

Documentație – Object classification

Balea Raluca, sg1

Introducere

Rețelele neuronale artificiale simulează modul de interacțiune neuronală a creierului uman înspre învățare.

Acest proiect prezintă realizarea unui model de rețea neuronală care este capabil să clasifice imagini cu o rezoluție foarte mică și să le încadreze într-o anumită categorie.

Pentru ca acest lucru să fie posibil am utilizat un set de date care conține aproximativ 40000 de imagini cu diferite obiecte pe care le voi enunța mai jos. Însă pentru a salva timp și resurse am utilizat doar primele 20000 de imagini.

Pentru realizarea proiectului am folosit limbajul python și mediul de dezvoltare Visual Studio Code.

Librăriile utilizate:

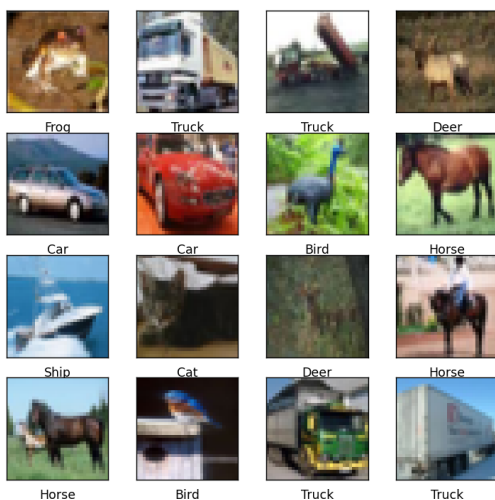
tensorflow- Este o bibliotecă open source care ajută la dezvoltarea de rețele neuronale să fie mai rapidă

numpy- Adaugă suport pentru matrici de dimensiuni mari și multidimensionale pentru a putea opera pe aceste matrici

matplotlib- Ajută la crearea de chart-uri

Keras- Ajută la testare

Clasificarea imaginilor din setul de date:



Etapele parcurse în realizarea proiectului

1. Preluarea și pregătirea datelor

Setul de date este deja în keras, ceea ce este necesar de făcut este să stocăm datele în tuple de antrenare și testare. Pentru a lucra mai ușor cu imaginile le vom scala, astfel că dividem la 255.

```
(training_images, training_labels), (testing_images, testing_labels) = datasets.cifar10.load_data()
training_images, testing_images = training_images / 255, testing_images / 255
```

2. Asignarea numelor pentru labels (sunt în ordinea în care apar în labels)

```
class_names = ['Plane', 'Car', 'Bird', 'Cat', 'Deer', 'Dog', 'Frog', 'Horse', 'Ship', 'Truck']
```

3. Afișarea datasetului folosit

```
for i in range(16):
    plt.subplot(4,4,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.imshow(training_images[i], cmap=plt.cm.binary)
    plt.xlabel(class_names[training_labels[i][0]])

plt.show()
```

Practic trece prin toată matricea și transmite numerele din dataset ca index listei cu nume, astfel că de exemplu pentru 3 este asignat cuvântul “Cat”.

4. Construirea rețelei neuronale și antrenarea acesteia

Primul pas al acestei etape este declararea modelului drept model *Sequential()*, deoarece această rețea neuronală are mai multe straturi.

Adăugăm modelului straturile de convoluție cu activare ReLU, max pooling pentru a reduce dimensiunea spațială.

Convolutional layer- filtrează particularități din imagini

MaxPooling- reduce imaginea la informațiile esențiale

Ultimul strat conține metodele Flatten și Dense și softmax.

Flatten – Transformă o matrice bidimensională într-un vector

Softmax- scalează rezultatul

Dense - adaugă stratul complet conectat la rețeaua neuronală

```
model = models.Sequential()  
model.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(32,32,3))) //definirea rețelei neuronale  
model.add(layers.MaxPooling2D((2,2)))  
model.add(layers.Conv2D(64, (3,3), activation='relu'))  
model.add(layers.MaxPooling2D((2,2)))  
model.add(layers.Conv2D(64, (3,3), activation='relu'))  
model.add(layers.Flatten())  
model.add(layers.Dense(64, activation="relu"))  
model.add(layers.Dense(10, activation="softmax"))  
  
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])  
  
model.fit(training_images, training_labels, epochs = 10, validation_data=(testing_images, testing_labels))
```

5.Evaluare, testare, salvare

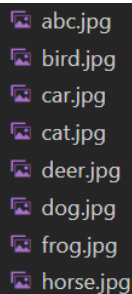
Pentru a nu fi nevoie sa le antrenam de fiecare data cand rulăm scriptul utilizăm secvența de cod
„model.save('image_classifier.model')”

```
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])  
  
model.fit(training_images, training_labels, epochs = 10, validation_data=(testing_images, testing_labels))  
  
loss, accuracy = model.evaluate(testing_images, testing_labels)  
print(f"Loss: {loss}")  
print(f"Accuracy: {accuracy}")  
  
model.save('image_classifier.model')
```

6.Incarcarea modelului pe care l-am antrenat

```
model = models.load_model('image_classifier.model')
```

Am descărcat apoi imagini si le-am schimbat rezolutia, apoi le-am încărcat în Visual Studio Code



- abc.jpg
- bird.jpg
- car.jpg
- cat.jpg
- deer.jpg
- dog.jpg
- frog.jpg
- horse.jpg

7. Incarcarea si vizualizarea imaginilor

```
img = cv.imread('abc.jpg')
img = cv.cvtColor(img, cv.COLOR_BGR2RGB)

plt.imshow(img, cmap=plt.cm.binary)
```

8. Predictie si afişare

```
prediction = model.predict(np.array([img]) / 255)
index = np.argmax(prediction)
print(f'Prediction is {class_names[index]}')

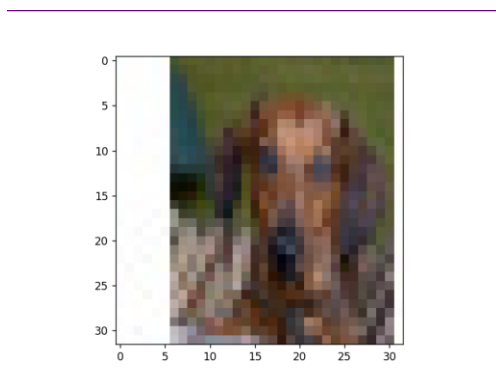
plt.show()
```

Demonstrare funcționalitate

Oferim ca input o imagine denumită intuitiv din cele încărcate

```
img = cv.imread('dog.jpg')
```

După rulare imaginea se afișează, iar în terminal ne spune care este predicția potrivită pentru imagine



```
2023-01-20 00:57:31.308725: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations: AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
1/1 [=====] - 0s 98ms/step
Prediction is Dog
```

Prediction is Dog

Bibliografie

1. <https://classroom.google.com/u/0/c/NTUxNTY1MTk1Nzkw/m/NTgxNTM2MjQxNzU4/details>
2. <https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html>
3. https://en.wikipedia.org/wiki/Softmax_function
4. <https://pyimagesearch.com/2020/10/05/object-detection-bounding-box-regression-with-keras-tensorflow-and-deep-learning/>
5. <https://stackabuse.com/image-recognition-in-python-with-tensorflow-and-keras/>
6. <https://densenetworks.com/ai-machine-learning>