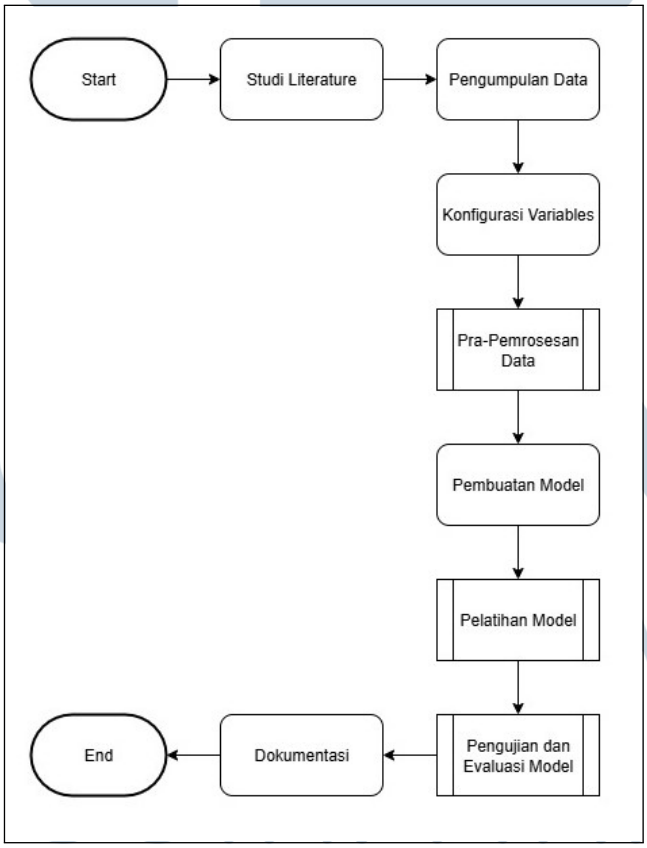
BAB 3
METODOLOGI PENELITIAN

3.1 Gambaran Umum

Tahap pertama yang dilakukan pada penelitian adalah studi literatur untuk mencari penelitian yang relevan serta mengumpulkan data yang akan digunakan. Lalu mengkonfigurasi *variables* serta melakukan pemisahan dan augmentasi data pada pra-pemrosesan gambar. Setelah pra-pemrosesan, dilakukan *5-fold cross validation* untuk mencari parameter terbaik. Selanjutnya, parameter terbaik model akan diterapkan pada pelatihan model *fine tuning*. Model yang sudah dilatih akan dievaluasi menggunakan *confusion matrix* untuk mengukur nilai *accuracy*, *precision*, *recall*, dan *F1-score*. Gambar 3.1 merupakan *flowchart* gambaran umum penelitian.



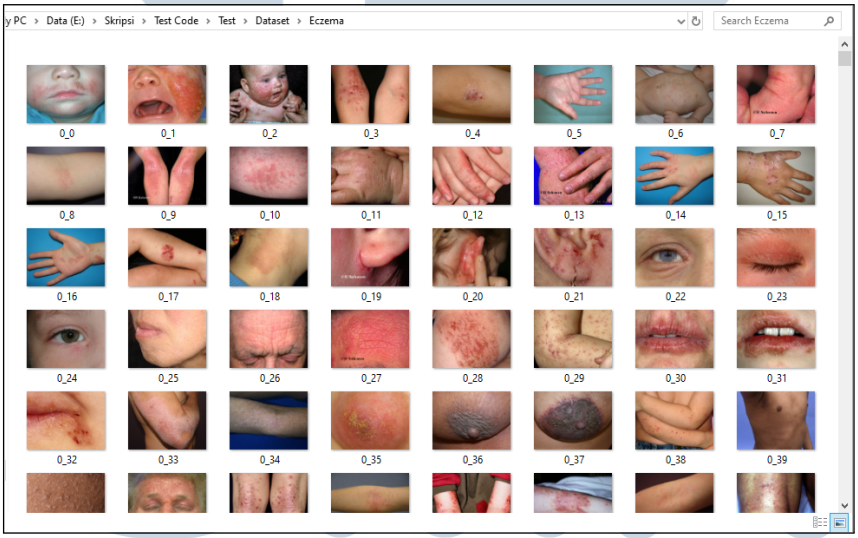
Gambar 3.1. Flowchart Gambaran Umum Penelitian

