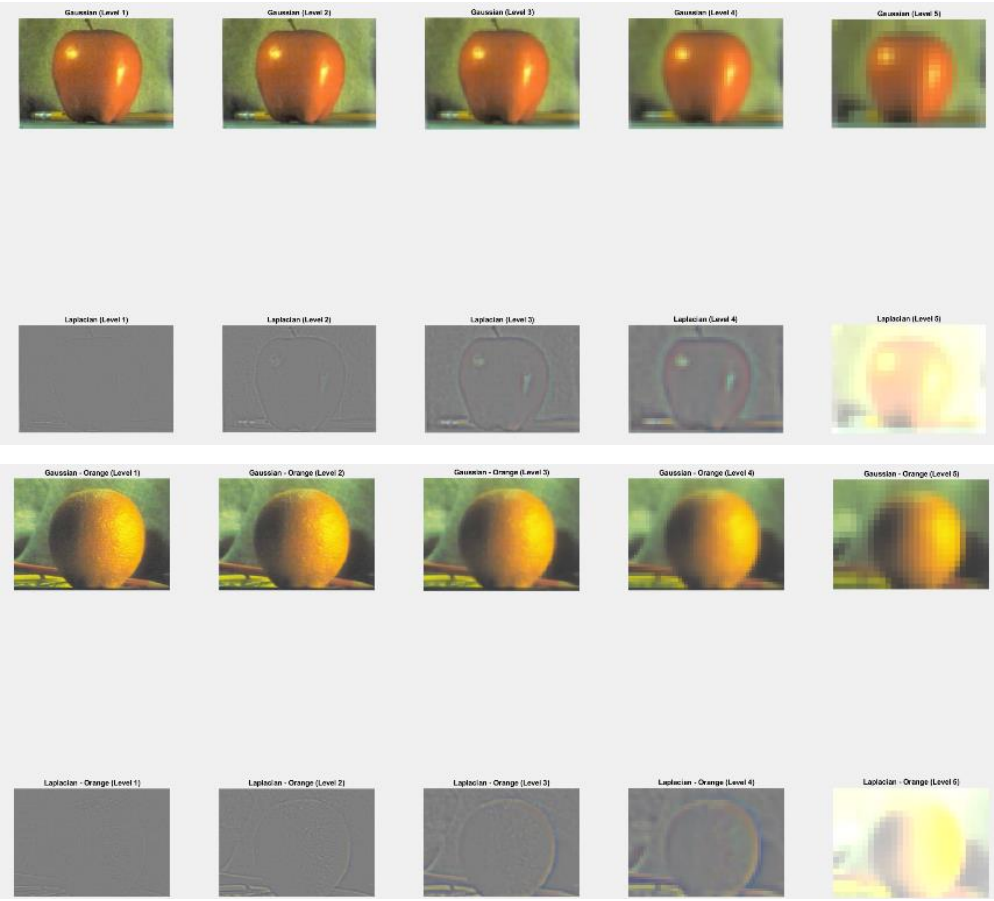
Για το κομμάτι υλοποίησης, ξεκινάμε με τη φόρτωση των δύο αρχικών εικόνων και την διαμόρφωση τους σε κοινές διαστάσεις ώστε να γίνει σωστά η διαδικασία. Στην συνέχεια, ορίζουμε τον αριθμό επίπεδων της πυραμίδας (numLevels = 5) και την μάσκα που προσδιορίσει ποιο τμήμα έρχεται από την πρώτη εικόνα και ποιο από την δεύτερη.

Παρακάτω, χρησιμοποιούμε τη genPyr για να φτιάξει τις Λαπλασιανές πυραμίδες των εικόνων, όπου εσωτερικά ενεργοποιούνται οι pyr_reduce και pyr_expand για τη μείωση και επαναφορά της ανάλυσης, αντίστοιχα. Παρομοίως, η ίδια genPyr χρησιμοποιείται με παράμετρο 'gauss' προκειμένου να δημιουργήσει τη Γκαουσιανή πυραμίδα της μάσκας, εξασφαλίζοντας ότι σε κάθε κλίμακα θα έχουμε μια αντίστοιχα πιο θολή εκδοχή της. Κάθε επίπεδο υπολογίζει ένα blendPyr{i} χρησιμοποιώντας maskPyr{i} (την τιμή της μάσκας στη συγκεκριμένη κλίμακα) και τις Λαπλασιανές συνιστώσες των εικόνων

(lapApple{i}, lapOrange{i}). Όσο μεγαλύτερη είναι η τιμή της μάσκας σε ένα σημείο, τόσο περισσότερο συμβάλλει η πρώτη εικόνα και αντιστρόφως όταν η μάσκα είναι μικρή, κυριαρχεί η δεύτερη εικόνα.

