BAB III
METODE PENELITIAN

3.1 Objek Penelitian

Objek utama dari penelitian ini adalah citra medis berupa gambar lesi pada permukaan kulit manusia yang menunjukkan tanda-tanda atau gejala khas cacar monyet. Citra medis ini akan digunakan sebagai data utama atau untuk mendeteksi apakah suatu gambar menunjukkan adanya gejala cacar monyet atau tidak.

3.2 Lokasi dan Waktu Penelitian

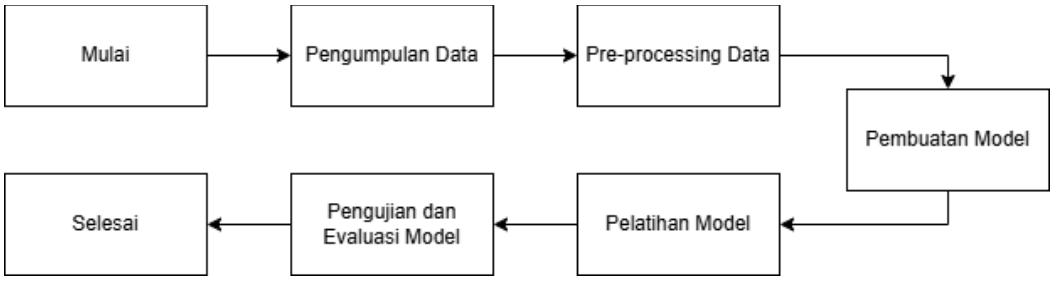
Penelitian ini akan dilaksanakan di Laboratorium Universitas Buana Perjuangan Karawang. Penelitian direncanakan berlangsung selama 5 bulan, dimulai pada bulan Oktober 2024 hingga Februari 2025. Proses penelitian akan melalui beberapa tahapan, mulai dari pengumpulan data hingga penyusunan laporan. Rincian tahapan penelitian dapat dilihat pada Tabel 3.1.

Tabel 3. 1 Tahapan Penelitian

No	Tahapan Penelitian	2024												2025							
		Oktober				November				Desember				Januari				Februari			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Pengumpulan Data																				
2	Pra-proses Data																				
3	Pembuatan Model																				
4	Pelatihan Model																				
5	Evaluasi Model																				
6	Penyusunan Laporan																				

3.3 Rancangan Penelitian

Penelitian ini akan dilaksanakan melalui beberapa tahapan yang sistematis dan berurutan, dimulai dari pengumpulan data, *pre-processing* data, pembuatan model, pelatihan model, hingga pengujian dan evaluasi model. Rancangan penelitian dapat dilihat pada gambar 3.1.



Gambar 3. 1 Rancangan Penelitian

3.4 Pengumpulan Data

Pada gambar 3.2, dapat dilihat data yang akan digunakan dalam penelitian adalah citra medis yang menunjukkan tanda-tanda cacar monyet pada manusia. Dataset yang akan digunakan berjumlah 316 gambar yang terdiri dari gambar-gambar yang dilabeli seperti cacar monyet dan bukan cacar monyet. Gambar-gambar ini berasal dari *kaggle*. Dataset ini disimpan dalam 2 folder yaitu folder *Monkey Pox* dan *Others*.

(A) (B)

Gambar 3. 2 Dataset

(A): gambar *monkeypox*.
(B): gambar *others* (bukan *monkeypox*)
Sumber : Kaggle

3.5 *Pre-Processing* Data

Data yang telah dikumpulkan akan melalui tahap *pre-processing* yang meliputi langkah-langkah penting untuk memastikan data yang digunakan dalam pelatihan model memenuhi standar kualitas yang diperlukan. Proses *pre-processing* ini bertujuan untuk mempersiapkan data sehingga model dapat diproses dengan baik. Berikut adalah langkah-langkah yang akan dilakukan.

a. Penepatan Parameter Acak

Penetapan parameter acak merupakan langkah penting dalam proses eksperimen *machine learning* untuk memastikan hasil yang konsisten dan dapat direproduksi. Dalam konteks penelitian, replikabilitas menjadi aspek krusial agar eksperimen dapat diuji ulang dengan hasil yang sama.

