BAB 3
METODOLOGI PENELITIAN

3.1 Tahapan - Tahapan Penelitian

Penelitian ini dilakukan untuk mendeteksi penyakit Alternaria, Cucumber Mosaic Virus (CMV), Downy Mildew, dan Powdery Mildew pada tanaman melon melalui analisis citra daun yang mengalami infeksi menggunakan algoritma CNN. Metodologi penelitian yang akan diterapkan mencakup langkah-langkah sebagai berikut :

1. Studi Literatur

   Tahap awal dalam proses ini adalah mengumpulkan dan mengevaluasi informasi serta teori yang relevan sebagai dasar penelitian. Studi literatur memegang peran penting dalam memberikan pemahaman terhadap studi-studi sebelumnya yang berada dalam bidang yang sama. Teori-teori yang ditemukan melalui kajian tersebut berfungsi sebagai pijakan dalam memperdalam analisis penelitian yang sedang dilakukan, dengan mengakses referensi berupa artikel, jurnal, dan buku dari sumber online yang kredibel.

2. Pengumpulan Data

   Dalam tahap ini, data berupa gambar tanaman melon dikumpulkan dari sumber yang kredibel, yaitu *Roboflow*, yang menyediakan beragam gambar tanaman melon dalam berbagai kondisi. Total sebanyak 1218 gambar berhasil dihimpun, mencakup tanaman melon dalam berbagai kondisi yang menjadi fokus penelitian.

3. *Pre-processing Data*

   Pre-processing data adalah tahap persiapan dan pembersihan data sebelum digunakan untuk melatih model. Hal ini dilakukan agar model dapat menggunakan data secara efektif.

4. Pelatihan Model

   Tahap ini bertujuan sebagai pelatihan model yang kemudian menghasilkan model CNN ResNet-50, CNN EfficientNet-B0, dan CNN MobileNet-V3-Small . Ketiga model ini memiliki tujuan yang sama yaitu mendeteksi

penyakit *Powdery Mildew, Cucumber Mosaic Virus (CMV), Alternaria*, dan *Downy Mildew* pada tanaman Melon.

5. Implementasi

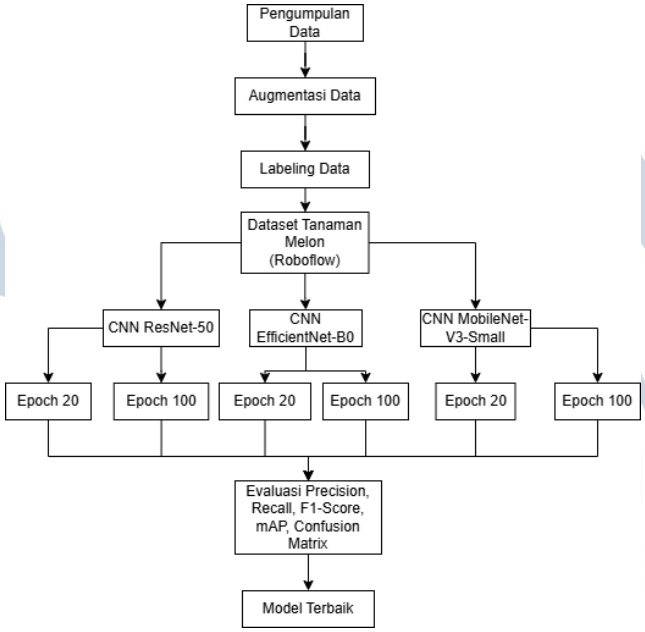
   Pada tahap ini ketiga model akan dibuat dengan menggunakan bahasa pemorgraman Python yang kemudian model tersebut akan mengambil sampel data dari data set yang telah didapatkan dari *Roboflow* sebelumnya. Kemudian setelah itu masing-masing model akan diuji dan dievaluasi.

6. Pengujian dan Evaluasi

   Ketiga model akan diuji dengan menggunakan gambar yang sebelumnya belum pernah dilihat oleh model. Kemudian dari hasil tersebut, dilakukan evaluasi terhadap model dengan membandingkan mAP (*mean Average Precision*) masing-masing model.

