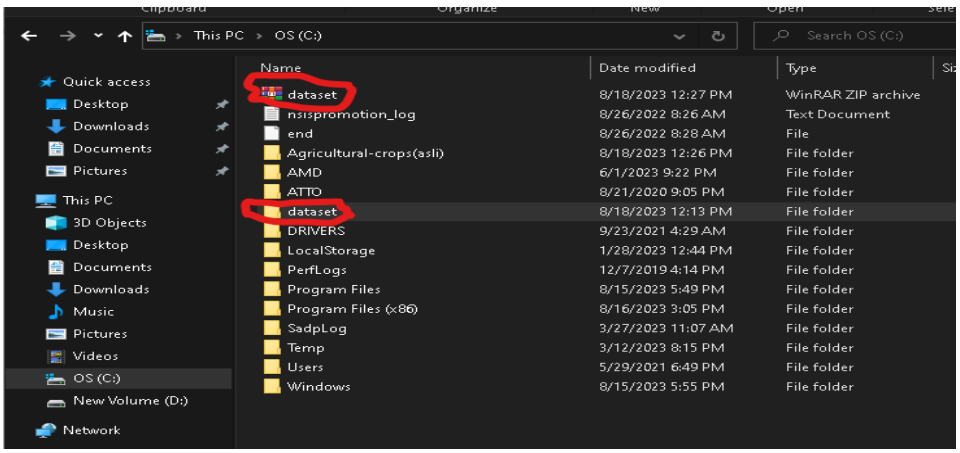
BAB IV
HASIL PENELITIAN DAN PEMBAHASAN

4.1 Hasil Penelitian

Hasil dari penelitian ini adalah sebuah model klasifikasi dengan objek sampel berupa daun dari beberapa tanaman. Dataset tanaman ini didapatkan dari Kaggle dengan jumlah data sebanyak 829 buah yang terbagi menjadi 30 kelas, yaitu sugarcane (tebu), almond, pisang, cardamom, ceri, cabe, clove, kelapa, kopi, kapas, ketimun, kacang, gram, jowar, jute, lemon, maize, mustard-oil, olive-trees, papaya, pearl-millet, nanas, rice, kedelai, sunflower, teh, tobacco-plant, tomato, vigna-radiati dan wheat. Pemrosesan data pada penelitian ini dilakukan dengan menggunakan bahasa *Python* di *Google Colab* (*Google Colaboratory*). Berikut hasil penelitian yang dilakukan berdasarkan alur penelitian pada Gambar 3.1.

a. Start

Pada penelitian ini digunakan dataset yang diambil dari *Kaggle*. Seteah diambil dari Kaggle, dataset berupa citra ini diolah kembali. Awalnya, dataset ini memiliki 30 kelas, namun diolah kembali menjadi 2 kelas saja, yaitu, *sugarcane* dan *others*. Dan dataset yang telah diolah kembali tersebut diubah format menjadi zip.



Gambar 4.1 Dataset Terbaru (dataset.zip)

b. *Input Image*

Setelah melakukan pengolahan dataset secara mandiri, tahapan berikutnya adalah *input dataset* (berupa citra) dengan format .zip (dataset.zip). Cara melakukan *input image* adalah dengan menggunakan coding dibawah ini.

Coding Input Image :

```
#from google.colab import files

# Mengunggah dataset
uploaded = files.upload()

# Nama file yang diunggah
for filename in uploaded.keys():
    print(f"Uploaded file: {filename}")

import zipfile

# Nama file ZIP yang telah diunggah
zip_file_name = 'dataset.zip'

# Ekstrak ZIP
with zipfile.ZipFile(zip_file_name, 'r') as
zip_ref:
    zip_ref.extractall('')  # Ganti 'temp_folder'
dengan folder tujuan ekstraksi
```

Output Coding :

Gambar 4.2 *Output Coding Upload dataset.zip*

Penjelasan :

```
#from google.colab import files
```

Mengimpor modul *files* dari *google.colab*. Modul ini digunakan untuk menangani *file* dalam lingkungan *Google Colab*, seperti mengunggah *file* dari komputer lokal ke *Colab*.

```
# Mengunggah dataset
uploaded = files.upload()
```

Fungsi *files.upload*() membuka antarmuka yang memungkinkan pengguna untuk mengunggah *file* dari komputer ke sesi Colab. Setelah *file* diunggah, hasilnya disimpan dalam variabel *uploaded*, yang merupakan *dictionary*. Kunci dalam *dictionary* adalah nama *file*, dan nilainya adalah isi *file* yang diunggah.

```
# Nama file yang diunggah
for filename in uploaded.keys():
    print(f"Uploaded file: {filename}")
```

Loop ini iterasi melalui semua kunci (nama *file*) dalam *dictionary uploaded* dan mencetak setiap nama *file* yang telah diunggah. Ini memberikan umpan balik kepada pengguna tentang *file* apa saja yang berhasil diunggah.

```
import zipfile
```

Modul *zipfile* digunakan untuk membaca dan menulis *file* ZIP. Dalam kode ini, modul ini akan digunakan untuk mengekstrak isi *file* ZIP yang diunggah.

```
# Nama file ZIP yang telah diunggah
zip_file_name = 'dataset.zip'
```

