LAPORAN UAS DEEP LEARNING
KLASIFIKASI KOMPONEN ELEKTRONIIK MENGGUNAKAN EFFICICIENTNETV2

DISUSUN OLEH:

1. **Revaldo Relinsyah (G1A021060)**
2. **Parulian Agustinus Hutapea (G1A021064)**
3. **Fajar Adhitia Suwandhi (G1A021086)**

PROGRAM STUDI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS BENGKULU

2024

A. Busines Understanding

Laboratorium teknik Universitas Bengkulu menjadi pusat aktivitas mahasiswa untuk melakukan berbagai kegiatan belajar mengajar maupun penelitian. Dalam kegiatan tersebut, mengidentifikasi komponen dengan baik adalah suatu yang sangat penting. Namun, dalam proses tersebut sering terjadi kesalahan dan memakan banyak waktu yang menghambat efisiensi.

Terutama bagi sebagian besar mahasiswa yang kurang memahami atau tidak terbiasa dengan jenis-jenis komponen elektronik. Banyak mahasiswa mengalami kesulitan dalam membedakan komponen seperti resistor, kapasitor, transistor, atau IC karena bentuknya yang mirip, ukurannya, atau kode yang tercetak pada komponen itu sendiri. Kesulitan ini menyebabkan ketidakakuratan dalam pengambilan atau penggunaan komponen di laboratorium, yang kemudian bisa menghambat prosedur eksperimen, merusak perangkat, atau menyebabkan hasil penelitian menjadi tidak akurat.

Dengan penerapan deep learning dapat mempermudah indentifikasi komponen elektronik di laboratorium menjadi lebih mudah dan dapat menghindari kesalahan yang sering terjadi. Teknologi ini dapat membantu mahasiswa memahami lebih baik komponen-komponen yang digunakan dalam eksperimen, yang pada gilirannya meningkatkan kualitas penelitian dan proses pembelajaran secara keseluruhan.

B. Data Understanding

Dataset merupakan komponen yang penting dalam deep learning karena kualitas dan kuantitas dari dataset secara langsung mempengaruhi kinerja dan hasil yang di bangun model. Dataset adalah sekumpulan data yang dapat digunakan sebagai bahan percobaan riset. Beberapa studi mengumpulkan data mereka sendiri sebagai bahan percobaan.

Dataset komponen elektronik pada project ini terdiri dari 10 komponen (BreadBoard, IntegratedCircuit, Kapasitor, Resistor, connector, dioda, pcb, sensor, swicth, transistor) yang masing-masing dari komponen memiliki 40 gambar yang diambil dari internet. Total dari dataset ini adalah 400 gambar. Karena sumber gambar berasal dari internet menyebabkan gambar memiliki variasi. Untuk

standarisasi, setiap gambar direscale menjadi ukuran 244 x 244 pixel menggunakan parameter target_size (img_height, img_widht).

C. Data Preparation

```python
def prepare_dataset(data_dir, img_height=224, img_width=224, batch_size=32):
    def validate_images(directory):
        valid_formats = {'.jpg', '.jpeg', '.png', '.bmp'}
        invalid_files = []

        for root, _, files in os.walk(directory):
            for filename in files:
                file_path = os.path.join(root, filename)
                file_ext = os.path.splitext(filename)[1].lower()

                if file_ext not in valid_formats:
                    invalid_files.append(file_path)
                else:
                    try:
                        with Image.open(file_path) as img:
                            img.verify()
                    except Exception as e:
                        invalid_files.append(f"{file_path}: {str(e)}")

        if invalid_files:
            print("Warning: Ditemukan file yang tidak valid atau rusak:")
            for file in invalid_files:
                print(f"- {file}")

    print("Memvalidasi dataset...")
```

Gambar 1. Data Preparation

Data preparation dimulai dengan validasi format dari dataset yang valid (.jpg, .jpeg, .png, .bmp) dan memastikan tidak ada data yang rusak dengan menggunakan library PIL. Proses ini dilakukan untuk memastikan data yang akan diproses tidak memiliki kesalahan atau kerusakan yang dapat mengganggu proses pelatihan.

- Data Augmentation

```python
train_datagen = ImageDataGenerator(
    preprocessing_function=tf.keras.applications.efficientnet_v2.preprocess_input,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2
)
```

Gambar 2. Data Augmentation

Selanjutnya bagian augmentasi data, augmentasi data merupakan tahapan penting dalam mempersiapkan dataset untuk pelatihan model deep learning, terutama pada dataset yang memiliki jumlah yang terbatas. Proses ini bertujuan untuk memperbanyak variasi data pelatihan dengan

cara memodifikasi gambar asli dari dataset dengan berbagai transformasi, sehingga model dapat belajar dari banyak variasi.

