

LAPORAN UAS DEEP LEARNING
KLASIFIKASI GEDUNG BELAJAR UNIB



Disusun Oleh:

- | | |
|-------------------------------------|--------------------|
| 1. Ilham Dio Putra | (G1A021024) |
| 2. Arief Satrio Budi Prasajo | (G1A021076) |
| 3. Irfan Luthfi | (G1A021090) |

Dosen Mata Kuliah:

Arie Vatesia, S.T., M.T.I., Ph.D

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS BENGKULU

2024

A. Business Understanding

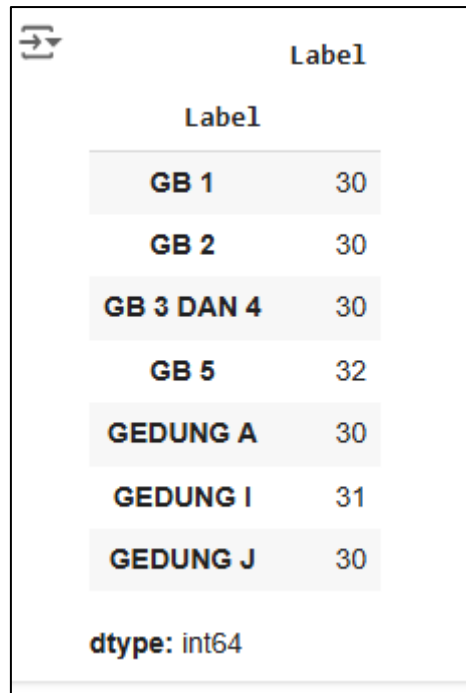
Universitas Bengkulu sebagai salah satu institusi pendidikan tinggi di Indonesia memiliki berbagai gedung belajar yang tersebar di area kampus. Namun, mahasiswa, dosen, dan pengunjung sering menghadapi kesulitan dalam mengenali atau membedakan gedung-gedung belajar, terutama bagi yang baru pertama kali berada di kampus atau saat ada perubahan tata ruang.

Untuk mengatasi tantangan ini, diperlukan pengembangan teknologi berbasis machine learning yang dapat mengklasifikasikan gedung belajar secara otomatis. Teknologi ini dirancang untuk mengenali gedung belajar berdasarkan data visual atau atribut lainnya, sehingga mempermudah pengguna dalam mendapatkan informasi yang diperlukan dengan cepat dan akurat.

Pengembangan model ini menggunakan algoritma seperti EfficientNetV2B3 yang terbukti efektif dalam memproses data visual. Dengan solusi ini, Universitas Bengkulu dapat meningkatkan efisiensi layanan informasi kampus dan memberikan pengalaman yang lebih baik bagi mahasiswa, dosen, maupun pengunjung.

B. Data Understanding

Dataset merupakan elemen penting dalam pengembangan model machine learning untuk klasifikasi gedung. Pada pengembangan ini, dataset gedung belajar dikumpulkan melalui pengambilan berbagai gambar di lingkungan kampus Universitas Bengkulu. Secara keseluruhan, dataset terdiri dari 243 gambar yang terbagi ke dalam beberapa kategori, yaitu GB 1 sebanyak 30 gambar, GB 2 sebanyak 30 gambar, GB 3 dan GB 4 masing-masing 30 gambar, GB 5 sebanyak 32 gambar, Gedung A sebanyak 30 gambar, Gedung I sebanyak 31 gambar, dan Gedung J sebanyak 30 gambar.



	Label
GB 1	30
GB 2	30
GB 3 DAN 4	30
GB 5	32
GEDUNG A	30
GEDUNG I	31
GEDUNG J	30

dtype: int64

Gambar 3. 1 Jumlah Dataset

C. Data Preparation

- **Resize**

Karena gambar-gambar ini diperoleh dari berbagai perangkat dan sudut pandang, ukuran setiap gambar tidak seragam. Untuk mengatasi hal tersebut, dilakukan proses rescale pada seluruh dataset dengan parameter

`img_height = 224`

`img_width = 224`

Langkah ini bertujuan untuk menyeragamkan dimensi gambar menjadi 224 x 224 piksel, sehingga mempermudah model dalam memproses data visual. Dataset yang telah disiapkan ini akan digunakan untuk melatih model klasifikasi gedung tanpa penambahan data melalui augmentasi pada tahap awal.

