

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΡΓΑΣΤΗΡΙΟ ΕΠΕΞΕΡΓΑΣΙΑΣ ΣΗΜΑΤΩΝ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΘΕΜΑΤΑ
ΟΡΑΣΗΣ ΥΠΟΛΟΓΙΣΤΩΝ & ΓΡΑΦΙΚΗΣ

ΓΚΑΟΥΣΙΑΝΕΣ & ΛΑΠΛΑΣΙΑΝΕΣ ΠΥΡΑΜΙΔΕΣ

Διδάσκων: Αναπλ. Καθηγητής Εμμανουήλ Ζ. Ψαράκης
Επικουρικό έργο: Παναγιώτης Γεωργαντόπουλος, Παναγιώτης Κάτσος

Πάτρα Δεκέμβριος 2021

ΣΤΟΙΧΕΙΩΔΗΣ ΘΕΩΡΙΑ

Σκοπός της άσκησης αυτής είναι η ανάπτυξη αναπαραστάσεων αποσύνθεσης εικόνων σε πολλαπλές κλίμακες με χρήση φίλτρων, με σκόπο:

- την αποθορυβοποίηση
- τον εντοπισμό και την εξαγωγή χαρακτηριστικών και/ή δομών για την αποδοτική:
 1. κωδικοποίηση [1], [2]
 2. συμπίεση [3], [4]
 3. βελτίωση [5], [6], [7] και
 4. σύνθεση και συρραφή εικόνων [8], [9].

Συγκεκριμένα, ο σκοπός της χρήσης των Λαπλασιανών πυραμίδων στο πλαίσιο της άσκησης αυτής θα είναι η συρραφή δύο ή περισσοτέρων εικόνων σε μωσαϊκό να γίνεται με τρόπο ώστε η ένωση τους να είναι σχεδόν φυσική και κατά συνέπεια κατά μία έννοια αόρατη. Η παραπάνω διαδικασία προϋποθέτει:

1. την αντιστοίχιση των εικόνων, και
2. την θόλωση της ένωσης τους

εξομαλύνοντας την περιοχή γύρω από την ένωση με έναν τρόπο εξαρτώμενο από την κλίμακα για την αποφυγή ανεπιθύμητων φωτομετρικών παραμορφώσεων.

Ας υποθέσουμε ότι έχουμε στη διάθεσή μας:

1. δύο (2) εικόνες, $I_k(\mathbf{n})$, $k = 1, 2$ και $\mathbf{n} = [n_1 \ n_2]$, με πεδίο ορισμού το ορθογώνιο:

$$\mathcal{S} = \{\mathbf{n} : n_i = 0, \dots, N_i - 1, i = 1, 2\}$$

τις οποίες ας θεωρήσουμε ότι είναι στοιχισμένες, όπως το παράδειγμα των εικόνων που ακολουθούν.



Σχήμα 1: Αρχικές εικόνες

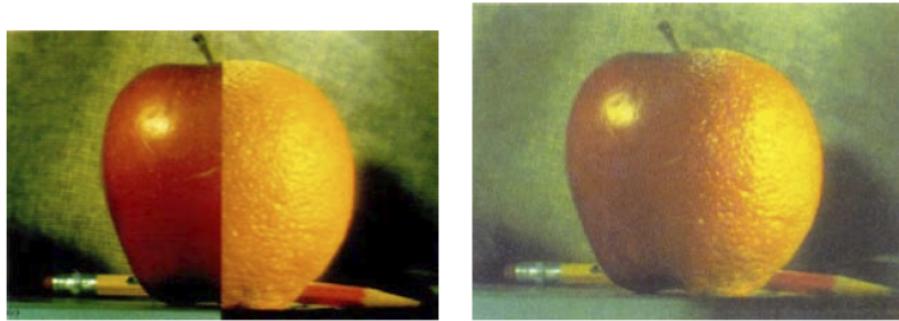
2. δύο (2) μάσκες m_k , $k = 1, 2$ του ιδίου μεγέθους με τις εικόνες, με τις ακόλουθες ιδιότητες:

- Ιδιότητα 1: $\sum_{k=1}^2 m_k(\mathbf{n}) = 1, \forall \mathbf{n} \in \mathcal{S}$
- Ιδιότητα 2: $m_k(\mathbf{n}) = 1$, αν και μόνο αν $I(\mathbf{n}) = I_k(\mathbf{n}), k = 1, 2, \forall \mathbf{n} \in \mathcal{S}$

Δύο μάσκες με τις παραπάνω ιδιότητες είναι οι ακόλουθες:

$$\begin{aligned} m_1(\mathbf{n}) &= \begin{cases} 1, & n_1 = 0, \dots, N_1 - 1, n_2 = 0, \dots, (N_2 - 1)/2 \\ 0, & \text{αλλού} \end{cases} \\ m_2(\mathbf{n}) &= 1 - m_1(\mathbf{n}) \end{aligned}$$

Το αποτέλεσμα της χρήσης της τεχνικής που θα υλοποιηθεί στο πλαίσιο της άσκησης στις εικόνες του Σχήματος 1 και τις μάσκες της παραπάνω σχέσης για δύο διαφορετικούς τρόπους ανάμιξης, φαίνεται στο Σχήμα 2.



