



UNIVERSITATEA DIN BUCUREȘTI



FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ

SPECIALIZAREA INFORMATICĂ

Lucrare de licență

APLICAȚIE WEB PENTRU CONSILIERE ÎN DOMENIUL VESTIMENTAR

Absolvent Ganea Carina-Eliza

**Coordonator științific
Conf.dr. Radu Boriga**

București, iunie 2021

Rezumat

Aplicația de consultanță în domeniul vestimentar are la bază modelul de Image Recognition ResNet50V2, reantrenat pe un set propriu de date. Modelul ales fiind foarte adânc și rezistent atât la overfitting, cât și la problema dispariției gradientului, acesta este capabil de învățarea prin transfer, pornind de la modelul pre-antrenat pe setul de date ImageNet și re-antrenând mai fin pe setul mai restrâns de date. Fiind un model foarte adânc și complex, Microsoft ResNet50V2 este capabil să creeze sisteme de clasificare foarte precise, din care pot fi extrase trăsături dominante utile în căutarea inversă a articolelor de îmbrăcăminte. Prin reclasificarea tuturor trăsăturilor extrase de modelul ResNet50V2, pot fi furnizate imagini similare cu cele analizate.

Obiectivul aplicației este cel de a sfătui utilizatorul neinstruit în vederea creării unei vestimentații care să încorporeze un articol de îmbrăcăminte deținut de acesta. Cu ajutorul performanțelor recente în domeniul identificării imaginilor și clasificării acestora, este posibilă construirea unei aplicații web care să simuleze un consultant în domeniul vestimentar.

Abstract

The fashion consultant web application relies on the ResNet50V2 Image Recognition model, retrained on a new personal dataset. Due to how deep and resistant to overfitting it is, as well as resistant to the disappearing gradient problem, the model is capable of transfer learning, through the use of the model pretrained on the ImageNet dataset and retraining it on the smaller dataset, to finetune it. Being a very deep and complex model, Microsoft ResNet50V2 is capable of creating very precise classification models, from which is possible to extract reliable bottleneck features, useful in the reverse search of the clothing articles. Through a second classification of these features, it is possible to return similar images to the original.

The application is tasked with the retrieval of a small collection of outfits which might contain an article of clothing resembling the one uploaded by the user. Deploying the latest techniques emerging in the image detection and classification branches, it is possible to develop a web application to mimic a fashion consultant.

Cuprins

UNIVERSITATEA DIN BUCUREȘTI	1
Absolvent Ganea Carina-Eliza	1
Cuprins	3
1 Introducere	4
1.1 Prezentarea generală	4
1.1.1 Structura lucrării	4
1.1.2 Scop și motivație	5
1.2 Tipul și domeniul de încadrare al aplicației	5
1.2.1 Contextul științific	6
1.2.2 Contribuția proprie	7
2 Preliminarii	10
2.1 Tehnologii utilizate	10
2.1.1 Tensorflow - Keras	10
2.1.2 Flask	10
2.1.3 Scikit-learn	11
2.2 Arhitectura aplicației	11
3 Dezvoltarea aplicației	13
3.1 Colectarea setului de date	13
3.1.1 Seturi de date open-source	13
3.1.2 Crearea unui set de date propriu	14
3.2 Antrenarea modelului	14
3.2.1 Data Augmentation	14
3.2.2 Arhitectura ResNet	17
3.2.3 Reverse Image Search cu Knn	20
3.3 Integrarea în aplicație web cu Flask	23
Concluzii	28
Bibliografie	35

Capitolul 1

1 Introducere

1.1 Prezentarea generală

Obiectivul aplicației este cel de a sfătui utilizatorul neinstruit în vederea creării unei vestimentații care să încorporeze un articol de îmbrăcăminte deținut de acesta. Ținutele prezentate utilizatorului de către aplicație sunt în format imagine, salvate în memoria locală de pe site-uri populare de fashion.

Utilizatorul poate încărca pe platformă o imagine care să reprezinte articolul de îmbrăcăminte dorit să fie asortat într-o ținută, fie probat de către o persoană, fie doar întins pe o suprafață plană. Aplicația va analiza articolul de îmbrăcăminte și îl va compara cu cele identificate în ținutele salvate local. Rezultatul va consta într-o colecție de ținute care conțin un articol de îmbrăcăminte similar cu cel introdus de utilizator.

1.1.1 Structura lucrării

Primul capitol al curenteii lucrări acoperă contextul și motivația dezvoltării curenteii aplicații. Astfel, introducerea conține un scurt istoric al domeniului de învățarea automată și ultimele reușite în sectorul de clasificare și identificare a imaginilor, ipoteza problemei pe care aplicația o va rezolva și contribuția proprie în acest sens.

Al doilea capitol relevă bibliotecile utilizate și motivația din spatele alegerii acestor tehnologii. În plus, tot în acest capitol există și o scurtă descriere a arhitecturii aplicației, din punct de vedere Front-End și Back-End.

Al treilea capitol acoperă ansamblul etapelor parcurse în dezvoltarea aplicației. Cu alte cuvinte, acest capitol cuprinde prezentarea seturilor de date care au fost încercate pentru antrenament și concluziile formate în urma acestei etape. Apoi, va fi prezentată pe scurt arhitectura modelului ales și modalitatea de antrenare pe colecția de date. În finalul acestui capitol, se va detalia și implementarea unei interfețe web intuitive pentru utilizator.

Ultimul capitol este dedicat concluziilor acestui proces de dezvoltare și prezentării

utilității aplicației. Tot în acest capitol sunt discutate și metode de îmbunătățire pe viitor a aplicației.

1.1.2 Scop și motivație

Ideea aplicației a apărut de la o anumită problemă foarte frecvent întâlnită în cotidian. Problema alegerii unei vestimentații, pornind de la articole deja deținute. În prezent, îmbrăcămintea este unul dintre cele mai importante elemente în crearea unei imagini publice. Ținuta este un important mijloc de comunicare non-verbală, motiv pentru care multe femei petrec mult timp încercând să își aleagă îmbrăcămintea corespunzătoare. Mai exact, în medie, o femeie își va petrece în jur de 17 minute zilnic căutând prin garderobă, în total pierzând în jur de 6 luni din perioada anilor de muncă. Pe lângă acest timp pierdut, majoritatea femeilor aflate în această dilemă ajung să aibă și accese de furie [1].

Pornind de la această problemă, se ajunge la întrebarea: „Se poate simplifica modul în care se alege o ținută?”. Este clar că, în final, vestimentația este subiectivă, deci o generalizare perfectă a problemei este imposibilă. Există însă, în prezent, noțiunea de „fashion advisor” – un profesionist în domeniul alegerilor vestimentare, menit să își ajute clienții să își construiască un stil vestimentar și să îi sfătuiască cum să își asorteze articolele de îmbrăcăminte. Dacă ne permitem să abstractizăm, un „fashion advisor” analizează ultimele ținute folosite de persoane publice iconice și ajută clientul să recreeze „acel look”. Evident, nu îl copiază perfect, ci folosesc articole asemănătoare.

În acest sens, scopul aplicației este cel de a emula un astfel de consultant. Utilizatorul – clientul din analogie – va putea să primească sfaturi în legătură cu articolele sale de îmbrăcăminte din partea aplicației, instruită pe baza unor ținute utilizate sau recomandate de persoane certificate în acest domeniu.

1.2 Tipul și domeniul de încadrare al aplicației

Prezenta lucrare are ca scop documentarea dezvoltării unei aplicații web bazate pe Machine Learning, dezvoltată în limbajul Python, prin intermediul framework-ului Flask și a bibliotecii Tensorflow.

1.2.1 Contextul științific

În ultimii ani, domeniul vederii artificiale a trecut prin multe schimbări fundamentale datorită introducerii de noi tehnologii. În directă relație cu aceste progrese, a devenit posibil ca vederea artificială să întrecă vederea umană din punctul de vedere al rezolvării eficiente a diferitelor probleme din zona de image recognition, detectare a obiectelor, identificare a fețelor, clasificării imaginilor etc.

În acest sens, introducerea rețelelor convoluționale neurale (CNN) a avut un impact demn de menționat. Aceste rețele au fost utilizate extensiv pentru analiza datelor de tip imagine cu o acuratețe remarcabilă.

Deși arhitectura acestora ne oferă alternativa de a adăuga mai multe straturi convoluționale în model pentru o mai mare putere de calcul în situații complexe din vederea artificială, acest fapt aduce la rândul său noi dificultăți. A fost sesizat că rețelele neurale devin din ce în ce mai complicat de antrenat pe măsură ce se introduc noi straturi convoluționale, chiar reducând acuratețea modelului, în anumite situații.

Considerând dificultățile antrenării rețelelor neurale adânci, s-a făcut remarcat modelul ResNet– prescurtarea pentru Rețea de Reziduuri (Residual Network), care a reușit să surmonteze această piedică, și anume, numărul mare de straturi convoluționale [2].

