BAB 3

METODOLOGI PENELITIAN

3.1 Sistematika Penelitian

Penelitian "Klasifikasi Covid-19 Berdasarkan Foto Rontgen Dada Menggunakan EfficientNet" menggunakan beberapa tahap. Tahap-tahap yang dilaksanakan antara lain adalah sebagai berikut.

1. Studi Literatur

Tahapan ini dilakukan dengan tujuan untuk mendukung proses penelitian. Dengan adanya teori dasar dapat mendukung penelitian ini. Teori yang digunakan antara lain CNN, pneumonia, foto rontgen, EfficientNet, Covid-19, anatomi paru, *cross validation*, *confusion matrix* dan *dataset* yang digunakan. Referensi dalam mencari teori di atas didapat dari buku, jurnal ilmiah, dan lain sebagainya.

2. Pengumpulan Data

Tahapan pengumpulan data merupakan tahap yang dilakukan untuk mencari dataset yang didapatkan dari berbagai sumber yang tersedia. Dataset dalam penelitian ini berjumlah 3.968 data dengan 3 kelas yaitu foto rontgen dada normal, pneumonia, dan Covid-19. Data yang dikumpulkan dalam penelitian ini berasal dari penelitian yang dilakukan oleh Chowdhury, dkk (2020).

3. Perancangan dan Pembuatan Sistem

Perancangan dan pembuatan sistem dalam penelitian ini dimulai dengan melakukan import *library* yang dibutuhkan, kemudian dilakukan tahap *preprocessing.* Tahap *preprocessing* yang dilakukan yaitu resize image, dimana d*ataset* yang telah didapat dilakukan proses penyeragaman ukuran spasial ke

dimensi 224x224 dengan menggunakan library OpenCv, kemudian *dataset* akan diubah ke dalam bentuk array. Tahap selanjutnya adalah pembagian *dataset* yang telah di *preprocessing.* Data akan dibagi dengan rasio 80:20 dimana 80% data akan menjadi data latih dan sisanya akan menjadi data uji.

Proses *training* dan *testing data* dilakukan. Selanjutnya yaitu melakukan klasifikasi pneumonia Covid-19 pada foto rontgen dada yang telah diberi label dan menghitung akurasi sistem tersebut dengan *confusion matrix*. *Dataset* yang digunakan dalam penelitian ini berjumlah 3.968 data dengan pembagian kelas; 1.200 gambar untuk kelas Covid-19, 1.345 pneumonia, dan 1.423 gambar untuk kelas normal. Perancangan dan pembuatan sistem dalam penelitian ini menggunakan bahasa pemrograman python melalui Jupyter Notebook dan Google Colaboratory.

4. Uji Coba dan Evaluasi

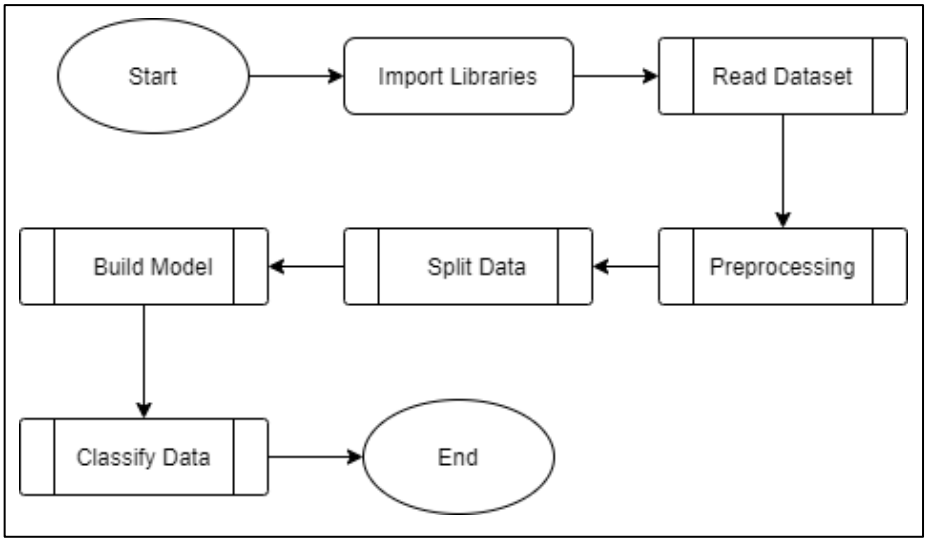
Pada tahapan ini, model yang telah dibangun pada proses sebelumnya akan dilakukan uji coba dan pengukuran akurasi yang didapat dengan data yang telah disiapkan sebelumnya. Pengujian ini dilakukan dengan tujuan memastikan model yang sudah dibangun memiliki kinerja yang baik dan menghasilkan hasil yang akurat.

