LAPORAN UAS DEEP LEARNING

KLASIFIKASI JENIS SAMPAH MENGGUNAKAN EFFICIENTNETV2B0

DISUSUN OLEH :

Gopi Mahendra G1A021005

Syakira Az Zahra G1A021057

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS BENGKULU

2024

A. Business Understanding

Sampah merupakan salah satu isu global yang semakin penting untuk ditangani, terutama dalam hal pengelolaan sampah yang ramah lingkungan dan efisien. Pemisahan jenis sampah (misalnya plastik, organik, kertas) adalah langkah penting dalam mendukung daur ulang dan pengelolaan sampah yang lebih baik. Namun, proses pemisahan sampah secara manual memerlukan waktu dan tenaga yang banyak, serta seringkali mengarah pada kesalahan manusia. Masalah sampah telah menjadi isu besar di berbagai daerah di Indonesia. Peningkatan jumlah penduduk dan aktivitas manusia yang tidak diimbangi dengan perbaikan sistem pengelolaan sampah di banyak tempat telah menyebabkan penumpukan sampah di area yang tidak semestinya. Sampah yang tidak dikelola dengan baik ini dapat menjadi sumber penyakit serta merusak keindahan dan kenyamanan lingkungan sekitar (Rapii et al., 2021).

Masalah pengelolaan sampah di Indonesia seringkali diperburuk oleh kesulitan dalam mengenali dan memilah jenis sampah, terutama bagi sebagian besar masyarakat yang kurang teredukasi mengenai cara pemilahan sampah yang benar. Banyak orang kesulitan membedakan jenis sampah, seperti plastik, kertas, kaca, atau sampah organik, karena adanya kesamaan bentuk atau warna antara jenis sampah yang berbeda. Hal ini menyebabkan sampah sering tercampur dalam satu tempat pembuangan, yang pada akhirnya memperburuk proses daur ulang dan pengolahan sampah.

Di sinilah teknologi deep learning dapat memberikan solusi yang efektif. Dengan menggunakan model deep learning untuk klasifikasi sampah, proses pemilahan dapat dilakukan secara otomatis dan lebih akurat.

B. Data Understanding

Dataset merupakan hal yang sangat penting dalam sebuah penelitian baik di bidang Informatika ataupun bidang lainnya. Pada kasus ini dataset jenis sampah dikumpulkan dengan mengambil berbagai gambar dari internet sebanyak 50 pada masing-masing kelas. Dikarenakan gambar yang dikumpulkan dari berbagai macam sumber, alhasil ukuran dari masing-

masing gambar tidak tentu. Kondisi ini mengharuskan penulis untuk melakukan proses rescale pada setiap data dengan

```
img_height = 224
img_width = 224
```

Fungsi ini akan menjadikan gambar pada dataset memiliki ukuran 224 x 224 px. Kemudian dataset akan di augmentasi, augmentasi dibutuhkan untuk menyesuaikan data dengan kondisi lingkungan yang beragam dan meningkatkan variasi data, sehingga model dapat mempelajari pola dengan lebih efektif.

No	Kelas	Jumlah Sebelum Augmentasi	Jumlah Setelah Augmentasi
1	Kaca	50	300
2	Kardus dan Kertas	50	300
3	Plastik	50	300
		Total	900

