BAB IV. ANALISIS DAN PERANCANGAN SISTEM

4.1 Deskripsi Sistem

Pelatihan yang dibangun dalam penelitian ini adalah penelitian tentang Implementasi *Image Classification* pada Jenis-Jenis Batik Menggunakan *Convolutional Neural Network* dengan Model *EfficientNet*. Pelatihan ini digunakan untuk mengetahui akurasi terbaik dalam pelatihan model pada dataset batik karena banyaknya pola batik di Indonesia mengakibatkan sulitnya untuk mengindentifikasi motif batik dengan akurat.

Dalam pelatihan ini dapat melihat akurasi terbaik pada dataset batik dengan menggunakan model arsitektur *EfficientNet* dengan melalui beberapa tahap seperti memasukkan dataset batik, melakukan perancangan *preprocessing*, melakukan pengumpulan data *training* dengan memuat dan memproses dataset untuk data *training*, melakukan pengumpulan data *validation* dengan memuat dan memproses dataset untuk data *validation*, proses *augmentasi*, menambahkan model *EfficientNet*, melakukan penambahan layer, menambahkan optimizer, dan melakukan pelatihan. Hasil dari pelatihan ini akan mendapatkan nilai akurasi yang baik.

4.2 Analisa Kebutuhan Sistem

Dalam membangun sistem pendukung keputusan ini, terdapat beberapa analisa pada sistem. Analisa yang dilakukan yakni analisa kebutuhan perangkat keras *(Hardware)* dan perangkat lunak *(Software)*. Berikut adalah spesifikasi dari analisa sistem :

A. Kebutuhan Perangkat Keras *(Hardware)*

Perangkat keras dibutuhkan untuk mendukung berjalannya program sesuai dengan ketentuan yang dibutuhkan. Brikut adalah spesifikasi perangkat yang digunakan dalam pembuatan sistem :

Tabel 4. 1 Kebutuhan Perangkat Keras *(Hardware)*

No	Nama Perangkat Keras
1	Intel Core i7-8550U
2	RAM 8GB

3	SSD 256GB M.2

B. Kebutuhan Perangkat Lunak *(Software)*

Perangkat lunak dibutuhkan untuk mmbuat program sesuai dengan ketentuan yang dibutuhkan. Berikut adalah spesifikasi perangkat lunak yang digunakan dalam membuat sistem :

Tabel 4. 2 Kebutuhan Perangkat Lunak (Software)

No	Nama Perangkat Lunak
1	Sistem Operasi Windows 10
2	Google Chrome

4.2.1 Kebutuhan Input

Dalam pelatihan *image classification* pada jenis-jenis batik ini membutuhkan data yang dibutuhkan untuk inputan citra pada pelatihan *image classification*. Inputan citra berupa *dataset* yang dapat diakses di https://www.kaggle.com/datasets/dionisiusdh/indonesian-batik-motifs. *Dataset* ini terdiri dari 20 jenis batik dengan jumlah data sebanyak 1000 citra batik. Selain itu, terdapat penambahan citra batik supaya menambahkan tingkat akurasi pada pelatihan ini.

4.2.2 Kebutuhan Proses

Setelah didapatkan dataset, kemudian dataset tersebut akan diproses supaya mendapatkan informasi yang dibutuhkan. Ada beberapa proses yang harus dilakukan, diantaranya:

4.2.2.1 Proses Augmentasi

Augmentasi data adalah strategi yang memungkinkan praktisi untuk secara signifikan meningkatkan keragaman data yang tersedia untuk model pelatihan, tanpa benar-benar mengumpulkan data baru (Sanjaya & Ayub, 2020). Penggunaan proses augmentasi diharapkan dapat menaikkan performa model karena mesin akan berhasil mengenali lebih banyak objek dari bentuk serta pola yang beragam jenisnya. Berikut merupakan hasil augmentasi dataset batik untuk pelatihan ini.

