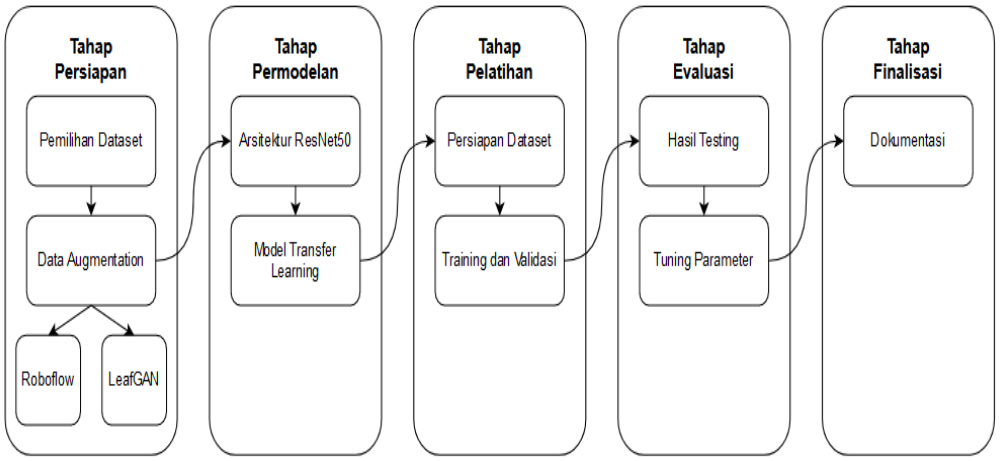
BAB III

ANALISIS DAN PERANCANGAN SISTEM

3.1 Metode Penelitian

Pada penelitian ini, terdapat lima tahap yang akan dilakukan dalam penelitian, tahap-tahap tersebut adalah tahap persiapan, tahap permodelan, tahap pelatihan, tahap evaluasi, dan tahap finalisasi. Alur pengerjaan dan tahap yang dilakukan selama penelitian dapat dilihat pada gambar 3.1 dibawah ini.



Gambar 3. 1 Tahapan Pengerjaan Penelitian

Penelitian dilakukan dengan menggunakan bahasa pemrograman Python dan environment pengembangan Jupyter Notebook. Penulis menggunakan Google Colab selama penelitian, penggunaan *Google Colab* didasarkan oleh ketersediaan perangkat keras seperti CPU dan GPU yang dapat dengan gratis dimanfaatkan untuk penelitian yang dilakukan, selain itu integrasi *Colab* dengan *Google Drive* juga mempermudah perpindahan data, serta *Google Colab* telah mendukung *framework TensorFlow* dan pengembangan dalam *Google Colab* didasarkan basis *Jupyter Notebook*.

Untuk pembuatan model klasifikasi citra digunakan berbagai *library* yang mendukung untuk pembuatan model *deep learning*, seperti dengan *framework* TensorFlow, *high-level API Keras*, dan *library-library* pendukung lainnya.

3.2 Tahap Persiapan

3.2.1 Pemilihan Dataset

Dataset yang digunakan didapatkan dari penelitian sebelumnya melalui platform Mendeley [44] dan merupakan dataset *opensource* yang disediakan penelitian sebelumnya oleh Dr. Kusrini dari AMIKOM Yogyakarta yang dijadikan sebagai referensi [4].

Dataset yang digunakan memiliki dua versi, dimana versi yang pertama merupakan dataset asli tanpa proses augmentasi data yang berisikan total 510 gambar daun mangga yang terdiri dari 16 kelas hama yang berbeda dengan pembagian 310 gambar sebagai *training data*, 103 gambar sebagai *validation data*, serta 97 gambar digunakan sebagai *testing data*, dan untuk versi kedua merupakan dataset yang telah melalui proses *data augmentation* pada gambar *training* sebanyak 150x lipat dari jumlah aslinya sehingga isinya berkembang menjadi 46,700 gambar dengan pembagian 46,500 gambar sebagai *training data* yang telah diaugmentasi dan pembagian *validation* serta *testing* tetap sama seperti pada versi pertama.

