ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Μάθημα Προπτυχιακών Σπουδών:
Ανάλυση Εικόνας
Ακαδημαϊκό έτος: 2023 - 2024
Εξάμηνο: 7ο
Υπολογιστική Εργασία

Ομάδα Εργασίας:	**Υπεύθυνοι Καθηγητές:**
Νίκη Δημοπούλου Π20057 Παναγιώτα Νικολάου Π20009 Δημήτρης Στυλιανού Π20004	Γεώργιος Τσιχριντζής Διονύσιος Σωτηρόπουλος

Περιεχόμενα

Ο κώδικας της εργασίας είναι αναρτημένος στο GitHub.

Εκφώνηση

Υπολογιστική Εργασία Ανάλυσης Εικόνας
Ακαδημαϊκό Έτος 2022 – 2023
Γ. Τσιχριντζής Δ. Σωτηρόπουλος

ΑΝΑΚΤΗΣΗ ΕΙΚΟΝΩΝ ΜΕ ΒΑΣΗ ΤΟ ΠΕΡΙΕΧΟΜΕΝΟ
(**Content – Based Image Retrieval**)

Στόχος της συγκεκριμένης υπολογιστικής εργασίας είναι ανάπτυξη γραφοθεωρητικών αλγορίθμων για την ανάκτηση εικόνων με βάση το περιεχόμενο. Τα βασικά βήματα της προτεινόμενης αλγοριθμικής προσέγγισης έχουν ως ακολούθως:

1. Κανονικοποίηση Σειράς Κατάταξης (**Rank Normalization**)
2. Κατασκευή Υπεργράφου **(Hypergraph Construction)**
3. Υπολογισμός Ομοιότητας Υπερακμών **(Hyperedge Similarities)**
4. Υπολογισμός Καρτεσιανού Γινομένου μεταξύ των στοιχείων των Υπερακμών **(Cartesian Product of Hyperedge Elements)**
5. Υπολογισμός Ομοιότητας βάσει του κατασκευασμένου Υπεργράφου **(Hypergraph – Based Similarity)**

Λεπτομερής περιγραφή της παραπάνω αλγοριθμικής διαδικασίας μπορείτε να βρείτε στο άρθρο με τίτλο "**Multimedia Retrieval through Unsupervised Hypergraph-based Manifold Ranking**", IEEE TRANSACTIONS ON IMAGE PROCESSING, VOL. 28, NO. 12, DECEMBER 2019 .

Το άρθρο θα αναρτηθεί στα έγγραφα του μαθήματος.Ζητούμενα:

i. Να παρουσιάσετε μια αναλυτική περιγραφή της υπολογιστικής διαδικασίας που παρουσιάζεται στο άρθρο ενσωματώνοντάς την στην τεκμηρίωση της εργασίας σας.(**20% του συνολικού βαθμού**)
ii. Να αναπτύξετε κώδικα σε Matlab ή Python για την προγραμματιστική υλοποίηση των παραπάνω υπολογιστικών βημάτων. (**20% του συνολικού βαθμού**)
iii. Το σύνολο των αντικειμενικών χαρακτηριστικών για την διανυσματική αναπαράσταση της κάθε εικόνας να εξαχθεί με την χρήση ενός προεκπαιδευμένου νευρωνικού δικτύου (**squeezenet**, **googlenet**, **resnet18**, **resnet50**, **resnet101**, κλπ.). Συγκεκριμένα, μπορείτε να χρησιμοποιήσετε ως αντικειμενικά χαρακτηριστικά της κάθε εικόνας έξοδο που αντιστοιχεί σε κάποιο από τα ενδιάμεσα κρυφά επίπεδα τωνπροαναφερθέντων νευρωνικών δικτύων. (**20% του συνολικού βαθμού**)
iv. Να παρουσιάσετε παραδείγματα της ορθής εκτέλεσης του κώδικάς σας. Χαρακτηρίστε κάποιες από τις εικόνες της βάσης ως εικόνες στόχο (**target images**) και παρουσιάστε μια λίστα με τις συναφέστερες εικόνες της βάσης. (**20% του συνολικού βαθμού**)
v. Προτείνετε μια συστηματική διαδικασία για την μέτρηση της ακρίβειας του συγκεκριμένου αλγορίθμου και παρουσιάσετε τα αποτελέσματά της. (**20% του συνολικού βαθμού**)

Μπορείτε να εργαστείτε σε ομάδες των 2 ή 3 φοιτητών.

Αναλυτική περιγραφή της υπολογιστικής διαδικασίας του άρθρου

Στο δοθέν άρθρο "**Multimedia Retrieval through Unsupervised Hypergraph-based Manifold Ranking**" περιγράφεται ένας αλγόριθμος μάθησης χωρίς επίβλεψη για ανάκτηση και κατάταξη πολυμέσων, ο οποίος ονομάζεται **Log-based Hypergraph of Ranking References (LHRR)**. Ο αλγόριθμος αποτελείται από πέντε βασικά βήματα και ένα προεπεξεργαστικό, όπου θα αναλύσουμε παρακάτω.

