LAPORAN AKHIR PROJEK
PENGOLAHAN CITRA DIGITAL

Oleh:

Martogi Jekson C. Siagian 2105541054
A. Jagad Miftahul Rizqy 2105541056

PROGRAM STUDI TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS UDAYANA

2023

KLASIFIKASI KESEGARAN BUAH TOMAT BERDASARKAN CITRA MENGGUNAKAN MODEL “DenseNet201 & Model VGG16”

BAB I
PENDAHULUAN

1.1 Latar Belakang

Buah tomat merupakan salah satu buah yang sangat populer dan sering digunakan dalam berbagai masakan dan produk makanan. Kualitas tomat sangat bergantung pada tingkat kesegarannya, yang dapat mempengaruhi rasa, tekstur, dan nilai nutrisinya. Penentuan kesegaran buah tomat secara manual melalui pemeriksaan visual dapat memakan waktu, tidak konsisten, dan rentan terhadap kesalahan manusia. Kualitas buah tomat memiliki peran yang signifikan dalam mempengaruhi preferensi konsumen dan keberhasilan industri pertanian serta pasar produk hortikultura. Penilaian kesegaran buah tomat secara manual sering kali melibatkan penilaian visual yang melihat aspek warna, tekstur, dan bentuk buah. Namun, pendekatan ini rentan terhadap penilaian yang tidak konsisten, subjektif, dan dapat memakan waktu, terutama pada skala produksi yang besar.

Oleh karena itu, penggunaan teknologi yang memanfaatkan kecerdasan buatan dan analisis citra dapat menjadi solusi yang menjanjikan. Metode klasifikasi kesegaran buah tomat berdasarkan citra menggunakan model jaringan saraf tiruan seperti "DenseNet201" dan "VGG16" telah menunjukkan kemampuan yang baik dalam pengenalan pola pada objek dalam citra. Dengan menggunakan teknologi citra dan kecerdasan buatan, seperti model "DenseNet201" dan "VGG16", dapat dikembangkan suatu sistem yang mampu secara otomatis mengklasifikasikan tingkat kesegaran buah tomat berdasarkan fitur-fitur visual yang terdapat pada citra tomat. Model-model tersebut telah terbukti efektif dalam pengenalan pola dan klasifikasi objek pada citra.

Penggunaan model "DenseNet201" dan "VGG16" dalam penelitian ini diharapkan dapat memberikan akurasi yang tinggi dalam mengenali ciri-ciri visual yang menandakan kesegaran buah tomat, sehingga sistem yang dikembangkan dapat membantu dalam proses penilaian kualitas buah tomat

secara cepat, otomatis, dan lebih akurat. Hal ini dapat memberikan manfaat yang signifikan dalam industri pertanian, distribusi, dan penjualan buah tomat serta produk turunannya.

1.2 Rumusan Masalah

Menurut pembahasan latar belakang masalah sebelumnya, maka dapat dirumuskan beberapa hal yang menjadi masalah sebagai berikut:

1. Bagaimana cara memanfaatkan model "DenseNet201" dan "VGG16" untuk mengidentifikasi ciri visual yang menandakan tingkat kesegaran buah tomat dari citra?
2. Bagaimana pelatihan dan pengujian untuk klasifikasi Kesegaran Buah Tomat dengan metode “DenseNet201 & Model VGG16”?
3. Bagaimana performa dan perbandingan antara model "DenseNet201" dan "VGG16" dalam mengklasifikasikan kesegaran buah tomat berdasarkan citra?

1.3 Tujuan Penelitian

Tujuan yang ingin dicapai dalam pembuatan projek ini, yaitu:

1. Merancang arsitektur "DenseNet201" dan "VGG16" untuk klasifikasi Kesegaran Buah Tomat.
2. Membangun sistem otomatis untuk mengklasifikasikan tingkat kesegaran buah tomat secara cepat dan akurat..
3. Mengidentifikasi ciri visual yang menandakan kesegaran buah tomat melalui model-model jaringan saraf.

BAB II
TINJAUAN PUSTAKA

2.1 Alat dan Bahan

1. Jupyter Notebook
2. Bahasa pemrograman Python
3. Library os, PIL, shutil, tensorflow, keras, sklearn, numpy, seaborn, pandas dan matplotlib.