Tabel 3. 1 Distribusi Kelas Hama

No	Nama Kelas	Jumlah Training	Jumlah Valid	Jumlah Test	Total per Kelas
1	Apoderus Javanicus	38	13	12	**63**
2	Aulacaspis Tubercularis	8	3	2	**13**
3	Ceroplastes Rubens	7	2	2	**11**
4	Cisaberoptus Kenyae	12	4	3	**19**
5	Dappula Tertia	23	8	7	**38**
6	Dialeuropora Decempuncta	18	6	6	**30**

7	Erosomyia Sp	8	3	2	**13**
8	Icerya Seychellarum	17	5	5	**27**
9	Ischnaspis Longirostris	3	2	2	**7**
10	Mictis Longicornis	51	18	17	**86**
11	Neomelicharia Sparsa	22	7	7	**36**
12	Normal	22	7	7	**36**
13	Orthaga Euadrusalis	14	4	4	**22**
14	Procontarinia Matteiana	17	5	5	**27**
15	Procontarinia Rubus	16	5	5	**26**
16	Valanga Nigricornis	34	11	11	**56**
	Jumlah per Split	**310**	**103**	**97**	

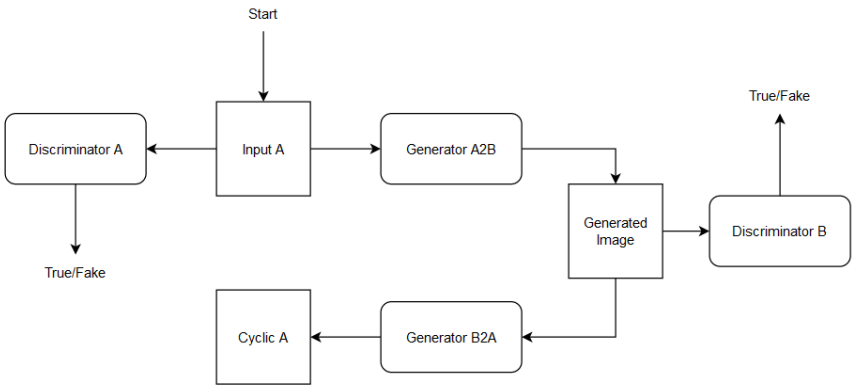
Penulis menggunakan kedua *dataset* yang disediakan baik versi pertama yag merupakan *dataset* asli tanpa augmentasi dan versi kedua yang merupakan *dataset* dengan augmentasi sebagai perbandingan serta penulis melakukan *data augmentation* sendiri dari *dataset* asli untuk penelitian lebih lanjut dengan memanfaatkan *platform Roboflow* untuk mempermudah proses penyimpanan dan perubahan *dataset* yang diaugmentasi.

3.2.2 Data Augmentation

Dengan menggunakan *dataset* versi pertama yang merupakan *dataset* asli tanpa data augmentation sebelumnya, penulis memanfaatkan metode *GAN* [5] sebagai salah satu metode augmentasi serta memanfaatkan fitur-fitur yang disediakan oleh *platform Roboflow* untuk memperluas dan mengembangkan dataset asli sehingga mempopulasi *dataset* dengan gambar yang telah

ditransformasi sehingga meningkatkan jumlah data menjadi cukup banyak untuk dapat digunakan ke dalam model deep learning nantinya.

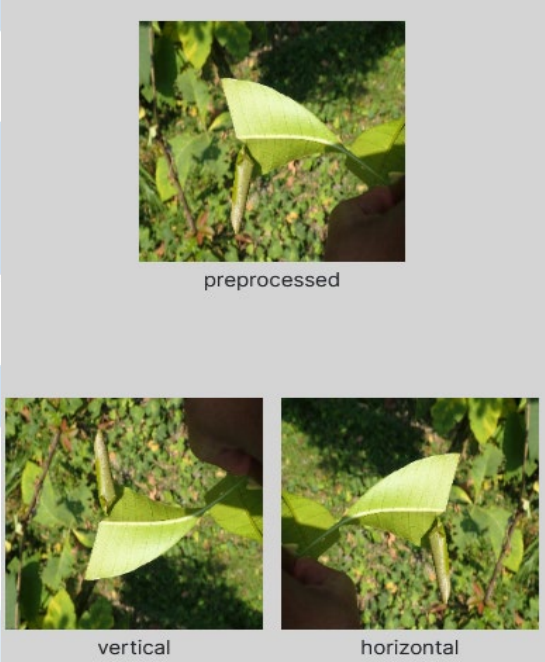
Augmentasi dengan *GAN* dicoba dengan memasukkan data pelatihan dan data percobaan ke dalam folder berbeda untuk menghasilkan citra rekayasa yang menyerupai citra asli yang dimasukkan, varian dari *GAN* yang digunakan adalah modifikasi dari *CycleGAN* dan disebut sebagai *LeafGAN* [45], *leafGAN* dimodifikasi untuk mempelajari dan menghasilkan karakteristik dari daun-daun seperti bentuk, warna, dan tulang daun. Cara kerja dari *leafGAN* serupa dengan *CycleGAN* dan dapat terlihat dari gambar dibawah ini.