C. Data Preparation

- **Augmentasi**

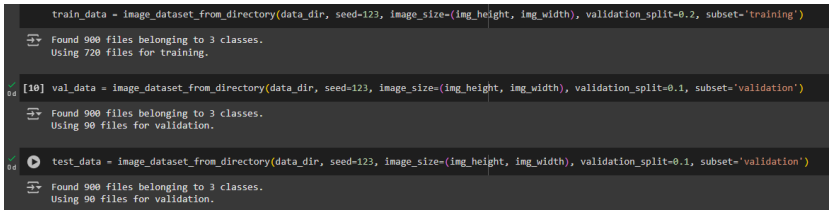
```
aug = iaa.Sequential([
    iaa.Fliplr(0.5),  # Horizontal flip
    iaa.Crop(percent=(0, 0.1)),  # Crop gambar secara acak
    iaa.LinearContrast((0.75, 1.5)),  # Ubah kontras gambar
    iaa.Multiply((0.8, 1.2)),  # Ubah kecerahan gambar
    iaa.Affine(
        scale={"x": (0.8, 1.2), "y": (0.8, 1.2)},  # Skala gambar
        translate_percent={"x": (-0.2, 0.2), "y": (-0.2, 0.2)},  # Translasi gambar
        rotate=(-25, 25),  # Rotasi gambar
        shear=(-8, 8)  # Shear gambar
    ),
    iaa.Sometimes(0.1, iaa.AdditiveGaussianNoise(scale=(10, 60))),  # Tambahkan Gaussian noise dengan probabilitas 30%
    iaa.Sometimes(0.1, iaa.SaltAndPepper(0.05))  # Tambahkan salt and pepper noise dengan probabilitas 30%
], random_order=True)
```

Pada tahap ini, penulis melakukan 7 parameter augmentasi.

Proses
Flip Horizontal (50%)
Crop (0%-10%)
Contrast (0.75-1.5)
Brightness (0.8-1.2)
Transformation
Gaussian Noise
Salt and Paper noise

Secara keseluruhan, teknik augmentasi ini membantu dalam menciptakan variasi dalam dataset yang kecil, memperluas dataset dengan cara yang bermakna, dan meningkatkan kemampuan generalisasi model.

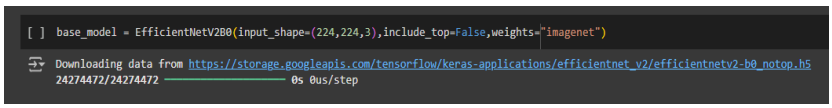
- **Split Data**



Kode diatas menggunakan fungsi image_dataset_from_directory untuk memuat gambar dari folder yang penulis gunakan (data_dir) pada kasus ini, dataset disimpan di google drive. Gambar-gambar diubah ukurannya menggunakan (img_height, img_width) agar sama, diacak dengan seed 123 untuk konsistensi, lalu dibagi menjadi data pelatihan (80%) dan validasi (20%) menggunakan validation_split=0.2. Bagian data yang dimuat ditentukan dengan subset='training' dan disimpan ke variabel train_data untuk pelatihan. Hal yang sama dilakukan untuk data validasi dan data tes.

D. Modeling

- **Pre-trained**



Kode tersebut membuat sebuah model dasar menggunakan arsitektur EfficientNetV2B0. input_shape diguanakn untuk menentukan bentuk input gambar yang diharapkan oleh model, yaitu gambar dengan ukuran 224x224 piksel dan 3 saluran warna (RGB). include_top digunakan menunjukkan bahwa lapisan atas dari model tidak disertakan. Dengan tidak menyertakan lapisan atas, kita bisa menambahkan lapisan khusus untuk tugas kita sendiri.

- **Fine Tuning**

```
model=Sequential(name='EfficientNetV2B0')
model.add(base_model)
model.add(MaxPooling2D())
model.add(Dropout(0.5))
model.add(Flatten())
model.add(BatchNormalization())
model.add(Dense(128,kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(10,activation='softmax'))

model.summary()
```

```
Model: "EfficientNetV2B0"
```

Layer (type)	Output Shape	Param #
efficientnetv2-b0 (Functional)	(None, 7, 7, 1280)	5,919,312
max_pooling2d (MaxPooling2D)	(None, 3, 3, 1280)	0
dropout (Dropout)	(None, 3, 3, 1280)	0
flatten (Flatten)	(None, 11520)	0
batch_normalization (BatchNormalization)	(None, 11520)	46,080
dense (Dense)	(None, 128)	1,474,688
batch_normalization_1 (BatchNormalization)	(None, 128)	512
activation (Activation)	(None, 128)	0
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 10)	1,290

```
Total params: 7,441,882 (28.39 MB)
Trainable params: 7,357,978 (28.07 MB)
Non-trainable params: 83,904 (327.75 KB)
```

