KLASIFIKASI GAMBAR MAKANAN KHAS INDONESIA BERBASIS CNN: STUDI KASUS PENGOLAHAN DATA BESAR MENGGUNAKAN EFFICIENTNETB0

LAPORAN UJIAN TENGAH SEMESTER

(UTS)

Dosen Pengampu:

Dr. Tukiyat, M.Si

Oleh:

ASEP RIDWAN HIDAYAT

231012050036

PROGRAM STUDI TEKNIK INFORMATIKA S-2

PROGRAM PASCASARJANA

UNIVERSITAS PAMULANG

TANGERANG SELATAN

2025

DAFTAR ISI

BAB I PENDAHULUAN

1.1 Latar Belakang

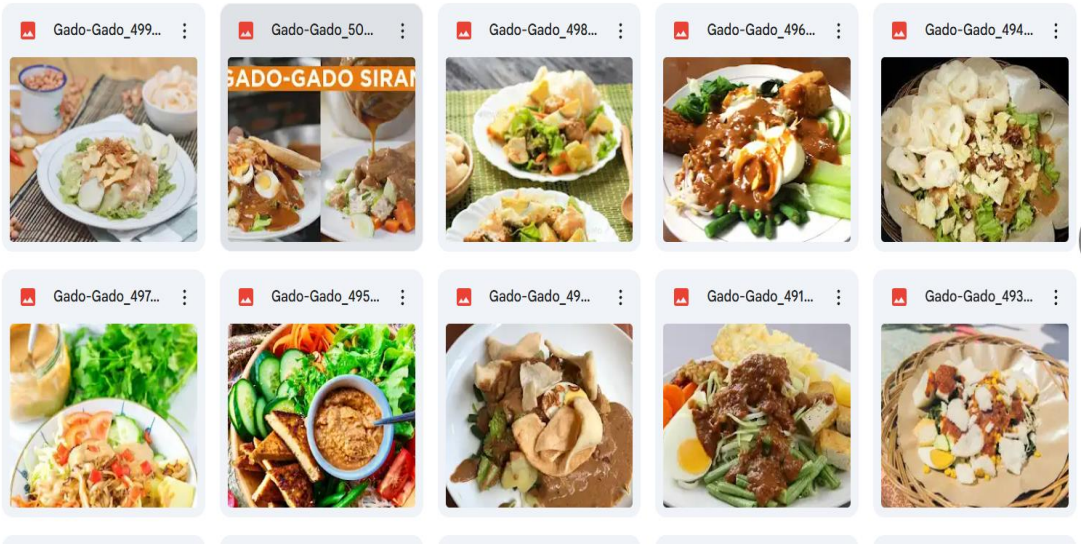
Pengolahan data gambar dalam skala besar seperti pengolahan data ini sangat relevan dengan mata kuliah Analisis Big Data dan Analisis, selain untuk pemenuhan ujian tengah semester (UTS) tetapi sekaligus pembelejaran bagi penulis untuk penerapan mata kuliah computer vision. Pengolahan data kali ini mencakup proses ekstraksi informasi dari dataset besar, pengklasifikasian data dengan model yang terbentuk sekaligus prediksi keakuratan model dengan menggunakan dataset test, dan evaluasi performa model secara sistematis.

Pengklasifikasian data menggunakan algoritma CNN dengan salah sati librarynya yaitu efficientnetb0.

1.2 Deskripsi Dataset

Dataset yang digunakan untuk pengolafhan data UTS ini adalah "Dataset Food Classification" yang bersumber dari situs Kaggle (https://www.kaggle.com/datasets/rizkyyk/dataset-food-classification). Tetapi tidak semua data yang diproses, hanya beberapa saja karena keterbatasan prasarana hadware, mesti membutuhkan GPU tinggi untuk melakukan klasifikasi gambar ini jika dengan keseluruhan data set.

Dataset ini berisi gambar berbagai jenis makanan khas Indonesia, seperti bakso, sate, nasi goreng, ayam goreng, dan lain-lain. Dari sumbernya dataset terdiri dari total lebih dari 6.500 gambar tetapi yang diambil untuk pengolahan data kali ini hanya 5.500 dengan 13 kelas makanan berbeda yang diambil. Gambar-gambar tersebut memiliki resolusi bervariasi dan berada dalam folder sesuai dengan label kelas masing-masing.