Gambar 3. 2 Diagram Kerja LeafGAN

Dari diagram, cara kerja *leafGAN* adalah terdapat gambar A dan gambar B, model menerima gambar A untuk selanjutnya diubah oleh Generator A2B menjadi gambar baru yang merupakan campuran antara *domain* A dengan *domain* gambar B, hasil *generated image* dicoba untuk dibedakan oleh Discriminator B sebagai asli atau palsu, dan Generator B2A mengubah kembali *generated image* menyerupai gambar A semula. Proses pelatihan berulang ini bertujuan untuk *leafGAN* menemukan pola dari kedua gambar asli, sehingga gambar buatan menjadi sulit untuk dibedakan oleh Discriminator karena memiliki karakteristik yang serupa dengan gambar asli.

Untuk transformasi citra yang dilakukan pada *platform Roboflow* meliputi *N-degree rotation, 90-degree rotation, image flip, brightness adjustment, noise, blur,* serta gabungan dari beberapa transformasi citra tersebut. Berbagai versi augmentasi dataset yang dihasilkan berjumlah 3x lipat dari aslinya, dan selain transformasi citra, *platform Roboflow* juga menyediakan fitur *image preprocessing* seperti *image resize, auto-orient image*, dan lain-lain. Gambar 3.3 menunjukkan *preview* dari hasil augmentasi *roboflow.*



Gambar 3. 3 Preview Augmentasi Roboflow

Pemilihan *platform Roboflow* dikarenakan kemudahan pilihan untuk melakukan *preprocessing image* dan augmentasi data dengan *user interface* yang mudah dimengerti, serta pada setiap versi augmentasi yang dihasilkan akan tetap tercatat pada *workspace* sehingga memudahkan *tracking data augmentation* yang telah dilakukan [43].

Augmentasi yang terdapat pada *platform Roboflow* didasari dari fungsi-fungsi library *OpenCV,* dan *Scikit* dengan distribusi *random* yang dikemas ke dalam *user interface* interaktif untuk

kemudahan augmentasi, opsi pada augmentasi seperti tingkat *brightness* maupun tingkat *blur* telah dipermudah dengan menyesuaikan tingkat batasan mengikuti nilai *input user* dengan metrik persentase, pixel, dan sebagainya.

3.3 Tahap Permodelan

3.3.1 Pemilihan Arsitektur Deep Learning

Arsitektur atau model deep learning yang akan digunakan adalah ResNet. ResNet sendiri merupakan perkembangan dari VGG untuk mengatasi permasalahan *exploding gradient*. ResNet memiliki beberapa variasi dengan nilai angka dibelakangnya merujuk kepada tingkatan layer yang terdapat pada modelnya, variasi ResNet antara lain ResNet50, ResNet101, dan ResNet152.

Untuk varian ResNet yang digunakan adalah ResNet50 yang memiliki 50 layer, pemilihan ResNet50 didasari oleh pertimbangan performa yang dapat dicapai dari hasil-hasil yang didapatkan oleh berbagai penelitian sebelumnya [6] [9] [10].

Model ResNet50 yang diambil untuk penelitian telah disediakan oleh *TensorFlow* dan merupakan model yang telah dilatih sebelumnya pada *dataset ImageNet* serta diambil *feature vector*-nya sehingga dapat digunakan untuk *transfer learning* terutama pada penggunaan untuk klasifikasi citra.

