Universitatea din București
Facultatea de Matematică și Informatică

LUCRARE DE LICENȚĂ

EXTRACTOR DE STIL ŞI DISTANȚA DINTRE STILURI

Utilizând Computer Vision și Machine Learning

COORDONATOR ȘTIINȚIFIC
Conf. Dr. Bogdan Alexe

ABSOLVENT
Lupașcu Marian

București,
iunie 2019

LUCRARE DE LICENȚĂ

Universitatea din București
Facultatea de Matematică și Informatică
Specializarea Informatică

Adobe România

Februarie – Iunie, 2019

„It's better to solve the right problem approximately than to solve the wrong problem exactly.”

— John Tukey, matematician

REZUMAT

Prezenta lucrare de licență propune automatizarea procesului de extracție a caracteristicilor de stil esențiale dintr-o interfață grafică și expunerea acestora pentru a fi folosite drept bază de construcție pentru o nouă interfață grafică. Utilitatea unei asemenea aplicații constă în construcția unui design pe baza altuia, astfel încât noul design să nu se realizeze de la zero, ci sa aibă un punct de start, anume caracteristicile extrase cu extractorul de stil. Extragerea caracteristicilor de stil ale unei interfețe grafice presupune colectarea culorilor, gradienților, stilului de text și simbolurilor[1]. Extractorul de stil este implementat ca o pereche de servicii unul de NodeJS și unul de Python care comunică cu un plugin de Adobe XD.

Serviciul de NodeJS realizează extracția de culori, gradienți și stil de text, inspectând și procesând proprietățile aferente elementelor ce compun interfața grafică cu un browser headless. Serviciul de Python realizează extracția de simboluri, procesând varianta imagistă a interfeței grafice randată tot cu un browser headless, folosind un algoritm de propunerea a regiunilor candidat pe bază de template matching, transformata Hough pentru cercuri, detectorul de muchii Canny, detectorul de text EAST, o rețea convoluțională folosită pentru clasificare, s.a. Iar pluginul de Adobe XD, singurul aflat pe mașina clientului, încarcă de la cele două servicii caracteristicile de design, astfel încât designerul interfeței grafice sa poată utiliza aceste caracteristici de design într-un mod optim.

În această lucrare de licență se introduce și un concept nou, anume distanța dintre stilul a două interfețe grafice, pe baza caracteristicilor de design extrase cu extractorul de stil. Aceasta distanță măsoară disimilaritatea dintre stilul a două interfețe grafice iar pentru calculul ei sunt folosite o serie de noțiuni din mai multe domenii precum: procesarea imaginilor, inteligența artificială, statistică și matematică. Distanța este calculată ca o compunere a distanțelor caracteristicilor

[1] *Prin simboluri se înțeleg elementele centrale din punct de vedere perceptiv dintr-o aplicație grafică, cum ar fi elementele repetitive, care contribuie la formarea stilului recurent al unei interfețe grafice.*

de design pe categorii, anume: (i.) pentru culori se folosește distanța Bhattacharyya dintre trei histograme de culoare, (ii.) pentru fonturi se folosește distanța Chord dintre imaginile vectorizate ale caracterelor randate cu fonturile analizate, (iii.) pentru simboluri se folosesc o serie de cinci rețele siameze, pentru fiecare categorie de simbol în parte.

Pentru a antrena cele cinci rețele siameze dar și o rețea convoluțională care face clasificare și eliminare de fals negativ construim un dataset format din șase clase, corespunzătoare a cinci tipuri de elemente de UI plus clasa zgomot.

Performanța timp a extractorului de stil este liniară cu complexitatea (numărul și tipul elementelor din arborele interfeței grafice) și dimensiunea interfeței grafice (în număr de pixeli), spre exemplu 14 secunde pentru o pagină de 588.000 pixeli (980 x 600) și 180 de secunde pentru o pagină de 35.294.400 pixeli (800 x 44118). Acuratețea extracției de culori, gradienți și fonturi este de 100%, iar a extracției de simboluri de aproximativ 80% cu recall de 92%.

În prezent extractorul și distanța de stil sunt implementate doar pentru pagini și aplicații web (interfețe grafice de tip web), însă conceptele și noțiunile introduse sunt general valabile și pentru celelalte tipuri de interfețe (spre exemplu de mobile).

ABSTRACT

This bachelor thesis proposes to automate the process of extracting essential style features from a graphical interface and exposing these features to be used as a basis for construction for a new graphical interface. The utility of such an application is to construct a design on the basis of another so that the new design does not come from scratch but has a starting point, namely the features extracted with the style extractor. Extracting the style features of a graphical interface involves collecting colors, gradients, text style and symbols[2]. The style extractor is implemented as a pair of NodeJS and Python services that communicate with an Adobe XD plugin.

The NodeJS service extracts colors, gradients and text style by inspecting and processing the properties of the elements that make up the graphical interface with a headless browser. The Python service performs symbol extraction, processing the imagery of the graphical interface rendered with a headless browser, using a candidate matching algorithm based on template matching, Hough transform for Circles, Canny Edge Detector, EAST Text Detector, convolutional network used for classification And the Adobe XD plug-in, the only one on the client machine, loads the design features from the two services so that the graphic designer can use these design features in an optimal way.

In this work, a new concept is introduced, namely the distance between the style of two graphical interfaces, based on the design features extracted with the style extractor. This distance measures the disparity between the style of two graphical interfaces, and for its calculation a number of concepts are used in several areas such as image processing, artificial intelligence, statistics and mathematics. The distance is calculated as a composition of the spacing of the design features by categories, namely: (i.) For colors we use the Bhattacharyya distance between three color histograms, (ii.) For fonts the distance used is the Chord distance between the vectorized images of the characters rendered with

[2] *Symbols are the perceptual central elements of a graphic application, such as repetitive elements, which help to form the recurrent style of a graphical interface.*

the fonts analysed, (iii.) for symbols we use a series of five siamese networks, for each category of symbol.

To train the five Siamese networks and a convoluted network that classifies and eliminate false negatives we build a dataset consisting of six classes corresponding to five types of UI elements plus the class noise.

The style extractor's time performance is linear with the complexity (number and type of graph interface tree elements) and the size of the graphical interface (in pixels), for example 14 seconds for a page of 588,000 pixels (980 x 600) and 180 seconds for a 35,294,400 pixel (800 x 44118) page. The accuracy of extraction of colors, gradients and fonts is 100%, and symbol extraction of about 80% with 92% recall.

Currently, the extractor and style spacing are only deployed for web pages and applications (web-based graphical interfaces), but the concepts and notions introduced are generally valid for other types of interfaces (such as mobile).

CUPRINS

LISTA FIGURILOR 1

LISTA TABELELOR 3

1. INTRODUCERE 5

1.1. PREZENTAREA PROBLEMEI 6
1.2. DEFINIREA CONCEPTULUI DE CARACTERISTICĂ / FEATURE DE DESIGN 6
1.3. MOTIVAȚIA 8
1.4. STRUCTURA TEZEI 8
1.5. CONTRIBUȚIA ÎN DOMENIU 8

2. PLUGINUL DE ADOBE XD 9

2.1. CONTEXT 10
2.2. MODUL DE FUNCȚIONARE 11
2.3. DEMERSUL IMPLEMENTĂRII 12
2.3.1. XMLHttpRequest 12
2.3.2. Browser Headless 13
2.3.3. Extensie de Google Chrome 14
2.3.4. Serviciu extern 15

3. COMPONENTELE THIRD-PARTY 17

3.1. SERVICIUL DE NODEJS 18
3.2. DEPENDINȚELE SERVICIULUI DE NODEJS 18
3.3. EXTRACȚIA CULORILOR ȘI GRADIENȚILOR 18
3.3.1. Colectarea culorilor 18
3.3.2. Sortarea culorilor 19
3.3.3. Colectarea gradienților 20
3.4. EXTRACȚIA DE STIL DE TEXT 20
3.5. SERVIREA CERERILOR 20
3.6. SERVICIUL DE PYTHON 21
3.7. DEPENDINȚELE SERVICIULUI DE PYTHON 21
3.8. EXTRACȚIA SIMBOLURILOR CIRCULARE 21
3.8.1. Filtru median de blurare 22
3.8.2. Transformata Hough pentru cercuri 22
3.9. EXTRACȚIA SIMBOLURILOR REGULATE 23
3.9.1. Detectorul de text EAST 23
3.9.2. Reconstrucția imaginilor 24
3.9.3. Algoritmul Canny pentru detecția de muchii 24
3.9.4. Algoritmul de template matching 25
3.9.5. Non Maximum Suppression pentru detecția de obiecte 26
3.9.6. Propunerea regiunilor candidat 26
3.10. ELIMINAREA REDUNDANȚEI 29
3.10.1. Indicele de similaritate structurală (SSIM) 29
3.10.2. Indicele de similaritate a culorii medii 29
3.11. CLASIFICAREA REZULTATELOR ȘI ELIMINAREA FALSULUI NEGATIV 30
3.11.1. Noțiuni teoretice 30
3.11.2. Construcția datasetului 32
3.11.3. Arhitectura 33
3.11.4. Rezultate 34
3.12. PERFORMANȚA ȘI ACURATEȚEA EXTRACȚIEI 35
3.13. PERFORMANȚA TIMP 35

4. DISTANȚA DINTRE STILURI **37**
4.1. DISTANȚA DINTRE CARACTERISTICILE DE TIP CULOARE 38
4.1.1. Histograma de culoare *38*
4.1.2. Distanța dintre histogramele de culoare *39*
4.2. DISTANȚA DINTRE CARACTERISTICILE DE TEXT DE TEXT 41
4.2.1. Distanța dintre fonturi *41*
4.2.2. Distanța dintre culorile elementelor de tip text *43*
4.2.3. Compunerea distanței de text din caracteristicile secundare *45*
4.3. DISTANȚA DINTRE FEATURE-URILE SIMBOL 47
4.3.1. Rețea siameză *47*
4.3.2. Construcția datasetului *47*
4.3.3. Arhitectura *48*
4.3.4. Funcția de loss *49*
4.3.5. Formalizare și rezultate *50*
4.3.6. Distanța între simbolurile interfețelor grafice *52*
4.4. FORMALIZAREA ȘI IMPLEMENTAREA DISTANȚEI DE STIL 55
4.4.1. Formalizare *55*
4.4.2. Implementare *55*
4.4.3. Rezultate *56*
CONCLUZII **59**
BIBLIOGRAFIE **63**
ANEXĂ **65**

LISTA FIGURILOR

Figura 1.1 – Trei capturi de ecran ale aceluiași site 8
Figura 2.1 – Tutorialul de Adobe XD 10
Figura 2.2 – Extractorul de stil implementat în Adobe XD 11
Figura 3.1 – Spațiile de culoare RGB (a.) și HSV (b.) 19
Figura 3.2 – Ordonarea culorilor după hue (nuanță) 20
Figura 3.3 – Arhitectura detectorului de text EAST [2] 23
Figura 3.4 – Perceptronul 30
Figura 3.5 – Exemplificarea operației de pooling 31
Figura 3.6 – Distribuția datelor înainte (stânga) și după (dreapta) eliminarea redundanței 32
Figura 3.7 – Arhitectura rețelei convoluționale care clasifică simbolurile 33
Figura 3.8 – Matricele de confuzie **(a)** nenormalizată și **(b)** normalizată 34
Figura 3.9 – Evoluția funcției de loss **(a)** și a acurateței **(b)** 34
Figura 3.10 – Distribuțiile timpilor de execuție între entități 35
Figura 3.11 – Corelația dintre timpul de execuție și dimensiunea paginilor analizate 36
Figura 4.1 – Histogramele de culoare extrase de pe pagina Flickr 39
Figura 4.2 – Spațiul de culoare CIELAB 43
Figura 4.3 – Compoziția datasetului pentru rețelele siameze 48
Figura 4.4 – Arhitectura rețelei siameze 49
Figura 4.5 – Graficul evoluției funcției de loss pentru rețelele siameze ale claselor radioButton și checkBox 50

LISTA TABELELOR

TABELUL 3.1 – HIPERPARAMETRII ALGORITMULUI DE PROPUNERE A REGIUNILOR CANDIDAT 27
TABELUL 3.2 – REZULTATELE ALGORITMULUI DE PROPUNERE A REGIUNILOR CANDIDAT 28
TABELUL 3.3 – DISTRIBUȚIA DATELOR ÎN CADRUL DATASETULUI 32
TABELUL 3.4 – PERFORMANȚA TIMP A EXTRACTORULUI DE STIL, ÎN SECUNDE..... 35
TABELUL 4.1 – DISTANȚELE DINTRE CARACTERISTICILE DE CULOARE DE PE DOUĂ PERECHI DE PAGINI WEB. 40
TABELUL 4.2 – REZULTATELE DISTANȚEI DINTRE DOUĂ FONTURI. 42
TABELUL 4.3 – DIFERENȚELE ÎNTRE SPAȚIILE DE CULOARE RGB ȘI CIELAB. 44
TABELUL 4.4 – DISTANȚELE DINTRE CARACTERISTICILE DE DESIGN A DOUĂ PERECHI DE PAGINI WEB. 46
TABELUL 4.5 – DOUĂ GRUPURI DIN DATASETUL REȚELEI SIAMEZE DE GERARE A DISTANȚELOR DINTRE ELEMENTELE DE TIP CHECK BOX 47
TABELUL 4.6 – O SERIE DE CÂTEVA DISTANȚE ÎNVĂȚATE DE REȚELELE SIAMEZE CORESPUNZĂTOARE CLASELOR DE SIMBOLURI 52
TABELUL 4.7 – REZULTATELE DISTANȚEI DINTRE SIMBOLURI 54
TABELUL 4.8 – REZULTATELE DISTANȚEI DE STIL PENTRU O SERIE DE PAGINI WEB 57

Capitolul 1

INTRODUCERE

În prima parte se explică problema abordată, se definește conceptul de caracteristică de design, se enunța motivația alegerii acestei teme ca proiect de licență și se oferă o linie de perspectivă asupra structurii tezei. În partea a doua se oferă o descriere a contribuției proprii în domeniu, prin introducerea conceptului de distanță dintre stiluri.

INTRODUCERE

Determinarea caracteristicilor de design dintr-o interfață grafică reprezintă o informație valoroasă ce oferă un punct de start bun pentru construcția unui nou design plecând de la un design deja existent. Acest lucru oferă un avantaj enorm unui designer deoarece acesta poate trece peste partea de colectare a informațiilor ce alcătuiesc stilul noului design, într-o manieră elegantă, anume colectarea acestora cu un extractor de stil. În plus având aceste caracteristici de design se pot introduce noi concepte matematice precum cel de distanță dintre stilul a două interfețe grafice.

1.1. Prezentarea problemei

Când se dezvoltă un nou design, timpul este o resursă critică, iar luarea unei decizii cu privire la conceperea structurii unei interfețe grafice pentru o aplicație (de exemplu de web, de mobile, etc.) în care merită investit timp este un lucru dificil. Piața actuală este atât de competitivă încât nu permite irosirea timpului pe prototipuri de design neglijabile. Principiul Pareto afirmă ca: Un procent mare din feature-urile unui sistem voluminos sunt generate de un număr mic de variabile [13]. Generalizând și extrapolând, 80% din stilul general al unei interfețe grafice este compus de 20% din caracteristicile de design, atunci restul de 80% din caracteristicile de design trebuie reevaluate și verificate pentru a vedea dacă merită sau nu să fie integrate în design. Acest lucru ne conduce la următoarea întrebare: *pe ce caracteristici de design ar trebui să ne concentrăm?* Înainte de a răspunde la acesta întrebare, trebuie mai întâi să definim conceptul de caracteristică de design. Ca o consecință directă va trebui să extragem caracteristicile de design cu impactul cel mai mare.

În plus, atunci când se face căutare automată (web crawling) de interfețe cu design asemănător cu cel la care se lucrează se dorește existența unei măsuri a similarității sau a disimilarității dintre designul celor două interfețe analizate. De aceea în prezenta lucrare de licență se introduce un concept nou, cel de distanță dintre stiluri, care măsoară disimilaritatea dintre stilul a două interfețe grafice.

1.2. Definirea conceptului de caracteristică / feature de design

O caracteristică sau feature de design reprezintă o componentă structurală dintr-o interfață grafică care definește o parte din stilul acesteia.

În acest sens culorile care compun elementele grafice ale unei interfețe contribuie în mod semnificativ la definirea stilului, deci sunt un feature de design important. Un alt aspect

semnificativ îl constituie elementele de design care conțin text, aici poate varia fontul, culoare și mărimea iar odată cu acestea variază și stilul general al interfeței grafice deci și stilul de text este un feature de design ce trebuie luat în calcul.

În plus față de culori și stil de text, stilul designului general este definit de multe alte elemente structurale ale interfeței grafice. Aici există o întreagă discuție legată de ce anume definește designul general. În cele ce vor urma vom răspunde la întrebarea din **Prezentarea problemei** și anume: *care sunt caracteristicile de design pe care ar trebui să ne concentrăm?*

Definim noțiunea de **simbol** al unei interfețe grafice ca fiind un element sau un grup de elemente arhitecturale care sunt centrale din punct de vedere perceptiv într-o aplicație grafică. Elementele repetitive, care descriu stilul recurent al interfeței sau un element singular care atrage multa atenție asupra sa sunt niște simboluri veritabile. Prin definiție simbolurile contribuie cu un impact major la stilul general unei interfețe grafice, deci simbolurile sunt design features. Considerând că o imagine nu este un element refolosibil, din punct de vedere al designului, înseamnă că elementele de tip imagine nu vor fi considerate simboluri. În continuare pe post de simboluri vom considera elementele de UI (user interface), deoarece sunt elementele care se repeta cel mai mult în stilul general al unei aplicații, cu un impact perceptiv semnificativ.

Deci feature-urile de design pe care ar trebuie să ne concentrăm sunt culorile, fonturile și simbolurile, care compun mare parte din stilul general, rămânând componentele structurale care nu sunt centrale și nu atrag atenție în mod direct, deci nu prezintă interes suficient pentru a fi analizate.

Remarcăm faptul că textul ca și conținut, din orice element structural al unei interfețe grafice nu este un design feature cu impact mare, ci numai felul în care este randat reprezintă un feature de design care contează. Dacă am pleca de la premisa că textul ar fi caracteristică de design atunci orice modificare în text introduce un nou feature de design, ceea ce este în general fals, întrucât modificarea textului lasă în mare parte stilul general nemodificat. Spre exemplu in **Figura1.1**, nu există diferențe de design în ceea ce privește stilul general al conținutului interfețelor grafice deși tot textul este schimbat între cele trei versiuni ale interfețelor grafice, aici pagini web.

Figura 1.1 *– Trei capturi de ecran ale aceluiași site*

Se poate observa **în Figura 1.1** ca deși paginile web IMDb (Hidden Figures) au text complet diferit, în stânga - engleză, în mijloc - franceză, în dreapta - spaniolă, stilul general este constant.

1.3. Motivația

Am ales acest proiect, de extragerea a stilului dintr-o aplicație grafică și de construcție a unei noi distanțe, datorită faptului că este o problema unică. Acest proiect permite folosire tehnicilor din mai multe discipline (statistică, machine learning, computer vision, matematică, design, etc.) și îmbinarea algoritmilor din disciplinele enumerate anterior în diferite moduri.

1.4. Structura tezei

În cel de-al doilea capitol este prezentat pluginul de Adobe XD, se oferă un context al acestei aplicații și este prezentat demersul implementării extractorului. În cel de-al doilea capitol sunt prezentate cele două servicii externe implicate în procesul de extracție: modul de lucru, dependințele și algoritmii utilizați. Iar în ultimul capitol se formalizează distanța dintre stilul a două interfețe grafice și se prezintă o schiță a implementării acestei distanțe.

1.5. Contribuția în domeniu

Această lucrare nu prezintă numai algoritmi teoretici, ci și rezultate aplicate. Extractorul de stil extrage culorile, gradienții și fonturile nu din varianta declarată a conținutului HTML ci din varianta dinamică, cea pe care utilizatorul o vede în browser, cu HTML-ul dinamic încărcat, folosind un browser Google Chrome Headless [18]. Simbolurile sunt extrase din varianta imagistă a paginii web folosind o serie de algoritmi de computer vision și inteligență artificială. Pe lângă extractorul de stil, prezenta lucrare de licență definește un nou concept teoretic: distanța dintre stilul a două interfețe grafice. Această distanță se construiește pe baza feature-urilor de design extrase cu extractorul de stil (clientul primește cate un JSON pentru fiecare pagina de pe care face extracție de stil apoi aceste fișiere constituie inputul pentru distanță). Ideea de baza a acestei distanțe este de a găsi o măsură a similarității / disimilarității dintre stilul a două interfețe grafice.

Capitolul 2

PLUGINUL DE ADOBE XD

În prima parte a acestui capitol se prezintă în mod sumar aplicația Adobe XD. În a doua parte se descrie modul de funcționare al pluginului: (i.) realizarea unei comunicări cu clientul, (ii.) expunerea într-o manieră organizată a caracteristicilor de design, (iii.) construirea inputul pentru distanța de stil. Iar în partea de final a acestui capitol se descrie demersul implementării și motivul principal pentru care funcționalitatea extracției nu este implementată local ci extern ca o pereche de servicii.

PLUGINUL DE ADOBE XD

Un plugin/plug-in este o componentă software care adaugă un feature nou unei aplicații. În plus aplicația trebuie să permită implementarea plugin-urilor. Interacțiunea utilizatorilor cu extractorul de stil se face doar prin pluginul de Adobe XD.