Σχήμα 2: Συρραφή των εικόνων που προκύπτουν από τη χρήση διαφορετικών τρόπων ανάμιξης. Χώρις ανάμιξη (αριστερά), ανάμιξη με χρήσης λαπλασιανής πυραμίδας (δεξιά)

ΠΡΟΤΕΙΝΟΜΕΝΗ ΜΕΘΟΔΟΣ

1. Δημιουργία της $L + 1$ επιπέδων Γκαουσιανής πυραμίδας της μάσκας $m_1(\mathbf{n})$:

$$\mathcal{G}_{m_1} = \{g_0(\mathbf{n}), g_1(\mathbf{n}), \dots, g_{L-1}(\mathbf{n}), g_L(\mathbf{n})\} \quad (1)$$

2. Δημιουργία των $L + 1$ επιπέδων Λαπλασιανών πυραμίδων των εικόνων $I_k(\mathbf{n})$:

$$\mathcal{L}_{I_k} = \{l_{k,0}(\mathbf{n}), l_{k,1}(\mathbf{n}), \dots, l_{k,L-1}(\mathbf{n}), g_{k,L}(\mathbf{n})\}, k = 1, 2 \quad (2)$$

3. Δημιουργία της ακόλουθης $L + 1$ επιπέδων πυραμίδας:

$$\mathcal{B} = \{b_0(\mathbf{n}), b_1(\mathbf{n}), \dots, b_{L-1}(\mathbf{n}), g_{0,L}(\mathbf{n})\} \quad (3)$$

όπου:

$$\begin{aligned} b_j(\mathbf{n}) &= g_j(\mathbf{n})l_{1,j}(\mathbf{n}) + (1 - g_j(\mathbf{n}))l_{2,j}(\mathbf{n}), \quad j = 0, \dots, L-1 \\ g_{0,L}(\mathbf{n}) &= g_L(\mathbf{n})g_{1,L}(\mathbf{n}) + (1 - g_L(\mathbf{n}))g_{2,L}(\mathbf{n}). \end{aligned} \quad (4)$$

4. Ανακατασκευή της επιθυμητής εικόνας χρησιμοποιώντας την πυραμίδα \mathcal{B} .

ΔΗΜΙΟΥΡΓΙΑ ΓΚΑΟΥΣΙΑΝΗΣ ΠΥΡΑΜΙΔΑΣ

Μία γκαουσιανή πυραμίδα δεν είναι τίποτα άλλο παρά μία ακολουθία από υποδειγματοληπτημένες εικόνες φιλτραρισμένες ώστε να περιέχουν μόνο πληροφορία χαμηλών συχνοτήτων. Για το σκοπό αυτό:

- για το φιλτράρισμα συνήθως χρησιμοποιούμε ένα διαχωρίσιμο μονοδιάστατο πυρήνα. Ένας τυπικός πυρήνας με διωνυμικούς συντελεστές που αποτελεί την κρουστική απόκριση του επιθυμητού φίλτρου με χαμηλοπερατά χαρακτηριστικά, είναι ο ακόλουθος:

$$\mathbf{h} = \frac{1}{16}[1 \ 4 \ 6 \ 4 \ 1]^t. \quad (5)$$

Αν υποθέσουμε ότι θέλαμε να δημιουργήσουμε την πυραμίδα ενός μονοδιάστατου σήματος \mathbf{x}_k μήκους N , έχοντας ορίσει την κρουστική απόκριση του χαμηλοπερατού φίλτρου, μπορούμε να υπολογίσουμε την γραμμική συνέλιξη από την ακόλουθη, μητρικής μορφής, σχέση:

$$\mathbf{y}_k = T\mathbf{x}_k \quad (6)$$

όπου $T = \text{toeplitz}(\frac{1}{16}\mathbf{e}_N, \mathbf{h}_N^t)$ μητρώο Toeplitz μεγέθους $N \times N$ με \mathbf{e}_N διάνυσμα μήκους N με το πρώτο στοιχείο ίσο με τη μονάδα και όλα τα υπολοιπά μηδενικά και \mathbf{h}_N η επαυξημένη με $N - 5$ μηδενικά κρουστική απόκριση του συστήματος.

- Αν ορίσουμε τώρα το ακόλουθο μητρώο υποδειγματοληψίας:

$$D_N = \begin{bmatrix} \mathbf{e}_{\frac{N}{2}}^{1 \ t} \\ \mathbf{e}_{\frac{N}{2}}^{2 \ t} \\ \vdots \\ \mathbf{e}_{\frac{N}{2}}^{N \ t} \end{bmatrix} \quad (7)$$

όπου $\mathbf{e}_{\frac{N}{2}}^i$, $i = 1, 3, 5, 7, 9, \dots, N/2-1$ διάνυσμα μήκους $N/2$ με $N/2-1$ μηδενικά και μία μονάδα στην i -στή θέση, μπορούμε να κάνουμε υποδειγματοληψία κατά παράγοντα 2^1 αν εφαρμόσουμε την ακόλουθη εξίσωση:

$$\mathbf{x}_{k+1} = (D_N \otimes \mathbf{e}_2^{1 \ t})\mathbf{y}_k \quad (8)$$

ή ισοδύναμα:

$$\mathbf{x}_{k+1} = (D_N \otimes \mathbf{e}_2^{1 \ t})T\mathbf{x}_k. \quad (9)$$

όπου $A \otimes B$ συμβολίζει το γινόμενο Kronecker των μητρώων A, B .

¹Στην περίπτωση μίας εικόνας το μέγεθός της υποτετραπλασιάζεται κάθε φορά.

