📌 : Klasifikasi Telur Ayam **Fertile** dan **Infertile** Menggunakan CNN (TensorFlow)

Deskripsi:

Notebook ini membangun model klasifikasi biner menggunakan Convolutional Neural Network (CNN) untuk mendeteksi telur ayam fertile dan infertile berdasarkan citra. Dataset sudah mencakup augmentasi dan dibagi ke dalam data train, validasi, dan test.

Source Dataset 🔗

Penjelasan Augmentasi Dataset yang Digunakan

Dataset yang digunakan dalam notebook ini sudah dilengkapi dengan augmentasi data bawaan, sehingga tidak perlu menambahkan augmentasi lagi di tahap preprocessing. Augmentasi ini melibatkan teknik-teknik berikut untuk memperkaya variasi data dan mencegah overfitting pada model:

1. **Auto-Orient**: Menerapkan orientasi otomatis pada gambar jika diperlukan.
2. **Resize**: Mengubah ukuran gambar menjadi 640x640 piksel.
3. **Augmentasi pada Citra**:
   - **Crop**: 0% Minimum Zoom dan 20% Maximum Zoom (untuk pemotongan acak).
   - **Rotasi**: Gambar dapat diputar antara -15° hingga +15°.
   - **Shear**: Pergeseran horizontal dan vertikal hingga ±10°.
4. **Bounding Box**:
   - **Crop**: 0% Minimum Zoom dan 20% Maximum Zoom pada Bounding Box.
   - **Rotasi**: Rotasi gambar di dalam Bounding Box antara -10° hingga +10°.
   - **Shear**: Pergeseran Bounding Box horizontal dan vertikal hingga ±5°.

Augmentasi ini dilakukan untuk memastikan bahwa model dapat belajar dari data yang lebih beragam dan lebih kuat terhadap variasi di dunia nyata, seperti rotasi, zoom, atau pergeseran objek dalam gambar.

Untuk tahap preprocessing di notebook ini, hanya dilakukan normalisasi pada data citra dengan mengatur `rescale=1./255`, yang akan mengubah nilai pixel gambar agar berada di rentang [0, 1].

```python
import os

def print_folder_tree(startpath, prefix=""):
    items = sorted(os.listdir(startpath))
    dirs = [item for item in items if os.path.isdir(os.path.join(startpath, item))]
    pointers = ['├── '] * (len(dirs) - 1) + ['└── ']

    for pointer, folder in zip(pointers, dirs):
        path = os.path.join(startpath, folder)
        print(prefix + pointer + folder)
        extension = '│   ' if pointer == '├── ' else '    '
        print_folder_tree(path, prefix + extension)

root_dir = "/content/drive/MyDrive/ColabNotebooks/EggClassification~11"
print(root_dir)
print_folder_tree(root_dir)
```

```
/content/drive/MyDrive/ColabNotebooks/EggClassification~11
├── .ipynb_checkpoints
├── dataset
│   ├── test
│   │   ├── .ipynb_checkpoints
│   │   ├── fertile
│   │   └── unfertile
│   ├── train
│   │   ├── .ipynb_checkpoints
│   │   ├── fertile
│   │   └── unfertile
│   └── valid
│       ├── .ipynb_checkpoints
│       ├── fertile
│       └── unfertile
└── models
```

🔷 Step 1: Import Library

Import semua library yang diperlukan untuk membangun, melatih, dan mengevaluasi model klasifikasi, termasuk TensorFlow, Keras, sklearn, matplotlib, dan lainnya.

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import layers, models
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
import matplotlib.pyplot as plt
import os
import numpy as np
```

🔷 Step 2: Menentukan Path Dataset

Tentukan direktori untuk dataset train, validasi, dan test yang sudah disiapkan dan dipisahkan sebelumnya.

```
# Step 2: Set Dataset Path
base_dir = "/content/drive/MyDrive/ColabNotebooks/EggClassification~11/dataset/"

train_dir = os.path.join(base_dir, "train")
valid_dir = os.path.join(base_dir, "valid")
test_dir = os.path.join(base_dir, "test")
```

🔷 Step 3: Preprocessing Data

Melakukan normalisasi dengan rescale=1./255 untuk semua subset data. Dataset sudah memiliki augmentasi sebelumnya, sehingga tidak ditambahkan lagi di sini.

```
# Step 3: Data Preprocessing
# Augmentasi sudah diterapkan di dataset, hanya rescale untuk normalisasi
train_datagen = ImageDataGenerator(rescale=1.0/255)
valid_datagen = ImageDataGenerator(rescale=1.0/255)
test_datagen = ImageDataGenerator(rescale=1.0/255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)

valid_generator = valid_datagen.flow_from_directory(
    valid_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary',
    shuffle=False
)
```

```
Found 1362 images belonging to 2 classes.
Found 50 images belonging to 2 classes.
Found 50 images belonging to 2 classes.
```

** 🔷 Step 4: Membangun Model CNN **

Membangun arsitektur CNN dengan 3 blok konvolusi dan 1 lapisan Dense, serta Dropout dan Regularization untuk mencegah overfitting.