Τέλος, η pyrReconstruct κάνει την αντίστροφη πυραμίδα. Ξεκινώντας από το τελευταίο και πιο συμπιεσμένο επίπεδο, ανεβάζει (upsample) κάθε φορά την εικόνα με pyr_expand και προσθέτει τις υψηλές συχνότητες που περιέχονται στη Λαπλασιανή συνιστώσα του αμέσως επόμενου επιπέδου. Έτσι, ανακτάται σταδιακά η πλήρης ανάλυση της εικόνας, μόνο που πλέον αυτή η εικόνα είναι το ομαλό μείγμα των δύο αρχικών.

Παρακάτω, βλέπουμε τα αποτελέσματα του κώδικα.



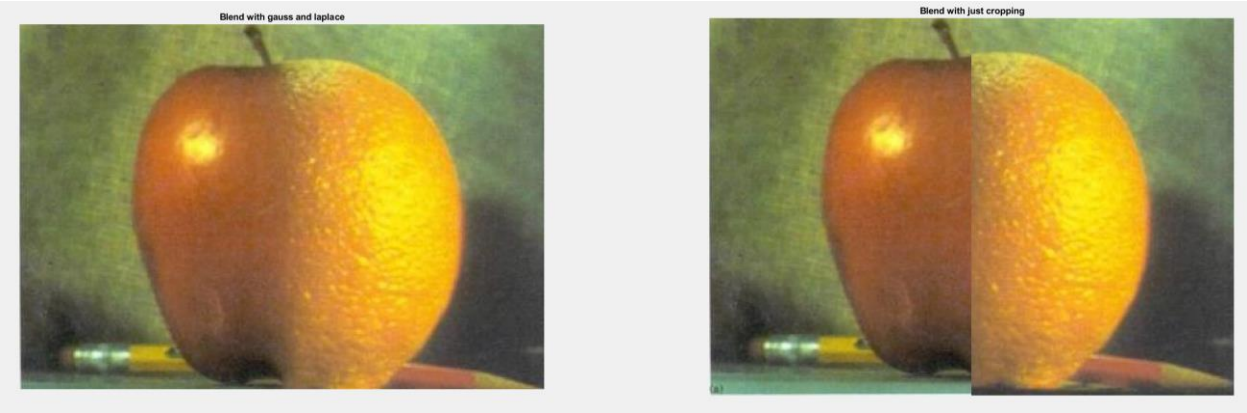
Όπως φαίνεται στις εικόνες, στο πρώτο επίπεδο (Gaussian Level 1) η εικόνα είναι ακόμη σχετικά καθαρή, διατηρώντας την πλειονότητα των λεπτομερειών της αρχικής μορφής. Στο δεύτερο επίπεδο (Gaussian Level 2) η εικόνα γίνεται λίγο πιο θολή και εμφανώς μικρότερη σε ανάλυση, μιας και έχει απορριφθεί ένα ποσοστό των υψηλών συχνοτήτων και των pixel κατά την υποδειγματοληψία. Όσο ανεβαίνουμε στα επόμενα επίπεδα (Gaussian Level 3, 4, 5), παρατηρούμε όλο και πιο θολές όψεις του μήλου και του πορτοκαλιού. Οι βασικές περιοχές φωτεινότητας και σκίασης παραμένουν, αλλά οι λεπτομέρειες χάνονται σταδιακά. Στο υψηλότερο επίπεδο (Level 5) διακρίνουμε

ουσιαστικά μόνο την ευρύτερη μορφή του αντικειμένου (σχήμα, χρώμα), αφού όλες σχεδόν οι μικρές διακυμάνσεις έχουν απορροφηθεί από τη διαδικασία φιλτραρίσματος και υποδειγματοληψίας.