Gambar 1. Contoh gambar data set kelas Gado-gado

Berikut 13 kelas dari dataset yang diambil:

1. Gado-gado
2. Rawon
3. Pizza
4. Burger
5. Ikan Goreng
6. Soto
7. Ayam Goreng
8. Mie Goreng
9. sate
10. French fries
11. Rendang
12. Nasi padang
13. Nasi Goreng

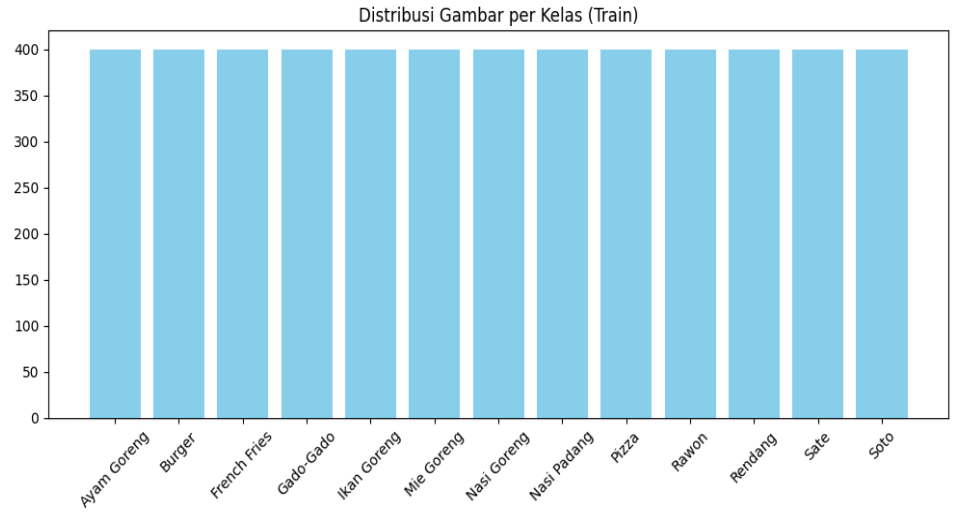
1.2.1 Preposesing Data

Preprosesing yang diterapkan adalah mengambil data dari 6500 data menjadi 5500 data disesuaikan denga resource yang ada. Kemudian dilakukan pemilihan gambar yang resolusi yang paling tinggi dari gambar lainnya. dilakukan pendistribusian data menjadi dua bagian, yaitu data train sebanyak 4158 gambar dan data test 1366 gambar

Berikut sumber dataset di googledrive yang sudah diambil dari kaggle dan dilakukan preproseis:

https://drive.google.com/drive/folders/1toVs4itO13WuOlGW4eUR5dVxc7YkNu_J?usp=sharing

Berikut pendistribusian gambar data train seperti diagram dibawah ini.



Gambar 2 Distribusi Gambar Train

1.3 Tujuan dan Manfaat Penelitian

Tujuan dari pengolahan data ini adalah untuk membangun sebuah model klasifikasi gambar makanan menggunakan arsitektur *Convolutional Neural Network* (CNN), khususnya dengan memanfaatkan model transfer learning EfficientNetB0.

Tujuan spesifiknya antara lain:

1. Melatih model untuk mengenali berbagai jenis makanan dari gambar.
2. Mengevaluasi performa model menggunakan metrik akurasi, loss, dan confusion matrix.
3. Menganalisis hasil pengolahan data untuk mengukur efektivitas model.

BAB II PEMBAHASAN DAN ANALISA

Model yang digunakan adalah EfficientNetB0, sebuah arsitektur CNN yang dikembangkan oleh Google dengan efisiensi dan akurasi tinggi. Model ini digunakan sebagai feature extractor dengan menambahkan beberapa layer klasifikasi pada bagian atasnya.

Data di-load menggunakan *ImageDataGenerator*, kemudian dilakukan augmentasi sederhana (rotation, flip, zoom).

