Metode de vedere artificială şi învăţare automată pentru clasificarea obiectelor în industria porţelanului

Autori: Copîndean Alin Gabriel, Golgoț Mihai Radu
Coordonator: Conf. univ. dr. Bîrluțiu Adriana
Instituția: Universitatea „1 Decembrie 1918" din Alba Iulia, Facultatea de Științe Exacte și Inginerești, Specializarea: Informatică.

Abstract: *This paper presents an application for classifying the shape of objects manufactured at a porcelain factory. The application was created to help the companies from the porcelain industry in the production process. In the first phase, we investigated an approach that makes use of several computer vision techniques. With the help of some functions from the OpenCV library, we analyzed the contours of the objects and classify them according to their size. In the second phase, we investigated a machine learning algorithm, artificial neural networks, for solving this problem. We used a data set to train the artificial neural network and evaluate how accurately the artificial neural network learned to classify images.*

1. Introducere

Tema acestui proiect este crearea unei aplicații care să recunoască automat forma unei farfurii dintr-o imagine, dar și tipul farfuriei. Pentru aceasta e nevoie să se identifice zona din imagine unde se află farfuria pentru a putea recunoaște apoi forma și a putea ulterior să o încadrăm într-o anumită categorie.

Această aplicație poate fi utilă companiilor din industria porțelanului deoarece poate să înlocuiască operatorii umani, astfel contribuind la reducerea costurilor salariale [5]. De asemenea, această aplicație ajută și la reducerea timpului de producție reușind într-un timp scurt să clasifice un număr mare de imagini cu pozele ale farfuriilor.

2. Tehnologii folosite

Pentru realizarea acestei aplicații am folosit limbajul de programare Python, și bibliotecile specializate pentru vederea artificială și învățare automată existente în acest limbaj.

Python este un limbaj de programare dinamic, având o anumită simplitate a codului, facilitând astfel scrierea de cod spre deosebire de alte limbaje de programare [1].

Biblioteca *openCV* [4] conține algoritmi și tehnici pentru prelucrarea imaginilor. Cu ajutorul acestei biblioteci putem să citim imagini din fișier, putem salva imagini, și putem realiza diferite prelucrări ale imaginilor cum ar fi: aplicarea diferitelor filtre, redimensionări, translatări, rotiri, etc.

Biblioteca *glob* este utilizată pentru a găsi calea spre fișierele din directorul specificat. Folosind funcțiile acestei biblioteci putem aplica mai multe filtre în funcție de formatul fișierului.

Biblioteca *numpy* este folosită pentru programarea științifică în Python. Această bibliotecă conține tool-uri pentru integrarea limbajului de programare C/C++ în Python, algebră liniară, transformări Fourier și generarea numerelor aleatoare.

Biblioteca *TensorFlow* [6] conține mai mulți algoritmi predefiniți pentru învățarea automată bazată pe rețele neuronale adânci.

3. Descrierea setului de date folosit

Universitatea noastră colaborează cu o companie din industria porțelanului care ne-a pus la dispoziție setul de date conținând imagini ale unor farfurii, imaginile fiind făcute efectiv în procesul de producție. În total, setul de date conține 141 de imagini, care sunt împărțite în 3 categorii: farfurii rotunde (Figura 1), farfurii dreptunghiulare (Figura 2) și farfurii pătrate (Figura 3).

Fiecare imagine are o rezoluție de 3840x2748 de pixeli, ocupând un spațiu de memorie de aproximativ 4,5Mb.

Figura 1 – Imagine reprezentând o farfurie rotundă.