2.2 Dasar Teori

1. Pengolahan Citra Digital

Pengolahan citra digital adalah bidang yang berkaitan dengan manipulasi gambar atau video menggunakan algoritma komputer. Tujuan utamanya adalah untuk meningkatkan citra tersebut agar lebih sesuai dengan kebutuhan tertentu, seperti perbaikan kualitas, ekstraksi informasi, atau analisis. Dasar-dasar pengolahan citra melibatkan beberapa langkah utama, di antaranya adalah pre-processing (pra-pemrosesan), enhancement (pemperbaikan), segmentation (segmentasi), dan feature extraction (ekstraksi fitur). Pre-processing melibatkan langkah-langkah seperti pengurangan noise, normalisasi, dan resizing untuk mempersiapkan citra sebelum pemrosesan lebih lanjut. Enhancement melibatkan teknik-teknik seperti peningkatan kontras, penajaman, atau peningkatan kejelasan untuk memperbaiki kualitas citra. Segmentasi mencakup pemisahan objek dari latar belakang dengan teknik seperti pengklasifikasian piksel atau region-of-interest. Ekstraksi fitur melibatkan identifikasi dan pemanfaatan fitur-fitur penting dari citra, seperti tepi, tekstur, atau warna, yang dapat digunakan dalam analisis lebih lanjut. Selain itu, pengolahan citra digital juga melibatkan penggunaan berbagai filter, transformasi, dan metode matematika seperti transformasi Fourier atau transformasi wavelet untuk manipulasi yang lebih kompleks. Semua langkah ini bertujuan untuk meningkatkan pemahaman kita terhadap citra serta memungkinkan aplikasi

yang lebih luas, seperti pengenalan pola, deteksi objek, atau pengolahan medis.

Dasar-dasar dalam pengolahan citra digital melibatkan serangkaian langkah penting yang membentuk proses komprehensif. Pertama-tama, pra-pemrosesan (pre-processing) menjadi langkah awal yang melibatkan serangkaian teknik untuk mempersiapkan citra sebelum tahap analisis lebih lanjut. Proses ini mencakup pengurangan noise untuk menghilangkan gangguan yang tidak diinginkan, normalisasi untuk menyesuaikan citra dengan rentang nilai yang diharapkan, serta resizing untuk menyesuaikan resolusi atau ukuran citra agar sesuai dengan kebutuhan. Tahap selanjutnya adalah pemperbaikan (enhancement), yang bertujuan untuk meningkatkan kualitas citra dengan memperbaiki kontras, kejelasan, atau aspek visual lainnya. Teknik-teknik seperti peningkatan kecerahan, penajaman (sharpening), atau peningkatan kontras digunakan dalam tahap ini untuk meningkatkan citra agar lebih mudah dianalisis atau divisualisasikan.

Segmentasi merupakan langkah penting dalam pengolahan citra yang melibatkan pemisahan objek yang menarik dari latar belakangnya. Proses ini dapat dilakukan dengan berbagai metode, termasuk pengklasifikasian piksel berdasarkan atribut tertentu atau pembentukan region-of-interest (ROI) untuk menyoroti area yang relevan dalam citra Ekstraksi fitur (feature extraction) juga merupakan tahap kunci yang melibatkan identifikasi dan pemanfaatan fitur-fitur penting dalam citra. Ini bisa mencakup deteksi tepi, ekstraksi tekstur, analisis warna, atau bahkan ekstraksi fitur berbasis bentuk objek. Fitur-fitur ini kemudian dapat digunakan untuk analisis lanjutan atau untuk memasukkan informasi yang ditemukan ke dalam sistem yang lebih besar.

2. DenseNet201

DenseNet-201 adalah sebuah arsitektur jaringan saraf konvolusional (CNN) yang mengusung konsep koneksi yang padat, yang dirancang untuk meningkatkan kinerja jaringan secara signifikan. Dibangun di atas konsep koneksi yang kuat antar-lapisan, setiap lapisan pada DenseNet-201

terhubung langsung dengan semua lapisan di depannya dan di belakangnya. Pendekatan ini memungkinkan transfer informasi yang kaya dan langsung antara setiap lapisan, mempromosikan penggunaan kembali fitur yang telah dipelajari, dan mengatasi masalah hilangnya informasi yang sering muncul dalam jaringan yang dalam. Arsitektur ini terdiri dari blok-blok pembangunan yang saling terhubung, di mana setiap blok terdiri dari beberapa lapisan konvolusi, normalisasi batch, aktivasi, dan penggabungan hasil dari semua lapisan dalam blok tersebut. Melalui lapisan transisi yang mengurangi dimensi fitur, DenseNet-201 mempertahankan kompleksitas model yang terkendali. Keunggulan utama DenseNet-201 terletak pada kemampuan untuk mengurangi jumlah parameter yang digunakan dalam jaringan, meningkatkan akurasi dengan kedalaman yang lebih besar, dan meningkatkan efisiensi dalam tugas-tugas pengenalan gambar. Dengan 201 lapisan, DenseNet-201 memberikan representasi yang lebih dalam dari fitur-fitur dalam gambar, yang menjadikannya cocok untuk berbagai tugas pengolahan citra dengan kinerja yang optimal.