- **Split Data**

```
# Load dataset untuk pelatihan
train_data = image_dataset_from_directory(
    image_dir,
    validation_split=0.2,
    subset="training",
    seed=1,
    image_size=(img_height, img_width),
    batch_size=16
)

# Load dataset untuk validasi sekaligus testing
test_data = image_dataset_from_directory(
    image_dir,
    validation_split=0.2,
    subset="validation",
    seed=1,
    image_size=(img_height, img_width),
    batch_size=16
)

Found 213 files belonging to 8 classes.
Using 171 files for training.
Found 213 files belonging to 8 classes.
Using 42 files for validation.
```

Gambar 3. 2 Split Data

Setelah melakukan resize maka perlu dilakukan Split Data, yang ditunjukkan pada gambar 3. 2 dalam bentuk kode dan output. Kode tersebut melakukan pembagian gambar-gambar menjadi data training 80% dari total data (171 jumlah data) dan data testing 20% dari total data (42 jumlah data).

D. Modeling

```
base_model = EfficientNetV2B3(input_shape=(224,224,3),include_top=False,weights="imagenet")
```

Gambar 3. 3 Base Model

Dalam bagian modeling hal yang pertama dilakukan adalah mengambil model pre-trained yang telah dilatih, dalam kasus ini model yang diambil EfficientNetV2B3. Parameter input_shape digunakan untuk menetapkan dimensi input gambar pada model, yaitu gambar berukuran 224x224 piksel dengan 3 saluran warna (RGB). Sementara itu, parameter include_top diatur untuk menghilangkan lapisan atas dari model bawaan.

<pre> from tensorflow.keras.models import Sequential from tensorflow.keras.layers import BatchNormalization, Dropout, Flatten, Dense, Activation, MaxPooling2D model=Sequential(name='EfficientNetV2B3') model.add(base_model) model.add(MaxPooling2D()) model.add(Dropout(0.3)) model.add(Flatten()) model.add(Dense(128,activation='relu')) model.add(Dropout(0.3)) model.add(Dense(7,activation='softmax')) model.summary() </pre>		
Model: "EfficientNetV2B3"		
Layer (type)	Output Shape	Param #
efficientnetv2-b3 (Functional)	(None, 7, 7, 1536)	12,930,622
max_pooling2d_4 (MaxPooling2D)	(None, 3, 3, 1536)	0
dropout_9 (Dropout)	(None, 3, 3, 1536)	0
flatten_5 (Flatten)	(None, 13824)	0
dense_10 (Dense)	(None, 128)	1,769,600
dropout_10 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 7)	903
Total params: 14,701,125 (56.08 MB) Trainable params: 14,591,909 (55.66 MB) Non-trainable params: 109,216 (426.62 KB)		

Gambar 3. 4 Fine Tuning

Setelah itu ditambahkan layer MaxPooling2D untuk mengurangi dimensi fitur dan layer Dropout untuk mencegah overfitting. Flatten berfungsi untuk mengubah data multidimensional menjadi satu dimensi, sehingga dapat diproses oleh lapisan fully connected. Dilanjutkan dengan lapisan dense berisi 128 neuron yang memakai fungsi aktivasi ReLU. Dropout ditambahkan untuk meningkatkan generalisasi. Pada akhirnya, lapisan Dense terakhir dengan 7 neuron dan aktivasi softmax menghasilkan probabilitas untuk setiap kelas.