3.1.1 Studi Literature

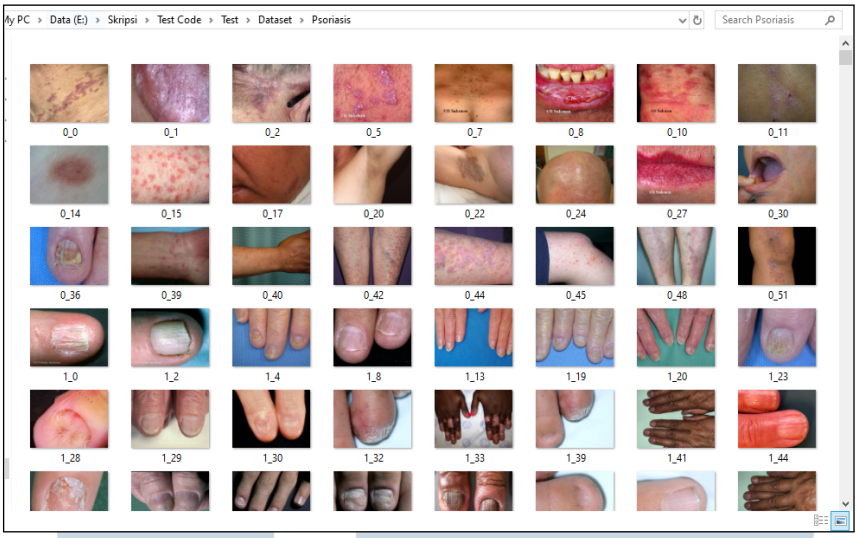
Penelitian ini dimulai dengan melakukan studi literatur yang mencakup berbagai sumber ilmiah yang relevan dengan klasifikasi penyakit kulit berbasis gambar, *Convolutional Neural Network* (CNN) dengan penerapan *transfer learning* pada arsitektur *ResNet152*.

3.1.2 Pengumpulan Data

Dataset yang digunakan dalam penelitian ini bernama *Skin Disease Image Dataset* yang diambil dari platform Kaggle. Dataset ini telah tersedia sejak tahun 2021 [47]. Dataset *Skin Disease Image Dataset* berisi 1.677 gambar penyakit kulit *eczema* dan 2.055 gambar penyakit kulit *psoriasis*, sehingga total seluruh gambar adalah sebanyak 3.732 gambar. Gambar 3.2 menunjukkan isi dari folder *eczema*, dan Gambar 3.3 menunjukkan isi dari folder *psoriasis*.