Gambar 4. 1 Hasil Proses Augmentasi

Berikut merupakan hasil augmentasi data dari citra dataset batik yang digunakan untuk pelatihan. Citra tersebut mengalami proses rescale, pembalikan citra secara horizontal dan mengalami rotasi acak.

```
train_datagen = ImageDataGenerator(
                    validation_split=0.3,
                    rescale=1./255,
                    horizontal_flip=True,
                    rotation_range=30)
```

Pada proses augmentasi ini, citra akan melakukan proses augmentasi guna memperkaya data baru untuk mendapatkan performa model yang baik. Citra akan mengalami beberapa proses augmentasi, seperti:

A. Validation_split

   Pada statement ini berfungsi untuk membagi data citra menjadi data training dan data validation. Proporsi dari argumen 0,3 (30%) dari jumlah dataset akan digunakan untuk data validasi. Sementara sisanya sebesar 0,7 (70%) akan digunakan untuk data training

B. Rescale

   Pada statement ini berfungsi untuk menykalakan nilai piksel dalam citra menjadi rentang 0 hingga 1. Hal ini dilakukan dengan membagi setiap nilai piksel dengan 255. Prosedur ini dilakukan sebagai langkah normalisasi pada data citra

C. Horizontal_flip

   Pada statement ini berfungsi untuk menerapkan atau mengaktifkan pembalikan horizontal secara acak pada citra dalam dataset

D. Rotation_range

Pada statement ini berfungsi untuk mengatur rentang rotasi acak pada citra dalam dataset secara 30 derajat. Selama dalam pelatihan, citra pada dataset akan secara acak diputar dalam rentang -30 derajat hingga +30 derajat

4.2.2.2 Proses Preprocessing

Proses preprocessing digunakan untuk meningkatkan kualitas gambar agar lebih baik daripada sebelumnya, dengan tujuan mempermudah pengenalan ciri-ciri pada gambar tersebut. Pada pelatihan kali ini citra akan melalui proses preprocessing dengan menggunakan filter Gaussian Noise

```
    augmentasi = tf.keras.Sequential([

    tf.keras.layers.experimental.preprocessing.RandomFlip('hori
    zontal'),

    tf.keras.layers.experimental.preprocessing.RandomRotation(0
    .2),

    tf.keras.layers.experimental.preprocessing.RandomZoom(0.2),
        tf.keras.layers.GaussianNoise(0.2)
])
```

Pada proses preprocessing ini, citra akan melakukan proses prepocessing dari beberapa tahap preprocessing, seperti:

A. RandomFlip

Pada statement ini berfungsi untuk mengatur augmentasi berupa pembalikan acak secara horizontal pada citra. Argumen dari ‘horizontal’ menunjukkan citra dalam dataset akan mengalami pembalikan acak secara horizontal

B. RandomRotation

Pada statement ini berfungsi untuk mengatur augmentasi berupa rotasi acak pada citra sebesar 0,2. Argumen dari 0,2 menunjukkan citra dalam dataset pelatikan akan secara acak di putar atau di rotasi sebesar 20%

C. RandomZoom

Pada statement ini berfungsi untuk mengatur augmentasi berupa pembesaran acak pada citra sebesar 0,2. Argumen dari 0,2 menunjukkan citra dalam dataset pelatihan akan diperbesar sebesar 20%

D. GaussianNoise

Pada statement ini berfunsi untuk mengaktifkan noise Gaussian secara acak pada citra dalam dataset. Argumen yang diberikan 0,2 merupakan standar deviasi dari distribusi Gaussian yang akan digunakan untuk menghasilkan noise

4.2.2.3 Proses Penambahan Model Pelatihan

Pada proses ini, pelatihan akan menambahkan model pelatihan yang berfokus pada EfficientNet

```
base_model = EfficientNetB0(include_top=False,
weights='imagenet', input_shape=(224,224,3))
```

Pada proses ini pelatihan akan mengaktifkan model EfficientNetB0. Model akan menerima citra dengan ukuran 224 x 224 piksel dan 3 saluran warna (RGB)