3.2 Alur Kerja Penelitian

Alur kerja untuk mendeteksi penyakit *Powdery Mildew, Cucumber Mosaic Virus (CMV), Alternaria*, dan *Downy Mildew* pada tanaman melon dapat dilihat pada Gambar 3.1.



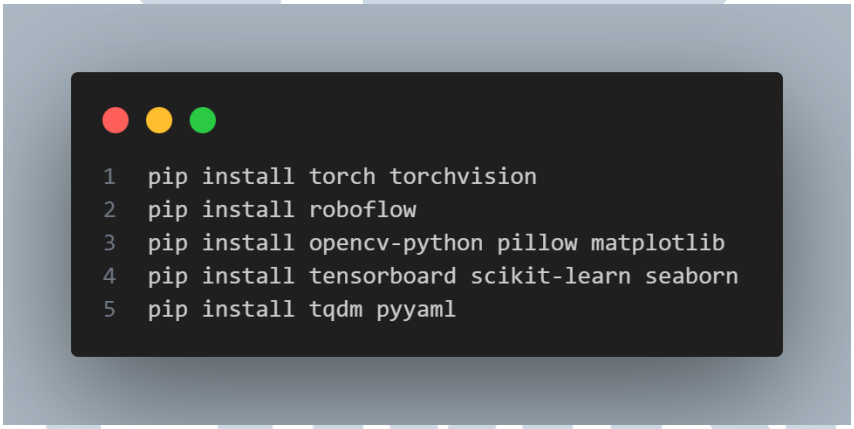
Gambar 3.1. Alur Penelitian

3.3 Perancangan dan Pengembangan Sistem

Perancangan dan pengembangan sistem untuk penelitian deteksi penyakit pada melon menggunakan algoritma CNN (Convolutional Neural Network) dengan arsitektur ResNet-50, EfficientNet-B0, dan MobileNet_V3_Small terdiri dari beberapa bagian, yaitu instalasi *library* dan pengunduhan dataset, persiapan data dan augmentasi, perancangan arsitektur model, proses pelatihan model, evaluasi model, dan perancangan flowchart sistem.

3.3.1 Instalasi *Library* dan Pengunduhan Dataset

Tahap ini dimulai dengan instalasi *library* yang diperlukan untuk implementasi CNN. *Library* utama yang digunakan adalah PyTorch untuk deep learning, Roboflow untuk pengelolaan dataset, dan *library* pendukung lainnya. Instalasi dapat dilakukan dengan perintah pada Gambar 3.2



Gambar 3.2. Install Library

Selanjutnya, dataset akan diunduh dari Roboflow menggunakan API key yang telah disiapkan. Dataset akan diunduh dalam format YOLO. Dataset kemudian akan dikonversi dan disiapkan untuk pelatihan model CNN.Dataset akan dibagi menjadi 80% untuk data pelatihan, 10% untuk validasi, dan 10% untuk pengujian.

3.3.2 Perancangan Arsitektur Model

Untuk mendapatkan perbandingan performa, sistem ini mengimplementasikan tiga arsitektur CNN sebagai berikut:

1. CNN ResNet50

   ResNet-50 merupakan model deep residual network dengan 50 layer, menggunakan pre-trained weights dari ImageNet untuk *transfer learning*. Pada ResNet-50, *fully connected layer* terakhir diganti dengan kombinasi *dropout layer* dengan rate 0.5 untuk regularisasi dan linear layer yang disesuaikan dengan jumlah kelas penyakit melon.

2. EfficientNet-B0

   EfficientNetB0 dirancang dengan prinsip *compound scaling* untuk mencapai efisiensi optimal antara akurasi dan ukuran model. *Classifier* pada EfficientNet-B0 dimodifikasi dengan menambahkan *dropout layer* (rate=0.5) dan *linear layer* untuk klasifikasi akhir.

