

Tugas 2: Visi Komputer

Judul Tugas 2: Klasifikasi Jenis Citra Mobil dengan Menggunakan Random Forest

Ketua Team:

- Alvito Jordan (201110586)

Anggota Team:

- Evander (201110440)
- Enrique Kuandy (201111290)
- Willy Wijaya (201110435)

I. Tahapan Algoritma

Link Dataset : [Link Dataset \(sama seperti Tugas 1\)](#)

Link Google Colab : [Link Tools](#)

Algoritma yang digunakan : Algoritma Random Forest

Tools yang digunakan adalah Google Colab. Dataset yang digunakan adalah dataset citra yang berupa 25 buah citra mobil sedan dan 25 buah mobil SUV. Sehingga total dataset yang digunakan dalam tugas ini adalah 50 buah citra. Algoritma yang digunakan adalah Random Forest.

Random Forest sendiri adalah salah satu algoritma yang menggabungkan keluaran dari beberapa decision tree. Sesuai dengan nama 'Forest' yang berarti hutan dibentuk dari banyak pohon. Oleh karena itu Random Forest juga disebut sebagai algoritma hasil dari pengembangan Algoritma Decision Tree. Cara kerjanya secara sederhana adalah sebagai berikut :

- Algoritma akan memilih sampel acak dari dataset yang di sediakan.
- Membuat decision tree untuk setiap sampel yang dipilih akan didapatkan hasil prediksi dari setiap decision tree yang telah dibuat.
- Dilakukan proses voting untuk setiap hasil prediksi.
- Hasil prediksi yang paling banyak dipilih (vote terbanyak) akan digunakan sebagai prediksi akhir.

Tahap 1. Import Library

Langkah pertama yang kami lakukan adalah mengimport Library yang sesuai dengan judul kami. Berikut adalah library yang kami gunakan :

```
from google.colab import files
import numpy as np
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
import os
import cv2
from joblib import dump, load
import imghdr
```

Berikut adalah penjelasan setiap import :

- **files** akan digunakan untuk mengelola file di Google Colab.
- **np** akan digunakan untuk mengelola array

- **RandomForestClassifier** akan digunakan untuk membangun model klasifikasi menggunakan metode Random Forest.
- **train_test_split** akan digunakan untuk membagi dataset menjadi subset pelatihan dan pengujian.
- **accuracy_score**, **confusion_matrix**, dan **classification_report** digunakan untuk mengukur performa model klasifikasi.
- **plt** akan digunakan untuk membuat plot grafik dan visualisasi data.
- **os** akan digunakan untuk berinteraksi dengan sistem operasi, seperti mengakses direktori dan file.
- **cv2** akan digunakan untuk pengolahan citra dan penglihatan komputer.
- **dump**, dan **load** akan digunakan untuk menyimpan dan memuat model dari file.
- **imghdr** akan digunakan untuk mengecek tipe file gambar dengan memeriksa header file.

Tahap 2. Import Dataset

Berikut adalah code yang digunakan untuk mengimport dataset. Dataset yang berhasil diupload akan langsung diunzip. (Input dataset yang sudah berbentuk zip disini)

```
# Upload file
files.upload()

# ekstrak file zip
!unzip CarDataset.zip
```

Choose Files CarDataset.zip

- **CarDataset.zip(application/x-zip-compressed)** - 4286275 bytes, last modified: 5/9/2023 - 100% done

Saving CarDataset.zip to CarDataset.zip
Archive: CarDataset.zip

Tahap 3. Preprocessing Dataset Citra

Berikut adalah fungsi yang kami buat untuk preprocessing. Preprocessing yang kami gunakan adalah mengubah ukuran gambar menjadi sama besar (Hal ini bertujuan agar model Random Forest dapat digunakan), Gaussian Blur dan Deteksi Tepi dengan menggunakan Canny.

```
def preprocessing(image_path):
    # Membaca gambar menggunakan OpenCV
    image = cv2.imread(image_path)

    # Resize gambar menjadi ukuran 100x100
    resized_image = cv2.resize(image, (500, 500))

    # Menghaluskan gambar dengan filter Gaussian
    gaussian = cv2.GaussianBlur(resized_image, (5, 5), 0)

    # Deteksi tepi menggunakan Canny edge detection
    edges = cv2.Canny(gaussian, 100, 200)

    return edges
```