2.1. Context

Adobe Experience Design (XD) este o aplicație bazată pe vectori care oferă o soluție UI/UX all-in-one pentru proiectarea de aplicații web, aplicații mobile și nu numai, toate într-o aplicație puternică [3].

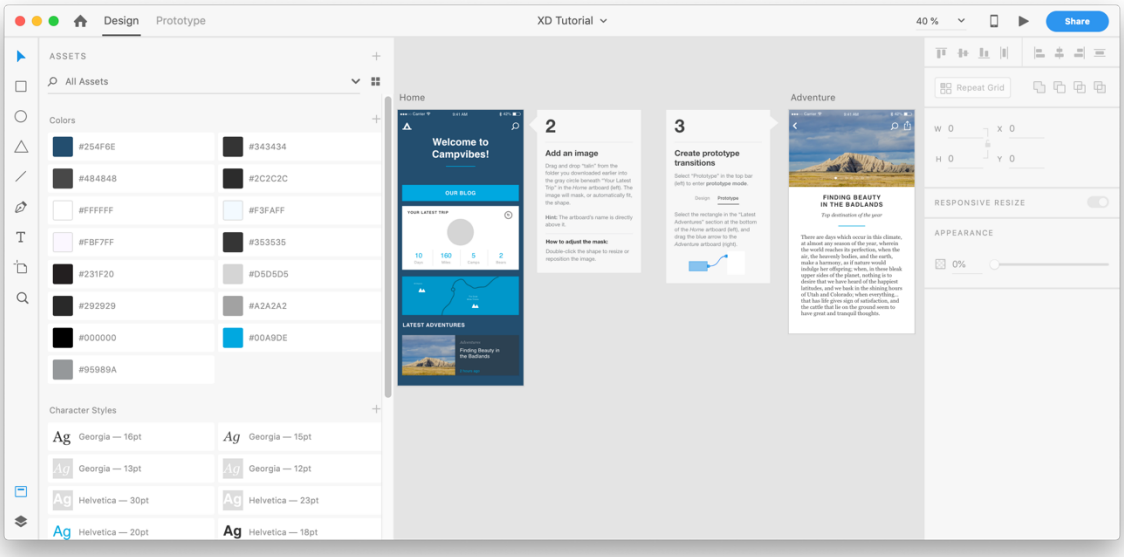


Figura 2.1 *– Tutorialul de Adobe XD*

În **Figura 2.1** este prezentată o captură de ecran cu **Adobe XD**, în partea stângă se afla Assets Panel, care este împărțit în trei categorii: culori, fonturi și componente (în figură sunt populate doar culorile și fonturile). Central se află suprafața de lucru, care conține două artboard-uri.

În continuare exemplificăm câteva concepte din Adobe XD, necesare pentru înțelegerea modalității de funcționare a extractorului de stil implementat ca plugin.

- **Assets Panel** – funcționalitate oferită de XD pentru gestionarea culorilor, a gradienților, a stilurilor de text și a componentelor. În Assets Panel utilizatorul poate insera diferite caracteristici de design pe care le poate refolosi ulterior cu ușurință.
- **Artboard (AB)** – reprezintă baza pentru orice fel de interfață care se dorește a fi proiectată, există o suita întreagă de artboarduri, pentru web pentru mobile sau custom.

[3] *Adobe Experience Design (XD) - adobe.com/products/xd.html*

- **Grup** – reprezintă o combinație de mai multe elemente (forme geometrice, text, etc) astfel încât acestea să fie tratate ca o singură entitate.

2.2. Modul de funcționare

Aplicația în Adobe XD este implementată ca un plugin care interacționează cu clientul [17]. Declanșarea pluginului se face din meniul de pluginuri din XD. La click pe ***Style Extractor*** (denumirea pluginului) se afișează un dialog care permite utilizatorului să introducă URL-ul paginii de pe care dorește să facă extracția (**Figura 2.2**). După confirmarea extracției (la click pe OK) se trimite URL-ul către cele două servicii, de NodeJS și Python, unde are loc esența extracției caracteristicilor de design (mai multe despre cele două servicii și despre comunicarea dintre entități se află în capitolul următor). Din acest punct de vedere, prezentul plugin poate fi privit ca un ambalaj al extractorului de stil deoarece extracția propriu zisă are loc extern în perechea de servicii.

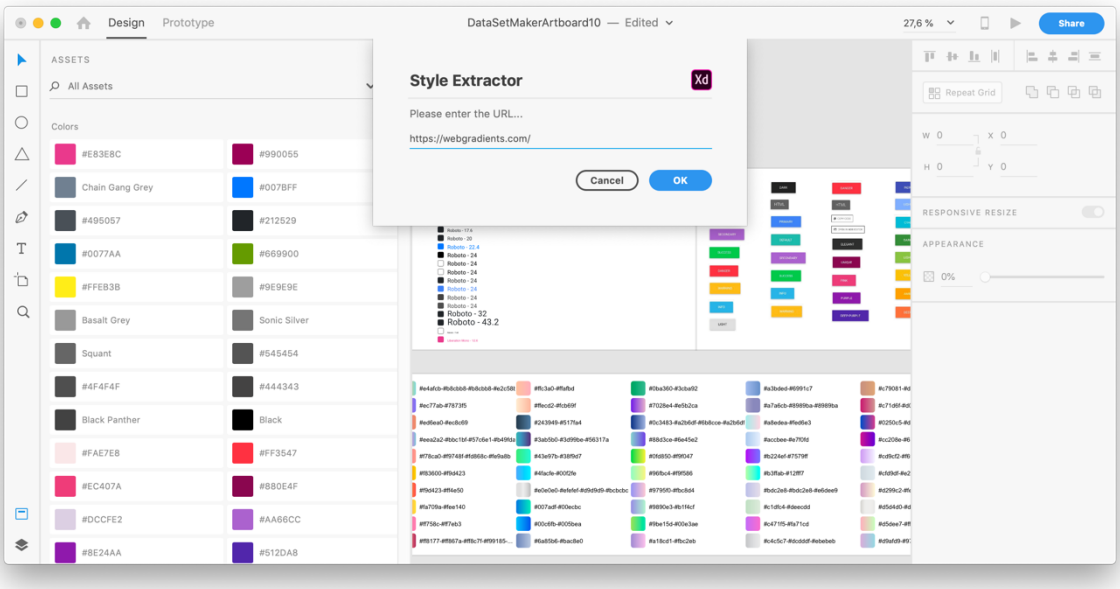


Figura 2.2 *– Extractorul de stil implementat în Adobe XD*

După ce se finalizează procesul de extracție din cele două servicii, se trimit două fișiere JSON către plugin unde se randează caracteristicile de design. De la serviciul de NodeJS se primește un JSON cu informațiile de culoare, gradienți și stil de text, iar de la serviciul de Python un JSON cu informațiile despre simboluri. Pluginul populează Assets Panelul cu culorile, gradienții și stilul de text extrase și creează câte un artboard pe care randează toate aceste caracteristici de design pe categorii. Aceste categorii sunt: culori, culori de fundal, gradienți liniari (pentru gradienții radiali încă nu există API așa că aceștia sunt expuși numai Assets Panel), stilurile de text și simbolurile. Ca observație simbolurile nu sunt puse în Assets Panel

deoarece API-ul pentru pluginuri expus de Adobe XD încă nu are aceasta funcționalitate implementată (în prezent se află pe un branch în lucru, însă nu este gata).

În plus pluginul are implementată și funcționalitatea de construcție a inputului pentru distanța dintre stiluri. Inputul pentru distanță într-o pereche de fișiere JSON. Fiecare fișier este format prin cumularea celor doua fișiere de același tip primite de la servicii. Pluginul salvează acest fișier JSON într-un folder pe care utilizatorul îl alege din sistemul sau de fișiere, cu titlul paginii de pe care s-a făcut extracția și cu extensia json.

Mai multe informații despre manifestul pluginului și despre structura fișierelor JSON se găsesc în **Anexă**.

2.3. Demersul implementării

În continuare voi prezenta posibilele abordări de extracție, enumerând pentru fiecare avantajele și dezavantajele.

2.3.1. XMLHttpRequest

XMLHttpRequest [19] este un wrapper peste XML care permite stocarea textuala organizată și flexibilă. XMLHttpRequest este folosit foarte mult în programarea AJAX. În ciuda numelui său, XMLHttpRequest poate fi folosit pentru a prelua orice tip de date, nu doar XML, și suportă alte protocoale decât HTTP (inclusiv file și ftp).

Avantaje:

- Este rapid și clar, făcându-se doar un GET la serverul pagii web cerute. Legat de timpul de răspuns acesta variază pană la 1,248 secunde (pentru Youtube) sau 0,744 secunde (pentru imgur.com). Bineînțeles acest lucru va depinde foarte mult de calitatea rețelei, lățimea de bandă și multe alte variabile care nu țin efectiv de Adobe XD.
- Este scalabil și implementat de majoritatea browserelor web, făcând excepție numai Safari pentru iPhone. Nu este neapărat un avantaj deoarece va fi folosit intern în pluginul de Adobe XD, însă acest lucru demonstrează că este o abordare folosită.

Dezavantaje:

- Reține informația sub formă de text DOMString (UTF-16 String (XML care reține text)) ceea ce nu permite folosirea selectorilor de CSS precum .style sau getComputedStyle() pentru extragerea de informație deci trebuie folosit un DOMParser.

- În cazul în care un site are un preloader această metodă va încărca pe post de cod sursă HTML codul preloaderului sau al paginii care conține preloaderul.

Cel de-al doilea dezavantaj conduce la problema HTML-ului dinamic – în cazul în care există conținut încărcat dinamic (injectat de codul din backend) acesta nu va putea fi extras parsând conținutul HTML, deoarece nu se va obține nimic relevant. Acesta reprezintă principalul motiv pentru care nu se va aborda această metodă.

2.3.2. Browser Headless

Un browser headless [18] este un browser fără interfață grafică, care oferă un control automat pentru manipularea paginilor web și este similar cu un browser. Oferă același mediu de randare al paginilor, fiind capabil să redea și să înțeleagă conținutul HTML în același mod în care o face și un browser normal, inclusiv elemente de styling sau conținut dinamic de JavaScript sau AJAX.

Extragerea de informații cu un browser headless ca **PhantomJS** [20]**, Chrome headless** [18], **CasperJS, ZombieJs**, etc. este foarte ofertantă deoarece acesta permite randarea paginilor în modul dinamic (pagina arată fix ca în browserul utilizatorului). Oferă totodată un **API** pentru **JavaScript** și **Python**.

Avantaje:

- Un browser headless oferă posibilitatea de simula un browser, având URL-ul unei pagini web neîncărcate în browserul utilizatorului, se poate după un request de GET să se facă extracție de informație;
- Un al avantaj este că un browser headless oferă suport pentru varianta în care se face captură de ecran la pagina web apoi permițând o analiză pe varianta tip imagine (lucru ofertant în cazul unei abordări ce folosește Computer Vision pentru extracția de simboluri);
- Un avantaj important este dat de posibilitatea încărcării paginilor ce conțin elemente cu conținut dinamic. Se rulează în spate toată logica paginii, apoi se randează fie logic pagina, fie sub forma de imagine.

Dezavantaje:

- Un dezavantaj major adus de utilizarea unui browser headless este dat de dependința acestuia de crearea de procese copil (child_process), instanța de browser headless pentru funcționare are nevoie de un proces nou, lucru care intră în contradicție cu modul în

care poate funcționa un plugin de Adobe XD [17]. Presupunând că un plugin ar putea să creeze un proces, atunci ar exista numeroase leak-uri de securitate. Acest dezavantaj va constitui motivul principal pentru care nu vom folosi această abordare direct în pluginul de Adobe XD (adică nu este fezabilă o implementare locală cu un browser headless).

2.3.3. Extensie de Google Chrome

Comunicarea browserului în care este randată pagina cu Adobe XD, printr-un **WebSocket** [22], oferă o perspectivă nouă. Acest lucru permite extragerea de informații direct din browser făcând lucrurile extrem de rapide [21].

WebSocket [22] este un protocol de comunicare de tip duplex pe o singură conexiune TCP. Acesta este capabil să realizeze o conexiune între un browser web și un server web, în dublu sens, însă cu o viteza mai mică decât o comunicare într-un singur sens printr-un protocol de tip HTTP pooling.

Avantaje:

- Este o modalitate facilă de a rezolva problema extracției de stil, pagina web fiind randată în browserul utilizatorului, cu tot HTML-ul dinamic încărcat, cu toate event-urile declanșate pană la un punct (dorit de utilizator), din extensie putând fi încărcat cu rapiditate tot DOM-ul actual al paginii web.
- Această variantă de extracție este mai generală, și permite capturarea informației pe care utilizatorul o deține în momentul extracției. Pe când un browser headless, din motive de securitate și confidențialitate, în momentul interogării unei pagini se construiește pagina exact cum este randată în modul incognito (de exemplu un utilizator este logat pe o pagina web și face extracție – în varianta cu browser headless acesta ar primi informațiile care îi sunt oferite normal în modul incognito, însă cu o extensie de Google Chrome acesta primește informația curentă, pagina analizată fiind cea din browserul utilizatorului, nu extern).

Dezavantaje:

- Nu există posibilitatea de a comunica direct între Plugin-ul de XD și Extensia de Chrome. Singura posibilitate fezabilă în XD este prin WebSocket-uri, deci în continuare se va vorbi doar despre comunicare prin WebSocket-uri. În extensiile de Chrome nu poate fi implementat un Server WebSocket ci doar funcționalitate de Client WebSocket,

similar în pluginul de Adobe XD. Acest lucru se întâmplă datorită paletei limitate de API-uri oferită de XD Plugin-urilor și de Chrome extensiilor. O modalitate care rezolvă acest dead-end este crearea un server care funcționează ca un bridge între cei doi clienți (Plugin și Extensie). Acest lucru creează următoarele probleme:

- Cum știm ce plugin Client să conectăm la extensia Client corespunzătoare?
- Un flow ciudat în extracția de stiluri, fiind necesară instalarea unei extensii în browserul utilizatorului (se plimbă utilizatorului care dorește să facă extracție de stil de la XD la Chrome și invers);
- Dependența de extensia de Chrome (care poate fi instalată și pe Firefox), imposibil de instalat pe un Safari spre exemplu.

Limitări:

- Posibilitatea folosirii unei extensii atrage de la sine mutarea a mare parte din funcționalitate din plugin în extensie, ceea ce conduce la eliminarea unor constrângeri generate de paleta limitată de API-uri oferită plugin-urilor de către Adobe XD, dar totodată atrage de la sine limitările cu care vin extensiile de Chrome (de exemplu, imposibilitatea de a crea un Server WebSocket).

Acest lucru (imposibilitate de a crea un Server WebSocket în nici una din părți – Plugin sau Extensie) ne va conduce la ultima alternativă fezabilă rămasă, anume serviciu extern.

2.3.4. Serviciu extern

Serviciul nu va fi implementat pe mașina utilizatorului de XD și va fi hostat pe un server extern (AWS, Azure, etc.) [23]. Pluginul de Adobe XD se va conecta la serviciu va face un GET cu URL-ul dat ca input și va aștepta răspuns de la serviciu. Implementarea unui serviciu extern va rezolva mare parte din dezavantajele metodelor enumerate mai sus însă acest lucru va atrage de la sine niște dezavantaje.

Avantaje:

- Un serviciu extern de NodeJS sau Python permite utilizarea fără constrângeri a tuturor funcționalităților din NodeJS sau Python (spre exemplu crearea unui Server WebSocket sau crearea de procese). Reușind să creăm procese vom folosi un browser headless (Chrome headless) pentru manipularea paginilor web, acest lucru va rezolva problema paginilor care conțin HTML dinamic, randând pagina în background exact cum arată pe pagina utilizatorului după încărcare.

- Un alt avantaj este oferit de independența de mașina utilizatorului de Adobe XD. Toate calculele și extracțiile făcându-se pe server, se poate estima pentru fiecare cerere timpul de răspuns. În plus, în acest mod se pot colecta date legate despre cererile utilizatorilor, date care pot fi folosite ulterior în crearea de statistici și predicții pentru utilizatori cu scopul de a-i servi mai rapid.

Dezavantaje:

- Implementarea ca serviciu extern vine cu costurile de întreținere și mentenanță specifice unui serviciu.
- Un alt dezavantaj îl constituie problemele de securitate, deoarece toată funcționalitatea este mutată din plugin în serviciul extern.

Însă comparând rezultatele de la fiecare metodă obținem că această metodă este cea mai promițătoare, fiind singura capabilă să realizeze o comunicare optimă între plugin și entitatea de prelucrare, totodată reușind să prelucreze HTML-ului dinamic.

În continuare vom prezenta funcționalitatea din serviciile implementate pentru extracție.

Capitolul 3

COMPONENTELE THIRD-PARTY

Componentele software third-party sau terță-parte implicate în procesul de extracție de caracteristici de stil sunt serviciile de NodeJS și de Python. În prima parte a acestui capitol vom prezenta extracția de culori, gradienți și stil de text care se realizează pe serviciul de NodeJS. Urmând ca în a doua parte a acestui capitol să prezentăm extracția de simboluri din varianta de tip imagine a interfeței grafice, extracție care se realizează pe serviciul de Python

3.1. SERVICIUL DE NODEJS

Un serviciu este o funcționalitate sau un set de funcționalități software a cărui scop constă în servirea clienților cu obiective diferite într-un mod diferit [23]. Serviciul de NodeJS implementează funcționalitatea de extracție de culoare, gradienți și stil de text.

3.2. Dependințele serviciului de NodeJS

Dependințele principale ale serviciului de NodeJS sunt:

- WebSocket Server (pachetul ws din npm) este folosit pentru a cerea un server cu scopul de a aștepta conexiuni cu clienții [24].
- Google Chrome Headless Browser (pachetul puppeteer din npm) este folosit pentru manipularea paginilor web, cu scopul de a obține designul paginilor web [18].
- 18181 color-names (pachetul color-name-list din npm), aceasta este o bază de date cu nume de culori. În prezent acoperă ~0.11% din totalul culorilor (18.181 nume de culori din 16.777.216 culori posibile din spațiul RGB) [26].

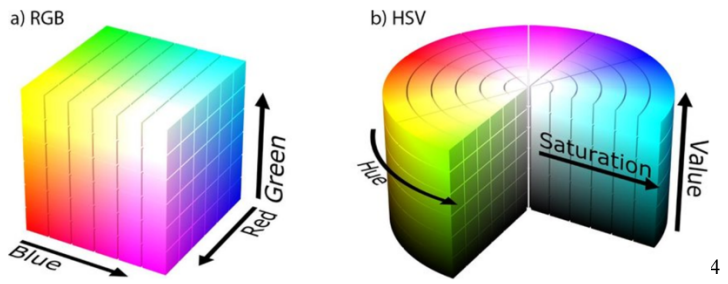
3.3. Extracția culorilor și gradienților

Acest pas al procesului de extracție presupune colectarea și procesarea caracteristicilor de stil de tip culoare. Pasul de procesare este necesar pentru a elimina culorile duplicat astfel încât răspunsul trimis pluginului de Adobe XD să nu conțină informație redundantă. O altă utilitate a pasului de procesare este sortarea culorilor. În continuare vom detalia acești doi pași.

3.3.1. Colectarea culorilor

Extracția caracteristicilor de tip de culoare este împărțita în două tipuri de extracție de culoare, în funcție de proprietățile specifice CSS ale unui element din DOOM-ul din pagina web. Proprietățile specifice culorilor sunt: color și background-color, iar în funcție de aceste două proprietăți se face extracție de culori (sau culori de conținut) și extracție de culori de fundal. Extracția de culoare de conținut este identică cu extracția de culoare de fundal deci în continuare vom vorbi doar despre extracția de culoare (indiferent de tipul ei).

Extracția de culoare se face inspectând proprietățile specifice din CSS (color sau background-color) ale paginii încărcate în browserul headless. Aceste sunt colectate și pasate scriptului de procesare a culorilor (se sortează culorile și se elimină duplicatele).



[4]

Figura 3.1 *– Spațiile de culoare RGB (a.) și HSV (b.)*

3.3.2. Sortarea culorilor

Culorile primite din proprietățile CSS sunt din cubul RGB (acesta este un model care caracterizează o culoare prin intensitatea în nuanță de roșu, verde și albastru). Problema apare la definirea ordinii în cubul RGB deoarece acesta descrie mulțimea $\{0,1,2,\dots,255\}^3$, ce reprezintă o submulțime a lui $\mathbb{R}^3$, iar o ordonarea pe componente a tripletelor de tipul (r,g,b) nu are impact vizual asupra ochiului (culorile par a fi luate aleator). Pentru a rezolva această problemă se trece de la modelul de culoare RGB la spațiul de culoare HSV (hue sau nuanță – corespunde poziției unghiulare din jurul axei cilindrului, saturația și value sau valoarea – depărtarea de negru). HSV este un spațiu de culoare mai apropiat de modul în care omul percepe atributele de culoare decât RGB. Transformările de la RGB la HSV sunt:

$$hue = \begin{cases} 60° \cdot \dfrac{g-b}{r-b}, & dacă\ r \geq g \geq b \\ 60° \cdot \left(2 - \dfrac{r-b}{g-b}\right), & dacă\ g > r \geq b \\ 60° \cdot \left(2 + \dfrac{b-r}{g-r}\right), & dacă\ g \geq b > r \\ 60° \cdot \left(4 - \dfrac{r-g}{b-g}\right), & dacă\ b > g > r \\ 60° \cdot \left(4 + \dfrac{r-g}{b-g}\right), & dacă\ b > r \geq g \\ 60° \cdot \left(6 - \dfrac{b-g}{r-g}\right), & dacă\ r \geq b > g \end{cases} \tag{3.1.}$$

$$saturatia = \frac{\max\{r,g,b\} - \min\{r,g,b\}}{\max\{r,g,b\}}$$

$$value = \max\{r,g,b\}$$

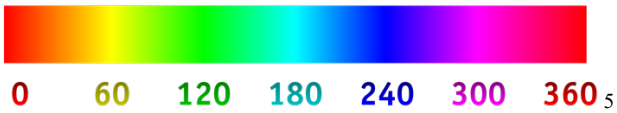
Sortarea culorilor se face în spațiul HSV ca o combinație liniară între hue (nuanță) și luminozitate, cu balansul de (80% hue si 20% luminozitate), adică:

$$valoareDeComparatie = 0{,}8 \cdot hue + 0{,}2 \cdot luminozitate$$

[4] *Imagini de Michael Horvath, disponibile sub licența Creative Commons (https://www.researchgate.net/figure/a-the-RGB-color-space-black-arrows-show-the-three-main-color-dimensions-whose-values_fig2_323952018)*

unde luminozitatea este:

$$luminozitate = 0{,}3 \cdot r + 0{,}59 \cdot g + 0{,}11 \cdot b \tag{3.2.}$$



[5]

Figura 3.2 *– Ordonarea culorilor după hue (nuanță)*

3.3.3. Colectarea gradienților

Extracția de gradienți este similară cu extracția de culoare, însă de această dată se inspectează proprietatea background-image din DOM. Dacă acesta corespunde unui gradient liniar sau radial atunci gradientul este colectat.