Parameter	Nilai
rotation_range	20
width_shift_range	0.2
height_shift_range	0.2
shear_range	0.2
zoom_range	0.2
horizontal_flip	True
fill_mode	'nearest'
validation_split	0.2

- Split Data

```
train_datagen = ImageDataGenerator(
    preprocessing_function=tf.keras.applications.efficientnet_v2.preprocess_input,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest',
    validation_split=0.2
)

print("Memuat dataset training...")
train_generator = train_datagen.flow_from_directory(
    data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

print("Memuat dataset validasi...")
validation_generator = train_datagen.flow_from_directory(
    data_dir,
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

Gambar 3. Split Data

Pada bagian split data, data pelatihan dan data validasi dimulai dengan penggunaan validation_split=0.2 pad image generator yang bertujuan untuk membagi dataset menjadi dua subset, yaitu 80% training dan 20% validation. Pembagian ini membuat model dilatih dengan 80% dataset, sementara 20% dataset digunakan untuk menguji kinerja model.

D. Modeling

- Model Pre-trained

```
print("Membuat model EfficientNetV2...")
base_model = EfficientNetV2B0(
    weights='imagenet',
```

Gambar 4. Model Pre-train

Project ini menggunakan model Pre-Trained yaitu model EfficientNetV2B0 sebagai model dasar yang telah dilatih sebelumnya menggunakan dataset ImageNet. Pemilihan model ini sendiri dikarenakan kemampuan model ini dalam melakukan ektraksi fitur dengan sangat baik.

- Model Creation

```
def create_model(num_classes, img_height=224, img_width=224):
    print("Membuat model EfficientNetV2...")
    base_model = EfficientNetV2B0(
        weights='imagenet',
        include_top=False,
        input_shape=(img_height, img_width, 3)
    )

    base_model.trainable = False

    inputs = tf.keras.Input(shape=(img_height, img_width, 3))
    x = base_model(inputs, training=False)
    x = layers.GlobalAveragePooling2D()(x)
    x = layers.BatchNormalization()(x)
    x = layers.Dropout(0.3)(x)
    x = layers.Dense(256, activation='relu')(x)
    x = layers.BatchNormalization()(x)
    x = layers.Dropout(0.4)(x)
    outputs = layers.Dense(num_classes, activation='softmax')(x)

    model = tf.keras.Model(inputs, outputs, name='EfficientNetV2_Electronic')
    return model, base_model
```

Gambar 5. Model Creation

Model ini dibuat menggunakan bobot yang berasal dari model dasar EfficientNetV2B0. Model ini sendiri tidak menambahkan lapisan klasifikasi dari model Pre-trained karena akan ditambahkan lapisan klasifikasi baru sesuai dengan kebutuhan. Selain itu pada bagian input_shape ditentukan tinggi serta lebar gambar serta 3 channel warna.

Karena memanfaatkan fitur dari model Pre-trained maka base model dibekukan sehingga bobotnya tidak berubah serta dapat mempercepat proses pelatihan dan mencegah model pre-trained kehilangan semua hasil dari pelatihan yang telah dilakukan sebelumnya. Terakhir ditambahkan lapisan-lapisan tambahan seperti GlobalAveragePooling2D,

BatchNormalization, Dropout, dan Dense untuk klasifikasi komponen elektonik dengan menggunakan dataset yang telah disiapkan.

Gambar 6. proses membuat model

- Fine Tunning

```python
def unfreeze_model(model, base_model):
    print("Unfreezing layers untuk fine-tuning...")
    base_model.trainable = True

    for layer in base_model.layers:
        if isinstance(layer, layers.BatchNormalization):
            layer.trainable = False