3.3.2 Dataset Tensor Model

Pada awal penelitian, penulis menggunakan model *ResNet50* yang menerima *input* dalam bentuk nilai *array* gambar sehingga setiap gambar akan diubah menggunakan *numpy array* lalu digabungkan menjadi satu *array* yang lebih besar, cara ini dapat digunakan oleh penulis saat percobaan dengan *dataset* yang memiliki jumlah *image*

tidak terlalu banyak seperti pada versi asli (510 gambar) hingga *dataset* versi augmentasi penulis yang jumlah gambarnya dibawah puluhan ribu (1,000 sampai 3,000 gambar).

Namun cara dengan representasi *array* ini tidak dapat digunakan untuk versi augmentasi *dataset* penelitian sebelumnya yang berjumlah 46,500 gambar *training*, saat proses pengubahan gambar menjadi *array*, nilai penyimpanan memori semakin besar, sedangkan kapasitas pada *google colab* tidak mampu untuk menyimpan *array* yang banyak tersebut sehingga menyebabkan *runtime error* yang diakibatkan oleh *out of memory* atau semua memori *RAM* telah habis dan tidak mampu menampung data yang masih diproses ke dalam *array* selanjutnya. Percobaan dengan ribuan gambar pun masih dapat menyebabkan sistem *out of memory* jika dilakukan pengulangan pelatihan maupun *preprocessing* tambahan.

Solusi yang penulis temukan adalah dengan mengubah representasi *array* ke dalam bentuk *tensor* [42], dimana nilai penyusun gambar tidak diambil seutuhnya menjadi satu *array* lalu digabungkan, melainkan *tensor* berisikan nilai yang merepresentasikan *path* gambar dalam bentuk *vector,* bentuk akhir *tensor* yang digunakan untuk pelatihan adalah *rank 4 tensor* yang memiliki jumlah *batch*, lebar, tinggi dan fitur. Bentuk *tensor* ini lebih ringan dalam hal *memory* sehingga dapat digunakan untuk memproses banyak gambar pada dataset augmentasi versi dua.

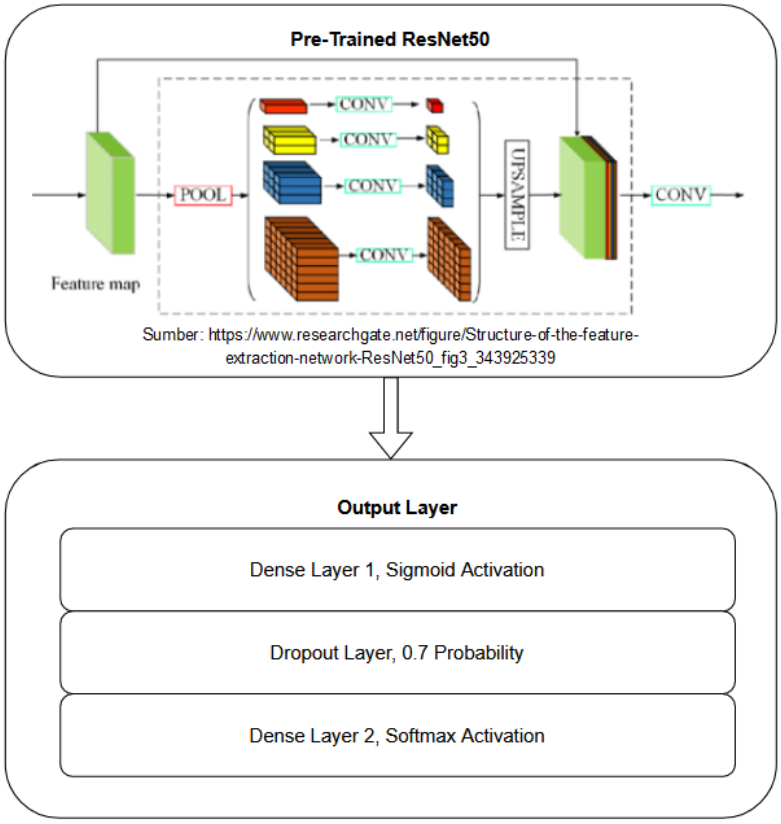
3.3.3 Transfer Learning Model

Penulis menggunakan *transfer learning* pada model *ResNet50* atas dasar pertimbangan peningkatan kecepatan pelatihan dan akurasi, hal ini telah dibuktikan penelitian [4] [6] [9] [10] [11] dimana performa model meningkat dengan penggunaan *transfer learning*. Dari bukti dan hasil tersebut, penulis memutuskan untuk

menggunakan *model pre-trained ResNet50* yang disediakan *TensorFlow*. Penulis tidak mencoba melatih model dari awal, karena performa *transfer learning* terbukti lebih baik, dan dataset penelitian berjumlah sedikit sehingga cenderung untuk *overfit*.