3.4. Extracția de stil de text

Extracția stilurilor de text presupune interogarea mai multor proprietăți din CSS specifice elementelor de tip text. Aceste proprietăți sunt fontul (font-family în CSS), stilul (font-style), mărimea fontului (font-size), culoarea textului (color), spațierea caracterelor (letter-spacing), spațierea rândurilor (line-height) și eventual decorațiile (text-decoration).

Procesarea acestor caracteristici de design presupune eliminarea duplicatelor și sortarea fonturilor după nume și mărime.

3.5. Servirea cererilor

Serviciul are la bază un Server WebSocket [23] care așteaptă conexiuni. În momentul în care se stabilește o conexiune serverul primește ca răspuns un string îl verifică să vadă dacă este un URL valid sau nu. Daca răspunsul nu este un URL valid atunci se trimite un mesaj de eroare altfel se instanțiază browserul headless cu URL-ul primit și se creează pagina de pe care sa va face extracția. În momentul de față se face extracție de culoare, gradienți și stil de text conform descrierilor de mai sus. Serverul este multi-client, adică poate răspunde mai multor clienți în același timp.

[5] *Sursa imaginii - https://en.wikipedia.org/wiki/Hue*

3.6. SERVICIUL DE PYTHON

Acest serviciu face extracție de simboluri de pe o pagină web (concept definit în Introducere). Serviciul are la bază un browser headless care este folosit pentru a crea capturi de ecran, mai apoi aceste imagini sunt analizate cu o serie de algoritmi de computer vision și machine learning cu scopul de a extrage simbolurile căutate. Am ales această variantă de extracție (din varianta de tip imagine) și nu din varianta cod sursă HTML deoarece pentru același design de simbol pot exista zeci de implementări, iar stilul extras trebuie să fie independent de aceste implementări. În plus această variantă de extracție este una generală care pate fi extinsă ușor și pentru alte interfețe grafice (altele decât web – implementarea curentă), tot ce trebuie făcut în plus pentru alte interfețe este tunarea hiperparametrilor. Ca observație, o variantă de tip machine learning pe codul HTML nu este fezabilă datorită inexistentei unui dataset (cod – stil/design). În continuare vor fi prezentate dependințele (bibliotecile și pachetele utilizate), metodele și algoritmii de extracție a simbolurilor.

3.7. Dependințele serviciului de Python

Dependințele principale ale serviciului de Python sunt:

- OpenCV – bibliotecă specializată în mare parte pe Computer Vision, este folosita pentru extracția de simboluri și procesarea lor; [27]
- PyTorch – framework de machine-learning folosit în cazul de față pentru antrenarea unui clasificator de simboluri; [28]
- Pyppeteer – browserul headless de la Google; [21][29]
- Public Image Ltd (PIL) – bibliotecă specializată pe procesarea de imagini
- Websockets – pentru crearea serverul WebSocket cu scopul a accepta conexiuni cu clienții
- NumPy – bibliotecă de Python specializată pentru algebră liniară, este folosită pentru procesarea imaginilor (care sunt matrice)
- SciPy – bibliotecă specializată pentru calcule științifice, este folosită pentru a măsura indicele de similaritate structurală dintre două imagini (folosit la eliminarea de redundanță)

3.8. Extracția simbolurilor circulare

Simbolurile circulare sunt reprezentate de radio butoane. Datorită modului unic de a implementa aceste elemente de UI (*input[type = radio]*), extracția acestor tipuri de simboluri

se va face astfel: inițial se face captură de ecran în jurul acestor tipuri de elemente (mai precis se face captură de ecran la subarborele de la părintele radio butonul pană la frunze), acest lucru se face cu browserul headless. Apoi se analizează aceste imagini aplicând o serie de transformări asupra lor. Aceste transformări sunt: bordarea cu alb a marginilor astfel încât elementul analizat sa nu aibă muchii lipite de marginile capturii de ecran, aplicarea unor serii de filtre de blur median și aplicarea transformatei Hough pentru cercuri [3] cu scopul de a determina radio butoanele.

3.8.1. Filtru median de blurare

Blurarea unei imaginii cu un filtru de medie presupune înlocuirea valorii din fiecare pixel al imaginii cu media vecinilor săi pe distanța dată de dimensiunea filtrului. Pe cazul de față se aplică o serie de astfel de transformări cu filtre de dimensiune 7, 5, 3 și 1 (ultimul lăsând imaginea pe loc). Blurarea are rolul de a elimina o parte din zgomotul de fundal, cercurile rămânând invariante la astfel de transformări. După aceste transformări se obțin 4 imagini pe care se aplica transformata Hough pentru cercuri.

3.8.2. Transformata Hough pentru cercuri

Aceasta algoritm este folosit pentru a detecta cercurile dintr-o imagine [3]. Cum cercurile sunt caracterizate de trei parametrii: doi pentru coordonatele (x, y) și unul pentru raza r, rezultă că trebuie făcută o căutare într-un spațiu tridimensional. Pentru a face acest lucru se utilizează o matrice de acumulare 3D care reține rezultatele parțiale. Algoritmului are ca input o imagine I de dimensiuni $W \times H$ și un număr natural R care reprezintă dimensiunea maximă a razei căutate. O schiță a algoritmului este descrisă mai jos:

$$
\begin{aligned}
&Pas_1: A[x, y, r] = 0\ \forall x = \overline{1, W}, y = \overline{1, H}, r = \overline{1, R} \\
&Pas_2: E = Canny(I) - vom\ descrie\ ulterior\ acest\ algoritm \\
&Pas_3: pentru\ fiecare\ x = \overline{1, W}\ din\ E \\
&\qquad pentru\ fiecare\ y = \overline{1, H}\ din\ E \\
&\qquad\qquad dacă\ E(x, y) == 0 \\
&\qquad\qquad\qquad pentru\ fiecare\ r = \overline{1, R} \\
&\qquad\qquad\qquad\qquad pentru\ fiecare\ \theta = \overline{0°, 360°} \\
&\qquad\qquad\qquad\qquad\qquad (p, q) = \left(x - r \cdot cos\left(t\frac{\pi}{180°}\right), y - r \cdot \sin\left(t\frac{\pi}{180°}\right)\right) \\
&\qquad\qquad\qquad\qquad\qquad A[p, q, r] += 1\ (pasul\ de\ votare)
\end{aligned}
$$

Acest algoritm este rulat pe fiecare din cele patru imagini blurate provenite de la 3.6.1., de unde se obțin patru matrice de acumulare. Din aceste matrice este selectat tripletul (p^*, q^*, r^*) care are valoare maximă. Acest triplet descrie cercul cu scorul cel mai mare din imagine, în speță

radio butonul din captura de ecran. Menționăm că a fost folosită transformata Hough pentru cercuri din OpenCV care are o serie de optimizări printre care analiza gradientului imaginii și nu muchiile detectate cu Canny [5].

3.9. Extracția simbolurilor regulate

Un simbol regulat este un simbol care își conservă stilul pe diverse interfețe grafice. Cu alte cuvinte arată asemănător în diversele sale modalități de implementare. Aceste simboluri sunt reprezentate de: butoane, dropdown-uri și texte de input. Deoarece aceste simboluri au un stil constant când vine vorba de formă și structură rezultă că pot fi extrase cu un algoritm de template matching (potrivire de formă) la o varietate de scale.

Algoritmul de template matching este aplicat pe captura de ecran a interfeței grafice din care a fost eliminat tot textul cu un detector de text EAST [2] și au fost reconstruite zonele eliminate (cele cu text). Eliminarea de text are rolul de a simplifica interfața grafică cu scopul de a crește acuratețea extracției cu template matching. Acest procedeu nu afectează în nici un fel rezultatele extracției deoarece nici unul din simbolurile de mai sus nu conțin text pe muchiile sale. După eliminarea de text se face reconstrucție [8] cu rol de netezire a interfeței grafice astfel încât la aplicarea detectorului de muchii Canny [5] să fie considerate ca fiind muchii zonele de la granița bucăților de text eliminate, deoarece acestea sunt muchii false care nu apăreau în captura de ecran a interfeței originale. Cu algoritmul de template matching [5] se obțin o serie de regiuni care seamănă din punct de vedere structural cu template-ul căutat.

În continuare va fi prezentat detectorul de text EAST, algoritmii de reconstrucție a imaginilor (inpainting), de detectare de muchii (Canny), de template matching și în final algoritmul de propunere a regiunilor candidat (region proposal) pe bază de template matching care a fost folosit.

3.9.1. Detectorul de text EAST

Detectorul de text EAST (Efficient and Accurate Scene Text Detector) este un model de deep leaning (învățare adâncă), bazat de detectarea de patternuri cu arhitectura descrisă în **Figura 3.3.** [2] Aceasta metodă de detectare a textului este cea mai precisă în momentul de față (este state-of-the-art) fiind capabilă să analizeze pana la 13fps (imagini-uri per secunda) cu dimensiunea de 1280×720 pixeli.

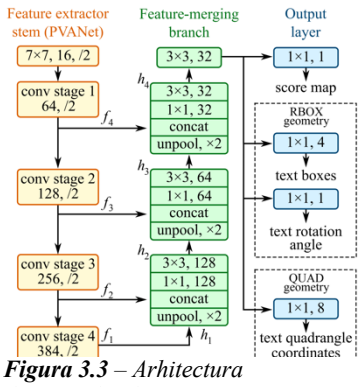


***Figura 3.3** – Arhitectura detectorului de text EAST [2]*

3.9.2. Reconstrucția imaginilor

Reconstrucția imaginilor sau inpainting-ul este procesul de reconstrucție a bucăților pierdute dintr-o imagine, la noi părțile eliminate. O metoda clasică de a realiza restaurare de imagine este cu ecuații diferențiale (cum ar fi ecuația Laplace) cu condiții la limită Dirichlet pentru continuitate (pentru a asigura potrivirea exactă a zonei umplute cu contextul exterior). Acest lucru funcționează foarte bine daca zona de reconstrucție se află într-un context omogen în imaginea originală (ceea ce se si întâmplă în cazul nostru, deoarece este de așteptat ca într-o interfață grafică zona de text eliminată să aibă un background uniform (de obicei alb)) [8]. În continuare, captura cu zone de text eliminate și reconstruite este dată algoritmului Canny pentru detecția de muchii [5].

3.9.3. Algoritmul Canny pentru detecția de muchii

Detectorul de muchii Canny [5] a fost dezvoltat în 1986 de către John F. Canny si este încă utilizat pe scară largă astăzi. Pentru o imagine color I de dimensiuni $W \times H$ și două praguri de confidența inf $și$ sup, algoritmul presupune următori pași:

$Pas_1: N = imageColor2Grayscale(I)$
$Pas_2: N' = bluareGaussiană(N)$
$Pas_3: S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix},\ G_x = S \therefore N', G_y = S^T \therefore N',\ G = \sqrt{{G_x}^2 + {G_y}^2}$
$Pas_4: pentru\ fiecare\ x = \overline{1, W}\ din\ G$
$\quad pentru\ fiecare\ y = \overline{1, H}\ din\ G$
$\quad\quad dacă\ G(x, y) \geq \max\{G(\widetilde{x+1}, y), G(\widetilde{x-1}, y), G(x, \widetilde{y+1}), G(x, \widetilde{y-1})\}$
$\quad\quad\quad G(x, y) = 255$
$\quad\quad altfel$
$\quad\quad\quad G(x, y) = 0$
$Pas_5: pentru\ fiecare\ x = \overline{1, W}\ din\ G$
$\quad pentru\ fiecare\ y = \overline{1, H}\ din\ G$
$\quad\quad dacă\ G(x, y) > \sup\ \ sau\ G(x, y) < \inf$
$\quad\quad\quad G(x, y) = 0$
$Pas_6: pentru\ fiecare\ x = \overline{1, W}\ din\ G$
$\quad pentru\ fiecare\ y = \overline{1, H}\ din\ G$
$\quad\quad dacă\ \max\{G[\widetilde{x-8}: \widetilde{x+8}, \widetilde{y-8}: \widetilde{y+8},]\} < \overline{G}$
$\quad\quad\quad G(x, y) = 0$

Unde $\sim$ reprezintă operația care nu permite ieșirea variabilelor x $și$ y din dimensiunile $\overline{1, W}$ respectiv $\overline{1, H}$.

Pasul 2 are scopul de a elimina zgomotul de frecvență înaltă. Ecuația unui filtru gaussian de dimensiune $(2k + 1)$ este: $H_{ij} = \frac{1}{2\pi\sigma^2} e^{-\frac{(i-(k+1))^2 + (j-(k+1))^2}{2\sigma^2}} cu\ i, j = \overline{1 \ldots (2k + 1)}$. La pasul 3,

G_x și G_y reprezintă două imagini în care fiecare pixel conține valoarea aproximativă a derivatei verticale respectiv orizontale, G reprezintă o imagine în care fiecare pixel are valoare de magnitudine a gradientului imaginii analizate I, iar $\therefore$ reprezintă operația de convoluție, despre care vom descrie în detaliu în secțiunea **3.9.1.** ($J(x,y) = S \cdot I[x-1{:}\,x+1, y-1, y+1]$). Pasul 4 reprezintă algoritmului de suprimarea a non-maximelor pentru a scăpa de răspunsul fals, întrucât după calcularea gradientului marginile sunt încă neclare. Algoritmul de suprimarea a non-maximelor reduce intensitatea tuturor pixelilor la 0 cu excepția maximelor locale, care indică locația celor mai mari schimbări de intensitate implicit a muchiilor. Pasul 6 elimină muchiile rezultate de la pasul anterior cu un gradient prea mic sau prea mare, cauzate de zgomot sau de variații de culoare. Iar pasul 6 elimină muchiile slabe care nu sunt conectate la muchii puternice.

După aplicarea acestui algoritm pe captura de ecran fără text și reconstruită se obține o nouă imagine care conține doar muchiile acesteia. Detectorul de muchii este aplicat și pe template-urile folosite apoi se determină potrivirile cu algoritmul de template matching.

3.9.4. Algoritmul de template matching

Acest algoritm își propune găsirea tuturor potrivirilor cu un anumit grad de asemănare a unui template (poză de dimensiuni reduse față de imaginea în care se face căutarea) într-o anumită imagine [5]. Acest lucru se face cu sliding-window (fereastră glisantă) anume: se glisează imaginea template de-a lungul imaginii în care se face căutarea și se calculează un grad de asemănare între template și patch-ul din imaginea de lucru (patch-ul are dimensiuni egale cu template-ul) [16]. Distanța folosită este clustering coefficients normed (coeficientul de clustering normat):

$$R(x,y) = \frac{\sum_{i=1}^{h}\sum_{j=1}^{w}\Big(\big(T(i,j)-\overline{T}\big)\big(I(x+i,y+j)-\overline{I}\big)\Big)}{\sqrt{\sum_{i=1}^{h}\sum_{j=1}^{w}\big(T(i,j)-\overline{T}\big)^2 \cdot \sum_{i=1}^{h}\sum_{j=1}^{w}\big(I(x+i,y+j)-\overline{I}\big)^2}} \qquad (3.3.)$$

unde T reprezintă templateul de dimensiuni (w,h), I imaginea de căutare de dimensiuni (W,H), $\overline{T} = \frac{1}{w \cdot h}\sum_{i'=1}^{h}\sum_{j'=1}^{w} T(i',j')$, $\overline{I} = \frac{1}{w \cdot h}\sum_{i'=1}^{h}\sum_{j'=1}^{w} I(x+i',y+j')$, $x = \overline{1 \ldots W-w}$ și $y = \overline{1 \ldots H-h}$. Se returnează gradul de asemănare în punctul (x,y), adică gradul de potrivire al template-ului peste patch-ul $I[x{:}\,x+w, y{:}\,y+h]$ din imaginea de căutare.

Algoritmul de template matching este rulat la diferite scale și cu diferite perechi de template-uri care corespund părților extremale stângi și drepte ale simbolurilor. Acest lucru reprezintă o optimizare uriașă deoarece nu este necesară o întreaga mulțime de template-uri cu proporții (aspect-ratio) diferite, ci de doar două perechi de template-uri, fiind necesar în plus un algoritm de grupare al extremității stângi cu extremitățile drepte corespunzătoare (algoritm exemplificat în 2.7.6).

În acest moment avem regiunile de interes din imaginea în care s-a făcut căutarea care conțin potențialele elemente de UI căutate (regiunile de interes corespund imaginii originale deoarece nici una din transformările de mai sus nu afectează dimensiunile imaginii de lucru). Însă există problema inevitabilă ca pentru același element din imaginea originală să existe mai multe propuneri care se suprapun. Pentru a rezolva acest lucru se utilizează un algoritm de suprimare a non maximelor (non maximum suppression) pentru detecția de obiecte [7].

3.9.5. Non Maximum Suppression pentru detecția de obiecte

Non-Maximum Suppression este folosit în detectarea de obiecte pentru ca un obiect să fie identificat o singură dată. Acest algoritm asigură identificarea optimă a tuturor candidaților posibili [7]. Modul în care se realizează eliminare non maximelor dintr-o listă de propuneri L:

Pas_1: $elimină\ toate\ propunerile\ cu\ scorul\ R(x,y)\ < 0{,}6$
Pas_2: $S = \emptyset$
Pas_3: $cât\ timp\ L\ \neq \emptyset$
$\quad m =\ propunerea\ cu\ scopul\ maxim\ din\ L$
$\quad S = S \cup \{m\}$
$\quad L = L\ /\ \{m\}$
$\quad pentru\ fiecare\ propunere\ p\ din\ L$
$\quad\quad dacă\ suprafața\ intersecției\ dintre\ p\ și\ m\ > \frac{suprafața\ lui\ p}{2}$
$\quad\quad\quad L = L\ /\ \{p\}$

3.9.6. Propunerea regiunilor candidat

Mai departe vom comasa toți algoritmii și toate metodele de mai sus într-unul singur pe care îl vom nota *algoritm de propunere a regiunilor candidat* sau *region proposal* pe baza de template matching. Acest region proposal primește pe post de input o captură de ecran a unei interfeței grafice și returnează o listă cu regiunile de simbolurile propuse. Pașii de funcționare sunt:

Pas_1: forward prin modelul de detecție de text EAST,
de unde se obțin dreptunghiurile care conțin text
Pas_2: se elimină aceste dreptunghiuri din imagine
Pas_3: se reconstruesc zonele eliminate
Pas_4: se determină muchiile imaginii de mai sus și muchiile

template − urilor selectate cu detectorul de muchii Canny

Pas_5: se rulează algoritmul de template matching la o varietate de scale, de praguri de confidență și de template − uri

Pas_6: regiunile de interes obținute din pasul de mai sus sunt analizate de algoritmul de Non Maximum Suppression pentru detecția de obiecte