Gambar 3.2. Isi File Eczema



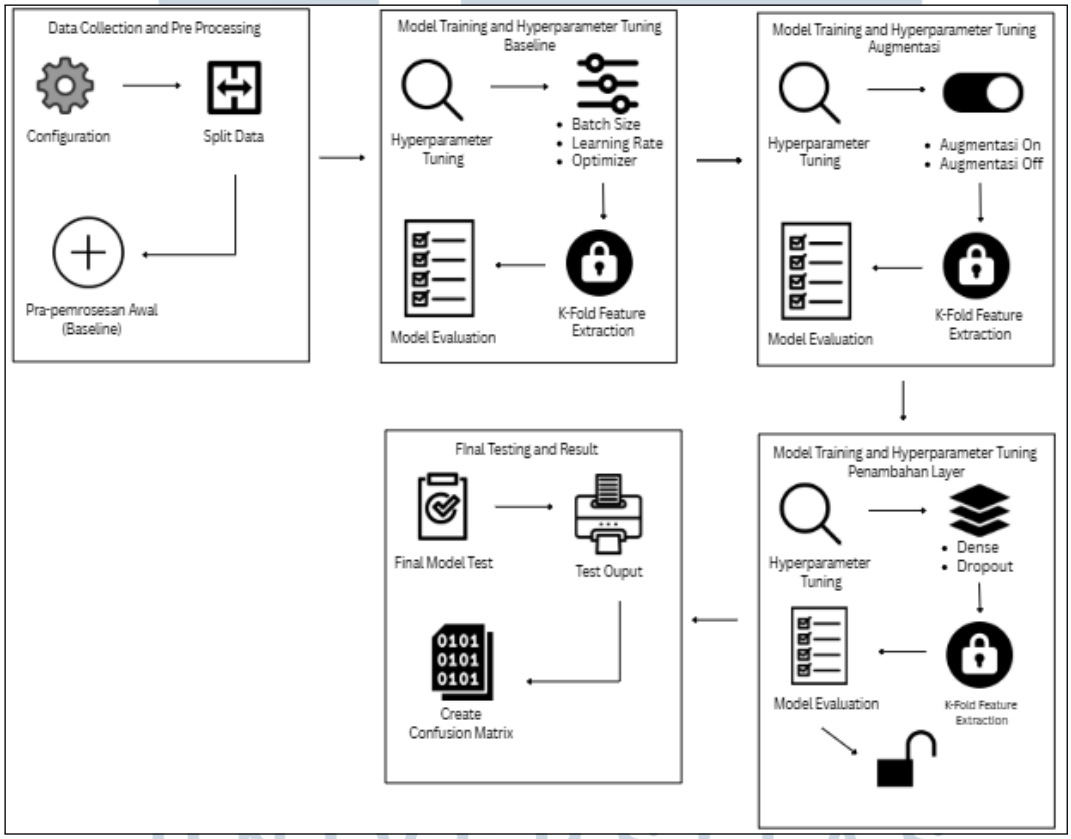
Gambar 3.3. Isi File Psoriasis

3.2 Perancangan Model

Perancangan model dalam penelitian ini dilakukan secara sistematis melalui lima tahapan utama seperti yang ditunjukkan pada Gambar 3.4. Setiap tahapan dirancang untuk memastikan proses pengembangan model berjalan terstruktur, terukur, dan dapat menghasilkan performa klasifikasi yang optimal. Tahap pertama adalah *Data Collection and Preprocessing*, yang mencakup proses konfigurasi awal, pembagian dataset ke dalam data pelatihan dan data pengujian, serta penerapan Pra-pemrosesan Awal. Pra-pemrosesan awal meliputi penyesuaian dimensi gambar dan normalisasi menggunakan fungsi `preprocess_input` sesuai kebutuhan arsitektur ResNet152. Pada tahap ini belum dilakukan augmentasi data, karena model masih membentuk baseline awal.

Tahap kedua adalah *Model Training and Hyperparameter Tuning Baseline*, di mana dilakukan pencarian kombinasi *hyperparameter* awal yang optimal. Parameter yang diuji antara lain *batch size*, *learning rate*, dan jenis *optimizer*. Proses evaluasi dilakukan menggunakan *K-Fold Cross Validation* pada tahap *feature extraction*, untuk memperoleh model baseline terbaik tanpa augmentasi. Tahap ketiga adalah *Model Training and Hyperparameter Tuning Augmentasi*, yaitu percobaan pembandingan antara model dengan dan tanpa augmentasi. Augmentasi data dilakukan menggunakan transformasi gambar untuk meningkatkan variasi data pelatihan. Seluruh proses pelatihan tetap menggunakan skema *K-Fold Cross Validation* dan hasilnya dievaluasi berdasarkan akurasi rata-rata dan nilai *gap*.

Tahap keempat adalah *Model Training and Hyperparameter Tuning Penambahan Layer*, yaitu penambahan *dense layer* dan *dropout* untuk meningkatkan kemampuan klasifikasi. Tahap ini masih menggunakan pendekatan *feature extraction* pada awalnya, kemudian dilanjutkan dengan proses *fine-tuning* untuk membuka sebagian bobot pada backbone ResNet152 agar model dapat belajar lebih dalam. Tahap terakhir adalah *Final Testing and Result*, di mana model terbaik hasil pelatihan diuji pada data *test set* yang belum pernah digunakan sebelumnya. Hasil dari pengujian ini disebut sebagai *test output*, dan digunakan untuk menghasilkan *confusion matrix* sebagai evaluasi mendalam terhadap akurasi dan performa klasifikasi pada masing-masing kelas.



Gambar 3.4. Perancangan Model

3.2.1 Konfigurasi variables

Bagian ini merinci semua variabel konfigurasi yang digunakan dalam *pipeline*, mulai dari persiapan data hingga pelatihan dan evaluasi model. Pengaturan variabel di satu tempat memastikan konsistensi, kemudahan modifikasi, dan *reproducibility* penelitian.

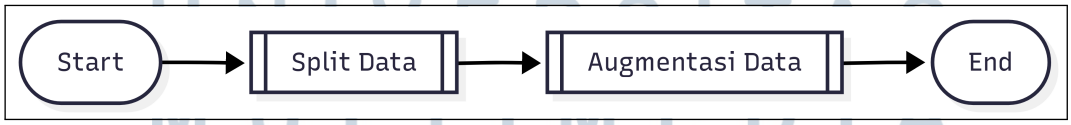
```
source_dataset_dir = '/content/drive/MyDrive/Dataset/Dataset'
initial_split_dir = '/content/drive/MyDrive/Dataset/initial_split1'
final_models_dir = '/content/drive/MyDrive/Dataset/final_models_resnet152B'
csv_fold_dir = '/content/drive/MyDrive/Dataset/cv_fold_results1'

IMG_HEIGHT = 224
IMG_WIDTH = 224
IMAGE_SIZE = (IMG_WIDTH, IMG_HEIGHT)
BATCH_SIZE = 8
N_FOLDS = 5
N_CLASSES = 1
```

Kode 3.1: Konfigurasi Variabel

Alur kerja kode pada Gambar 3.1 dimulai dengan direktori `source_dataset_dir` yang berisi dataset asli. Data ini kemudian dibagi menjadi set latih dan uji yang disimpan di direktori `initial_split_dir`. Selama proses ini, semua gambar akan diubah ukurannya menjadi `IMAGE_SIZE` sebesar 224×224 piksel, yang merupakan standar umum untuk model *pre-trained*, dan diproses dalam `BATCH_SIZE` berisi 8 gambar per iterasi. Untuk konfigurasi *Stratified K-Fold*, dilakukan sebanyak 5 *fold*. Hasil dari setiap fold akan disimpan dalam direktori `csv_fold_dir`, sedangkan hasil terbaik dari proses *K-Fold* akan disimpan dalam direktori `final_models_dir`.