Acest model a fost un mare succes, întrucât dezvoltatorii săi au câștigat competiția de clasificare ILSVRC din 2015 cu o rată de eroare de doar 3.57%. Mai mult decât atât, au fost clasati pe primul loc și pentru identificare și localizare ImageNet și identificare și segmentare COCO în competițiile ILSVRC & COCO din 2015.

Experții din domeniul ML tind să adauge din ce în ce mai multe straturi în arhitectura rețelelor neurale convoluționale folosite la rezolvarea problemelor de computer vision. Aceste straturi suplimentare sunt foarte utile în soluționarea eficientă a problemelor complexe – ele pot fi antrenate pe o gamă și mai largă de cerințe și pattern-uri pentru a genera rezultate extrem de precise.

Într-o oarecare măsură numărul sporit de straturi poate îmbogăți trăsăturile modelului, însă o rețea prea adâncă ar putea începe să prezinte semne de degradare. Mai pe larg, cu cât numărul de

straturi într-o rețea neurală convoluțională crește, acuratețea modelului riscă să ajungă suprasaturată, punct de unde aceasta începe să se degradeze. Directa consecință este deteriorarea modelului atât pe setul de date de antrenare, cât și pe cel de testare.

Totuși, această deteriorare nu este cauzată de overfitting, ci fie de inițializarea rețelei, fie de funcția de optimizare, fie de dispariția sau explozia problemei de gradient.

ResNet a fost creată întocmai pentru a elimina acest impediment. Rețelele adânci de reziduuri se folosesc de blocuri de reziduuri pentru a îmbunătăți acuratețea modelelor. Fundamentul acestui tip de rețea neurală se bazează pe noțiunea de „skip connection”, conexiuni care funcționează în două moduri distincte. În primul rând, acestea ameliorează impactul dispariției gradientului prin definirea unei scurtături prin care gradientul să poată să se propage în continuare. În al doilea rând, îi conferă modelului abilitatea de a învăța o funcție de identitate prin care se poate garanta că straturile superficiale ale rețelei nu vor avea o performanță mai slabă decât straturile mai adânci [3].

Mai concis, blocurile de reziduuri ușurează drastic sarcina de a învăța funcții de identitate. Ca urmare, ResNet aduce îmbunătățiri eficienței rețelelor neurale adânci cu ajutorul straturilor neurale adiționale, totodată păstrând marja de eroare la valori minime. Cu alte cuvinte, o „skip-connection” adună output-ul straturilor din blocurile convoluționale anterioare la output-ul tuturor straturilor stivuite, astfel devenind posibilă antrenarea unor rețele mult mai adânci decât în trecut [4].

1.2.2 Contribuția proprie

Pe lângă o arhitectură complexă și eficientă, pentru ca o rețea neurală de reziduuri să atingă performanțe satisfăcătoare, este necesară o pregătire temeinică a setului de date pentru antrenare și pentru testare. În prealabil, trebuie efectuată colectarea datelor astfel încât acestea să nu prezinte vreun bias. Cu alte cuvinte, setul de date de antrenament trebuie să reflecte întocmai varietatea posibilelor date de intrare în practică, fără să indice anumite corelări incorecte între o anumită trăsătură (fundal, poziție corporală etc.) și o anumită categorie.

Arhitectura originală a ResNet50 este antrenată pe un set foarte vast de date –

ImageNet conține peste 14 milioane de imagini, aparținând a nu mai puțin de 20 000 de categorii. Pentru scopul general de a recunoaște imagini fără o anumită caracteristică cunoscută în prealabil, modelul acesta pre-antrenat este mai mult decât suficient. Eficiența acestuia are de suferit, însă, când imaginile au fost deja supuse unei filtrări. Obiectivul acestei lucrări a fost simularea unui consultant de vestimentație, cât mai ușor accesibil și cât mai intuitiv. Din această ipoteză este ușor de concluzionat că modelul final trebuie să fie capabil nu doar să identifice hainele de alte obiecte, ci să repartizeze tipurile de articole de îmbrăcăminte și să își formeze o percepție asupra articolelor care se aseamănă. Antrenarea modelului de la început, fără ponderile puse la dispoziție în Keras pe setul de ImageNet, este posibil, dar numai în cazul unui set de date comparabil ca dimensiuni, ceea ce din păcate nu este cazul. Există totuși posibilitatea ca pornind de la acele ponderi, să reantrenăm modelul și pe setul curent de date, datorită caracterului rezidual al modelului. Această metodă diminuează timpul și costul de antrenare, păstrând în același timp și cunoștințele existente, asupra cărora se adaugă noile trăsături învățate.

În continuare, urmează procesul de analizare și colectare a datelor de antrenament. Pe parcursul dezvoltării am testat un set de date de articole de îmbrăcăminte disponibil pe Kaggle, care consta din aproximativ 4 000 de imagini cu articole de îmbrăcăminte de 20 de tipuri. Din păcate, acest set de date conținea doar articole de îmbrăcăminte întinse pe o suprafață plană, făcând asocierea cu forma articolelor de pe manechin foarte inconsistentă, cu mari confuzii. De aceea, am elaborat de la zero un nou set de date care să fie conform cerințelor aplicației în curs de dezvoltare. Reorganizând formele orientative ale articolelor de îmbrăcăminte, am păstrat 18 categorii de articole, însumând un total de 900 de mostre.

Comparat cu setul ImageNet pe care a fost antrenat originalul model ResNet, setul acesta clar nu este suficient de extensiv. Pentru ca acuratețea să nu aibă mult de suferit, am adăugat și tehnici de augmentare a imaginilor. Dintr-un punct obiectiv de vedere, multe articole de îmbrăcăminte sunt foarte asemănătoare, dar pot să difere și prin forma pe care o iau în diferite poziționări ale corpului uman. Prin data augmentation,

putem lărgi artificial setul de date în faza de antrenare a modelului, chiar dacă imaginile nu conțin un număr considerabil de articole de sine-stătătoare.

După ce modelul a fost antrenat să fie mai sensibil la trăsăturile articolelor, ultima fază este testarea acestuia pe ținute întregi, nu doar un articol singular. Această problemă am generalizat-o la un fel de identificare de imagini similare. Cu alte cuvinte, pe baza cunoștințelor generate de model la antrenament, pe de-o parte acesta are capacitatea de a distinge între articole, dar, totodată, și de a identifica care articole se aseamănă și, totodată, în ce măsură. În esență, fiecare imagine are o anumită probabilitate de a aparține uneia dintre clasele de articole în funcție de caracteristicile sale dominante și definitorii. Din acest punct de vedere se poate defini și o distanță calculabilă între două imagini care să exprime cât de asemănătoare sunt.

În momentul în care se poate distinge o regulă pentru distanța între două obiecte dorite a fi clasificate, modelul de clasificare K-Nearest Neighbours devine o posibilă soluție. KNN nu este un algoritm de clasificare prea eficient în general, dar în cazul curent, acesta ar utiliza și predicțiile furnizate de către modelul ResNet de clasificare.

Această metodă de a găsi imagini asemănătoare este folosită în practică și în Reverse-Image Search pentru motoarele de căutare care acceptă și imagini. Este utilă deoarece graficul generat prin KNN poate fi precalculat, ceea ce rezultă într-un timp foarte scurt de răspuns la interogări.

Capitolul 2

2 Preliminarii

2.1 Tehnologii utilizate

2.1.1 Tensorflow - Keras

TensorFlow este o librărie open-source, utilă pentru calcule numerice rapide. A fost creată și este în continuare întreținută de către Google și publicată în conformitate cu licența de tip open source Apache 2.0. API-ul este în esență dedicat limbajul de programare Python, însă conține și o adaptare ca API pentru limbajul C++.

Diferit de alte biblioteci numerice dedicate utilizării în Deep Learning precum Theano, de exemplu, TensorFlow a fost creat pentru a acoperi atât cerințele din cercetare, dezvoltare și sisteme de producție, cât și proiectele RankBrain în Google search și DeepDream.

Din biblioteca TensorFlow, am utilizat exclusiv funcționalitățile din Keras, utile pentru ML, mai exact Preprocessing pentru manipularea imaginilor, generarea seturilor de antrenare și validare și pentru Data Augmentation.

2.1.2 Flask

Flask este un framework web de tip Python, de dimensiuni reduse, ușor de rulat, dotat cu diverse facilități și tool-uri pentru crearea aplicațiilor web în Python. Acesta oferă dezvoltatorilor foarte multă flexibilitate, fiind un framework accesibil și programatorilor cu mai puțină experiență, deoarece este posibilă construirea aplicației pornind de la un singur fișier Python. De asemenea, Flask este și extensibil, nu forțează o anumită structură pentru proiect și nu necesită o arhitectură standard complicată de adăugat înainte de a porni dezvoltarea aplicației.