Pada penelitian ini, parameter acak (*random seed*) ditetapkan pada dua pustaka utama yang digunakan, yaitu *NumPy* dan *TensorFlow*. *NumPy* digunakan untuk operasi numerik dan manipulasi *array*, sedangkan *TensorFlow* merupakan kerangka kerja utama dalam membangun dan melatih model CNN.

Tahapan penetapan parameter acak dilakukan sebagai berikut:

1. *NumPy:* np.random.seed(1) digunakan untuk mengatur *seed* acak, sehingga proses seperti pembangkitan angka acak dan pengacakan data berjalan secara deterministik.
2. *TensorFlow:* tf.random.set_seed(1) digunakan untuk mengatur elemen-elemen acak dalam *TensorFlow*, seperti inisialisasi bobot model, pengacakan data saat pelatihan, dan *dropout*.

Dengan menetapkan nilai seed yang sama pada kedua pustaka tersebut, seluruh proses pelatihan model akan bersifat deterministik, sehingga hasil eksperimen dapat direproduksi dengan konsisten setiap kali dijalankan.

b. Mengubah Ukuran Gambar

Gambar-gambar dalam dataset akan diubah ukurannya menjadi seragam misalnya 224x224 piksel, agar sesuai dengan input yang dibutuhkan. Berikut merupakan contoh *pseudocode* untuk *Resize* (mengubah ukuran gambar).

Pseudocode :

```
Mulai
    // Baca gambar dari file atau dataset
    gambar = BacaGambar("path/to/gambar")

    // Ubah ukuran gambar menjadi 224x224 piksel
    gambarResize = UbahUkuran(gambar, 224, 224)
```

```
    // Tampilkan gambar hasil resize
    TampilkanGambar(gambarResize)
Selesai
```

Hasil :

Gambar asli *Resized Image* (224x224)

Gambar 3. 3 *Resized Image*

c. *Batch Size*

Digunakan *batch size* sebesar 32 dengan tujuan untuk mempercepat proses pelatihan dan mengoptimalkan penggunaan memori GPU. Pemilihan ukuran *batch* ini merupakan kompromi antara efisiensi komputasi dan stabilitas penurunan bobot selama training. Batch size yang terlalu kecil dapat menyebabkan pelatihan menjadi lambat dan kurang stabil, sedangkan *batch size* yang terlalu besar beresiko menyebabkan kehabisan memori pada *GPU*. Oleh karena itu *batch size* 32 dipilih sebagai ukuran yang ideal untuk menjaga keseimbangan antara kecepatan pelatihan, pemanfaatan sumber daya perangkat keras dan akurasi model yang dihasilkan

d. Normalisasi

Normalisasi adalah proses mengubah nilai pixel gambar ke rentang [0, 1] dengan membagi nilai pixel dengan 255.0, yang bertujuan untuk mempercepat pelatihan model dan membuat data lebih konsisten. Berikut merupakan contoh *pseudocode* untuk Normalisasi proses mengubah nilai pixel gambar).

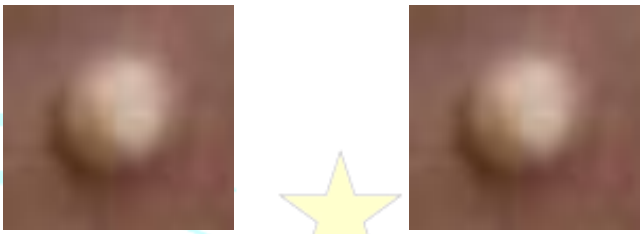
Pseudocode :

```
Mulai
    // Baca gambar dari file atau dataset
    gambar = BacaGambar("path/to/gambar")
```

```
    // Normalisasi gambar dengan membagi setiap piksel dengan 255.0
    gambarNormalisasi = NormalisasiGambar(gambar, 255.0)

    // Tampilkan gambar setelah normalisasi
    TampilkanGambar(gambarNormalisasi)
Selesai
```