Untuk melihat hasil dari preprocessing maka kita dapat menggunakan **plt** untuk mengelolanya. Berikut adalah codenya.

```
image_path = "/content/CarDataset/01.jpg"
result = preprocessing(image_path)

# Menampilkan gambar hasil preprocessing
plt.imshow(result, cmap='gray')
plt.axis('off')
plt.show()
```

Pada code diatas, kami mencoba menggunakan fungsi preprocessing pada salah satu sampel pada dataset (01.jpg). Berikut adalah hasilnya :



Tahap 4. Memberikan Label pada dataset

Dalam tahapan ini, dataset yang sudah diunzip sebelumnya akan diberikan labelnya. Di mana 25 citra pertama dalam dataset tersebut adalah **"Mobil Sedan"** dan 25 citra sisanya adalah **"Mobil SUV"**. Untuk **"Mobil Sedan"** akan kami beri label **0** dan **"Mobil SUV"** dengan label **1**. Berikut adalah code untuk memberikan label pada dataset yang kemudian akan disimpan kedalam array dengan menggunakan numpy.

```

data = []
labels = []

# Mendapatkan path folder yang berisi gambar
folder_path = "/content/CarDataset"

# Mendapatkan semua file gambar dalam folder
image_files = [file for file in os.listdir(folder_path) if file.endswith(('.jpg', '.jpeg', '.png'))]

# Mengurutkan daftar file
image_files.sort()

# Memproses setiap file gambar dalam folder
for i, image_file in enumerate(image_files):

    # Mendapatkan path file gambar
    image_path = os.path.join(folder_path, image_file)

    # Memanggil fungsi preprocessing untuk setiap citra
    result = preprocessing(image_path)

    data.append(result)

# Menetapkan label berdasarkan indeks
if i <= 24:
    labels.append(0) # Mobil Sedan
else:
    labels.append(1) # Mobil SUV

X = np.array(data)
y = np.array(labels)

```

Pada code diatas data dan label akan diinisialisasi dengan array kosong. Kemudian folder dataset akan dikumpulkan kedalam **image_files**. Selanjutnya file tersebut akan diurutkan agar tidak terjadi kesalahan label pada dataset. Lakukan perulangan untuk membaca setiap citra yang ada didalam folder. Gambar tersebut akan kita preprosesing dengan fungsi preprocessing sebelumnya dan disimpan kedalam array **data** dan sekaligus memberikan labelnya yang disimpan kedalam array **labels**. Library yang digunakan disini adalah **numpy** dan **os**.

Tahap 5. Pisahkan Data Menjadi Data Training dan Testing

Gunakan **train_test_split** untuk memisahkan dataset menjadi data training dan data testing. Disini kami memisahkan dataset menjadi 80% data training dan 20% data testing. Berikut adalah codenya.

```

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, stratify=y, random_state = 30)
print(len(X_train), len(X_test), len(y_train), len(y_test))

```

40 10 40 10

Dapat dilihat data training sebanyak 40 gambar dengan label dan data testing sebanyak 10 gambar dengan label.