3. Επιθεβαιώστε (επιλέξτε εσείς τον τρόπο) ότι η κρουστική απόκριση της Σχέσης (5) αποτελεί διακριτό "ισοδύναμο" του γκαουσιανού πυρήνα.
4. Επιλέξτε ένα σήμα μονοδιάστατο της αρεσκείας σας και χρησιμοποιώντας τις Σχέσεις (6-9) δημιουργήστε γκαουσιανή πυραμίδα όσων επιπέδων επιθυμείτε.
5. Επιθεβαιώστε ότι από την γκαουσιανή πυραμίδα, μπορούμε να δημιουργήσουμε την λαπλασιανή και αντίστροφα. Για το σκοπό αυτό, γράψτε κατάλληλη συνάρτηση στο Matlab.
6. Από την διεύθυνση:

<http://www.mathworks.com/matlabcentral/fileexchange/30790-image-pyramid-gaussian-and-laplacian->, μπορείτε να κατεβάσετε ένα toolbox που έχει έτοιμες τις συναρτήσεις:

- *gen_Pyr()* για την δημιουργία γκαουσιανής ή Λαπλασιανής πυραμίδας
- *pyr_Expand()* για interpolation
- *pyr_Reduce()* για decimation
- *pyrBlend()* για blending, και
- *pyr_Reconstruct()* για την ανακατασκευή της εικόνας από πυραμίδα

Εξοικειωθείτε με τις συναρτήσεις του toolbox και καταγράψτε αναλυτικά τις βασικές λειτουργίες κάθε μίας εξ αυτών. Συσχετίστε τις βασικές λειτουργίες της συνάρτησης *gen_Pyr()* με τις βασικές λειτουργίες που περιγράφονται για την μονοδιάστατη περίπτωση στις Σχέσεις (5-9).

ΠΥΡΑΜΙΔΕΣ ΚΑΙ ΒΑΘΙΑ ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

Χωρική Πυραμίδα (Spatial Pyramid Pooling (SPP))

Τα συνελικτικά νευρωνικά δίκτυα (CNN) αποτελούν μια από τις πλέον διαδεδομένες βαθειές αρχιτεκτονικές που χρησιμοποιούνται στην υλοποίηση πολλών τεχνικών της μηχανικής μάθησης:

- στην επεξεργασία εικόνας και
- στην όραση των υπολογιστών

και αποδεδειγμένα, επιδεικνύουν εξαιρετικές επιδόσεις σε εφαρμογές όπως είναι:

- η αναγνώριση,
- η κατηγοριοποίηση αντικειμένων και πολλές άλλες.

Ωστόσο, υπάρχει ένας περιορισμός κατά την εκπαίδευσή των, αυτός του συγκεκριμένου μεγέθους των εικόνων το οποίο καθορίζεται από το μέγεθος του πρώτου επιπέδου, δηλαδή της εισόδου του δικτύου. Ο παραπάνω περιορισμός επηρεάζει ουσιαστικά την επίδοση του νευρωνικού δικτύου καθώς δεν είναι πάντα δυνατόν όλες οι εικόνες που διαθέτουμε για την εκπαίδευση του δικτύου μας να είναι ιδιων διαστάσεων. Έτσι πολλές από τις εικόνες που έχουμε στην διάθεση μας, πριν ενταχθούν στο σύνολο των δεδομένων:

- είτε αναγκαζόμαστε να τις κόψουμε (crop)
- ή να παραμορφώνονται γεωμετρικά (warp).

Οι παραπάνω διαδικασίες, έχουν ως συνέπεια να χάνεται πληροφορία ή να δημιουργούνται μικρό-γεωμετρικές ή/και φωτομετρικές παραμορφώσεις στο τελικό σύνολο των δεδομένων.



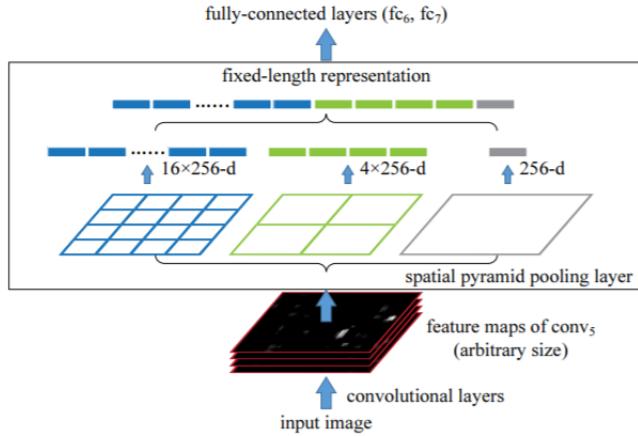
Σχήμα 3: Οπτική απεικόνιση του cropping και του warping

Ο βασικός λόγος για τον οποίο τα συμβατικά CNN απαιτούν συγκεκριμένο μέγεθος εισόδου, συνδέεται με την εκπαίδευση των πλήρως συνδεδεμένων επιπέδων (fully connected layers) που βρίσκονται, συνήθως, στα τελευταία επίπεδα (αν όχι στο τελευταίο) της αρχιτεκτονικής του δικτύου. Τα βάρη αυτών των επιπέδων, που θα πρέπει να εκπαιδευτούν, καθορίζουν την έξοδο του επόμενου επιπέδου (ή του δικτύου) με βάση την είσοδο την οποία δέχονται. Αυτό σημαίνει πώς κάθε αλλαγή στο μέγεθος της εισόδου τους θα απαιτούσε και την αναπροσαρμογή του αριθμού των βαρών του καθώς και την επανεκπαίδευση τους, γεγονός το οποίο δημιουργεί ένα σοβαρό και ουσιαστικά αξεπέραστο πρόβλημα.



Σχήμα 4: Διαγραμματική απεικόνιση ενός τυπικού CNN

Το παραπάνω πρόβλημα, το οποίο είναι γνωστό ως το πρόβλημα της διαστατικότητας λύνεται με ένα πολύ αποδοτικό τρόπο με τη χρήση των πυραμίδων. Συγκεκριμένα, με την τοποθέτηση ενός επιπέδου, που έχει την μορφή που φαίνεται στο Σχήμα 5 και το οποίο είναι γνωστό στη βιβλιογραφία ως επίπεδο χωρικής ομαδοποίησης (spatial pyramid pooling), μετά το τελευταίο συνελικτικό επίπεδο, αίρεται (γίνεται άρση) ο περιορισμός του μεγέθους της αρχικής εισόδου [10].