Variabel *zip_file_name* diisi dengan nama file ZIP yang akan diekstrak. Pastikan nama ini sesuai dengan *file* yang telah diunggah sebelumnya.

```
# Ekstrak ZIP
with zipfile.ZipFile(zip_file_name, 'r') as
zip_ref:
    zip_ref.extractall('')  # Ganti 'temp_folder'
dengan folder tujuan ekstraksi
```

***with zipfile.ZipFile(zip_file_name, 'r') as zip_ref* :** Baris ini membuka *file* ZIP dalam mode baca ('r'). Dengan menggunakan with, file akan otomatis ditutup setelah blok selesai dieksekusi, menjaga manajemen sumber daya yang baik.

zip_ref.extractall('') : Fungsi ini mengekstrak semua isi dari *file* ZIP ke direktori saat ini (direktori kerja). Jika ingin mengekstrak ke folder tertentu, kamu bisa mengganti '' dengan nama folder tujuan, seperti *'temp_folder'*.

c. *Preprocessing*

Sebelum melakukan proses *preprocessing*, hal pertama yang perlu dilakukan adalah melakukan *import library* yang diperlukan pada model yang akan dibuat. Berikut ini adalah library yang digunakan untuk model pada penelitian ini.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import
StratifiedKFold
from skimage import data, color, feature
from skimage.feature import hog
from sklearn.preprocessing import LabelEncoder
from sklearn import svm
from PIL import Image
from sklearn.model_selection import
train_test_split
from sklearn.model_selection import
cross_val_score, cross_val_predict
```

```
from sklearn.metrics import classification_report,
accuracy_score, confusion_matrix
from skimage.transform import resize
from skimage.io import imread
from skimage.io import imshow
from skimage.color import rgb2gray
import cv2
from sklearn.model_selection import
StratifiedKFold, GridSearchCV
```

Penjelasan :

Pandas (*import pandas as pd*)

Digunakan untuk manipulasi dan analisis data. Sangat berguna untuk bekerja dengan data dalam bentuk tabel (*DataFrame*).

NumPy (*import numpy as np*)

Digunakan untuk operasi numerik dan manipulasi *array* multidimensi. Sering digunakan bersamaan dengan Pandas.

Matplotlib (*import matplotlib.pyplot as plt*)

Digunakan untuk membuat visualisasi data. Dengan *pyplot*, kamu dapat membuat grafik, histogram, dan visualisasi lainnya.

OS (*import os*)

Digunakan untuk berinteraksi dengan sistem *file*, seperti mengakses direktori dan mengelola *file*.

Sklearn Preprocessing

StandardScaler (*from sklearn.preprocessing import StandardScaler*)

Digunakan untuk normalisasi data, yaitu mengubah fitur sehingga memiliki mean 0 dan deviasi standar 1.

LabelEncoder (*from sklearn.preprocessing import LabelEncoder*)

Mengonversi label kategorikal menjadi format numerik yang dapat diproses oleh model.

Sklearn Model Selection

StratifiedKFold (*from sklearn.model_selection import StratifiedKFold*)
Membagi data menjadi beberapa subset (*fold*) untuk validasi silang, dengan menjaga proporsi label di setiap *fold*.
GridSearchCV (from sklearn.model_selection import StratifiedKFold, GridSearchCV)
Digunakan untuk mencari parameter terbaik dari model dengan menggunakan validasi silang.
Train_test_split (*from sklearn.model_selection import train_test_split*)
Memisahkan dataset menjadi data latih dan data uji.
Cross_val_score dan **cross_val_predict** (from sklearn.model_selection import cross_val_score, cross_val_predict)
Digunakan untuk melakukan validasi silang pada model.
Skimage : Digunakan untuk pengolahan citra.
data, color, feature (*from skimage import data, color, feature*)
Mengimpor fungsi untuk bekerja dengan dataset gambar, konversi warna, dan ekstraksi fitur gambar.
HOG (from skimage.feature import hog)
Histogram of Oriented Gradients, digunakan untuk ekstraksi fitur pada gambar.
Resize (from skimage.transform import resize)
Digunakan untuk mengubah ukuran gambar.
imread, **imshow** (*from skimage.io import imread* dan *from skimage.io import imshow*)
Membaca dan menampilkan gambar.
rgb2gray (*from skimage.color import rgb2gray*)
Mengonversi gambar berwarna menjadi skala abu-abu.

OpenCV (`import cv2`)
Digunakan untuk pemrosesan gambar dan video, termasuk deteksi objek, pengenalan, dan transformasi gambar.
PIL Image (`from PIL import Image`)
Digunakan untuk membuka, memanipulasi, dan menyimpan gambar dalam berbagai format.

Kemudian hal kedua yang perlu dilakukan adalah melihat isi directory untuk memastikan bahwa dataset telah diekstrak dengan benar. Hal ini dilakukan dengan coding seperti dibawah ini.

Coding Cek Direktori :