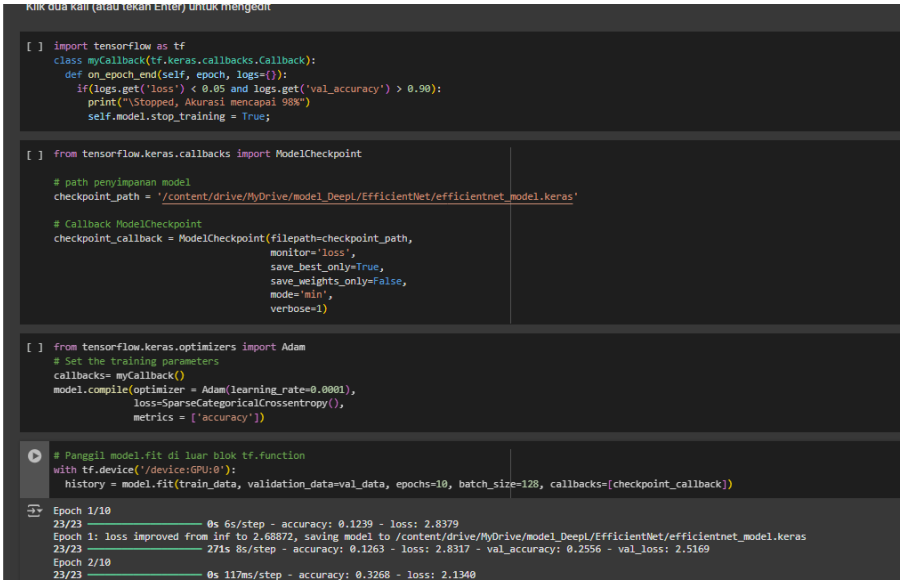
Model ini dibuat menggunakan arsitektur Sequential dengan memanfaatkan **EfficientNetV2B0** sebagai fitur utama untuk klasifikasi. Setelah itu, ditambahkan **MaxPooling2D** untuk mengurangi dimensi fitur dan **Dropout** untuk mencegah overfitting. **Flatten** berfungsi untuk mengubah data multidimensional menjadi satu dimensi, sehingga dapat diproses oleh lapisan fully connected. Untuk meningkatkan stabilitas pelatihan, lapisan **BatchNormalization** digunakan, diikuti dengan lapisan dense berisi 128 neuron yang memakai fungsi aktivasi **ReLU**. Dropout ditambahkan untuk meningkatkan generalisasi. Pada akhirnya, lapisan **Dense** terakhir dengan 10 neuron dan aktivasi **softmax** menghasilkan probabilitas untuk setiap kelas. Code diatas merupakan proses Fine Turning, yang dimana menggunakan base_model, yang merupakan model pre-trained yang telah dilatih sebelumnya.

i. **Total params: 7,441,882 (28.39 MB)**. Jumlah total parameter (bobot dan bias) dalam model, mencakup semua parameter yang dapat dan tidak dapat dilatih. Ukurannya sekitar 28,39 MB.

ii. **Trainable params: 7,357,978 (28.07 MB)**. Parameter yang dapat dilatih selama pelatihan model, yaitu 7.357.978, dengan ukuran sekitar 28,07 MB.

iii. **Non-trainable params: 83,904 (327.75 KB).** Parameter yang tidak dilatih, biasanya berasal dari model dasar pretrained, berjumlah 83.904 dan berukuran sekitar 327,75 KB.

- **Train Model**



Kode diatas bertujuan untuk melatih model deep learning menggunakan TensorFlow dengan menambahkan callback yang memungkinkan pelatihan model lebih efektif dan efisien. myCallback adalah kelas yang memiliki metode on_epoch_end, yang dipanggil setelah setiap epoch selesai. Terdapat logika untuk memeriksa apakah nilai loss lebih kecil dari 0.05 dan akurasi validasi (val_accuracy) lebih besar dari 90%. Jika kedua kondisi tersebut terpenuhi, pelatihan akan dihentikan lebih awal. Ini berguna untuk menghindari pelatihan berlebih (overfitting) dan menghemat waktu pelatihan.