Σχήμα 5: Σχηματική απεικόνιση ενός επιπέδου χωρικής δειγματοληψίας

Το επίπεδο αυτό δέχεται ως είσοδο τους χάρτες χαρακτηριστικών (feature maps) που έχουν παραχθεί από τα προηγούμενα επίπεδα. Κάθε χάρτης περνάει από πλέγματα, σημείο στο οποίο υπεισέρχεται η έννοια των πυραμίδων των 4x4, 2x2, 1x1 (οι αριθμοί είναι ενδεικτικοί) παράλληλα, όπου σε κάθε παράθυρο (bin) καθενός πλέγματος γίνεται η διαδικασία της χωρικής ομαδοποίησης, της οποίας ειδική μορφή αποτελεί η διαδικασία της δειγματοληψίας που ακολουθείται στην επεξεργασία σημάτων.

Στη συνέχεια «διανυσματοποιούνται» οι τιμές των πλεγμάτων και τοποθετούνται σε ένα ενιαίο διάνυσμα. Με αυτό τον τρόπο έχουμε καταφέρει να κωδικοποιήσουμε την έξοδο των συνελικτικών επιπέδων αδιαφορόντας για τα μεγέθη. Η παραπάνω τεχνική έχει το πλεονέκτημα ότι διατηρεί ένα μέρος της χωρικής πληροφορίας καθώς και ότι μας δίνει την δυνατότητα το δίκτυο να εκπαιδευτεί σε εικόνες διαφορετικών κλιμάκων (scales) και λόγω προοπτικής (aspect ratio) μαθαίνοντας έτσι με έναν πιο αυθέραιτο τρόπο την πληροφορία που περιέχεται στις εικόνες.



Σχήμα 6: Διαγραμματική απεικόνιση του νέου CNN

Πυραμιδική Ομαδοποίηση

Ένα ακόμα βασικό πρόβλημα της όρασης των υπολογιστών είναι η πυραμιδική ομαδοποίηση εικονοστοιχείων (Pyramid Pooling που συνδέεται στενά με το προβλήμα της κατάτμησης εικόνων (image segmentation). Στην γενική περίπτωση η κατάτμηση έχει ως στόχο να υποδιαιρέσει την εικόνα σε συνιστώσες περιοχές και αντικείμενα. Μια περιοχή αναμένεται να έχει ομοιογενή χαρακτηριστικά όπως ένταση, υφή κ.α. Το πα-

ραπάνω πρόβλημα θα μπορούσε να διατυπωθεί και ως ένα πρόβλημα κατηγοριοποίησης κατά το οποίο αναθέτουμε κάθε εικονοστοιχείο σε μία κλάση.



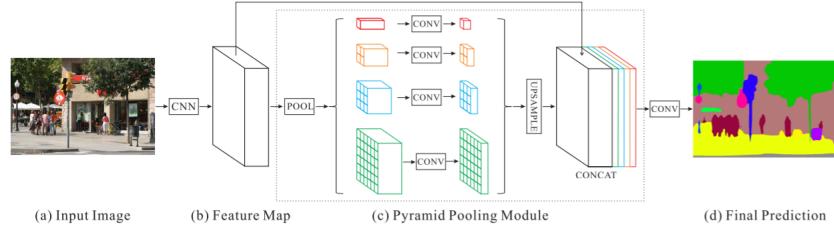
Σχήμα 7: Παράδειγμα κατάτμησης σκηνής

Στην προσπάθεια κατανόησης και αποδόμησης της σκηνής αντιμετωπίζονται κάποια εν γένει βασικά προβλήματα μερικά εκ των οποίων θα αναφερθούν στην συνέχεια.

1. Το πρόβλημα της μη μοναδικότητας. Πολλές από τις κλάσεις αναφέρονται σε παρόμοια έννοιολογικά αντικείμενα. Για παράδειγμα τα ζευγάρια αγρός – γη, ουρανοξύτης – κτίριο ουσιαστικά εννοιολογικά ταυτίζονται με αποτέλεσμα να μπορούν να κατηγοριοποιούνται σε παραπάνω από μία κλάσεις.
2. Πολλά αντικείμενα έχουν παρόμοιο σχήμα. Επομένως είναι πολύ πιθανόν να κατηγοριοποιηθούν λανθασμένα. Για παράδειγμα, λόγω σχήματος μια βάρκα μπορεί να κατηγοριοποιηθεί ως αμάξι. Ωστόσο, η λανθασμένη αυτή κατηγοριοποίηση θα αποφευχθεί αν το εν λόγω αντικείμενο βρίσκονταν στην όχθη ενός ποταμού. Τα παραπάνω μας οδηγούν στην ανάγκη το σύστημα μας να έχει την δυνατότητα εκμαίευσης γενικότερης πληροφορία για το περιεχόμενο της σκηνής και να μπορεί να την χρησιμοποιεί κατάλληλα.
3. Όλα τα αντικείμενα που υπάρχουν σε μία εικόνα, δεν έχουν το ίδιο μέγεθος. Για παράδειγμα, σε πολλές εφαρμογές απαιτείται ο εντοπισμός μικρών αντικειμένων όπως πινακίδες σήμανσης, φωτεινοί σηματοδότες κτλ. Το γεγονός αυτό επιφέρει την ανάγκη το σύστήμα αναγνώρισης να είναι ευαίσθητο στην διακριτικότητα τέτοιων αντικειμένων.

