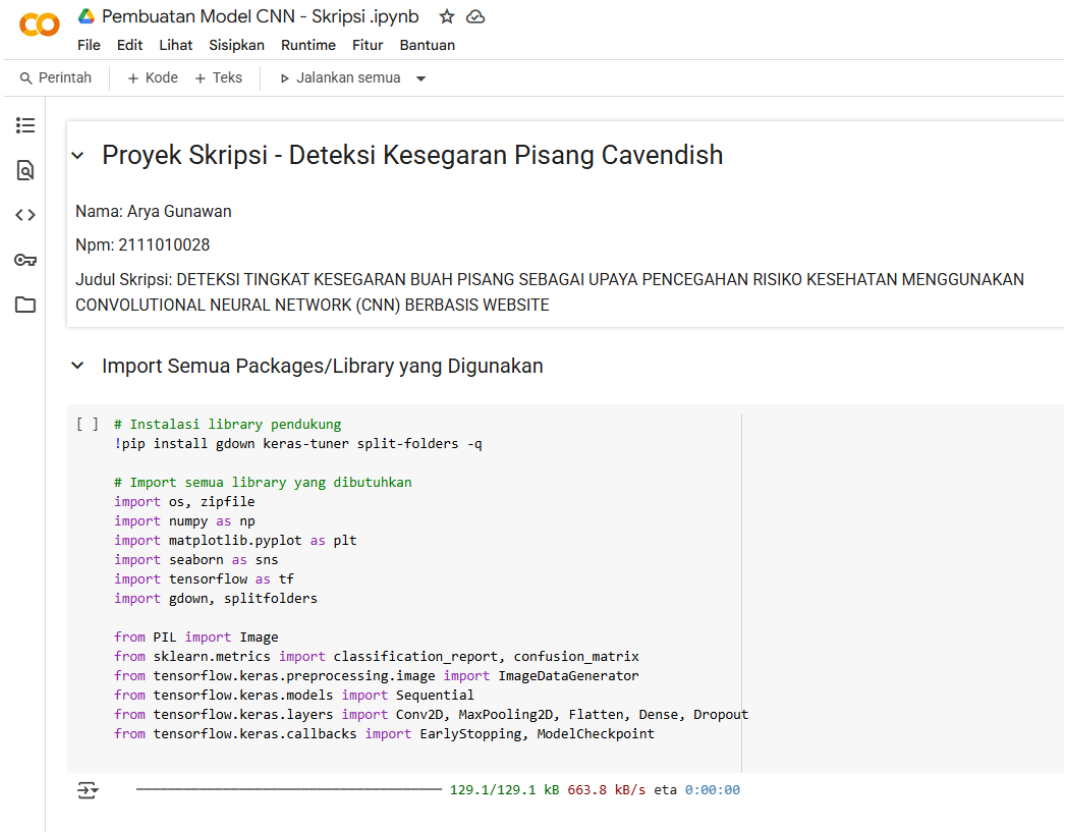
LAMPIRAN

Lampiran 1

Kode Python Membuat Model CNN



Pembuatan Model CNN - Skripsi .ipynb

File Edit Lihat Sisipkan Runtime Fitur Bantuan

Perintah + Kode + Teks ▷ Jalankan semua

Data Preparation

Data Loading

Download dan Ekstraksi Dataset

```
# Unduh dataset dari Google Drive
file_id = "1WoBWbnmIDfzOsvVB1gBVZ2GUYGZ8Gxlg"
output_file = "Dataset_Pisang_Cavendish.zip"
gdown.download(f"https://drive.google.com/uc?id={file_id}", output_file, quiet=False)

# Ekstrak file zip
with zipfile.ZipFile(output_file, 'r') as zip_ref:
    zip_ref.extractall("Dataset_Pisang")
```

```
Downloading...
From (original): https://drive.google.com/uc?id=1WoBWbnmIDfzOsvVB1gBVZ2GUYGZ8Gxlg
From (redirected): https://drive.google.com/uc?id=1WoBWbnmIDfzOsvVB1gBVZ2GUYGZ8Gxlg&confirm=t&uuid=cee36158-a8d8-4de8-bd6a-0378b49548b7
To: /content/Dataset_Pisang_Cavendish.zip
100%|██████████| 336M/336M [00:04<00:00, 75.5MB/s]
```