    return model
```

Gambar 7. Fine Tunning

Kemudian dilakukan proses fine-tuning dimana lapisan-lapisan pada base model statusnya akan diubah menjadi unfreeze sehingga base model dapat dilatih sehingga akan diperbaharui pada saat proses pelatihan. Namun untuk bagian BatchNormalization akan tetap dibekukan untuk menjaga stabilitas model karen telah dilatih dengan baik pada dataset ImageNet. Pembaharuan pada lapisan BatchNormalization sendiri dapat mengganggu stabilitas model terutama pada saat proses fine-tuning

Arsitektur Model:
Model: "EfficientNetV2_Electronic"

Layer (type)	Output Shape	Param #
input_layer_1 (InputLayer)	(None, 224, 224, 3)	0
efficientnetv2-b0 (Functional)	(None, 7, 7, 1280)	5,919,312
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
batch_normalization (BatchNormalization)	(None, 1280)	5,120
dropout (Dropout)	(None, 1280)	0
dense (Dense)	(None, 256)	327,936
batch_normalization_1 (BatchNormalization)	(None, 256)	1,024
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 10)	2,570

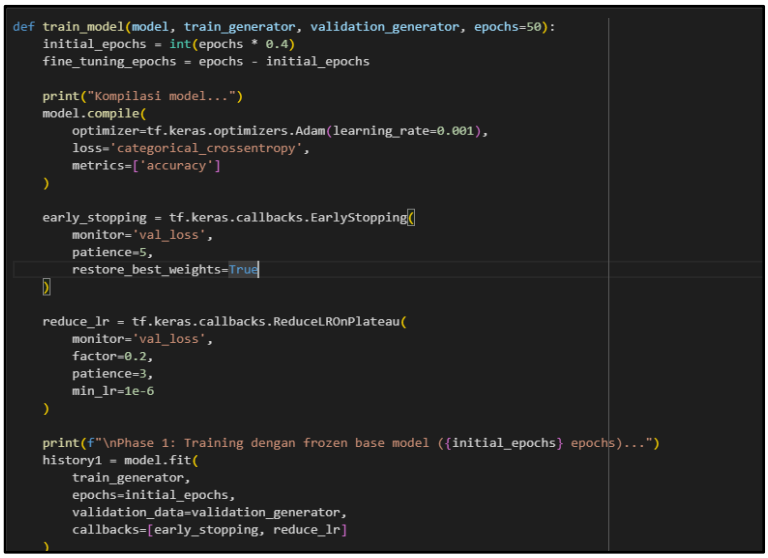
Total params: 6,255,962 (23.86 MB)
Trainable params: 333,578 (1.27 MB)
Non-trainable params: 5,922,384 (22.59 MB)

Gambar 8. Hasil Fine-Tunning

Hasil dari proses fine-tunning menunjukan sebagian parameter tidak dilatih ulang selama proses fine-tunning yaitu sekitar 5,9 juta parameter dari 6,2 juta parameter. Dalam proses ini hanya lapisan Dense dan

BatchNormalization yang dilatih menggunakan dataset baru yang disiapkan dengan jumlah sekitar 333 parameter. Model ini sendiri dimulai dengan mengolah inputan gambar berukuran 224x224 piksel. Kemudian melalui beberapa lapisan yaitu GlobalAveragePooling2D dan BatchNormalization untuk meproses fitur yang telah ada. Setelah itu lapisan dense dengan 256 unit digunakan diikuti dengan dropout untuk mencegah terjadinya overfitting. Dan terakhir lapisan output menggunakan aktivasi softmax untuk klasifikasi 10 kelas. Sehingga dari total 6,255,962 parameter model ini, hanya 333,578 parameter yang dilatih selama proses ini sementara lapisan lainnya dibiarkan tetap tidak dilatih.

- Train Model



```
def train_model(model, train_generator, validation_generator, epochs=50):
    initial_epochs = int(epochs * 0.4)
    fine_tuning_epochs = epochs - initial_epochs

    print("Kompilasi model...")
    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )

    early_stopping = tf.keras.callbacks.EarlyStopping(
        monitor='val_loss',
        patience=5,
        restore_best_weights=True
    )

    reduce_lr = tf.keras.callbacks.ReduceLROnPlateau(
        monitor='val_loss',
        factor=0.2,
        patience=3,
        min_lr=1e-6
    )

    print(f"\nPhase 1: Training dengan frozen base model ({initial_epochs} epochs)...")
    history1 = model.fit(
        train_generator,
        epochs=initial_epochs,
        validation_data=validation_generator,
        callbacks=[early_stopping, reduce_lr]
    )
```

Gambar 9. Train Model Fase 1

Proses pelaatihan model sendiri dilakukan sebanyak 2 fase. Fase pertama yaitu pada saat base mode dibekukan. Fase ini melibatkan sekitar 40% dari total epoch yang kemudian model dikompilasi menggunakan Adam optimizer dengan learning rate 0.001 serta menggunakan categorical_crossentropy sebagai fungsi loss untuk klasifikasi multi-kelas. Model dilatih menggunakan data yang ada pada train_generator selama initial_epochs dengan data validasi dari validation_generator. Selain itu pada model juga diterapkan callbacks seperti EarlyStopping dan ReduceLROnPlateau yang akan menghentikan proses jika val_loss tidak

mengalami perbaikan dalam 5 epoch berturut-turut serta mengurangi learning rate jika val_loss tidak membaik selama 3 epoch berturut-turut.