Berikut adalah hasil pengolahan data di googlecolab:
https://colab.research.google.com/drive/1De-Uz2mTH7WasQ_33nr3zSIfGOLRdmOd?usp=sharing

2.1 Analisa dan Hasil Pengamatan

2.1.1 Pengamatan Epoch pertama dengan 10 epoch

Epoch adalah satu putaran penuh di mana seluruh data pelatihan digunakan sekali untuk melatih model. Berikut tabel hasil training model dengan epoch 10

Epoch	Train Accuracy	Train Loss	Val Accuracy	Val Loss
1	0.0771	2.5846	0.0755	2.5647
2	0.0733	2.5650	0.0816	2.5648
3	0.0741	2.5651	0.0755	2.5649
4	0.0789	2.5651	0.0755	2.5648
5	0.0820	2.5650	0.0755	2.5649
6	0.0696	2.5652	0.0816	2.5648
7	0.0746	2.5651	0.0816	2.5647
8	0.0684	2.5651	0.0755	2.5650
9	0.0760	2.5651	0.0544	2.5651
10	0.0779	2.5650	0.0816	2.5650

Tabel 2.1 Percoaan train epoch 10 pada model

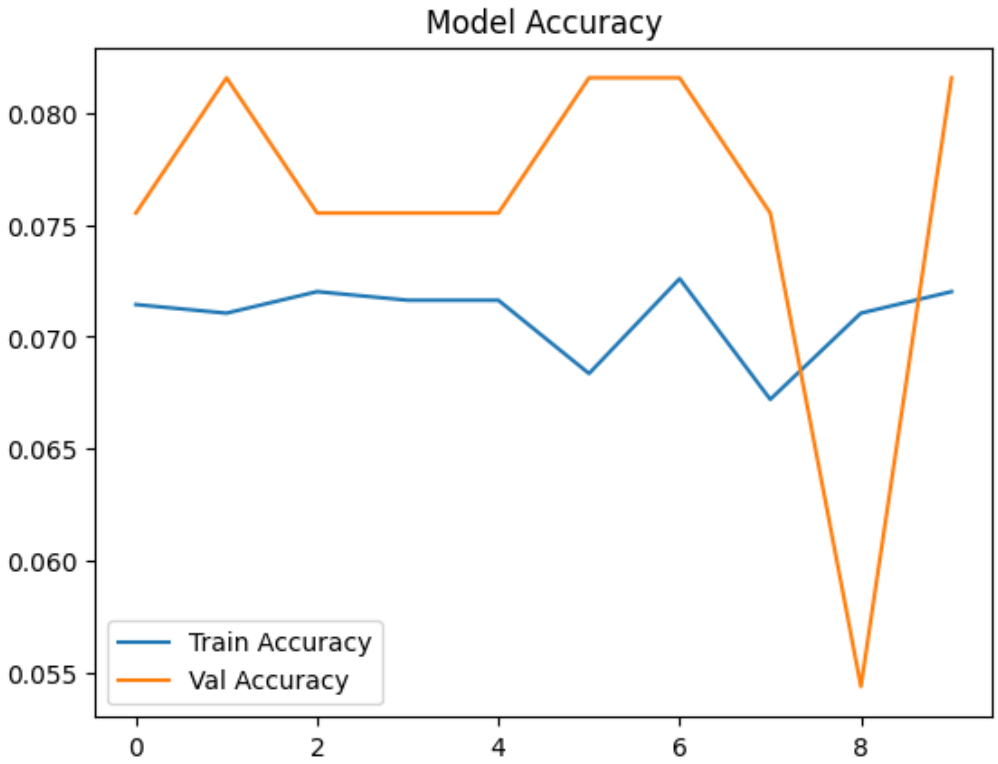
Dari tabel diatas terlihat **Train Accuracy (akuran training)** sebesar 0.0771 dengan Loss 2.5846 artinya pada epoch pertama modelnya hanya mempunyai akurasi

7% dan kemampuan generalasi model juga masih buruk. Dengan bertambah nya epoch model ada peningkatan tapi sangat rendah sampai epoch ke 10 hanya 0.0779.

Dari percobaan pertama ini dapat disimpulkan sebagai berikut:

- Akurasi Training hanya 7.7% artinya model tidak belajar pengklasifikasian dengan akurat
- Akurasi Validasi tidak stabil hanya 8%
- Loss tidak berubah signifikan

Jika digambarkan dengan grafik seperti berikut:



Gambar 2.1 Model acuracy pada epoch 10

Dari Gambar 2.1 Model akurasi cendrung tidak ada perubahan dan vaidasi nya sangat tidak stabil.