3. VGG16

VGG16 adalah salah satu arsitektur jaringan saraf konvolusional (CNN) yang terkenal karena keunggulannya dalam melakukan klasifikasi gambar. Dikembangkan oleh tim peneliti dari Visual Graphics Group (VGG) di Universitas Oxford, arsitektur ini diperkenalkan sebagai bagian dari kompetisi ImageNet pada tahun 2014. VGG16 memiliki struktur yang relatif sederhana tetapi sangat dalam, terdiri dari 16 lapisan konvolusi dan fully connected layers tanpa adanya struktur skip connections atau shortcut connections seperti pada arsitektur lainnya.

Salah satu karakteristik utama dari VGG16 adalah penggunaan blok-blok konvolusi yang relatif kecil (3x3) yang diikuti oleh lapisan pooling maksimum (max pooling) untuk mengurangi dimensi spatial dari representasi gambar. Struktur repetitif ini terdiri dari beberapa blok konvolusi, di mana setiap blok terdiri dari beberapa lapisan konvolusi berturut-turut, diikuti oleh lapisan pooling maksimum. Struktur yang sederhana dan simetris ini memungkinkan VGG16 untuk mempelajari

representasi hirarkis yang kompleks dari gambar, yang semakin mendalam di setiap blok konvolusi.

VGG16 telah terbukti sangat handal dalam tugas-tugas klasifikasi gambar karena kemampuannya untuk mengekstraksi fitur yang berguna dari citra. Namun, kelemahan utamanya adalah kompleksitasnya yang tinggi dengan jumlah parameter yang besar, yang membuatnya lebih memakan waktu dalam proses pelatihan dan membutuhkan sumber daya komputasi yang lebih besar pula.

Meskipun ada varian-varian lain dari arsitektur VGG dengan jumlah lapisan yang berbeda seperti VGG19 (dengan 19 lapisan), VGG16 tetap menjadi referensi penting dalam pengembangan jaringan saraf konvolusional dan telah memberikan kontribusi besar terhadap perkembangan dalam pengolahan citra. Meskipun terdapat arsitektur jaringan yang lebih kompleks dan efisien saat ini, penelitian dan kontribusi VGG16 masih menjadi dasar bagi banyak peneliti dalam memahami prinsip-prinsip dasar yang mendasari jaringan saraf konvolusional dan dalam mengembangkan arsitektur jaringan yang lebih canggih untuk tugas-tugas pengolahan citra yang kompleks.

BAB III
METODE

3.1 Pengumpulan Dataset

Pada Percobaan kali ini kami mengumpulkan dataset secara manual dengan mencari gambar dari Google. Selanjutnya kami mencari total 80 gambar dan membagi nya kedalam setiap kelas yaitu “matang busuk, matang segar, mentah busuk, mentah segar” Dimana dari setiap kelas tersebut memiliki 20 gambar perkelasnya. Setelah mengumpulkan data tersebut kemudian kamu melakukan proses augmentasi pada setiap kelas dengan tujuan untuk memperbanyak dataset. Berikut program yang mai buat untuk mengaugmentasi gambar dari setiap kelas dataset:

```
[8]: import tensorflow
from keras.preprocessing.image import ImageDataGenerator
from skimage import io
datagen = ImageDataGenerator(
        rotation_range=45,     #Random rotation between 0 and 45
        width_shift_range=0.2,   #% shift
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='reflect')
i = 0
for batch in datagen.flow_from_directory(directory='mentahbusuk',
                                         batch_size=7,
                                         target_size=(256, 256),
                                         color_mode="rgb",
                                         save_to_dir='mentahbusukaugment',
                                         save_prefix='aug',
                                         save_format='png'
                                         ):
    i += 1
    if i > 30:
        break
```

Gambar 3. 1 Kode Augmentasi

Dimana setelah menlakukan proses augmentasi didapatkan total 207 gambar per kelasnya dan ditambah 20 gambar dari setiap kelas yang sebelumnya. Dan didapatkan total gambar dari semua kelas yaitu 908 gambar.