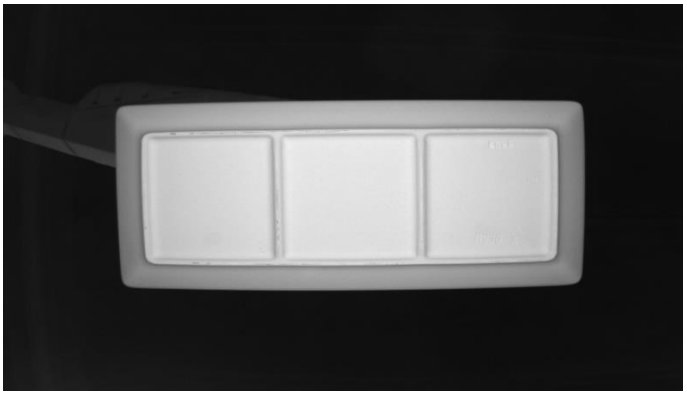

Figura 2 – Imagine reprezentând o farfurie dreptunghiulară.

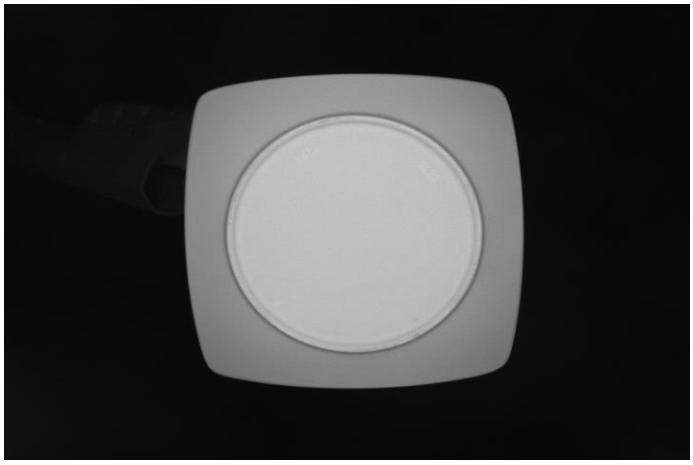

Figura 1 - Imagine reprezentând o farfurie pătrată.

4. Abordarea axată pe tehnici de vedere artificială

Pentru a identifica conturul farfuriilor am folosit mai multe funcții din biblioteca *openCV*.

Într-o primă fază, am citit imaginile cu ajutorul funcției *cv2.imread*. Apoi am delimitat conturul farfuriilor și am aflat dimensiunea lor cu ajutorul funcțiilor *cv2.cvtColor*, *cv2.GaussianBlur*, *cv2.threshold*, *cv2.findContours*. După ce am reușit să aflăm conturul și dimensiunea farfuriei avem o instrucțiune care încadrează dimensiunea farfuriei într-una din cele trei clase de farfurii, sau nu o încadrează în niciuna dintre ele dacă mărimea nu corespunde cu niciun interval. Figura 4 prezintă secvența de cod care execută acești pași.

```
for filename in glob.glob('Test_Final/*.jpg'): #assuming jpeg
    img=cv2.imread(filename)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5, 5), 0)
    (t, binary) = cv2.threshold(blur, t, 255, cv2.THRESH_BINARY)
    (_, contours, _) = cv2.findContours(binary, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    for (i, c) in enumerate(contours):
        #print("\tSize of contour %d: %d" % (i, len(c)))
        if len(c)>1000:
            if len(c)>=3000:
                print("Farfuria din imaginea %s are forma rotunda"% (filename))
                #contor=contor+1
               # print(contor)
            elif len(c)<=1450 and len(c)>=min_drept-13:
                print("Farfuria din imaginea %s are forma dreptunghiulara"% (filename))
                #contor=contor+1
                #print(contor)
            elif len(c)<=2999 and len(c)>=1451:
                print("Farfuria din imaginea %s are forma hexagonala"% (filename))
                #contor=contor+1
                #print(contor)
            else:
                print("Farfuria din imaginea %s NU are nici o forma dintre cele invatate."% (filename))
                #contor=contor+1
                #print(contor)
```

Figura 2 - Codul Python pentru metoda axată pe tehnici de vedere artificială.

La executarea algoritmului acesta returnează imaginea cu farfuria iar conturul acesteia este marcat (Figura 5 și Figura 6). De asemenea, ne este returnată și clasa din care face parte farfuria din imaginea respectivă (Figura 7).