3. MobileNet_V3_Small

MobileNet_V3_Small yang merupakan arsitektur ringan yang dioptimalkan untuk *deployment* pada perangkat dengan *resource* terbatas. Proses yang terjadi adalah Linear(in_features $\rightarrow$ 1024) $\rightarrow$ *Hard-Swish activation* $\rightarrow$ Dropout(0.5) $\rightarrow$ Linear$(1024 \rightarrow$ *num_classes*$)$

Semua model yang disebutkan diatas, menggunakan parameter yang telah disesuaikan dengan arsitektur dari masing-masing model. Berikut parameter yang digunakan:

Tabel 3.1. CNN Model Hyperparameters

Model	**LR**	**WD**	**DR**	**BS**	**Image(pixel)**
ResNet-50	0.0005	1e-4	0.5	64	224×224
EfficientNet-B0	0.0005	1e-4	0.2	64	224×224
MobileNetV3-Small	0.0005	1e-3	0.5	64	224×224

3.3.3 Proses Pelatihan Model

Pada parameter-parameter yang digunakan dapat diubah lagi sesuai kebutuhan. Untuk penjelasan parameter yang digunakan dapat dilihat pada Tabel 3.2

Tabel 3.2. Penjelasan Parameter Training CNN

Parameter	**Deskripsi**
IMG_SIZE	Ukuran gambar target untuk pelatihan. Semua gambar diubah ukurannya menjadi 224×224 pixel sebelum dimasukkan ke dalam model CNN. Mempengaruhi detail yang dapat ditangkap dan kebutuhan memori.
BATCH_SIZE	Jumlah sampel yang diproses bersamaan dalam satu iterasi. Nilai 64 memberikan keseimbangan antara kecepatan training dan penggunaan memori.
LEARNING_RATE	Laju pembelajaran yang mengontrol seberapa besar update bobot model.
NUM_EPOCHS	Jumlah total *epoch* pelatihan. Setiap *epoch* mewakili satu kali proses lengkap melalui seluruh dataset. Maksimum 100 *epoch* dengan *early stopping*.
DROPOUT_RATE	Probabilitas neuron di-nonaktifkan selama *training* untuk mencegah *overfitting*. Nilai 0.5 berarti 50% neuron akan di-dropout.
WEIGHT_DECAY	Parameter regularisasi L2 untuk mencegah *overfitting* dengan menambahkan penalty pada bobot yang besar.
model_type	Menentukan arsitektur CNN yang digunakan: resnet50, efficientnet_b0, atau mobilenet_v3_small. Masing-masing memiliki trade-off antara akurasi dan efisiensi.

Proses pelatihan dilakukan dengan 2 macam variasi. Masing-masing memiliki variasi pada jumlah *epoch dan patience*

Tabel 3.3. Parameter Training untuk Berbagai Model dengan Variasi Epoch dan Patience

Variasi	**Perintah**
Epoch 20 dengan Patience 0	

Berlanjut ke halaman berikutnya

Lanjutan dari halaman sebelumnya

Variasi	Perintah
ResNet-50	

```
model = create_model(num_classes,
    model_type="resnet50", pretrained=True)
train_model(model, train_loader, val_loader
    , criterion, optimizer,
              scheduler, num_epochs=20,
    patience=0)
```

Kode 3.1: ResNet-50 Training 20 Epochs 0 Patience

EfficientNet-B0	

```
model = create_model(num_classes,
    model_type="efficientnet_b0", pretrained
    =True)
train_model(model, train_loader, val_loader
    , criterion, optimizer,
              scheduler, num_epochs=20,
    patience=0)
```

Kode 3.2: EfficientNet-B0 Training 20 Epochs 0 Patience

MobileNet_V3_Small	

```
model = create_model(num_classes,
    model_type="mobilenet_v3_small",
    pretrained=True)
train_model(model, train_loader, val_loader
    , criterion, optimizer,
              scheduler, num_epochs=20,
    patience=0)
```