Αρχικά, στο προεπεξεργαστικό βήμα γίνεται κωδικοποίηση του κάθε πολυμεσικού αντικειμένου μέσω μιας διαδικασίας εξαγωγής διανυσμάτων χαρακτηριστικών (με τη βοήθεια ενός προεκπαιδευμένου νευρωνικού δικτύου), τα οποία έπειτα χρησιμοποιούνται για τον υπολογισμό της ομοιότητας μεταξύ δύο πολυμεσικών αντικειμένων.

Εξαγωγή Χαρακτηριστικών και Υπολογισμός Ομοιότητας – Feature Extraction and Similarity Computing

$$\boldsymbol{\varepsilon}: \boldsymbol{o_i} \rightarrow \mathbb{R}^{\boldsymbol{d}},$$

είναι η συνάρτηση που εξάγει το διάνυσμα χαρακτηριστικών $\overrightarrow{v_i}$ *από το πολυμεσικό αντικείμενο* o_i.

$$\boldsymbol{\delta}: \mathbb{R}^{\boldsymbol{d}} \times \mathbb{R}^{\boldsymbol{d}} \rightarrow \mathbb{R}^{+}$$

$$\boldsymbol{\delta}\left(\boldsymbol{\varepsilon}(\boldsymbol{o_i}), \boldsymbol{\varepsilon}(\boldsymbol{o_j})\right) = \boldsymbol{\delta}(\overrightarrow{\boldsymbol{v_i}}, \overrightarrow{\boldsymbol{v_j}}) = \|\overrightarrow{\boldsymbol{v_i}} - \overrightarrow{\boldsymbol{v_j}}\|,$$

είναι η συνάρτηση που υπολογίζει την απόσταση του πολυμεσικού αντικειμένου o_j *από το* o_i, *βάσει της απόστασης των αντίστοιχων διανυσμάτων χαρακτηριστικών.*

$$\boldsymbol{\rho}(\boldsymbol{o_i}, \boldsymbol{o_j}) = \frac{\mathbf{1}}{\boldsymbol{\delta}},$$

όπου $\rho(o_i, o_j)$ *είναι το μέτρο ομοιότητας του* πολυμεσικού αντικειμένου o_j *από το* o_i.

Ανάκτηση Πολυμέσων και Μοντέλο Κατάταξης – Multimedia Retrieval and Rank Model

Στη συνέχεια, ορίζουμε ως:

- $\boldsymbol{C} = \{\boldsymbol{o_1}, \boldsymbol{o_2}, \dots, \boldsymbol{o_n}\}$ ένα σύνολο πολυμεσικών αντικειμένων, όπου $\boldsymbol{n} = |\boldsymbol{C}|$ υποδηλώνει το μέγεθος του συνόλου C.
- $\boldsymbol{o_q}$ το αντικείμενο στόχος (query object).
- $\boldsymbol{\tau_q} = (\boldsymbol{o_1}, \boldsymbol{o_2}, \dots, \boldsymbol{o_n})$ μια διατεταγμένη λίστα – ranked list – βάσει του o_q και του $\rho(o_q, o_j)$. Δηλαδή η λίστα τ_q περιέχει όλα τα πολυμεσικά αντικείμενα του συνόλου C, τα οποία είναι διατεταγμένα σε φθίνουσα σειρά βάσει του μέτρου ομοιότητας του o_j από το o_q. Η διατεταγμένη λίστα τ_q είναι μια μετάθεση του συνόλου C. Επομένως, η διατεταγμένη λίστα τ_q μπορεί να οριστεί ως αμφιμονοσήμαντη απεικόνιση του συνόλου C στο σύνολο $[N] = \{1,2, \dots, n\}$. Εάν το πολυμεσικό αντικείμενο o_i προηγείται του o_j στη λίστα τ_q $(\delta\eta\lambda\acute{\alpha}\delta\eta\ \tau_q(i) < \tau_q(j))$, τότε $\rho(o_q, o_i) \geq \rho(o_q, o_j)$. Ως $\boldsymbol{\tau_q}(\boldsymbol{i})$ ορίζουμε την θέση του o_i στη διατεταγμένη λίστα τ_q.

  Όμως επειδή ο υπολογισμός της διατεταγμένης λίστας τ_q μπορεί να είναι πολύ απαιτητικός και χρονοβόρος – ειδικά όταν το πλήθος των πολυμεσικών αντικειμένων είναι μεγάλο, ορίζουμε ως τ_q

μια διατεταγμένη λίστα που περιέχει μερικά από τα πολυμεσικά αντικείμενα του συνόλου C.