4.2.2.4 Proses Penambahan Layer

Pada proses ini, pelatihan akan menambahkan beberapa layer guna meningkatkan hasil pelatihan dan mendapatkan hasil yang terbaik.

```
    model = tf.keras.Sequential([
        #augmentation,
        base_model,
        tf.keras.layers.GlobalAveragePooling2D(),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dropout(0.5),
        tf.keras.layers.Dense(512, activation='relu'),
        tf.keras.layers.Dropout(0.5),

    tf.keras.layers.Dense(len(train_generator.class_indices),
    activation='softmax')
])
```

Berikut merupakan beberapa penambahan layer yang akan digunakan untuk pelatihan ini:

A. Base_model

Pada argumen ini pelatihan akan memanggil objek base_model yang berisikan pengaktifan model EfficientNet.

B. GlobalAveragePooling2D

Pada argumen ini pelatihan akan menambahkan layers GlobalAveragePooling2D yang diharapkan dapat mengurangi jumlah parameter sehingga proses komputasi dapat berjalan lebih cepat.

C. Dropout

Pada argumen ini pelatihan akan menambahkan layer dropout dengan tingkat dropout sebesar 0,5. Dengan mengaktifkan layer dropout ini akan mencegah *overfitting* ketika pelatihan berlangsung. Penambahan layer Dropout ini dilakukan secara 2 kali dengan tingkat dropout yang sama.

D. Dense

Pada argumen ini pelatihan akan menambahkan layer dense yang akan dilakukan secara 3 kali. Untuk penambahan layer dense yang pertama akan menggunakan 128 unit neuron dan menggunakan fungsi aktivasi ReLu. Untuk penambahan layer dense yang kedua akan menggunakan 512 unit neuron dan menggunakan fungsi aktivasi ReLu.

Untuk penambahan layer dense yang terakhir menggunakan jumlah unit neuron yang sesuai dengan jumlah kelas dalam dataset pelatihan dengan statement `len(train_generator.class_indices)`. Sealnjutnya pada layer dense ini juga menggunakan fungsi aktivasi softmax yang digunakan untuk menghasilkan probabilitas prediksi untuk setiap kelas

4.2.2.5 Proses Penambahan Optimizer

Pada proses ini, pelatihan akan menambahkan optimizer. Optimizer ini digunakan mengoptimalkan parameter-parameter model neural network yang akan digunakan selama pelatihan.

```
model.compile(optimizer=RMSprop(learning_rate=0.0001),
loss='categorical_crossentropy',
metrics=['categorical_accuracy'])
```

Pada proses ini pelatihan akan membuat objek model.compile untuk menambahkan optimizer RMSprop yang akan digunakan selama proses pelatihan dengan menggunakan *learning rate* (tingkat pembelajaran) sebesar 0.0001

4.2.2.6 Proses Pelatihan

Pada proses ini mesin akan melakukan pelatihan dengan proses-proses yang telah dirancang

```
history = model.fit(train_generator,
                    validation_data=validation_generator,
                    batch_size=64,
                    epochs=100,
                    verbose=1)
```

Pada proses pelatihan ini akan menggunakan data train yang diargumenkan dengan train_generator dan akan divalidasi menggunakan data validasi yang diargumenkan dengan validation_data. Selama pelatihan, ukuran batch yang digunakan sebesar 64 dan model akan dilatih selama 100 epoch (siklus pelatihan penuh) dengan verbosity level 1.