Kode 3.3: MobileNet-V3-Small Training 20 Epochs 0 Patience

Epoch 100 dengan Patience 25

Berlanjut ke halaman berikutnya

Lanjutan dari halaman sebelumnya

Variasi	Perintah
ResNet-50	

```
model = create_model(num_classes,
    model_type="resnet50", pretrained=True)
train_model(model, train_loader, val_loader
    , criterion, optimizer,
              scheduler, num_epochs=100,
    patience=25)
```

Kode 3.4: ResNet-50 Training 100 Epochs 25 Patience

EfficientNet-B0	

```
model = create_model(num_classes,
    model_type="efficientnet_b0", pretrained
    =True)
train_model(model, train_loader, val_loader
    , criterion, optimizer,
              scheduler, num_epochs=100,
    patience=25)
```

Kode 3.5: EfficientNet-B0 Training 100 Epochs 25 Patience

MobileNet_V3_Small	

```
model = create_model(num_classes,
    model_type="mobilenet_v3_small",
    pretrained=True)
train_model(model, train_loader, val_loader
    , criterion, optimizer,
              scheduler, num_epochs=100,
    patience=25)
```

Kode 3.6: MobileNet-V3-Small Training 100 Epochs 25 Patience

3.3.4 Evaluasi Model

Evaluasi dapat dilakukan dengan memperhatikaan *Precision, Recall* dan mAP(*mean Average Precision*) yang didapatkan dari proses *training* masing-masing model.

3.4 Perancangan *Flowchart* Sistem

Flowchart sistem deteksi penyakit melon menggunakan CNN dirancang untuk menggambarkan alur kerja lengkap dari proses training model deep learning. Sistem dimulai dengan tahap inisialisasi dimana semua library yang diperlukan seperti PyTorch, NumPy, OpenCV, dan library pendukung lainnya diimpor ke dalam sistem. Konstanta-konstanta penting seperti random seed untuk reproducibility, device selection (CPU/GPU), dan parameter training juga dideklarasikan pada tahap ini.

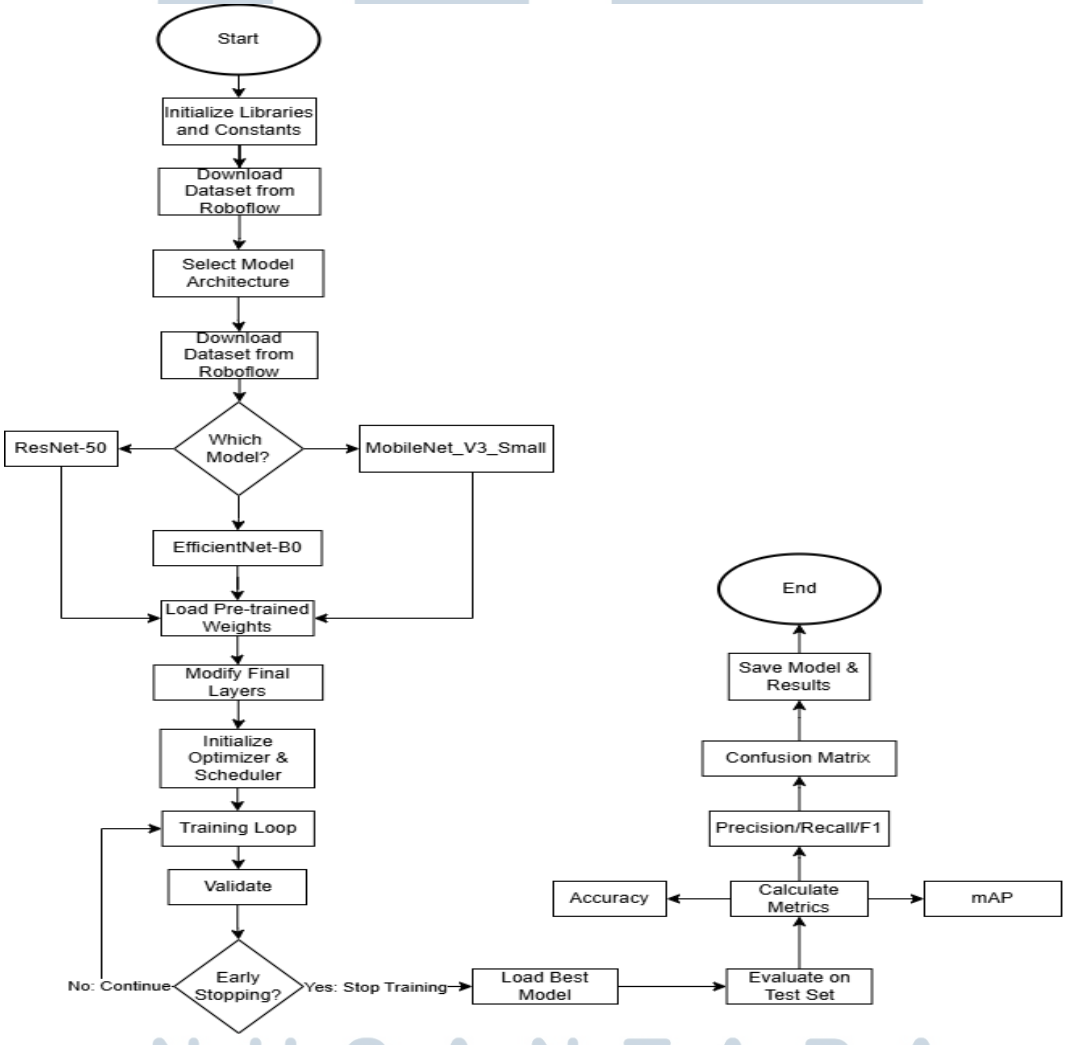
Proses *training* model CNN untuk klasifikasi penyakit tanaman melon dimulai dengan inisialisasi *library* dan konstanta yang diperlukan. Tahap awal meliputi *download dataset* dari Roboflow yang berisi gambar-gambar penyakit melon yang telah dianotasi. Setelah pemilihan arsitektur model, *dataset* dari Roboflow di-*download* kembali untuk memastikan konsistensi data yang digunakan dalam *training*.