Cek Jumlah Gambar per Kelas

```
dataset_path = "Dataset_Pisang/Dataset Pisang Cavendish"
for class_name in os.listdir(dataset_path):
    class_path = os.path.join(dataset_path, class_name)
    print(f"{class_name}: {len(os.listdir(class_path))} images")
```

```
matang: 249 images
segar: 264 images
busuk: 261 images
mentah: 256 images
```

Perintah + Kode + Teks ▷ Jalankan semua

```
segar: 264 images
```

Sample Gambar Dataset

```
import os
import matplotlib.pyplot as plt
import gdown, zipfile, random

# --- Visualisasi 4 gambar acak per kelas ---
for class_name in os.listdir(dataset_path):
    class_path = os.path.join(dataset_path, class_name)
    sample_imgs = random.sample(os.listdir(class_path), 4)  # ambil 4 gambar acak

    fig, axes = plt.subplots(1, 4, figsize=(12, 4))
    fig.suptitle(f"Contoh Dataset - {class_name}", fontsize=14)

    for idx, img_file in enumerate(sample_imgs):
        img_path = os.path.join(class_path, img_file)
        img = plt.imread(img_path)
        axes[idx].imshow(img)
        axes[idx].set_title(f"{class_name} {idx+1}")
        axes[idx].axis("off")

    plt.tight_layout()
    plt.show()
```

Contoh Dataset - mentah

Contoh Dataset - busuk

Contoh Dataset - matang

Contoh Dataset - segar

Pembuatan Model CNN - Skripsi .ipynb

File Edit Lihat Sisipkan Runtime Fitur Bantuan

Perintah + Kode + Teks Jalankan semua

Pra-Pemrosesan Data (Preprocessing)

```
[ ] # Parameter dasar gambar
    img_height, img_width = 150, 150
    batch_size = 32

    # Inisialisasi ImageDataGenerator dengan normalisasi piksel
    datagen = ImageDataGenerator(rescale=1./255)
```

Split Data

```
[ ] # Membagi dataset menjadi train (70%), val (15%), dan test (15%)
    splitfolders.ratio("Dataset_Pisang/Dataset Pisang Cavendish", output="dataset_split",
                       seed=42, ratio=(.7, .15, .15), move=True)

    # Load dataset ke dalam generator
    train_data = datagen.flow_from_directory(
        "dataset_split/train", target_size=(img_height, img_width),
        batch_size=batch_size, class_mode='categorical', shuffle=True
    )

    val_data = datagen.flow_from_directory(
        "dataset_split/val", target_size=(img_height, img_width),
        batch_size=batch_size, class_mode='categorical', shuffle=True
    )

    test_data = datagen.flow_from_directory(
        "dataset_split/test", target_size=(img_height, img_width),
        batch_size=batch_size, class_mode='categorical', shuffle=False
    )
```

```
Copying files: 1030 files [00:00, 20959.82 files/s]Found 719 images belonging to 4 classes.
Found 153 images belonging to 4 classes.
Found 158 images belonging to 4 classes.
```