<pre> class EarlyStoppingAtMaxAccuracy(tf.keras.callbacks.Callback): def on_epoch_end(self, epoch, logs = {}): if (logs.get('val_accuracy') > 0.85 and logs.get('accuracy') > 0.85): print("\nAkurasi sudah mendapatkan hasil yang baik!") self.model.stop_training = True callbacks = EarlyStoppingAtMaxAccuracy() </pre>	
<pre> # Tentukan path penyimpanan model checkpoint_path = '/content/drive/MyDrive/Deep Learning UAS/model/efficientnetv2b3.keras' # Buat callback ModelCheckpoint checkpoint_callback = ModelCheckpoint(filepath=checkpoint_path, monitor='val_loss', # Metrik yang akan dipantau save_best_only=True, # Hanya menyimpan model terbaik save_weights_only=False, # Menyimpan seluruh model mode='min', # Mode pemantauan (misalnya, 'min', 'max', atau 'auto') verbose=1) # Menampilkan pesan saat menyimpan </pre>	

Gambar 3. 5 Callback

Untuk meningkatkan efisiensi pelatihan dan memastikan model mencapai performa terbaik, digunakan beberapa callback selama proses pelatihan. Salah satunya adalah callback `EarlyStoppingAtMaxAccuracy`, yang secara otomatis menghentikan pelatihan jika model telah mencapai akurasi dan validasi akurasi di atas 85%, mencegah pelatihan berlebih (overfitting) dan menghemat waktu. Selain itu, callback `ModelCheckpoint` diterapkan untuk menyimpan model terbaik berdasarkan nilai `val_loss` selama pelatihan. Dengan konfigurasi ini, model disimpan secara otomatis hanya ketika mencapai kinerja terbaik, memastikan bahwa model akhir adalah yang paling optimal untuk klasifikasi 7 kelas.

```
history = model.fit(
    train_data,
    batch_size=16,
    validation_data=test_data,
    epochs=30, callbacks=([callbacks],[checkpoint_callback]))
```

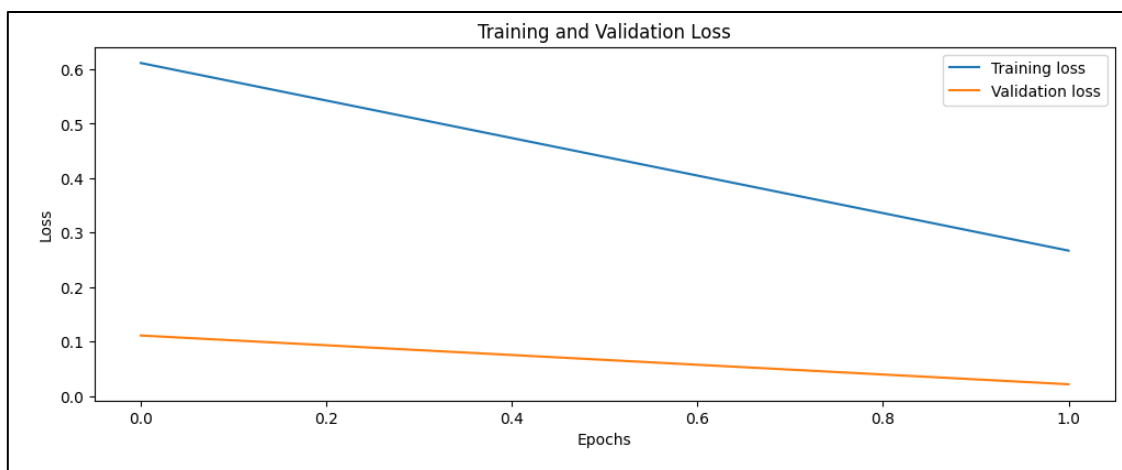
Epoch 1/30
11/11 ————— 0s 8s/step - accuracy: 0.7831 - loss: 0.6676
Epoch 1: val_loss improved from inf to 0.11098, saving model to /content/drive/MyDrive/Deep Learning UAS/model/efficientnetv2b3.keras
11/11 ————— 227s 10s/step - accuracy: 0.7855 - loss: 0.6629 - val_accuracy: 1.0000 - val_loss: 0.1110
Epoch 2/30
11/11 ————— 0s 1s/step - accuracy: 0.9394 - loss: 0.2376
Akurasi sudah mendapatkan hasil yang baik!