Με την χρήση των πυραμιδών στα συνελικτικά επίπεδα επιτυγχάνεται η λύση όλων των παραπάνω προβλημάτων, που συνδέονται με το πρόβλημα της κατανόησης της σκηνής, αναζητώντας χαρακτηριστικά σε πολλαπλές κλίμακες [11].

Έχοντας παράξει τους χάρτες χαρακτηριστικών, μέσω των «τυπικών» συνελικτικών επιπέδων, εφαρμόζουμε πάλι μια διαδικασία ομαδοποίησης (pooling) παρόμοια με αυτή

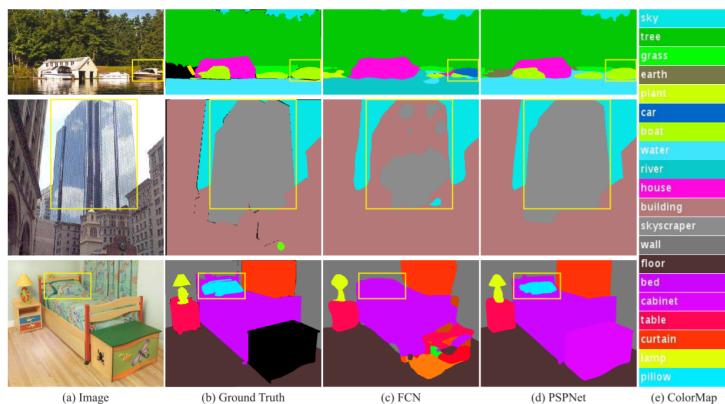


Σχήμα 8: Απεικόνιση της αρχιτεκτονικής του δικτύου PSPNet (τα χρώματα παιζουν ρόλο)

της προηγούμενης αρχιτεκτονικής, με την μόνη διαφορά, ότι χρησιμοποιούμε τον μέσο όρο των τιμών του κάθε bin από τα πλέγματα.

Στην συνέχεια εφαρμόζεται 1×1 συνέλιξη για την μείωση του αριθμού των χαρτών. Έπειτα προκειμένου να επιστρέψουμε στις αρχικές διαστάσεις των χαρτών, εφαρμόζεται η διγραμμική παρεμβολή σε κάθε κλίμακα και οι χάρτες των πυραμιδών συνενώνονται και οδηγούνται σε ένα τελευταίο συνελικτικό επίπεδο ώστε να γίνει η τελική πρόβλεψη.

Στο υψηλότερο επίπεδο της πυραμίδας αποτυπώνεται το κυρίαρχο χαρακτηριστικό της σκηνής (με αυτό τον τρόπο το σύστημα μαθαίνει μια συνολική πληροφορία για την εικόνα). Όσο πιο χαμηλά κινούμαστε στα επίπεδα της πυραμίδας τόσο πιο λεπτομερή χαρακτηριστικά αποτυπώνονται.



Σχήμα 9: Σύγκριση αποτελεσμάτων ένος πλήρως συνελικτικού δικτύου με το PSPNet

Στην παραπάνω εικόνα μπορούμε να δούμε πώς η επίδοση του δικτύου με την χρήση πυραμίδων βελτιώνεται. Στην πρωτή σειρά εικόνων η βάρκα αναγνωρίζεται ορθά, ενώ στην δεύτερη η επιφάνεια του ουρανοξύστη κατηγοριοποιείται σε μια ετικέτα. Τέλος στην τρίτη σειρά εικόνων η διάκριση του μαξιλαριού από την κουβέρτα είναι πολύ κοντά στην αλήθεια.

ΔΙΑΔΙΚΑΣΙΑ

1. Τις εικόνες του Σχήματος 1 μπορείτε να τις βρέτε στα αρχεία `apple.jpg` και `orange.jpg` αντίστοιχα.

- Ακολουθήστε τα βήματα της προτεινόμενης μεθόδου και δημιουργείστε τις αντίστοιχες πυραμίδες \mathcal{L}_{I_k} , $k = 1, 2$ και \mathcal{B} .
- Απεικονίστε τα αποτελέσματά σας σε κατάλληλο σχήμα και εξηγήστε αναλυτικά την μορφή, σε κάθε επίπεδο (κλίμακα) της πυραμίδας, των εικόνων που προκύπτουν.
- Εξηγήστε αναλυτικά την μορφή, της δεξιάς εικόνας του Σχήματος 2 που ουσιαστικά είναι η ανακατασκευασμένη επιθυμητή εικόνα.

2. Τις εικόνες του Σχήματος 3 μπορείτε να τις βρείτε στα αρχεία `woman.png` και `hand.png` αντίστοιχα.



Σχήμα 10: Αρχικές εικόνες

- Ορίστε κατάλληλες μάσκες $m_k(\mathbf{n})$, $k = 1, 2$ και κάνετε όλες τις απαραίτητες ενέργειες ώστε το αποτέλεσμα μετά την συρραφή να είναι παρόμοιο² με αυτό του Σχήματος 4.
- Αναλύστε και δικαιολογήστε όλες τις επιλογές σας.