Από την άλλη, η Λαπλασιανή πυραμίδα εστιάζει στις λεπτομέρειες που αφαιρούνται από τη μία κλίμακα της πυραμίδας στην επόμενη. Στις εικόνες (Laplacian Level 1, 2, 3, 4, 5), βλέπουμε ότι κάθε επίπεδο εμφανίζει εντονότερα τις ακμές, τις απότομες μεταβολές χρώματος και φωτεινότητας. Στο Laplacian Level 1, καθώς βρισκόμαστε κοντά στην πλήρη ανάλυση, μπορούμε να διακρίνουμε τα περιγράμματα αρκετά καθαρά. Στα επόμενα επίπεδα η εικόνα γίνεται όλο και πιο θολή, εστιάζοντας σε όλο και πιο μεγάλες αλλαγές φωτεινότητας, αφού οι λεπτομέρειες έχουν ήδη αφαιρεθεί στο ανώτερο επίπεδο. Τέλος, στο Level 5 παρατηρούμε μια αρκετά θολή αποτύπωση του αντικειμένου, η οποία προκύπτει από τη διαφορά μεταξύ των πολύ χαμηλών συχνοτήτων (που συνιστούν τη μικρότερη Γκαουσιανή εικόνα) και της επαναφοράς αυτής της εικόνας με upsampling.

Συμπερασματικά, η Γκαουσιανή πυραμίδα μάς δείχνει την εξέλιξη της εικόνας σε κλίμακες θόλωσης, εστιάζοντας στις αργές μεταβολές και στη γενική μορφή του αντικειμένου. Αντίθετα, η Λαπλασιανή πυραμίδα μάς επιτρέπει να οπτικοποιήσουμε τις λεπτομέρειες και τις ακμές που χάνονται από το ένα επίπεδο Γκαουσιανής στο επόμενο.

Το τελικό αποτέλεσμα του κώδικα μας είναι αυτό:



Στην εικόνα αριστερά, όπου εφαρμόστηκε η ανάμειξη με Γκαουσιανές και Λαπλασιανές πυραμίδες, το σημείο ένωσης ανάμεσα στο μήλο και το πορτοκάλι είναι εξαιρετικά ομαλό και δεν παρουσιάζει κάποιον έντονο διαχωρισμό. Αυτό συμβαίνει επειδή, σε κάθε κλίμακα της Λαπλασιανής πυραμίδας, η μάσκα θολώνει σταδιακά τα όρια, με αποτέλεσμα το ράψιμο των λεπτομερειών από τις δύο εικόνες να γίνεται χωρίς μεγάλες μεταβάσεις. Έτσι, οι ακμές και οι διαφορές φωτεινότητας που θα μπορούσαν να δείχνουν τη συγχώνευση εξομαλύνονται σε όλα τα επίπεδα της πυραμίδας.

Αντίθετα, στη δεξιά εικόνα, η οποία προκύπτει από μια απλή μέθοδο cropping, παρατηρείται μια ξεκάθαρη γραμμή διαχωρισμού ανάμεσα στα δύο φρούτα.

Διαδικασία 2

Για να ενώσουμε δύο εικόνες με ομαλό και φυσικό τρόπο, ορίζουμε μάσκες που δείχνουν σε ποια σημεία θα επικρατήσει η πρώτη εικόνα και σε ποια η δεύτερη. Δημιουργούμε μια δισδιάστατη μάσκα m1(n) (ίδιας διάστασης με τις εικόνες), όπου αρχικά όλες οι τιμές είναι μηδενικές. Στη συνέχεια επιλέγουμε μια ορθογώνια περιοχή που αντιστοιχεί στο μάτι της γυναίκας και εκεί αναθέτουμε τη τιμή 1. Αυτό σημαίνει ότι μέσα σε αυτή την περιοχή θα ισχύει η εικόνα της γυναίκας, ενώ εκτός περιοχής (όπου m1(n)=0) θα υπερισχύει η εικόνα του χεριού. Για να διατηρηθεί η σωστή λογική του blending, ορίζεται παράλληλα m2(n)=1−m1(n), έτσι ώστε παντού να ισχύει m1(n)+m2(n)=1.