Figura 3 – Detectarea conturului unei farfurii rotunde.

Figura 4 – Detectarea conturului unei farfurii dreptunghiulare.

```
Farfuria din imaginea Test_Final\cerc1 (43).jpg are forma rotunda
Farfuria din imaginea Test_Final\cerc1 (44).jpg are forma rotunda
Farfuria din imaginea Test_Final\cerc1 (45).jpg are forma rotunda
Farfuria din imaginea Test_Final\cerc1 (46).jpg are forma rotunda
Farfuria din imaginea Test_Final\cerc1 (47).jpg are forma rotunda
Farfuria din imaginea Test_Final\cerc1 (48).jpg are forma rotunda
Farfuria din imaginea Test_Final\cerc1 (49).jpg are forma rotunda
Farfuria din imaginea Test_Final\dreptunghi (21).jpg are forma dreptunghiulara
Farfuria din imaginea Test_Final\dreptunghi (22).jpg are forma dreptunghiulara
Farfuria din imaginea Test_Final\dreptunghi (24).jpg are forma dreptunghiulara
Farfuria din imaginea Test_Final\dreptunghi (25).jpg are forma dreptunghiulara
Farfuria din imaginea Test_Final\dreptunghi (26).jpg are forma dreptunghiulara
Farfuria din imaginea Test_Final\dreptunghi (27).jpg are forma dreptunghiulara
Farfuria din imaginea Test_Final\dreptunghi (28).jpg are forma dreptunghiulara
Farfuria din imaginea Test_Final\dreptunghi (29).jpg are forma dreptunghiulara
Farfuria din imaginea Test_Final\dreptunghi (30).jpg are forma dreptunghiulara
Farfuria din imaginea Test_Final\dreptunghi (32).jpg are forma dreptunghiulara
Farfuria din imaginea Test_Final\dreptunghi (33).jpg are forma dreptunghiulara
```

Figura 5 – Clasificarea farfuriilor.

5. Abordarea axată pe tehnici de învățare automată

Pentru citirea imaginilor am folosit funcția *cv2.imread*. Pentru a identifica forma farfuriilor mai întâi avem nevoie de un set de date de antrenare și un set de date de testare [1]. Setul de date te antrenare este format din 63 de imagini (câte 21 pentru fiecare formă). În primă fază se citesc imaginile pe care le redimensionăm la 120x120 pixeli, sub forma unei matrici, folosind funcția *cv2.resize* din pachetul *OpenCV*. Fiecare valoare din interiorul matricei reprezintă nivelul de gri pentru fiecare pixel.

Pentru crearea modelului de clasificare am folosit funcția *tflearn.dnn()* din biblioteca TensorFlow (modulul *tflearn*) care construiește o rețea neuronală artificială cu mai multe straturi [6]. Pentru învățarea modelului am aplicat funcția *model.fit()* unde numărul de epoci ales a fost de 40 ceea ce înseamnă că pentru crearea rețelei neuronale datele noastre au fost procesate de 40 de ori. Figura 8 de mai jos redă secvența de cod Python pe care am descris-o mai sus.

```
import tensorflow as tf
tf.reset_default_graph()

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name = 'input')
convnet = conv_2d(convnet, 32, 5, activation = 'relu')
convnet = max_pool_2d(convnet, 5)
convnet = conv_2d(convnet, 64, 5, activation = 'relu')
convnet = max_pool_2d(convnet, 5)
convnet = conv_2d(convnet, 32, 5, activation = 'relu')
convnet = max_pool_2d(convnet, 5)
convnet = conv_2d(convnet, 64, 5, activation = 'relu')
convnet = max_pool_2d(convnet, 5)
convnet = conv_2d(convnet, 32, 5, activation = 'relu')
convnet = max_pool_2d(convnet, 5)
convnet = conv_2d(convnet, 64, 5, activation = 'relu')
convnet = max_pool_2d(convnet, 5)
convnet = fully_connected(convnet, 1024, activation = 'relu')
convnet = dropout(convnet, 0.8)
convnet = fully_connected(convnet, 3, activation = 'softmax')
convnet = regression(convnet, optimizer = 'adam', learning_rate = LR, loss = 'categorical_crossentropy
model = tflearn.DNN(convnet, tensorboard_verbose=3)

train = train_data[-50:] #Train Set
test = train_data[:-50] #Validation Set

X = np.array([i[0] for i in train]).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
Y = [i[1] for i in train]

test_x = np.array([i[0] for i in test]).reshape(-1, IMG_SIZE, IMG_SIZE, 1)
test_y = [i[1] for i in test]

model.fit({'input': X}, {'targets': Y}, n_epoch=40, validation_set=({'input': test_x}, {'targets': tes
```