Selain itu, callback ModelCheckpoint digunakan untuk menyimpan model terbaik selama proses pelatihan. Model akan disimpan ke dalam file yang terletak di checkpoint_path, yang diatur pada /content/drive/MyDrive/model_DeepL/EfficientNet/efficientnet_model

.keras. Setelah callback disiapkan, model dikompilasi menggunakan model.compile dengan optimizer Adam yang memiliki learning rate sebesar 0.0001. Fungsi loss yang digunakan adalah SparseCategoricalCrossentropy, yang merupakan pilihan umum untuk masalah klasifikasi multi-kelas di mana label kelas diberikan dalam format integer (bukan one-hot encoding).

Proses pelatihan dilakukan dengan memanggil model.fit, yang mengatur data pelatihan dan data validasi. train_data berisi data untuk melatih model, sedangkan val_data digunakan untuk memvalidasi model setelah setiap epoch. Pelatihan dilakukan selama 10 epoch dengan ukuran batch 128.

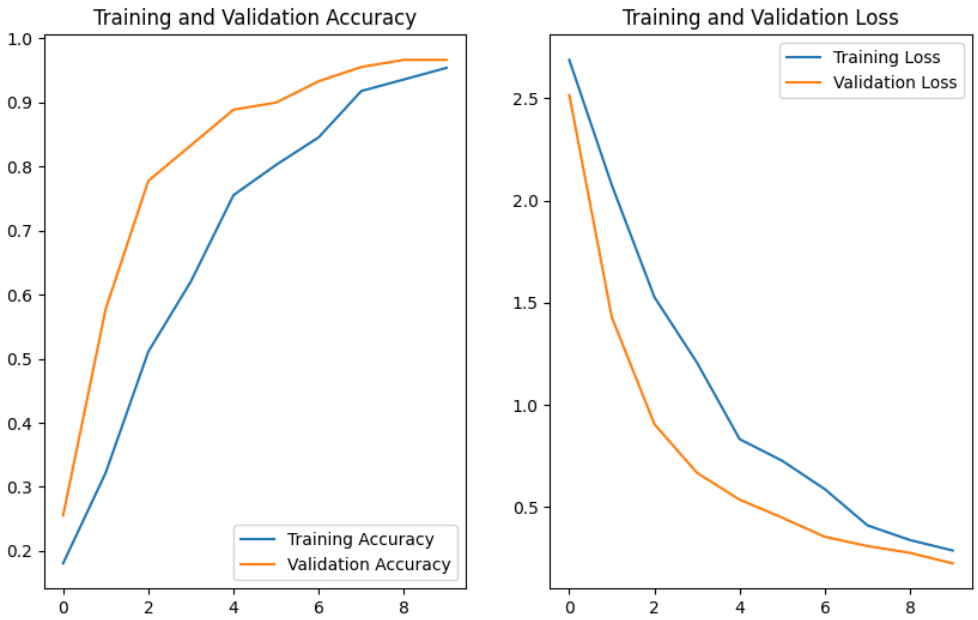
Pelatihan dilakukan di GPU, yang ditentukan dengan menggunakan konteks with tf.device('/device:GPU:0'). Ini bertujuan untuk mempercepat proses pelatihan, terutama ketika bekerja dengan model yang besar atau dataset yang kompleks, dengan memanfaatkan perangkat keras yang tersedia.

E. Evaluasi

```
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs_range = history.epoch

plt.figure(figsize=(10, 6))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

Kode diatas digunakan untuk menampilkan grafik hasil pelatihan model (training) dan validasi model dalam proses pelatihan. Kode ini membuat dua subplot: subplot pertama "Training and Validation Accuracy" menunjukkan grafik akurasi pelatihan dan validasi sepanjang epoch, dengan legenda yang membedakan keduanya. Subplot kedua "Training and Validation Loss" menunjukkan grafik kerugian pelatihan dan validasi dengan legenda yang membedakan keduanya. Dengan mengatur ukuran gambar dan menampilkan grafik menggunakan plt.show(), kode ini

bertujuan untuk memvisualisasikan dan membandingkan performa model selama pelatihan dan validasi.



Berdasarkan grafik di atas, akurasi pelatihan dan akurasi validasi menunjukkan kestabilan yang sangat baik tanpa adanya fluktuasi signifikan. Garis akurasi validasi tetap stabil tanpa perubahan besar setelah beberapa epoch pertama. Ini mengindikasikan bahwa model memiliki performa yang optimal dan memperoleh cukup data untuk mempelajari pola dengan efektif.