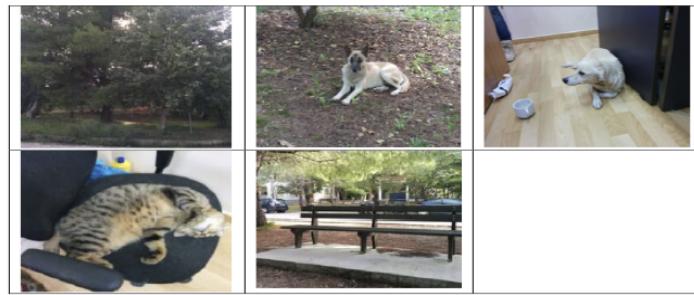
3. Δίνονται οι ακόλουθες εικόνες (δες Σχήμα 5):

- (α) `P200.jpg`
- (β) `dog1.jpg`
- (γ) `dog2.jpg`
- (δ) `cat.jpg`
- (ε) `bench.jpg`

²Μπορείτε να χρησιμοποιήσετε οποιαδήποτε συνάρτηση του Matlab που σας διευκολύνει προς την κατεύθυνση αυτή



Σχήμα 11: Επιθυμητό αποτέλεσμα



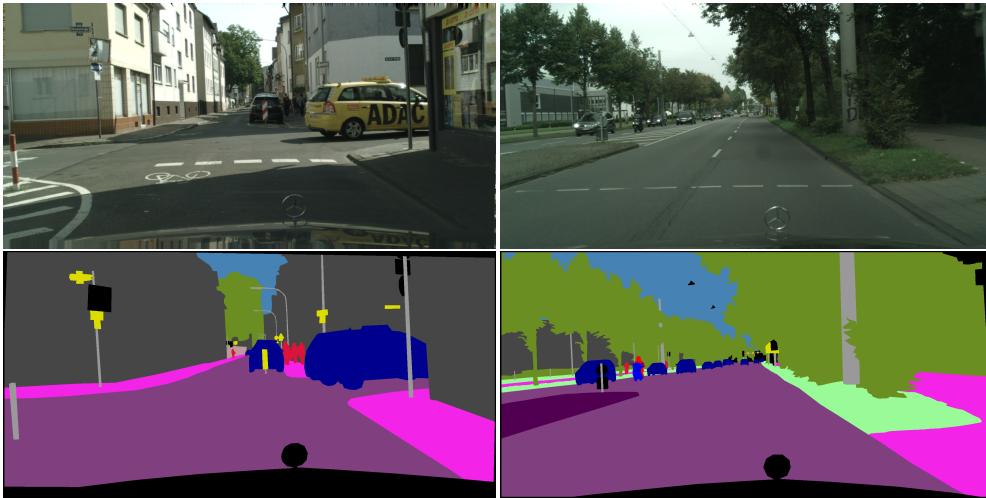
Σχήμα 12: Εικόνες που θα χρησιμοποιηθούν στην υλοποίηση του ερωτήματος 3

και σκοπός είναι χρησιμοποιώντας τις παραπάνω εικόνες και μία δικιά σας (η οποία θα ταυτοποεί την εργασία σας), να δημιουργήσετε μία σύνθεση της αρεσκείας σας.

Για το σκοπό αυτό :

- Ορίστε κατάλληλα αναγκαίες ποσότητες όπως:
 - (α') τις μάσκες m_k , $k = 1, 2, \dots, 6$ η μορφή των οποίων θα εξαρτηθεί από τις θέσεις που θα επιλέξετε να τοποθετήσετε αυτά που σας ζητούνται και το μέγεθός τους.
 - (β') τις πυραμίδες \mathcal{G}_{m_k} , $k = 1, 2, \dots, 5$ (δες Σχέση (1))
 - (γ') τις πυραμίδες \mathcal{L}_{I_k} , $k = 1, 2, \dots, 6$ (δες Σχέση (2))
 - (δ') την πυραμίδα \mathcal{B} της Σχέσης (3), αφού ορίστε κατάλληλα τις ποσότητες (δες Σχέση (4)) που θα απαιτηθούν.
 - Καταγράψτε τις αντίστοιχες ιδιότητες 1 & 2 της Σελίδας 2, που θα ισχύουν στην περίπτωσή σας.
4. Στη συνέχεια, σας δίνετε σύνολο δεδομένων, στο οποίο περιέχονται εικόνες που απεικονίζουν τοπία από πόλεις όπως μπορείτε να δείτε στην παρακάτω εικόνα που φαίνονται μερικά παραδείγματα.

Σας δίνονται επίσης τα εξής αρχεία :



Σχήμα 13: Παραδείγματα εικόνων που περιέχονται στο σύνολο εικόνων που σας δίνεται (πάνω), και οι αντίστοιχες ground truths (κάτω).

- (α) **pspnet.py**: Περιέχει την κλάση PSPNet του μοντέλου που περιγράφαμε παραπάνω.
- (β) **cityscapes_dataset.py**: Περιέχει την κλάση του συνόλου δεδομένων Cityscapes. Σε αυτό ορίζονται από που θα φορτώνονται:
 - i. οι εικόνες,
 - ii. οι μετασχηματισμοί προεξεργασίας των εικόνων κλπ,
 πριν δωθούν ως είσοδος στο μοντέλο.
- (γ) **pspnet_train.py**: script στο οποίο κωδικοποιείται η διαδικασία εκπαίδευσης του μοντέλου.
- (δ) **cityscapes_val_dataset.zip**: Σύνολο εικόνων, αντίστοιχων με αυτές που χρησιμοποιήθηκαν στην εκπαίδευση του μοντέλου, που θα χρησιμοποιηθούν στην φάση της επικύρωσής του validation.
- (ε) **train_epoch_200_CPU.pth**: Στιγμιότυπο του εκπαίδευμένου μοντέλου (βάρη, ενδεχομένως και άλλες παράμετροι) που θα χρησιμοποιήσετε.
- (ζ) **cityscapes_colors.txt**: Κατάλογος με τους συνδυασμούς RGB που αντιστοιχούν σε κάθε κλάση, ώστε να γίνεται σωστά η αποκωδικοποίηση των έγχρωμων μασκών.
- (η) **cityscapes_names.txt**: Κατάλογος με τα ονόματα των κλάσεων, σε πλήρη αντιστοιχία με το cityscapes_colors.txt.