Alternativa acestui framework ar fi fost Django, pentru a rămâne într-un mediu Python, care are mai multă libertate decât Flask în partea de design front-end. Acesta însă este mai riguros

din punctul de vedere al arhitecturii, fiind un framework de tip MVC, ceea ce ar adăuga mai multe fișiere de cod pentru partea de interacțiune utilizator-aplicație. Acest tip de arhitectura este de preferat în aplicațiile care conțin mai multe pagini web, fiecare cu comportament diferit [5]. În cazul aplicației curente, este necesară o singură pagină pentru a ilustra funcționalitatea platformei. Din acest motiv, am preferat să lucrez în Flask decât în Django, chiar dacă Flask este compatibil doar cu pagini HTML statice, fără componente de Ajax sau jQuery, spre deosebire de Django.

2.1.3 Scikit-learn

Scikit-learn este o bibliotecă open-source de învățare automată pentru limbajul Python și conține algoritmi utili pentru clasificare, regresie și clustering, ca de exemplu mașini de vectori-suport (SVM), random forests, gradient boosting, k-means și DBSCAN. Este proiectată să funcționeze alături de bibliotecile de calcul numeric în Python NumPy și SciPy.

Din această bibliotecă, am utilizat algoritmi Nearest Neighbours și PCA pentru clasificare și clustering, în a doua parte aplicației, pentru căutarea ținutelor similare și scăderea timpului de răspuns.

2.2 Arhitectura aplicației

Aplicația poate fi împărțită în 3 secțiuni: antrenarea modelului de rețea neurală adâncă – ResNet50V2, crearea sistemului de repere pentru definirea similarității dintre două articole de îmbrăcăminte pentru funcționalitatea de Reverse Image Search și platforma pentru interacțiune aplicație-utilizator – front-end.

Antrenarea modelului reprezintă, în mare parte, componenta bazată pe Machine Learning a aplicației. Aceasta este deja rulată și încheiată la momentul rulării aplicației. Procesul de învățare este foarte lung, dar nu trebuie efectuat decât o dată, pe datele de antrenament, care sunt invariabile. Modelul trebuie antrenat pe imagini care conțin doar câte un articol de îmbrăcăminte care poate fi repartizat pe categoriile de bază, iar datele create prin antrenare

pot fi stocate și reîncărcate în aplicație la rulare.

Sistemul de repere creat pentru starea curentă a bazei de date a ținutelor este supus schimbărilor în măsura în care ținutele se pot schimba după tendințe sau prin adăugarea de noi ținute. Totuși, aceste schimbări nu sunt dependente de utilizator, ceea ce înseamnă că această parte de redefinire a trăsăturilor ținutelor poate fi rulată în prealabil pe server, nu local de către utilizator. La fel ca ponderile rețelei neurale, acest sistem de repere poate fi creat și stocat pentru a fi încărcat în starea sa funcțională.

Ultima componentă reprezintă interfața aplicației, unde utilizatorul poate adăuga date (imagini) care trebuie analizate (cu ajutorul rețelei neurale gata-antrenate) și inserate în sistemul de repere pentru a găsi rezultatele cele mai asemănătoare din colecția de vestimentații.

Considerând că, în cazul curent, aplicația rulează doar local și nu necesită autentificare, nu am creat o bază de date de sine-stătătoare pentru stocarea imaginilor. În cazul extinderii acestei platforme, ar fi necesară portarea ei pe un server cu mai multă putere de calcul pentru a efectua interogările asupra unei colecții mai mari de ținute. Ideal, aplicația ar căuta singură ținutele din surse de încredere, astfel încât aceasta să fie mereu la curent cu ultimele tendințe.

Capitolul 3

3 Dezvoltarea aplicației

3.1 Colectarea setului de date

3.1.1 Seturi de date open-source

Pentru rezultate satisfăcătoare, modelul trebuie antrenat pe un set de date cât mai adecvat domeniului problemei. Cele mai populare seturi de imagini cu articole de îmbrăcăminte sunt Deep Fashion [6], Fashion-MNIST [7] și, mai mic relativ la celelalte, Clothing Dataset [8].

Primul set, Deep Fashion, este de dimensiuni foarte mari – 700 000 de imagini –, cu numeroase adnotații, inclusiv bounding-boxes pentru articolul de îmbrăcăminte prezent într-o anumită imagine, și clasificări în funcție de tipul de cusături ale articolelor. Este un set foarte extensiv, pe baza căruia un model ar putea atinge o mare acuratețe la clasificare, chiar și dacă ar fi antrenat fără ponderi prealabile. Totuși, fiind atât de voluminos, este destul de greu de manevrat pe spații de stocare mai restrânse, iar datele sunt excesiv de etichetate și minuțioase pentru scopul aplicației pe care o voi dezvolta. Din aceste motive am ales să evit acest set de date.

Fashion-MNIST ar fi fost o alternativă mai light-weight, însumând 70 000 de imagini din 10 clase. Din păcate, asemenea setului MNIST clasic, acest set este monocromatic. Prin definiție, aplicația necesită un set de date color pentru o mai mare sensibilitate și a culorilor, nu doar a formelor.

Din aceste motive, am folosit inițial un set de date disponibil pe Kaggle – Clothing Dataset – consistând din 5 000 de imagini cu articole de îmbrăcăminte din 20 de categorii. Deși este cu mult mai restrâns în eșantioane față de celelalte două, modelul a atins o acuratețe la antrenare de 50%, fără data augmentation. După mai multe antrenări, am realizat că setul de date are totuși o problemă semnificativă – hainele ilustrate nu erau pozate îmbrăcate, ceea ce le distorsiona forma într-o oarecare măsură. Considerând că modelul ar trebui să fie capabil să distingă articolele de îmbrăcăminte de pe un manechin, forma întinsă a articolelor crea un bias la

antrenare non-conform cu datele reale, care urmau să fie întâlnite în practică. Pe lângă formă, apăreau și inconsistențe în distingerea culorii din cauza pozelor efectuate în medii de luminozitate diferită, inclusiv temperaturi luminoase diferite. Pentru un model antrenat doar pentru identificarea formelor, setul ar fi fost satisfăcător, încât astfel de diferențe luminoase ajută la scăderea sensibilității față de culoare, dar acest efect nu este dorit în aplicația curentă.

3.1.2 Crearea unui set de date propriu

În lipsa unei alternative, am întocmit un set de date propriu cu articole de îmbrăcăminte, de dimensiuni reduse. Orientativ, se disting 18 clase de articole distincte: „Blazer”, „Body”, „Blouse”, „Dress”, „Hat”, „Hoodie”, „Longsleeve”, „Outwear”, „Pants”, „Polo”, „Pullover”, „Shirt”, „Shoes”, „Shorts”, „Skirt”, „T-Shirt”, „Top”, „Undershirt”. Pentru o bună antrenare a modelului, am ales ca fiecare clasă să aibă cel puțin câte 50 de mostre, iar clasele mai prevalente în vestimentații, precum „Outwear”, „Pants”, „Longsleeve” să aibă în jur de 150 de imagini.

În plus, imaginile trebuie să conțină articolele pozate în același tip de lumină, cât mai neutră, pentru ca modelul să poată corela și nuanțele articolelor. Hainele trebuie să fie pozate exact cum arată îmbrăcate de o persoană, pentru a ușura apoi procesul de identificare al articolelor când acestea sunt folosite într-o ținută, dar și într-o formă neîmbrăcată, nu foarte distorsionată. Astfel, modelul poate să coreleze forma articolelor îmbrăcate cu forma neîmbrăcată, în cazul în care utilizatorul inserează o poză cu un articol de îmbrăcăminte neprobat.

3.2 Antrenarea modelului

3.2.1 Data Augmentation

Pentru a compensa cât de restrâns este setul de date și faptul că imaginile cu fiecare articol ilustrează o singură poziție a corpului, putem utiliza și biblioteca de ImageGenerator din Keras pentru a crea imagini augmentate. Astfel, chiar dacă setul de date creat și utilizat este de dimensiuni reduse, creștem rezistența la overfitting în timpul antrenării modelului [9,10].



Fig. 3.1 Original – Articolul de îmbrăcăminte este fotografiat în prim plan, dar doar dintr-o anumită poziție corporală și acoperind o mare suprafață din imagine.

În cazul aplicației curente, prin data augmentation încercăm să emulăm diferitele poziții ale corpului uman în imagini care conțin ținutele de analizat. Mai exact, putem avea ținute pozate mai de la distanță, necentrate în cadru sau cu poziție înclinată. Dar cunoaștem că ținutele sunt mereu în întregime în cadru, au orientarea verticală, culoarea curenta este mereu definitorie și nu ar trebui să fie diferențe mari de iluminare.

Ajustând parametrii generatorului în acest fel, putem genera noi imagini pornind de la original, care să ofere informații suplimentare modelului în timpul antrenării.



Fig 3.2 Imagini augmentate - În ținute, articolul original ar putea apărea minimizat, oglindit, „stretched”, rotit sau deplasat față de centrul imaginii.