```
print(f"\nPhase 2: Fine-tuning model ({fine_tuning_epochs} epochs)...")
model = unfreeze_model(model, base_model)

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

history2 = model.fit(
    train_generator,
    epochs=fine_tuning_epochs,
    validation_data=validation_generator,
    callbacks=[early_stopping, reduce_lr]
)
```

Gambar 10. Train model fase 2

Pada fase kedua, beberapa layer pada model akan di unfrezee untuk proses fine tunning. Fase ini memungkinkan model untuk mengubah bobot pada lapisan yang dibekukan sebelumnya. Kemudian model kembali dilakukan kompilasi dengan learning rate yang lebih rendah sehingga proses fine-tunning dilakukan secara hati-hati. Proses pelatihan ini dilakukan menggunakan data train_generator dan validasi validation_generator dengan menerapkan kembali callbacks.

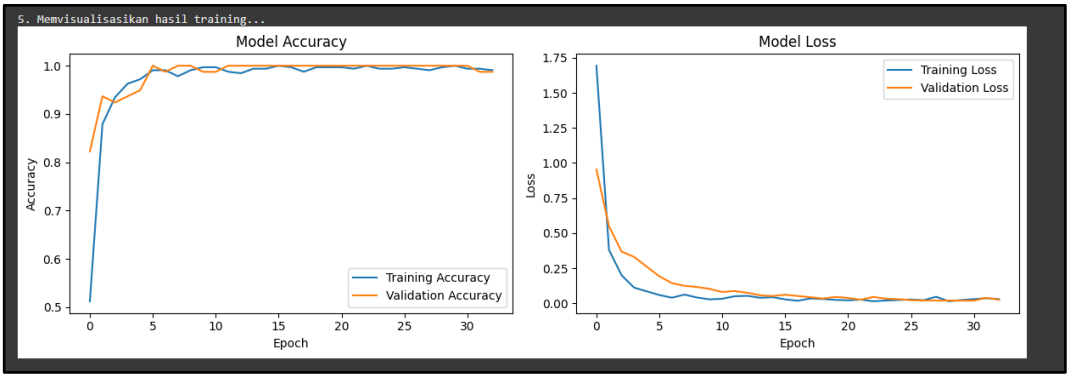
E. Model Evaluation

```
def plot_training_results(history):
    plt.figure(figsize=(12, 4))

    plt.subplot(1, 2, 1)
    plt.plot(history['accuracy'], label='Training Accuracy')
    plt.plot(history['val_accuracy'], label='Validation Accuracy')
    plt.title('Model Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(history['loss'], label='Training Loss')
    plt.plot(history['val_loss'], label='Validation Loss')
    plt.title('Model Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()

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

Gambar 11. Kode Visualisasi

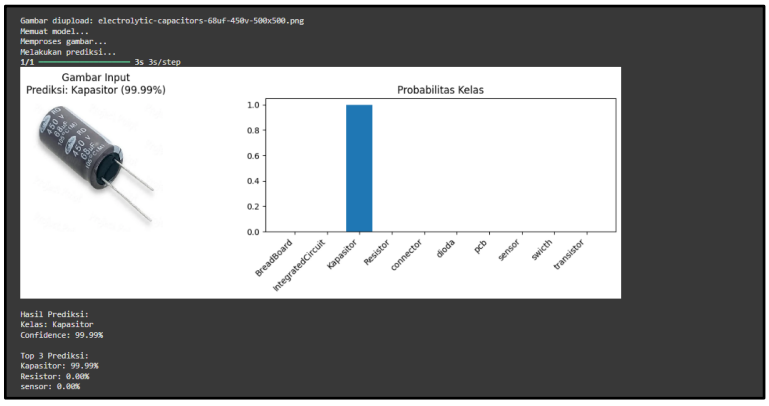
Setelah proses pelatihan model selesai dilakukan, selanjtunya dilakukan evaluasi pelatihan model. Fungsi diatas digunakan untuk memvisualisasikan hasil

dari pelatihan model khususnya pada bagian perkembangan akurasi serta loss selama pelatihan dan validasi pada setiap epoch.