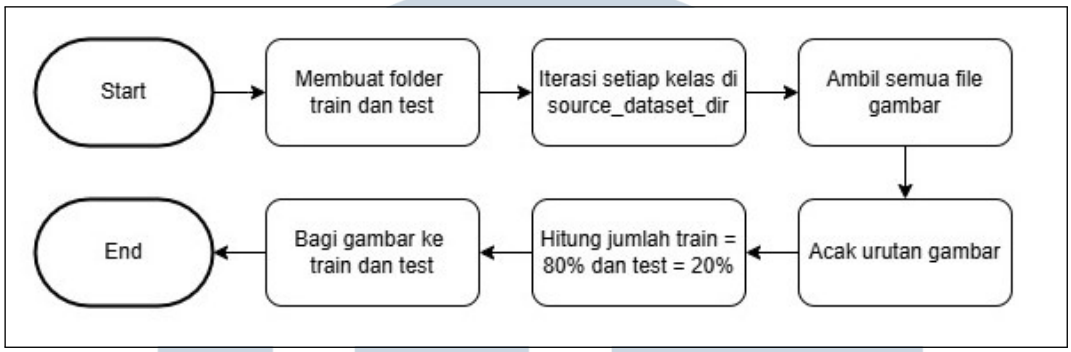
3.2.2 Pra-pemrosesan Data

Pra-pemrosesan data dilakukan untuk memastikan bahwa data berada dalam kondisi yang sesuai untuk digunakan dalam proses pelatihan dan evaluasi model. Gambar 3.5 menunjukkan alur *flowchart* dari tahapan pra-pemrosesan data.



Gambar 3.5. Flowchart Gambaran Umum Pra-Pemrosesan Data

A Split Data



Gambar 3.6. Flowchart Split Data

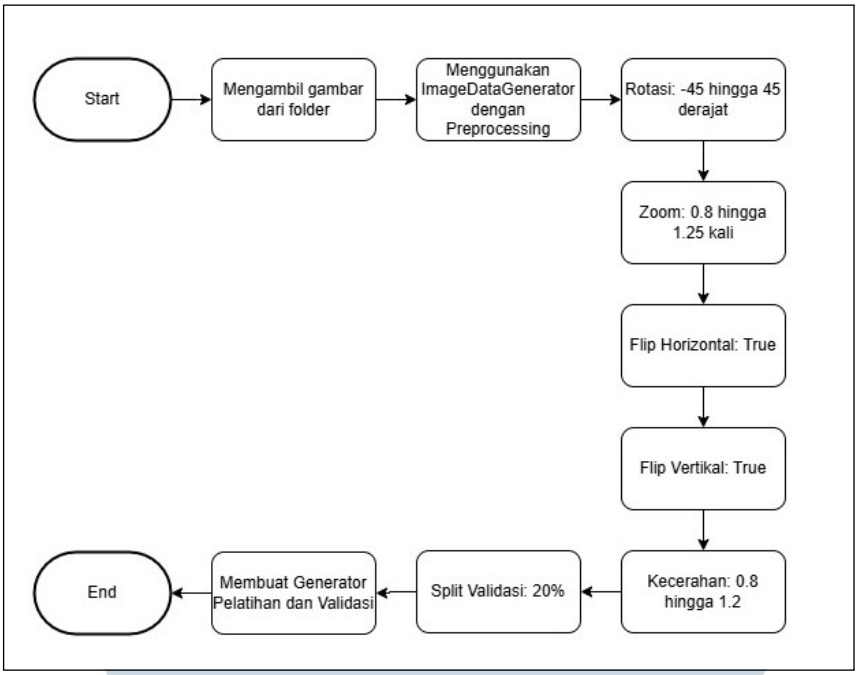
Langkah pertama adalah melakukan pemisahan *dataset* awal ke dalam dua *subset*, yaitu data pelatihan sebesar 80% dan data pengujian sebesar 20%. Proses ini dilakukan secara acak untuk setiap kelas guna menjaga distribusi kelas tetap seimbang. Direktori tujuan dibersihkan terlebih dahulu untuk mencegah duplikasi atau konflik dengan data sebelumnya. Selanjutnya, gambar dalam setiap kelas dibagi dan disalin ke dalam direktori `train` dan `test`. Hasil pembagian data ditampilkan pada Tabel 3.1, sedangkan Gambar 3.6 menggambarkan alur *flowchart* dari proses pemisahan data.

Tabel 3.1. Memulai Split Data Awal 80% Train, 20% Test

Kelas	Total Data	Train	Test
Eczema	1677	1342	335
Psoriasis	2055	1644	411

B Augmentasi Data

Penerapan augmentasi data dalam penelitian ini tidak dilakukan sejak awal, melainkan melalui pendekatan eksperimental yang terstruktur untuk memastikan dampaknya terhadap performa model. Alur proses ini diilustrasikan pada Gambar 3.7.