Cu ajutorul modulului Image din Keras.Processing, putem utiliza ImageDataGenerator pentru a încărca setul de date și a îi face în timp real atât partiționarea în set de antrenare și validare, cât și aplicarea parametrilor pentru Data Augmentation.

```
datagen = ImageDataGenerator(
featurewise_center=True,
featurewise_std_normalization=True,
samplewise_std_normalization=False,
rotation_range=30,
width_shift_range=0.1, height_shift_range=0.1,
shear_range=25.0,
zoom_range=[1.0, 2.0],
channel_shift_range=0.,
fill_mode='nearest',
horizontal_flip=True,
```



```
rescale=1.0 / 255.0,  
validation_split=0.15)
```

Cu ajutorul clasei `ImageDataGenerator`, se efectuează foarte simplu procesul de augmentare, setând corespunzător parametrii relevanți. `Featurewise_center` și `featurewise_std_normalization` asigură normalizarea vectorului de trăsături ale imaginilor – adică transformarea valorilor originale în valori subunitare – și centrarea acestora relativ la 0, cu referire la întregul set de date. În contrapartidă, parametrii `samplewise_center` și `samplewise_std_normalization` efectuează aceleași transformări, dar la nivel de eșantion, distrugând efectiv relațiile între trăsăturile globale ale setului de date. Din acest motiv, acești parametri nu sunt utilizați. `Rotation_range` este parametru care definește numărul maxim de grade sexagesimale a rotației la care poate fi supusă imaginea. `Width_shift_range` și `height_shift_range` este un procent care semnifică magnitudinea maximă a translatărilor pe lățime și înălțime care pot avea loc. `Shear_range` semnifică o transformare asemănătoare rotației, însă aceasta se efectuează fixând una dintre axele imaginii, creând un efect de „stretch”. `Zoom_range` semnifică apropierea sau depărtarea imaginii. În cazul curent, `zoom_range` este în intervalul 1 și 2, adică imaginea poate fi doar depărtată de observator până devine de maxim două ori mai mică. `Channel_shift_range` și `brightness_range` nu au fost utilizați deoarece aceștia afectează culoarea obiectului, respectiv, luminozitatea acestuia, pentru a scădea sensibilitatea modelului relativ la culori. `Horizontal_flip` oglindește imaginea față de axul vertical al imaginii, efect dorit în cazul acesta, iar `vertical_flip` față de axul orizontal, efect nedorit. `Rescale` aduce valorile pixelilor în intervale subunitare, iar `validation_split` este procentul de eșantioane păstrate înafara setului de antrenament, pentru validare [11].

3.2.2 Arhitectura ResNet

Modelele premiate la competițiile ILSVRC și COCO au fost ResNet50, ResNet101 și ResNet152, dar, înafară de ResNet50 pe care am ales să îl folosesc, acestea necesită mult timp și spațiu pentru antrenare pe un calculator clasic cu GPU integrat. În plus, versiunile 101 și 150 sunt foarte adânci, fiind dedicate setului de date ImageNet. Nu este nevoie de a le testa pentru a ajunge

la concluzia că ar ajunge la overfitting pe un set de date de 700 de ori mai restrâns și cu de 1000 de ori mai puține clase de etichetare.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

3.3 Arhitecturile ResNet – Se observă împărțirea pe câte 5 blocuri reziduale, iar arhitecturile mai adânci conțin straturi de bottleneck la începutul și finalul fiecărui bloc de convoluție, pentru a minimiza numărul parametrilor.

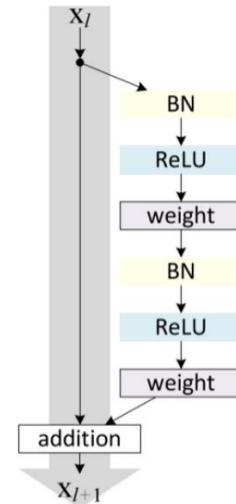
Adâncimea unui model ResNet diferă în numărul de straturi aflate în blocul de reziduuri, dar numărul de blocuri este constant 5, cu tot cu stratul pentru input. Pentru rețelele mai adânci, se observă că blocurile reziduale încep și se termină cu 2 straturi cu convoluții de tip 1×1, numite bottlenecks. Acestea forțează minimizarea numărului de parametrii pentru a avea înmulțiri matriceale cu dimensiuni mai mici de la bloc la bloc. Arhitecturile mai puțin adânci nu necesită astfel de minimizări forțate.

În plus, apare noțiunea de „skip-connection”, nemaîntâlnită până la ResNet în rândul rețelelor neurale. Fiecare strat adăugat în model aparține unei ierarhii non-liniare. La fiecare adăugare a unui strat convoluțional (ponderat - „weighted”), se creează două ramuri de calcul: $F(x)$ – care semnifică aplicarea convoluției din segmentul de la ultima „skip-connection” până la stratul de adiție – și x – funcția identitate, care „sare” peste blocul rezidual curent. Rezultatul acestei secvențe bloc este $F(x) + x$, cu alte cuvinte, o nouă valoare a ponderilor care încorporează ponderea anterioară asemănător unui reziduu.

Pentru setul de date creat, am utilizat arhitectura mai puțin adâncă de ResNet50V2, care poate fi ușor importată din librăria Keras. În acest model, ierarhia straturilor unui bloc de reziduuri este de tipul:

1. Batch Normalization
2. Activation(ReLU)
3. Strat convolutional (Conv2D)
4. Batch Normalization
5. Activation(ReLU)
6. Strat convolutional (Conv2D)
7. Addition (Add)

Fig. 3.4 Skip-connection – Între stratul x_l și x_{l+1} există atât o legătură de identitate (directă, fără transformări), cât și o legătură transformativă (cu Batch Normalization -> Activation -> Convolution2D -> Batch Normalization -> Activation -> Convolution2D) care sunt cumulate pentru a crea efectul de „reziduu” de la trecerea de pe un strat pe altul.



Există mai multe variații asupra componentelor unui astfel de bloc, în funcție de ordinea straturilor de BatchNormalization și Activation. În practică, s-a observat că păstrarea a câte unui strat de BatchNormalization și Activation la începutul blocului, ajută la propagarea gradientului indiferent dacă s-ar fi efectuat un salt (skip connection), detaliu care îi conferă acestui tip de arhitectura o mai mare rezistență la overfitting prin păstrarea acestor conexiuni „curate”[12].

Pentru a prelua corect modelul de ResNet50V2 din Keras, trebuie îndepărtat stratul de output pentru ca arhitectura să se potrivească cu setul de date. În locul acestuia, putem folosi modelul Sequential din Keras pentru a incorpora noul strat de output pentru ResNet50V2. În plus, parametrul „weights” trebuie inițializat cu „imagenet”. Considerând că imaginile cu articole de îmbrăcăminte se aseamănă setului ImageNet, putem utiliza aceste ponderi suplimentare pentru o antrenare mai rapidă. În practică, aceasta tehnică oferă rezultate de acuratețe foarte bune, mai ales când se marchează ca cele mai adânci straturi să nu mai fie antrenate. Am ales ca doar ultimele 10 straturi să mai fie supuse antrenării, deoarece, pentru arhitectura ResNet50V2, setul curent de

date este foarte restrâns și ar ajunge să se degradeze ponderile. În plus, acuratețea la validare a atins 85% în 30 de epoci, un rezultat suficient pentru obiectivul aplicației.

```
model = ResNet50V2(include_top=False, weights='imagenet', pooling='avg', input_shape=(256, 256, 3))
model1 = Sequential()
model1.add(model)
for layer in model.layers[:len(model.layers) - 9]:
    layer.trainable = False
model.summary()
model1.add(layers.Dense(18, activation='softmax'))
model = model1
model.compile(loss='categorical_crossentropy',
              optimizer=Adam(learning_rate=1e-3),
              metrics=['accuracy'])
```

Pentru a nu relua antrenarea modelului la fiecare pornire a aplicației, am adăugat parametrul „callbacks” în apelul metodei fit(). Prin clasa ModelCheckpoint() putem defini directorul și fișierul în care se va salva arhitectura și ponderile de la finalul unei epoci în care s-a atins un nou maxim de acuratețe pe setul de date de validare.

3.2.3 Reverse Image Search cu Knn

După ce modelul a fost antrenat pe articolele de îmbrăcăminte din setul de antrenare, acesta este capabil să ofere predicții asupra ținutelor din baza de date.

Următorul pas constă în procesul de „Feature Extraction”, o metodă prin care se pot reprezenta imagini într-o formă redusă pentru a ușura probleme de decizie asemenea pattern detection, clasificării sau recunoașterii. Identificarea și extragerea unor trăsături de încredere și discriminante este mereu o etapă importantă de acoperit în procesele de image recognition și computer vision [13].

Pornind de la modelul ResNet antrenat în prealabil, încărcăm modelul antrenat, eliminând stratul cel mai superficial, care este necesar la atribuirea etichetelor pentru clasificare. După îndepărtarea „top”-ului, imediat dedesubt, se află stratul de tip „bottleneck” – ultimul strat din mașină unde se efectuează activarea neurală înainte de cumularea caracteristicilor pentru a oferi o predicție.

