Readme Tema 1 AM

Bondoc Ion-Tudor

333AA

1. Modelul generativ Stable Diffusion-KerasCV

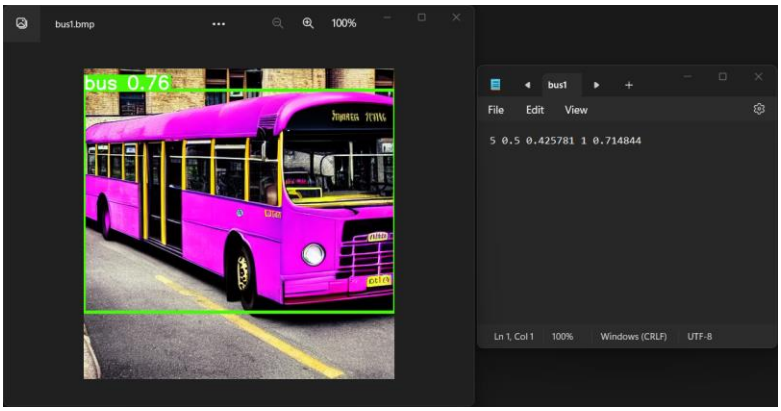
Dupa instalarea keras-cv, am declarat un model, conform keras.io/guides. Acest model primeste 2 parametri care reprezinta dimensiunea imaginilor generate (width si height). Am ales sa generez imagini de 512 x 512 pixeli. Ulterior, modelul este transformat in imagini, prin model.text_to_image, cu 2 parametri: propozitia(prompt-ul) de generare, si dimensiunea batch-ului (cate poze vrem sa generam), in cazul meu: “photograph of a purple bus”, batch_size=3. La final, salvez imaginile uncompressed (.bmp), accesand fiecare element din array-ul generat: Image.fromarray(images[0]).save("bus1.bmp").

```
Downloading data from https://github.com/openai/CLIP/blob/main/clip/bpe_simple_vocab_16e6.txt.gz?raw=true
1356917/1356917 [==============================] - 0s 0us/step
Downloading data from https://huggingface.co/fchollet/stable-diffusion/resolve/main/kcv_encoder.h5
492466864/492466864 [==============================] - 65s 0us/step
Downloading data from https://huggingface.co/fchollet/stable-diffusion/resolve/main/kcv_diffusion_model.h5
3439090152/3439090152 [==============================] - 431s 0us/step
50/50 [==============================] - 2305s 37s/step
Downloading data from https://huggingface.co/fchollet/stable-diffusion/resolve/main/kcv_decoder.h5
198180272/198180272 [==============================] - 25s 0us/step
```

Prelucrarea s-a realizat pe procesor si a durat aproximativ la 30 de minute.

2. Reteaua de detectie de obiecte yolov5

Am utilizat reteaua yolov5 direct in linia de comanda dupa ce am clonat repository-ul disponibil pe github. Folosind fisierul detect.py, am specificat imaginea sursa, optiunea de save-txt precum si numele fisierului text de la iesire, ce contine informatiile despre bounding box-ul gasit in format Darknet(yolov5): (clasa_obiect, X, Y, width, height – normalizate de la 0 la 1). Nu am specificat ce versiune sa se utilizeze, astfel ca s-a ales in mod automat yolov5s. Rezultatele se gasesc in folderul runs si contin atat fisierele text, cat si imaginile cu obiectele detectate. Am convertit ulterior acest format in python pentru a putea cropa imaginile (am gasit x1, x2, y1, y2).



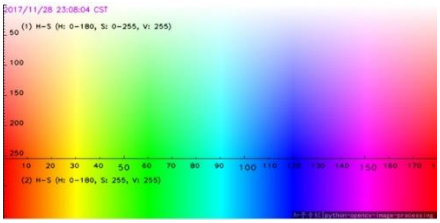
Rezultat detectie yolov5

3. Spatiul de culoare CIE Luv

Pentru a face conversia din RGB in CIE Luv, avemm nevoie de un spatiu de culoare auxiliar, XYZ. Astfel, cu ajutorul unor matrice de transformare, se realizeaza pe rand conversia RGB->XYZ, XYZ->CIE Luv. Cele 3 componente care descriu spatiul de culoare sunt L, u si v (luminozitate, u, v). Este folosit in special in aplicatii de grafica computerizata ce prelucreaza “culori luminate” (afectate de lumina).

4. Aplicare masca si transparenta

Pentru a extrage culoarea dorita, convertesc mai intai imaginea in spatiul de culoare HSV(este util pentru ca pot alege usor un interval minim si maxim pentru culoarea dorita, conform unei mape). Ulterior, specific ca masca sa contina numai valorile care fac parte din intervalul lower_purple – upper_purple. Ultimul pas il reprezinta aplicarea unui bitwise and intre imaginea initiala cu ea insasi, insa cu parametrul mask = masca pe care am obtinut-o mai sus.



Pentru transparenta, creez o noua imagine identica cu masca de mai sus, doar ca ii adaug si al 4-lea canal (alpha) care da transparenta: 0 = transparenta maxima, 255 = opacitate. Astfel, acolo unde in imaginea masca, avem pixeli negri (deci fundal), setez canalul alpha la 0 pentru a obtine transparenta. Metoda este asemanatoare cu laboratorul 5.