Δίνονται και άλλα βοηθητικά αρχεία που καλούνται μέσα από τα παραπάνω.

Το μοντέλο είναι ήδη εκπαίδευμένο με το **pspnet_train.py** σε εικόνες αντίστοιχες με αυτές που σας δίνονται. Ο κώδικας εκπαίδευσης σας δίνεται απλώς για

βοηθητικούς λόγους και για να κατανοήσετε την βασική διαδικασία που ακολουθούμε συνήθως όταν θέλουμε να εκπαιδεύσουμε ένα νευρωνικό δίκτυο. Εσείς καλείστε να φτιάξετε αντίστοιχο script pspnet_eval.py στο οποίο θα αξιολογείτε το εκπαιδευμένο μοντέλο ως προς την επίδοσή του ως προς:

- (α') την σωστή κατηγοριοποίηση των αντικειμένων και
- (β') τον σωστό εντοπισμό τους μέσα στην εικόνα.

Για να το κάνετε αυτό θα πρέπει να διαλέξετε μια μετρική συνάρτηση της επιλογής σας, που να αρμόζει στο πρόβλημα που πάμε να λύσουμε (κατάτμηση), και στη συνέχεια θα υπολογίσετε:

- (α') την μέση τιμή και
- (β') τη διασπορά της μετρικής ανά κλάση,

ελέγχοντας όλες τις εικόνες. Τέλος, θα υπολογίσετε την μέση τιμή της επίδοσης ανά εικόνα και έπειτα θα δώσετε την μέση τιμή και την διασπορά αυτών των αποτιμήσεων.

Όλα τα παραπάνω αρχεία μπορείτε να τα βρείτε εδώ³

Υποδείξεις

(α') Για την εκτέλεση του παραπάνω ερωτήματος θα χρειαστείτε ένα περιβάλλον προγραμματισμού το οποίο να υποστηρίζει την γλώσσα Python. Σας προτείνουμε τρεις επιλογές κατά αύξουσα σειρά περιπλοκότητας.

Σαν πρώτη επιλογή και ανεξάρτητα του λογισμικού του υπολογιστή σας, μπορείτε να αξιοποιήσετε το περιβάλλον του colab.research.google.com στο οποίο μπορείτε να εκτελείτε python scripts και notebooks, εφόσον διαθέτετε λογαριασμό Google.

Εποκεφθείτε τον παραπάνω σύνδεσμο. Στον κεντρικό φάκελο πάνω (CV_1-PYRAMIDS-files) κάνοντας δεξί κλικ θα επιλέξετε την προσθήκη συντόμευσης στο Drive σας. Έπειτα θα ανοίξετε το περιβάλλον colab.research.google.com. Επιλέξετε στην γραμμή εργαλειών, που βρίσκεται αριστερά στην οθόνη, την επιλογή Αρχεία και έπειτα προσάρτηση στο Drive. Στη συνέχεια θα εμφανιστεί ένα script ώστε να συνδέσετε το Drive σας με το colab το οποίο και θα τρέξετε. Θα σας ζητηθεί ένας κωδικός ταυτοποίησης τον οποίο θα αποκτήσετε πατώντας το αντίστοιχο link. Με την ολοκλήρωση της παραπάνω διαδικασίας θα πρέπει να έχει δημιουργηθεί ένας νέος φάκελος που θα ονομάζεται drive περιέχοντας πλέον και τον φάκελο CV_1-PYRAMIDS-files. Τέλος δημιοργήστε καινούργιο notebook με τον δικό σας κώδικα σε ξεχωριστό φάκελο (αφού στον παραπάνω έχετε μόνο δικαίωμα ανάγνωσης) και γράψτε τον κώδικά σας.

³<https://drive.google.com/drive/folders/1Pe5cTVaFYq2lOjTZyBBtefS9N3Mi9E6z?usp=sharing>

Σαν δεύτερη επιλογή, μπορείτε να κατεβάσετε το περιβάλλον Anaconda το οποίο υποστηρίζει τις γλώσσες Python και R. Ανοίγοντας το Navigator μπορείτε να δημιουργήσετε διαφορετικά περιβάλλοντα στα οποία να κατεβάσετε τις απαραίτητες βιβλιοθήκες και πακέτα ανάλογα με τις απαιτήσεις της εφαρμογής που θέλετε να υλοποιήσετε (εναλλακτικά μπορείτε να δημιουργήσετε απευθείας ένα περιβάλλον από το command prompt του Anaconda).

(β) Απαιτείται η χρήση του πακέτου PyTorch. Για όσους επιλέξουν για περιβάλλον εργασίας το Anaconda μπορείται να ανατρέξετε στον παρακάτω σύνδεσμο και συμπληρώνοντας τα κατάλληλα πεδία των χαρακτηριστικών του συστήματος σας <https://pytorch.org/> να σας επιστραφεί μια εντολή (στο πεδίο Pytorch Build επιλέξτε το πεδίο Stable και στο package το πεδίο Conda). Για δική σας ευκολία σας δίνεται και το αρχείο requirements.txt ώστε τρέχοντας conda env create -f requirements.txt να έχετε ένα έτοιμο περιβάλλον με τα απαραίτητα πακέτα εγκατεστημένα. Ενδεχομένως να περιέχει και πακέτα που δεν χρειάζεστε, οπότε εναλλακτικά μπορείτε να ακολουθήσετε την παρακάτω διαδικασία και να έχετε το environment.yaml ως σημείο αναφοράς για τις εκδόσεις των πακέτων που θα χρειαστείτε. Για την δημιουργία ενός περιβάλλοντος με εγκατεστημένο το πακέτο PyTorch ανοίξτε το command prompt και εκτελέστε με την σειρά τις παρακάτω εντολές:

- conda deactivate
- conda create - -name onoma_env
- conda activate onoma_env
- Εκτελέστε την εντολή που προέκυψε από τον σύνδεσμο.

