Pengaplikasian Pengolahan Gambar Menggunakan Fungsi Lambda dan Map

Asa Do'a Uyi (122450005), Try Yani Rizki Nur Rohmah (122450020), Marleta Cornelia Leander (122450092), Sofyan Fauzi Dzaki Arif (122450116), Amalia Melani Putri (122450122)

1. Pendahuluan

Dalam era 4.0, penggunaan gambar dan pengolahan data berfungsi untuk memvisualisasikan ekspresi. Pengolahan gambar merupakan suatu aspek penting dalam perkembangan teknologi untuk meningkatkan efisiensi dari banyaknya teknologi yang salah satunya ialah kecerdasan buatan. Salah satu teknologi untuk meningkatkan fleksibilitas dalam pengolahan gambar diperlukannya inovasi seperti penggunaan fungsi lambda dan fungsi Map.

Fungsi lambda dalam bahasa pemrograman python adalah suatu fungsi yang dibuat dalam satu baris yang bisa digunakan kembali an bersifat tidak memiliki nama dan dapat digunakan dalam membuat fungsi *custom* dan digunakan untuk membuat suatu fungsi yang sederhana seperti penjumlahan, pembagian, perkurangan dan perkalian.

Fungsi map dalam pemrograman python adalah fungsi yang dapat mengambil fungsi yang diimplementasikan ke setiap elemen dalam data set yang berguna untuk mengembalikan array yang dibentuk dengan cara memetakan setiap nilainya kedalam fungsi array ke nilai baru yang menerapkan suatu fungsi lambda guna membuat suatu nilai baru.

2. Metode

Pada praktikum ini, metode atau fungsi yang digunakan adalah metode lambda dan fungsi map serta fungsi tambahan yaitu fungsi *grayscale* (Fungsi *user-defined)*. Fungsi lambda digunakan untuk membuat suatu fungsi tanpa sebuah nama atau biasa juga disebut sebagai anonim dalam Python. Selain itu, fungsi lambda ini juga dapat memiliki nol atau lebih parameter. Jadi, secara singkat fungsi lambda dapat digunakan untuk membuat suatu fungsi yang sederhana, sehingga tidak banyak memakan baris kode.

Selanjutnya fungsi lambda akan diteruskan ke fungsi map. Fungsi map berguna untuk menerapkan fungsi lambda tersebut pada setiap elemen yang ada dalam list, dalam hal ini list yang dimaksud adalah baris gambar. Oleh karena itu hasilnya adalah matriks pixel yang telah diubah menjadi warna *grayscale*. Sebagai tambahan fungsi *grayscale* berfungsi untuk mengubah warna pada pixel.

Kita dapat membahas satu per satu tentang metode yang digunakan kali ini:

```python
# Misalkan kita memiliki fungsi untuk mengubah nilai piksel menjadi grayscale
def grayscale(pixel):
    # Hitung nilai rata-rata dari nilai RGB piksel
    gray_value = sum(pixel) // 3
    # Kembalikan nilai grayscale dalam bentuk tuple (nilai, nilai, nilai)
    return (gray_value, gray_value, gray_value)
```

Fungsi *grayscale* ini bekerja dengan cara mengambil sebuah piksel dalam bentuk *tuple Red Green and Blue*, menghitung nilai rata-rata dari nilai RGB yang kita dapatkan, dan mengembalikan nilai *tuple* baru yang merepresentasikan warna *grayscale* pada foto yang dipilih.

```python
# Transformasi grayscale menggunakan map dan lambda
image_pixels = [
    [(255, 0, 0), (0, 255, 0), (0, 0, 255)],
    [(255, 255, 0), (255, 0, 255), (0, 255, 255)],
    [(128, 128, 128), (64, 64, 64), (192, 192, 192)]
]

transformed_image = list(map(lambda row: list(map(grayscale, row)), image_pixels))
```

Selanjutnya adalah penerapan transformasi foto untuk *grayscale* yang menggunakan fungsi map serta lambda. Dapat dilihat pada baris kode di atas. Fungsi map di atas digunakan untuk memakai transformasi fungsi *grayscale* pada setiap baris yang ada di dalam matriks image_pixels. Selanjutnya, fungsi map kedua dipakai untuk menggunakan transformasi fungsi *grayscale* pada setiap piksel yang ada dalam baris. Lalu, untuk pemakaian fungsi lambda di sini digunakan untuk menginisiasi fungsi anonim yang mengambil satu argumen yaitu *row* (untuk kasus pertama) dan *pixel* (untuk kasus kedua). Setelah itu diterapkannya fungsi *grayscale* ke dalam argumen/parameter tersebut.

```python
import cv2
import numpy as np

def is_blurry(image_path, threshold=100):
    # Baca gambar menggunakan OpenCV
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Menghitung perbedaan intensitas antara piksel yang berdekatan
    blur_metric = np.mean(np.abs(np.diff(img)))

    # Jika blur_metric lebih kecil dari threshold, anggap gambar blur
    return blur_metric < threshold

# Daftar nama file gambar yang ingin diuji
image_paths = ["bluring.jpg"]

# Tentukan threshold sesuai kebutuhan
threshold_value = 100

# Gunakan fungsi lambda dan fungsi map untuk memeriksa setiap gambar
results = map(lambda path: is_blurry(path, threshold_value), image_paths)

# Tampilkan hasil
for path, result in zip(image_paths, results):
    if result:
        print(f"{path}: Gambar blur.")
    else:
        print(f"{path}: Gambar tidak blur.")
```