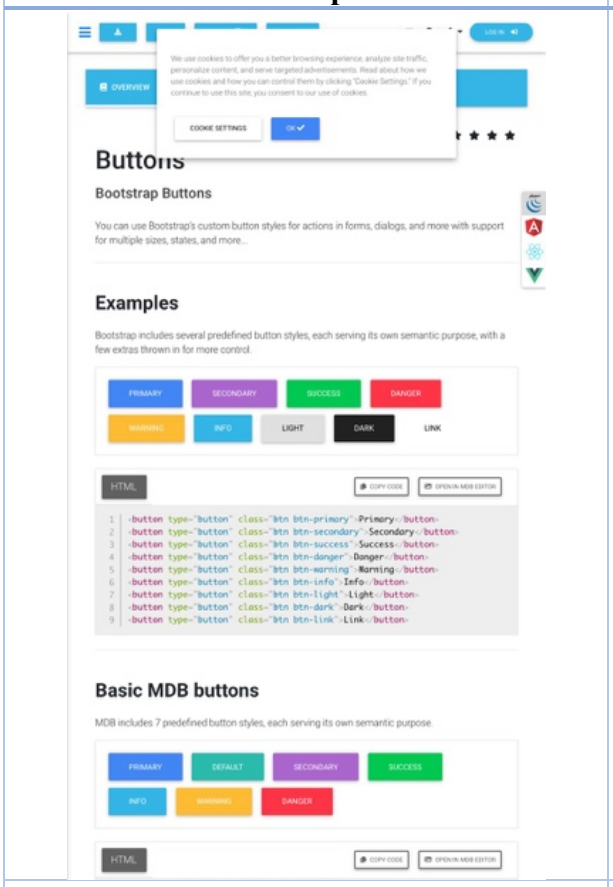
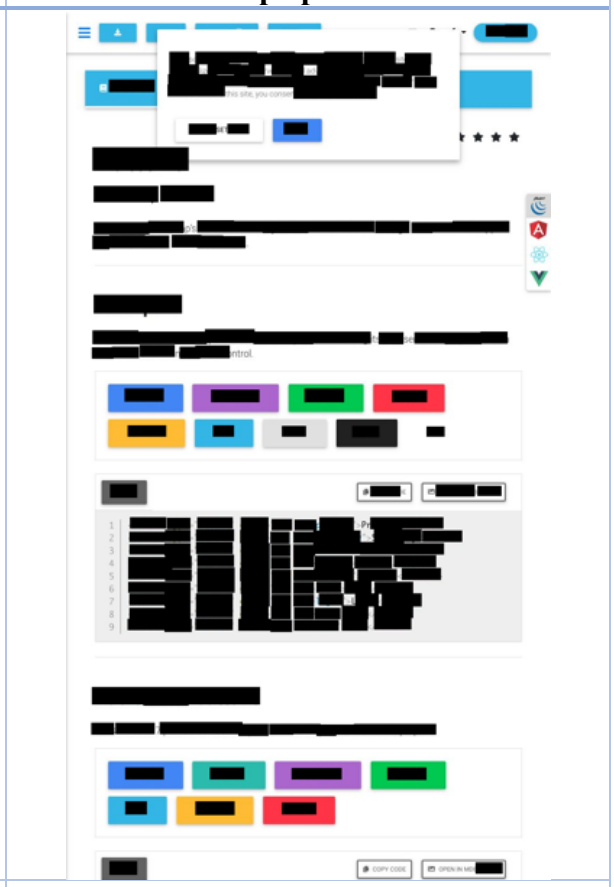
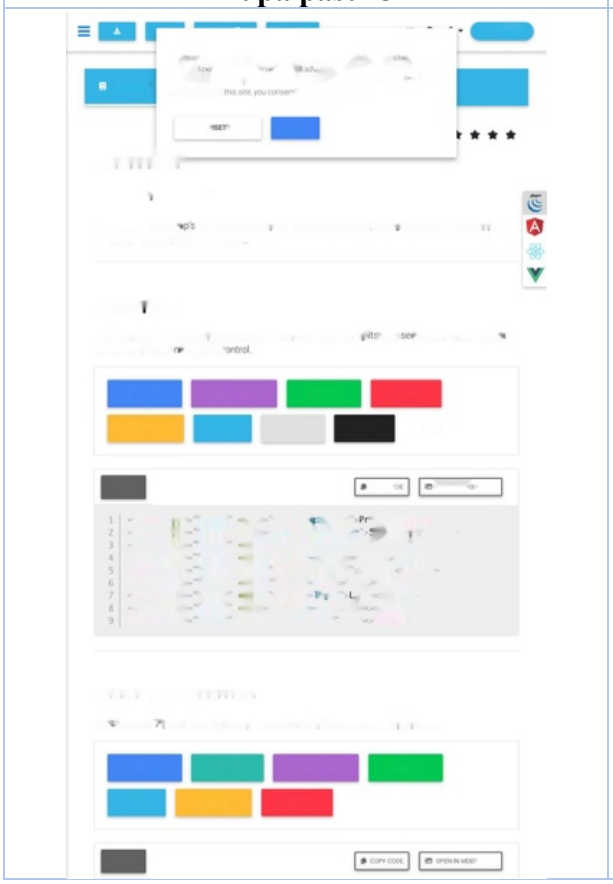
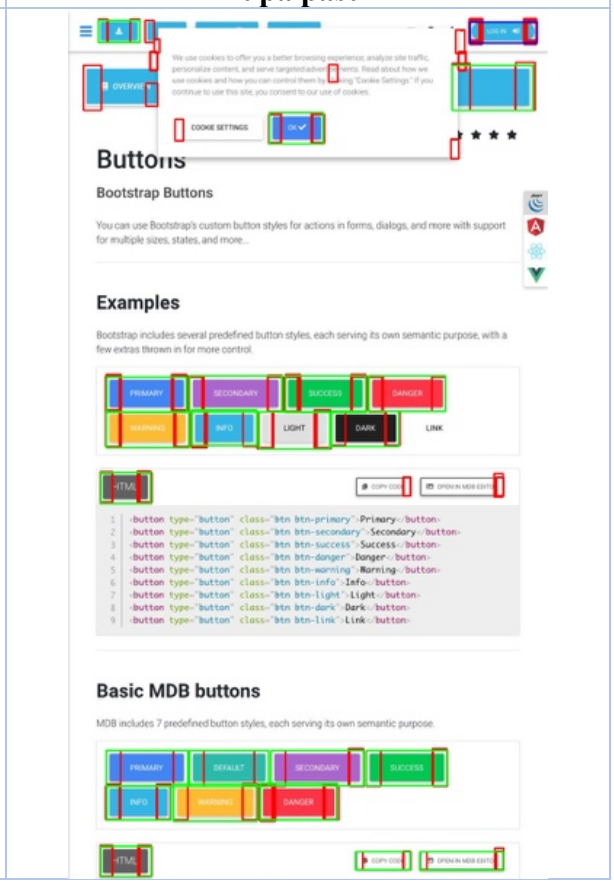
Pas_7: se returnează regiunile finale (cele păstrate la pasul anterior)

În funcție de dimensiunile interfețelor analizate se utilizează diferite template-uri și diferiți hiperparametrii. Acest lucru contribuie la îmbunătățirea performanței timp a extractorului. Hiperparametrii utilizați sunt:

Înălțimea paginii analizate	**Template stânga**	**Template dreapta**	**Pragurile Canny**	**Pragul de confidență Canny**	**Numărul de scale**
<3.000px			(25, 50)	0,85	250
			(10, 40)	0,55	150
<50.000px			(75, 125)	0,85	100
			(10, 40)	0,60	100

Tabelul 3.1 *– Hiperparametrii algoritmului de propunere a regiunilor candidat*

Pașii algoritmului de propunere a regiunilor candidat exemplificați pe un caz concret (https://mdbootstrap.com/docs/jquery/components/buttons/) sunt exemplificați în **Tabelul 3.2**:

Înainte de pasul 1	După pasul 2
	
După pasul 3	**După pasul 7**
	

Tabelul 3.2 *– Rezultatele algoritmului de propunere a regiunilor candidat*

3.10. Eliminarea redundanței

Apariția redundanței este inevitabilă deoarece pot exista mai multe elemente care arată la fel pe o interfața grafică. Iar o extracție a simbolurilor dorește preluarea unică a informației (adică dacă un simbol apare de două sau mai multe ori pe o interfața grafică va trebui preluat o singură dată). Pentru a elimina redundanța se folosește un indice de similaritate structurală între simboluri peste care a fost adăugat un indice de similaritate a culorii medii. Culoarea medie are rolul de a diferenția simbolurile care arata la fel (au aceeași structură) însă au compoziție diferită (culoare diferită) deoarece indicele de similaritate structurală analizează imaginea în tonuri de gri, iar informația culorii se pierde [11].

3.10.1. Indicele de similaritate structurală (SSIM)

SSIM este bazat principiul de percepție care consideră că degradarea unei imaginii este o schimbare perceptuală a informațiilor structurale [11]. Informația structurală are la bază ideea că pixelii au interdependențe puternice între ei, mai ales când sunt aproape unul de celălalt din punct de vedere spațial. Aceste dependențe conțin informații importante despre contextul obiectelor în scenă vizuală. Formula de calcul pentru două imagini grayscale I_1, I_2 de dimensiuni n, m este:

$$SSIM(I_1, I_2) = \frac{(2\overline{I_1}\overline{I_2} + (k_1L)^2)(Cov(I_1, I_2) + (k_2L)^2)}{\left(\overline{I_1}^2 + \overline{I_2}^2 + (k_1L)^2\right)(Var(I_1) + Var(I_2) + (k_2L)^2)} \tag{3.4.}$$

unde $\overline{I_r} = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} I_r(i,j)$, $Var(I_r) = \frac{1}{nm-1}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(I_r(i,j) - \overline{I_r}\right)^2$,

$$Cov(I_1, I_2) = \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m}(I_1(i,j) - \overline{I_1})(I_2(i,j) - \overline{I_2}), k_1 = 0{,}01, k_2 = 0{,}03, L = 2^N - 1,$$

k_1 *și* k_2 sunt constante de stabilizare, N reprezintă numărul de biți necesari stocării unui pixel iar L o variabila care depinde de modalitatea de stocare a pixelilor și contribuie la stabilizarea formulei.

În cazul de față două simboluri sunt considerate a fi cu aceeași structură din punct de vedere al similarității structurale date de SSIM dacă această metrică este mai mare sau egală decât 0,6 (similaritate mai mare sau egală decât 60%).

3.10.2. Indicele de similaritate a culorii medii

Sunt comparate culorile medii a două simboluri date sub formă de imagine. Culorile sunt comparate în modelul de culoare RGB și nu în CIELAB [10] din cauza complexității calculelor

de trecere de la RGB la CIELAB dar și a complexității metricilor din spațiul CIELAB (se vor face aceste transformări în capitolul următor pentru a calcula distanța dintre stilul a două interfețe grafice). Menționăm că în cazul de față nu este foarte importantă măsura distanței dintre cele două imagini comparate ci o valoare binară care să specifice dacă au sau nu aceeași culoare de medie. Pentru două imagini color I_1, I_2 de dimensiuni n, m se calculează următoarea metrică:

$$MCSI(I_1, I_2) = \sqrt{\frac{\sum_{c\in\{R,G,B\}}\left(\sum_{i=1}^{n}\sum_{j=1}^{m} I_1(i,j,c) - \sum_{i=1}^{n}\sum_{j=1}^{m} I_2(i,j,c)\right)^2}{255^3}} \qquad (3.5.)$$

Două imagini sunt considerate având culoare medie similară dacă $MCSI$ este mai decât 0,05. Pentru ca două imagini să nu fie considerate simultan redundante trebuie ca cei doi indici să îndeplinească simultan condițiile $MCSI \leq 0{,}05$ *și* $SSIM \leq 0{,}6$.

3.11. Clasificarea rezultatelor și eliminarea falsului negativ

Informația despre tipul simbolurilor extrase nu este cunoscută deoarece simbolurile au fost extrase din varianta de tip imagine a interfeței grafice și nu din varianta cod sursă. Pentru a recăpăta această informație se va folosi o rețea convoluțională [15] care va clasifica imaginile cu simboluri cu scopul de a prezice tipul lor. Aceasta predicție poate fi folosită atât pentru clasificare cât și pentru eliminarea eventualului falsului negativ. Acest lucru se face prin adăugarea clasei zgomot pe lângă clasele deja existente, iar o predicție cu scor mare pentru clasa zgomot va indica ignorarea simbolului aferent.

3.11.1. Noțiuni teoretice

- Strat dens / fully connected (FC)

Este unul dintre cele mai simpliste straturi ale unei rețele neurale. Este format dintr-o serie de perceptroni în care fiecare dintre ei este conectat la toți perceptronii de la stratul anterior și la toți perceptronii de la stratul următor.

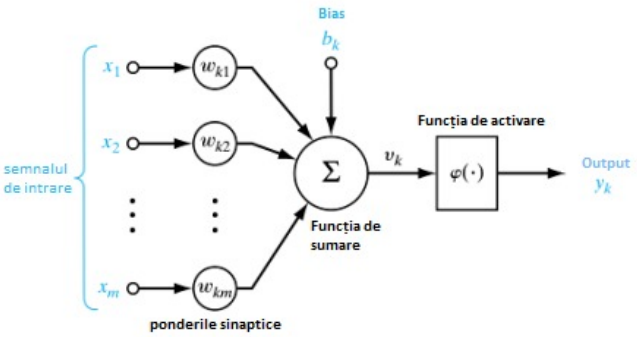


Figura 3.4 *– Perceptronul*

$$y_k = \varphi\left(\overbrace{\sum_{i=1}^{m} x_i \cdot w_{ki} + b_k}^{v_k}\right) \tag{3.6.}$$

Aceste straturi sunt de obicei ultimele starturi ascunse ale unei rețele convoluționale, acestea ocupând-se de clasificarea vectorilor de caracteristici primite de la straturile convoluționale.

- Strat convoluțional

Convoluția sau corelația matematică este o operație care ține seama de contextul în care se află pixelii într-o imagine . Un neuron convoluțional procesează date doar din câmpul lui perceptiv (dat de dimensiunea filtrului/ kernelului), spre deosebire de un neuron al stratului dens care primește toata informația stratului anterior [15]. Acest lucru are ca rezultat neutralizarea gradienților care se anulează (vanishing gradients) sau care devin prea mari (exploding gradients). Formula operației de convoluție este:

$$J(i,j) = \sum_{m=-k}^{k} \sum_{n=-k}^{k} I(i+m, j+n)F(m,n)\,, i = \overline{k+1, N-k}, j = \overline{k+1, M-k} \tag{3.7.}$$

unde I este imaginea inițială de dimensiuni $N \times M$, F este filtrul convoluțional de dimensiuni $2k+1 \times 2k+1$, J este imaginea rezultată de dimensiuni $\frac{N-(2k+1)+2P}{S} + 1 \times \frac{M-(2k+1)+2P}{S} + 1$, P este dimensiunea paddingului (cu cât se bordează imaginea I cu pixeli egali cu 0) și S este stride-ul (pasul de mutare al filtrelor). Ca observație adâncimea imaginii J va fi egală cu a imaginii I.

- Strat de pooling

Aceste straturi nu au parametrii de învățat ele doar reduc dimensiunea lățimii și înălțimii imaginilor care provin de obicei de la convoluție. Aceste straturi au un impact perceptiv asemănător cu cortexul uman preluând informația cea mai importantă (start de max-pooling) sau estompată (strat de average-pooling) din anumite regiuni ale unei imagini.

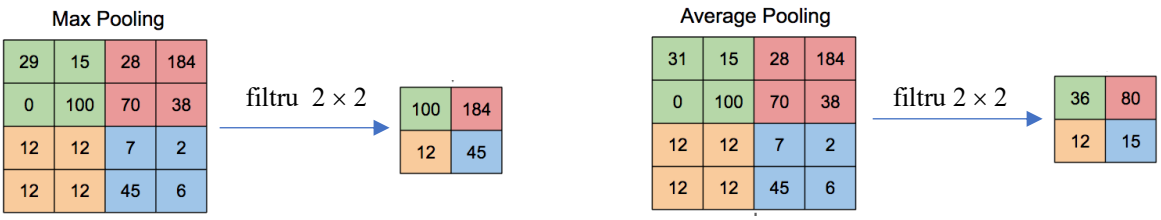


Figura 3.5 *– Exemplificarea operației de pooling*

3.11.2. Construcția datasetului

Datasetul a fost construit pe baza extractorului de stil fără curenta funcționalitate (de clasificare și eliminare de fals negativ). Au fost analizate o serie de câteva sute de pagini web de pe care s-a făcut extracție și salvare de simboluri. Apoi au fost eliminate duplicatele pe baza de distanță euclidiană (deoarece un simbol poate exista pe mai multe interfețe, chiar și pe aceeași interfața se poate repeta). Apoi au fost grupate elementele pe clase: butoane, dropdown-uri, texte de input, radio butoane și check box (ultimele două clase au fost adăugate doar pentru funcționalitatea de eliminare a fals negativului, nu și pentru clasificare deoarece pentru aceste clase se știe deja eticheta). Peste aceste date au fost adăugate imagini de pe Google Images (cropate și curățate de zgomot). Apoi datele au fost redimensionate la mărimea de $(224 \times 224)\ pixeli$ cu păstrarea proporției (aspect-ratio-ului). Acest lucru s-a realizat prin mărirea imaginii pană când una din dimensiuni atinge dimensiunea de 224 pixeli, apoi se umple golul rămas cu alb. În următorul tabel este sumarizată o distribuție a datelor:

Clasa	Numărul exemplelor de antrenare	Numărul exemplelor de validare	Numărul exemplelor de testare	Total
buton	800	95	48	943
text field	260	31	17	308
drop down	136	17	9	162
radio button	57	7	5	69
check box	103	13	7	123
zgomot	1223	145	73	1441

Tabelul 3.3 *– Distribuția datelor în cadrul datasetului*

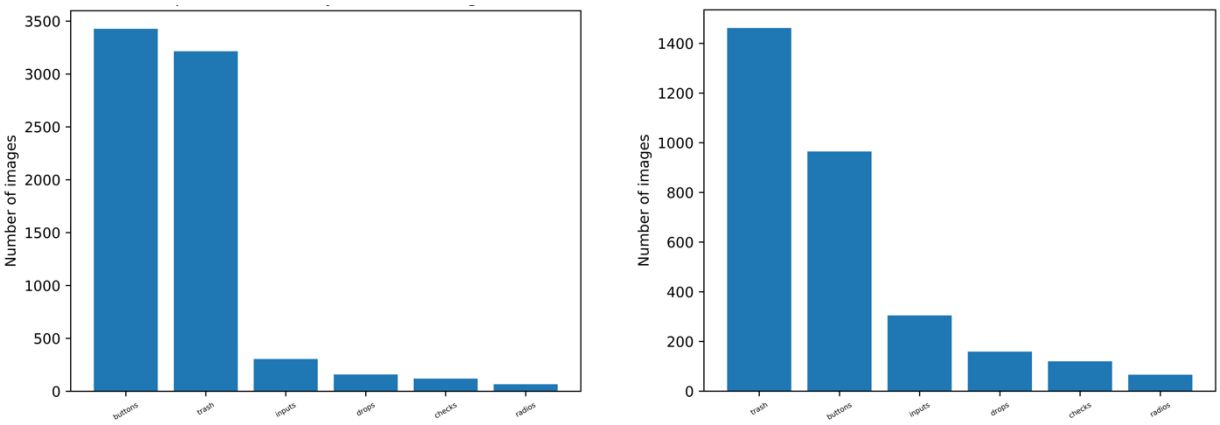


Figura 3.6 *– Distribuția datelor înainte (stânga) și după (dreapta) eliminarea redundanței*

Se poate observa cu ușurință că datele sunt puternic debalansate și insuficiente unei rețele convoluționale. Pentru a crea date artificiale se folosește un procedeu de augmentare a datelor

anume un ColorJitter care face schimbare de iluminare, saturație și nuanță rezultând date noi [28].

3.11.3. Arhitectura

Se folosește o rețea convoluțională deoarece simbolurile analizate sunt imagini [15]. Arhitectura folosită este descrisă în **Figura 3.7.**

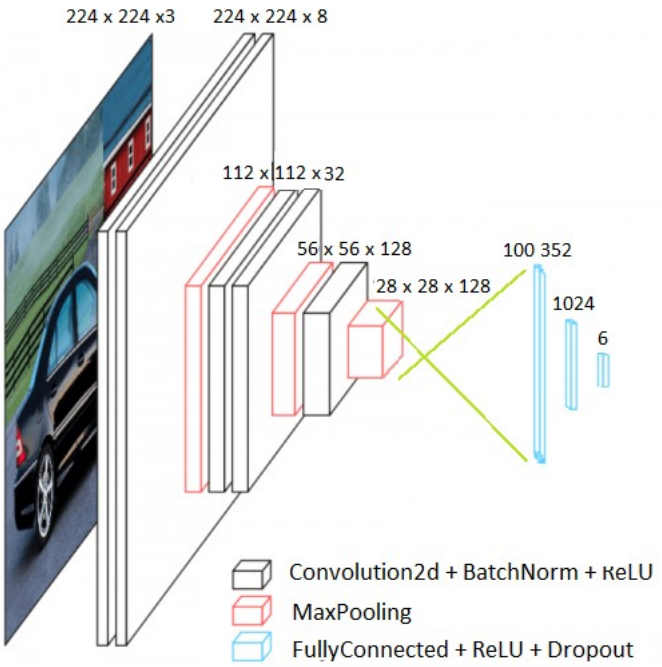


***Figura 3.7** – Arhitectura rețelei convoluționale care clasifică simbolurile*

O descriere detaliată a arhitecturii este:

a. Stratul pentru datele de intrare, care reprezintă o imagine color de dimensiune 224×224 cu 3 canale;
b. Un strat convoluțional cu 8 filtre de dimensiune (5×5), padding de (2×2) și pas 1 cu funcția de activare ReLU ($LeLU(x) = \max\{0, x\}$);
c. Un strat identic cu b. însă cu filtre de dimensiune (3×3) și padding de (1×1)
d. Un strat de max-pooling în care subzona în care se aplica max-poolingul are dimensiunea de (2×2) cu stride de 2, după acest pas se obțin imagini cu dimensiunea de $(112 \times 112 \times 8)$.
e. Un strat identic cu stratul b. însă cu 32 de filtre;
f. Un strat identic cu stratul c. însă cu 32 de filtre;
g. Un strat de max-pooling în care subzona în care se aplica max-poolingul are dimensiunea de (2×2) cu stride de 2, după acest pas se obțin imagini cu dimensiunea de $(56 \times 56 \times 32)$.

h. Un strat identic cu stratul b. însă cu 128 de filtre;

i. Un strat de max-pooling în care subzona în care se aplica max-poolingul are dimensiunea de (2×2) cu stride de 2, după acest pas se obțin imagini cu dimensiunea de $(28 \times 28 \times 128)$.

j. Un strat dens de 100.352 perceptori care reprezintă imaginea de $28 \times 28 \times 128$ vectorizată, cu funcția de activare ReLU;

k. Un strat dens de 1024 de perceptori, cu aceeași funcție de activare, care este conectat în totalitate cu startul f. și stratul h.;

l. Stratul final de 6 perceptori la care se adaugă funcția softmax.