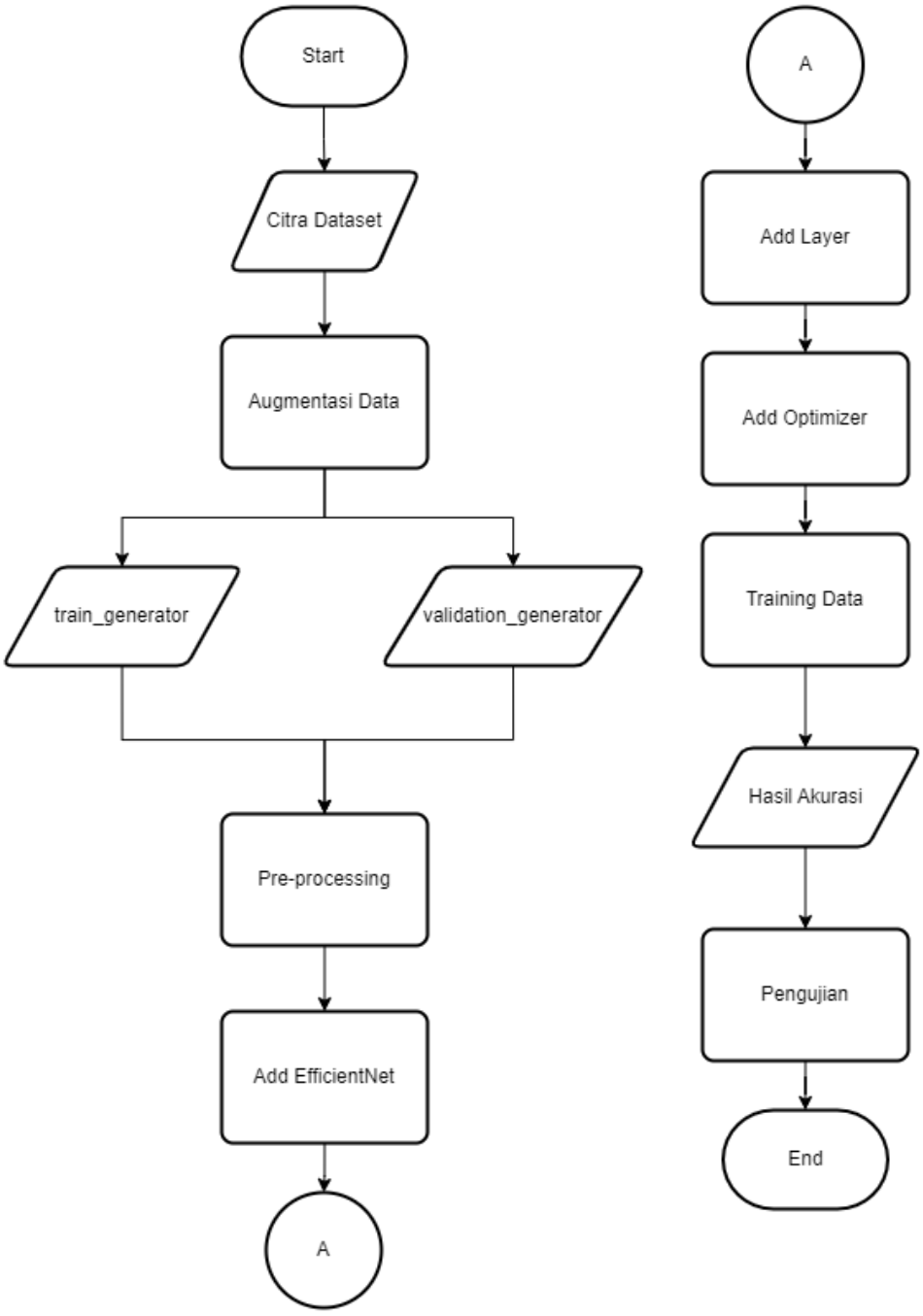
4.2.3 Kebutuhan Output

Output dari hasil input dataset dan beberapa proses seerta pelatihan dataset yaitu menghasilkan informasi berupa hasil akurasi yang diharapkan baik dan sesuai.

4.3 Perancangan Sistem

Pada tahap ini merupakan proses perancangan sistem untuk menghasilkan akurasi yang terbaik dari pelatihan yang telah dirancang

4.3.1 *Flowchart* Sistem



Gambar 4. 2 *Flowchart* Sistem

Berikut merupakan alur jalan sistem atau *flowchart* sistem pada penelitian tentang Implementasi *Image Classification* pada Jenis-Jenis Batik Menggunakan *Convolutional Neural Network* dengan Model *EfficientNet.* Pada alur pertama mesin akan melakukan input data berupa dataset citra batik yang berisikan 20 class dengan jumlah citra sebanyak 2000 gambar.