După încărcarea modelului, extragem cate un „feature-list” pentru fiecare ținută din baza de date, rezultat din predicția modelului.

```
def extractFeatures(pathToImage, model):  
    input_shape = (256, 256, 3)  
    img = image.load_img(pathToImage, target_size=(input_shape[0], input_shape[1]))  
    imageArray = image.img_to_array(img)  
    expandedArray = np.expand_dims(imageArray, axis=0)  
    preprocessedImage = preprocess_input(expandedArray)  
    features = model.predict(preprocessedImage)  
    flatFeatures = features.flatten()  
    normFeatures = flatFeatures / norm(flatFeatures)  
    return normFeatures [14]
```

Conform arhitecturii ResNet50, „feature list”-ul rezultat conține 2048 de trăsături, pe care le putem reprezenta într-un grafic 2048-dimensional.

Algoritmul Knn, deși este considerat o metoda de Machine Learning, nu efectuează o „învățare” propriu-zisă. Acesta creează spații multidimensionale în care sunt plasate obiectele de clasificat pe axe în funcție de valorile vectorilor de „features”. Din reprezentarea în spațiu, se pot interoga cei mai apropiați „k” vecini după una dintre distanțele: euclidiană, Manhattan, Minkowski, cosine, Hamming, Jaccard, dintre care cea mai uzuală este distanța euclidiană.



Fig. 3.5 Reverse Image Search - După rularea algoritmului Knn pentru toți vectorii de „features” și interogarea a NearestNeighbours cu parametru $k = 5$ pentru o imagine, putem observa că ținutele rezultate conțin cămăși de culori și forme asemănătoare.

Rezultatele furnizate de KNN sunt în mare parte viabile, mai ales considerând că numărul de ținute disponibile este destul de mic pentru a furniza 5 rezultate cu distanțe destul de apropiate (algoritmul rulează pe aproape 200 de ținute cât mai variate – considerând câte trăsături pot fi identificate la nivel de articol, este un număr mic, care încă nu afișează completa eficacitate a programului). Totuși, și pe un asemenea set de date restrâns, se conturează un oarecare timp de așteptare pentru răspunsul funcției k-neighbours. Acest lucru este cauzat de numărul de trăsături de analizat, iar pe măsură ce setul de date se îmbogățește, acest timp de așteptare crește polinomial față de numărul de eșantioane.

Pentru a îmbunătăți performanța algoritmului KNN, se poate utiliza PCA (Principal

Component Analysis) prin care cele 2048 de caracteristici furnizate prin extracția cu ajutorul ResNet sunt analizate și filtrate, astfel încât doar cele mai discriminatorii trăsături ale ținutelor din setul de Outfits să fie trecute prin KNN. Modul în care PCA reușește să facă abstracție de trăsături redundante fără să scadă din corectitudinea clasificării constă în introducerea domeniului de trăsături într-o reprezentare în spațiu, găsirea direcțiilor importante în reprezentarea setului de date și importanța fiecăreia pentru setul de date, alegerea direcțiilor cel mai importante în acest sens și alinierea forțată a eșantioanelor la direcția cea mai apropiată, fără a pierde din eșantioane și trăsături relevante [15].

```
features = extractFeatures(img_path, model)
featureList = []
for i in notebook.tqdm(range(len(filenames))):
    featureList.append(extractFeatures(filenames[i], model))
featureDimensions = 100
pca = PCA(n_components=featureDimensions)
pca.fit(featureList)
featureListCompressed = pca.transform(featureList)
neighbors = NearestNeighbors(n_neighbors=5,
                             metric='euclidean',
                             algorithm='brute').fit(featureListCompressed)

compressedFeatures = pca.transform(features.reshape(1, -1))
distances, indices = neighbors.kneighbors(compressedFeatures) [14]
```

3.3 Integrarea în aplicație web cu Flask

După antrenarea modelului ResNet50V2 pe setul de articole de îmbrăcăminte, crearea și salvarea modelului KNN antrenat pe trăsăturile compresate de PCA, mai rămâne doar crearea

aplicației web cu Flask.

Este de ajuns un fisier python pentru codul aplicației care va configura portul și adresa la care va rula aplicația, folderul „static” din care aplicația își va prelua modelele și datele utilizate pentru afișare și folderul „uploads” în care se vor salva imaginile încărcate de către utilizator. Cele două modele care urmează a fi utilizate sunt ușor de manevrat cu librăria Pickle și modulele Sequential și Model din Keras. În total, există două funcționalități de bază.

Prima funcționalitate este reprezentată de încărcarea unui fișier de tip imagine și afișarea acesteia în timp real, pentru ca utilizatorul să aibă o experiență plăcută la interacțiunea cu interfața și pentru a putea verifica mai ușor imaginea încărcată. Partea aceasta a fost implementată cu ajutorul JavaScript, fără reîmprospătarea paginii.

A doua funcționalitate constă în analiza articolului de îmbrăcăminte cu ajutorul modelului ResNet50V2 antrenat – citit cu ajutorul Pickle din directorul „static” al aplicației, inserarea în modelul KNN a vectorului de trăsături astfel rezultat din predicția imaginii utilizatorului și apoi compresat și reîmprospătarea paginii, trimițând parametrii noi prin intermediul Flask: ținutele „prezise” – care ar conține un articol de îmbrăcăminte asemenea celui adăugat de utilizator, și imaginea care a fost procesată și care trebuie afișată din nou în template-ul de „home.html”. În cazul în care pagina „Home” a fost invocată după oferirea unei predicții, ea va fi încărcată deja derulată automat în zona ținutelor rezultate, pentru confortul utilizatorului.

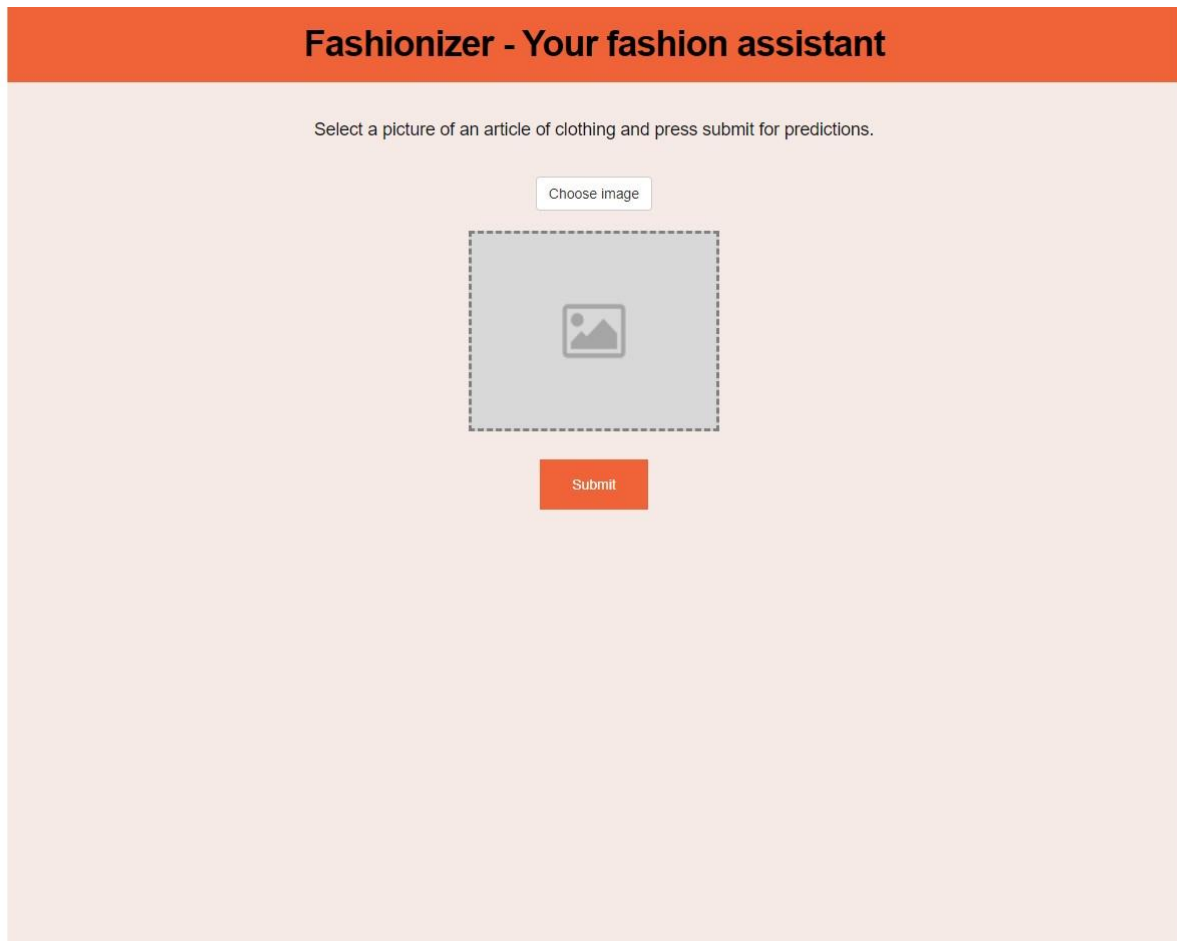


Fig. 3.6 Prima accesare a aplicației – La prima accesare a site-ului, este vizibil doar placeholder-ul pentru imagine.