```
base_dir = '/content/dataset/dataset'
print(os.listdir(base_dir))
```

Hasil Output :

```
['others', 'sugarcane']
```

Gambar 4.3 Output Cek Direktori

Penjelasan :

`base_dir = '/content/dataset/dataset'`

Baris ini mengatur variabel *base_dir* sebagai *path* (jalur) menuju direktori yang berisi dataset, yaitu *'/content/dataset/dataset'*. Jalur ini biasanya digunakan di lingkungan *Google Colab* dimana *file* telah diunggah atau disimpan.

`print(os.listdir(base_dir))`

Fungsi *os.listdir()* digunakan untuk mengembalikan daftar berisi nama-nama *file* dan *folder* yang ada didalam direktori yang ditentukan oleh *base_dir*. *print()* akan menampilkan daftar nama *file* dan *folder* tersebut di *output*.

Dan hal ketiga yang perlu dilakukan adalah mengetahui daftar kelas atau label gambar dari direktori, dengan menggunakan coding seperti dibawah ini.

Coding Cek Kelas :

```
from tensorflow.keras.preprocessing.image import
ImageDataGenerator

# Definisikan generator
data_generator = ImageDataGenerator(base_dir)  #
Isi dengan parameter yang sesuai

# Muat data dari direktori
train_data = data_generator.flow_from_directory(
    base_dir,
    target_size=(64, 128),
    #batch_size=batch_size,
    class_mode='categorical'  # atau 'binary'
tergantung pada kasus Anda
)

# Menampilkan daftar kelas atau label gambar
class_indices = train_data.class_indices
print("Daftar kelas:", class_indices)
```

Hasil *Output* :

```
Found 150 images belonging to 2 classes.
Daftar kelas: {'others': 0, 'sugarcane': 1}
```

Gambar 4.4 Output Indikasi Kelas

Penjelasan :

```
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
```

Mengimpor ImageDataGenerator, sebuah kelas dari Keras yang digunakan untuk menghasilkan data gambar yang siap digunakan untuk pelatihan

model, dengan atau tanpa augmentasi gambar seperti rotasi, flipping, dan lainnya.

```
# Definisikan generator
data_generator = ImageDataGenerator(base_dir)  # Isi dengan parameter yang sesuai
```

Membuat instance ImageDataGenerator yang akan digunakan untuk memproses gambar. Biasanya, ImageDataGenerator membutuhkan parameter untuk augmentasi seperti rescale, rotation_range, width_shift_range, height_shift_range, dll. Namun, di sini, argumen base_dir langsung dimasukkan secara salah. Seharusnya parameter seperti rescale atau augmentasi lain yang dimasukkan di sini, bukan path dataset.

```
# Muat data dari direktori
train_data = data_generator.flow_from_directory(
    base_dir,
    target_size=(64, 128),
    #batch_size=batch_size,
    class_mode='categorical'  # atau 'binary' tergantung pada kasus Anda
)
```

Fungsi *flow_from_directory* digunakan untuk memuat data gambar langsung dari direktori dengan struktur tertentu. Fungsi ini mencari gambar dalam *subfolder*, dan setiap *subfolder* diidentifikasi sebagai kelas yang berbeda.

base_dir : Lokasi direktori utama yang berisi *subfolder* gambar. *Subfolder* akan dianggap sebagai label kelas.

***target_size*=(64,128)** : Mengubah ukuran gambar menjadi 64x128 piksel saat dimuat. Ukuran ini perlu disesuaikan dengan arsitektur model yang akan digunakan.

batch_size=batch_size : Jumlah gambar yang akan diproses dalam satu batch. Parameter ini masih dikomentari; perlu didefinisikan sesuai kebutuhan.

Class_mode='categorical' : Menentukan mode klasifikasi; *'categorical'* digunakan untuk multi-kelas (lebih dari dua kelas). Jika masalahnya adalah klasifikasi biner, gunakan *'binary'*.

```
# Menampilkan daftar kelas atau label gambar
class_indices = train_data.class_indices
```

Mengambil dictionary yang berisi label kelas dan indeks numeriknya yang digunakan oleh generator. Ini berguna untuk memahami bagaimana label ditetapkan ke kelas secara internal.

```
print("Daftar kelas:", class_indices)
```

Menampilkan daftar kelas dan indeksnya, contoh *outputnya* bisa berupa: {'class1': 0, 'class2': 1, ...}.

Setelah selesai melakukan 3 hal diatas, sekarang gambar sudah dapat diproses (*Image Processing*). Pada codingan ini, gambar diproses dengan melakukan pengecekan (apakah data gambar termasuk RGB atau bukan?), perubahan ukuran (*resize*), dan perubahan gambar dari RGB menjadi *gray*. Berikut penulisan *coding* pada proses *image processing*.

Coding Image Processing :

```
Categories = {'others': 0, 'sugarcane': 1}


# Process the data
data = []
target = []

for category in os.listdir(base_dir):
    if category in Categories:
        class_idx = Categories[category]
        path = os.path.join(base_dir, category)
        for img in os.listdir(path):
            try:
                img_array =
imread(os.path.join(path, img))

                # Periksa format gambar
                img_pil =
Image.open(os.path.join(path, img))
                if img_array.shape[2] == 3 and
img_pil.mode == 'RGB':
```

```
                    img_resized =
resize(img_array, (64, 128, 3))
                    img_gray =
rgb2gray(img_resized)
```

Penjelasan :

```
Categories = {'others': 0, 'sugarcane': 1}
```

Membuat *dictionary Categories* yang memetakan nama kategori menjadi label numerik. Dalam hal ini, '*others*' diberi label 0, dan '*sugarcane*' diberi label 1.