Gambar 3.7. Flowchart Augmentasi Data

Langkah pertama dalam pencarian *hyperparameter* adalah membangun performa dasar atau *baseline*. Tahap ini, yang mencakup pencarian *batch size*, *learning rate*, dan *optimizer* yang optimal, sengaja dilakukan tanpa mengaktifkan augmentasi data. Tujuannya adalah untuk mendapatkan titik ukur performa model yang murni sebagai acuan untuk perbandingan.

Setelah performa dasar ditetapkan, pengaruh augmentasi dievaluasi secara spesifik. Model dilatih kembali menggunakan teknik augmentasi, dan hasilnya dibandingkan secara langsung dengan performa *baseline*. Proses ini membuktikan seberapa efektif augmentasi dalam meningkatkan kemampuan generalisasi model.

Untuk implementasinya, augmentasi data pada data pelatihan dilakukan secara *on-the-fly* menggunakan *ImageDataGenerator* dari *Keras*. Teknik yang diterapkan mencakup rotasi acak, *flipping* horizontal dan vertikal, serta variasi kecerahan, seperti yang dirinci pada Tabel 3.2. Sementara itu, data validasi dan data pengujian hanya dikenai proses *preprocessing* tanpa augmentasi untuk menjaga kemurnian evaluasi. Seluruh gambar juga diproses menggunakan fungsi `preprocess_input` dari *ResNet152* untuk menyesuaikan format input dengan arsitektur pralatih tersebut.

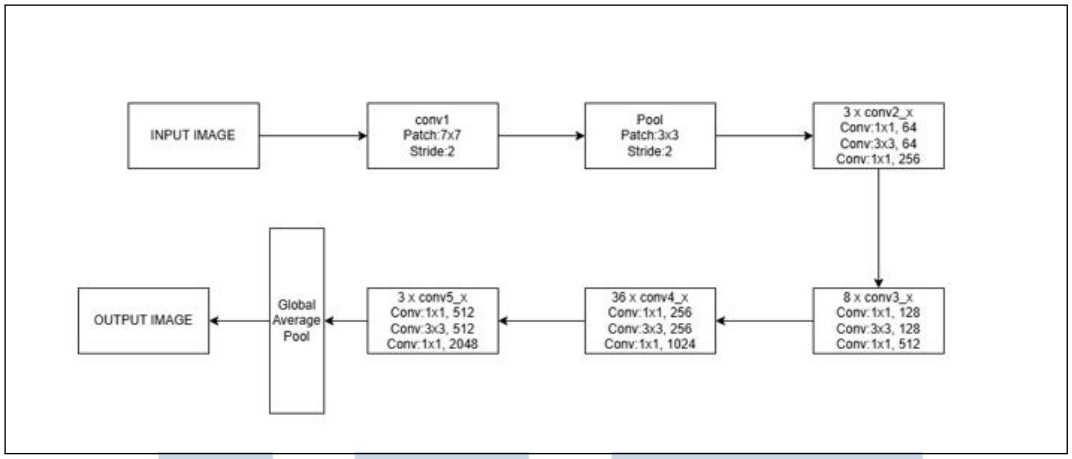
Tabel 3.2. Parameter Augmentasi Data yang Digunakan

Parameter	**Penjelasan**
`rotation_range=45`	Memutar gambar secara acak hingga ±45 derajat.
`horizontal_flip=True`	Membalik gambar secara horizontal (kiri dan kanan).
`vertical_flip=True`	Membalik gambar secara vertikal (atas dan bawah).
`brightness_range=[0.8, 1.2]`	Variasi pencahayaan acak dalam rentang 80% hingga 120% dari kecerahan asli.

3.2.3 Pembuatan Model

Model klasifikasi pada penelitian ini dibangun menggunakan arsitektur *ResNet152* sebagai *base model* yang telah dilatih sebelumnya menggunakan dataset ImageNet. Proses diawali dengan memuat model dasar tanpa bagian *top layer* (`include_top=False`) dan parameter `trainable=False`, sehingga seluruh bobot pada model dasar dibekukan untuk keperluan *feature extraction*. Setelah itu, ditambahkan beberapa lapisan klasifikasi kustom (*custom head*) di atas output model dasar. Lapisan pertama yaitu `GlobalAveragePooling2D` yang berfungsi untuk meratakan fitur spasial dari output konvolusional terakhir menjadi vektor satu dimensi. Selanjutnya, digunakan satu lapisan `Dense` dengan 1 unit serta fungsi aktivasi sigmoid untuk melakukan klasifikasi biner.