Fashionizer - Your fashion assistant

Select a picture of an article of clothing and press submit for predictions.

Choose image



Submit

Fig. 3.7 Încărcarea imaginii – Apăsând pe butonul „Choose image”, utilizatorului i se deschide o fereastră cu documentele din care poate alege imaginea. O dată aleasă imaginea, aceasta este afișată în locul placeholder-ului.

Some outfits that might include this:



Fig. 3.8 Rezultatul aplicației – După apăsarea butonului de submit, pagina se reîmprospătează și se derulează pentru a se observa mai întâi rezultatul interogării.

Capitolul 4

Concluzii

Rezultatele finale ale aplicației ilustrează comportamentul dorit, așa cum reiese din cerința inițială pentru aplicație. Prin rularea căutărilor pentru mai multe articole de îmbrăcăminte, rezultatele sunt evident foarte asemănătoare cu articolul căutat.

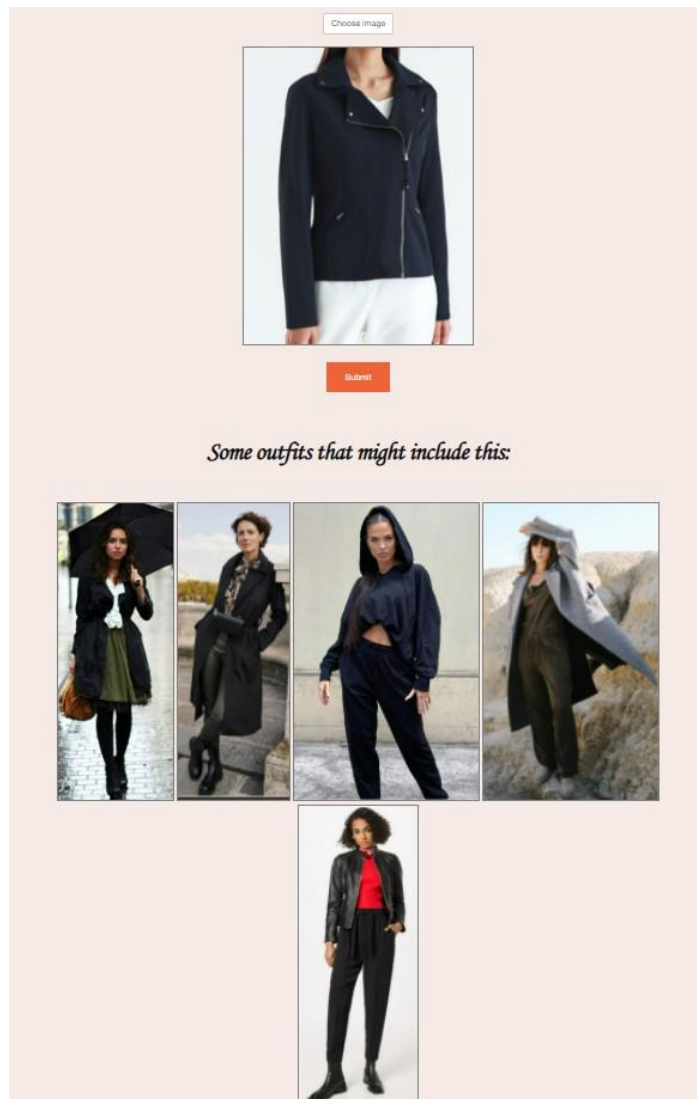


Fig. 4.1 Rezultatul căutării unei geci scurte negre – Se observă identificarea corectă a formei gulerului, culorii. Lungimea gecii este considerată mai orientativă, dar este un rezultat corect din punctul de vedere al unui utilizator neinstruit.

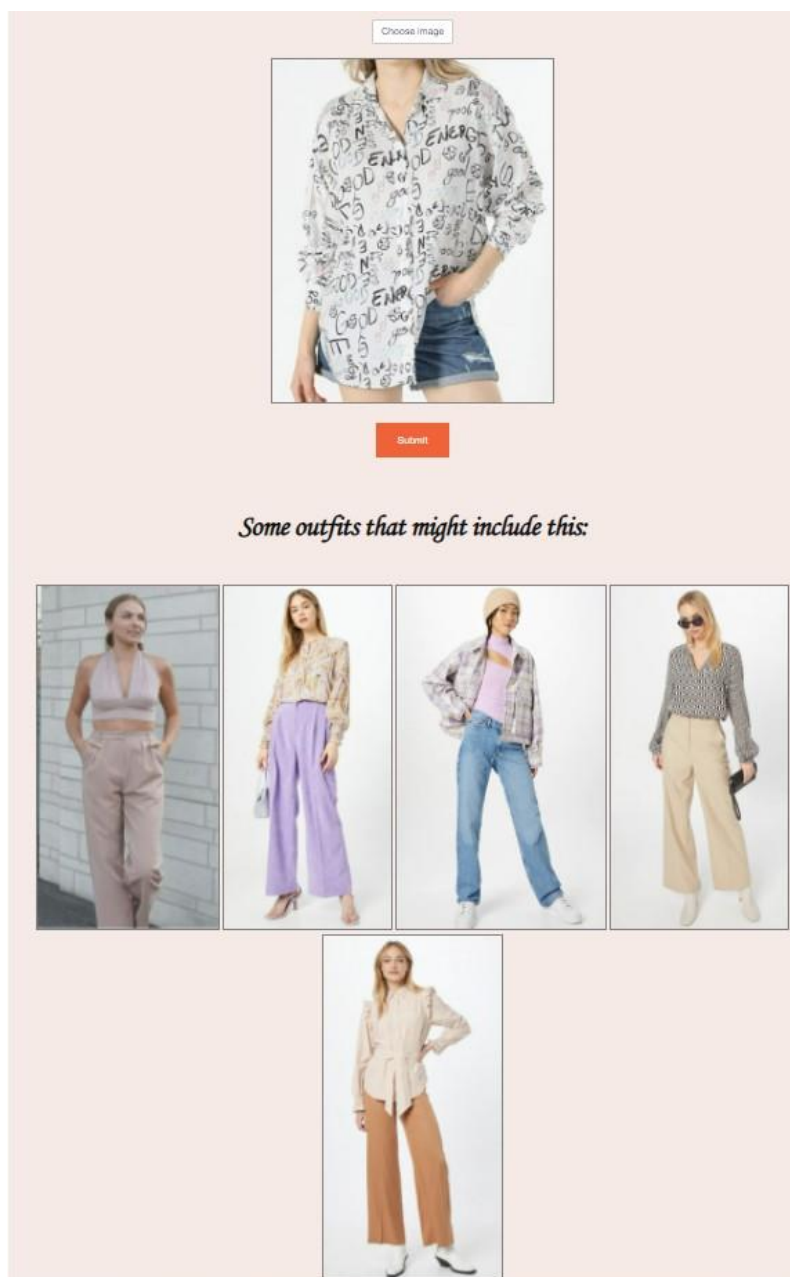


Fig. 4.2 Un rezultat corect – Se observă identificarea corectă a formei mânecii, a culorii de

ansamblu. Articolele de îmbrăcăminte returnate seamănă din mai multe puncte de vedere cu originalul, deci aplicația se comportă corespunzător.