Selanjutnya adalah penggunaan fungsi map dan fungsi lambda pada bagian pengecekan gambar apakah *blur* atau tidak. Dapat kita lihat di baris ke 20 (Pada komen #Gunakan Fungsi Lambda dan Fungsi Map untuk Memeriksa Setiap Gambar), terdapat penggunaan fungsi lambda untuk membuat fungsi anonim yang memeriksa gambar di suatu path

tertentu (parameter path) apakah masuk ke dalam kategori gambar *blur* atau tidak menggunakan fungsi is_blurry. Lalu, terdapat fungsi map yang dipakai pada list image_paths. Hal ini berfungsi agar setiap gambar diperiksa apak blur atau tidak. Hasil yang diberikan akan dijadikan elemen dari objek map.

3. Pembahasan

3.1 *Import Library*

```
import cv2
import numpy as np
```

OpenCV (cv2) digunakan untuk manipulasi gambar, sedangkan NumPy (numpy) diterapkan untuk operasi *array* dan komputasi numerik pada pengolahan gambar. Kedua modul ini memainkan peran penting dalam mengelola dan menganalisis data gambar dengan efisiensi.

3.2 Fungsi is_blurry

```
def is_blurry(image_path, threshold=100):
    # Baca gambar menggunakan OpenCV
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Menghitung perbedaan intensitas antara piksel yang berdekatan
    blur_metric = np.mean(np.abs(np.diff(img)))

    # Jika blur_metric lebih kecil dari threshold, anggap gambar blur
    return blur_metric < threshold
```

Fungsi is_blurry menguji tingkat kekaburan gambar dengan membaca path dan threshold, kemudian mengembalikan *True* jika kekaburan melebihi ambang yang ditentukan.

3.3 Menyediakan daftar gambar yang akan diuji

```
# Daftar nama file gambar yang ingin diuji
image_paths = ["bluring.jpg"]
```

Variabel image_paths dengan nilai ["bluring.jpg"] berisi daftar nama file gambar yang akan diuji kekaburannya, menjadi input untuk memastikan fungsi pengujian dapat diaplikasikan pada gambar dengan nama "bluring.jpg".

3.4 Menentukan nilai ambang dan menggunakan fungsi lambda untuk memeriksa setiap gambar dalam daftar

```
# Tentukan threshold sesuai kebutuhan
threshold_value = 100

# Gunakan fungsi lambda dan fungsi map untuk memeriksa setiap gambar
results = map(lambda path: is_blurry(path, threshold_value), image_paths)
```

Variabel threshold_value (nilai 100) menetapkan ambang untuk pengujian kekaburan. Hasil pengujian setiap gambar dalam daftar image_paths diperoleh melalui fungsi lambda dan map, di mana is_blurry diaplikasikan dengan ambang yang ditetapkan untuk menentukan gambar yang dianggap "blur".

3.5 Menampilkan hasil

Hasil dari kode yang dibuat :

- Menerapkan fungsi map untuk menguji setiap gambar dengan fungsi lambda.
- Menampilkan hasil pengujian, memberikan informasi apakah gambar dianggap "*blur*" atau tidak.

```
# Tampilkan hasil
for path, result in zip(image_paths, results):
    if result:
        print(f"{path}: Gambar blur.")
    else:
        print(f"{path}: Gambar tidak blur.")
```

```
bluring.jpg: Gambar blur.
```

4. Kesimpulan

Penggunaan fungsi lambda dalam pengolahan gambar memungkinkan untuk pembuatan fungsi yang sederhana dan tanpa nama dan fungsi map dalam pengolahan gambar menerapkan transformasi tersebut pada setiap elemennya. Selain itu fungsi lambda dan fungsi map juga memeriksa apakah gambar masuk kedalam kategori *blur* atau tidak dengan menggunakan ambang batas tertentu. Didalam metode ini juga dapat memberikan pemahaman yang baik tentang efisiensi dalam pengolahan gambar.

Link Colab:
https://colab.research.google.com/drive/1MkzKnREFXFb_xIwgmZebYT7b9ZMNHNm9?usp=sharing