3.2 Membagi Dataset

```
# dataset path directory
dataset_path = "C:/Users/ASUS/projekakhir/datatomat"

# Path for  directory for  training and validation data
base_output_path = "C:/Users/ASUS/projekakhir/split_data"
training_path = os.path.join(base_output_path, "train_data")
validation_path = os.path.join(base_output_path, "val_data")

# List all class in dataset
classes = os.listdir(dataset_path)

# Looping through every class
for class_name in classes:
    class_path = os.path.join(dataset_path, class_name)

    # get list file  in each class
    files = os.listdir(class_path)

    # split dataset to training and  validation data, training data(70%) and validation(30%)
    train_files, validation_files = train_test_split(files, test_size=0.1, random_state=42)

    # making new directory
    os.makedirs(os.path.join(training_path, class_name), exist_ok=True)
    os.makedirs(os.path.join(validation_path, class_name), exist_ok=True)

    # move file to training data directory
    for file in train_files:
        source_path = os.path.join(class_path, file)
        destination_path = os.path.join(training_path, class_name, file)
        shutil.copyfile(source_path, destination_path)

    # move file to  validation data directory
    for file in validation_files:
        source_path = os.path.join(class_path, file)
        destination_path = os.path.join(validation_path, class_name, file)
        shutil.copyfile(source_path, destination_path)

print("Split  Dataset  to 70% for training and 30% for validation successfully.")
```

```
Split  Dataset  to 70% for training and 30% for validation successfully.
```

Gambar 3. 2 Kode Split Dataset

Kode diatas bertujuan untuk membagi dataset ke dalam data pelatihan (training data) dan data validasi (validation data) dengan perbandingan 70% untuk pelatihan dan 30% untuk validasi. Proses ini dilakukan untuk persiapan penggunaan dataset dalam pembuatan model atau pengujian model pembelajaran mesin.

Langkah-langkah utama dalam kode tersebut adalah sebagai berikut:

- Inisialisasi path atau jalur direktori untuk dataset tomat dan direktori untuk menyimpan data pelatihan dan validasi.
- Mendapatkan daftar kelas atau kategori dari dataset dengan menggunakan os.listdir.
- Melakukan iterasi melalui setiap kelas di dataset.

- Untuk setiap kelas, kode akan memisahkan file-file yang termasuk ke dalam kelas tersebut.
- Penggunaan train_test_split dari pustaka Scikit-Learn untuk membagi file-file dalam setiap kelas menjadi data pelatihan dan validasi sesuai dengan perbandingan 80:20.
- Membuat direktori baru untuk setiap kelas di dalam direktori data pelatihan dan validasi menggunakan os.makedirs.
- Memindahkan file-file ke direktori data pelatihan dan validasi dengan menggunakan shutil.copyfile.

3.3 Pemeriksaan Distribusi Data

```
#Check data distribution
def count_files_in_directory(directory):
    return len(os.listdir(directory))

def check_data_distribution(training_dir, validation_dir):
    classes_training = os.listdir(training_dir)
    classes_validation = os.listdir(validation_dir)

    print("Data distribution in training directory:")
    for class_name in classes_training:
        class_path = os.path.join(training_dir, class_name)
        num_files = count_files_in_directory(class_path)
        print(f"{class_name}: {num_files} files")

    print("\nData distribution in validation directory:")
    for class_name in classes_validation:
        class_path = os.path.join(validation_dir, class_name)
        num_files = count_files_in_directory(class_path)
        print(f"{class_name}: {num_files} files")

# Replace with the appropriate path
TRAINING_DIR = "C:/Users/ASUS/projekakhir/split_data/train_data"
VALIDATION_DIR = "C:/Users/ASUS/projekakhir/split_data/val_data"

check_data_distribution(TRAINING_DIR, VALIDATION_DIR)
```

```
Data distribution in training directory:
matangbusuk: 204 files
matangsegar: 204 files
mentahbusuk: 204 files
mentahsegar: 204 files

Data distribution in validation directory:
matangbusuk: 23 files
matangsegar: 23 files
mentahbusuk: 23 files
mentahsegar: 23 files
```

Gambar 3. 3 Kode Distribusi Dataset

Kode yang diberikan memiliki tujuan untuk melakukan pemeriksaan distribusi data di dalam direktori data pelatihan (training directory) dan data validasi (validation directory) setelah proses pembagian dataset dilakukan sebelumnya.

Langkah-langkah utama dalam kode ini adalah sebagai berikut:

- Mendefinisikan fungsi count_files_in_directory yang mengembalikan jumlah file dalam sebuah direktori dengan menggunakan os.listdir dan len.
- Mendefinisikan fungsi check_data_distribution yang menerima dua argumen, yaitu direktori data pelatihan (training_dir) dan direktori data validasi (validation_dir).
- Mengambil daftar kelas atau kategori dari direktori data pelatihan dan data validasi.
- Melakukan iterasi pada setiap kelas di direktori data pelatihan dan data validasi.
- Menghitung jumlah file dalam setiap kelas menggunakan fungsi count_files_in_directory dan mencetak informasi jumlah file untuk setiap kelas dalam kedua direktori.