Transfer Learning yang digunakan adalah dari model ResNet50 yang telah dilatih sebelumnya dengan dataset *ImageNet* seperti yang digunakan penelitian serupa [9] [10], perubahan terdapat pada layer akhir model *ResNet50* dimana pada *pre-trained model* ditambahkan *dense layer* dengan aktivasi *sigmoid* dan *output layer* dengan aktivasi *softmax.*

ResNet50 yang digunakan telah dilatih sebelumnya dengan *dataset ImageNet* untuk mempelajari berbagai karakteristik objek, perbedaan *ResNet50* yang menyimpan *feature vector* dengan *ResNet50* biasa terdapat di *layer* akhir model, *fully connected layer* atau FCL yang umumnya digunakan untuk memprediksikan label kelas dihilangkan dan diganti dengan *pooling layer*, hal ini membuat karateristik yang telah dipelajari dari berbagai citra, dijadikan sebagai *vector*, nantinya *vector* yang berisikan fitur-fitur dari citra yang telah dipelajari, dapat dimanfaatkan untuk mengklasifikasi kelas lain yang belum pernah dipelajari sebelumnya dengan menambahkan *fully connected layer* baru, hal ini merupakan salah satu contoh penggunaan *transfer learning* [46]. Gambar 3.4 menunjukkan model *ResNet50* yang dipakai dengan tambahan *custom fully connected layer* untuk melakukan klasifikasi hasil.



Gambar 3. 4 Model ResNet50 yang digunakan

Selain penambahan *layer*, terdapat *hyperparameter* yang digunakan antra lain *optimizer, loss function* dan *metrics*. *Optimizer* yang digunakan adalah *Adam*, penggunaan *Adam* didasari karena menunjukkan hasil yang terbaik diantara *optimizer* lain pada penelitian ini dan pemilihan juga didasari bukti hasil penelitian perbandingan *optimizer* [47].

Untuk *loss function* yang digunakan adalah *categorical cross entropy,* hal ini didasari karena pada *dataset* hama yang digunakan terdapat 16 kelas yang berbeda atau *multi-class*. *Categorical cross entropy* bekerja dengan menentukan hasil klasifikasi menjadi satu kelas dari nilai probabilitas yang didapatkan [48].

Untuk *metrics* yang digunakan adalah akurasi, yang menghitung tingkat ketepatan antara hasil prediksi model dengan nilai label atau kelas asli objek. Diluar metrik akurasi ini, digunakan metrik

tambahan untuk evaluasi seperti *confusion matrix, precision, recall,* dan *F1-Score.*

Pada model yang digunakan juga ditambahkan fungsi *callbacks, callbacks* sendiri adalah fungsi pada *TensorFlow* untuk memonitor suatu nilai tertentu selama pelatihan model dan jika terdapat perubahan pada nilai yang tidak menunjukkan perbaikan, maka pelatihan model akan dihentikan. *Callbacks* yang digunakan pada model memonitor nilai *validation loss*, yang dimana pada pelatihan nilai tersebut bertambah maka mengarah kepada model menjadi *overfitting*, sehingga dengan waktu tunggu 3 *epoch*, *callbacks* akan memberi sinyal untuk memberhentikan pelatihan model, sehingga selain menghindari *overfitting* juga menghemat waktu *training* dan *resources* yang digunakan.