Untuk memastikan agar label 0 dan 1 itu seimbang didata testing maka gunakan stratify=y. Hal itu bertujuan agar data yang kita uji coba seimbang. Untuk membuktikannya dapat menggunakan code berikut.

```
# Menampilkan jumlah data mobil sedan dan mobil SUV pada data training dan data testing
print("Jumlah data mobil sedan (training):", sum(y_train == 0))
print("Jumlah data mobil SUV (training):", sum(y_train == 1))
print("Jumlah data mobil sedan (testing):", sum(y_test == 0))
print("Jumlah data mobil SUV (testing):", sum(y_test == 1))
```

```
Jumlah data mobil sedan (training): 20
Jumlah data mobil SUV (training): 20
Jumlah data mobil sedan (testing): 5
Jumlah data mobil SUV (testing): 5
```

Dapat dilihat jumlah data mobil sedan dan mobil SUV yang ditesing masing masing berjumlah 5 buah. Maka data testing sudah menjadi seimbang. Ini adalah dampak dari menggunakan stratify=y pada code sebelumnya. Hal ini tentu akan jauh lebih baik dibandingkan datanya tidak seimbang.

Tahap 6. Mengubah Dimensi data menjadi 2 dimensi

Sebelum membuat model Random Forest, kita terlebih dahulu cek dimensi pada gambar dan labelnya. Kita dapat menggunakan shape untuk melihatnya.

```
print(X.shape)
(50, 500, 500)
```

Dapat dilihat bahwa dimensi pada X atau data gambar masih lebih dari 2 dimensi dimana angka 50 pertama adalah jumlah dataset dan 500, 500 selanjutnya adalah ukuran gambar yang sudah kita resize sebelumnya ditahapan preproesing. Kita perlu mengubahnya menjadi 2 dimensi agar model Random Forest dapat digunakan. Dengan cara mengubahnya menjadi -1. Seperti tampak pada code dibawah ini.

```
X_train = X_train.reshape(X_train.shape[0], -1)
X_test = X_test.reshape(X_test.shape[0], -1)
print(X_train.shape)
print(X_test.shape)
```

Disini karena data sudah displit sebelumnya maka data X_train sama X_test akan kita ubah keduanya menjadi 2 dimensi. Hasilnya adalah sebagai berikut.

```
(40, 250000)
(10, 250000)
```

Sekarang dimensi X telah menjadi 2 dimensi. Angka 40 dan 10 pertama adalah jumlah data training dan testing, dan 250000 adalah ukurannya. Data inilah yang akan digunakan kedalam model Random Forest.

Tahap 7. Membuat, Melatih dan Menguji Model Random Forest.

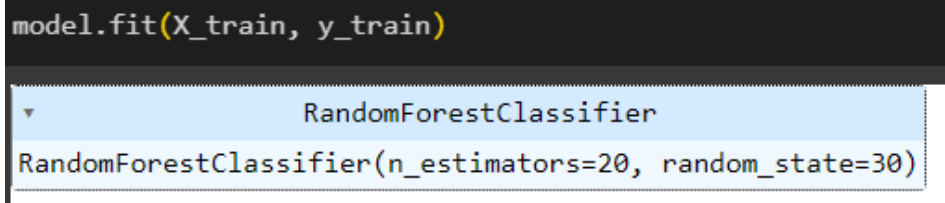
Membuat model Random Forest. Kita dapat menggunakan library sklearn dan mengimport **RandomForestClassifier**. Disini kami menggunakan parameter **n_estimators** dan **random_state**.

- **n_estimators** adalah parameter yang digunakan untuk mengatur jumlah pohon keputusan (decision trees) yang akan dibangun.
- **random_state** adalah untuk mengontrol inisialisasi random (tujuannya agar membuat hasil yang konsisten setiap kali program dijalankan)

```
model = RandomForestClassifier(n_estimators= 20, random_state = 30)
```

Setelah kita sudah memiliki model maka kita dapat melatih modelnya dengan menggunakan code dibawah ini.

```
model.fit(X_train, y_train)
```



Untuk menguji model kita dapat menggunakan predict. Berikut ini adalah code untuk menguji dataset. Dataset yang diprediksi adalah data testing (X_test).

```
y_pred = model.predict(X_test)
```