Η κρίσιμη θεωρητική αρχή είναι ότι αν μια μάσκα είναι 1 σε κάποια ζώνη και 0 στην υπόλοιπη, το πέρασμα μεταξύ των δύο εικόνων μπορεί να είναι απότομο και να δημιουργήσει εμφανείς αλλαγές. Για να το αποφύγουμε, φτιάχνουμε μία Γκαουσιανή Πυραμίδα της μάσκας (δηλαδή εξομαλύνουμε σταδιακά τη μετάβαση μεταξύ 1 και 0 σε πολλά επίπεδα). Στα αντίστοιχα επίπεδα της Λαπλασιανής Πυραμίδας των δύο εικόνων, το βάρος (weight) καθορίζεται από την τιμή της μάσκας σε εκείνη την κλίμακα. Έτσι, σε περιοχές που η μάσκα είναι κοντά στο 1, κυριαρχεί η πρώτη εικόνα, ενώ σε περιοχές κοντά στο 0, κυριαρχεί η δεύτερη. Στα ενδιάμεσα (π.χ. 0.4 ή 0.5), οι δύο εικόνες αναμειγνύονται αναλογικά.

Στον κώδικα, ορίζουμε να ορθογώνιο παράθυρο με τη συνάρτηση round([rows*0.3 rows*0.7 cols*0.3 cols*0.7]) που αντιστοιχεί περίπου στη ζώνη του ματιού και εκεί βάζουμε m1=1. Έπειτα, η μάσκα ομαλοποιείται στα διάφορα επίπεδα της πυραμίδας και για να ολοκληρωθεί η διαδικασία κάνουμε ανακατασκευή και επιστρέφουμε στο αρχικό μέγεθος της εικόνας.

Αναλυτικότερα, ας δούμε συνοπτικά τις επιλογές που κάναμε για τον κώδικα:

Προσαρμόζουμε τις εικόνες ώστε να έχουν ιδίες διαστάσεις και στη συνέχεια, με τις εντολές genPyr(m1, 'gauss', numLevels), genPyr(woman, 'lap', numLevels) και genPyr(hand, 'lap', numLevels), δημιουργούνται οι πυραμίδες. Η μάσκα παράγεται ως Γκαουσιανή πυραμίδα (για να διατηρηθεί μια προοδευτική θόλωση ανά επίπεδο), ενώ οι δύο εικόνες γίνονται Λαπλασιανές πυραμίδες (για να χωριστούν οι χωρικές συχνότητες σε πολλά επίπεδα λεπτομέρειας). Προσθέτοντας, διατρέχουμε κάθε επίπεδο της πυραμίδας. Δηλαδή, γίνεται imresize(Gm1{j}, [levelRows, levelCols]) ώστε να αντιστοιχίσουμε σωστά το επίπεδο της μάσκας στη χωρική διάσταση των Λαπλασιανών επιπέδων και συνδυάζουμε γραμμικά τις δύο εικόνες στο στο B{j} = gj .* Ll1{j} + (1 - gj) .* Ll2{j}.

Διαδικασία 3

Αρχικά, πρέπει να ορίσουμε τις μάσκες για καθεμιά από τις 6 εικόνες. Άρα, κάθε μάσκα mk αντιστοιχεί σε μία εικόνα και όπου mk(n)=1 υποδηλώνει ότι το συγκεκριμένο σημείο (pixel) της τελικής σύνθεσης θα προέρχεται από την k-οστή εικόνα ενώ αντίστοιχα, αν $\Sigma_{k=1}^{6} m_k(n) = 1$, διασφαλίζουμε ότι σε κάθε θέση η συμμετοχή αθροίζεται στο 1.

