Flower recognition în Python

-jupyter-

student: Bonyhai Alexandru

prof. Vicaș Cristian Marian

Introducere:

În acest proiect s-a realizat o aplicație în Python, utilizând jupyter, prin care sunt identificate diferite specii de flori din diverse familii. În prezent, aplicația identifică doar:

* Crini și familiile lor *(lilly)*
* Lotus și familia sa *(lotus)*
* Orhidee și familiile lor *(orchid)*
* Floarea-soarelui *(sunflower)*
* Lalele și familiile lor *(tulip)*

Pentru o dezvoltare ulterioară se pot include mai multe flori (și familiile lor).

Modelul realizat este pe Inteligență Artificială și a trebuit antrenat. Pot exista erori în recunoașterea unor flori dacă se află modele diferite în aceeași imagine.

Realizare:

Pentru început am descărcat imaginile grupate în 5 foldere (categorii corespunzătoare fiecărei flori). Aceste 5 categorii, le-am grupat într-un folder mai mare, numit *test\_set*.

Folderul *test\_set* l-am duplicat, iar copia am redenumit-o *training\_set*. În training\_set am șters 25% dintre imagini de pe fiecare categorie (de exemplu: din 1000 de imagini cu lalele, am șters 250).

Am creat un folder de input, acolo unde se pun imaginile care se vor a fi identificate, numit *prediction*. De asemenea am creat și fișierul *flower\_recognition.ipynb* – fișier care implementează logica și funcționalitatea programului.

*flower\_recognition.ipynb*:

Imports:

* **import numpy as np**: Numpy este o bibliotecă în Python care oferă suport pentru matrice și matrici mari, multidimensionale, împreună cu o colecție mare de funcții matematice de nivel înalt pentru a opera cu aceste matrici.
* **import tensorflow as tf**: TensorFlow este o platformă open-source de învățare automată dezvoltată de Google. Este folosită pentru a construi și antrena modele de învățare automată.
* **from tensorflow.keras.preprocessing.image import ImageDataGenerator**: ImageDataGenerator este o clasă din TensorFlow care permite preprocesarea imaginilor și încărcarea lor în memorie în timpul antrenării modelului. Acest lucru este util atunci când avem un set mare de imagini și nu dorim să încărcăm toate imaginile în memorie simultan.
* **from keras.preprocessing import image**: Modulul image din biblioteca Keras oferă o serie de funcții pentru prelucrarea imaginilor. Acesta include funcții pentru încărcarea imaginilor, convertirea imaginilor în diferite formate (de exemplu, convertirea unei imagini într-un array numpy), aplicarea de transformări pe imagini (de exemplu, rotire, scalare), etc.

Aceste biblioteci sunt adesea utilizate pentru a construi și antrena modele de învățare automată. Fiecare import este necesar pentru diferite părți ale procesului de construire a modelului.

În *training\_set* și *test\_set* s-au efectuat niște modificări:

* rescalări
* normalizări
* transformări tăiere și zoom
* redimensionări
* încărcări pe loturi
* tipizări

În variabila cnn (Convolutional neural network) s-a făcut un build (**cnn = tf.keras.models.Sequential()**) și au fost adăugate minim 2 straturi de convoluție pentru a se putea efectua o identificare corectă.

Observație: la construcția primului strat de convoluție poate exista un warning, fiindcă pot exista la versiuni superioare alte funcții sau alt mod de realizare a acelorași funcții pentru cnn.

cnn.add(tf.keras.layers.Conv2D(filters = 64,

kernel\_size = 3,

activation = 'relu',

input\_shape = [64, 64, 3]))

Funcția input\_shape nu este recomandată pe versiunile mai noi de Python (3.11+), dar în acest proiect funcționează fără probleme.

S-a folosit ca pooling un Maxpooling pentru a diferenția mai bine nuanțele de culori.

Fiecare imagine a fost convertită cu Flatten în vector în cnn, și s-au adăugat 2 straturi de neuroni:

1. un strat cu 128 neuroni, cu funcția *relu* pentru non-liniaritate
2. un strat cu 5 neuroni, cu funcția *softmax* în straturile de ieșire ale problemelor de clasificare multi-clasă și returnează probabilități pentru fiecare clasă.

S-a compilat și s-a antrenat modelul prin testare cu 2 serii de *epochs* (o serie de 100 și una de 10), unde acuratețea a ajuns stabilă la 97-98%, de la instabilitatea 40-60%.

În etapa de preprocesare a imaginilor din input (folderul *prediction*), s-au efectuat operații la fel ca cele anterioare, diferența este că numai pe o singură imagine – cea selectată/introdusă din path-ul oferit.

Rezultatul evaluării și recunoașterii este afișat sub formă de matrice, convertită prin if-elif la rezultat final în cuvinte. În matricea 2D cu un singur rând, avem 5 indici: 4 indici 0 și un indice 1 – indicele 1 reprezentând numărul categoriei găsite, aferente rezultatului. Pozițiile din linia matricei sunt:

1. Crin
2. Lotus
3. Orhidee
4. Floarea soarelui
5. Lalea

Fiecare poziție are un indice de găsire: 0 sau 1.

Fiecare poziție este tradusă în rezultat final în cuvinte.

Surse:

[Image Recognition Project | Building Flower Recognition System using Python Tensorflow and Keras - YouTube](https://www.youtube.com/watch?v=2gMZC-4WAQ4&t=4663s)

[Keras 3 API documentation](https://keras.io/api/)

[CS 230 - Convolutional Neural Networks Cheatsheet (stanford.edu)](https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks)