

Міністерство освіти і науки України
Національний університет «Запорізька політехніка»
Кафедра програмних засобів

ЗВІТ

Дисципліна «Розробка прикладних програм»

Робота №5

Тема «Машинне навчання»

Виконав варіант 19

Студент КНТ-122

Онищенко О. А.

Прийняли

Викладач

Дейнега Л. Ю.

2024

МЕТА РОБОТИ

Ознайомитись з основними пакетами, які використовуються для машинного навчання в програмах, написаних мовою Python.

Навчитися розробляти сучасні інтелектуальні системи з використанням методів машинного навчання.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

Розробити програмне забезпечення, яке дозволяє виділити в даному зображені набір основних кольорів. Кількість кольорів може бути вибрана попередньо з заданого діапазону. Алгоритм кластеризації обрати самостійно. Для обраного зображення визначити також набір зображень з подібними кольорами. Банк зображень повинен формуватися автоматизовано.

ТЕКСТ ФАЙЛУ

```
from sklearn.cluster import KMeans
import matplotlib.image as img
from nicegui import ui, app
import numpy as np
import os

class Color:
    def __init__(self,
                 r: int,
                 g: int,
                 b: int,
                 ):
        self.red=int(r)
        self.green=int(g)
        self.blue=int(b)

    def __str__(self):
        return f'{self.red} {self.green} {self.blue}'

    def __repr__(self):
        return f'{self.red} {self.green} {self.blue}'
```

```

def get_html_rgb_value(self):
    return f'rgb({self.red},{self.green},{self.blue})'

def get_average(self):
    return (self.red+self.green+self.blue)//3

def get_n_most_common_colors(
    image_path:str,
    colors_count:int,
) :
    ...
    < image_path: str
        full image path as string
    < colors_count: int
        number of most common colors to return for a given image

    > palette: list[Color]
        has length of colors_count
        each element is a Color object with values for red, green and
blue as ints
    ...

# Read an image, get it into an object to work with
image=img.imread(image_path)
# Read image's width, height, depth
w,h,d=tuple(image.shape)
# Get a pixel size by reshaping the image into rows
pixel=np.reshape(image, (w*h,d))
# Make a K-Means algorithm model, fit it into a pixel
model=KMeans(n_clusters=colors_count,random_state=40).fit(pixel)
# Make a colors palette by extracting cluster centers from the
model
palette=np.uint8(model.cluster_centers_)
# Make a list for storing Colors data
res=[]
# Go through each color in palette, colors_count times
for c in palette:
    # Extract the red, green and blue values for each color
    r,g,b=c
    # Make and add a Color object for the colors
    res.append(Color(r,g,b))
# Return the produced list of Colors
return res

def get_colors_dataset(
    images_folder:str,
    image_files:list[str],
) :
    colors_dataset=dict()
    for image_file_name in image_files:
        image_path=os.path.join(images_folder,image_file_name)
        this_image_colors=get_n_most_common_colors(
            image_path=image_path,
            colors_count=colors_count
        )
        colors_dataset[image_file_name]=this_image_colors
    return colors_dataset

```

```

def get_average_colors(
    colors_dataset:dict,
):
    def get_average_color(
        colors:list[Color]
    ):
        return round(sum([c.get_average() for c in colors])/len(colors))

    return {name:get_average_color(colors) for name,colors in colors_dataset.items()}

ROOT_FOLDER=os.path.dirname(os.path.abspath(__file__))
IMAGE_FOLDER_PATH=os.path.join(ROOT_FOLDER,'images')
IMAGE_FILE_NAMES=os.listdir(IMAGE_FOLDER_PATH)

LABEL_CLASSES='font-medium text-lg'
COLOR_CLASSES='flex-1 py-7 rounded-md'

app.add_media_files('/images',IMAGE_FOLDER_PATH)
image_file=IMAGE_FILE_NAMES[0]
colors_count=3

image_colors=get_colors_dataset(IMAGE_FOLDER_PATH,IMAGE_FILE_NAMES)
average_image_colors=get_average_colors(image_colors)

def update_image_output():
    results_image.source=os.path.join(IMAGE_FOLDER_PATH,image_file)

def update_colors():
    colors_count_label.text=f'The image has {colors_count} most common
color{"s" if colors_count>1 else ""}'

    image_average_color=average_image_colors[image_file]
    colors_list=list(set((image_name,color_value) for
image_name,color_value in average_image_colors.items() if
color_value<=image_average_color and image_name!=image_file))
    closest_colors=sorted(
        colors_list,
        key=lambda color: color[1],
        reverse=True
    )

    this_image_colors:list[Color]=image_colors[image_file]
    closest_color_one.style(f'background:
{this_image_colors[0].get_html_rgb_value()};')
    closest_color_two.style(f'background:
{this_image_colors[1].get_html_rgb_value()};')
    closest_color_two.visible=colors_count>=2
    closest_color_three.style(f'background:
{this_image_colors[2].get_html_rgb_value()};')
    closest_color_three.visible=colors_count>=3

    closest_color_image_paths=[
        os.path.join(IMAGE_FOLDER_PATH,this_file_name) for
this_file_name,this_color_value in closest_colors
    ]

```

```

similar_image_one.source=closest_color_image_paths[0]
similar_image_two.source=closest_color_image_paths[1]
similar_image_three.source=closest_color_image_paths[2]

def update_ui():
    update_image_output()
    update_colors()

ui.label('Image to process').classes(LABEL_CLASSES)
ui.select(
    options=IMAGE_FILE_NAMES,
    with_input=True,
    value=image_file,
    on_change=update_ui,
).classes('w-full').bind_value(globals(),'image_file')

ui.label('Number of colors').classes('mt-7 '+LABEL_CLASSES)
ui.slider(
    min=1,
    max=3,
    value=colors_count,
    on_change=update_ui,
).bind_value(globals(),'colors_count')
with ui.row().classes('flex justify-between w-full'):
    ui.label(1)
    ui.label(2)
    ui.label(3)

ui.label(f'Results for {image_file}').classes('mt-12 '+LABEL_CLASSES)
results_image=ui.image(os.path.join(IMAGE_FOLDER_PATH,image_file)).classes('rounded-md object-center').style('max-height: 37vh;')

colors_count_label=ui.label(f'Most common colors ({colors_count})').classes('mt-7 '+LABEL_CLASSES)
with ui.row().classes('w-full flex gap-3'):
    closest_color_one=ui.element('span').classes(COLOR_CLASSES)
    closest_color_two=ui.element('span').classes(COLOR_CLASSES)
    closest_color_three=ui.element('span').classes(COLOR_CLASSES)

ui.label(f'Images with similar colors').classes('mt-7 '+LABEL_CLASSES)
with ui.row().classes('w-full flex gap-3'):
    similar_image_one=ui.image().classes('flex-1 rounded-md max-h-60 display-cover')
    similar_image_two=ui.image().classes('flex-1 rounded-md max-h-60 display-cover')
    similar_image_three=ui.image().classes('flex-1 rounded-md max-h-60 display-cover')

update_ui()
ui.run(favicon='img',title='Image Colors')

```

РЕЗУЛЬТАТИ ВИКОНАННЯ

Показ роботи програми у вигляді відео за [посиланням](#).

При вході на сторінку обрано перше зображення зі списку:

Image to process

alaska_wilderness_mountains_67829.jpg

Number of colors

1 2 3

Results for alaska_wilderness_mountains_67829.jpg



The image has 3 most common colors



Images with similar colors



Рисунок 1.1 – Вигляд програми при вході

При зміні зображення оновлюється користувачький інтерфейс: зображення у секції результатів, кольори та схожі зображення. Приклад нижче:

Image to process

mountains_landscape_mountain_nature.jpg

Number of colors

1 2 3

Results for alaska_wilderness_mountains_67829.jpg



The image has 3 most common colors



Images with similar colors



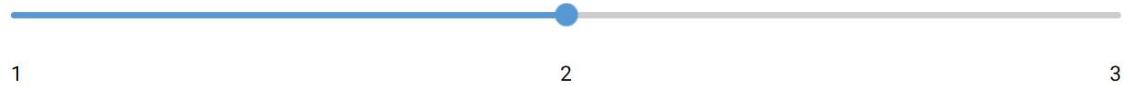
Рисунок 1.2 – Оновлення інтерфейсу при обранні іншого зображення

При зміні кількості кольорів змінюється вивід у секції кольорів:

Image to process

nature_autumn_river_958823.jpg

Number of colors



Results for alaska_wilderness_mountains_67829.jpg



The image has 2 most common colors



Images with similar colors



Рисунок 1.3 – Зміна кількості кольорів

Image to process

el_tatio_chile_south.jpg

Number of colors



Results for alaska_wilderness_mountains_67829.jpg



The image has 1 most common color



Images with similar colors



Рисунок 1.4 – Зміна кількості кольорів на один

Опис алгоритма розв'язання задачі

Алгоритм працює наступним чином:

1. Завантажує зображення з наданого шляху;
2. Отримує розміри зображення: ширину, довжину та глибину;
3. Змінює форму зображення на двомірний масив з кожним рядком як піксель;
4. Створює модель алгоритму K-середніх. Як кількість кластерів приймає значення кількості кольорів, отримане від користувача у інтерфейсі;
5. Підганяє модель під піксель, отримує центри кластерів, які є найпоширенішими кольорами;
6. Конвертує кластери до типу цілого числа, повертає значення як список змінних типу Color.

Джерело

Текст алгоритму:

```
def get_n_most_common_colors(
    image_path:str,
    colors_count:int,
) :
    ...
    < image_path: str
        full image path as string
    < colors_count: int
        number of most common colors to return for a given image

    > palette: list[Color]
        has length of colors_count
        each element is a Color object with values for red, green and
blue as ints
    ...

# Read an image, get it into an object to work with
image=img.imread(image_path)
# Read image's width, height, depth
w,h,d=tuple(image.shape)
# Get a pixel size by reshaping the image into rows
```

```

pixel=np.reshape(image, (w*h, d))
# Make a K-Means algorithm model, fit it into a pixel
model=KMeans(n_clusters=colors_count,random_state=40).fit(pixel)
# Make a colors palette by extracting cluster centers from the
model
palette=np.uint8(model.cluster_centers_)
# Make a list for storing Colors data
res=[]
# Go through each color in palette, colors_count times
for c in palette:
    # Extract the red, green and blue values for each color
    r,g,b=c
    # Make and add a Color object for the colors
    res.append(Color(r,g,b))
# Return the produced list of Colors
return res

```

Результати тестування

Після проведення тестування (zmіни зображень, zmіни кількості кольорів) помилок не було виявлено, окрім однією: іноді показує те саме зображення у секції Схожих зображень. Проблему не було вирішено, але шматок коду для її потенційного вирішення:

```

colors_list=list(set((image_name,color_value) for
image_name,color_value in average_image_colors.items() if
color_value<=image_average_color and image_name!=image_file))

```

Перелік проблем під час виконання

Під час виконання роботи виникло кілька проблем. Нижче наведено опис кожної та її вирішення:

1 Як змінити колір коробочки кольору для секції найпоширеніших кольорів?

Опис: у секції результатів є підсекція яка виводить список найпоширеніших кольорів зображення. До елементів користувацького інтерфейсу було додано елементи `span` з відповідними стилями (закруглені

края, трохи вертикального внутрішнього відступу, задання ширини на таку що покриває все вільне місце), але для зміни кольору потрібно:

Використання кастомного кольору: Після спроби застосувати кастомний колір як клас Tailwind (`bg-[#customColorHexCode]`) нічого не спрацювало. Тоді для вирішення цього потрібно прописати користувацький CSS код із вказанням кольору як значення CSS `(background: rgb(custom, color, values);)`.

Вирішення: після пошуку шляхів вирішення проблеми Господь Ісус Христос підказав спосіб оновлення стилів елементу через зміну властивості `styles`. Проблему було вирішення застосуванням методу `style` для кожного елементу коробки кольору зі значенням кольору, отриманого від класу `Color`.

2 Як зв'язати позицію елементу користувацького інтерфейсу зі змінною в коді аби використовувати його значення для динамічного відображення даних?

Опис: ця проблема виникла рано під час розробки. Потрібно було отримати значення елементу повзунка (використовується для отримання від користувача кількості кольорів) для застосування його у обрахунку кольорів.

Вирішення: благодаттю Господа нашого Ісуса Христа було знайдено метод `bind_value()`, який і було застосовано до глобальної змінної `colors_count`. Після цього значення цієї змінної почало оновлюватися зі зміною позиції повзунка.

ВИСНОВКИ ПО РОБОТИ

В результаті виконання було розроблено веб-застосунок на мові програмування Python використовуючи бібліотеки scikit-learn та nicegui. Застосунок дозволяє обрати зображення із завантажених до файлів проєкту і для нього показує список найбільш використаних кольорів. Застосунок працює у браузері, що дозволяє користувачам мати доступ до нього з будь-якого пристроя, який має доступ до веб-браузера. Вся слава Господу Ісусу Христу Всемогутньому БОГУ Амінь.