- $\boldsymbol{C_L}$ το σύνολο που περιέχει τα μερικά πολυμεσικά αντικείμενα από το σύνολο C. Επομένως, $C_L \subset C$.
- $\boldsymbol{L} = |\boldsymbol{C_L}|$ το μέγεθος του συνόλου C_L.
- $\boldsymbol{T} = \{\boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \dots, \boldsymbol{\tau_n}\}$ ένα σύνολο που περιέχει όλες τις διατεταγμένες λίστες τ_q.
- $\boldsymbol{N(q,k)}$ ένα σύνολο που περιέχει τους k γείτονες με το μεγαλύτερο μέτρο ομοιότητας βάσει του αντικειμένου στόχου o_q και ορίζεται ως:

$$\boldsymbol{N(q,k) = \{S \subseteq C, |S| = k \wedge \forall o_i \in S, o_j \in C - S : t_q(i) < t_q(j)\}}$$

Στη συνέχεια, θα αναλυθούν τα πέντε βασικά αλγοριθμικά βήματα:

Κανονικοποίηση Σειράς Κατάταξης – Rank Normalization

Σε αυτό το βήμα γίνεται κανονικοποίηση των διατεταγμένων λιστών, διότι οι σχέσεις που δημιουργούνται μέσω αυτών και του συνόλου $N(q,k)$ δεν είναι συμμετρικές. Επομένως, υπολογίζεται ένα νέο μέτρο ομοιότητας (κανονικοποίηση αμοιβαίας κατάταξης – reciprocal rank normalization) μεταξύ των o_i και o_j:

$$\boldsymbol{\rho_n(o_i, o_j) = 2L - (\tau_i(j) + \tau_j(i))}$$

Άρα από το παραπάνω προκύπτει ότι το ρ_n αποτελεί συμμετρικό μέτρο ομοιότητας μεταξύ των o_i και o_j, δηλαδή ισχύει $\boldsymbol{\rho_n(o_i, o_j) = \rho_n(o_j, o_i)}$.

Κατασκευή Υπεργράφου – Hypergraph Construction

Ο όρος **"υπεργράφος – hypergraph"** είναι μια γενίκευση του βασικού γράφου, όπου οι ακμές είναι μη κενά υποσύνολα του συνόλου των κορυφών και επομένως μπορούν να συνδέουν οποιονδήποτε αριθμό κορυφών.

Ορίζουμε ως:

- $\boldsymbol{G} = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{w})$ τον υπεργράφο
- $\boldsymbol{V}$ ένα πεπερασμένο σύνολο κορυφών
  Κάθε κορυφή $v \in V$ συσχετίζεται με ένα πολυμεσικό αντικείμενο $o_i \in C$.
- $\boldsymbol{e_i}$ μια υπερακμή του υπεργράφου
  Ο όρος **"υπερακμή – hyperedge"** αναφέρεται σε ένα υποσύνολο κορυφών του υπεργράφου, δηλαδή $e_i = \{v_1, v_2, \dots, v_m\}$. Για κάθε $o_i \in C$ ορίζεται μια υπερακμή με βάση το k-οστό σύνολο γειτνίασης και τους αντίστοιχους γείτονες του.
- $\boldsymbol{E}$ μια οικογένεια υποσυνόλων του συνόλου V, έτσι ώστε $\bigcup_{e \in E} e = V$.
- $\boldsymbol{H}$ μία μήτρα (continuous incidence matrix) μεγέθους $|E| \times |V|$ και έστω $\boldsymbol{r: E \times V \to R^+}$ μια συνάρτηση με σύνολο τιμών το $\boldsymbol{R^+}$. Επομένως,

$$\boldsymbol{h(e_i, v_j) = \begin{cases} r(e_i, v_j), & \text{όταν } v_j \in e_i \\ 0, & \text{αλλιώς} \end{cases}}$$

- $\boldsymbol{r(e_i, v_j)}$ τον βαθμό στον οποίο η κορυφή v_j ανήκει στην υπερακμή e_i και υπολογίζεται ως εξής:

$$\boldsymbol{r(e_i, v_j) = \sum_{o_x \in N(i,k) \wedge o_j \in N(x,k)} w_p(i,x) \times w_p(x,j)},$$

$$\boldsymbol{\text{όπου } o_x \in N(i,k) \text{ γείτονας του } o_i \text{ και } o_j \in N(x,k) \text{ γείτονας του } o_x.}$$

- $\boldsymbol{w_p(i,x)}$ μια συνάρτηση που αποδίδει ένα βάρος συσχέτισης στο o_x ανάλογα με τη θέση του στην διατεταγμένη λίστα τ_i και υπολογίζεται ως εξής:

$$\boldsymbol{w_p(i,x) = 1 - \log_k \tau_i(x)}$$

  Η συνάρτηση αποδίδει μέγιστο βάρος συσχέτισης ίσο με 1 στο πρώτο πολυμεσικό αντικείμενο της διατεταγμένης λίστας τ_i, το οποίο πάντα αντιστοιχεί στο αντικείμενο στόχος o_i.