3.11.4. Rezultate

În **Figura 3.8. (a)** este prezentată evoluția funcției de loss, in **Figura 3.8. (b)** este prezentată evoluția acurateței, iar în **Figura 3.9.** matricele de confuzie, simplă și normalizată (suma pe liniile acestei matrice de confuzie este egală cu 1) pentru modelul antrenat pe 20 de epoci, cu crossentropia pe post de loss $CrossEntropyLoss(x, l) = -\frac{1}{N}\sum_{i=1}^{N}[y_n log\widehat{y_n} + (1 - y_n)\log(1 - \widehat{y_n})]$ (unde N numărul de exemple din batch, la noi 32, $\widehat{y_n} = f(w \cdot x_n)$).

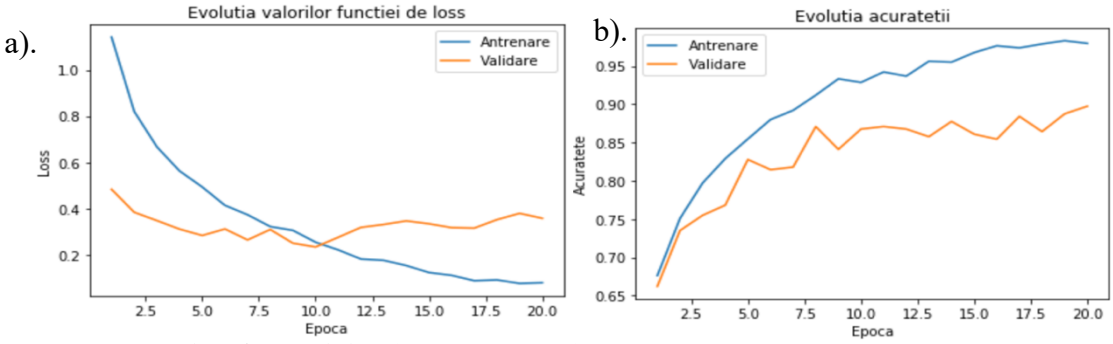


Figura 3.9 *– Evoluția funcției de loss* ***(a)*** *și a acurateței* ***(b)***

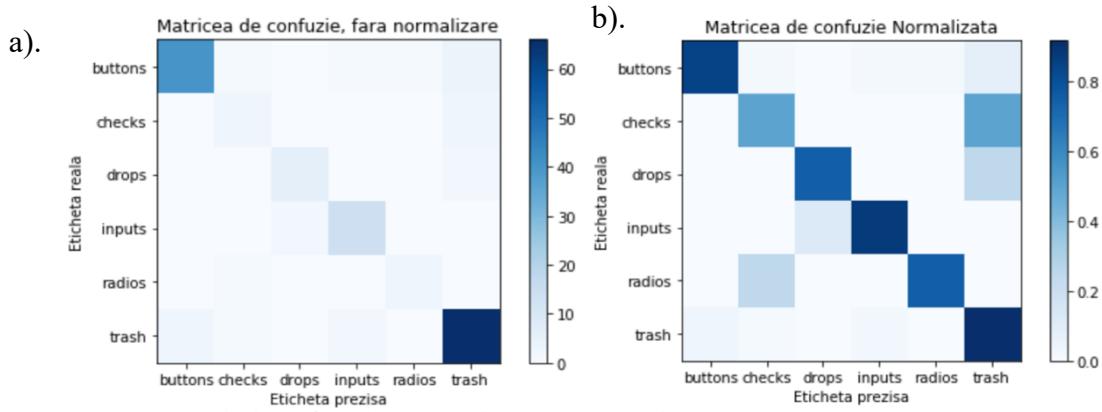


Figura 3.8 *– Matricele de confuzie* ***(a)*** *nenormalizată și* ***(b)*** *normalizată*

Acuratețea pe mulțimea de test este de 86%. A fost folosit algoritmul de coborâre pe gradient (SGD - stochastic gradient descent) cu learning rate de 0,001 si momentum de 0,9. Iar acuratețea recunoașterii clasei *zgomot* este de 92%.

3.12. PERFORMANȚA ȘI ACURATEȚEA EXTRACȚIEI

Acuratețea extracției de culori, gradienți și stil de text este de 100% deoarece sunt inspectate proprietățile aferente elementelor din codul interfeței grafice. Iar acuratețea extracției de simboluri a fost calculată pe baza a douăzeci de pagini web și este de 80% cu un recall de 92% (falsului negativ rămas după clasificare și eliminare de zgomot).

3.13. Performanța timp

	Request + plugin	Serviciul de NodeJS	Serviciul de Python							
			Lansare browser headless	Captura de ecran	Detecția de text cu EAST	Inpating	Template matching	Eliminarea redundanței	Clasificarea + eliminarea falsului pozitiv	Total serviciul de Python
Media	**3.18**	**6.22**	3.203	1.140	10.64	21.792	26.487	0.281	1.441	**57.19**
Deviația standard	**3.10**	**3.11**	1.753	0.486	16.791	28.363	18.925	0.899	0.613	**47.15**

Tabelul 3.4 *– Performanța timp a extractorului de stil, în secunde.*

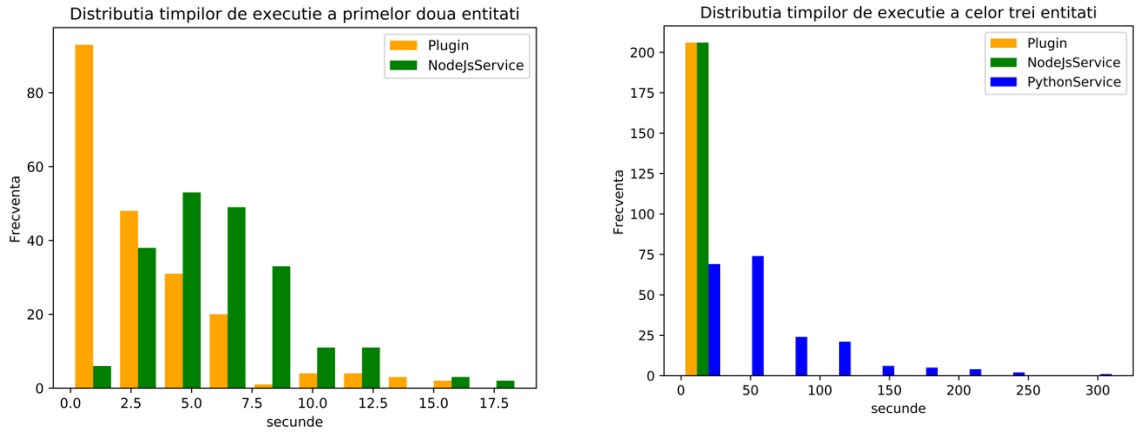


Figura 3.10 *– Distribuțiile timpilor de execuție între entități*

Mai jos vom prezenta câteva grafice care vor exemplifica relația dintre timpul de execuție pentru fiecare funcționalitate din serviciul de Python cu dimensiunea paginii analizate.

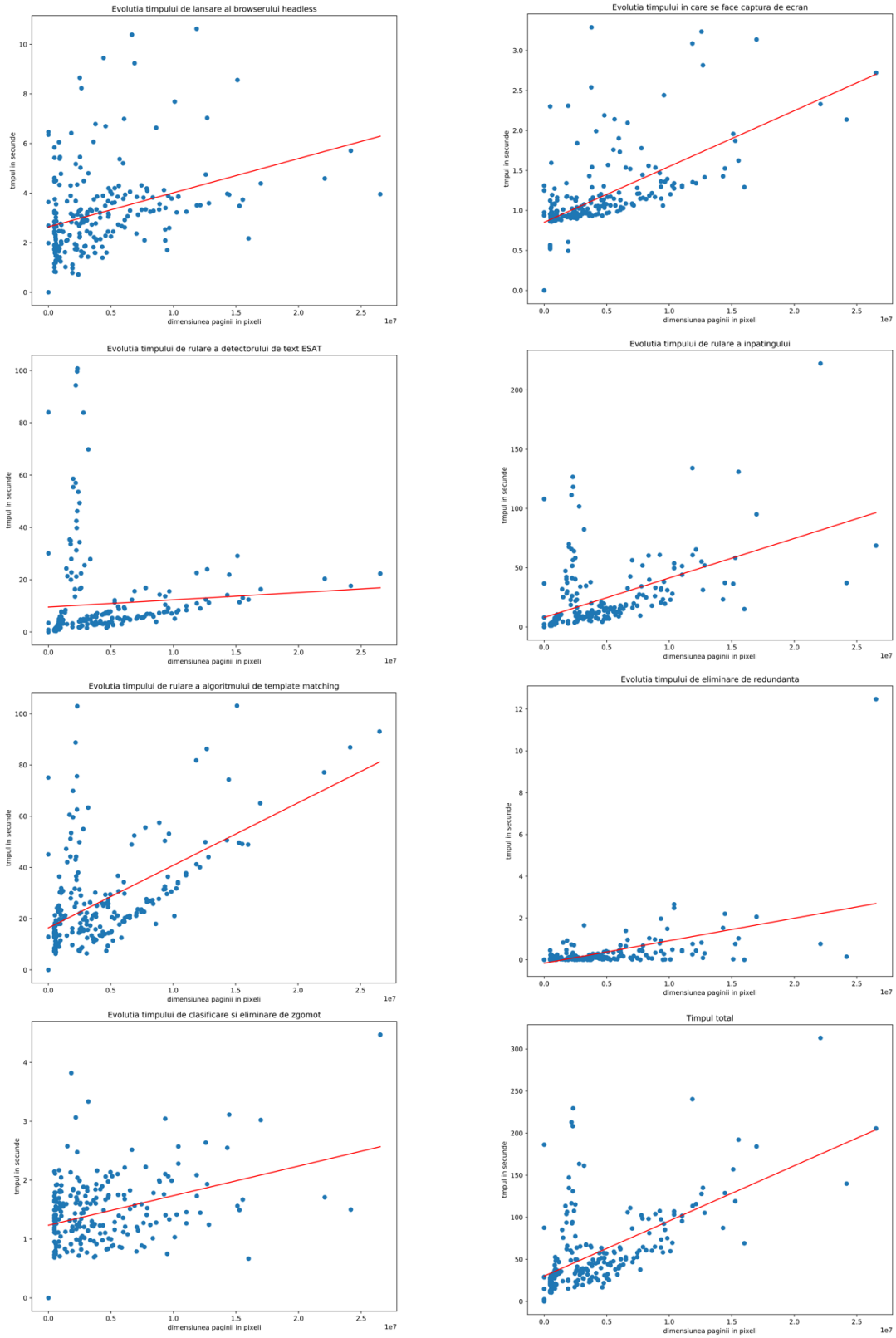


Figura 3.11 *– Corelația dintre timpul de execuție și dimensiunea paginilor analizate*

Se poate observa că timpul de lansare al browserului headless este constant (independent de dimensiunile paginii) iar restul timpilor sunt liniari cu dimensiunea în pixeli a paginii. Cu roșu este prezentată dreapta de regresie liniară a timpului relativ la dimensiunea în pixeli.

Capitolul 4

DISTANȚA DINTRE STILURI

În acest capitol se pun bazele conceptului de distanță între stilul a două interfețe grafice. Se explică modul de construcție al acestei distanțe ca o compunere a distanțelor dintre caracteristicile de design de același tip. Această distanță măsoară disimilaritatea dintre stilul a două interfețe grafice.

4.1. DISTANȚA DINTRE CARACTERISTICILE DE TIP CULOARE

Caracteristicile de tip culoare primite de extractorul de stil sunt culorile de conținut și culorile de fundal. În plus față de aceste două informații se va extrage informație și din captura de ecran a întregii pagini. Culorile extrase din captura de ecran funcționează pe ideea de mediere a informației, nefiind analizată doar informația declarată în codul sursă al interfeței grafice (culori de conținut și culori de fundal) ci și informația pe care utilizatorul o percepe. Această idee reprezintă o generalizare din [31]. Astfel se vor analiza și culorile din imaginile și frameurile prezente în interfața grafică, culori care contribuie la percepția vizuală asupra întregii interfețe grafice.

Pe baza acestor trei informatii: culori de conținut, culori de fundal și culori globale (cu rol de estompare, extrase din captura de ecran) se vor construi trei histograme de culoare pentru fiecare categorie de culori. Distanța dintre caracteristicile de culoare va fi o combinație baricentrică a distanțelor dintre cele trei histograme de culoare. În cele ce vor urma, vom fundamenta noțiunile de histogramă de culoare, distanță dintre două histograme de culoare și modul în care este compusă distanța dintre caracteristicile de culoare.

4.1.1. Histograma de culoare

O histogramă de culoare reprezintă un grafic care descrie distribuția culorilor într-o imagine. Aceasta contabilizează numărul de pixeli dintr-o imagine care au culori într-o lista predefinită din spațiul/ modelul de culoare [6]. Construcția histogramelor de culori de conținut și de fundal se face din câmpurile specifice culorilor din JSON-ul primit de la extractorul de stil. Aceste câmpuri conțin informații despre culoare, și suprafața elementelor din interfața grafică. Pentru construcția acestor două histograme de culoare (de conținut și de fundal) nu se ține cont de contextul în care se află elementele în scena grafică a interfeței, ci doar de dimensiunea lor (greutatea binurilor este dată de suma ariilor elementelor pentru culoarea respectivă). Contextul va fi preluat în histograma de culoare construită pe baza întregii pagini. Neținând cont de contextul elementelor din interfață, putem sorta descrescător histogramele după greutatea binurilor astfel încât primele binuri ale histogramei să fie și cele cu greutatea cea mai mare, corespunzătoare culorilor centrale, și cele de la sfârșitul histogramei să fie cele care corespund culorilor secundare. Histograma obținută este una neorientată. Sortarea are rolul de aranja culorile astfel încât distanța dintre histogramele de culoare să compare culorile principale între ele și culorile secundarea între ele.

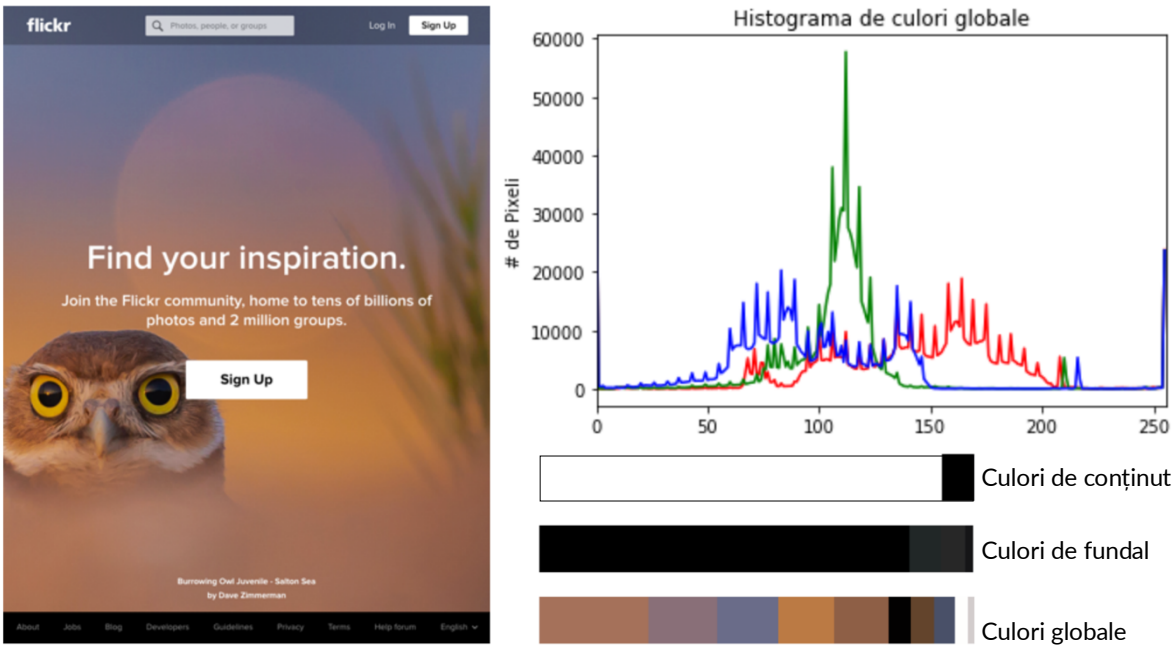


Figura 4.1 *– Histogramele de culoare extrase de pe pagina Flickr*

În **Figura 4.1** în partea stângă este pagina analizată (https://www.flickr.com/) în partea dreaptă sus este histograma de culoare a interfeței grafice (doar cu culorile roșu, verde și albastru), iar în partea dreaptă jos se afla histograma sortată de culori de conținut, fundal și globale.

Construcția histogramei de culori de pe întreaga interfață grafică, adică construită pe baza capturii de ecran a interfeței, este singura care tine cont de modul în care sunt repartizate culorile pe elementele de design, acesta are rol de estompare a informației declarate (culori de conținut si fundal) [31]. Acest lucrul are la bază ideea de mediere, și are impact asupra modului în care ochiul uman percepe o imagine, în particular o interfață grafică. Acesta histogramă nu va fi sortată după greutatea binurilor deoarece se ține cont de repartiția culorilor pixelilor din interfață, fiind o histogramă orientată.

4.1.2. Distanța dintre histogramele de culoare

Histogramele construite se normalizează, adică se împarte greutatea fiecărui bin la suma greutăților tuturor binurilor astfel încât suma binurilor să facă 1. Acest lucru oferă independență distanței relativ la dimensiunea paginii.

Pentru calcularea distanțelor dintre histogramele de culoare există mai multe metode consacrate: Corelația, Chi-Square, Intersecția, divergenta Kullback-Leibler, printre care și distanța Bhattacharyya [6]. În continuare vom folosi distanța Bhattacharyya deoarece acesta ia valori între 0 (care indica potrivire maximă) și 1 (care indică nepotrivire), este simetrică ($d_{Bhattacharyya}(H_1, H_2) = d_{Bhattacharyya}(H_2, H_1)$) și este cea mai discriminantă pentru cazul

nostru de analiză (are cele mai depărtate valori pentru interfețe cu elemente colorate diferit). Formula de calcul a distanței Bhattacharyya este:

$$d_B(H_1, H_2) = \sqrt{1 - \frac{1}{N \cdot \sqrt{\sum_{i=1}^{n} H_1(b_i) \cdot \sum_{i=1}^{n} H_2(b_i)}} \cdot \sum_{i=1}^{n} \sqrt{H_1(b_i) \cdot H_2(b_i)}} \qquad (4.1.)$$

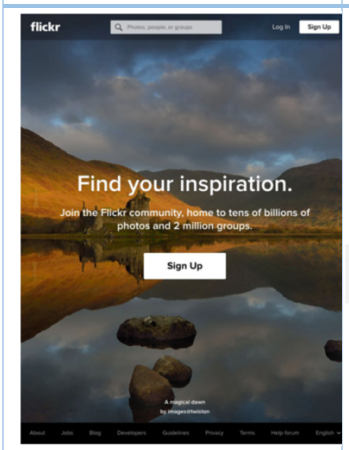
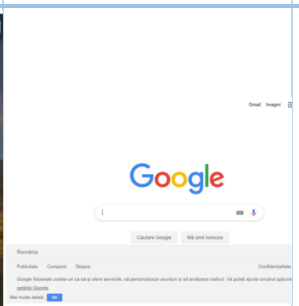
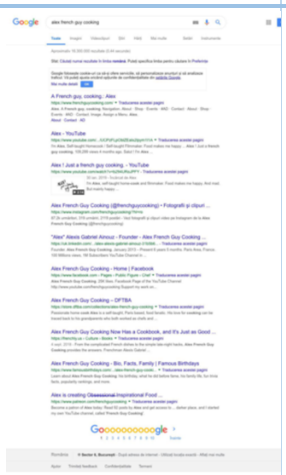
unde N este o constantă de normalizare, la noi 1 deoarece histogramele au fost normalizate, n numărul de binuri din histograma, iar b_i reprezintă binul numărul i din cele n [6].

Distanța finală dintre caracteristicile de tip culoare a două interfețe, este o combinație baricentrică a distanțelor Bhattacharyya dintre celor trei histograme de culoare:

$$d_{colorFeatures}(I_1, I_2) = \chi \cdot d_{\mathrm{B}}\left(H_{I_{1_C}}, H_{I_{2_C}}\right) + \psi \cdot d_{\mathrm{B}}\left(H_{I_{1_F}}, H_{I_{2_F}}\right) + \omega \cdot d_{\mathrm{B}}\left(H_{I_{1_G}}, H_{I_{2_G}}\right)$$

unde $H_{I_{i_C}}$ este histograma de culori, $H_{I_{i_F}}$ este histograma de culori de fundal, $H_{I_{i_G}}$ este histograma de culori globale a interfeței I_i si χ, ψ, ω sunt coeficienții combinației baricentrice.

Experimental au fost luați parametrii $\chi = 0{,}1$, $\psi = 0{,}3$, $\omega = 0{,}6$. Intuiția este că impresia generală este mai importantă din punct de vedere perceptiv pentru ochiul uman decât stilul declarat în codul interfeței grafice. Iar impactul vizual al unei culori de fundal este mai puternic decât al unei culori de conținut. În **Tabelul 4.1** pe coloanele 1 si 2 se analizează paginile: https://www.flickr.com/ și https://www.google.com, iar pe coloanele 3 si 4 avem calculată distanța de pe două căutări Google, pe prima coloană a fosta căutat „Alex french guy cooking" iar pe coloană a două a fost căutat „google fonts".