Setelah mesin melakukan input dataset yang berisikan citra batik, mesin akan melakukan proses augmentasi dataset yang berisikan citra batik. Proses augmentasi ini memiliki fungsi untuk meningkatkan keanekaragaman dan jumlah data pelatihan tersedia. Dari proses augmentasi ini juga dapat meningkatkan performa model dan membantu untuk mencegah *overfitting*.

Setelah mesin melakukan proses augmentasi data, mesin akan membagi data citra menjadi 2, yaitu data train yang diargumenkan dengan train_genetaror dan data validasi yang diargumenkan dengan validation_generator. Proporsi untuk pembagian antara data train dengan data validasi sebanyak 70% untuk data train dan 30% untuk data validasi.

Setelah mesin melakukan proses pembagian data citra, mesin akan melakukan proses pre-processing. Pada tahap proses pre-processing kali ini, mesin akan menambahkan filter GaussianNoise senilai 0,2 yang akan digunakan untuk menghasilkan noise. Selain itu menambahkan filter Gaussian Noise, juga menambahkan Random Flip secara horizontal, Random Rotation, dan Random Zoom.

Setelah mesin melakukan proses pre-processing, mesin akan menambahkan model *EfficientNet* didalam objek base_model. Pada EfficientNet kali ini akan menerima inputan citra dengan ukuran 224 x 224 piksel dan 3 saluran warna (RGB) yang diatur dalam input_shape.

Setelah mesin melakukan proses penambahan *EfficientNet*, mesin akan memperkenalkan lapisan tambahan yang bermanfaat untuk meningkatkan hasil pelatihan dan mencapai hasil yang optimal. Rancangan lapisan tambahan yang akan digunakan untuk pelatihan adalah sebagai berikut:

1. Lapisan pertama untuk *EfficientNet* yang diargumenkan dengan objek base_model.
2. Lapisan kedua untuk GlobalAveragePooling2D dengan harapan dapat mengurangi jumlah parameter sehingga komputasi dapat dilakukan dengan lebih cepat.
3. Lapisan ketiga untuk Dropout sebesar 0.5 untuk mencegah adanya *overfitting* ketika pelatihan berlangsung.

4. Lapisan keempat untuk Dense dengan menggunakan 128 unit neuron dan menggunakan fungsi aktivasi ReLu.
5. Lapisan kelima untuk Dropout sebesar 0.5 untuk mencegah adanya *overfitting* ketika pelatihan berlangsung.
6. Lapisan keenam untuk Dense dengan menggunakan 512 unit neuron dan menggunakan fungsi aktivasi ReLu.
7. Lapisan ketujuh untuk Dropout sebesar 0.5 untuk mencegah adanya *overfitting* ketika pelatihan berlangsung.
8. Lapisan kedelapan untuk Dense dengan menggunakan jumlah unit neuron yang disesuaikan dengan jumlah kelas yang dalam dataset pelatihan. Pada lapisan ini juga mengaktifkan fungsi softmax.

Setelah melakukan penambahan layers untuk pelatihan, tahap selanjutnya yaitu menambahkan *optimizer* yang digunakan mengoptimalkan parameter-parameter model neural network yang akan digunakan selama pelatihan.

Setelah mesin melakukan penambahan *optimizer*, mesin akan melakukan proses training data. Proses training data ini akan melibatkan seluruh proses yang telah dirancang. Pada proses training data ini akan dilakukan sebanyak 100 epoch dan nantinya akan menghasilkan sebuah nilai akurasi dari perancangan yang telah dirancang. Setelah mesin melakukan proses training dan menghasilkan hasil akurasi, akan melakukan pengujian dengan menggunakan *confussion matrix*.