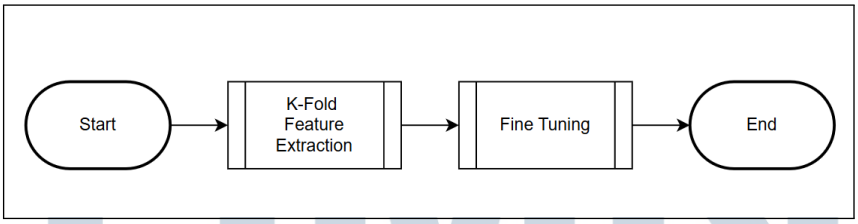
Model ini digunakan pada tahap awal pelatihan feature extraction. Jika pada tahap selanjutnya dilakukan *fine-tuning*, maka sebagian lapisan pada model dasar dapat diaktifkan kembali (`trainable=True`) agar dapat menyesuaikan bobot terhadap karakteristik dataset yang digunakan. Pada gambar 3.8 adalah lapisan-lapisan dari resnet152.



Gambar 3.8. Lapisan Konvulusi ResNet152

3.2.4 Pelatihan Model

Proses pelatihan model pada penelitian ini dilakukan melalui dua tahap utama, yaitu *Pencarian Hyperparameter dengan K-Fold Feature Extraction* dan *Fine-Tuning*, yang merupakan bagian dari pendekatan *transfer learning*. Gambar 3.9 menunjukkan alur *flowchart* dari proses pelatihan model secara keseluruhan.



Gambar 3.9. Flowchart Gambaran Umum Pelatihan Model

A Pencarian Hyperparameter dengan K-Fold Feature Extraction

Tahap pertama dari pelatihan model berfokus pada pencarian *hyperparameter* terbaik secara sistematis. Proses ini dilakukan pada tahap *feature extraction*, di mana seluruh bobot pada model dasar ResNet152 dibekukan (`trainable=False`). Tujuannya adalah untuk mengoptimalkan kinerja *custom head* atau lapisan klasifikasi tambahan yang dilatih di atasnya, sebelum melangkah ke tahap *fine-tuning*.

Untuk mendapatkan evaluasi yang *robust* pada setiap konfigurasi, penelitian ini menggunakan metode *5-Fold Stratified Cross Validation*. Dalam metode ini,

data pelatihan dibagi menjadi lima bagian (*fold*) yang proporsional, di mana model dilatih sebanyak lima kali dan setiap *fold* secara bergantian digunakan sebagai data validasi. Kinerja model dinilai berdasarkan rata-rata akurasi validasi dan stabilitas model, yang diukur dari *gap* (selisih) antara akurasi pelatihan dan akurasi validasi.

Pencarian *hyperparameter* dilakukan secara bertahap dan berurutan, di mana konfigurasi terbaik dari satu tahap akan menjadi dasar untuk tahap pengujian berikutnya. Urutan pencarian yang dilakukan adalah sebagai berikut:

1. *Batch Size*: Menguji beberapa nilai *batch size* (64, 32, 16, dan 8) dengan konfigurasi awal yaitu menggunakan *optimizer* Adam, *learning rate* 0,001, tanpa augmentasi aktif, dan tanpa *dense layer* tambahan atau *dropout*. Tujuannya adalah untuk menentukan ukuran batch yang paling efektif bagi pelatihan awal.

2. *Learning Rate*: Dengan menggunakan *batch size* terbaik dari tahap sebelumnya, dilakukan pengujian terhadap beberapa nilai *learning rate* (0,001 dan 0,0001) untuk menentukan laju pembaruan bobot yang paling optimal.

3. *Optimizer*: Setelah memperoleh kombinasi *batch size* dan *learning rate* terbaik, dilakukan perbandingan performa antara dua jenis *optimizer*, yaitu Adam dan Nadam, untuk memilih algoritma optimasi yang paling cocok.

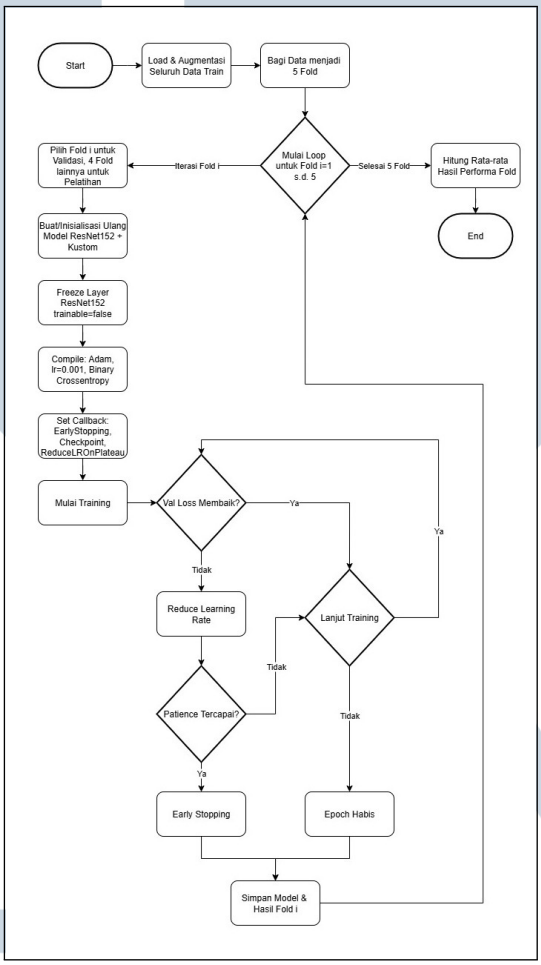
4. Augmentasi Data: Pengaruh augmentasi data dievaluasi dengan membandingkan performa model yang dilatih dengan dan tanpa augmentasi aktif. Pengujian ini dilakukan menggunakan konfigurasi *hyperparameter* terbaik yang telah diperoleh dari tahapan sebelumnya.