Perintah | + Kode + Teks | Jalankan semua

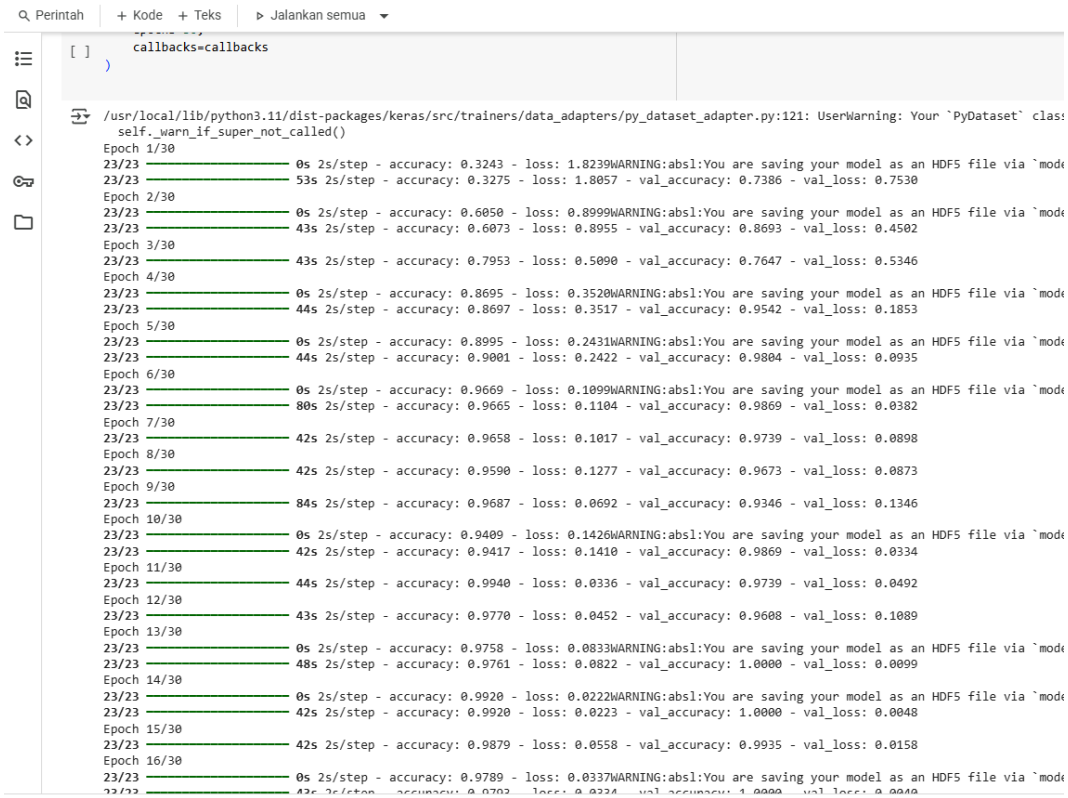
Perancangan CNN

```
[ ] # Arsitektur CNN
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(img_height, img_width, 3)),
        MaxPooling2D(2, 2),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D(2, 2),
        Conv2D(128, (3, 3), activation='relu'),
        MaxPooling2D(2, 2),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(train_data.num_classes, activation='softmax')
    ])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: User
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Pelatihan dan Validasi Model CNN

```
[ ] # Compile Model
    model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    # Konfigurasi callback
    callbacks = [
        EarlyStopping(patience=5, restore_best_weights=True),
        ModelCheckpoint("best_model.h5", save_best_only=True)
    ]

    # Train model
    history = model.fit(
        train_data,
        validation_data=val_data,
        epochs=30,
        callbacks=callbacks
    )
```



23/23 42s 2s/step - accuracy: 0.9879 - loss: 0.0558 - val_accuracy: 0.9935 - val_loss: 0.0158
Epoch 16/30
23/23 0s 2s/step - accuracy: 0.9789 - loss: 0.0337WARNING:absl:You are saving your model as an HDF5 file via `moc
23/23 43s 2s/step - accuracy: 0.9793 - loss: 0.0334 - val_accuracy: 1.0000 - val_loss: 0.0040
Epoch 17/30
23/23 42s 2s/step - accuracy: 0.9978 - loss: 0.0148 - val_accuracy: 1.0000 - val_loss: 0.0045
Epoch 18/30
23/23 42s 2s/step - accuracy: 0.9929 - loss: 0.0198 - val_accuracy: 1.0000 - val_loss: 0.0104
Epoch 19/30
23/23 43s 2s/step - accuracy: 0.9877 - loss: 0.0445 - val_accuracy: 1.0000 - val_loss: 0.0079
Epoch 20/30
23/23 0s 2s/step - accuracy: 0.9941 - loss: 0.0180WARNING:absl:You are saving your model as an HDF5 file via `moc
23/23 42s 2s/step - accuracy: 0.9941 - loss: 0.0178 - val_accuracy: 1.0000 - val_loss: 0.0013
Epoch 21/30
23/23 42s 2s/step - accuracy: 0.9985 - loss: 0.0099 - val_accuracy: 1.0000 - val_loss: 0.0037
Epoch 22/30
23/23 81s 2s/step - accuracy: 0.9991 - loss: 0.0096 - val_accuracy: 0.9935 - val_loss: 0.0130
Epoch 23/30
23/23 42s 2s/step - accuracy: 0.9856 - loss: 0.0372 - val_accuracy: 0.9804 - val_loss: 0.0350
Epoch 24/30
23/23 42s 2s/step - accuracy: 0.9830 - loss: 0.0472 - val_accuracy: 1.0000 - val_loss: 0.0119
Epoch 25/30
23/23 42s 2s/step - accuracy: 0.9958 - loss: 0.0211 - val_accuracy: 1.0000 - val_loss: 0.0087

Perintah | + Kode | + Teks | Jalankan semua

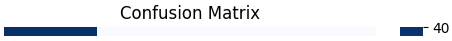
Evaluasi Model