2.1.2 Pengamatan Epoch Kedua dengan 20 epoch

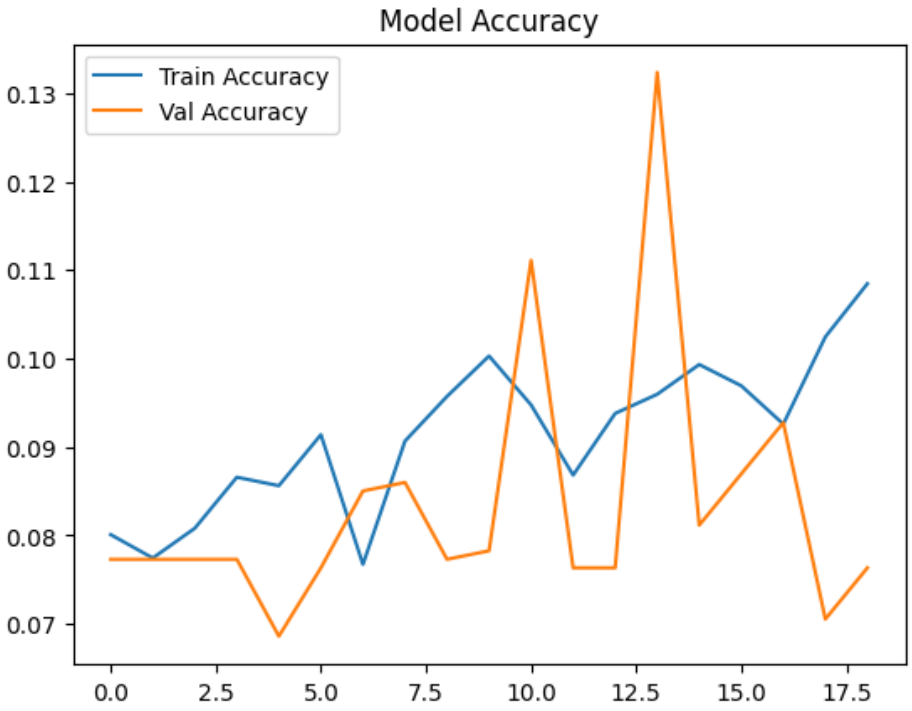
Dari beberapa sumber akurasi train model bisa ditingkatkan salahsatunya dengan meningkatkan jumlah training model yang ditambah. Berikut percobaan pengamatan dengan 20 epoch dengan waktu compile sekitar 4 jam.

Epoch	Train Accuracy	Train Loss	Val Accuracy	Val Loss
1	0.0777	2.6172	0.0773	2.5793
2	0.0786	2.5876	0.0773	2.5688
3	0.0779	2.5819	0.0773	2.5681
4	0.0900	2.5653	0.0773	2.5698
5	0.0887	2.5690	0.0686	2.5603
6	0.0963	2.5592	0.0763	2.5701
7	0.0837	2.5592	0.0850	2.5605
8	0.0942	2.5524	0.0860	2.5704
9	0.0973	2.5472	0.0773	2.5582
10	0.0992	2.5517	0.0783	2.5744
11	0.0904	2.5483	0.1111	2.5269
12	0.0868	2.5387	0.0763	2.5880
13	0.0996	2.5377	0.0763	2.7776
14	0.0976	2.5326	0.1324	2.5114
15	0.0995	2.5415	0.0812	2.6274
16	0.0917	2.5374	0.0870	2.5305
17	0.0916	2.5433	0.0928	2.5301
18	0.0980	2.5351	0.0705	2.5601
19	0.1083	2.5289	0.0763	2.5741