5. Penambahan Jumlah Neuron: Setelah augmentasi diaktifkan, dilakukan pengujian terhadap beberapa variasi jumlah neuron pada *dense layer* tambahan, yaitu 128, 256, 384, dan 512, untuk menentukan kapasitas representasi fitur yang paling sesuai.

6. Tingkat *Dropout*: Dengan jumlah neuron terbaik yang telah diperoleh, dilakukan evaluasi terhadap beberapa nilai *dropout rate* (berkisar dari 0,1 hingga 0,5) untuk mendapatkan strategi regularisasi yang paling efektif dalam mengurangi risiko.

Pada setiap tahap pencarian *hyperparameter*, model dilatih selama maksimal 50 *epoch*. Performa model dievaluasi berdasarkan rata-rata *validation accuracy*

dari kelima *fold*, serta nilai *gap* (selisih antara *training accuracy* dan *validation accuracy*) untuk mengukur kemampuan generalisasi model terhadap data yang belum pernah dilihat sebelumnya. Model dengan akurasi validasi rata-rata tertinggi dari seluruh *fold* pada tahap ini disimpan dan digunakan sebagai dasar untuk proses *fine-tuning* pada tahap pelatihan berikutnya. Alur proses tahap ini digambarkan dalam Gambar 3.10.

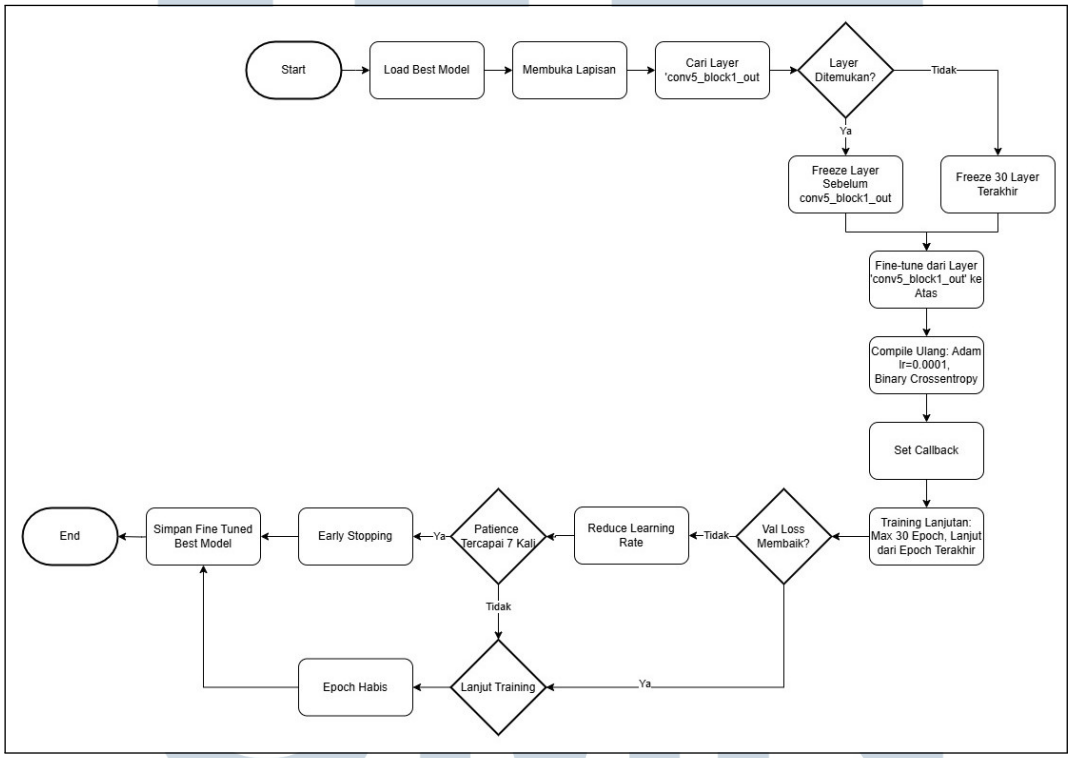


Gambar 3.10. Flowchart Feature Extraction

B Fine Tuning

Setelah mendapat parameter terbaik, tahap kedua dilakukan untuk menyempurnakan representasi fitur pada bagian atas model dasar agar lebih spesifik terhadap karakteristik *dataset* penyakit kulit yang digunakan. Model terbaik dari tahap *feature extraction* dimuat kembali, dan sebagian bobot pada model dasar diaktifkan kembali dengan `trainable = True`, namun hanya pada lapisan-lapisan