```
[ ] # Evaluasi akurasi pada test set
    test_loss, test_acc = model.evaluate(test_data)
    print(f"Test Accuracy: {test_acc*100:.2f}%")
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 5s 1s/step - accuracy: 0.9945 - loss: 0.0176
Test Accuracy: 98.73%
```

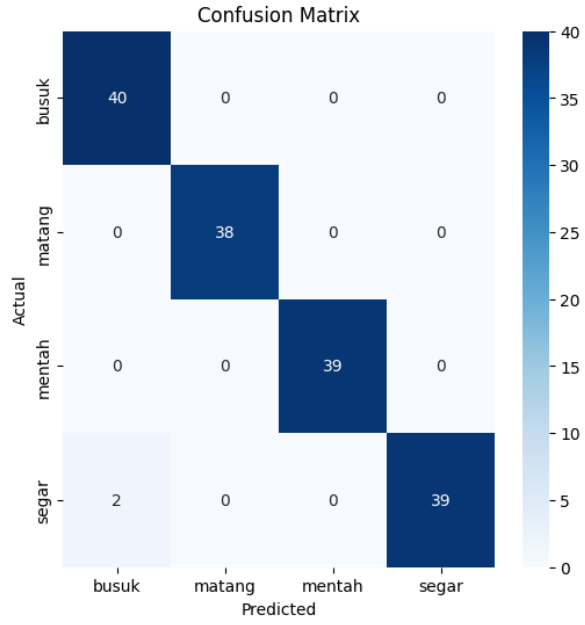
```
# Prediksi dan evaluasi lanjutan
y_pred = model.predict(test_data)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = test_data.classes

# Confusion Matrix
conf_matrix = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(6,6))
sns.heatmap(conf_matrix, annot=True, fmt="d",
            xticklabels=test_data.class_indices,
            yticklabels=test_data.class_indices, cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Classification Report
print(classification_report(y_true, y_pred_classes, target_names=list(test_data.class_indices.keys())))
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 3s 700ms/step
```



Perintah | + Kode | + Teks | Jalankan semua

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 3s 700ms/step
```



	precision	recall	f1-score	support
busuk	0.95	1.00	0.98	40
matang	1.00	1.00	1.00	38
mentah	1.00	1.00	1.00	39
segar	1.00	0.95	0.97	41
accuracy			0.99	158
macro avg	0.99	0.99	0.99	158
weighted avg	0.99	0.99	0.99	158

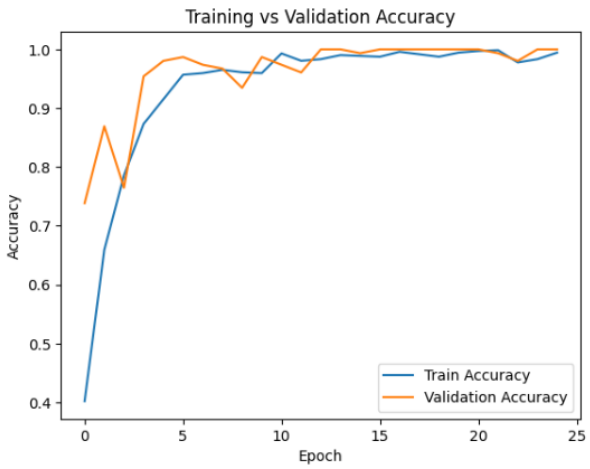
Simpan Model

```
[ ] model.save("model_pisang.h5")
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This fi
```

Visualisasi Akurasi

```
[ ] # Visualisasi kurva akurasi pelatihan dan validasi
    plt.plot(history.history['accuracy'], label='Train Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.legend()
    plt.title('Training vs Validation Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.show()
```



Lampiran 2

Implementasi Website