```
data = []
target = []
```

Dua list kosong, data digunakan untuk menyimpan data gambar yang diproses, dan target digunakan untuk menyimpan label kategori dari setiap gambar.

```
for category in os.listdir(base_dir):
```

Memulai iterasi melalui setiap *subfolder* (kategori) yang ada di dalam direktori *base_dir*.

```
if category in Categories:
```

Memeriksa apakah kategori yang ditemukan di dalam direktori ada di *dictionary Categories*. Jika ya, proses lanjut ke pemrosesan gambar dalam kategori tersebut.

```
class_idx = Categories[category]
```

Mengambil label numerik dari kategori yang sedang diproses. Nilai label diambil dari *dictionary Categories*.

```
path = os.path.join(base_dir, category)
```

Membuat jalur lengkap ke direktori kategori saat ini dengan menggabungkan *base_dir* dan nama kategori menggunakan *os.path.join()*.

```
for img in os.listdir(path):
```

Melakukan iterasi melalui setiap *file* gambar di dalam direktori kategori saat ini.

```
try:
    img_array = imread(os.path.join(path, img))
```

Membaca gambar menggunakan fungsi *imread* dari *skimage*. Jika terjadi kesalahan, seperti gambar yang rusak atau tidak bisa dibaca, blok *try* akan menangkap *error* tersebut.

```
img_pil = Image.open(os.path.join(path, img))
```

Membuka gambar yang sama menggunakan *library* PIL (*Python Imaging Library*), yang menyediakan cara berbeda untuk memeriksa format gambar.

```
if img_array.shape[2] == 3 and img_pil.mode == 'RGB':
```

Memeriksa apakah gambar adalah gambar berwarna (RGB). Gambar RGB memiliki 3 channel (merah, hijau, biru), sehingga shape[2] == 3 memastikan gambar tidak berupa gambar grayscale atau format lain. Selain itu, mode gambar dalam PIL juga diperiksa untuk memastikan gambar dalam mode 'RGB'.

```
img_resized = resize(img_array, (64, 128, 3))
```

Mengubah ukuran gambar menjadi dimensi (64, 128, 3) menggunakan fungsi resize dari skimage. Ini memastikan semua gambar memiliki ukuran yang konsisten untuk digunakan dalam model pembelajaran mesin. Dimensi (64, 128) adalah ukuran gambar, dan 3 adalah jumlah channel (RGB).

```
img_gray = rgb2gray(img_resized)
```

Mengonversi gambar RGB yang telah diubah ukurannya menjadi skala abu-abu menggunakan fungsi rgb2gray dari skimage. Meskipun gambar diubah menjadi grayscale, proses konversi gambar berwarna ini bisa bermanfaat untuk beberapa aplikasi seperti deteksi objek berbasis intensitas.

d. HOG Extraction Features

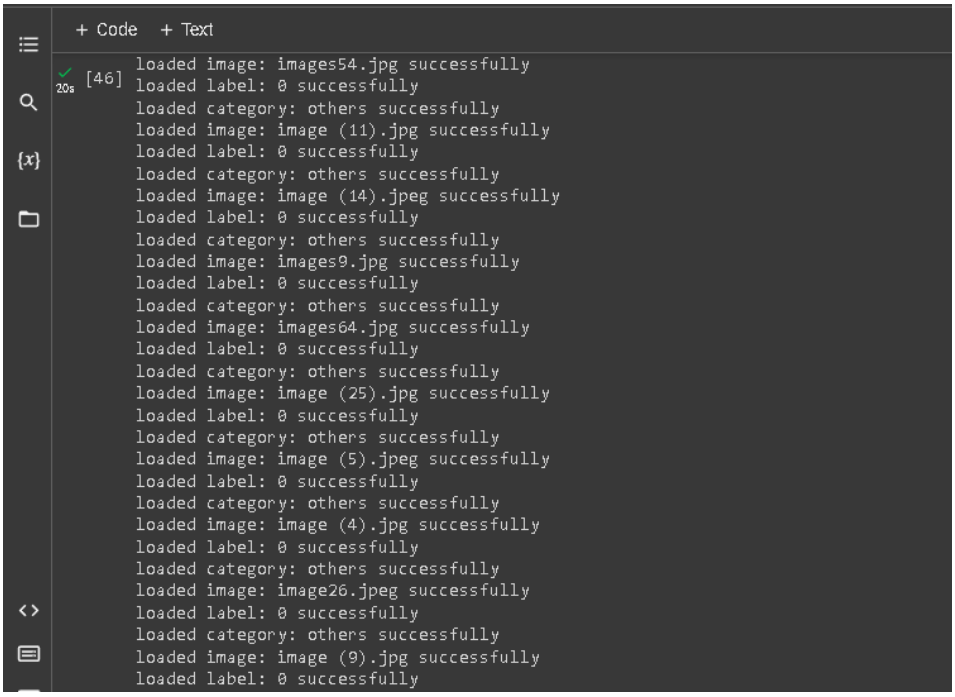
Setelah dilakukan image processing, tahap berikutnya adalah melakukan ekstraksi fitur dengan Histogram of Oriented Gradients (HOG). Langkah pertama pada HOG adalah melakukan konversi citra dari RGB ke gray, dan langkah ini telah dilakukan pada coding Gambar 4.8. Setelah melakukan konversi citra RGB ke gray, selanjutnya adalah penentuan jumlah bin, pixel

per cell, dan cells per block (orientations = 9, pixels_per_cell = (8,8), cells_per_block = (2,2)).

Coding HOG :