```
import numpy as np

predictions = model.predict(test_data)

y_pred = []
y_true = []

for image_batch, label_batch in test_data:
    y_true.append(label_batch)
    preds = model.predict(image_batch)
    y_pred.append(np.argmax(preds, axis = - 1))

true_labels = tf.concat([item for item in y_true], axis = 0)
predicted_labels = tf.concat([item for item in y_pred], axis = 0)
```

```
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 4s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
```

```
from sklearn.metrics import classification_report

print(classification_report(true_labels, predicted_labels, target_names=['kaca','kardus dan kertas','plastik']))
```

```
                  precision    recall  f1-score   support

            kaca       0.97      0.97      0.97        31
kardus dan kertas       0.97      1.00      0.99        35
         plastik       0.96      0.92      0.94        24

        accuracy                           0.97        90
       macro avg       0.97      0.96      0.96        90
    weighted avg       0.97      0.97      0.97        90
```

Kode diatas untuk melakukan prediksi pada data uji menggunakan model dan mengevaluasi kinerjanya. Pertama, model membuat prediksi

pada data uji, dan kemudian, dalam sebuah loop, untuk setiap batch gambar dan label, kode ini menyimpan label yang benar dan hasil prediksi model. Setelah itu, semua label yang benar dan hasil prediksi digabungkan menjadi satu tensor menggunakan tf.concat(). Untuk mengevaluasi hasilnya, kode ini menggunakan fungsi classification_report untuk menghasilkan laporan klasifikasi, yang mencakup metrik seperti precision, recall, F1-score, dan support untuk setiap kelas yang diuji, yakni 'kaca', 'kardus dan kertas', dan 'plastik'. Model menunjukkan performa yang sangat baik dengan akurasi 97%, F1-score tinggi untuk semua kelas, dan precision serta recall yang sangat stabil. Kelas 'kardus dan kertas' memiliki kinerja terbaik dengan recall sempurna (1.00). Model cukup baik dalam memprediksi kelas 'kaca' dan 'plastik' dengan F1-score yang sangat mendekati 1.

Analisa bagaimana model dapat dikatakan sebagai deep learning dan bukan shallow learn.

Shallow learning merupakan pembelajaran dalam Machine Learning yang merujuk pada penggunaan teknik pembelajaran mesin yang beroperasi pada fitur yang telah ditentukan dan dirancang secara manual serta algoritma yang relatif sederhana (Dwi Poetra, 2019). Model EfficientNetV2B0 termasuk dalam kategori deep learning karena menggunakan arsitektur jaringan saraf dalam (deep neural network) dengan banyak lapisan tersembunyi yang mampu belajar dari data secara bertahap. Berbeda dengan shallow learning yang hanya menggunakan beberapa lapisan dan terbatas dalam kemampuan untuk mengenali pola yang rumit. EfficientNetV2B0 adalah model berbasis convolutional neural network (CNN) yang dirancang untuk ekstraksi fitur dari gambar. Sebaliknya, shallow learning model lebih sederhana, biasanya hanya menggunakan beberapa lapisan dan tidak dapat menangkap representasi yang sangat kompleks dari data, yang membuatnya kurang efektif dalam menangani masalah seperti klasifikasi gambar yang memerlukan pemahaman hierarkis dan dalam terhadap data.

DAFTAR PUSTAKA

Dwi Poetra, R. (2019). BAB II Tinjauan Pustaka BAB II TINJAUAN PUSTAKA 2.1. 1–64. *Gastronomía Ecuatoriana y Turismo Local.*, *1*(69), 5–24.

Rapii, M., Majdi, M. Z., Zain, R., & Aini, Q. (2021). Pengelolaan Sampah Secara Terpadu Berbasis Lingkungan Masyarakat Di Desa Rumbuk. *Dharma Raflesia : Jurnal Ilmiah Pengembangan Dan Penerapan IPTEKS*, *19*(1), 13–22. https://doi.org/10.33369/dr.v19i1.13201