Untuk melihat hasil prediksinya kita dapat menggunakan code dibawah ini.

```

# Melakukan prediksi pada data uji
y_pred = model.predict(X_test)

# Menentukan jumlah baris dan kolom dalam grid subplot
num_rows = 2
num_cols = 5

imageResized = (500, 500)

# Mengatur ukuran gambar subplot
fig, axes = plt.subplots(nrows=num_rows, ncols=num_cols, figsize=(10, 10))

# Loop untuk menampilkan gambar dengan hasil prediksi
for i, ax in enumerate(axes.flatten()):
    # Mendapatkan gambar dan label yang diprediksi
    test_image = X_test[i].reshape(imageResized[0], -1)
    if(y_pred[i] == 0):
        predicted_label = "Mobil Sedan"
    else:
        predicted_label = "Mobil SUV"

    # Menampilkan gambar dan label
    ax.imshow(test_image, cmap='gray')
    ax.axis('off')
    ax.set_title('Prediksi: ' + predicted_label)

# Menyesuaikan tata letak subplot
plt.tight_layout()
plt.show()

```

Untuk menampilkan gambarnya kembali maka kita kembalikan ke ukuran semula dengan **reshape**.

Berikut adalah hasil dari code tersebut :



Tahap 8. Menghitung Akurasi dan Evaluasi Peforma Model

Akurasi

Berikut adalah code untuk menghitung akurasi dengan menggunakan sklearn.metrics. Untuk memudahkan, maka kami mengubahnya kedalam format persen.

```
accuracy = accuracy_score(y_pred, y_test) * 100  
print("Accuracy: {:.2f}%".format(accuracy))
```

```
Accuracy: 90.00%
```

Akurasi Model yang telah dibuat adalah 90% dari 10 data mobil yang ditesting.

Confussion Matrix

Dari akurasi tersebut, kami menggunakan Confussion Matrix untuk melihat seberapa baik model dapat melakukan klasifikasi pada setiap kelas target. Berikut adalah codenya.


```
# Menghitung confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Menampilkan setiap elemen secara terpisah
TN = cm[0, 0] # True Negative
FP = cm[0, 1] # False Positive
FN = cm[1, 0] # False Negative
TP = cm[1, 1] # True Positive

# Menampilkan hasil
print("True Negative (TN):", TN) # Jumlah mobil sedan yang berhasil diklasifikasikan dengan benar sebagai sedan oleh model.
print("False Positive (FP):", FP) # Jumlah mobil sedan yang salah diklasifikasikan sebagai SUV oleh model.
print("False Negative (FN):", FN) # Jumlah mobil SUV yang salah diklasifikasikan sebagai sedan oleh model.
print("True Positive (TP):", TP) # Jumlah mobil SUV yang berhasil diklasifikasikan dengan benar sebagai SUV oleh model.

True Negative (TN): 5
False Positive (FP): 0
False Negative (FN): 1
True Positive (TP): 4
```

Dari code tersebut dapat dilihat bahwa:

- Jumlah **Mobil Sedan** yang berhasil diklasifikasikan dengan **benar** sebagai **Sedan** oleh model : 5 gambar
- Jumlah **Mobil Sedan** yang **salah** diklasifikasikan sebagai **SUV** oleh model : 0 gambar
- Jumlah **Mobil SUV** yang **salah** diklasifikasikan sebagai **Sedan** oleh model : 1 gambar
- Jumlah **Mobil SUV** yang berhasil diklasifikasikan dengan **benar** sebagai **SUV** oleh model : 4 gambar

Classification Report

Untuk melihat laporan klasifikasinya dapat menggunakan **classification_report** yang sudah dimport sebelumnya. Berikut adalah code dan hasilnya.

```
print(classification_report(y_pred, y_test))
```

	precision	recall	f1-score	support
0	1.00	0.83	0.91	6
1	0.80	1.00	0.89	4
accuracy			0.90	10
macro avg	0.90	0.92	0.90	10
weighted avg	0.92	0.90	0.90	10

Dari hasil laporan tersebut dapat dilihat maka dapat disimpulkan bahwa model memiliki performa yang baik dalam mengklasifikasikan **Mobil Sedan (Label 0)** dengan tingkat presisi yang tinggi, sedangkan dalam mengklasifikasikan **Mobil SUV (Label 1)**, model memiliki tingkat recall yang sempurna.