```
                    fd = hog(img_gray,
orientations=9, pixels_per_cell=(8, 8),
cells_per_block=(2, 2), visualize=False)
                    data.append(fd)
                    target.append(class_idx)
                    print(f'loaded label:
{class_idx} successfully')
                    print(f'loaded category:
{category} successfully')
                    print(f'loaded image: {img}
successfully')
                else:
                    print(f'Skipped image (not RGB):
{img}')
            except Exception as e:
                print(f'Error loading image:
{img}. Error message: {e}')

# Buat DataFrame dari data dan target
df = pd.DataFrame({'data': data, 'target':
target})

# Pisahkan fitur dan target
x = np.array(df['data'].tolist())# Konversi list
of arrays ke array numpy
y = np.array(df['target'])

# Cetak bentuk dari x dan y
print("Shape of x:", x.shape)
print("Shape of y:", y.shape)
```

Hasil *Output* :



Gambar 4.5 *Output Image Processing and HOG*

Penjelasan :

```
fd = hog(img_gray, orientations=9,
pixels_per_cell=(8, 8), cells_per_block=(2, 2),
visualize=False)
```

`hog()`

Fungsi ini digunakan untuk mengekstraksi fitur ***Histogram of Oriented Gradients* (HOG)** dari gambar. HOG adalah metode untuk mendeteksi tepi dan arah gradien dalam sebuah gambar, yang sering digunakan dalam pengenalan pola dan deteksi objek.

Parameter *`img_gray`*

Input gambar grayscale yang telah dikonversi sebelumnya.

Parameter *`orientations=9`*

Menentukan jumlah bin (*buckets*) untuk arah gradien (orientasi sudut). Nilai 9 berarti membagi 180 derajat menjadi 9 bagian (20 derajat per bin).

Parameter *`pixels_per_cell=(8, 8)`*

Gambar dibagi menjadi sel-sel kecil (8x8 piksel), dan dalam setiap sel, histogram gradien dihitung.

Parameter *cells_per_block=(2, 2)*

Blok berukuran 2x2 sel (dari sel 8x8 piksel) digunakan untuk normalisasi.

Parameter *visualize=False*

Visualisasi HOG tidak diperlukan, jadi pengaturan ini dinonaktifkan.

Output : fd

vektor fitur yang merepresentasikan HOG dari gambar yang diproses.

`data.append(fd)`

Menambahkan vektor fitur fd yang dihasilkan oleh HOG ke dalam list data. Vektor fitur ini akan digunakan sebagai *input* untuk model pembelajaran mesin.

`target.append(class_idx)`

Menambahkan label kelas (kategori) yang sesuai dengan gambar yang diproses ke dalam *list* target.

```
print(f'loaded label: {class_idx} successfully')
```

Menampilkan pesan yang menandakan bahwa label kelas berhasil dimuat.

```
print(f'loaded category: {category} successfully')
```

Menampilkan pesan bahwa kategori gambar saat ini berhasil dimuat.

```
print(f'loaded image: {img} successfully')
```

Menampilkan pesan bahwa gambar spesifik (dengan nama img) berhasil dimuat dan diproses.

```
else:
     print(f'Skipped image (not RGB): {img}')
```

Jika gambar tidak dalam format RGB, gambar tersebut dilewati, dan pesan ditampilkan bahwa gambar itu dilewati.

```
except Exception as e:
       print(f'Error loading image: {img}. Error
message: {e}')
```

Blok ***try-except*** menangkap error selama pemrosesan gambar. Jika ada kesalahan saat memuat gambar, pesan kesalahan akan ditampilkan.