```
# Step 4: Build CNN Model
model = models.Sequential([
    # Convolutional Block 1
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),
    layers.MaxPooling2D((2, 2)),
```

```
    # Convolutional Block 2
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    # Convolutional Block 3
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    # Fully Connected Layers
    layers.Flatten(),
    layers.Dense(128, kernel_regularizer=tf.keras.regularizers.l2(0.01), activation='relu'),
    layers.Dropout(0.6),  # Dropout to reduce overfitting
    layers.Dense(1, activation='sigmoid')  # Binary classification: fertile/unfertile
])
```

🔷 Step 5: Kompilasi Model

Mengkompilasi model dengan optimizer Adam dan loss function binary_crossentropy, karena ini adalah kasus klasifikasi biner.

```
# Step 5: Compile the Model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

🔷 Step 6: Menentukan Callback

Menambahkan EarlyStopping dan ModelCheckpoint untuk menghentikan pelatihan dini jika model tidak membaik dan menyimpan model terbaik berdasarkan val_loss.

```
# Step 6: Callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
checkpoint = ModelCheckpoint('simple_cnn_model.h5', monitor='val_loss', save_best_only=True, mode='min')
```

🔷 Step 7: Melatih Model

Melatih model pada data train dan validasi selama maksimal 20 epoch, dengan callback diterapkan. Proses ini akan menampilkan metrik akurasi dan loss setiap epoch.

```
# Step 7: Train the Model
history = model.fit(
    train_generator,
    validation_data=valid_generator,
    epochs=20,
    callbacks=[early_stopping, checkpoint]
)
```

```
Epoch 1/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 191ms/step - accuracy: 0.6801 - loss: 1.6666WARNING:absl:You are saving your model as an HDF5 file via `mc
43/43 ━━━━━━━━━━━━━━━━━━━━ 13s 227ms/step - accuracy: 0.6818 - loss: 1.6533 - val_accuracy: 0.8200 - val_loss: 0.6246
Epoch 2/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 168ms/step - accuracy: 0.8459 - loss: 0.5722WARNING:absl:You are saving your model as an HDF5 file via `mc
43/43 ━━━━━━━━━━━━━━━━━━━━ 8s 180ms/step - accuracy: 0.8461 - loss: 0.5717 - val_accuracy: 0.8200 - val_loss: 0.5456
Epoch 3/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 153ms/step - accuracy: 0.8798 - loss: 0.4417WARNING:absl:You are saving your model as an HDF5 file via `mc
43/43 ━━━━━━━━━━━━━━━━━━━━ 7s 163ms/step - accuracy: 0.8800 - loss: 0.4411 - val_accuracy: 0.8600 - val_loss: 0.4216
Epoch 4/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 153ms/step - accuracy: 0.9018 - loss: 0.4310WARNING:absl:You are saving your model as an HDF5 file via `mc
43/43 ━━━━━━━━━━━━━━━━━━━━ 10s 164ms/step - accuracy: 0.9018 - loss: 0.4303 - val_accuracy: 0.9400 - val_loss: 0.3094
Epoch 5/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 11s 178ms/step - accuracy: 0.9301 - loss: 0.3394 - val_accuracy: 0.9000 - val_loss: 0.4590
Epoch 6/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 155ms/step - accuracy: 0.9308 - loss: 0.3476WARNING:absl:You are saving your model as an HDF5 file via `mc
43/43 ━━━━━━━━━━━━━━━━━━━━ 7s 166ms/step - accuracy: 0.9309 - loss: 0.3471 - val_accuracy: 0.9200 - val_loss: 0.2647
Epoch 7/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 8s 172ms/step - accuracy: 0.9508 - loss: 0.2941 - val_accuracy: 0.9000 - val_loss: 0.3338
Epoch 8/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 160ms/step - accuracy: 0.9425 - loss: 0.3149WARNING:absl:You are saving your model as an HDF5 file via `mc
43/43 ━━━━━━━━━━━━━━━━━━━━ 7s 173ms/step - accuracy: 0.9427 - loss: 0.3146 - val_accuracy: 1.0000 - val_loss: 0.1827
Epoch 9/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 11s 176ms/step - accuracy: 0.9697 - loss: 0.2339 - val_accuracy: 0.9600 - val_loss: 0.2408
Epoch 10/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 8s 178ms/step - accuracy: 0.9550 - loss: 0.2623 - val_accuracy: 0.9200 - val_loss: 0.4638
Epoch 11/20
```

```
43/43 ━━━━━━━━━━━━━━━━━━━━ 7s 159ms/step - accuracy: 0.9712 - loss: 0.2726 - val_accuracy: 0.9800 - val_loss: 0.2086
Epoch 12/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 8s 177ms/step - accuracy: 0.9665 - loss: 0.2154 - val_accuracy: 0.9600 - val_loss: 0.2466
Epoch 13/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 0s 162ms/step - accuracy: 0.9702 - loss: 0.2199WARNING:absl:You are saving your model as an HDF5 file via `mc
43/43 ━━━━━━━━━━━━━━━━━━━━ 8s 176ms/step - accuracy: 0.9703 - loss: 0.2192 - val_accuracy: 1.0000 - val_loss: 0.1145
Epoch 14/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 10s 158ms/step - accuracy: 0.9817 - loss: 0.1543 - val_accuracy: 0.9600 - val_loss: 0.3362
Epoch 15/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 7s 165ms/step - accuracy: 0.9791 - loss: 0.2241 - val_accuracy: 0.9800 - val_loss: 0.1627
Epoch 16/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 10s 157ms/step - accuracy: 0.9864 - loss: 0.1793 - val_accuracy: 0.9600 - val_loss: 0.2163
Epoch 17/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 7s 163ms/step - accuracy: 0.9777 - loss: 0.2168 - val_accuracy: 0.9800 - val_loss: 0.1335
Epoch 18/20
43/43 ━━━━━━━━━━━━━━━━━━━━ 7s 168ms/step - accuracy: 0.9958 - loss: 0.1379 - val_accuracy: 0.9800 - val_loss: 0.1375
```