Interfața 1	Interfața 2	Interfața 1	Interfața 2
			
0.731318		**0.034996**	

Tabelul 4.1 *– Distanțele dintre caracteristicile de culoare de pe două perechi de pagini web.*

4.2. DISTANȚA DINTRE CARACTERISTICILE DE TEXT DE TEXT

O caracteristică de tip text este compusă din mai multe caracteristici secundare, precum fontul, dimensiunea textului și culoarea acestuia (ignorăm pentru moment stilul acestuia, adică dacă este scris normal, italic sau subliniat). Aceste informații sunt existente în răspunsul primit de la serviciu de NodeJS. Ca observație, datorită practicii de a pune înălțimea și lățimea ca „auto" la elementele de tip text, aceste atribute nu vor putea fi extrase în pixeli din proprietățile aferente elementelor cu un browser headless. Deci ponderea unei caracteristici de text specifice unui element din interfața grafică nu va fi legată de suprafața acestui element, ci va fi legată de cardinalitate. În cele ce vor urma se vor formaliza distanțele dintre caracteristicile secundare ale unei caracteristici de text și se va compune distanța dintre două feature-uri de text.

4.2.1. Distanța dintre fonturi

Un font poate fi definit ca un stil unitar specific unei mulțimi de caractere, care au aceeași grosime și dimensiune. Un fost are în general extensia .ttf și poate fi randat sub forma unei imagini. Distanța dintre două fonturi se va defini ca medie după toate caracterele printabile dintre o distanță care ia ca parametrii două caractere de același tip sub formă de imagine. În continuare rămâne de văzut ce distanță dintre două imagini se potrivește pentru acest caz particular.

Cele mai importante distanțe dintre două imagini sunt: distanța Euclidiană, distanța Mahalanobis, distanța Chord, coeficientul de corelație a lui Pearson și coeficientul de rang a lui Spearman [9]. Dintre acestea distanța Chord este cea mai discriminantă (pentru fonturi diferite are valorile cele mai mari comparativ cu celelalte distanțe enumerate și pentru fonturi asemănătoare are valori apropiate de 0). Formula distanței Chord este:

$$d_{Chord}(A_1, A_2) = \left\| \frac{\widehat{A_1}}{\|\widehat{A_1}\|_2} - \frac{\widehat{A_2}}{\|\widehat{A_2}\|_2} \right\|_2 = 2 \cdot \sin\left(\frac{\alpha}{2}\right) \qquad (4.2.)$$

unde

- A_1 si A_2 sunt imagini cu n linii si m coloane, cu $A_1, A_2 \in \Gamma^{n \times m}$, $\Gamma = \{0, 1, \dots, 255\}^3 \subset \mathbb{R}^3$
- ˆ reprezintă operația de vectorizare, adică $\widehat{A_1}, \widehat{A_2} \in \{0, 1, \dots, 255\}^{n \times m \times 3}$ sunt vectori de dimensiune $n \times m \times 3$
- $\| \|_2$ reprezintă norma euclidiană ($\|x\|_2 = \sqrt{\sum_{i=1}^{p} x_i^2}, x = (x_1, x_2, \dots, x_p) \in \mathbb{R}^p, p \in \mathbb{N}^*$)
- α reprezintă unghiul dintre vectorii $\widehat{A_1}, \widehat{A_2}$.

Distanța dintre două fonturi se formalizează în funcție de distanța Chord astfel:

$$d_{font}(F_1, F_2) = \frac{1}{|\mathcal{C}| \cdot n \cdot m} \cdot \sum_{c \in \mathcal{C}} d_{Chord}\left(A_{c_{F_1}}, A_{c_{F_s}}\right) = \frac{1}{|\mathcal{C}| \cdot n \cdot m} \cdot \sum_{c \in \mathcal{C}} \left\| \frac{\widehat{A_{c_{F_1}}}}{\left\|\widehat{A_{c_{F_1}}}\right\|_2} - \frac{\widehat{A_{c_{F_2}}}}{\left\|\widehat{A_{c_{F_2}}}\right\|_2} \right\|_2 \quad (4.3.)$$

unde

- F_1, F_2 sunt fonturile analizate
- $\mathcal{C}$ reprezintă mulțimea caracterelor printabile
- $A_{c_{F_i}}$ imaginea care conține caracterul c randat cu fontul F_i
- n, m dimensiunile imaginilor la care se face randarea
- $\widehat{A_{c_{F_i}}}$ reprezintă imaginea care conține caracterul c randat cu fontul F_i vectorizată

Câteva rezultate ale distanței dintre fonturi sunt prezentate în tabelul de mai jos, menționăm că dimensiunile la care se face randarea caracterelor sunt $n = 64, m = 64$.

Fontul 1 (F_1)		**Fontul 2 (F_2)**		$d_{font}(F_1, F_2)$
Nume font 1	$I_{c_{F_1}}, c = "A"$	Nume font 2	$I_{c_{F_2}}, c = "A"$	0.425300
Guci Font		Lato Regular		
Nume font 1	$I_{c_{F_1}}, c = "L"$	Nume font 2	$I_{c_{F_2}}, c = "L"$	0.150493
Montserrat Regular		Montserrat Medium		
Nume font 1	$I_{c_{F_1}}, c = "M"$	Nume font 2	$I_{c_{F_2}}, c = "M"$	0
Roboto Medium		Roboto Medium		

Tabelul 4.2 *– Rezultatele distanței dintre două fonturi.*

S poate observa că fonturile Gucci și Lato sunt foarte diferite deci și distanța este mare, pe linia a doua avem același font însă cu stil diferit – distanța este neglijabilă între cele două fonturi deoarece au un grad de asemănare mare. Iar pe a treia linie avem același font, și se poate observa că distanța dintre două fonturi identice este 0, ceea ce era de așteptat. Menționăm că au fost

afișate doar o parte din caractere în tabel, iar distanța a fost calculată după toate caractere printabile.

4.2.2. Distanța dintre culorile elementelor de tip text

Pentru a calcula distanța dintre două culori de text am folosit distanța CIEDE2000 [10][12] din spațiul de culoare CIELAB [10]. În continuare vom descrie acest spațiu și această distanță, iar în final vom prezenta câteva rezultate.

Spațiul de culoare CIELAB

Spațiul CIELAB sau CIE L* a* b* este un spațiu de culoare definit de Comisia Internațională de Iluminare. Acest spațiu exprimă culoarea prin trei parametrii L^*, a^*, b^*, unde L^* exprima luminozitatea culorii, a^* si b^* exprimă nuanțele culorii. Unde L^* ia valori de la 0 (negru) la 100 (alb), a^* ia valori de la -128 (verde) pana la 127 (roșu), iar b^* ia valori de la -128 [6](albastru) pana la 127 (galben). Acest spațiu a fost proiectat astfel încât unei cantități de schimbare numerica să corespundă aproximativ aceleiași cantități de modificare a percepției la nivel vizual .

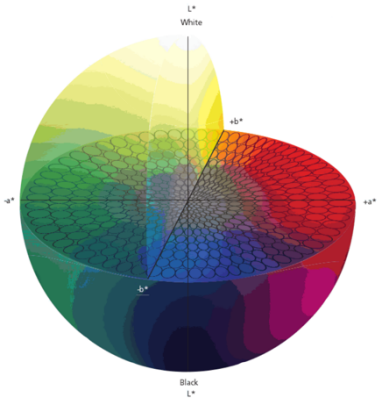


Figura 4.2 *– Spațiul de culoare CIELAB*

Distanța CIEDE2000

Cum ochiul uman este mai sensibil la unele culori decât, rezultă că o metrica bună trebuie să țină cont de acest lucru. Cum o anumită distanță Δ dintr-o parte a spațiului de culoare poate fi mai semnificativă decât aceeași distanță Δ din alta parte a spațiului de culoare obținem că metrica nu trebui să fie simetrică. De aceea CIE a introdus o serie de distanțe care rezolvă acest lucru. Una din cele mai recente distanțe este CIEDE2000 (CIE ΔE^* 2000) [12].

Iar $d_{culoare}(c_1, c_2) \overset{notație}{=} \Delta E^*_{2000}(t(c_1), t(c_2))$, unde t este funcția de transformare a unei culori din RGB în CIE LAB, pentru mai multe detalii consultați **Anexa**. În continuare pentru distanța de culoarea a două texte se va folosii metrica ΔE^*_{2000}.

Câteva rezultate vor fi prezentate în tabelul de mai jos (se va face referire la culorile analizate distanța euclidiană normalizată din spațiul RGB și distanța CIEDE2000 normalizată din spațiul CIELAB).

[6] *Sursa https://www.xrite.com/blog/tolerancing-in-flexo-and-offset-printing*

$$d_{Euclid}\big((\boldsymbol{R_1}, \boldsymbol{G_1}, \boldsymbol{B_1}), (\boldsymbol{R_2}, \boldsymbol{G_2}, \boldsymbol{B_2})\big) = \frac{1}{3 \cdot 255^3} \sqrt{\sum_{C \in \{R,G,B\}} (C_2 - C_1)^2} \qquad (4.4.)$$

Culoare 1	Culoare 1 RGB $(\boldsymbol{R_1}, \boldsymbol{G_1}, \boldsymbol{B_1})$	Culoare 1 LAB $(\boldsymbol{L_1^*}, \boldsymbol{a_1^*}, \boldsymbol{b_1^*})$	Culoare 2	Culoare 2 RGB $(\boldsymbol{R_2}, \boldsymbol{G_2}, \boldsymbol{B_2})$	Culoare 2 LAB $(\boldsymbol{L_2^*}, \boldsymbol{a_2^*}, \boldsymbol{b_2^*})$	Distanța Euclidiană din RGB de normă 1	Distanța CIELAB 2000 de normă 1
	(255, 0, 0)	(53.23, 80.10, 67.22)		(0, 255, 0)	(87.73, -86.18, 83.18)	0.816	0.866
	(255, 0, 0)	(53.23, 80.10, 67.22)		(0, 0, 255)	(32.3, 79.19, -107.86)	0.816	0.528
	(255, 255, 255)	(100, 0.005, -0.01)		(0, 0, 0)	(0, 0, 0)	1	0.999
	(255, 0, 255)	(60.31, 98.25, -60.84)		(255, 0, 0)	(53.23, 80.10, 67.22)	0.577	0.425
	(0, 255, 0)	(87.73, -86.18, 83.18)		(255, 0, 255)	(60.31, 98.25, -60.84)	0.816	0.332
	(0, 0, 255)	(32.3, 79.19, -107.86)		(255, 0, 255)	(60.31, 98.25, -60.84)	0.816	0.640
	(255, 0, 0)	(53.23, 80.10, 67.22)		(255, 255, 255)	(60.31, 98.25, -60.84)	0.816	0.458
	(255, 0, 255)	(60.31, 98.25, -60.84)		(255, 0, 255)	(60.31, 98.25, -60.84)	0	0

Tabelul 4.3 *– Diferențele între spațiile de culoare RGB și CIELAB.*

În **Tabelul 4.3.** este descrisă proiecția în spațiul nativ (RGB) și proiecția în spațiul destinație (CIELAB) pentru o serie de culori. Sunt analizate distanțele în aceste puncte. Pe penultima coloană este distanța euclidiană dintre puncte iar pe ultima coloană distanța CIEDE2000 din spațiul CIELAB. Se poate observa că aceste distanțe nu corespund decât în diferențele alb-negru și în culori egale (acolo unde era de așteptat ca amândouă sa facă 0). În plus, se poate observa în spațiul CIELAB vedele este cel mai aproape de non culoarea negru față de roșu sau albastru, iar în spațiul RGB aceste puncte (roșu, vede și albastru) sunt echidistante față de negru.

Se mai poate observa că albastru este mai aproape de roșu decât vede, acest lucru nu contrazice intuiția că roșu și verde sunt culori opuse, în spectrul vizual.

4.2.3. Compunerea distanței de text din caracteristicile secundare

Pană acum au fost formalizate distanțele a două caracteristici de text, anume: fontul textului și culoarea acestuia. Pentru mărimea textului se alege următoarea distanță:

$$d_{mărime}(m_1, m_2) = \frac{|m_2 - m_1|}{\max\{m_1, m_2\}} \tag{4.5.}$$

Intuiția alegerii acestei distanțe este că va calcula proporția dintre cele două mărimi de text analizate. Este o proporție relativă, adică pentru diferențe mici la fonturi de dimensiuni mici returnează valori mari, iar pentru aceleași diferențe la fonturi de dimensiuni mari returnează valori mici. Ca distanță finală dintre caracteristicile de text se alege:

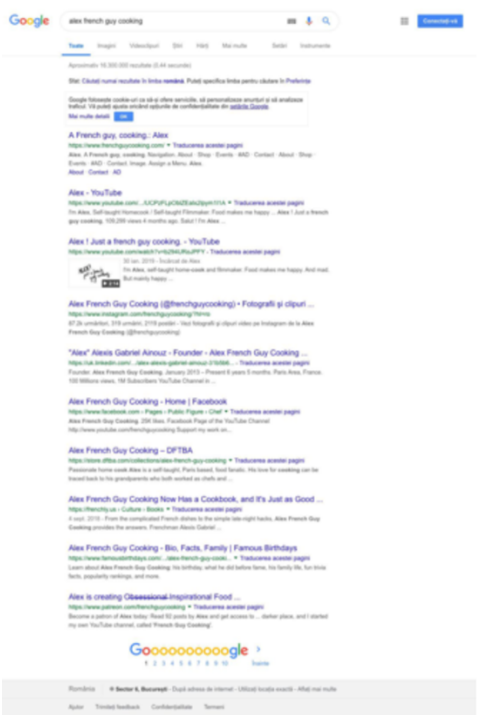
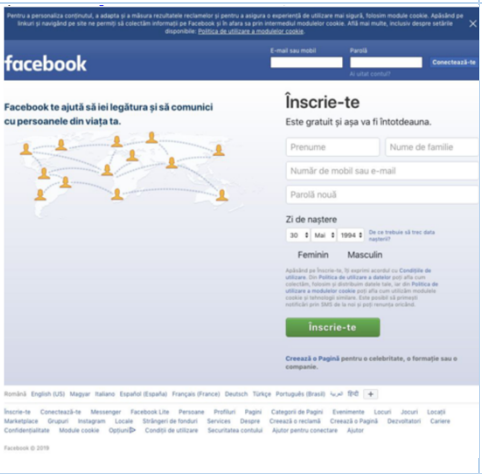
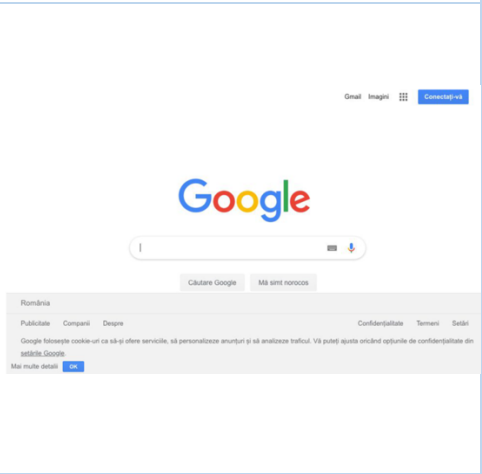
$$d_{text}\,(T_1, T_2) = \delta \cdot d_{font}(F_1, F_2) + \eta \cdot d_{culoare}(C_1, C_2) + \vartheta \cdot d_{mărime}(m_1, m_2) \tag{4.6.}$$

Unde $T_i = (F_i, C_i, m_i)$ textul T_i caracterizat de tripletul font F_i, culoare C_i (in RGB) și mărime m_i iar d_{font} și $d_{culoare}$ au fost definite mai sus. $\delta = \eta = \vartheta = 0.3333$ au fost luate experimental, cu valori egale deoarece atât fontul, mărimea și culoare ca feature de text au cam același impact în designul general.

În continuare având definita noțiunea de distanță dintre caracteristicile de text se introduce distanța dintre caracteristici de text a două interfețe grafice, ca fiind:

$$d_{textFeatures}(I_1, I_2) = \frac{1}{N_1 N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} d_{text}\left(T_{1_i}, T_{2_j}\right) \tag{4.7.}$$

unde T_{y_i}, reprezintă un element de tip text din interfața $I_y, y \in \{1, 2\}, i = \overline{i, N_y}$, N_y numărul total de elemente de tip text din I_y. $d_{textFeatures}$ reprezintă media distanțelor stilurilor de text după produsul cartezian al textelor din cele două interfețe analizate. Câteva rezultate sunt prezentate în tabelul de mai jos:

Interfața I_1	Interfața I_2	Distanța dintre stilurile de text $d_{textFeatures}(I_1, I_2)$
		0.002322314
		0.2384030627

Tabelul 4.4 *– Distanțele dintre caracteristicile de design a două perechi de pagini web.*

În **Tabelul 4.4** sunt analizate prima dată două pagini de căutare google („Alex french guy cooking” iar pe coloana a doua a fost căutat „google fonts”), în acest caz se poate observa că distanța este infimă deoarece paginile sunt randate cu același font, aceeași mărime de text și aproximativ aceleași culori. În al doilea caz unde sunt analizate paginile Google și Facebook care conțin fonturi diferite (la Facebook este predominant fontul Times New Roman iar la Google, Arial), la spațialități și culori diferite, se obține o distanță mare între caracteristicile de tip text.

4.3. DISTANȚA DINTRE FEATURE-URILE SIMBOL

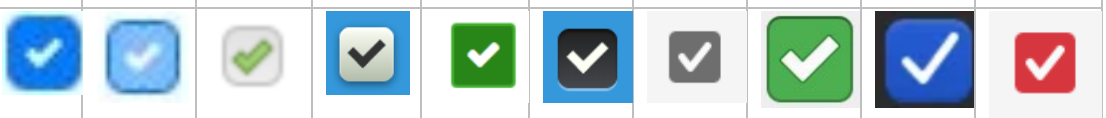
Caracteristicile de tip simbol se împart în mai multe clase precum: clasa buton, clasa radio buton, clasa check box, clasa input și clasa drop down. Iar în interiorul fiecărei clase se dorește găsirea unei măsuri de similarite între elemente. Acest lucru va fi realizat cu ajutorul a cinci rețele siameze [1] care vor fi antrenate să găsească o măsură a disimilarității dintre două elemente din aceeași clasă. În continuare se va prezenta conceptul de rețea siameză și cum este realizată distanța dintre caracteristicile de tip simbol pe baza metricilor date de rețelele siameze. Menționăm că arhitectura pentru fiecare din cele cinci rețele este identică singurul lucru care variază este datasetul.

4.3.1. Rețea siameză

O rețea siameză este o rețea neurală artificială care are ca input două intrări pe care le procesează în tandem. Outputul rețelei poate fi de două tipuri: binar – indică dacă intrările sunt sau nu din aceeași clasă sau real – indică cât de similare sunt cele două intrări. În cazul nostru inputul este o pereche de doua imagini din aceeași clasă, iar outputul este un număr real pozitiv care reprezintă o metrică a asemănării între cele două imagini date la intrare. Ideea din spatele acestui concept este de a calcula o distanță în funcție de hashing-ul local (datasetul). În general rețelele siameze sunt folosite în detecția semnăturilor false, recunoaștere facială sau indexare de documente [1].

4.3.2. Construcția datasetului

Datasetul este extras din datasetul clasificatorului descris în capitolul anterior și presupune un pas în plus de procesare. Acest pas este reprezentat de gruparea în interiorul fiecărei clase a elementelor pe stiluri . Gruparea s-a realizat în funcție de tipul clasei (unele clase au mai multe grupuri decât altele), însă fiecare grup este format din 10 elemente de tip simbol. În tabelul de mai jos se pot observa două grupuri ale clasei check box:

Grup 1										
Grup 2										

Tabelul 4.5 *– Două grupuri din datasetul rețelei siameze de gerare a distanțelor dintre elementele de tip check box*

În continuare vor fi prezentate câteva statistici legate de rapoartele elementelor între clase și al rapoartelor grupurilor din cadrul fiecărei clase.