- $\boldsymbol{N_h}$ το σύνολο γειτνίασης του υπεργράφου. Δεδομένου μιας υπερακμής e_i, το σύνολο N_h περιέχει τις k κορυφές με τις μεγαλύτερες τιμές σύμφωνα με τη συνάρτηση $h(e_i,\cdot)$ και ορίζεται ως:

$$\boldsymbol{N_h(q,k) = \{S \subseteq e_q, |S| = k \wedge \forall o_i \in S, o_j \in e_q - S : h(q,i) > h(q,j)\}}$$

- $\boldsymbol{w(e_i)}$ το βάρος μιας υπερακμής e_i που υποδηλώνει τον βαθμό βεβαιότητας των σχέσεων που δημιουργούνται μεταξύ των κορυφών από την υπερακμή. Δηλαδή το βάρος αντικατοπτρίζει το πόσο σίγουροι είμαστε ότι οι κορυφές που συνδέονται με την υπερακμή, σχετίζονται πραγματικά με κάποιο ουσιαστικό τρόπο και υπολογίζεται ως εξής:

$$\boldsymbol{w(e_i) = \sum_{j \in N_h(i,k)} h(i,j)}$$

Υπολογισμός Ομοιότητας Υπερακμών – Hyperedge Similarities

Για τον υπολογισμό της ομοιότητας μεταξύ δύο υπερακμών προτείνονται δύο υποθέσεις, όπου βασίζονται στον υπολογισμό μίας μήτρας ομοιότητας $\boldsymbol{S}$ (pairwise similarity matrix).

Η **πρώτη υπόθεση** δηλώνει ότι παρόμοια πολυμεσικά αντικείμενα παρουσιάζουν παρόμοιες διατεταγμένες λίστες και, συνεπώς παρόμοιες υπερακμές. Μόλις όλη πληροφορία ομοιότητας κωδικοποιηθεί από την μήτρα H, ένα μέτρο ομοιότητας μεταξύ των υπερακμών e_i και e_j μπορεί να υπολογιστεί από το άθροισμα των τιμών της συνάρτησης h πολλαπλασιάζοντας τις αντίστοιχες κορυφές $h(e_i, v_x) \times h(e_j, v_x)$. Αυτή η υπολογιστική διαδικασία μοντελοποιείται για όλα τα στοιχεία, πολλαπλασιάζοντας την μήτρα H και την ανάστροφη της και έχει ως εξής:

$$\boldsymbol{S_h = HH^T}$$

Η **δεύτερη υπόθεση** δηλώνει ότι παρόμοια πολυμεσικά αντικείμενα, αναμένεται να αναφέρονται από τις ίδιες υπερακμές. Για τον υπολογισμό του μέτρου ομοιότητας μεταξύ των κορυφών v_i και v_j, οι τιμές της συνάρτησης h στις αντίστοιχες υπερακμές θα πρέπει να πολλαπλασιαστούν $h(e_x, v_i) \times h(e_x, v_j)$. Αυτή η διαδικασία υπολογίζεται πολλαπλασιάζοντας την H^T και έχει ως εξής:

$$\boldsymbol{S_v = H^T H}$$

Δεδομένου ότι οι μήτρες S_h και S_v κωδικοποιούν σχετικές και συμπληρωματικές πληροφορίες, συνδυάζονται μέσω ενός πολλαπλασιασμού στοιχείο με στοιχείο $s(i,j) = s_h(i,j) \times s_v(i,j)$. Κατά συνέπεια, η μήτρα ομοιότητας S μπορεί να υπολογιστεί με ένα γινόμενο Hadamard:

$$\boldsymbol{S = S_h \circ S_v}$$

Υπολογισμός Καρτεσιανού Γινομένου μεταξύ των στοιχείων των Υπερακμών – Cartesian Product of Hyperedge Elements

Προκειμένου να εξαχθούν οι ανά ζεύγη σχέσεις απευθείας από το σύνολο των στοιχείων που ορίζονται από μια υπερακμή, χρησιμοποιείται μια πράξη καρτεσιανού γινόμενου. Στόχος είναι η μεγιστοποίηση της πληροφορίας ομοιότητας, η οποία μπορεί να συγκεντρωθεί στις ομοιότητες των υπερακμών. Δεδομένων δύο υπερακμών $e_q, e_i \in E$, το καρτεσιανό γινόμενο μεταξύ αυτών ορίζεται ως:

$$\boldsymbol{e_q \times e_i = \{(v_x, v_y): v_x \in e_q \wedge v_y \in e_i\}}$$

Έστω $\boldsymbol{e_q^2}$ το καρτεσιανό γινόμενο μεταξύ των στοιχείων της ίδιας υπερακμής e_q, έτσι ώστε $e_q \times e_q = e_q^2$. Για κάθε ζεύγος κορυφών $(v_i, v_j) \in e_q^2$ δημιουργείται μια σχέση ομοιότητας $p: E \times V \times V \to R^+$. Η συνάρτηση $\boldsymbol{p}$ υπολογίζει τη σχέση ομοιότητας μεταξύ των κορυφών v_i, v_j εντός της υπερακμής e_q και υπολογίζεται ως εξής:

$$\boldsymbol{p(e_q, v_i, v_j) = w(e_q) \times h(e_q, v_i) \times h(e_q, v_j)}$$

Στη συνέχεια, μέσω μιας μήτρας $\boldsymbol{C}$ ορίζεται ένα μέτρο ομοιότητας βάσει του καρτεσιανού γινομένου, η οποία εξετάζει τις σχέσεις που περιέχονται σε όλες τις υπερακμές. Το σκεπτικό πίσω από αυτή τη διατύπωση βασίζεται στη συνεκτίμηση της συνύπαρξης των κορυφών v_i, v_j σε διαφορετικές υπερακμές, συσσωρεύοντας τις αντίστοιχες $p(\cdot, v_i, v_j)$ τιμές. Κάθε στοιχείο της μήτρας C υπολογίζεται ως εξής:

$$\boldsymbol{c(i,j) = \sum_{e_q \in E \wedge (v_i, v_j) \in e_q^2} p(e_q, v_i, v_j)}$$