```
df = pd.DataFrame({'data': data, 'target':
target})
```

Setelah seluruh gambar diproses, data (vektor fitur HOG) dan target (label kelas) dikonversi menjadi sebuah **DataFrame** menggunakan Pandas. **df** adalah DataFrame yang berisi dua kolom, yaitu **data** (Menyimpan vektor fitur HOG dari setiap gambar) dan **target** (Menyimpan label kelas dari setiap gambar)

e. SVM Classification

Tahap kelima pada penelitian ini, setelah melakukan ekstrasi fitur HOG yaitu melakukan klasifikasi dengan algoritma *Support Vector Machine* (SVM). Namun sebelum melakukan klasifikasi, satu hal yang penting dilakukan yaitu, pembagian data (Data Splitting). Data Splitting adalah membagi dataset menjadi set pelatihan (training set) dan set pengujian (testing set). Tujuan dari proses ini adalah untuk memastikan bahwa model yang dibangun tidak hanya "menghafal" data yang telah dilihatnya, tetapi juga mampu melakukan generalisasi pada data yang belum pernah dilihat sebelumnya. Berikut adalah penulisan coding dari tahapan ini.

Coding Data Splitting :

```
(x_train, x_test, y_train, y_test) =
train_test_split(x, y, test_size=0.2, random_state=0)
```

Setelah dilakukan pembagian data, ada satu proses lagi yang perlu dilakukan, yaitu normalisasi fitur. Pada codingan ini, normalisasi fitur dilakukan menggunakan StandardScaler dari Scikit-learn.

Coding Normalisasi Fitur :

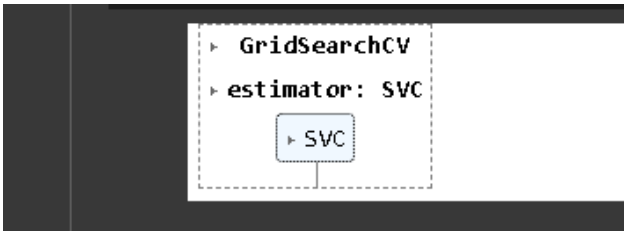
```
# Normalisasi fitur
scaler = StandardScaler()
```

```
x_train_scaled = scaler.fit_transform(x_train)
x_test_scaled = scaler.transform(x_test)
```

Selanjutnya, setelah melalui 2 proses diatas (Data Splitting dan Normalisasi Fitur), data gambar akhirnya melakukan klasifikasi *Support Vector Machine* (SVM) dengan kernel linear dan penyetelan parameter menggunakan GridSearchCV. Klasifikasi SVM dilakukan dengan coding seperti dibawah ini.

Coding Klasifikasi :

```
# Inisialisasi model SVM
regularized_svc = svm.SVC(kernel='linear')

# Penyetelan parameter menggunakan GridSearchCV
param_grid = {'C': [0.001, 0.01, 0.1, 1, 10]}
              #'gamma': [0.01, 0.1, 1, 10]
grid_search = GridSearchCV(regularized_svc ,
param_grid, cv=StratifiedKFold(n_splits=10,
shuffle=True, random_state=42))
grid_search.fit(x_train_scaled, y_train.ravel())
```



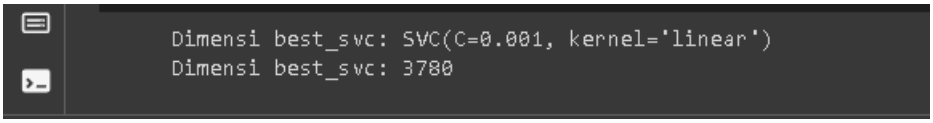
Gambar 4.6 *Setting* Parameter SVM

```
# Hasil penelusuran parameter
print("Best parameters found:",
grid_search.best_params_)
```

```
Best parameters found: {'C': 0.001}
```

Gambar 4.6 *Output* Parameter SVM

```
# Evaluasi skor menggunakan model terbaik
best_svc = grid_search.best_estimator_
print("Dimensi best_svc:", best_svc)
print("Dimensi best_svc:", best_svc.n_features_in_)
```



Gambar 4.7 *Output* Dimensi SVM Terbaik

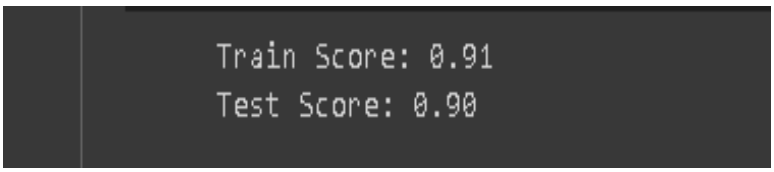
f. Akurasi Model (Cross Validation)

Pada coding penelitian ini, akurasi dan prediksi model dilakukan dengan menggunakan validasi silang (cross validation). Nilai atau score yang dihasilkan oleh validasi silang dapat berupa akurasi, presisi, recall, dan f-1 score. Dengan demikian, maka cross validation (cv) dapat memberikan gambaran yang lebih menyeluruh tentang seberapa baik model yang dibuat dapat bekerja. Keakurasian tersebut dapat dilakukan dengan menggunakan coding dibawah ini.

Coding Akurasi Model 1 :