Η ακριβής μορφή των μασκών εξαρτάται από την λειτουργική διάταξη που θέλουμε να δώσουμε στις εικόνες. Αν, για παράδειγμα, επιθυμούμε να τοποθετήσουμε την εικόνα “P200” στο αριστερό τμήμα της σύνθεσης και μια δεύτερη εικόνα “dog1” να εμφανίζεται σε κάποια κεντρική περιοχή, σχεδιάζουμε κατάλληλα m1 και m2 αντίστοιχα, με τιμές 1 στις περιοχές που θα καταλαμβάνει η κάθε εικόνα και 0 οπουδήποτε αλλού.

Μάλιστα, αν θέλουμε να ξεχειλίζουν ή να αλληλεπικαλύπτονται οι εικόνες, ορίζουμε τις μάσκες να έχουν κοινά τμήματα όπου και οι δύο μάσκες λαμβάνουν τιμές μεγαλύτερες από το μηδέν. Το κρίσιμο σημείο της διαδικασίας είναι η κανονικοποίηση όλων των μασκών, ώστε να πληρείται η συνθήκη (το αθροισμα που γραψαμε παραπανω) δηλαδή σε κάθε πίξελ της εικόνας το άθροισμα των τιμών όλων των μασκών να ισούται με 1.

Για την περίπτωση μας, αθροίζουμε τις τιμές όλων των μασκών (παίρνοντας τον πίνακα maskSum=m1+m2+⋯+m6) και μετά διαιρούμε κάθε μάσκα mk. Έτσι, οπουδήποτε οι μάσκες επικαλύπτονται, οι τιμές τους μετατρέπονται σε κλάσματα που αθροίζουν στη μονάδα. Οι μάσκες μας είναι αρχικά γεμάτες με 0 και καθορίσαμε σε καθεμία ένα συγκεκριμένο ορθογώνιο τμήμα όπου λάμβαναν την τιμή 1.

Η διαδικασία ορισμού και κατασκευής των Γκαουσιανών πυραμίδων G_{mk} για κάθε μία από τις μάσκες mk (k=1,2,…,5) αποτελεί ένα βασικό βήμα στην πολυκλιμακική προσέγγιση της ανάμειξης (blending). Κάθε πυραμίδα περιλαμβάνει L+1 επίπεδα, όπου το καθένα είναι ουσιαστικά μια θολωμένη (blurred) και υποδειγματοληπτημένη εκδοχή της μάσκας mk. Στη χαμηλότερη

κλίμακα ($g_{k,0}$) αποθηκεύουμε την αρχική μάσκα υψηλής ανάλυσης. Κατόπιν, για κάθε επόμενο επίπεδο $g_{k,i}$ εφαρμόζεται φίλτρο χαμηλοπερατής φύσης και στη συνέχεια γίνεται μείωση διαστάσεων κατά δύο. Αυτό έχει ως αποτέλεσμα οι απότομες μεταβάσεις στην αρχική μάσκα (0 προς 1) να εξομαλύνονται.

Για να δημιουργηθούν οι Λαπλασιανές πυραμίδες {LIk} για τις έξι διαφορετικές εικόνες Ik, χρησιμοποιείται η ίδια αρχή, δηλαδή ξεκινάμε από την εικόνα στην πλήρη ανάλυσή της και δημιουργούμε τη Γκαουσιανή πυραμίδα εσωτερικά, φιλτράροντάς την διαδοχικά και υποδειγματοληπτώντας την. Στη συνέχεια, για κάθε επίπεδο (εκτός από το τελευταίο) αφαιρούμε (από την τρέχουσα Γκαουσιανή) την reconstructed εκδοχή του επόμενου επιπέδου, ώστε να απομονώσουμε τις λεπτομέρειες και ακμές.

Επιπλέον, για την πυραμίδα Β λαμβάνονται τα επίπεδα $\{l_{1,i}(n), l_{2,i}(n), \dots, l_{6,i}(n)\}$ από τις Λαπλασιανές πυραμίδες όλων των εικόνων και σταθμίζονται μέσω των τιμών από τα αντίστοιχα επίπεδα των Γκαουσιανών πυραμίδων των μασκών. Συγκεκριμένα, σε κάθε κλίμακα i συνδυάζονται τα λαπλασιανά επίπεδα σύμφωνα με την σχέση $b_i(n) = \Sigma_\kappa(g_{k,i}(n)l_{k,i}(n))$ όπου το g είναι η θολή μάσκα της κλίμακας i για την k-οστή εικόνα, κανονικοποιημένη έτσι ώστε το συνολικό άθροισμα να παραμένει 1. Ως τελευταία συνιστώσα χρησιμοποιείται μια συγχώνευση των πολύ χαμηλών συχνοτήτων των εικόνων, πάλι βάση της μάσκας.