Υπολογισμός Ομοιότητας βάσει του κατασκευασμένου Υπεργράφου (Hypergraph – Based Similarity)

Η ανά ζεύγη ομοιότητα που ορίζεται με βάση τις πράξεις των υπερακμών και του καρτεσιανού γινομένου, παρέχει ξεχωριστές και συμπληρωματικές πληροφορίες σχετικά με την πολλαπλότητα του συνόλου δεδομένων. Επομένως, και οι δύο πληροφορίες αξιοποιούνται με τον υπολογισμό μιας μήτρας $\boldsymbol{W}$ (affinity matrix) που συνδυάζει τις μήτρες C και S ως εξής:

$$\boldsymbol{W = C \circ S}$$

Επομένως, βάσει της μήτρας W, μπορεί να εκτελεστεί μια διαδικασία κατάταξης που οδηγεί σε ένα νέο σύνολο T διατεταγμένων λιστών. Αφότου οριστούν τόσο η είσοδος όσο και η έξοδος της διαδικασίας βάσει κάποιων κριτηρίων κατάταξης, η διαδικασία μπορεί να επαναληφθεί. Έστω ότι $T^{(0)}$ υποδηλώνει το σύνολο των διατεταγμένων λιστών που ορίζονται από το διάνυσμα χαρακτηριστικών, τότε μπορούμε να πούμε ότι το $T^{(t+1)}$ μπορεί να υπολογιστεί με βάση το $W^{(t)}$. Μετά από έναν ορισμένο αριθμό επαναλήψεων λαμβάνεται ένα τελικό σύνολο διατεταγμένων λιστών T_r.

Γενική περιγραφή της υπολογιστικής διαδικασίας που υλοποιήθηκε

Παρακάτω θα γίνει επεξήγηση της διαδικασίας που ακολουθήσαμε για την υλοποίηση του αλγορίθμου Log-based Hypergraph of Ranking References (LHRR).

Αρχικά, φορτώνουμε το σύνολο δεδομένων που βρήκαμε με την μέθοδο **load_dataset** και στη συνέχεια παίρνουμε ένα τυχαίο υποσύνολο του, μέσω της μεθόδου **get_subset_dataset**. Έπειτα, επιλέγουμε έναν τυχαίο αριθμό εικόνων στόχων μέσω της μεθόδου **get_target_indices**, η οποία μας επιστρέφει τις θέσεις που τους αντιστοιχούν μέσα στο υποσύνολο δεδομένων. Ακολούθως, με την βοήθεια του προεκπαιδευμένου νευρωνικού δικτύου **resnet50**, εξάγουμε τα χαρακτηριστικά διανύσματα για όλες τις εικόνες του υποσυνόλου δεδομένων. Αφού ολοκληρωθεί η διαδικασία εξαγωγής των χαρακτηριστικών διανυσμάτων, μέσω της μεθόδου **calculate_similarity_measures** υπολογίζουμε το μέτρο ομοιότητας για κάθε εικόνα και με τη βοήθεια της μεθόδου **get_ranked_list** κατατάσσουμε τις εικόνες (σε φθίνουσα σειρά) βάσει των μέτρων ομοιότητας τους.

Με την ολοκλήρωση των προαναφερθέντων προεπεξεργαστικών βημάτων, ακολουθεί η υλοποίηση των βασικών αλγοριθμικών βημάτων. Πρώτα γίνεται η κανονικοποίηση των διατεταγμένων λιστών μέσω της μεθόδου **normalize_ranked_list** και ύστερα με την μέθοδο **create_neighborhood_set_matrix** δημιουργούμε την μήτρα γειτνίασης N με τις k πιο όμοιες εικόνες. Στη συνέχεια, υπολογίζουμε την μήτρα H (continuous incidence matrix) μέσω της μεθόδου **calculate_continuous_incidence_matrix**, όπου έπειτα την χρησιμοποιούμε για να υπολογίζουμε την μήτρα ομοιότητας S (pairwise similarity matrix) με την μέθοδο **calculate_pairwise_similarity_matrix**. Ακολούθως, υπολογίζουμε το καρτεσιανό γινόμενο με την μέθοδο **calculate_cartesian_product** και τέλος υπολογίζουμε την μήτρα W (affinity matrix). Έχοντας υπολογίσει την μήτρα W, ενημερώνουμε την μήτρα T με τις αντίστοιχες τιμές της W και την κατατάσσουμε σε φθίνουσα σειρά. Η παραπάνω διαδικασία επαναλαμβάνεται για ένα ορισμένο πλήθος επαναλήψεων.