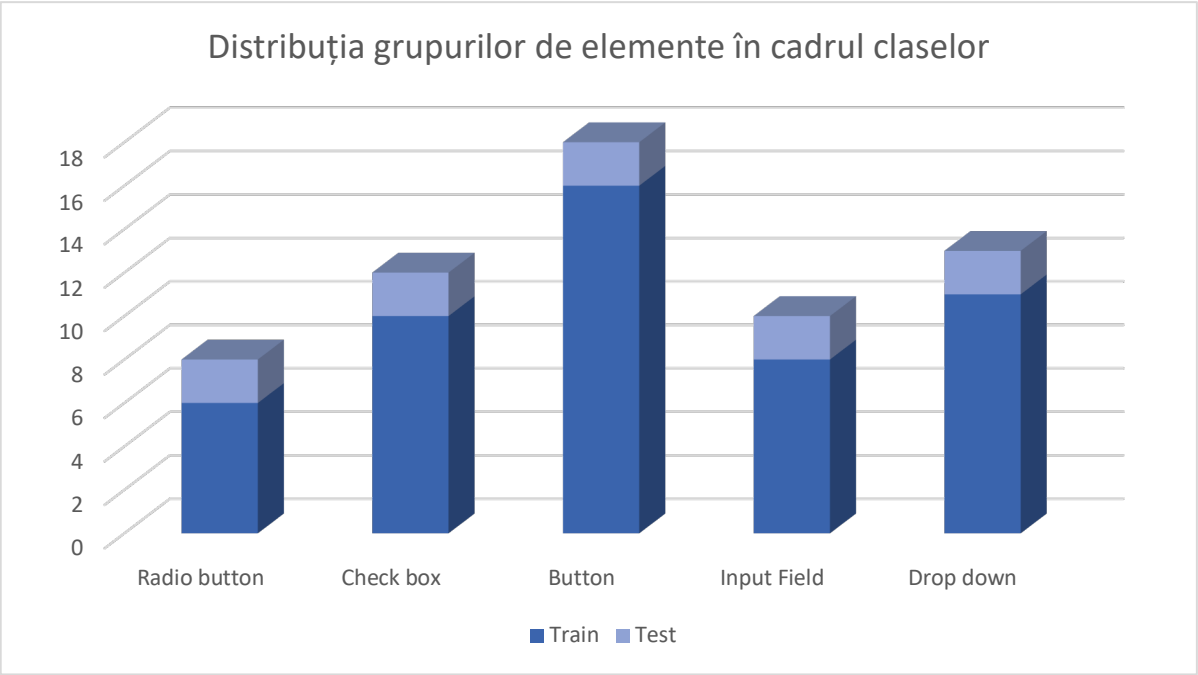


Figura 4.3 *– Compoziția datasetului pentru rețelele siameze*

Imaginile din datasetul descris sunt redimensionate cu păstrarea aspect-ratio-ului pană la dimensiunile maxime de 224×224 (pentru butoane, texte de input și drop down) și la 100×100 (pentru radio si check), apoi se face padding alb al marginilor pană se ajunge la dimensiunile descrise anterior. Ca augmentarea a datelor se folosește o schimbare ce nuanță a culorilor (ColorJitter(brightness=0, contrast=0, saturation=0, hue=0.25) din PyTorch [28]).

4.3.3. Arhitectura

Deoarece sunt analizate elemente sub forma de imagine primele layere ale rețelei sunt convoluții, urmate alte straturi fully-connected. O descriere detaliata a arhitecturii (pentru cazul imaginilor date la intrare de 100×100, pentru 224×224 este foarte asemănător) este:

m. Stratul pentru datele de intrare, care reprezintă o imagine grayscale de dimensiune 100×100;

n. Un strat convoluțional cu 4 filtre de dimensiune (3×3) și pas 1 cu funcția de activare ReLU ($LeLU(x) = \max\{0, x\}$), urmat de un BatchNormalization (cu are rolul de a accelera învățarea prin reducerea covarianței interne a datelor $y = \frac{x-E(x)}{\sqrt{Var(x)+\varepsilon}} \cdot \gamma + \beta$, γ *și* β sunt parametrii învățabili, inițial luați repartizați uniform în intervalul [0, 1] și $\varepsilon = 10^{-5}$) [28], aceasta convoluție se aplică pe imaginea la care s-a făcut padding de 1 (brodarea marginilor imaginii cu 1 pixel, are rolul de a nu altera dimensiunea imaginii);

o. Un strat identic cu stratul b.

p. Un strat convoluțional cu un filtru de dimensiune (3×3) si pas 1, aplicat în aceleași condiții ca straturile de la b. și c.
q. Un strat liniar de 80.000 perceptori care reprezintă imaginea de 100×100 de 8 canale vectorizată, cu funcția de activare ReLU;
r. Un strat liniar de 500 de perceptori, cu aceeași funcție de activare, care este conectat în întregime cu startul e. și stratul g.;
s. Stratul final de 5 perceptori se la care se obține un feature vector de 5 componente (va fi pasat funcției de loss).

În figura de mai jos este prezentată o imagine de ansamblu asupra arhitecturii:

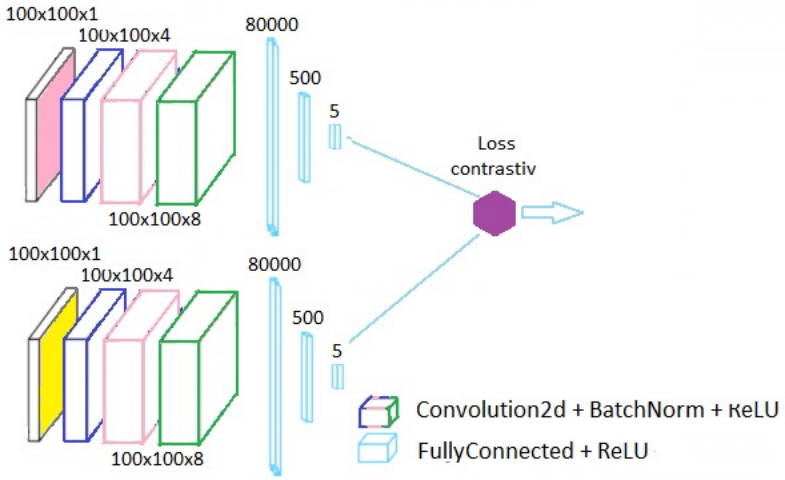


Figura 4.4 *– Arhitectura rețelei siameze*

4.3.4. Funcția de loss

Pentru antrenarea unei rețele siameze se folosesc în general două tipuri de funcții de loss: funcția de loss a tripletului și loss contrastiv. Principiul la funcția de loss a tripletului este de a compara inputul (feature vectorul de 5 componente) cu un vector pozitiv (provenit din același grup) și unul negativ (provenit din grup diferit). Vectorul negativ forțează învățarea în timp ce vectorul pozitiv acționează pe post de regularizator. O funcție de loss contrastivă este folosită pentru a învață parametrii W a unei funcții parametrizate (în particular un model) astfel încât vecinii evaluați prin aceasta funcție sunt plasați la depărtate mică iar non-vecinii evaluați au valoare mare (adică vecinii sunt împinși în același cluster), aici nu se ține cont de alegerea inputului (adică un exemplu pozitiv și unul negativ) [30]. Pentru antrenarea rețelelor siameze din aceasta lucrare de licența s-a folosit funcția de loss contrastiv:

$$loss_{contrastive}(y_1, y_2) = \bigwedge\left(\frac{1}{2}(1-Y)\|y_1 - y_2\|_2^2 + \frac{1}{2}Y(\max\{0, m - \|y_1 - y_2\|_2^2\})^2\right) \quad (4.8.)$$

unde

- $(y_1, y_2) = f(x_1, x_2)$ (reprezintă feature vectori/ vectorii de logiți)
- x_i este imaginea analizată, $i \in \{1, 2\}$;
- f funcția care reprezintă operația de forward în rețeaua siameza pentru o intrare (x_1, x_2);
- Y reprezintă eticheta perechii (x_1, x_2), $Y = \begin{cases} 0, dacă\ x_1 si\ x_2\ sunt\ din\ același\ grup \\ 1, dacă\ x_1 si\ x_2\ nu\ sunt\ din\ același\ grup \end{cases}$
- m reprezintă o margine, care limitează învățarea dacă $\|y_1 - y_2\|_2 > m$ (a fost ales $m = 3$);
- $\wedge$ reprezintă funcția care calculează media componentelor unui vector $\wedge(x) = \frac{1}{n}\sum_{i=1}^{n} x_i$, $x \in \mathbb{R}^n$.

Se poate observa că atunci când $x_1 și\ x_2$ provin din același grup $Y = 0$, funcția de loss devine $loss_{contrastive}(y_1, y_2) = \wedge\left(\frac{1}{2}\|y_1 - y_2\|_2^2\right)$, iar cand exemplele sunt luate din grupuri diferite funcția de loss devine $loss_{contrastive}(y_1, y_2) = \wedge\left[\frac{1}{2}(\max\{0, m - \|y_1 - y_2\|_2^2\})^2\right]$.

Gradientul funcției de loss este:

$$\frac{\partial loss_{contrastive}}{\partial W} = \begin{cases} \|y_1 - y_2\|_2 \dfrac{\partial \|y_1 - y_2\|_2}{\partial W}, dacă\ x_1 si\ x_2\ sunt\ din\ același\ grup \\ 0,\ \ dacă\ x_1 si\ x_2\ sunt\ din\ același\ grup\ și\ \|y_1 - y_2\|_2 > m \\ -(m - \|y_1 - y_2\|_2) \dfrac{\partial \|y_1 - y_2\|_2}{\partial W}, altfel \end{cases} \quad (4.9.)$$

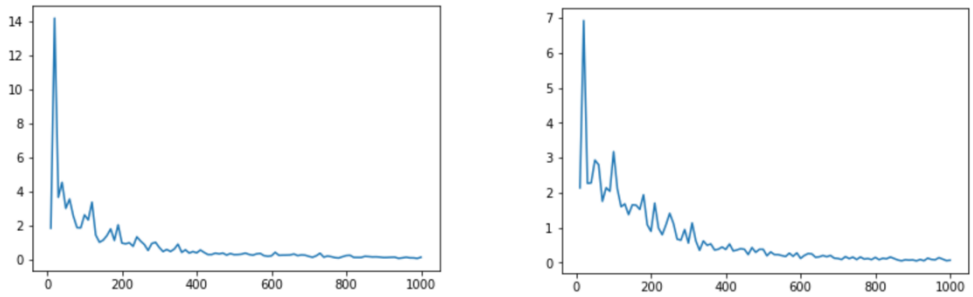


Figura 4.5 *– Graficul evoluției funcției de loss pentru rețelele siameze ale claselor radioButton și checkBox*

Antrenarea unei rețele neurale presupune în general minimizarea unei funcții de loss (în cazul nostru lossul contrastiv). O metoda clasică de a face acest lucru este coborârea pe gradientul funcției de loss numita și gradient descent. Aceasta metoda presupune găsirea punctului de minim local al unei funcții (funcția de loss).

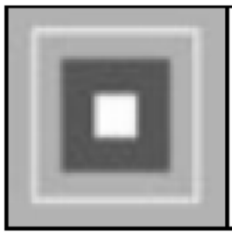
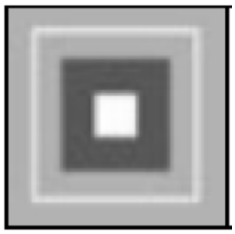
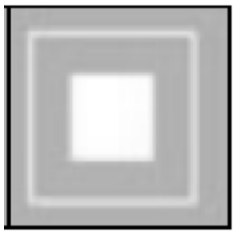
4.3.5. Formalizare și rezultate

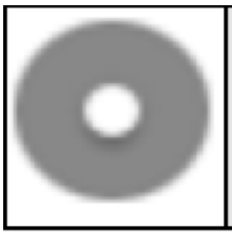
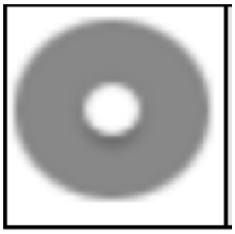
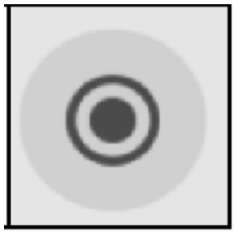
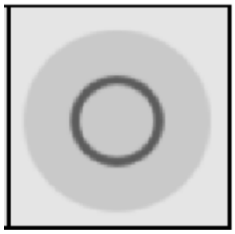
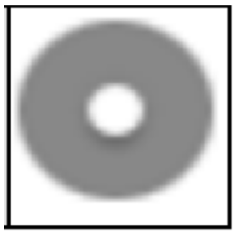
Rețelele astfel obținute vor determina pentru fiecare două elemente din interiorul aceleași clase o măsura a disimilarității dintre ele. Aceasta metrică nu este altceva decât outputul rețelei și se definește ca:

$$d_{simbol}(S_1, S_2, t) = \|\pi_1(f_t(S_1, S_2)) - \pi_2(f_t(S_1, S_2))\|_2 \qquad (4.10.)$$

- unde S_1, S_2 sunt simbolurile analizate (imagini),
- π_1, π_2 funcțiile proiecție canonică din $\mathbb{R}^2$
- t tagul simbolurilor analizate (tipul lor)
- f_t funcția de forward în rețeaua siameză corespunzătoare tagului t (de exemplu dacă S_1 $și$ S_2 sunt butoane atunci $t = "buton"$ și se va folosi rețeaua siameză antrenată pe butoane)

Deoarece nu știm ground truth-ul (adică distanța reală din punct de vedere al percuției ochiului și cortexului uman) nu se poate calcula acuratețea acestor rețele, singurele măsurători fezabile fiind cele vizuale. În tabelul de mai jos sunt prezente câteva rezultate:

Tip	**Simbol 1 (x_1)**	**Simbol 2 (x_2)**	**Distanța**
Check box			1.14
Check box			0.79
Check box			0.38
Check box			0

Radio button			4.15
Radio button			3.09
Radio button			1.23
Radio button			1.79

Tabelul 4.6 *– O serie de câteva distanțe învățate de rețelele siameze corespunzătoare claselor de simboluri*

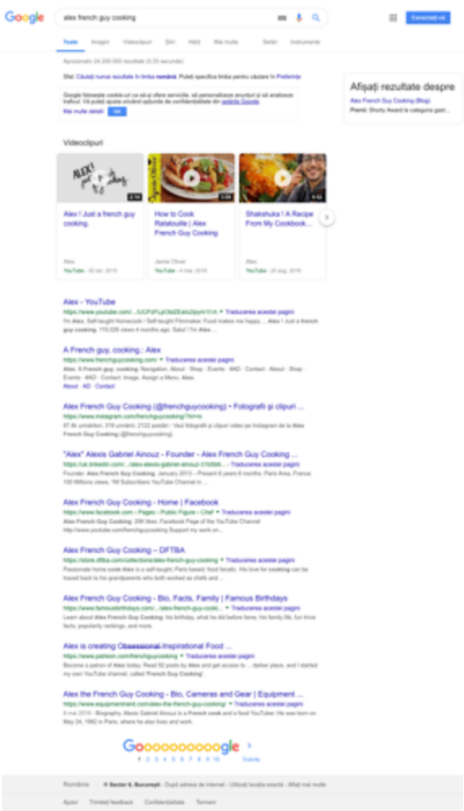
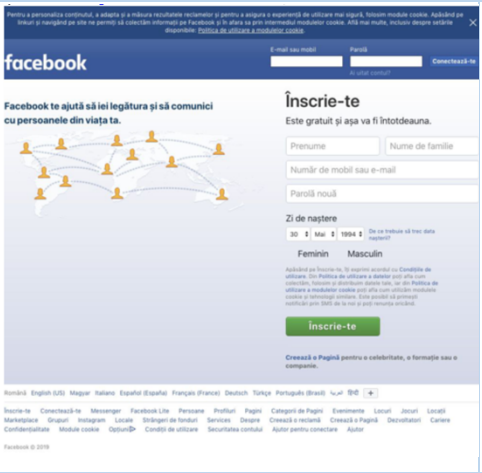
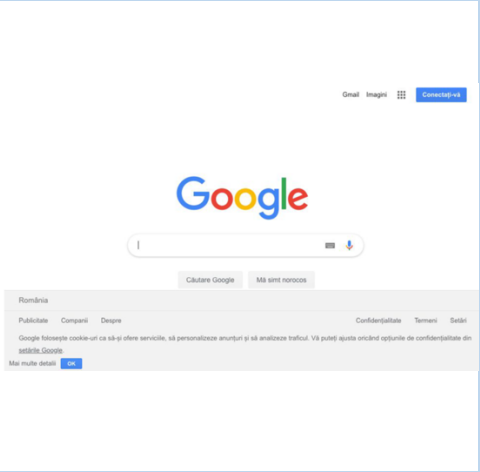
4.3.6. Distanța între simbolurile interfețelor grafice

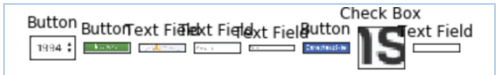
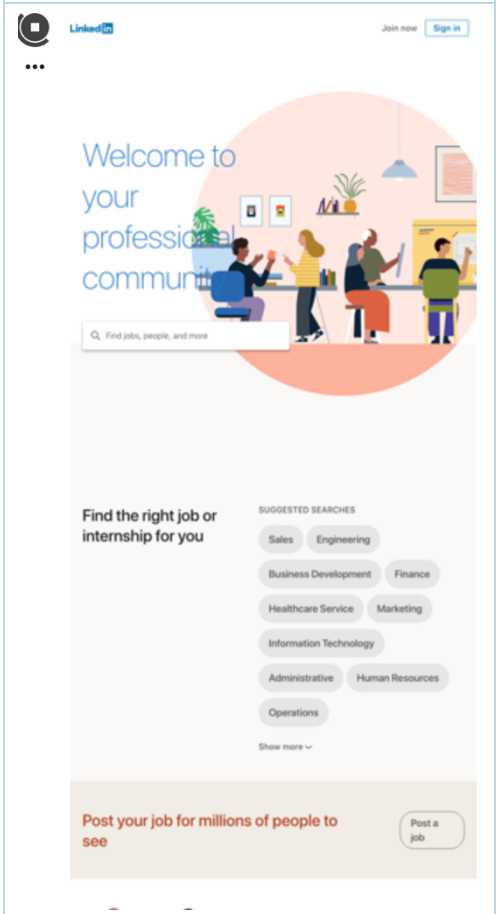
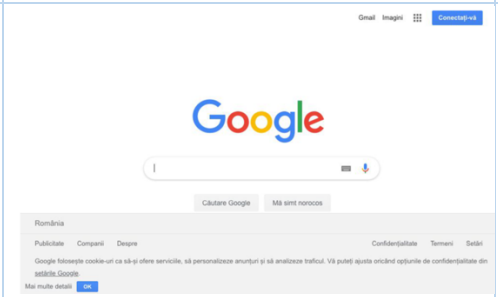
Având formalizată noțiunea de distanță între două simboluri nu rămâne decât să formalizăm distanța între simbolurile a două interfețe grafice:

$$d_{symbolFeatures}(I_1, I_2) = \frac{1}{\sum_{t\in\mathcal{T}} M_{1_t} + M_{2_t}} \sum_{t\in\mathcal{T}} \frac{M_{1_t} + M_{2_t}}{\left(\max\limits_{\substack{i=\overline{1\ldots M_{1_t}},\\ j=\overline{1\ldots M_{2_t}}}} d_{simbol}\left(S_{1_i}, S_{2_j}, t\right)\right)\left(M_{1_t} M_{2_t}\right)} \sum_{i=1}^{M_{1_t}} \sum_{j=1}^{M_{2_t}} d_{simbol}\left(S_{1_i}, S_{2_j}, t\right) \quad (4.11.)$$

unde S_{y_i}, reprezintă un simbol din clasa t din interfața $I_y, y \in \{1, 2\}, i = \overline{1, M_{y_t}}$, M_y numărul total de elemente de tip simbol din clasa $t \in \mathcal{T}$ din I_y iar $\mathcal{T}$ reprezintă mulțimea de clase/ taguri.

Factorul $\frac{1}{\left(\max\limits_{\substack{i=\overline{1\ldots M_{1_t}},\\ j=\overline{1\ldots M_{2_t}}}} d_{simbol}\left(S_{1_i}, S_{2_j}, t\right)\right)\left(M_{1_t} M_{2_t}\right)}$ are rol de regularizare a distanțelor deoarece rezultatele obținute de la rețeaua siameza nu sunt valori între 0 si 1. Câteva rezultate sunt prezentate în tabelul de mai jos:

Interfața I_1	Interfața I_2	Distanța dintre simboluri $d_{symbolFeatures}(I_1, I_2)$
		0.330348895
		0.418547497

		
		0.136989608

Tabelul 4.7 *– Rezultatele distanței dintre simboluri*

4.4. FORMALIZAREA ȘI IMPLEMENTAREA DISTANȚEI DE STIL

4.4.1. Formalizare

Având definite distanțele auxiliare de mai sus ($d_{colorFeatures}$, $d_{textFeatures}$ și $d_{symbolFeatures}$) se introduce distanța generală între două interfețe grafice (pe scurt distanța de stil) ca fiind următoarea combinație baricentrică a celor trei distanțe:

$$d_{designFeatures} = \xi \cdot d_{colorFeatures} + \rho \cdot d_{textFeatures} + \phi \cdot d_{symbolFeatures} \qquad (4.12.)$$

unde $\xi = \rho = \phi = 0.3333$ sunt parametrii combinației baricentrice și au fost luați egali experimental (deoarece toate tipurile de caracteristici de design au aproximativ același impact).

Pentru ca $d_{designFeatures}$ să reprezinte o distanță trebuie să respecte următoarele proprietăți:

- non-negativitate: $d_{designFeatures}(I,J) \geq 0$
- anularea doar în puncte egale: $d_{designFeatures}(I,J) = 0 \Leftrightarrow I\ si\ J$ au același stil
- simetrie: $d_{designFeatures}(I,J) = d_{designFeatures}(J,I)$
- inegalitatea triunghiului:

$$d_{designFeatures}(I,K) \leq d_{designFeatures}(I,J) + d_{designFeatures}(J,K)$$

unde I, J, K sunt trei interfețe grafice analizate.

Non-negativitatea este verificată deoarece acesta distanță este o compunere de funcții pozitive. Anularea în puncte egale este verificata în implicația inversă, iar în implicația directă se verifică doar dacă stilul restricționat la caracteristicile analizate este egal între cele două interfețe. Simetria este verificată deoarece această distanță este o compunere de funcții simetrice. Iar inegalitatea triunghiului este trivială deoarece distanța de stil este o compunere de distanțe consacrate.