(γ) Το περιβάλλον του Anaconda διαθέτει πλειάδα επιλογών text editors (πχ. JupyterLab, JupyterNotebook, Spyder). Μπορείτε να διαλέξετε έναν editor της αρεσκείας σας και να εκτελέσετε εκεί τα πειράματά σας.

(δ) Για τυχόν απορίες και διευκρινίσεις που αφορούν την διαδικασία της ασκήσης μπορείτε να απευθυνθείτε στα παρακάτω email: katsos@ceid.upatras.gr, pgeorgant@ceid.upatras.gr.

Βιβλιογραφία

- [1] Burt, Peter J and Adelson, Edward H. The Laplacian pyramid as a compact image code. In *Readings in computer vision*. Elsevier, 1987.
- [2] Stiller, Christoph and Lappe, Dirk. Laplacian pyramid coding of prediction error images. In *Visual Communications and Image Processing '91: Visual Communication*. International Society for Optics and Photonics, 1991.
- [3] Unser, Michael. An improved least squares Laplacian pyramid for image compression. *Signal Processing*, 27(2), 1992.
- [4] Yun, Se-Hwan and Kim, Jin Heon and Kim, Suki. Image enhancement using a fusion framework of histogram equalization and Laplacian pyramid. *IEEE Transactions on Consumer Electronics*, 56(4), 2010.
- [5] Burt, Peter J and Adelson, Edward H. Merging images through pattern decomposition. In *Applications of Digital Image Processing VIII*. International Society for Optics and Photonics, 1985.
- [6] Teng, Yanwen and Liu, Fuyan and Wu, Ruoyu. The research of image detail enhancement algorithm with Laplacian pyramid. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*. IEEE, 2013.
- [7] J.M. Ogden, E.H. Adelson, J.R. Bergen, P.J. Burt. Pyramid-based computer graphics. *RCA Engineer*, 30(5), 1985.
- [8] Burt, Peter J and Adelson, Edward H. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics (TOG)*, 2(4), 1983.
- [9] J.M. Ogden, E.H. Adelson, J.R. Bergen, P.J. Burt. Pyramid-based computer graphics. *ACM Transactions on Graphics]*, 2(4), 1983.
- [10] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 37(9), 2015.

- [11] Zhao, Hengshuang and Shi, Jianping and Qi, Xiaojuan and Wang, Xiaogang and Jia, Jiaya. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017.

ΠΑΡΑΡΤΗΜΑ

pspnet_train.py

```

import numpy as np
from pspnet import *
import torch.nn as nn
import torch.optim as optim
from datetime import datetime
import matplotlib.pyplot as plt
from torch.utils.data import DataLoader
from cityscapes_dataset import Cityscapes

# Set device
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# Dataset
dataset_train = Cityscapes(split='train',
                           data_root='cityscapes_dataset/',
                           data_list='cityscapes_dataset/list/cityscapes/
                           fine_train.
                           txt')

# Dataloader
train_dataloader = DataLoader(dataset_train, batch_size=1, shuffle=True,
                             num_workers=8)

# Loss Criterion
criterion = nn.CrossEntropyLoss(ignore_index=255)

# Model
model = PSPNet(layers=50, bins=(2, 3, 6, 8), dropout=0.1, classes=35,
               zoom_factor=8, use_ppm=True,
               pretrained=True, criterion=criterion).
               to(device)

# Optimizer and list of parameters to optimize
modules_ori = [model.layer0, model.layer1, model.layer2, model.layer3, model.
               layer4]
modules_new = [model.ppm, model.cls, model.aux]
lr = 0.001
params_list = []
for module in modules_ori:
    params_list.append(dict(params=module.parameters(), lr=lr))
for module in modules_new:
    params_list.append(dict(params=module.parameters(), lr=lr * 10))

```

```

optimizer = optim.SGD(params_list, lr=lr, momentum=0.9, weight_decay=0.0001)

timestamp = datetime.fromtimestamp(datetime.timestamp(datetime.now())).
                           strftime("%d-%m-%Y, %H:%M:%S")
exp_path = timestamp
os.mkdir(exp_path)

# Train loop
print('Started at ' + timestamp)
train_losses = []
for epoch in range(200):

    train_sum_loss = 0
    for img, mask in train_dataloader:

        # Zero stored gradients
        optimizer.zero_grad()

        # Forward pass
        _, main_loss, aux_loss = model(img.to(device), mask.to(device))
        loss = torch.mean(main_loss) + 0.4 * torch.mean(aux_loss)

        # Loss back-propagation
        loss.backward()

        # Optimization step
        optimizer.step()

        train_sum_loss += loss.detach().cpu().numpy() / len(train_dataloader)
    train_losses.append(train_sum_loss)

    print('epoch ' + str(epoch) + ' train loss ' + str(np.round(
        train_sum_loss, 4)))

timestamp = datetime.fromtimestamp(datetime.timestamp(datetime.now())).
                           strftime("%d-%m-%Y, %H:%M:%S")
print('Ended at ' + timestamp)

# Save trained model
torch.save(model, exp_path + '/pspnet_' + str(epoch) + 'ep.pt')
torch.save(train_losses, exp_path + '/pspnet_' + str(epoch) +
                           'ep_TRAIN_LOSSES.pt')

plt.figure(figsize=(20,10))
plt.semilogy(train_losses)
plt.legend(['train'])
plt.show()

```