3.4 Generator Data

```
def train_val_generators(TRAINING_DIR, VALIDATION_DIR):
    TRAINING_DIR = "C:/Users/ASUS/projekakhir/split_data/train_data"
    VALIDATION_DIR = "C:/Users/ASUS/projekakhir/split_data/val_data"

    # NORMALIZE
    training_datagen = ImageDataGenerator(
        rescale=1.0 / 255.0,
        #rotation_range=40,
#        width_shift_range=0.2,
#        height_shift_range=0.2,
#        shear_range=0.2,
#        zoom_range=0.2,
#        horizontal_flip=False,
#        fill_mode='nearest',
        )

    validation_datagen = ImageDataGenerator(rescale=1.0 / 255.0)

    # Image size use 64x64 and color mode is grayscale
    # Used "categorical"
    train_generator = training_datagen.flow_from_directory(TRAINING_DIR,
                                                           target_size=(224, 224),
                                                           batch_size=32,
                                                           class_mode='categorical',
                                                           shuffle = True)
    validation_generator = validation_datagen.flow_from_directory(VALIDATION_DIR,
                                                                  target_size=(224, 224),
                                                                  batch_size=32,
                                                                  class_mode='categorical',
                                                                  shuffle = True)
    return train_generator, validation_generator
```

```
train_generator, validation_generator = train_val_generators(TRAINING_DIR, VALIDATION_DIR)
```

```
Found 816 images belonging to 4 classes.
Found 92 images belonging to 4 classes.
```

Gambar 3. 4 Kode Generator Data

Tujuan dari kode ini adalah untuk menyediakan generator data yang dapat secara dinamis memuat batch-batch dari direktori yang berisi data gambar dalam proses pelatihan dan validasi model jaringan saraf tiruan. Dengan menggunakan generator ini, data dapat dimuat secara efisien ke dalam model selama proses pelatihan, yang biasanya dilakukan secara batch demi batch untuk mengoptimalkan proses pelatihan dan penggunaan memori komputer**.**

langkah-langkah utama dalam kode ini:

- Menginisialisasi objek ImageDataGenerator untuk proses augmentasi gambar
- Dua generator data dibuat: satu untuk data pelatihan (train_generator) dan satu lagi untuk data validasi (validation_generator).
- Fungsi flow_from_directory dari ImageDataGenerator digunakan untuk menghasilkan generator yang mengambil data langsung dari direktori.
- Parameter-parameter yang digunakan dalam fungsi flow_from_directory termasuk target_size (ukuran gambar yang diharapkan), batch_size (jumlah sampel per batch), class_mode (tipe label kelas), dan shuffle (untuk mengacak urutan sampel).
- Generator data untuk pelatihan dan validasi dikembalikan dari fungsi sebagai output.

3.5 Model DenseNet201

```
[21]: # Loading DenseNet201 model
base_densenet_model = tf.keras.applications.DenseNet201(include_top=False,
                                                         weights='imagenet',
                                                          input_shape=(224,224,3),
                                                         pooling='max')
base_densenet_model.trainable=False
#train_data.preprocessing_function = tf.keras.applications.densenet.preprocess_input

[22]: # Transfer Learning DenseNet201
densenet_model = tf.keras.models.Sequential([
  base_densenet_model,
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(64, activation='relu'),
  tf.keras.layers.Dense(4, activation='softmax')
])

# Compiling model
densenet_model.compile(
    loss='categorical_crossentropy',
    optimizer=tf.keras.optimizers.Adam(),
    metrics=['accuracy']
  )

[26]: # Melatih model DenseNet201
densenet_hist = densenet_model.fit(
    train_generator,
    epochs=20,
    validation_data=validation_generator
)
```

Gambar 3. 5 Kode Model DenseNet201

Tujuan dari kode yang disediakan adalah untuk melakukan transfer learning dengan menggunakan pre-trained model DenseNet201 yang telah dilatih pada dataset ImageNet, dan menyesuaikannya untuk tugas klasifikasi khusus yang melibatkan empat kelas pada dataset yang dimiliki. Tujuan akhirnya adalah mendapatkan model yang dapat melakukan klasifikasi gambar pada dataset khusus dengan tingkat akurasi yang tinggi, menggunakan pengetahuan yang telah dipelajari dari dataset ImageNet melalui transfer learning pada arsitektur DenseNet201.