atas, dimulai dari blok `conv5_block1_out` hingga akhir. Lapisan-lapisan yang lebih dalam tetap dibekukan untuk menjaga fitur general yang telah dipelajari sebelumnya. Selanjutnya, model dikompilasi ulang dengan *learning rate* yang jauh lebih kecil guna menghindari perubahan drastis yang dapat merusak bobot yang sudah optimal. Pelatihan dilanjutkan dari *epoch* yang sama menggunakan *callbacks* yang sama seperti sebelumnya untuk memantau dan mengendalikan proses pelatihan. Model hasil *fine-tuning* inilah yang kemudian digunakan sebagai model akhir untuk pengujian pada data uji. Pada Gambar 3.11 merupakan alur *flowchart* dari proses *fine-tuning*.



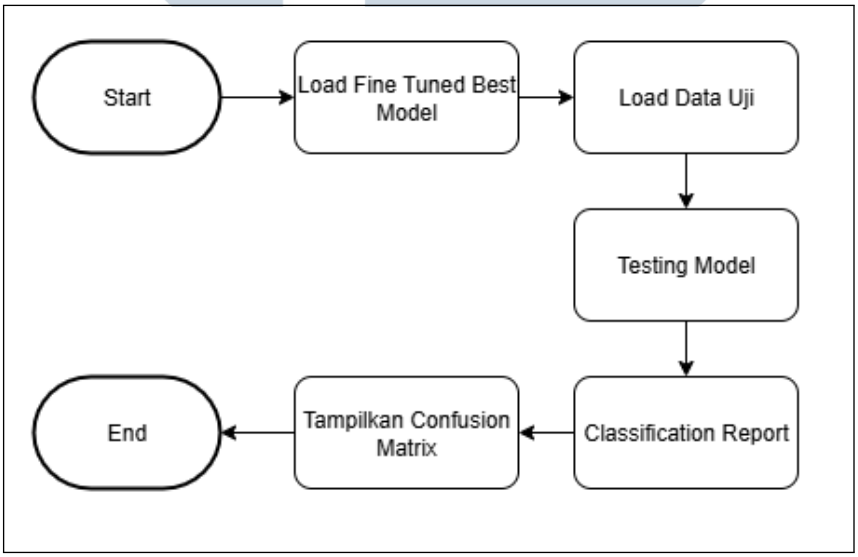
Gambar 3.11. Flowchart Fine Tuning

3.2.5 Pengujian dan Evaluasi Model

Evaluasi akhir dilakukan untuk mengukur performa model terbaik yang diperoleh dari tahap *fine-tuning* dengan menggunakan data uji yang sama sekali belum pernah dilihat oleh model selama proses pelatihan maupun validasi. Model terbaik yang digunakan dalam evaluasi ini dipilih dari model hasil *fine-tuning*, dan jika model tersebut tidak tersedia, maka digunakan model hasil *feature extraction*. Data uji yang digunakan sebesar 20% dari data asli dan tidak melalui proses

augmentasi.

Evaluasi dilakukan dengan beberapa metrik utama. Pertama, metrik *Loss* dan *Accuracy* dihitung secara langsung dari performa model pada data uji untuk memberikan gambaran umum mengenai efektivitas klasifikasi. Selanjutnya, dilakukan evaluasi lebih mendalam menggunakan *Classification Report*, yang mencakup nilai *Precision*, *Recall*, dan *F1-score* untuk masing-masing kelas. Hal ini memberikan informasi tentang ketepatan dan sensitivitas model dalam mengklasifikasikan gambar *eczema* dan *psoriasis*. Terakhir, performa klasifikasi divisualisasikan menggunakan *Confusion Matrix*, yang menunjukkan jumlah prediksi benar dan salah untuk masing-masing kelas, termasuk nilai *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN). Visualisasi *confusion matrix* disajikan dalam bentuk diagram berwarna untuk memudahkan interpretasi. Evaluasi ini memberikan gambaran menyeluruh terhadap kemampuan generalisasi model pada data baru yang belum pernah dilatih sebelumnya. Pada Gambar 3.12 merupakan *flowchart* proses pengujian dan evaluasi.



Gambar 3.12. Flowchart Evaluasi Model

3.2.6 Dokumentasi

Penelitian ini didokumentasikan secara menyeluruh mulai dari tahap studi literatur, pengumpulan dan pengolahan data, pelatihan model, hingga tahap analisis hasil pengujian dan evaluasi performa model.