Figura 6 – Codul Python pentru metoda axată pe învățarea automată.

Rezultatul este afișat cu ajutorul funcției *model.predict()* sub forma unei matrici cu 3 coloane, fiecare coloană reprezentând tipurile de farfurii, iar valorile de pe linii conțin probabilitățile de a face parte din fiecare dintre cele 3 categorii (Figura 9).

```
[0.03621624 0.9324939  0.03128983]
[0.01007884 0.01735226 0.97256887]
[0.03202585 0.9389889  0.02898526]
[0.91955215 0.02842229 0.05202558]
[0.03876398 0.9252014  0.03603461]
[0.01833073 0.02071527 0.96095407]
[0.03203131 0.9388454  0.0291232 ]
[0.0126402  0.01846875 0.968891  ]
[0.01097048 0.01706885 0.9719607 ]
[0.9311663  0.02716203 0.04167164]
```

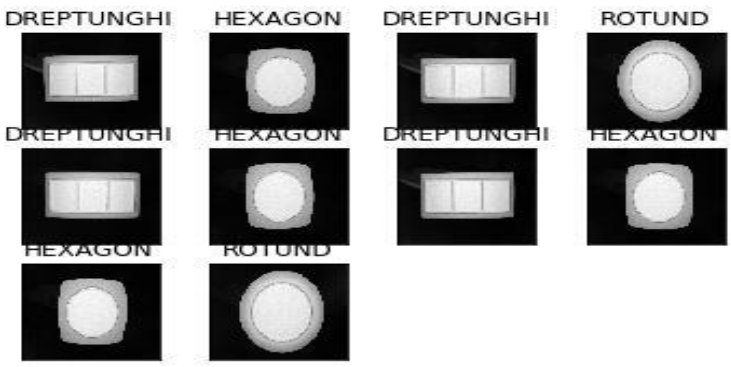

Figura 7 - Clasificarea farfuriilor prin metode de învăţare automată.

6. Concluzii și direcții viitoare de extindere a aplicației

Această aplicație este utilă pentru companiile din industria porțelanului și poate fi folosită pentru identificarea și clasificarea farfuriilor. Prin folosirea acestei aplicații, companiile își pot reduce timpul de producție și costurile salariale.

Ca şi direcţii de extindere viitoare, ne propunem să îmbunătățim această aplicație prin introducerea mai multor categorii de obiecte. Ne propunem să investigăm și alte tehnici de vedere artificială și alți algoritmi de învățare automată pentru această problemă.

Bibliografie

[1] Brownlee, J. (2017). Deep Learning With Python.

[2] Gareth James, D. W. (2013). An Introduction to Statistical Learning with.

[3] Kamber, J. H. (2006). Data mining-Concepts and Techniques.

[4] Mordvintsev, A. (2017). OpenCV-Python Tutorials Documentation.

[5] Onița Daniela, G. I. (2017). Detecția defectelor în industria porțelanului folosind rețele neuronale adânci. Sesiunea de comunicări a studenților.

[6] www.tensorflow.org. (n.d.).