🔷 Step 8: Evaluasi Model pada Data Uji

Menghitung akurasi dan loss model terhadap dataset pengujian untuk mengetahui performa generalisasi model.

```
# Step 8: Evaluate Model on Test Dataset
test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Accuracy: {test_accuracy * 100:.2f}%")
print(f"Test Loss: {test_loss:.4f}")
```

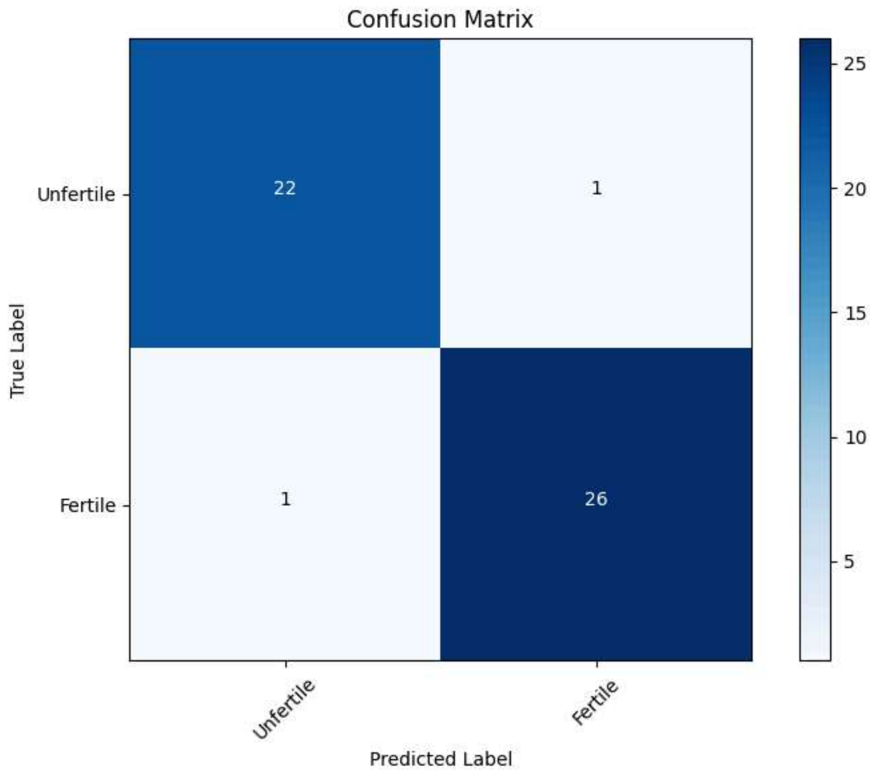
```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 85ms/step - accuracy: 0.9629 - loss: 0.1645
Test Accuracy: 96.00%
Test Loss: 0.1730
```