```
cv_train_score = cross_val_score(best_svc,
x_train_scaled, y_train.ravel(),
cv=StratifiedKFold(n_splits=5, shuffle=True,
random_state=1))
print("Train Score: %.2f" % cv_train_score.mean())

cv_test_score = cross_val_score(best_svc,
x_test_scaled, y_test.ravel(),
cv=StratifiedKFold(n_splits=2, shuffle=True,
random_state=1))
print("Test Score: %.2f" % cv_test_score.mean())
```

Hasil Output :



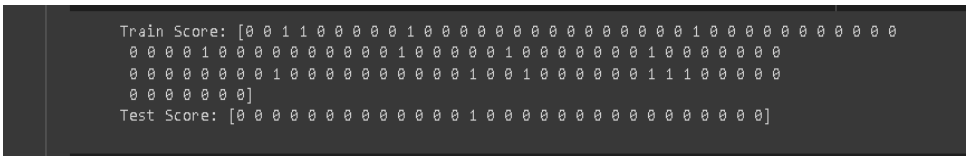
Gambar 4.8 *Output* Akurasi Model 1

Coding Akurasi Model 2 :

```
cv_train_predict = cross_val_predict(best_svc,
x_train_scaled, y_train.ravel(),
cv=StratifiedKFold(n_splits=5, shuffle=True,
random_state=1))
```

```
print("Train Score:", cv_train_predict)

cv_test_predict = cross_val_predict(best_svc,
x_test_scaled, y_test.ravel(),
cv=StratifiedKFold(n_splits=2, shuffle=True,
random_state=1))
print("Test Score:", cv_test_predict)
```

Hasil Output :



Gambar 4.9 *Output* Akurasi Model 2

g. Testing Model

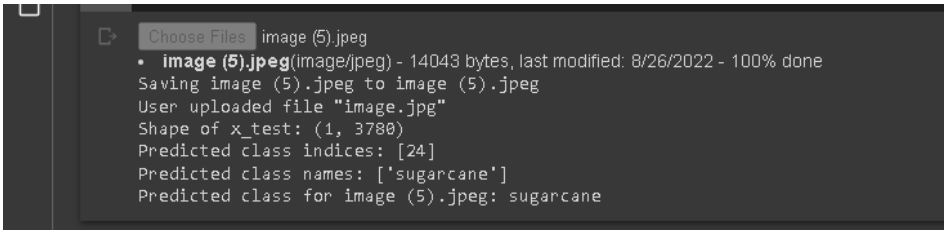
Berikut coding testing model.

```
from google.colab import files
from skimage.io import imread
from skimage.transform import resize
import os
import io

Categories = {'others': 0, 'sugarcane': 1}

# Definisikan direktori data uji
uploaded = files.upload()

# Misalnya, jika Anda mengunggah file "image.jpg",
maka:
test_data_filename = 'image.jpg'

print('User uploaded file
"{}"'.format(test_data_filename))

# Inisialisasi variabel untuk data uji
x_test = []

# Loop melalui data gambar yang diunggah
for img_data in uploaded.values():
    try:
        img_array = imread(io.BytesIO(img_data))

        if img_array.shape[2] == 3:  # Gambar RGB
            img_resized = resize(img_array, (64, 128,
3))
            img_gray = rgb2gray(img_resized)
            fd = hog(img_gray, orientations=9,
pixels_per_cell=(8, 8), cells_per_block=(2, 2),
visualize=False)
```

```
            if len(fd.shape) > 1:
                fd = fd.flatten()
                print("Bentuk dari fd:", fd.shape)

            x_test.append(fd)
        else:
            print("Skipped image (not RGB)")
    except Exception as e:
        print(f'Error loading image. Error message:
{e}')

# Konversi ke array numpy
x_test = np.array(x_test)

# Print the shape of x_test for debugging purposes
print("Shape of x_test:", x_test.shape)

if x_test.shape[0] > 0:  # Periksa apakah ada data
untuk diuji
    # Normalisasi data uji
    x_test_scaled = scaler.transform(x_test)  #
Menggunakan scaler yang telah didefinisikan sebelumnya

    # Memprediksi kelas data uji
    predicted_class_indices =
best_svc.predict(x_test_scaled)

    print("Predicted class indices:",
predicted_class_indices)  # Cetak prediksi indeks
kelas

    # Konversi indeks yang diprediksi menjadi nama
kelas
    predicted_class_names = [key for key, value in
class_indices.items() if value ==
predicted_class_indices[0]]

    print("Predicted class names:",
predicted_class_names)  # Cetak prediksi nama kelas