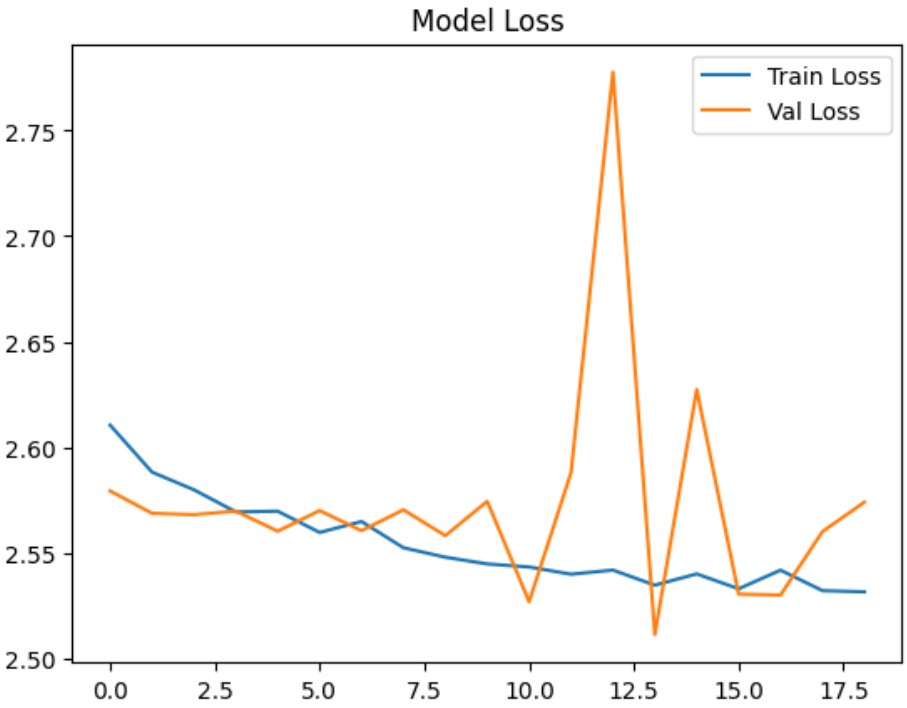
Tabel 2.2 Train model dengan 20 eporch

Dari tabel diatas **dapat dilihat ada kenaikan akurasi disetiap epoch sampai eporch ke 19 dengan nilai akurasi 10.8% (0.1083) dengan validasi model sekitar 7% (0.076) tentunya ini masih jauh dari model yang maksimal**. Tetapi ada peningkatan pada tahap training dengan eporch 10 dan 20

Grafik dari pengamatan kedua dengan jumlah train model sebanyak 20 perulangan berikut grafiknya.



Gambar 2.2 Model akuracy dengan epoc 20



Gambar 2.3 Train Los dengan epoch 20

Dari gambar 2.2 train model acuracy bisa terlihat:

- Nilai akurasi di data pelatihan maupun validasi cenderung rendah (sekitar 0.07–0.13), dan fluktuatif terutama pada data validasi.
- Akurasi validasi tidak stabil, bahkan sering kali lebih rendah dari akurasi pelatihan, serta mengalami lonjakan dan penurunan yang tajam.

Dari Gambar 2.3 train los terlihat bahwa:

- Nilai loss pada data pelatihan cenderung turun perlahan, yang berarti model sedikit demi sedikit belajar dari data pelatihan.
- loss pada data validasi sangat fluktuatif alias naik turun, bahkan ada lonjakan yang sangat tinggi pada epoch tertentu.
- Fluktuasi besar pada loss validasi menunjukkan model tidak generalisasi dengan baik ke data baru, kemungkinan besar karena data validasi tidak cukup representatif atau model mengalami masalah seperti underfitting atau data yang terlalu sedikit.

2.1.3 Pengamatan Uji model dalam prediksi gambar

Berikut ini adalah hasil output dari pengolahan model dalam memprediksi gambar pada data testing, dengan akurasi yang belum bagus epoch 20 nilai akurasi 10.8 %, berikut hasil grafik dan nilai precision recall dan f1-scorenya.

2.1.3.1 Nilai Precision, recall, f1-score

	precision	recall	f1-score	support
Ayam Goreng	0.00	0.00	0.00	18
Burger	0.14	0.44	0.22	18
French Fries	0.70	0.26	0.38	27
Gado-Gado	0.00	0.00	0.00	27
Ikan Goreng	0.08	0.19	0.11	27
Mie Goreng	0.00	0.00	0.00	27
Nasi Goreng	0.00	0.00	0.00	27
Nasi Padang	0.00	0.00	0.00	27
Pizza	0.00	0.00	0.00	27
Rawon	0.18	0.16	0.17	25
Rendang	0.07	0.07	0.07	27
Sate	0.10	0.56	0.17	27
Soto	0.00	0.00	0.00	27
accuracy			0.12	331
macro avg	0.10	0.13	0.09	331

Gambar 2.4 Hasil Precision recall f1-score dan support

Dari hasil output diatas dapat diliha nilai precisi nya tidak bagus dengan nilai **precision** gambar burger kesetiap kelas hanya 0.14, frenc fries 0.70 ,rawon 0.18 rendang 0.07 sate 0.10 dan yang lainnya model tidak bisa memprediksi gambar makanan tersebut.