Με το πέρας των βασικών αλγοριθμικών βημάτων, εμφανίζουμε στην οθόνη τις k πιο όμοιες εικόνες για κάθε εικόνα στόχο (με την ακρίβεια του αλγορίθμου για την συγκεκριμένη εικόνα στόχο) με την μέθοδο **show_images**.

Περιγραφή διαδικασίας μέτρησης της ακρίβειας του αλγορίθμου

Ορίζουμε ως f μια δίκλαδη συνάρτηση

$$\boldsymbol{f(i)} = \begin{cases} \mathbf{1} & \boldsymbol{, αν\ η\ εικόνα\ i\ είναι\ της\ ίδιας\ κατηγορίας\ με\ την\ εικόνα\ στόχο} \\ \mathbf{0} & \boldsymbol{, αλλιώς} \end{cases}$$

και ως w μια συνάρτηση βάρους

$$\boldsymbol{w(i) = f(i)(n - i), όπου\ n\ το\ πλήθος\ των\ n\ πιο\ όμοιων\ γειτονικών\ εικόνων}$$

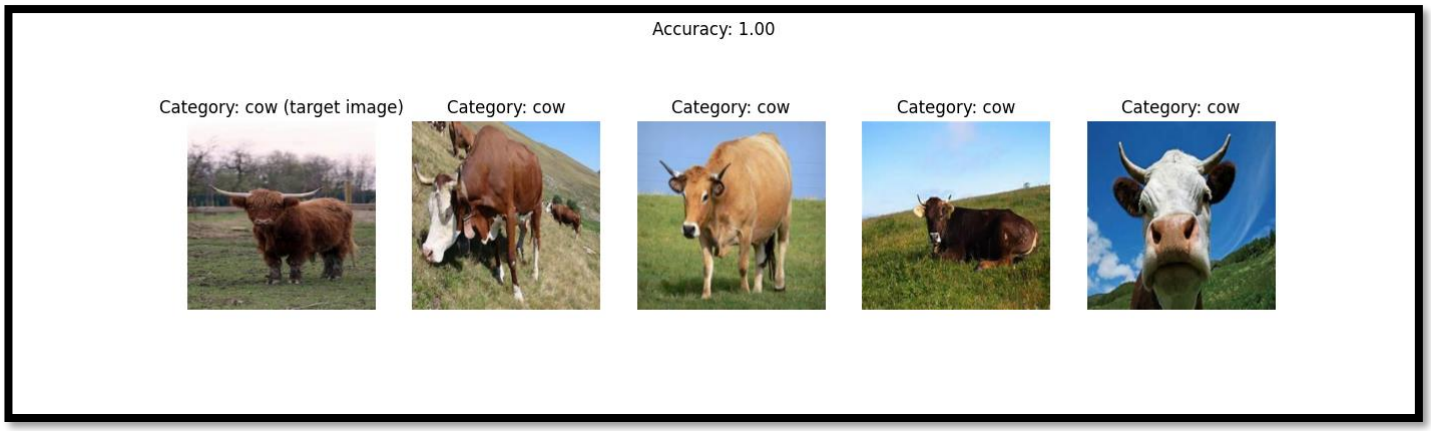
και υπολογίζουμε την ακρίβεια του αλγορίθμου για μια εικόνα στόχο με τον παρακάτω τύπο:

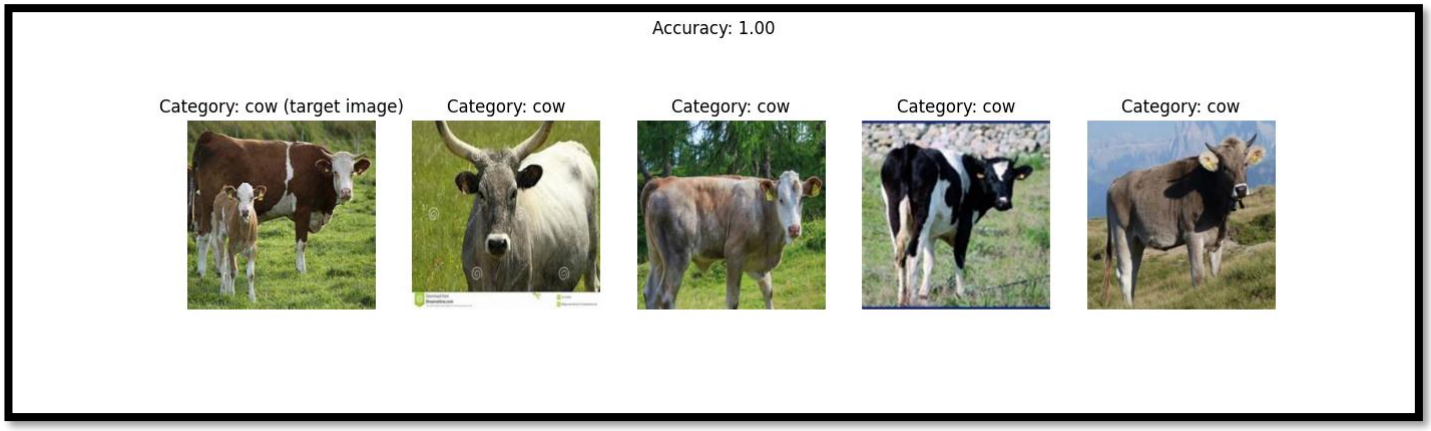
$$\boldsymbol{accuracy} = \left(\sum_{\boldsymbol{i=0}}^{\boldsymbol{n-1}} \boldsymbol{w(i)}\right) / \left(\sum_{\boldsymbol{i=1}}^{\boldsymbol{n+1}} \boldsymbol{i}\right)$$