Langkah Langkah model diatas:

- Pemanggilan DenseNet201 Pre-trained Model
- Membekukan Bobot pada Model Dasar
- Pembuatan Model Transfer Learning
- Kompilasi Model
- Pelatihan Model DenseNet201

3.6 Model VGG16

[72]:
```
import tensorflow as tf
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Load the VGG16 model
base_vgg_model = VGG16(weights="imagenet", include_top=False, input_shape=(224,224,3))
base_vgg_model.trainable = False

# Define the preprocessing function
vgg_preprocess = tf.keras.applications.vgg16.preprocess_input

# Assuming you have defined your ImageDataGenerator for training and validation data:
train_generator, validation_generator = train_val_generators(TRAINING_DIR, VALIDATION_DIR)

# Transfer learning with VGG16
vgg_model = tf.keras.models.Sequential([
    base_vgg_model,
    tf.keras.layers.Dropout(0.7),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(4, activation='softmax')
])

# Compile the model
vgg_model.compile(
    loss='categorical_crossentropy',
    optimizer=tf.keras.optimizers.Adam(),
    metrics=['accuracy']
)

# Train the VGG16 model
vgg_hist = vgg_model.fit(
    train_generator,
    epochs=20,
    validation_data=validation_generator
)
```

```
Found 816 images belonging to 4 classes.
Found 92 images belonging to 4 classes.
```

Gambar 3. 6 Kode Model VGG16

Tujuan dari kode tersebut adalah melakukan transfer learning menggunakan model VGG16 yang telah dilatih pada dataset ImageNet untuk menyesuaikan model dengan dataset khusus yang memiliki empat kelas pada tugas klasifikasi gambar.

Berikut adalah tujuan dari setiap bagian kode:

- Memuat Model VGG16 yang Telah Dilatih
- Pendefinisian Fungsi Prapemrosesan (Preprocessing Function)
- Penggunaan ImageDataGenerator untuk Data Pelatihan dan Validasi
- Pembuatan Model Transfer Learning dengan VGG16
- Kompilasi Model
- Pelatihan Model VGG16

Tujuan akhirnya dari kode tersebut adalah mendapatkan model VGG16 yang telah disesuaikan dengan dataset khusus, dapat melakukan klasifikasi gambar pada empat kelas dengan akurasi yang tinggi setelah pelatihan dilakukan.

BAB IV
HASIL PERCOBAAN

4.1 Model DenseNet201

1. Hasil Train

Setelah dilakukan training sebanyak 20 epoch, menggunakan Model DenseNet201 Maka didapatkan hasil sebagai berikut pada epoch ke-20:

-Training Loss: 0.0016

-Training Accuracy: 1.0000

-Validation Loss: 0.0130

-Validation Accuracy: 0.9891

2. Grafik Perbandingan

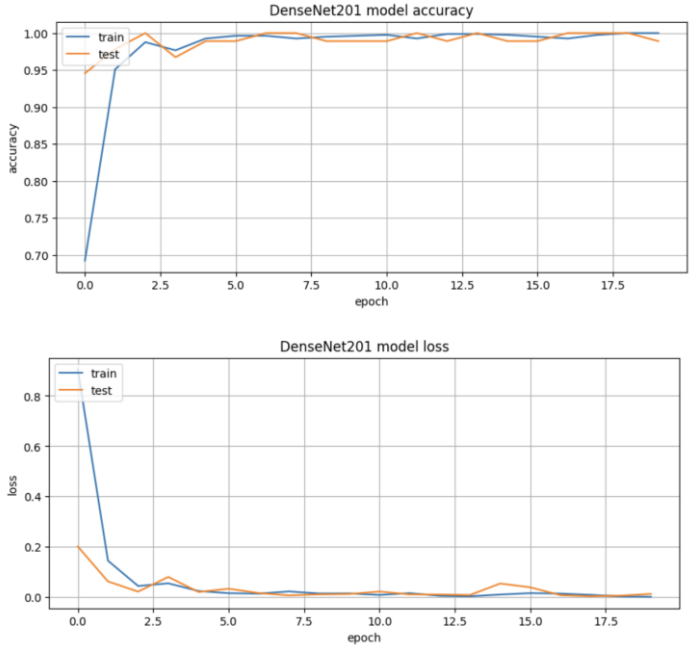
```
# Membuat plot akurasi model DenseNet201
plt.figure(figsize=(10,4))
plt.plot(densenet_hist.history['accuracy'])
plt.plot(densenet_hist.history['val_accuracy'])
plt.title('DenseNet201 model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()

print()

# Membuat plot loss model DenseNet201
plt.figure(figsize=(10,4))
plt.plot(densenet_hist.history['loss'])
plt.plot(densenet_hist.history['val_loss'])
plt.title('DenseNet201 model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()
```

Gambar 4. 1 Kode Grafik Akurasi dan Loss

Kode Diatas dibuat untuk melihat grafik akurasi dan loss pada Model DenseNet201, dan berikut hasilnya:

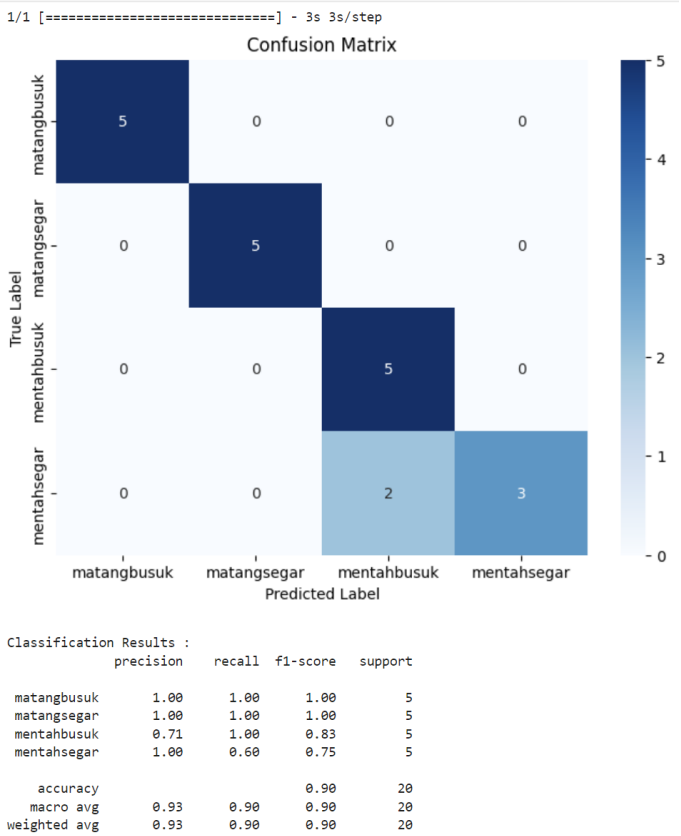


Gambar 4. 2 Grafik Akurasi dan Loss

Secara keseluruhan, grafik tersebut menunjukkan bahwa model DenseNet201 dapat mencapai akurasi yang tinggi pada data pelatihannya. Akurasi model pelatihan meningkat secara bertahap seiring bertambahnya jumlah epoch. Ini menunjukkan bahwa model tersebut belajar untuk membedakan antara data pelatihan dan data non-pelatihan dengan lebih baik seiring waktu.

3. Performa Model

Selanjutnya untuk melakukan pengujian performa model, kami melakukan pengujian menggunakan Confusion Matriks, dan kami menggunakan total 20 data uji, Dimana terdapat 5 gambar data testing untuk setiap kelasnya yaitu matang busuk, matang segar, mentah busuk, mentah segar. Berikut hasil pengujian dari Confusion Matriks yang kami Buat:



Classification Results :

	precision	recall	f1-score	support
matangbusuk	1.00	1.00	1.00	5
matangsegar	1.00	1.00	1.00	5
mentahbusuk	0.71	1.00	0.83	5
mentahsegar	1.00	0.60	0.75	5
accuracy			0.90	20
macro avg	0.93	0.90	0.90	20
weighted avg	0.93	0.90	0.90	20

Gambar 4. 3 confusion matrix

Jumlah model memprediksi benar dari seluruh data test yang tersedia adalah 5 + 5 + 5 + 3, yaitu sebanyak 18 prediksi dari 20 data test yang tersedia. Hal tersebut ditunjukkan oleh nilai accuracy model yang hanya sebesar 0.90.

4.2 Model VGG16

1. Hasil Train

Setelah dilakukan training sebanyak 20 epoch, menggunakan Model VGG16 Maka didapatkan hasil sebagai berikut pada epoch ke-20:

-Training Loss: 0.0134

-Training Accuracy: 0.9939

-Validation Loss: 0.1347

-Validation Accuracy: 0.9565

2. Grafik Perbandingan

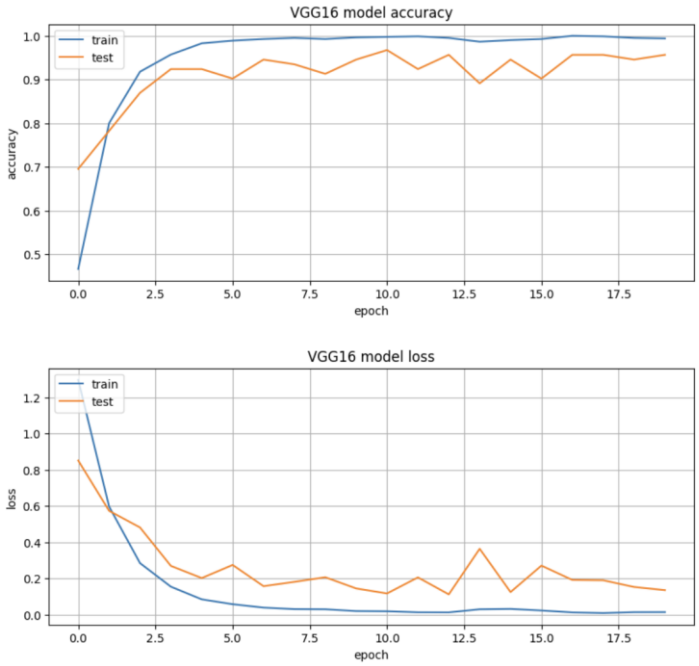
```
# Membuat plot akurasi model VGG16
plt.figure(figsize=(10,4))
plt.plot(vgg_hist.history['accuracy'])
plt.plot(vgg_hist.history['val_accuracy'])
plt.title('VGG16 model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()

print()

# Membuat plot loss model VGG16
plt.figure(figsize=(10,4))
plt.plot(vgg_hist.history['loss'])
plt.plot(vgg_hist.history['val_loss'])
plt.title('VGG16 model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.grid(True)
plt.show()
```