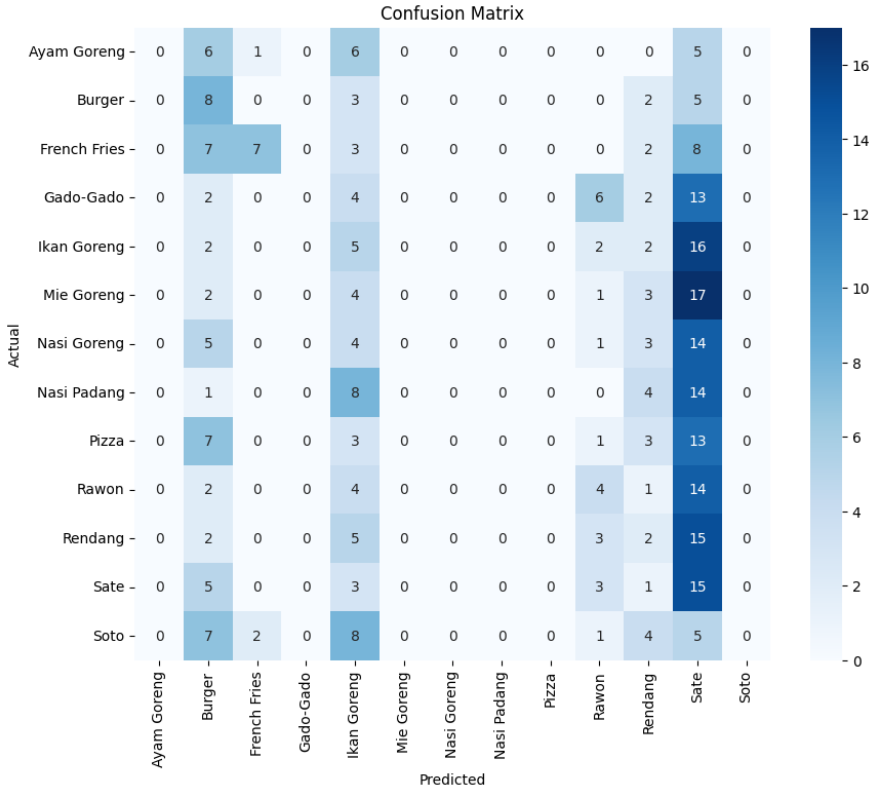
Dari gambar 2.4 secara gari besar dapat disimpulkan, sebagai berikut:

1. kurasi keseluruhan: 12% → model hanya benar dalam 12 dari 100 prediksi.
2. Macro average (rata-rata semua kelas):
3. Nilai Precision: 0.10
4. Recall: 0.13
5. F1-score: 0.09

Dan hal ini menunjukkan bahwa model sangat lemah di semua kelas.

2.1.3.2 Confusion Matrik Dan Intepretasinya

Jika kita visualisasikan model predeksi diatas dengan grafik confusion matrix maka hasilnya sebagai berikut