Pada tahap pemilihan arsitektur model, terdapat titik percabangan dimana sistem dapat memilih salah satu dari tiga arsitektur CNN: ResNet-50, EfficientNet-B0, atau MobileNet_V3_Small. Ketiga arsitektur ini memiliki *trade-off* berbeda antara akurasi dan efisiensi komputasi. Setelah arsitektur dipilih, *pre-trained weights* dari ImageNet dimuat untuk memanfaatkan *transfer learning*, kemudian *final layers* dimodifikasi sesuai dengan jumlah kelas penyakit melon yang akan diklasifikasi.

Proses *training* menggunakan *optimizer* AdamW dan *scheduler* CosineAnnealingLR untuk mengatur *learning rate* secara dinamis. *Training loop* berjalan secara iteratif, dimana setiap *epoch* melakukan *forward pass* untuk mendapatkan prediksi, menghitung *loss* menggunakan *CrossEntropyLoss* berbobot, *backward pass* untuk menghitung *gradients*, dan *update weights*. Validasi dilakukan setelah setiap *epoch* untuk memantau performa model pada data yang tidak terlihat selama *training*. *Early stopping mechanism* diterapkan untuk menghentikan *training* jika *validation loss* tidak mengalami perbaikan selama

periode *patience* yang ditentukan, sehingga mencegah *overfitting* dan menghemat waktu komputasi.

Setelah *training* selesai atau dihentikan oleh *early stopping*, sistem memuat model *checkpoint* terbaik yang telah disimpan selama proses *training*. Model terbaik ini kemudian dievaluasi secara komprehensif pada *test set* untuk menghasilkan metrik performa seperti *accuracy*, *precision*, *recall*, *F1-score* per kelas, *confusion matrix*, dan *mean Average Precision* (mAP). Hasil evaluasi beserta model final disimpan dalam format PyTorch (.pth) untuk *deployment* dan analisis lebih lanjut. *Flowchart* lengkap dari proses ini dapat dilihat pada Gambar 3.3.



Gambar 3.3. *Flowchart* Sistem Deteksi Penyakit pada Tanaman Melon