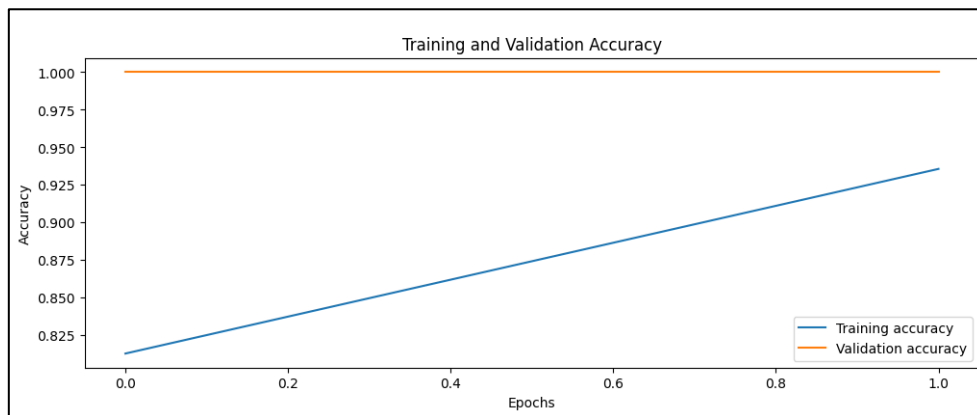
Epoch 2: val_loss improved from 0.11098 to 0.02144, saving model to /content/drive/MyDrive/Deep Learning UAS/model/efficientnetv2b3.keras
11/11 ————— 49s 1s/step - accuracy: 0.9391 - loss: 0.2400 - val_accuracy: 1.0000 - val_loss: 0.0214

Gambar 3. 6 Training Model

Pada bagian ini, melakukan proses training yang mendapatkan hasil 2 Epoch saja karena telah mencapai akurasi pada callback yaitu diatas 0.85. Yang akhirnya didapatkan akurasi 0.9391.

E. Evaluation dan Testing





Gambar 3. 7 Validasi dan Testing Model

Grafik di atas menunjukkan performa model selama pelatihan, yang terdiri dari training loss, validation loss, training accuracy, dan validation accuracy. Pada grafik pertama, terlihat bahwa nilai validation loss lebih rendah dibandingkan training loss, menunjukkan bahwa model mampu memprediksi data validasi dengan baik tanpa indikasi overfitting. Nilai validation loss yang terus menurun menandakan model memiliki generalisasi yang baik. Pada grafik kedua, validation accuracy stabil di angka 100% sejak awal pelatihan, menunjukkan bahwa model dapat memprediksi seluruh data validasi dengan benar. Sementara itu, training accuracy terus meningkat mendekati 100%, mengindikasikan model semakin baik dalam mempelajari pola dari data pelatihan.

F. Analisa Bagaimana Model Dapat Dikatakan Sebagai Deep Learning dan Bukan Shallow Learn?

Model yang digunakan dalam penelitian ini dapat dikategorikan sebagai deep learning karena memiliki sejumlah karakteristik yang membedakannya dari shallow learning. Salah satu cirinya adalah arsitektur yang mendalam, seperti pada EfficientNetV2B3, yang terdiri dari banyak lapisan tersembunyi seperti lapisan konvolusi, batch normalization, dan fungsi aktivasi ReLU. Lapisan-lapisan ini memungkinkan model untuk secara hierarkis mengekstraksi fitur dari data, mulai dari fitur sederhana hingga fitur yang lebih kompleks. Selain itu, proses ekstraksi fitur dilakukan secara otomatis oleh model, tanpa memerlukan rekayasa fitur manual seperti pada shallow learning. Model ini juga mampu menangani data gambar yang besar dan kompleks, seperti dataset gambar gedung berukuran 224x224 piksel dengan tiga saluran warna (RGB), yang sulit ditangani oleh algoritma shallow learning. Dengan karakteristik ini, model yang digunakan jelas masuk dalam kategori deep learning, yang jauh lebih canggih dibandingkan pendekatan shallow learning.