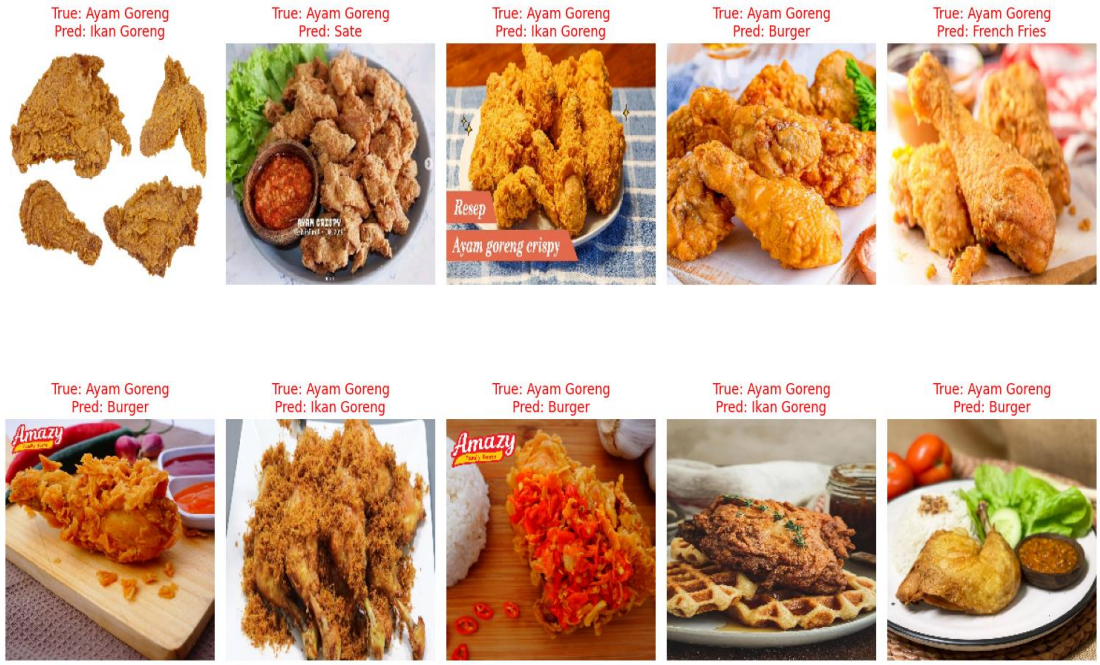
Gambar 2.5 Confusion Matrik Train model Klasifikasi gambar food

Dari hasil confusion matrix bisa kita lihat model tidak bagus dalam prediksi, kita ambil contoh :

- burger, model hanya bisa memprediksi dengan benar (actual) itu 8 dari 18 gambar sisanya diprediksi ikan goreng 3 redang 2 sate 5.
- French fries, model hanya bisa memprediksi dengan benar sebanyak 7, sisanya 7 burger 3 ikan goreng 2 rendang 8 sate
- Sate, model hanya bisa memprediksi dengan benar sebanyak 15, selebihnya diaggap burger 5, ikan goreng dan rawon 3 dan rendan 1 dan sampai seterusnya.

2.1.4 Hasil Model Prediksi Gambar Food

Jika divisualisasikan prediksi model terhadap gambar food dengan jumlah data test seperti ini:



Gambar 2.5 Output Prdeiksi model

Dari gambar diatas terlihat jelas bahwa **model masih sangat jauh dari bagus (tidak bagus)**, karena dengan acurasy model hanya 10 % dan val loss nya yang sangat besar.

BAB III KESIMPULAN

Pengolahan data klasifikasi gambar makanan menggunakan CNN dengan arsitektur EfficientNetB0 berhasil dilakukan dengan hasil:

3.1.1 Percobaan Pertama (10 Epoch)

- Train Accuracy awal–akhir: 7.7% → 7.8%
- Val Accuracy awal–akhir: 7.6% → 8.1% (cenderung stagnan)
- Loss sangat stabil, nyaris tidak berubah (~2.5650) dari epoch 2 sampai 10

  Dengan kesimpulan Tidak ada tanda-tanda model belajar dari data

3.1.2 Percobaan Kedua (20 epoch)

- Train Accuracy awal–akhir: 7.8% → 10.8%
- Val Accuracy awal–akhir: 7.7% → 7.6% (sempat naik hingga 13.2%)
- Loss menunjukkan sedikit penurunan namun fluktuatif

  Ada sedikit sinyal perbaikan, tapi tidak konsisten (indikasi learning lambat dan model tidak stabil)

Dengan Accuracy hanya 12% dengan pengertian model hanya benar dalam 12 dari setiap 100 prediksi. F1-score per kelas mayoritas di bawah 0.2. artinya model tidak bagus pada epoch 20 perulangan, tetapi ada peningkatan train epoch 10 ke 20 meskipun tidak signifikan, artinya model masih bisa belajar dalam prediksi gambar.

3.1.3 Analisa dari Pemodelan dan saran

Dari hasil pemodelan ada beberapa faktor dari sumber yang didapat ketika nilai acuracy kurang baik.