Hasil :



Gambar asli *Normalized Image*

Gambar 3. 4 *Normalized Image*

e. Pembagian Data

Dataset yang berjumlah 316 gambar dibagi dengan proporsi 80% sebagai data latih dan 20% sebagai data validasi untuk memastikan model mendapatkan cukup data untuk belajar sekaligus dievaluasi selama pelatihan. Sementara itu, data uji menggunakan subset yang sama dengan data validasi karena keterbatasan jumlah dataset. Meskipun idealnya data uji bersifat terpisah, pendekatan ini tetap memberikan gambaran awal performa model terhadap data yang belum dilihat sebelumnya. Sehingga menghasilkan pembagian data sebanyak 285 gambar untuk data latih, 62 gambar untuk data validasi, dan 31 gambar untuk data uji.

3.6 Pembuatan Model

ResNet-101 adalah arsitektur yang digunakan pada penelitian ini, yang sudah dilatih sebelumnya menggunakan dataset *ImageNet* dan digunakan sebagai fitur ekstraktor. Parameter yang digunakan dalam pembuatan model adalah:

a. *Base Model*

ResNet-101 tanpa lapisan *fully connected* (*include_top=False*) dan menggunakan bobot *pre-trained* dari *ImageNet.*

b. Lapisan Tambahan

Global Average Pooling 2D untuk mereduksi dimensi fitur. *Batch Normalization* setelah setiap lapisan dense untuk meningkatkan stabilitas pelatihan. Lapisan *Dense* dengan jumlah neuron bertingkat: 1024, 512, dan 128 dengan fungsi aktivasi *ReLU*. *Dropout* dengan nilai 0.2 pada setiap lapisan *dense* untuk menghindari *overfitting*. Lapisan dari *Output* terdapat 2 *neuron* dengan aktivasi *softmax* untuk klasifikasi dua kelas (cacar monyet dan bukan cacar monyet).

c. Optimasi

Optimizer Adam dengan *learning rate* awal 5e-4. *Loss function categorical crossentropy*. Metode evaluasi berdasar dari akurasi.

3.7 Pelatihan Model

Proses Pelatihan model dilaksanakan selama 100 *epoch* dengan mekanisme pengurangan *learning rate* sebesar 0.2 jika validasi *loss* tidak mengalami peningkatan selama 3 *epoch*. Pelatihan dilakukan selama 100 epoch agar model memiliki waktu yang cukup untuk mempelajari pola dari data. Untuk meningkatkan efisiensi pelatihan, digunakan mekanisme *Reduce Learning Rate on Plateau*, yaitu penurunan learning rate sebesar 0.2 ketika *validation loss* tidak membaik selama 3 epoch. Hal ini membantu model beradaptasi lebih baik dan mencapai konvergensi yang lebih optimal.

Beberapa *callback* yang diterapkan dalam pelatihan model adalah:

Callback digunakan untuk mengatur proses pelatihan secara otomatis agar lebih efektif. Dengan adanya callback, model dapat menyimpan hasil terbaik dan menyesuaikan parameter saat diperlukan, tanpa perlu intervensi manual.

a. *Model Checkpoint*

Callback ini menyimpan bobot model terbaik selama pelatihan berlangsung. Model akan disimpan otomatis ketika akurasi validasi mencapai nilai tertinggi. Dengan begitu, meskipun pelatihan terus berjalan, pengguna tetap dapat menggunakan model terbaik yang tersimpan untuk pengujian.

b. *Reduce Learning Rate on Plateau*

Jika tidak ada peningkatan akurasi validasi dalam beberapa *epoch*, *callback* ini akan menurunkan *learning rate*. Penurunan ini membuat proses pelatihan lebih stabil dan membantu model melakukan penyesuaian parameter dengan lebih hati-hati.