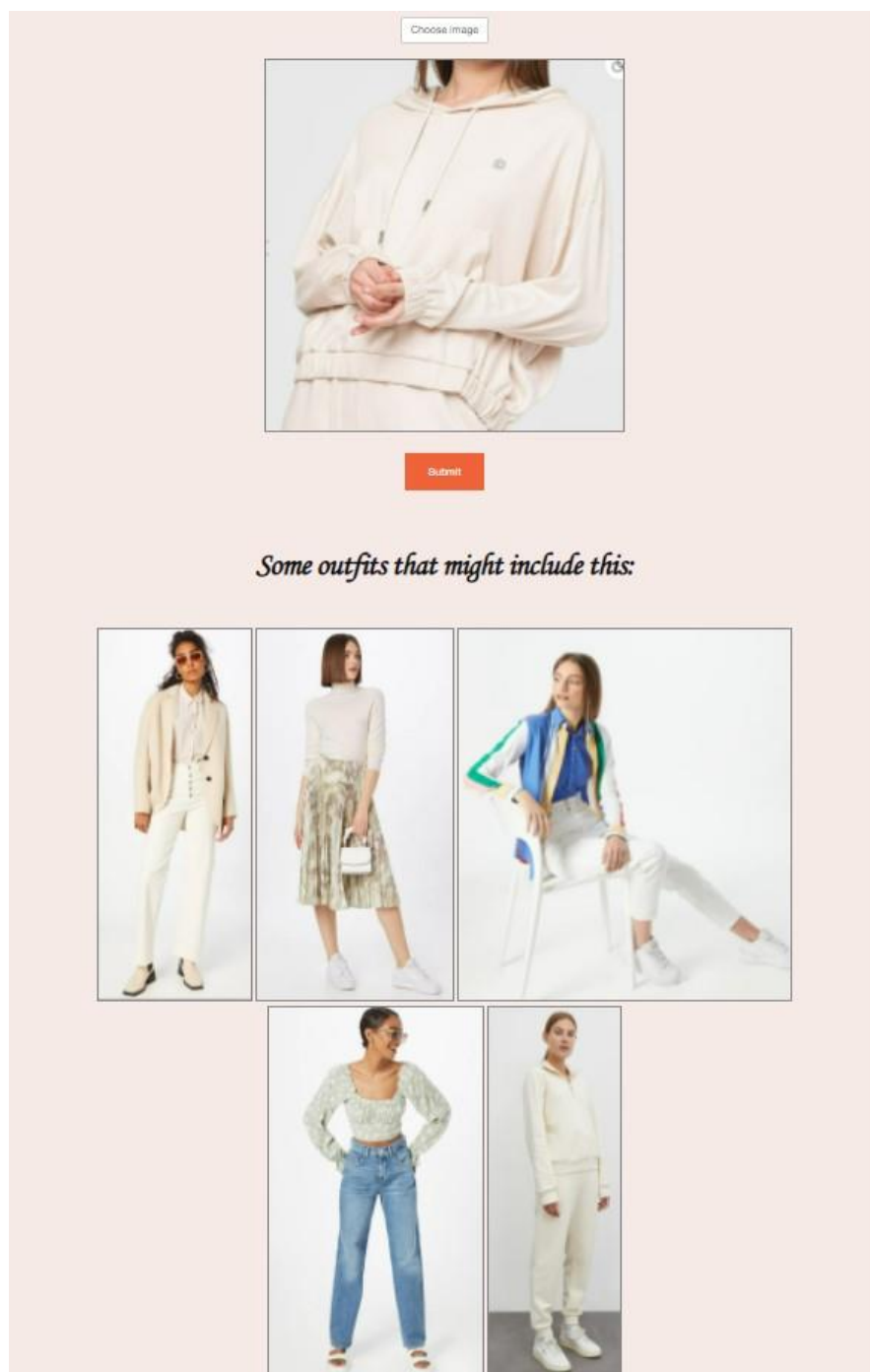


Fig. 4.3 Rezultat corect – Se distinge culoarea comună a majorității bluzelor din imagine, forma

seamănă per ansamblu cu articolul căutat.

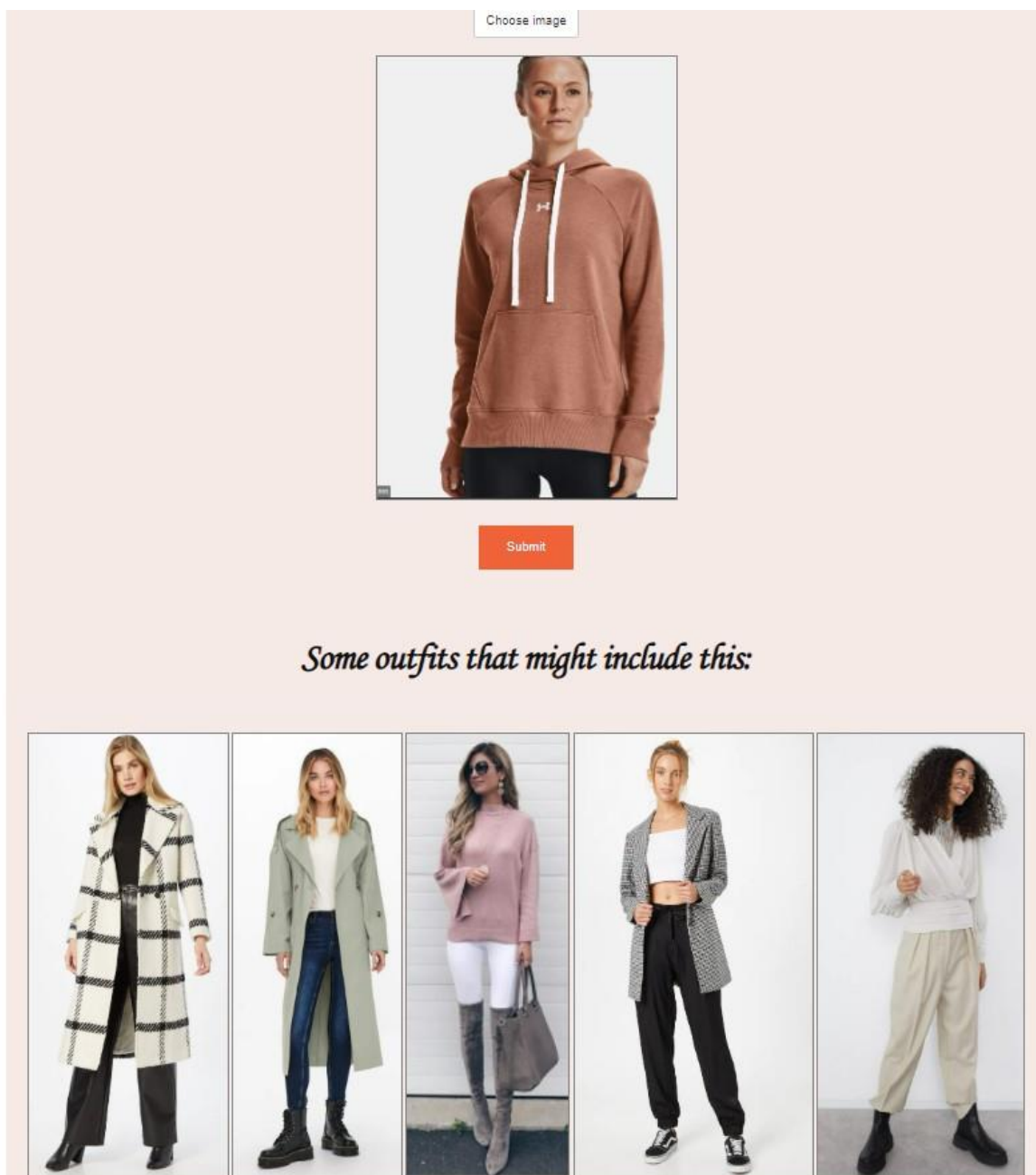



Fig. 4.4 Un rezultat cu asemănări mai forțate – Articolele de îmbrăcăminte au culori care nu seamănă cu cel căutat, cu excepția pullover-ului roz, care este un rezultat foarte asemănător. Este


normal, deoarece nu există alte ținute care să conțină un articol chiar atât de asemănător cu originalul.


Choose image





Submit

Some outfits that might include this:












Fig. 4.5 Un rezultat pentru o căutare de pantaloni șalvari – 3 ținute conțin perechi de pantaloni care seamănă în ceea ce privește forma, deci modelul a putut găsi ceva asemănător, chiar dacă nu există ținute cu pantaloni de aceeași culoare. Este de așteptat ca aplicația să țină cont mai mult de forma articolului.