Διαδικασία 4

Στο πρόβλημα της κατάτμησης εικόνων (image segmentation) καλούμαστε να προβλέψουμε για κάθε pixel σε ποια κλάση ανήκει (π.χ. δρόμος, πεζός, αυτοκίνητο). Για να εκτιμήσουμε πόσο αποτελεσματικό είναι ένα μοντέλο που εκτελεί κατάτμηση, χρειάζεται να ορίσουμε μια μετρική συνάρτηση η οποία θα αντικατοπτρίζει πόσο καλά ανταποκρίνονται οι προβλέψεις του μοντέλου στις πραγματικές επισημάνσεις (ground truth). Μια καλή μετρική για την

κατάτμηση είναι η Intersection over Union (IoU). Για κάθε κλάση, συγκρίνουμε το σύνολο των pixels που προβλέφθηκε πως ανήκουν σε αυτήν με το σύνολο των pixels που όντως σε αυτήν και υπολογίζουμε τον λόγο του κοινού μέρους τους (intersection) προς την ένωση (union). Ουσιαστικά, το IoU μας δείχνει πόσο επικαλύπτονται οι προβλέψεις του μοντέλου με το ground truth σε κάθε κλάση.

Άρα στον κώδικα θέλουμε την εξής διαδικασία:

1. Για κάθε εικόνα του συνόλου επικύρωσης, εξετάζουμε την πρόβλεψη του μοντέλου και τη συγκρίνουμε με το ground truth. Για κάθε κλάση (cls), προκύπτει ένα IoU που δείχνει πόσο καλά εντοπίστηκε η συγκεκριμένη κλάση στην εικόνα. Έτσι, σε όλες τις εικόνες του συνόλου επικύρωσης, συλλέγονται πολλές τιμές αυτής της μετρικής για την ίδια κλάση.

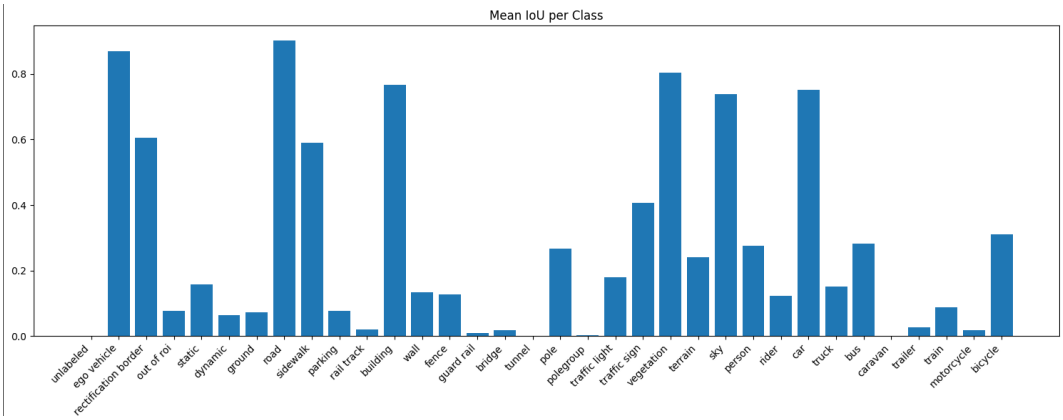
2. Έπειτα, για κάθε κλάση χωριστά, λαμβάνουμε τον μέσο όρο των τιμών που προέκυψαν από όλες τις εικόνες και υπολογίζουμε επίσης τη διασπορά. Η μέση τιμή μάς δείχνει πόσο καλά τα πηγαίνει το μοντέλο συνολικά σε εκείνη την κλάση και η διασπορά μάς αποκαλύπτει αν η επίδοση έχει μεγάλες διακυμάνσεις ανάλογα με την εκάστοτε εικόνα (π.χ. ίσως μια κλάση να ανιχνεύεται εύκολα σε κάποιες συνθήκες αλλά δυσκολότερα σε άλλες).