🔷 Step 9: Membuat Confusion Matrix

Menghitung dan memvisualisasikan confusion matrix dari hasil prediksi untuk mengevaluasi performa klasifikasi secara lebih rinci.

```
# Step 9: Confusion Matrix
# Predict labels on the test dataset
Y_pred = model.predict(test_generator)
y_pred = (Y_pred > 0.5).astype(int)  # Convert probabilities to binary classes

# Generate confusion matrix
conf_matrix = confusion_matrix(test_generator.classes, y_pred)

print("Confusion Matrix:")
print(conf_matrix)

# Visualize confusion matrix
plt.figure(figsize=(8, 6))
plt.imshow(conf_matrix, interpolation='nearest', cmap=plt.cm.Blues)
plt.title("Confusion Matrix")
plt.colorbar()
tick_marks = np.arange(2)
plt.xticks(tick_marks, ['Unfertile', 'Fertile'], rotation=45)
plt.yticks(tick_marks, ['Unfertile', 'Fertile'])
plt.ylabel('True Label')
plt.xlabel('Predicted Label')

# Annotate confusion matrix
thresh = conf_matrix.max() / 2.0
for i, j in np.ndindex(conf_matrix.shape):
    plt.text(j, i, f'{conf_matrix[i, j]}', horizontalalignment="center",
             color="white" if conf_matrix[i, j] > thresh else "black")

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

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 1s 320ms/step
Confusion Matrix:
[[22  1]
 [ 1 26]]
```



🔷 Step 10: Menampilkan Classification Report

Menampilkan metrik evaluasi model seperti precision, recall, dan f1-score untuk masing-masing kelas (fertile dan unfertile).

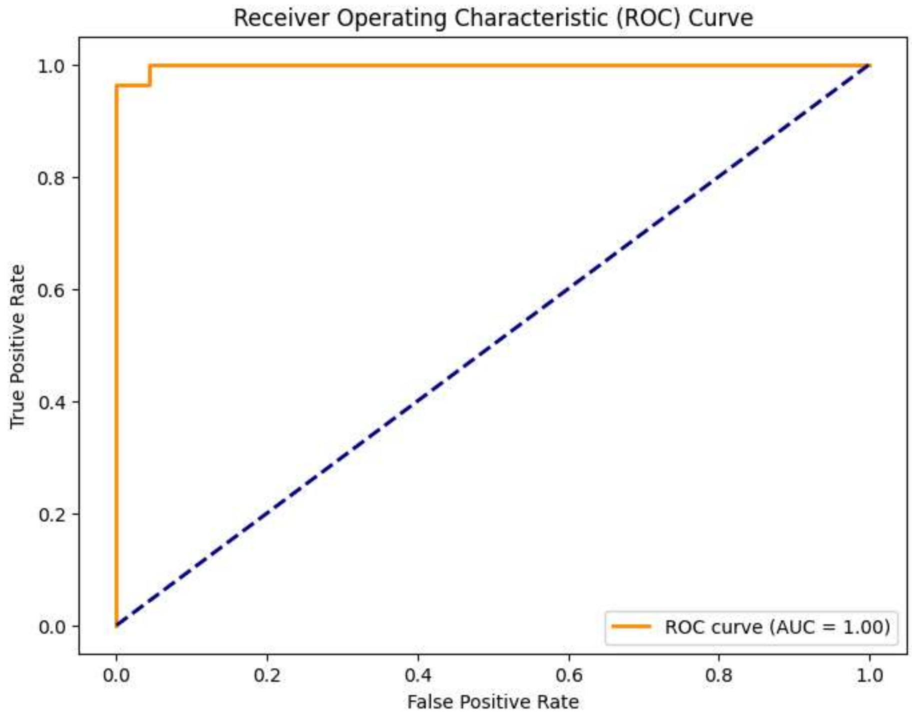
```
# Step 10: Classification Report
print("Classification Report:")
print(classification_report(test_generator.classes, y_pred, target_names=['Unfertile', 'Fertile']))
```

```
Classification Report:
              precision    recall  f1-score   support

   Unfertile       0.96      0.96      0.96        23
     Fertile       0.96      0.96      0.96        27

    accuracy                           0.96        50
   macro avg       0.96      0.96      0.96        50
weighted avg       0.96      0.96      0.96        50
```

🔷 Step 11: Plot ROC Curve

Menggambar ROC Curve dan menghitung AUC untuk mengukur kinerja klasifikasi secara keseluruhan.

```
# Step 11: ROC Curve
fpr, tpr, _ = roc_curve(test_generator.classes, Y_pred)
roc_auc = auc(fpr, tpr)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc="lower right")
plt.show()
```



🔷 **Step 12: Menyimpan Model dalam Format Keras**

Menyimpan model terlatih dalam format .keras yang direkomendasikan oleh TensorFlow untuk digunakan atau diekspor di masa depan.

```
# Step 12: Save Model in Keras Native Format
model.save('/content/drive/MyDrive/ColabNotebooks/EggClassification~11/models/final_model.keras')
print("Model saved to /content/drive/MyDrive/ColabNotebooks/EggClassification~11/models/final_model.keras")
```

```
Model saved to /content/drive/MyDrive/ColabNotebooks/EggClassification~11/models/final_model.keras
```