Παραδείγματα εκτέλεσης

Εκτέλεση προγράμματος για 300 δείγματα εικόνων, 5 εικόνες στόχο και εμφάνιση των 5 πιο όμοιων εικόνων

```
dimit@MSI MINGW64 ~/Desktop/image-analysis-project-2023-2024/Project (main)
$ python main.py

Fetching the dataset...
Creating a subset of the dataset...
Selecting the target images...
Extracting the feature vectors...
Calculating the similarity measures...
Ranking the images based on the similarity measures...

Starting iteration 1/5:
        Normalizing the ranked list T...
        Creating the neighborhood set matrix N...
        Calculating the continuous incidence matrix H...
        Calculating the pairwise similarity matrix S...
        Calculating the Cartesian product C...
        Calculating the affinity matrix W...
        Updating the ranked list T with the new weights...
        Sorting the ranked list T...

Starting iteration 2/5:
        Normalizing the ranked list T...
        Creating the neighborhood set matrix N...
        Calculating the continuous incidence matrix H...
        Calculating the pairwise similarity matrix S...
        Calculating the Cartesian product C...
        Calculating the affinity matrix W...
        Updating the ranked list T with the new weights...
        Sorting the ranked list T...

Starting iteration 3/5:
        Normalizing the ranked list T...
        Creating the neighborhood set matrix N...
        Calculating the continuous incidence matrix H...
        Calculating the pairwise similarity matrix S...
        Calculating the Cartesian product C...
        Calculating the affinity matrix W...
        Updating the ranked list T with the new weights...
        Sorting the ranked list T...

Starting iteration 4/5:
        Normalizing the ranked list T...
        Creating the neighborhood set matrix N...
        Calculating the continuous incidence matrix H...
        Calculating the pairwise similarity matrix S...
        Calculating the Cartesian product C...
        Calculating the affinity matrix W...
        Updating the ranked list T with the new weights...
        Sorting the ranked list T...

Starting iteration 5/5:
        Normalizing the ranked list T...
        Creating the neighborhood set matrix N...
        Calculating the continuous incidence matrix H...
        Calculating the pairwise similarity matrix S...
        Calculating the Cartesian product C...
        Calculating the affinity matrix W...
        Updating the ranked list T with the new weights...
        Sorting the ranked list T...
```





Accuracy: 1.00

Category: dog (target image)

Category: dog

Category: dog

Category: dog

Category: dog

Accuracy: 1.00

Category: squirrel (target image)

Category: squirrel

Category: squirrel

Category: squirrel

Category: squirrel

Accuracy: 1.00

Category: squirrel (target image)