3. Αφού έχουμε υπολογίσει τις τιμές της μετρικής για κάθε εικόνα (π.χ. τον μέσο όρο όλων των κλάσεων στην ίδια εικόνα), προκύπτει η συνολική επιδόση του μοντέλου σε αυτήν. Συνεχίζοντας, παίρνουμε τον μέσο όρο αυτών των επιδόσεων από όλες τις εικόνες του συνόλου, για να έχουμε μια συνολική εκτίμηση της ποιότητας του μοντέλου. Ταυτόχρονα, υπολογίζουμε και τη διασπορά των επιδόσεων ανά εικόνα, ώστε να διαπιστωθεί κατά πόσο το μοντέλο αποδίδει σταθερά καλά ή αν υπάρχουν κάποιες εικόνες στις οποίες η κατάτμηση δεν δουλεύει.

Συνοπτικά, παίρνουμε τον κώδικα που δίνεται στην εκφώνηση και προσθέτουμε την συνάρτηση compute_iou που υπολογίζει τις επιδόσεις ανά κλάση σε κάθε εικόνα και τις αποθηκεύει στο per_class_ious για την ανάλυση ανά κλάση, ενώ υπολογίζουμε και έναν συνολικό (average) δείκτη για την ίδια την εικόνα, τον οποίο συγκεντρώνουμε στο image_scores.

Ας δούμε τα αποτελέσματα του psp net:

```
Per-class IoU statistics:
unlabeled            : Mean IoU = 0.0000, Variance = 0.0000
ego vehicle          : Mean IoU = 0.8699, Variance = 0.0046
rectification border: Mean IoU = 0.6056, Variance = 0.0582
out of roi           : Mean IoU = 0.0774, Variance = 0.0035
static               : Mean IoU = 0.1583, Variance = 0.0297
dynamic              : Mean IoU = 0.0639, Variance = 0.0215
ground               : Mean IoU = 0.0730, Variance = 0.0322
road                 : Mean IoU = 0.9023, Variance = 0.0371
sidewalk             : Mean IoU = 0.5902, Variance = 0.0878
parking              : Mean IoU = 0.0778, Variance = 0.0291
rail track           : Mean IoU = 0.0203, Variance = 0.0078
building             : Mean IoU = 0.7679, Variance = 0.0317
wall                 : Mean IoU = 0.1333, Variance = 0.0550
fence                : Mean IoU = 0.1261, Variance = 0.0497
guard rail           : Mean IoU = 0.0086, Variance = 0.0001
bridge               : Mean IoU = 0.0185, Variance = 0.0057
tunnel               : Mean IoU = 0.0000, Variance = 0.0000
pole                 : Mean IoU = 0.2677, Variance = 0.0262
polegroup            : Mean IoU = 0.0036, Variance = 0.0004
traffic light        : Mean IoU = 0.1792, Variance = 0.0514
traffic sign         : Mean IoU = 0.4077, Variance = 0.0745
vegetation           : Mean IoU = 0.8035, Variance = 0.0279
terrain              : Mean IoU = 0.2403, Variance = 0.0906
sky                  : Mean IoU = 0.7381, Variance = 0.0842
person               : Mean IoU = 0.2757, Variance = 0.0660
rider                : Mean IoU = 0.1230, Variance = 0.0311
car                  : Mean IoU = 0.7511, Variance = 0.0626
truck                : Mean IoU = 0.1518, Variance = 0.0834
bus                  : Mean IoU = 0.2831, Variance = 0.1421
caravan              : Mean IoU = 0.0000, Variance = 0.0000
trailer              : Mean IoU = 0.0269, Variance = 0.0195
train                : Mean IoU = 0.0871, Variance = 0.0542
motorcycle           : Mean IoU = 0.0170, Variance = 0.0049
bicycle              : Mean IoU = 0.3097, Variance = 0.0708
license plate        : Mean IoU = nan, Variance = nan
```