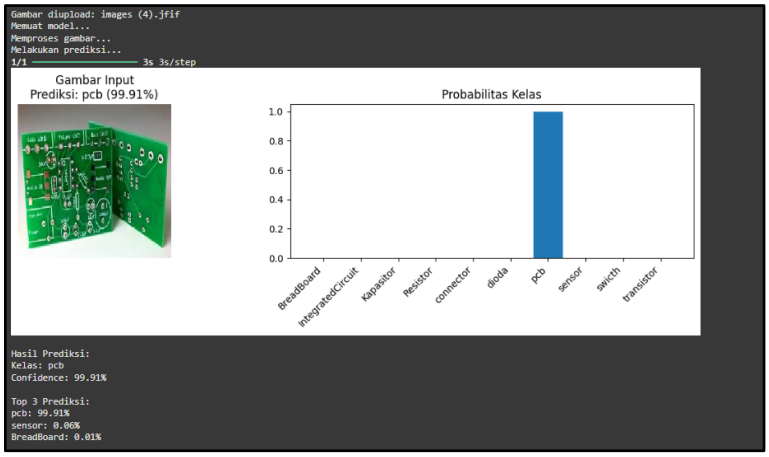


Gambar 12. Visualisai Model

Hasil dari pelatihaan model menunjukan hasil yang sangat baik dalam mengenali pola pada data. Akurasi model yang didapatkan mendekati 100% baik itu pada data pelatihan maupun data validasi. Ini menunjukan jika model yang dilatih belajar dengan baik tanpa adanya overfitting. Selain itu dengan akurasi data validasi yang stabil menunjukan proses fine-tunning yang dilakukan berhasil dan berdampak positif terhadap kinerja model. Secara keseluruhan model berhasil mempelajari fitur penting yang ada dan menghindari overfitting melalui teknik early stopping dan learning rate reduction sehingga model menunjukan hasil yang sangat baik.



Gambar 13. Hasil Prediksi 1



Gambar 14. Hasil Prediksi 2

F. Saving Model

```
print("\n6. Menyimpan model...")
model.save(MODEL_SAVE_PATH)
print(f"Model berhasil disimpan di: {MODEL_SAVE_PATH}")
```

Gambar 15. Saving Model

Terakhir model yang telah dilatih disimpan kedalam bentuk file yang memungkinkan untuk menyimpan semua struktur model dan bobot pelatihan tanpa harus melakukan pelatihan ulang. Setelah model berhasil di simpan, variabel MODEL_SAVE_PATH akan menyimpan lokasi file tempat model disimpan. Teknik ini memungkinkan enggunaan model untuk prediksi lebih lanjut sehingga menghemat waktu dan sumber daya.

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

Model pada project ini dikatakan sebagai deep learning karena beberapa hal, pertama model ini terdiri dari banyak lapisan atau layer dalam proses ektraksi fitur, model dasar EfficientNetV2B0 yang digunakan menggunakan jaringan saraf dengan banyak lapisan. Dengan banyaknya lapisan ini memungkinkan model untuk belajar dengan lebih komplek dibandingkan dengan shallow learn yang biasanya hanya memiliki satu atau dua lapisan saja.

Kedua, model ini menggunakan metode transfer learning dimana model ini memanfaatkan model yang telah dilatih sebelumnya yaitu model EfficientNetV2B0

yang dilatih menggunakan dataset imageNet. Ini menunjukan salah satu aspek pada deep learning yaitu teknik transfer learning Selain itu untuk meningkatkan variasi data pelatihan, model ini menggunakan proses augmentasi data dimana teknik ini umum digunakan dalam deep learning. Berbeda dengan shallow learning yang biasanya tidak bisa menangani data dalam jumlah besar atau data yang memiliki dimensi tinggi dengane fisien.

Dan terakhir model ini melalui proses pelatihan selama 50 epoch, dimana proses ini cukup panjang serta umum dilakukan dalam proses deep learning dalam melatih model dengan banyak parameter dan lapisan. Proses ini menunjukan penggunaan waktu yang lebih lama karena banyaknya parameter dan lapisan yang harus diperbaharui. Berbeda dengan shallow learning yang biasanya dilatih dengan lebih cepat dengan dataset yang lebih kecil dan masalah yang lebih sederhana.