Setelah pelatihan, dilakukan visualisasi hasil pelatihan dengan grafik akurasi dan *loss* pada setiap *epoch*. Visualisasi digunakan untuk melihat perkembangan akurasi dan *loss* selama pelatihan. Grafik ini membantu mengevaluasi kinerja model dan mendeteksi adanya *overfitting* atau *underfitting*, sehingga proses pelatihan dapat dianalisis dengan lebih baik.

3.8 Pengujian Dan Evaluasi Model

Pengujian model dilakukan menggunakan data uji yang diproses dengan metode yang sama seperti data latih. Sebelum digunakan dalam proses pengujian, data uji diperlakukan dengan tahapan praproses yang sama seperti data latih untuk menjaga konsistensi input terhadap model. Hal ini penting agar hasil prediksi model tetap valid dan tidak terpengaruh oleh perbedaan format atau skala data. Model kemudian diuji menggunakan data tersebut untuk menilai sejauh mana kemampuannya dalam mengklasifikasikan gambar yang belum pernah dilihat sebelumnya.

Evaluasi dilakukan dengan evaluasi performa model dilakukan melalui beberapa metode untuk mendapatkan gambaran menyeluruh mengenai akurasi dan keandalan model. Evaluasi ini bertujuan untuk menilai efektivitas model dalam mendeteksi kelas yang benar serta mengidentifikasi kelemahan dalam proses klasifikasi.

a. *Classification Report*

Classification report menyajikan ringkasan performa model dalam bentuk metrik evaluasi seperti *precision*, *recall*, *F1-score*, dan *accuracy*. Metrik-metrik ini digunakan untuk mengevaluasi ketepatan dan kecermatan model dalam memprediksi masing-masing kelas. Nilai-nilai tersebut membantu dalam menilai keseimbangan performa model terhadap berbagai aspek penting dari klasifikasi.

b. *Confusion Matrix*

Confusion matrix menampilkan hasil prediksi model dalam bentuk matriks yang memperlihatkan perbandingan antara label sebenarnya dan label prediksi. Matriks ini memudahkan dalam melihat jumlah prediksi benar untuk setiap kelas serta jenis kesalahan klasifikasi yang terjadi, seperti kesalahan positif atau negatif palsu.

c. Visualisasi Prediksi

Untuk memberikan gambaran nyata tentang kinerja model, dilakukan visualisasi terhadap beberapa data uji. Gambar-gambar ini ditampilkan bersama dengan label asli dan hasil prediksi dari model. Visualisasi ini berguna untuk mengevaluasi hasil klasifikasi secara langsung dan mengamati sejauh mana model mampu membedakan antara kelas yang tersedia.

3.9 Pengujian Pada Data Eskternal

Selain menggunakan dataset utama, model juga diuji dengan data atau gambar eksternal yang mewakili berbagai kondisi kulit, pencahayaan, dan latar belakang yang berbeda guna menguji kemampuan generalisasi model terhadap data dunia nyata. Setiap gambar eksternal diuji secara individual menggunakan model terbaik yang telah diperoleh dari proses pelatihan. Sebelum dimasukkan ke dalam model, gambar terlebih dahulu diubah ukurannya menjadi 224x224 piksel dan dinormalisasi agar sesuai dengan format input saat pelatihan. Model kemudian menghasilkan prediksi dalam bentuk probabilitas kelas, dan hasil akhir ditampilkan dalam bentuk gambar yang dilengkapi label klasifikasi. Jika gambar diklasifikasikan sebagai cacar monyet, label akan ditampilkan berwarna merah, sedangkan jika diklasifikasikan sebagai bukan cacar monyet, label akan ditampilkan berwarna hijau. Visualisasi ini bertujuan untuk memberikan interpretasi yang lebih jelas terhadap hasil klasifikasi dan memudahkan dalam menilai keakuratan model secara langsung.