Webová aplikace Color Collector

Erica Nakada

2022/2023

Obsah

1 O aplikaci

Color Collector je webová aplikace, která umožňuje získat různou informaci o barvách v obrázku, který nahraje uživatel. V aplikaci jsou dostupné celkem tři nástroje: Color Picker pro zjištění konkrétního názvu barvy, Color Palette pro vygenerování barevné palety z obrázku a Color Detector pro detekci 8 barev z video streamu.

Projekt je rozdělen do dvou částí:

- Backend – potřebné výpočty pro zpracování informace od uživatele
- Frontend – samotná webová aplikace Color Collector

2 Použité knihovny

- Python 3.11.4 - https://www.python.org/

- Streamlit 1.25.0 - https://streamlit.io/
- Pillow 9.5.0 - https://pillow.readthedocs.io/en/stable/
- Scikit-Learn 1.3.0 - https://scikit-learn.org/stable/
- opencv-python-headless 4.8.0.76 - https://docs.opencv.org/4.x/
- Pandas - https://pandas.pydata.org/

Projekt je tvořen pomocí těchto nejpodstatnějších knihoven:

Streamlit
Tento open-source webový framework je klíčovou součástí aplikace. Zajišťuje tvorbu webových stránek v Pythonu. Platforma též poskytuje cloudové uložiště Streamlit Community Cloud, ve kterém lze veřejně publikovat a realizovat svoje stránky a metody.

Pillow a OpenCV
Obě knihovny slouží ke zpracování obrázku.

Scikit-Learn
Tato knihovna poskytuje řadu algoritmů a užitečných nástrojů strojového učení. Generátor barevné palety a detektor barev používají Scikit-Learn pro třídění pixelů do jednotlivých skupin dle barvy či zjištění nejbližších sousedů konkrétního barevného vektoru. Podrobný popis a vysvětlení použitých funkcí lze najít v tomto dokumentu v sekcích 4 a 5.

3 Struktura aplikace

```
mff_pg2_ColorCollector/
    models/
        training.csv
    .streamlit/
        config.toml
    pages/
        1_⛏_Color_Picker.py
        2_🎨_Color_Palette.py
        3_🦹_Color_Detector.py
    Home.py
    utils.py
    requirements.txt
```

3.1 Frontend

O samotný vzhled webové aplikace a její fungování se stará soubor domovské stránky Home.py a soubory ve složkách pages (vedlejší stránky) a .streamlit (design webové aplikace).

Veškerou dokumentaci a použití streamlit lze najít na oficiálních stránkách knihovny.

1_Color_Picker.py

Metoda streamlit_image_coordinates() je zodpovědná za zjištění souřadnic pixelu, na který uživatel kliknul (vrací slovník s klíčemi x a y). Tyto souřadnice se předají do funkce getpixel() z knihovny Pillow, která vrátí tuple s RGB hodnotou tohoto pixelu. Pomocí funkce int2hex() se zjistí i jeho HEX hodnota.

2_Color_Palette.py

Nahraný obrázek se pomocí knihovny Pillow zmenší na rozměr 500x500 (se zachováním poměru výšky a šířky) a do proměnné rgb_list se uloží seznam seznamů se 3 integer hodnotami, což reprezentuje sled RGB hodnot každého pixelu od levého horního rohu obrázku. Tento seznam pixelů se předá do 1 z funkcí pick_colors či pick_colors2, které vrátí opět seznam stejné struktury, který teď ale reprezentuje barvy vygenerované palety.

3_Color_Detector.py

Metoda webrtc_streamer() je zodpovědná za video stream z kamery v reálném čase. Pomocí parametru video_frame_callback se na každý postupující snímek zavolá stejnojmenná funkce. Snímek na vstupu a výstupu této funkce je objektem třídy VideoFrame z knihovny PyAV (zajišťuje použití FFmpeg v pythonu, což je soubor knihoven pro práci s audio a video multimédiemi v C, je automaticky importován spolu s streamlit-webrtc). Po přeformátování vstupního snímku do seznamu pixelů stejné struktury jako dříve se proměnná předá do funkce get_rgb(), která vytěží 1 RGB hodnotu charakterizující daný snímek. Tato informace se předá do natrénovaného modelu strojového učení.

3.2 Backend

V souboru utils.py se nachází všechny potřebné výpočty pro vytvoření dynamického obsahu stránky. Nachází se tu funkce ze sekce 4 a 5.

4 Vývoj modelů pro stránku Color Palette

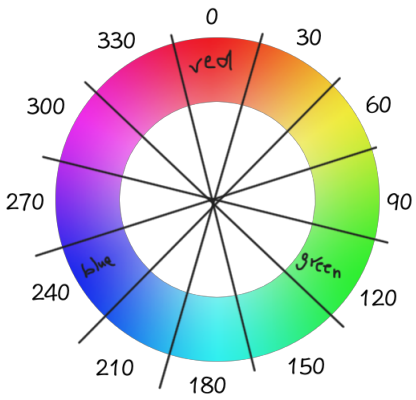
Aplikace využívá dva modely pro vygenerování barevné palety z obrázku. Základ u obou tvoří klasický klasterizační algoritmus K-Means, jehož realizaci poskytuje knihovna scikit-learn. Kód pro oba modely se nachází v souboru utils.py.

def pick_colors():

V prvním primitivnějším modelu si uživatel může sám vybrat počet barev ve

výsledné paletě. Tento počet se rovnou předá do povinného parametru n_clusters v Kmeans() spolu se seznamem RGB hodnot všech pixelů v obrazu. Funkce vrací centroidy (RGB hodnoty centrálních barev v seznamu) nalezených klasterů.

def pick_colors2():
Druhý model se sám postará o počet barev ve výsledné paletě. Do K-means algoritmu se předá opět seznam RGB hodnot všech pixelů v obrazu a parametr n_clusters = 24. Nalezené 24 barev v HSV kódu (kde hodnota hue se udává ve stupních) se pomocí funkcí final_colors() a get_category() roztřídí do příslušných barevných skupin (na obr. rozdělení do skupin dle barevného kola na obrázku po 30 stupních). V rámci každé nalezené skupiny se vybere nejvíc saturovaná a nejjasnější barva.



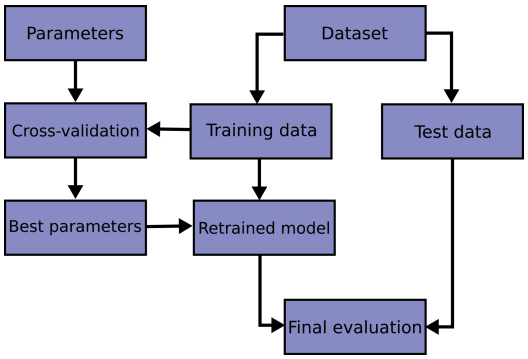
5 Vývoj detektoru barev pro stránku Color Detector

Poznámka: ve složce models se nachází obrázky pro testovací data, obrázky pro trénovací data a ipynb soubor Jupyter Notebooku, které nejsou součástí samotné webové aplikace, ale sloužili k vývoji modelu. Obrázky byly staženy z platformy Kaggle (viz sekce 6. zdroje).

Detekce 8 barev ze snímku se provádí pomocí algoritmu KNN (K nejbližší sousedy), jehož realizaci poskytuje knihovna scikit-learn. Závěrečný kód pro model se nachází v souboru utils.py a 3_Color_Detector.py.

Vývoj modelu v KNN_for_Color_Detector.ipynb souboru se skládal ze dvou částí:

- vytěžení featury z obrázků pro tvorbu trénovacích a testovacích dat
- cross validation pro hledání vhodného parametru n_neighbors algoritmu KNN konkrétně pro trénovací datový soubor z prvního bodu (tento soubor a parametr se poté použije v závěrečné verzi modelu pro webovou aplikaci)



def extract_color_histogram():
funkce v KNN_for_Color_Detector.ipynb souboru byla aplikována na každý obrázek ze složky models/training_dataset a models/test_dataset. Pomocí OpenCV a barevného histogramu se v každém z barevných kanálu (červená, zelená, modrá) obrázku najde hodnota s největším počtem pixelů. Tyto tři hodnoty se z každého obrázku uloží do DataFramů df_train a df_test spolu s označením třídy, do které patří (černá, bílá, modrá, zelená, oranžová, červená, fialová a žlutá).

Cross validation se prováděl na datech z df_train postupně pro n_neighbors od 2 po 31. V každé iteraci se uložilo průměrné skóre přesnosti pro daný počet n nejbližších sousedů, což se na konci vyvedlo do grafu. Pro danou sadu trénovacích dat se našel optimální počet n_neighbors = 4. Přetrénovaný model s tímto parametrem na testovacích datech df_test ukázal skóre přesnosti 80%, což je dostatečné pro jeho použití.

def load_model():
Dataframe df_train exportovaný do csv souboru ve složce models/training.csv se použije v utils.py pro trénování modelu na detekci barev, spolu s nalezeným optimálním parametrem n_neighbors=4

def get_rgb():
Přijímá na vstup snímek z video streamu a těží z něj featuru stejným způsobem jako extract_color_histogram(). Na výstupu je seznam seznamu ze tří čísel, který se zadá do natrénovaného modelu v 3_Color_Detector.py

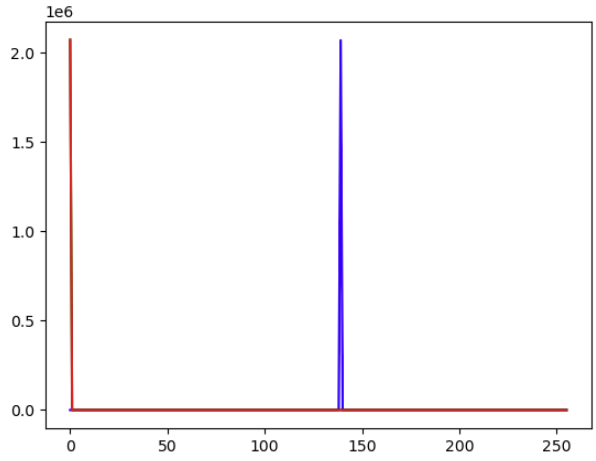
6 Zdroje

6.1 Použité knihovny

- Python 3.11.4 - https://www.python.org/
- Streamlit 1.25.0 - https://streamlit.io/
- Pillow 9.5.0 - https://pillow.readthedocs.io/en/stable/

```
In [3]: img_path = 'blue.png'
row = {'r': None, 'g': None, 'b': None, 'category': 'blue'}
img = cv2.imread(img_path)
channels = cv2.split(img)
for i, chan in enumerate(channels):
    hist = cv2.calcHist([chan], [0], None, [256], [0, 256])
    peak = np.argmax(hist)
    if i == 0:
        row['b'] = peak
        plt.plot(hist, color='b')
    elif i == 1:
        row['g'] = peak
        plt.plot(hist, color='g')
    else:
        row['r'] = peak
        plt.plot(hist, color='r')
print(row)
plt.show()
```

```
{'r': 0, 'g': 0, 'b': 139, 'category': 'blue'}
```



Obrázek 1: Barevný histogram modrého obrázku

- Scikit-Learn 1.3.0 - https://scikit-learn.org/stable/
- opencv-python-headless 4.8.0.76 - https://docs.opencv.org/4.x/
- Pandas - https://pandas.pydata.org/

6.2 Jiné

- Kaggle obrázky - https://www.kaggle.com/datasets/ayanzadeh93/color-classification, https://www.kaggle.com/datasets/adikurniawan/color-dataset-for-color-recognition
- KNN, cross validation - https://medium.com/swlh/k-nearest-neighbor-ca2593d7a3c4, https://inside-machinelearning.com/en/cross-validation-tutorial/, https://www.datacamp.com/tutorial/k-nearest-neighbor-classification-scikit-learn
- feature extraction - https://ijirt.org/master/publishedpaper/IJIRT150658_PAPER.pdf,file:///C:/Users/erica/Downloads/admin,+4981-Article+

Text-20829-3-10-20200810.pdf

- webrtc - https://dev.to/whitphx/developing-web-based-real-time-videoaudio-processing-app
  https://github.com/whitphx/streamlit-webrtc
- color palette, kmeans - https://towardsdatascience.com/k-means-clustering-algorithm-applicat