Gambar 4. 4 Kode Grafik Akurasi dan Loss

Kode Diatas dibuat untuk melihat grafik akurasi dan loss pada Model VGG16, dan berikut hasilnya:



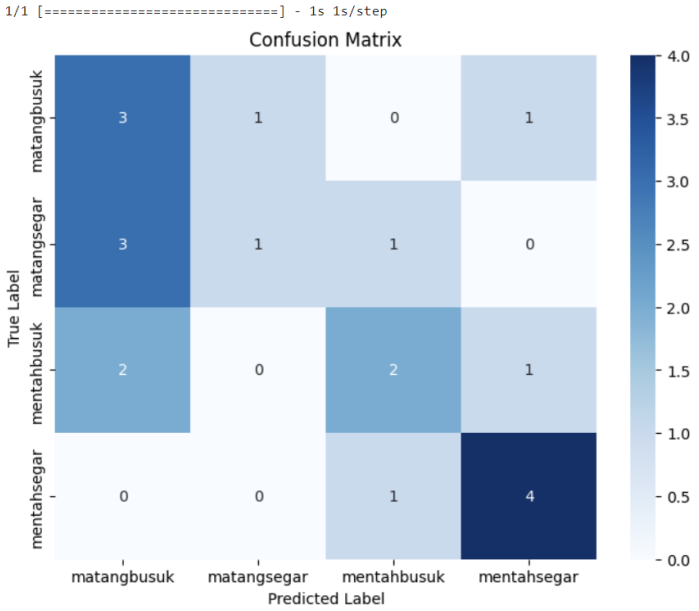
Gambar 4. 5 Grafik Akurasi dan Loss

Grafik tersebut menunjukkan dua kurva, satu untuk akurasi model pelatihan dan satu untuk akurasi model pengujian. kurva akurasi model pelatihan dan pengujian berpotongan pada sekitar epoch ke-10. Ini menunjukkan bahwa model tersebut telah mencapai batas

kinerjanya dan tidak akan meningkatkan akurasi lebih lanjut dengan pelatihan lebih lanjut, Begitu juga dengan kurva loss.

3. Performa Model

Sama Seperti Model sebelumnya pada model VGG16 ini , kami juga melakukan pengujian menggunakan Confusion Matriks, dan kami menggunakan total 20 data uji yang sama seperti model yang sebelunya. Berikut hasik dari confusion matrix nya:



Classification Results :

	precision	recall	f1-score	support
matangbusuk	0.38	0.60	0.46	5
matangsegar	0.50	0.20	0.29	5
mentahbusuk	0.50	0.40	0.44	5
mentahsegar	0.67	0.80	0.73	5
accuracy			0.50	20
macro avg	0.51	0.50	0.48	20
weighted avg	0.51	0.50	0.48	20

Gambar 4. 6 confusion matrix

Jumlah model memprediksi benar dari seluruh data test yang tersedia adalah 3 + 1 + 2 + 4, yaitu sebanyak 10 prediksi dari 20 data test yang tersedia. Hal tersebut ditunjukkan oleh nilai accuracy model yang hanya sebesar 0.50.

BAB V
KESIMPULAN

Dari Hasil Percobaan tersebut dapat disimpulkan bahwa Model DenseNet201 adalah alat yang cukup efektif untuk klasifikasi kesegaran buah tomat, jika dibandingkan dengan model VGG16. Model DenseNet201 mencapai akurasi keseluruhan 0.90, sedangkan untuk model VGG16 hanya mencapai akurasi 0.50. Terdapat selisih 0,40 antara dua model tersebut. Sehingga dari percobaan kali ini dengan total dataset sebanyak 908 dimana 816 diantaranya sebagai data train dan 92 nya untuk data validasi, dan juga setelah dilakukan pengujian menggunakan confusion matrix didapatkan bahwa penggunaan Model DenseNet201 Lebih Efektif dibandingkan model VGG16 untuk kasus Deteksi Kesegaran Buah Tomat.