3.4 Tahap Pelatihan

3.4.1 Persiapan Dataset

Sebelum data dapat dimasukkan untuk pelatihan maupun validasi ke dalam model, terlebih dahulu dilakukan *preprocessing*, pada tahap ini *preprocessing* berfungsi untuk memastikan bahwa data yang digunakan telah memenuhi ketentuan model, seperti *shape, size, normalization,* dan sebagainya. Dikarenakan model *transfer learning* yang digunakan berbasis *feature vector*, data harus terlebih dahulu diubah menjadi bentuk *tensor.*

Pada *TensorFlow*, *tensor* adalah bentuk 1-dimensi yang merepresentasikan suatu objek, dalam hal ini objek tersebut adalah gambar atau citra yang terdapat didalam *dataset*. Penggunaan *tensor* dapat mempermudah pengolahan data yang banyak, dan cocok digunakan untuk model *deep learning* [42]. Data yang diubah menjadi *tensor* berisikan nilai representasi suatu citra dengan label asli citra

tersebut, penggabungan ini merupakan bentuk *dataset TensorFlow* dan dimungkinkan dengan fungsi yang terdapat pada *TensorFlow*.

Untuk mendapatkan dimensi yang sesuai dengan ketentuan model *transfer learning* yakni bersikan *batch size,* lebar, tinggi, dan *channel* warna, *dataset* akan menjalani tahap *preprocessing* untuk menambahkan dimensi tersebut, serta perlu dilakukan pembuatan *batch* data dengan ukuran 32 per *batch* sehingga dimensi dari *tensor* dapat diterima dan dimasukkan ke dalam model *feature vector*.

3.4.2 Training dan Validasi Dataset

Pelatihan dilakukan dengan berbagai variasi *dataset* diantaranya versi asli yang menggunakan 310 gambar yang telah tanpa penambahan melalui data augmentation, versi kedua dengan augmentasi data dari penelitian sebelumnya [4] dengan jumlah gambar *training* menjadi 46,500 setelah dilakukan data augmentation, dan berbagai versi yang dikembangkan oleh penulis untuk mengukur *benchmark* model dan dampak augmentasi data yang dilakukan. Selanjutnya untuk data validasi dilakukan dengan menggunakan 103 gambar asli yang telah dibagi sebelumnya mengikuti penelitian terdahulu [4].

Pelatihan model mengikuti ketentuan dan parameter model yang telah dijelaskan sebelumnya, dengan masing-masing iterasi pelatihan model dilakukan dengan 50 *epoch*, namun tidak jarang sebelum mencapai total 50 *epoch*, model telah selesai melakukan training untuk menghindari *overfitting* sesuai dengan fungsi *callbacks* yang digunakan.

3.5 Tahap Evaluasi

3.5.1 Hasil Testing

Hasil akurasi percobaan model didapatkan dengan memasukkan gambar *testing* yakni gambar yang tidak digunakan untuk pelatihan dan validasi sehingga model tidak pernah mempelajari maupun melihat gambar tersebut sebelumnya. Jumlah gambar *testing* yang digunakan sebesar 97 gambar yang terbagi ke dalam 16 kelas hama berbeda, baik gambar *testing* maupun *validasi* tidak dilakukan augmentasi mengikuti penggunaan penelitian terdahulu dan ketersediaan data yang diberikan oleh penelitian sebelumnya [4].

Metrik untuk evaluasi yang digunakan oleh penulis adalah tingkat akurasi, *confusion matrix, precision, recall,* dan *F1-score*.

3.5.2 Tuning Parameter

Pada tahap ini penulis melakukan fine-tuning pada parameter yang digunakan sehingga dapat memaksimalkan performa model yang dibuat sebelumnya terutama untuk dapat meningkatkan akurasi model.

Fine-tuning yang dilakukan antara lain dengan melakukan pelatihan ulang suatu model *pre-trained* setelah dilatih untuk mengklasifikasi kelas tujuan, dengan menggunakan *learning rate* yang jauh lebih rendah dari pelatihan awal sebelumnya. Selain pelatihan ulang, mencoba mengganti konfigurasi *layer* yang digunakan diharapkan dapat meningkatkan performa model secara keseluruhan.

3.6 Tahap Finalisasi

3.6.1 Dokumentasi

Dilakukan dokumentasi menyeluruh penelitian yang dijalankan berikut hasil berbentuk *datasheet*, dengan didalamnya termasuk keterangan model, parameter yang digunakan, keterangan *dataset*, dan berbagai data pendukung lainnya seperti grafik.

Penulis juga menyimpan beberapa model hasil pelatihan yang menunjukkan performa terbaik sehingga model dapat digunakan untuk inferensi dengan data percobaan baru,