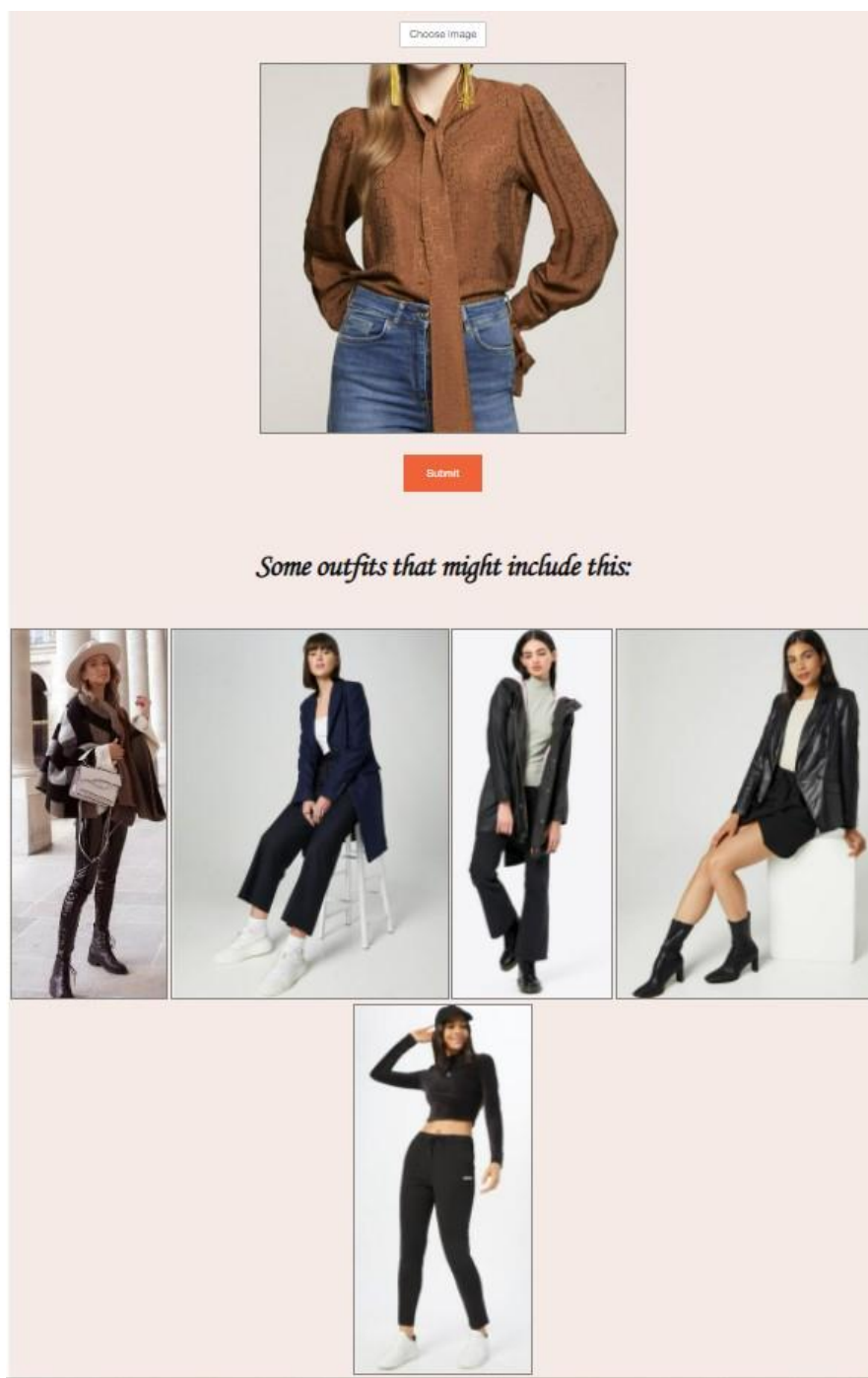


Fig. 4.6 Un rezultat care conține asemănări forțate față de articolul inițial – Ținutele nu conțin articole care să semene la prima vedere nici ca formă, nici culoare. Totuși, imaginile conțin ținute cu care s-ar potrivi și articolul căutat, deci este un rezultat decent pentru colecția curentă de ținute.

Din rezultatele de mai sus, putem concluziona că modelul antrenat reușește să extragă trăsături foarte precise relativ la cum se asortează articolele între ele. Predicțiile expuse sunt ținute care pot incorpora fără dificultăți articolul căutat.

Aceste rezultate pot fi îmbunătățite și de o gamă mai extinsă de ținute din care se poate alege. În momentul în care articolul adăugat se distanțează mult de forma orientativă a articolelor din ținute, modelul trebuie totuși să întoarcă 5 ținute, cele mai apropiate, chiar dacă distanțele sunt mari raportat la alte rezultate.

Problema poate fi rezolvată prin adăugarea mai multor ținute diferite, dar și atunci există posibilitatea ca mai puțin de 5 chiar să conțină ceva asemănător cu articolul căutat, caz în care rezultatul nu ar fi în întregime corect. Totuși, probabilitatea ca acest lucru să aibă loc scade cu cât setul de date s-ar extinde mai mult.

Acest detaliu ar putea fi integrat prin selectarea automată a noilor ținute de adăugat în colecție. Câțiva furnizori de ținute permit și efectuarea de request-uri către bazele lor de date pentru preluarea imaginilor. Dar ar fi necesară o manevră optimă a datelor primite din punct de vedere al timpului și spațiului dacă acest proces se efectuează pe un calculator personal.

Există și fenomenul de „tendință”, prin care un articol de îmbrăcăminte ar apărea în mai mult de 5 ținute, dar doar primele 5 cel mai apropiate ar fi returnate. Aceasta este o caracteristică a algoritmului KNN care încorsetează utilitatea aplicației.

Anumite aplicații care se bazează pe reverse image search (Instagram, Pinterest etc.) au implementat un algoritm asemănător KNN-ului, care să returneze mai mulți vecini, chiar dacă aceștia ar fi doar „aproximativi”. Acest algoritm se numește ANNOY (Aproximate Nearest Neighbours Oh Yeah), dezvoltat în cadrul organizației Spotify, și rezolvă în plus și eventuala întârziere a interogărilor, cauzate de o evidență prea mare a obiectelor în care se efectuează căutarea de tip k-neighbours [16].

Bibliografie

- [8] Alexey Grigorev, „Clothing Dataset (full, high resolution)”, <https://www.kaggle.com/datasets/agrigorev/clothing-dataset-full>, ultima accesare: 18/05/2022
- [3] Claudio F. Gonçalves dos Santos, João P. Papa, „Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks”, în *ACM Computing Surveys*, 4.3 (Jan. 2022), pp. 12, ISSN: 0360-0300, DOI: 10.1145/3510413, URL: <https://dl.acm.org/doi/10.1145/3510413>
- [10] Connor Shorten, Taghi M. Khoshgoftaar, „A survey on Image Data Augmentation for Deep Learning”, în *Journal of Big Data*, 6, 60 (Jul. 2019), DOI: 10.1186/s40537-019-0197-0 URL: <https://doi.org/10.1186/s40537-019-0197-0>
- [16] Erik Bernhardsson, „Annoy (Approximate Nearest Neighbors Oh Yeah)”, <https://github.com/spotify/annoy>, ultima accesare: 18/05/2022
- [7] „Fashion-MNIST”, <https://www.kaggle.com/datasets/zalando-research/fashionmnist>, ultima accesare: 18/05/2022
- [4] Gaudenz Boesch, „Deep Residual Networks (ResNet, ResNet50) – Guide in 2022”, <https://viso.ai/deep-learning/resnet-residual-neural-network/>, ultima accesare: 18/05/2022
- [5] Gaurav Sharma, „Which is Better for Machine Learning: Flask vs Django?”, <https://www.analyticsvidhya.com/blog/2022/01/which-is-better-for-machine-learning-flask-vs-django/>, ultima accesare: 18/05/2022
- [13] Heba A. Elnemr, Nourhan M. Zayed, Mahmoud A. Fakhreldein, „Feature Extraction Techniques: Fundamental Concepts and Survey”, în *Handbook of Research on Emerging Perspectives in Intelligent Pattern Recognition, Analysis, and Image Processing*, 13 (2016), pp. 264-294, ISBN13: 9781466686540, DOI: 10.4018/978-1-4666-8654-0, URL: <https://doi.org/10.4018/978-1-4666-8654-0>
- [9] Jason Wang, Luis Perez, „The Effectiveness of Data Augmentation in Image Classification using Deep Learning”, (Dec. 2017), DOI: 10.48550/arXiv.1712.04621, URL: <https://doi.org/10.48550/arXiv.1712.04621>

- [2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, „Deep Residual Learning for Image Recognition”, în Microsoft Research, (Dec. 2015), DOI: 10.48550/arXiv.1512.03385 URL: <https://doi.org/10.48550/arXiv.1512.03385>
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, „Identity Mappings in Deep Residual Networks”, în Microsoft Research, (Jul. 2016), DOI: 10.48550/arXiv.1603.05027, URL: <https://doi.org/10.48550/arXiv.1603.05027>
- [14] Koul Anirudh, Ganju Siddha, Kasam Meher, „Practical Deep Learning for Cloud, Mobile, and Edge: Real-World AI & Computer-Vision Projects Using Python, Keras & TensorFlow”, în O'Reilly Media, (2019), pp. 144-174, ISBN13: 9781492034865, URL: https://archive.org/details/python_ebooks_2020/Practical_Deep_Learning_for_Cloud%2C_Mobile%2C_and_Edge_Real_World_AI
- [6] Liu, Ziwei and Luo, Ping and Qiu, Shi and Wang, Xiaogang and Tang, Xiaoou, „DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations”, (Jun. 2016), <https://mmlab.ie.cuhk.edu.hk/projects/DeepFashion.html>, ultima accesare: 18/05/2022
- [15] Matt Brems, „A One-Stop Shop for Principal Component Analysis”, <https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c>, ultima accesare: 18/05/2022
- [1] Martha Cliff, „Women spend SIX MONTHS of their working lives deciding what to wear - and suffer 'wardrobe rage' over trying to choose the right outfit”, <https://www.dailymail.co.uk/femail/article-3626069/Women-spend-SIX-MONTHS-working-lives-deciding-wear.html>, ultima accesare: 19/05/2022
- [11] Sumit Sarin, „Exploring Data Augmentation with Keras and TensorFlow”, <https://towardsdatascience.com/exploring-image-data-augmentation-with-keras-and-tensorflow-a8162d89b844>, ultima accesare: 19/05/2022