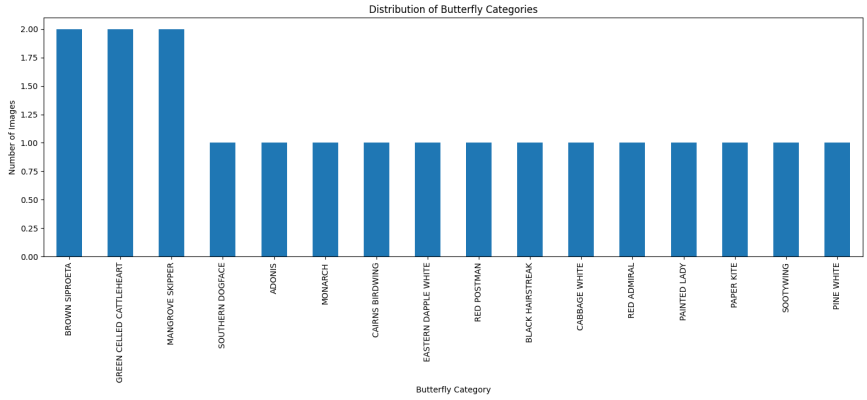
```
from google.colab import drive

drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.applications import ResNet50
from tqdm import tqdm
from sklearn.model_selection import train_test_split
import cv2
import shutil
import time
from sklearn.metrics import classification_report


# Path dataset
train_csv = "/content/drive/MyDrive/butterfly/Training_set.csv"
train_folder = "/content/drive/MyDrive/butterfly/train"

test_csv = "/content/drive/MyDrive/butterfly/Testing_set.csv"
test_folder = "/content/drive/MyDrive/butterfly/Testing_set.csv"

## Result path
result_path = f"/kaggle/working/run/"
os.makedirs(result_path, exist_ok=True)

checkpoint_path = os.path.join(result_path, "best_model.h5")
loss_image_path = os.path.join(result_path, 'validation loss.png')
acc_image_path = os.path.join(result_path, 'validation accuracy.png')
confusion_image_path = os.path.join(result_path, 'confusion matrix.png')


train_df = pd.read_csv(train_csv)
test_df = pd.read_csv(test_csv)
```

```
plt.figure(figsize=(15, 7))
train_df['label'].value_counts().plot(kind='bar')
plt.title('Distribution of Butterfly Categories')
plt.xlabel('Butterfly Category')
plt.ylabel('Number of Images')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



```
## Hyperparameter
image_size = (150, 150)
batch_size = 32
epochs = 100
learning_rate = 0.0001


class_name = list(set(train_df['label']))
print(class_name)
```

```
['SOOTYWING', 'CABBAGE WHITE', 'RED ADMIRAL', 'PAINTED LADY', 'PAPER KITE', 'ADONIS', 'RED POSTMAN', 'CAIRNS BIRDWING', 'PINE WHITE
```

```
print(len(features))
print(len(labels))

features = np.asarray(features)
labels = np.asarray(labels)
```

```
9
9
```

```
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.3, shuffle=True, random_state=42)
X_valid, X_test, y_valid, y_test = train_test_split(X_test, y_test, test_size=0.5, shuffle=True, random_state=42)


del features
del labels


# Membuat model MobileNet
base_model = ResNet50(
    weights='imagenet',
    include_top=False,
    input_shape=(image_size[0], image_size[1], 3),
)

num_layers_to_train = int(np.ceil(0.2 * len(base_model.layers)))

for layer in base_model.layers[:num_layers_to_train] :
    layer.trainable = False

x = base_model.output
x = Flatten()(x)
x = Dense(256, activation='relu', kernel_regularizer='l2')(x)
predictions = Dense(75, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

model.summary()


model.compile(optimizer=Adam(learning_rate), loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=10)
model_checkpoint = ModelCheckpoint(checkpoint_path, monitor='val_loss', save_best_only=True)

# Hitung waktu training
start_time = time.time()

# Latih model dengan menggunakan model checkpoint
history = model.fit(
    X_train,
    y_train,
    epochs=epochs,
    validation_data = (X_valid,y_valid),
    callbacks=[model_checkpoint, early_stopping],
    batch_size = batch_size,
)