    # Cetak hasil prediksi untuk setiap gambar yang
diunggah
    for img_file, predicted_class_name in
zip(uploaded.keys(), predicted_class_names):
        print(f"Predicted class for {img_file}:
{predicted_class_name}")
else:
    print("No valid images to predict.")
```

h. Output

Dan berikut ini merupakan hasil dari testing model.



Gambar 4.10 *Output Testing Model*

i. Finish

Jika dilihat dari hasil *output testing model*, testing model berhasil melakukan klasifikasi objek berupa citra yang sesuai dengan kelas atau labelnya dengan hasil validasi silang sebesar 0, 91 (data latih) dan 0,90 (data uji)

4.2 Pembahasan

Model klasifikasi objek daun tebu dengan dataset tanaman pertanian sebanyak 829 data yang didapatkan dari Kaggle, diolah kembali sehingga menjadi 2 kelas saja, yaitu *sugarcane* atau bukan yang dibangun dengan ekstraksi fitur HOG dan algoritma SVM ini menghasilkan nilai validasi silang yaitu sebesar 0,91 % (data latih) dan 0,90 % (data uji). Pada saat testing model, model ini berhasil melakukan klasifikasi tebu dan sesuai dengan kelas atau labelnya, sehingga model ini dapat dijadikan acuan sebagai model klasifikasi yang menghasilkan output klasifikasi yang akurat dan dengan waktu yang singkat.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Pada penelitian ini dibangun sebuah model yang digunakan untuk melakukan klasifikasi objek daun tebu dari dataset tanaman pertanian sebanyak 829 data yang diambil dari Kaggle. Awalnya, dataset ini memiliki 30 kelas, namun diolah kembali menjadi 2 kelas saja, yaitu, *sugarcane* dan bukan. Klasifikasi dilakukan dengan menggunakan HOG sebagai ekstraksi fitur dan SVM sebagai *classifier*. Ekstraksi fitur HOG dilakukan pada orientasi bin = 9, pixels per cell = 8,8 dan cells per block = 2,2. Namun sebelum dilakukan ekstraksi fitur, citra telah melakukan proses resize menjadi ukuran 64 x 128 dan koversi citra dari RGB ke *gray*. Kemudian setelah dilakukan ekstraksi fitur, citra selanjutnya melakukan proses klasifikasi SVM. Klasifikasi SVM dilakukan dengan memberikan parameter terbaik menggunakan GridSearchCV (kernel = liniear dan C = 0,001). Namun sebelum dilakukan klasifikasi dengan SVM, sudah terlebih dulu dilakukan proses splitting dan normalisasi menggunakan StandardScaler. Model ini sudah dilakukan pengujian akurasi menggunakan cross validation, hasilnya sebesar 0,91 (data latih) dan 0,90 (data uji). Pada pengujian model (testing model), model berhasil melakukan klasifikasi berdasarkan kelas atau labelnya.

5.2 Saran

Model yang telah dibangun ini dapat dijadikan acuan sebagai model klasifikasi yang akurat dan cepat. Saran untuk pengembang penelitian ini selanjutnya adalah dengan mencoba menggunakan, membandingkan dan menggabungkan dengan metode lainnya.

DAFTAR PUSTAKA

[1] V. A. Dihni, “10 Negara Produsen Tebu Terbesar di Dunia.”

[2] M. A. Rizaty, “5 Provinsi Penghasil Tebu Terbesar Nasional (2021).”

[3] J. T. Geomatika, F. Teknik, and I. T. Sepuluh, “Menggunakan Metode Klasifikasi Berbasis Obyek,” no. October, 2016.

[4] M. Ichwan, I. A. Dewi, and Z. M. S, “Klasifikasi Support Vector Machine (SVM) Untuk Menentukan TingkatKemanisan Mangga Berdasarkan Fitur Warna,” *MIND J.*, vol. 3, no. 2, pp. 16–23, 2019, doi: 10.26760/mindjournal.v3i2.16-23.

[5] Y. Prabowo and K. N. Nasahara, “Detecting and Counting Coconut Trees in Pleiades Satellite Imagery Using Histogram of Oriented Gradients and Support Vector Machine,” *Int. J. Remote Sens. Earth Sci.*, vol. 16, no. 1, p. 87, 2019, doi: 10.30536/j.ijreses.2019.v16.a3089.

[6] Y. Zheng *et al.*, “Development of a Phenology-Based Method for Identifying Sugarcane Plantation Areas in China Using High-Resolution Satellite Datasets,” *Remote Sens.*, vol. 14, no. 5, 2022, doi: 10.3390/rs14051274.

[7] A. Nurhidayat, A. Asmunin, and D. F. Suyatno, “Prediksi Kinerja Akademik Mahasiswa Menggunakan Machine Learning dengan Sequential Minimal Optimization untuk Pengelola Program Studi,” *J. Inf. Eng. Educ. Technol.*, vol. 5, no. 2, pp. 84–91, 2021, doi: 10.26740/jieet.v5n2.p84-91.

[8] “Kanker Payudara Invasive,” Binus, 2020.

[9] A. I. Kurnia, M. T. Furqon, and B. Rahayudi, “Klasifikasi Kualitas Susu Sapi Menggunakan Algoritme Support Vector Machine (SVM) (Studi Kasus: Perbandingan Fungsi Kernel Linier dan RBF Gaussian),” *J. Pengemb. Teknol. Inf. dan Ilmu Komput.*, vol. 2, no. 11, pp. 4453–4461, 2018.

[10] R. Y. Endra, A. Cucus, F. N. Afandi, and M. B. Syahputra, “Deteksi Objek Menggunakan Histogram of Oriented Gradient (Hog) Untuk Model Smart Room,” *Explor. J. Sist. Inf. dan Telemat.*, vol. 9, no. 2, 2018, doi: 10.36448/jsit.v9i2.1075.

[11] H. Azis, P. Purnawansyah, F. Fattah, and I. P. Putri, “Performa Klasifikasi