Tahap 9. Simpan Model

Pada bagian ini, library joblib digunakan untuk menyimpan (import dump) dan memuat (import load) file. Tujuan menyimpan model adalah agar kita tidak perlu melakukan proses membuat model lagi untuk selanjutnya. Berikut ini adalah code untuk menyimpan model.

```
# Menyimpan model ke dalam file
dump(model, 'RFmodel.joblib')

['RFmodel.joblib']
```

Maka model sudah berhasil disimpan dengan nama 'Rfmodel.joblib'

Tahap 10. Load Model untuk Melakukan Prediksi Gambar yang Diupload Dari User.

Untuk memuat model, gunakan library joblib (import load).

```
# Membaca model dari file
model_from_file = load('RFmodel.joblib')
```

Dalam code tersebut, file yang diupload hanya boleh 1 file gambar dan file tersebut harus berekstensi **.jpg**, **.jpeg**, dan **.png** saja. Untuk mengecek ekstensi file maka **imghdr** digunakan. Hal ini dilakukan bertujuan untuk memastikan user memasukkan file berupa gambar.

```
# Membaca model dari file
model_from_file = load('RFmodel.joblib')

# Upload file gambar
uploaded_files = files.upload()

# Memastikan hanya satu file gambar yang diupload
if len(uploaded_files) != 1:
    raise ValueError("Mohon upload satu file gambar saja.")

# Mendapatkan path file gambar yang diupload
image_path = next(iter(uploaded_files))

# Memeriksa ekstensi file yang diupload
allowed_extensions = {'jpg', 'jpeg', 'png'}
file_extension = imghdr.what(image_path)

if file_extension is None or file_extension not in allowed_extensions:
    raise ValueError("File yang diupload bukan merupakan file gambar yang valid.")
```

Setelah diterima maka gambar akan melalui tahapan yang sama seperti dataset kita sebelumnya. Yaitu melalui proses preprocessing dan mengubah dimensi pada gambar menjadi 2 dimensi.

```
# Melakukan preprocessing pada gambar
preprocessed_image = preprocessing(image_path)

# Mengubah dimensi gambar
image_vector = preprocessed_image.reshape(1, -1)
```

Untuk melakukan prediksi dengan menggunakan model yang kita load, maka dapat menggunakan **.predict(gambar yang ingin diprediksi).**

```
# Melakukan prediksi menggunakan model yang telah dimuat
prediction = model_from_file.predict(image_vector)
```

Karena label diberikan 0 dan 1 sebelumnya maka untuk menampilkan hasilnya dapat menggunakan kondisi, jika 0 maka “Mobil Sedan” dan jika 1 maka “Mobil SUV”.

```
if(prediction == 0):
    prediction = "Mobil Sedan"
if(prediction == 1):
    prediction = "Mobil SUV"
```

Terakhir, tampilkan gambar dan hasil prediksinya.

```
# Membaca gambar
image = cv2.imread(image_path)

# Konversi warna BGR ke RGB
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Menampilkan gambar
plt.imshow(image_rgb)
plt.axis('off')
plt.show()
print("\nHasil Prediksi =", prediction)
```

Setelah hasilnya ditampilkan maka filenya akan langsung dihapus dengan menggunakan code dibawah ini.

```
os.remove(image_path)
```