# Hitung waktu training
end_time = time.time()
```

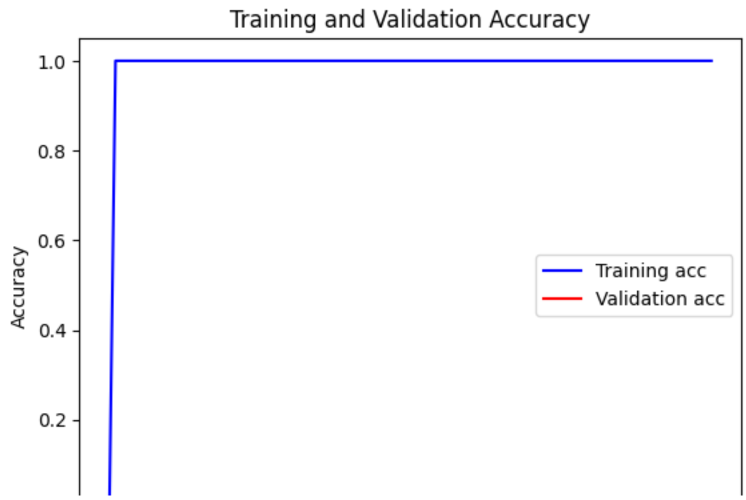
```
Epoch 90/100
1/1 [==============================] - 4s 4s/step - loss: 1.6928 - accuracy: 1.0000 - val_loss: 3.8371 - val_accuracy: 0.0000e+0
Epoch 91/100
1/1 [==============================] - 5s 5s/step - loss: 1.6819 - accuracy: 1.0000 - val_loss: 3.8140 - val_accuracy: 0.0000e+0
Epoch 92/100
1/1 [==============================] - 5s 5s/step - loss: 1.6712 - accuracy: 1.0000 - val_loss: 3.7894 - val_accuracy: 0.0000e+0
Epoch 93/100
1/1 [==============================] - 4s 4s/step - loss: 1.6607 - accuracy: 1.0000 - val_loss: 3.7642 - val_accuracy: 0.0000e+0
Epoch 94/100
1/1 [==============================] - 7s 7s/step - loss: 1.6505 - accuracy: 1.0000 - val_loss: 3.7394 - val_accuracy: 0.0000e+0
Epoch 95/100
1/1 [==============================] - 4s 4s/step - loss: 1.6404 - accuracy: 1.0000 - val_loss: 3.7093 - val_accuracy: 0.0000e+0
Epoch 96/100
1/1 [==============================] - 4s 4s/step - loss: 1.6306 - accuracy: 1.0000 - val_loss: 3.6822 - val_accuracy: 0.0000e+0
Epoch 97/100
1/1 [==============================] - 5s 5s/step - loss: 1.6210 - accuracy: 1.0000 - val_loss: 3.6490 - val_accuracy: 0.0000e+0
Epoch 98/100
1/1 [==============================] - 4s 4s/step - loss: 1.6115 - accuracy: 1.0000 - val_loss: 3.6178 - val_accuracy: 0.0000e+0
Epoch 99/100
1/1 [==============================] - 5s 5s/step - loss: 1.6023 - accuracy: 1.0000 - val_loss: 3.5739 - val_accuracy: 0.0000e+0
Epoch 100/100
1/1 [==============================] - 5s 5s/step - loss: 1.5932 - accuracy: 1.0000 - val_loss: 3.5167 - val_accuracy: 0.0000e+0
```

```
print("Training Time", end_time - start_time)
```

```
Training Time 582.6654005050659
```

```
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(loss)+1)
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.savefig(loss_image_path)
plt.show()
```


```
acc = history.history['accuracy']

val_acc = history.history['val_accuracy']

plt.plot(epochs, acc, 'b', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.savefig(acc_image_path)
plt.show()
```




```
y_pred = model.predict(X_test)

y_pred = np.argmax(y_pred, axis=1)

classification_rep = classification_report(y_test, y_pred)
print("Classification Report:\n", classification_rep)

classification_file = 'classification_report.txt'
with open(classification_file, 'w') as file:
    file.write(classification_rep)
```

```
1/1 [==============================] - 0s 211ms/step
Classification Report:
               precision    recall  f1-score   support

           5       0.00      0.00      0.00       0.0
           8       0.00      0.00      0.00       1.0
          12       0.00      0.00      0.00       0.0
          14       0.00      0.00      0.00       1.0

    accuracy                           0.00       2.0
   macro avg       0.00      0.00      0.00       2.0
weighted avg       0.00      0.00      0.00       2.0

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Recall and F-score are ill
  _warn_prf(average, modifier, msg_start, len(result))
```