1. Perlu pengulangan train model dengan epoch yang lebih besar
2. Evaluasi jumlah dan keseimbangan kelas apakah jumlah kelas terlalu banyak? Apakah ada kelas yang mendominasi
3. Coba learning rate lebih besar: karena loss stagnan, bisa jadi model stuck di local minimum atau learning rate terlalu kecil.

4. Mencoba algoritma dan arsitek yang berbeda bisa dilakukan jika ingin model lebaik baik

SCRIPT dari Pengolahan Model

```
# -*- coding: utf-8 -*-
"""BIG DATA DAN ANALISIS ASEP RIDWAN .ipynb

Automatically generated by Colab.

Original file is located at
                                    https://colab.research.google.com/drive/1De-Uz2mTH7WasQ_33nr3zSIfGOLRdmOd
"""

!pip install tensorflow

# akses ke google drive untuk mengambil dataset
from google.colab import drive
drive.mount('/content/drive')

# mengambil dataset
import os

base_path = '/content/drive/MyDrive/dataset-food-classification'
train_path = os.path.join(base_path, 'train')
test_path = os.path.join(base_path, 'test')

"""Prpocesing data"""

import matplotlib.pyplot as plt

def count_images(data_path):
    class_counts = {}
    for class_name in sorted(os.listdir(data_path)):
        class_dir = os.path.join(data_path, class_name)
        if os.path.isdir(class_dir):
```

```
            count = len(os.listdir(class_dir))
            class_counts[class_name] = count
    return class_counts

train_class_counts = count_images(train_path)
test_class_counts = count_images(test_path)

# Visualisasi training
plt.figure(figsize=(10, 5))
plt.bar(train_class_counts.keys(), train_class_counts.values(), color='skyblue')
plt.title('Distribusi Gambar per Kelas (Train)')
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

from tensorflow.keras.preprocessing.image import ImageDataGenerator

img_size = (224, 224)
batch_size = 32

# train_datagen = ImageDataGenerator(
#     rescale=1./255,
#     rotation_range=20,
#     zoom_range=0.2,
#     horizontal_flip=True,
#     validation_split=0.2
# )

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
```

```
    zoom_range=0.3,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    train_path,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
    train_path,
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)

test_datagen = ImageDataGenerator(rescale=1./255)
test_generator = test_datagen.flow_from_directory(
    test_path,
    target_size=img_size,
    batch_size=1,
    class_mode='categorical',
    shuffle=False
)

from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras import layers, models
from tensorflow.keras.optimizers import Adam
```

```
base_model = EfficientNetB0(weights='imagenet', include_top=False, input_shape=(224,224,3))
base_model.trainable = True  # fine-tune

# Fine-tune hanya sebagian besar layer terakhir
for layer in base_model.layers[:-50]:
    layer.trainable = False

model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(256, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(train_generator.num_classes, activation='softmax')
])

model.compile(optimizer=Adam(learning_rate=1e-4),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.summary()

from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=20,
    callbacks=[early_stop]
)
```

```
# Akurasi
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.legend()
plt.title('Model Accuracy')
plt.show()

# Loss
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.legend()
plt.title('Model Loss')
plt.show()

import numpy as np
from sklearn.metrics import classification_report, confusion_matrix
import seaborn as sns

# Prediksi data test
pred_probs = model.predict(test_generator)
pred_labels = np.argmax(pred_probs, axis=1)
true_labels = test_generator.classes
class_names = list(test_generator.class_indices.keys())

# Confusion matrix
cm = confusion_matrix(true_labels, pred_labels)

plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names,
yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
```

```
plt.show()

# Laporan klasifikasi
print(classification_report(true_labels, pred_labels, target_names=class_names))

"""Prediksi dan Visualisasi Beberapa Gambar Test"""

plt.figure(figsize=(15, 10))
inv_class_indices = {v: k for k, v in test_generator.class_indices.items()}

for i in range(10):
    img, _ = test_generator[i]
    pred = model.predict(img)
    pred_label = inv_class_indices[np.argmax(pred)]
    true_label = inv_class_indices[test_generator.classes[i]]

    plt.subplot(2, 5, i+1)
    plt.imshow(img[0])
    plt.axis('off')
    color = 'green' if pred_label == true_label else 'red'
    plt.title(f"True: {true_label}\nPred: {pred_label}", color=color)

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