Βλέπουμε ότι η Mean IoU ανά εικόνα ανέρχεται σε περίπου 0.4028, με σχετικά χαμηλή διασπορά (~0.0056), στοιχείο που υποδεικνύει ότι το μοντέλο επιτυγχάνει μια μετρίως ικανοποιητική απόδοση στο σύνολο των αστικών σκηνών που περιλαμβάνει το σετ επικύρωσης. Περνώντας σε κάθε κλάση ξεχωριστά, παρατηρούμε σημαντικές αποκλίσεις, αφού ορισμένες κλάσεις, όπως π.χ. το ego vehicle, road και το vegetation, φτάνουν σε υψηλά ποσοστά αναγνώρισης (άνω του 0.75 σε IoU). Αυτό σημαίνει ότι το PSPNet εντοπίζει με συνέπεια μεγάλα, συχνά εμφανιζόμενα ή εύκολα διακριτά τμήματα της σκηνής.

Σε άλλες κλάσεις, όπως τα “parking”, “fence”, “traffic light” ή “caravan”, το IoU είναι αρκετά χαμηλότερο. Αυτό γίνεται επειδή πρόκειται για σπανιότερα αντικείμενα στις σκηνές του εκπαιδευτικού συνόλου (οπότε το δίκτυο δεν έχει επαρκώς πολλές παραδείγματα για να μάθει) και επειδή αφορούν αντικείμενα με μεγάλη ποικιλία σε χρώμα/σχήμα (π.χ. πινακίδες, μικρά τροχόσπιτα, κλπ.), όπου το δίκτυο δυσκολεύεται να διατηρήσει συνέπεια. Σε ορισμένες περιπτώσεις, οι κλάσεις αυτές ίσως καταλαμβάνουν ελάχιστα pixel (είναι πολύ μικρά), με αποτέλεσμα να μη προλαβαίνει το PSPNet να αναγνωρίσει επαρκώς τις λεπτομέρειές τους.

Επίσης, σε κλάσεις όπως “license plate” εμφανίζεται μηδενικό ή μη ορισμένο (NaN) IoU. Αυτό υποδηλώνει ότι το δίκτυο είτε δεν συναντά καθόλου τέτοια ετικέτα στις σκηνές ή ότι οι πινακίδες είναι τόσο μικροσκοπικές που η πρόβλεψη δεν κατορθώνει να τις αναγνωρίσει.

Παρατηρώντας τη διασπορά (variance) στις κλάσεις, βλέπουμε ότι κλάσεις με πολύ χαμηλές τιμές IoU τείνουν να έχουν μικρή διασπορά (π.χ. “bridge”, “tunnel”), κάτι που σημαίνει ότι το μοντέλο σχεδόν πάντα αποτυγχάνει να τις αναγνωρίσει, ενώ κλάσεις με μεγαλύτερο εύρος επιδόσεων (π.χ. “traffic sign” ή

“bicycle”) παρουσιάζουν υψηλότερη διασπορά, ένδειξη ότι το δίκτυο κάποιες φορές τις αναγνωρίζει.

Συμπερασματικά, το ~0.40 στην Mean IoU φανερώνει ότι το PSPNet μαθαίνει αρκετά καλά τις βασικές, ευδιάκριτες κλάσεις (π.χ. δρόμο, ουρανό, όχημα) ωστόσο, υπάρχει χώρος για βελτίωση στις πιο απαιτητικές ή σπάνιες κλάσεις (π.χ. μικρά αντικείμενα, οριακά διακριτές περιοχές). Η χαμηλή διασπορά στα per-image αποτελέσματα δηλώνει ότι κατά μέσο όρο, η απόδοση του μοντέλου είναι σχετικά σταθερή σε όλες τις εικόνες. Από το paper που δίνεται στην εκφώνηση (https://openaccess.thecvf.com/content_cvpr_2017/papers/Zhao_Pyramid_Scene_Parsing_CVPR_2017_paper.pdf)

βλέπουμε ότι τα αποτελέσματα μας είναι λογικά εφόσον δεν έχει γίνει η βέλτιστη εκπαίδευση στο μοντέλο. Αυτό πετυχαίνει απόδοση 40% και αυξάνεται όταν εφαρμόζει κάποιες βελτιστοποιήσεις στο μοντέλο.