Category: squirrel

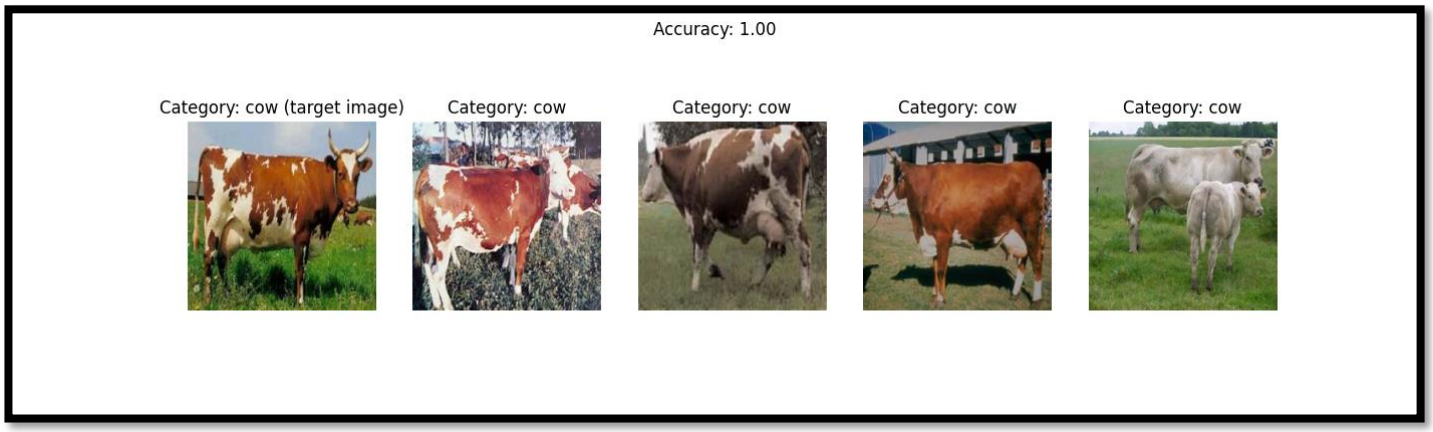
Category: squirrel

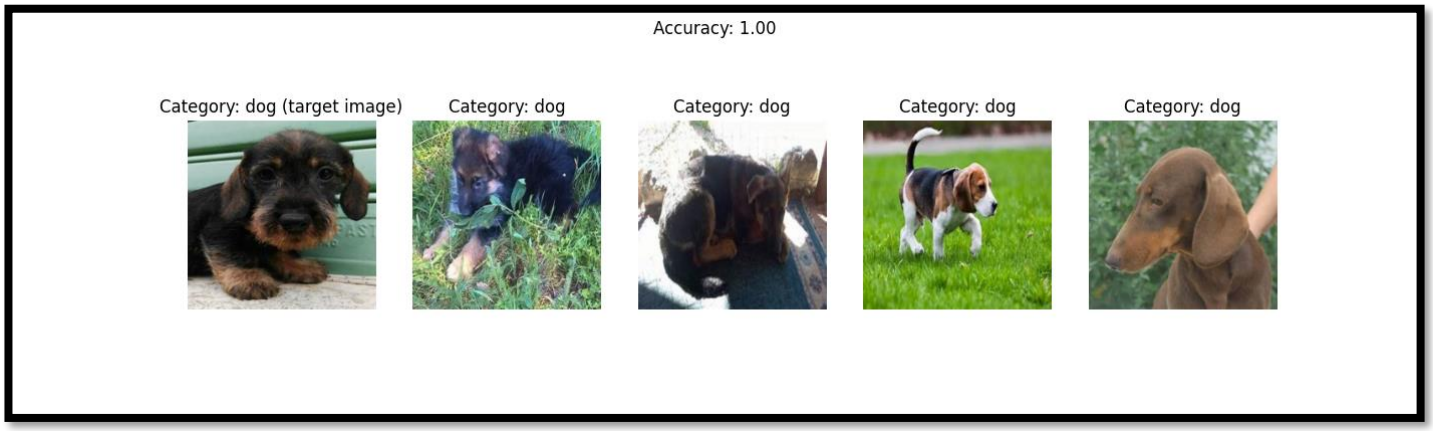
Category: squirrel

Category: squirrel

Εκτέλεση προγράμματος για 150 δείγματα εικόνων, 5 εικόνες στόχο και εμφάνιση των 5 πιο όμοιων εικόνων

```
dimit@MSI MINGW64 ~/Desktop/image-analysis-project-2023-2024/Project (main)
$ python main.py

Fetching the dataset...
Creating a subset of the dataset...
Selecting the target images...
Extracting the feature vectors...
Calculating the similarity measures...
Ranking the images based on the similarity measures...

Starting iteration 1/5:
        Normalizing the ranked list T...
        Creating the neighborhood set matrix N...
        Calculating the continuous incidence matrix H...
        Calculating the pairwise similarity matrix S...
        Calculating the cartesian product C...
        Calculating the affinity matrix W...
        Updating the ranked list T with the new weights...
        Sorting the ranked list T...

Starting iteration 2/5:
        Normalizing the ranked list T...
        Creating the neighborhood set matrix N...
        Calculating the continuous incidence matrix H...
        Calculating the pairwise similarity matrix S...
        Calculating the cartesian product C...
        Calculating the affinity matrix W...
        Updating the ranked list T with the new weights...
        Sorting the ranked list T...

Starting iteration 3/5:
        Normalizing the ranked list T...
        Creating the neighborhood set matrix N...
        Calculating the continuous incidence matrix H...
        Calculating the pairwise similarity matrix S...
        Calculating the cartesian product C...
        Calculating the affinity matrix W...
        Updating the ranked list T with the new weights...
        Sorting the ranked list T...

Starting iteration 4/5:
        Normalizing the ranked list T...
        Creating the neighborhood set matrix N...
        Calculating the continuous incidence matrix H...
        Calculating the pairwise similarity matrix S...
        Calculating the cartesian product C...
        Calculating the affinity matrix W...
        Updating the ranked list T with the new weights...
        Sorting the ranked list T...

Starting iteration 5/5:
        Normalizing the ranked list T...
        Creating the neighborhood set matrix N...
        Calculating the continuous incidence matrix H...
        Calculating the pairwise similarity matrix S...
        Calculating the cartesian product C...
        Calculating the affinity matrix W...
        Updating the ranked list T with the new weights...
        Sorting the ranked list T...
```





Accuracy: 1.00

Category: squirrel (target image)

Category: squirrel

Category: squirrel

Category: squirrel

Category: squirrel

Accuracy: 0.53

Category: dog (target image)

Category: squirrel

Category: cow

Category: dog

Category: dog

Accuracy: 1.00

Category: dog (target image)

Category: dog

Category: dog

Category: dog

Category: dog