4.4.2. Implementare

Aceasta distanță este implementată în Python și este construita pe baza extractorului de stil din capitolul anterior. Are ca input două fișiere JSON cu caracteristicile de design construite de extractorul de stil. Deoarece extractorul este implementat doar pentru interfețe grafice de tip web rezultă că și această distanță va fi implementată implicit doar pentru web, conceptele generale rămânând neschimbate. Structura fișierelor JSON se află în **Anexă**, acestea conțin informațiile în clar despre culori, stil de text, simboluri sub forma de imagine (informația primită sub formă de bytearray) și captura de ecran a întregii pagini (informație serializată).

Cele două fișiere JSON se parsează, se extrage întreaga informație și se deserializează imaginile.

Pentru calculul distanței dintre culorile pagini se construiesc cele trei histograme de culoare pentru fiecare pagină (două neorientate construite pe baza informației în clar și histograma de context care este orientată și construită din captura de ecran), apoi se calculează cele trei distanțe Bhattacharyya dintre perechile de histograme construite.

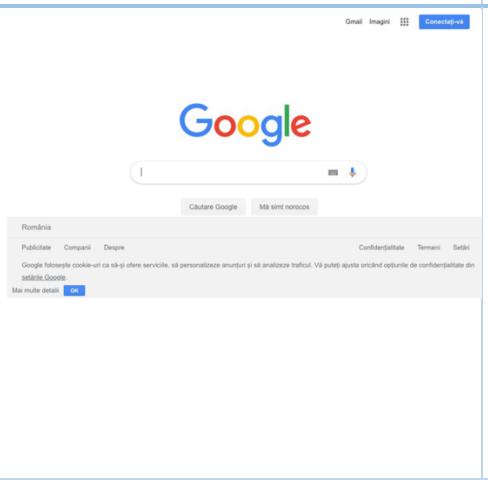
Pentru distanța dintre stiluri de text se face o serie de requesturi la Google Fonts pentru a descărca toate fonturile din fișiere JSON, apoi se randează două cate două fonturi la o dimensiune prestabilită (la noi 64) și se calculează distanța dintre fonturi. După se calculeze distanța dintre culorile și mărimea textului obținându-se distanța dintre caracteristicile de tip text.

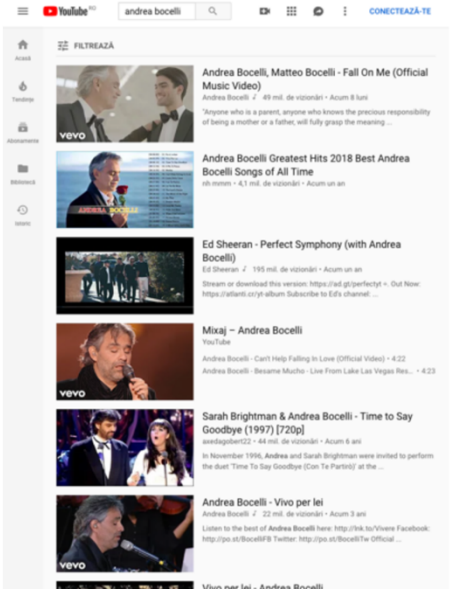
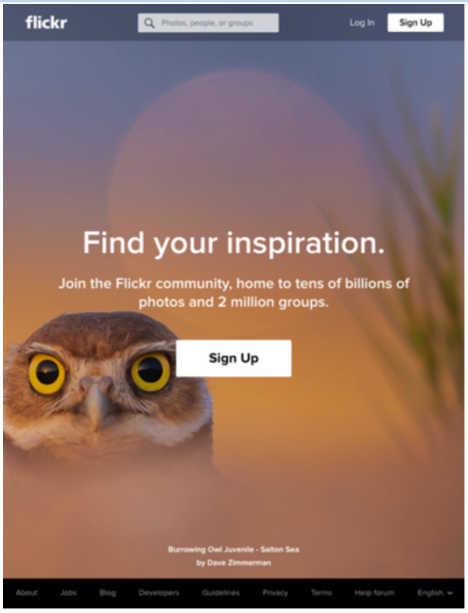
Pentru calcularea distanței dintre simboluri se utilizează cele cinci modele antrenate. Se trec perechile de simboluri din aceleași clase prin rețelele siameze corespunzătoare și se obțin distanțele din care se compune distanța dintre aceste caracteristici de design conform formulei descrise anterior în lucrare.

În final se compun distanțele de mai sus și se obține distanța totală dintre caracteristicile de design ale unei pagini web, formula distanței de stil se află în **Anexă**.

4.4.3. Rezultate

Câteva rezultate ale distanței de stil sunt prezentate în **Tabelul 4.8**

Interfata I_1	Interfata I_2	Distanța dintre stiluri $d_{designFeatures}(I_1, I_2)$
		0.355394428

		0.15071016473
		0.34492562515

Tabelul 4.8 *– Rezultatele distanței de stil pentru o serie de pagini web*

CONCLUZII

În acesta parte se face o sumarizare a rezultatelor atât pentru extractorul de stil cât și pentru metrica introdusă – distanța de stil Se face o scurta trecere în revista a contribuției în domeniu și se prezintă munca viitoare.

CONCLUZII

Determinarea caracteristicilor de design ale unei interfețe grafice presupune o analiză complexă [14], deoarece surprinde un task general. Designul unei aplicații poate arata în nenumărate moduri, sigurul obstacol fiind creativitatea dezvoltatorului. Această lucrare a prezentat o modalitate de a extrage feature-urile de design dintr-o interfață grafică sub forma a trei clase mari de elemente de design: culori, stil de text și simboluri, care speră să reducă efortul uman depus pentru a extrage aceste design features în mod manual la un nivel minim.

Funcționalitatea extracției este implementată pe două servicii externe utilizatorului: unul de NodeJS care face extracție de culori, gradienți și stil de text și unul de Python care face extracție de simboluri. Acestea sunt mascate de pluginul de Adobe XD care este singura entitate ce interacționează cu utilizatorul. Culorile, gradienții și stilul de text sunt preluate din proprietățile aferente elementelor din interfața grafică cu o acuratețe de 100% iar simbolurile sunt extrase cu o multitudine de algoritmi de computer vision și inteligență artificială din varianta imagista a interfeței grafice cu o acuratețe de aproximativ 80% și recall de 92%. O îmbunătățire ce poate fi adusă extracției de simboluri ar putea consta în construirea unui dataset mai mare decât cel format deja astfel încât localizarea simbolurilor să nu se realizeze pe bază de template matching ci pe bază de activare a unei rețele neuronale adânci. O alta abordare constă în construcția unui dataset pentru rețele recurente (care să facă extracția din varianta codului sursă a interfeței) sau al unui dataset pentru rețele GAN (generative adversarial network).

Un concept nou până în prezent este introdus în acesta lucrare de licența, anume distanță dintre stilul a două interfețe grafice. Acesta metrică măsoară disimilaritatea din punct de vedere al designului pentru două interfețe grafice de același tip. Are la bază mai multe metrici consacrate precum distanța Bhattacharyya dintre histogramele de culoare sau distanța Chord dintre imagini vectorizate dar și metrici mai extravagante învățabile generate de o serie de rețele siameze. O îmbunătățire ce poate fi adusă acestei distanțe este învățarea constantelor de importanță a fiecărui tip de caracteristică (constantele combinațiilor baricentrice), în prezent acestea sunt luate experimental. Aceasta

îmbunătățire se poate face cu alta rețea siameză sau cu diferite tehnici de feature selection. O alta îmbunătățire poate fi îmbogățirea dataseturilor pentru rețelele siameze. O parte din utilitățile acestei distanțe sunt căutarea automată (web crawling) a interfețelor grafice care au un stil asemănător cu o interfață fixată, în mod particular căutarea dezvoltatorilor care adoptă un anumit stil sau cercetarea evoluției trendului în materie de design al interfețelor grafice.

Toate aceste funcționalități au fost implementate doar pentru aplicații și pagini web însă conceptele pot fi extinse ușor și pentru restul tipurilor de interfețe grafice mai ales că simbolurile sunt examinate din punct de vedere imagistic având independență de tipul interfeței din acest punct de vedere.

BIBLIOGRAFIE

[1] Andrea Loreggia, Nicholas Mattei, Francesca Rossi, K. Brent Venable. *CPDist: Deep Siamese Networks for Learning Distances Between Structured Preferences*. 1 sept 2018 (https://arxiv.org/abs/1809.08350)

[2] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, Jiajun Liang. *EAST: An Efficient and Accurate Scene Text Detector*. 10 iul 2017 (https://arxiv.org/abs/1704.03155)

[3] J. Illingworth, J. Kittler. *The Adaptive Hough Transform*. p. 690-698. 1987

[4] Hong Liu, Yueliang Qian. Shouxun Lin *Detecting Persons using Hough Circle Transform in Surveillance Video*. ian 2010

[5] Gary Bradski, Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library.* p.151-160, p.214-218. 24 sept 2008 (Canny + Template matching histograme de culoare)

[6] Gary Bradski, Adrian Kaehler *Learning OpenCV: Computer Vision with the OpenCV Library*. p.199-206. 24 sept 2008 (distanța dintre histogramele de culoare)

[7] Rasmus Rothe, Matthieu Guillaumin, Luc Van Gool. *Non-Maximum Suppression for Object Detection by Passing Messages between Windows.* Asian Conference on Computer Vision (ACCV), nov 2014

[8] Luca Lorenzi, Farid Melgani, Grégoire Mercier. *Inpainting Strategies for Reconstruction of Missing Data in VHR Images*. IEEE Geoscience and Remote Sensing Letters (Volumul: 8), sept. 2011

[9] Wang, Zhou, Bovik A.C., Sheikh, H.R., Simoncelli E.P. *Image quality assessment: from error visibility to structural similarity*. IEEE Transactions on Image Processing, 4 apr 2004.

[10] Peter Kovesi. *Good Colour Maps: How to Design Them*. 12 sept 2015 (https://arxiv.org/abs/1509.03700)

[11] Chaur-Chin Chen, Hsueh-Ting Chu. *Similarity Measurement Between Images*. Conference: Computer Software and Applications Conference, 2005. COMPSAC 2005. 29th Annual International, Volumul: 2

[12] Lindbloom, Bruce Justin. *Delta E (CIE 2000)*. Brucelindbloom.com. 2009

[13] William Lidwell, Kritina Holden, Jill Butler. *Universal Principles of Design*. Rockport Publishers, 2010

[14] Radu Tudor Ionescu, Bogdan Alexe, Marius Leordeanu, Marius Popescu, Dim P. Papadopoulos, Vittorio Ferrari. *How hard can it be? Estimating the*

difficulty of visual search in an image, 3 mai 2017 (https://arxiv.org/abs/1705.08280)

[15] Jayanth Koushik. *Understanding Convolutional Neural Networks*, 30 mai 2016 (https://arxiv.org/abs/1605.09081)

[16] Sanja Fidler. *Sliding Windows*. Department of Computer Science University of Toronto - CS420 lecture 17

[17] Adobe XD Platform, API references and Samples (https://adobexdplatform.com/)

[18] Documentația Headless Chrome (https://developers.google.com/web/updates/2017/04/headless-chrome)

[19] Documentația XMLHttpRequest - MDN WebDocs (https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest)

[20] Documentația PhantomJs (http://phantomjs.org/)

[21] Documentația de Chrome extensions (https://developer.chrome.com/extensions)

[22] Documentația de WebSocket - MDN WebDocs (https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API)

[23] John Wiley, Sons Windows. *Server Administration Fundamentals*. Microsoft Official Academic Course. 2011. p. 2–3.

[24] Documentația ws: a Node.js WebSocket library (https://www.npmjs.com/package/ws)

[25] Documentația Puppeteer (https://www.npmjs.com/package/puppeteer)

[26] 18181 color-names (https://www.npmjs.com/package/color-name-list)

[27] Documentația de OpenCV

[28] Documentația PyTorch (https://pytorch.org/docs/stable/index.html)

[29] Documentația Pyppeteer (https://miyakogi.github.io/pyppeteer/)

[30] Raia Hadsell, Sumit Chopra, Yann LeCun. *Dimensionality Reduction by Learning an Invariant Mapping*. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition

[31] Hiroshi Fukui, Tsubasa Hirakawa, Takayoshi Yamashita, Hironobu Fujiyoshi. *Attention Branch Network: Learning of Attention Mechanism for Visual Explanation*. 10 Apr 2019 (https://arxiv.org/abs/1812.10025)

ANEXĂ

Manifestul Plugin-ului de Adobe XD

```
{
    "id": "fba564af",
    "author": "Marian Lupascu",
    "name": "Styling Extractor V2",
    "version": "2.0.0",
    "host": {
        "app": "XD",
        "minVersion": "13.0.0"
    },
    "uiEntryPoints": [
        {
            "type": "menu",
            "label": "Styling Extractor",
            "commandId": "menuCommand",
            "shortcut": { "mac": "Cmd+Shift+X", "win": "Ctrl+Shift+X" }
        }
    ]
}
```

Structura fișierelor json

```
//structura fisierului de la serviciul de NodeJS

{
        info: {
        colors: {
            pureColors: [{
                red: int,
                green: int,
                blue: int,
                opacity: float
            }, ... ],
            pureColorsBackground: [{
                red: int,
                green: int,
                blue: int,
                opacity: float
            }, ... ] ,
            linearGradients: [[{
                red: int,
                green: int,
                blue: int,
                opacity: float,
                stop: float

            }, ...] , ...],
            radialGradients: [[{
                red: int,
                green: int,
                blue: int,
```

```
                opacity: float,
                stop: float

            }, ...] , ...]
        },
        fonts: [{
            fontFamily: string,
            fontStyle: string,
            fontSize: string,
            colorFill: string,
            charSpacing: string,
            lineSpacing: string,
            underline: string
        }, ...],
        title: string,
        infoColorForDistance: //same as colors plus width and height
    },
    time: float,
    done: boolean
}

//structura fisierului de la serviciul de Python
{
    'done': boolean,
    'time': float,
    'info': {
        'UIelementsReal': [{
            'bytes': bytesArray,
            'height': int,
            'width': int,
            'channels': int, // 1 sau 3
            'type': int //intre 0 si 5
         }, ...],
         'VectorInfo': [{
            'color': {
                'r': int,
                'g': int,
                'b': int
            },
            'height': int,
            'width': int
        }, ...] ,
        'printScreen': {
            'bytes': bytesArray,
            'heightOriginal': int,
            'widthOriginal': int,
            'height': int, //deoarece daca este prea mare este redimensionata
            'width': int
        }
    }
}
```

Distanța dintre caracteristicile de design a două interfețe grafice:

$$
\begin{aligned}
d_{designFeatures} &= \xi \left\{ \chi \cdot \left[1 - \frac{1}{N \cdot \sqrt{\sum_{i=1}^{n} H_{I_{1_C}}(b_i) \cdot \sum_{i=1}^{n} H_{I_{2_C}}(b_i)}} \cdot \sum_{i=1}^{n} \sqrt{H_{I_{1_C}}(b_i) \cdot H_{I_{2_C}}(b_i)} \right] \right. \\
&+ \psi \left[1 - \frac{1}{N \cdot \sqrt{\sum_{i=1}^{n} H_{I_{1_F}}(b_i) \cdot \sum_{i=1}^{n} H_{I_{2_F}}(b_i)}} \cdot \sum_{i=1}^{n} \sqrt{H_{I_{1_F}}(b_i) \cdot H_{I_{2_F}}(b_i)} \right] \\
&\left. + \omega \left[1 - \frac{1}{N \cdot \sqrt{\sum_{i=1}^{n} H_{I_{1_G}}(b_i) \cdot \sum_{i=1}^{n} H_{I_{2_G}}(b_i)}} \cdot \sum_{i=1}^{n} \sqrt{H_{I_{1_G}}(b_i) \cdot H_{I_{2_G}}(b_i)} \right] \right\} \\
&+ \rho \left\{ \frac{1}{N_1 \times N_2} \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} \left[\delta \frac{1}{|\mathcal{C}| \cdot n \cdot m} \cdot \sum_{c \in \mathcal{C}} \left\| \frac{\widehat{A_{c_{F_1}}}}{\left\| \widehat{A_{c_{F_1}}} \right\|_2} - \frac{\widehat{A_{c_{F_2}}}}{\left\| \widehat{A_{c_{F_2}}} \right\|_2} \right\|_2 + \eta \sqrt{\left(\frac{\Delta L'}{k_L \cdot S_L} \right)^2 + \left(\frac{\Delta C'}{k_C \cdot S_C} \right)^2 + \left(\frac{\Delta H'}{k_H \cdot S_H} \right)^2 + R_T \cdot \frac{\Delta C'}{k_C \cdot S_C} \cdot \frac{\Delta H'}{k_H \cdot S_H}} \right. \right. \\
&\left. \left. + \vartheta \frac{\frac{m_1 + m_2}{2} - \min\{m_1, m_2\}}{|m_2 - m_1|}) \right] \right\} \\
&+ \phi \left\{ \frac{1}{\sum_{t \in \mathcal{T}} M_{1_t} + M_{2_t}} \sum_{t \in \mathcal{T}} \frac{M_{1_t} + M_{2_t}}{\left(\max\limits_{\substack{i=\overline{1 \ldots M_{1_t}}, \\ j=\overline{1 \ldots M_{2_t}}}} \|\pi_1(f_t(S_1, S_2)) - \pi_2(f_t(S_1, S_2))\|_2 \right) (M_{1_t} M_{2_t})} \sum_{i=1}^{M_{1_t}} \sum_{j=1}^{M_{2_t}} \|\pi_1(f_t(S_1, S_2)) - \pi_2(f_t(S_1, S_2))\|_2 \right\}
\end{aligned}
$$

Transformările din modelul RGB în spațiul CIE LAB se fac printr-un spațiu auxiliar CIE XYZ:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 3.240479 & -1.537150 & -0.498535 \\ -0.969256 & 1.875992 & 0.041556 \\ 0.055648 & -0.204043 & 1.057311 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

$$L^* = 116 \cdot f\left(\frac{Y}{Y_n}\right) - 16$$

$$a^* = 500 \cdot \left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right)$$

$$b^* = 200 \cdot \left(f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right)$$

Unde $f(x) = \begin{cases} \sqrt[3]{x}, & x > \frac{6}{29} \\ \frac{t}{3\delta^2} + \frac{4}{29}, & x \leq \frac{6}{29} \end{cases}$, $X_n = 95.0489, Y_n = 100, Z_n = 08.884$ valori de normalizare sub iluminatul D65.

- $l_i = (L_i^*, a_i^*, b_i^*), i \in \{1, 2\}$
- $\Delta L' = L_2^* - L_1^*$
- $\bar{C} = \frac{\sqrt{a_1^{*2} + b_1^{*2}} + \sqrt{a_2^{*2} + b_2^{*2}}}{2}$
- $C_i' = \sqrt{\left[a_i^* + \frac{a_i^*}{2}\left(1 - \sqrt{\frac{\bar{C}^7}{\bar{C}^7 + 25^7}}\right)\right]^2 + b_i^{*2}}, i \in \{1, 2\}$
- $\Delta C' = C_2' - C_1'$
- $atan2(x, y) = \begin{cases} arctg\left(\frac{y}{x}\right), & x > 0 \\ arctg\left(\frac{y}{x}\right) + \pi, x < 0 \text{ și } y \geq 0 \\ arctg\left(\frac{y}{x}\right) - \pi, x < 0 \text{ și } y < 0 \\ \frac{\pi}{2}, & x = 0 \text{ și } y > 0 \\ -\frac{\pi}{2}, & x = 0 \text{ și } y < 0 \\ nedefinit, x = 0 \text{ și } y = 0 \end{cases}$
- $h_i' = atan2\left[b_i^*, a_i^* + \frac{a_i^*}{2}\left(1 - \sqrt{\frac{\bar{C}^7}{\bar{C}^7 + 25^7}}\right)\right], i \in \{1, 2\}$
- $\Delta h' = \begin{cases} h_2' - h_1', & |h_1' - h_2'| \leq 180° \\ h_2' - h_1' + 360°, & |h_1' - h_2'| > 180° \text{ și } h_2' \leq h_1' \\ h_2' - h_1' - 360°, & |h_1' - h_2'| > 180 \text{ și } h_2' > h_1' \end{cases}$
- $\Delta H' = 2\sqrt{C_2' C_1'} \sin\left(\frac{\Delta h'}{2}\right)$

- $\widetilde{H'} = \begin{cases} h_2' - h_1', & |h_1' - h_2'| \leq 180° \\ h_2' - h_1' + 360°, & |h_1' - h_2'| > 180° \; si \; h_2' \leq h_1' \\ h_2' - h_1' - 360°, & |h_1' - h_2'| > 180 \; si \; h_2' > h_1' \end{cases}$
- $T = 1 - 0{,}17\cos\left(\widetilde{H'} - 30°\right) + 0{,}24\cos\left(2\widetilde{H'}\right) + 0{,}32\cos\left(3\widetilde{H'} + 6°\right) - 0{,}2\cos\left(4\widetilde{H'} - 63°\right)$
- $k_L = k_C = k_H = 1$
- $S_L = 1 + \frac{0{,}015\left(\frac{L_1^* + L_2^*}{2} - 50\right)^2}{\sqrt{20 + \left(\frac{L_1^* + L_2^*}{2} - 50\right)^2}}$
- $S_C = 1 + 0{,}045 \cdot \frac{C_1' + C_2'}{2}$
- $S_H = 1 + 0{,}015 \cdot T \cdot \frac{C_1' + C_2'}{2}$
- $R_T = -2\sqrt{\frac{\bar{C}^7}{\bar{C}^7 + 25^7}} \cdot sin\left[60° \cdot e^{-\left(\frac{\widetilde{H'} - 275°}{25°}\right)^2}\right]$