Dari penjelasan tersebut, berikut adalah code lengkapnya

```

# Upload file gambar
uploaded_files = files.upload()

# Memastikan hanya satu file gambar yang diupload
if len(uploaded_files) != 1:
    raise ValueError("Mohon upload satu file gambar saja.")

# Mendapatkan path file gambar yang diupload
image_path = next(iter(uploaded_files))

# Memeriksa ekstensi file yang diupload
allowed_extensions = {'jpg', 'jpeg', 'png'}
file_extension = imghdr.what(image_path)

if file_extension is None or file_extension not in allowed_extensions:
    raise ValueError("File yang diupload bukan merupakan file gambar yang valid.")

# Melakukan preprocessing pada gambar
preprocessed_image = preprocessing(image_path)

# Mengubah dimensi gambar
image_vector = preprocessed_image.reshape(1, -1)

# Melakukan prediksi menggunakan model yang telah dimuat
prediction = model_from_file.predict(image_vector)

if(prediction == 0):
    prediction = "Mobil Sedan"
if(prediction == 1):
    prediction = "Mobil SUV"

# Membaca gambar
image = cv2.imread(image_path)

# Konversi warna BGR ke RGB
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Menampilkan gambar
plt.imshow(image_rgb)
plt.axis('off')
plt.show()
print("\nHasil Prediksi =", prediction)

os.remove(image_path)

```

Berikut adalah hasilnya :

Percobaan Pertama

Disini saya mencoba upload Mobil SUV yang belum pernah dilihat oleh model sebelumnya.

Choose Files 51.jpg

- 51.jpg(image/jpeg) - 102311 bytes, last modified: 4/25/2023 - 100% done
Saving 51.jpg to 51.jpg



Hasil Prediksi = Mobil SUV

Hasilnya model dapat menebak dengan **BENAR**.

Percobaan Kedua

Disini saya mencoba upload Mobil Sedan yang belum pernah dilihat oleh model sebelumnya.

Choose Files GambarSedan1.jpg

• **GambarSedan1.jpg**(image/jpeg) - 82889 bytes, last modified: 7/8/2023 - 100% done
Saving GambarSedan1.jpg to GambarSedan1.jpg



Hasil Prediksi = Mobil Sedan

Hasilnya model dapat menebak dengan **BENAR**.

Untuk percobaan berikutnya dapat menggunakan gambar ini :

Link Gambar untuk Testing : [Link Testing](#)

II. Link Presentasi

Link Video Presentasi : [Video Presentasi](#)

Link Modul Presentasi : [Modul PPT](#)

III. Kesimpulan

Dari pengerjaan tugas 1 dan tugas 2, kami dapat menyimpulkan bahwa dalam tugas 1 bertujuan untuk memahami tentang bagaimana cara melakukan preprocessing terhadap citra. Preprocessing citra bertujuan untuk meningkatkan kualitas citra, mengurangi noise, meningkatkan kontras, deteksi tepi citra, deteksi fitur, dan lain sebagainya. Pemilihan metode preprocessing yang tepat akan membuat algoritma yang kita gunakan nantinya ditugas 2 dapat bekerja lebih baik dan lebih akurat. Sedangkan adanya tugas 2 ini, bertujuan untuk memahami bagaimana cara membangun sebuah model dan membuat model tersebut dapat memprediksi citra yang diinput oleh user. Bagaimana mengatur parameter pada model agar dapat meningkatkan akurasi. Dalam Tugas 2, kami menggunakan Random Forest sebagai algoritmanya. Algoritma Random Forest sendiri adalah algoritma hasil perkembangan dari Algoritma Decision Tree. Algoritma Random Forest akan membuat banyak decision tree yang dimana masing masingnya akan memberikan hasil prediksi. Dari hasil prediksi kemudian akan dipilih melalui divoting. Yang menjadi mayoritas dari voting tersebut akan menjadi hasil prediksinya. Dengan menggunakan Algoritma ini kami memperoleh akurasi sebesar 90%.

Hal itu berarti model sudah dilatih dengan baik. Hal ini tidak terlepas dari preprocessing yang dikerjakan pada Tugas 1.