5. Penulisan Laporan

Dalam tahapan ini, seluruh proses yang telah dilakukan dan hasil yang telah didapat akan didokumentasikan dalam sebuah laporan. Laporan penelitian ini, dibuat dengan tujuan sebagai bukti bahwa penelitian telah dilakukan.

3.2 Perancangan Sistem

Perancangan sistem dalam penelitian ini akan dilakukan dalam beberapa tahap. Tahap tersebut digambarkan dalam *flowchart* pada Gambar 3.1.



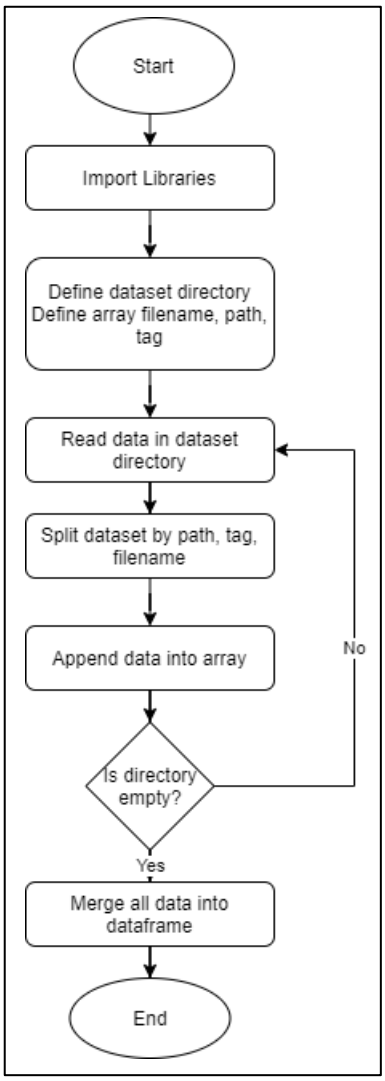
Gambar 3.1 *Flowchart* Perancangan Sistem

Sebelum melakukan klasifikasi, *library* yang dibutuhkan akan di-*import* terlebih dahulu. *Library* tersebut berguna untuk mendukung proses penelitian. Penjelasan lebih lanjut mengenai proses yang terdapat pada Gambar 3.1 dapat dilihat pada subbab 3.2.1 sampai dengan subbab 3.2.5.

3.2.1 Read Dataset

Dalam tahap ini yaitu memasukkan *dataset* yang telah dikumpulkan. *Dataset* yang digunakan berasal dari penelitian yang dilakukan oleh Chowdhury, dkk (2020). Jumlah sampel dari *dataset* yang digunakan sebanyak 3.968 dengan format PNG dan terbagi menjadi 3 kelas yaitu Covid-19, pneumonia, dan normal. Jumlah data pada kelas Covid-19 berjumlah 1.200 data, kelas pneumonia berjumlah 1.345 data, dan kelas normal dengan jumlah data 1.423. Berdasarkan Gambar 3.2 pengambilan *dataset* dilakukan dengan memuat semua gambar pada masing-

masing folder kelas dan memisahkan data ke dalam beberapa variabel yang telah ditentukan. *Dataset* dipisahkan berdasarkan *path*, nama file, dan label kelas. Data yang telah dipisahkan tersebut digabungkan dan dimasukkan ke dalam *dataframe*.

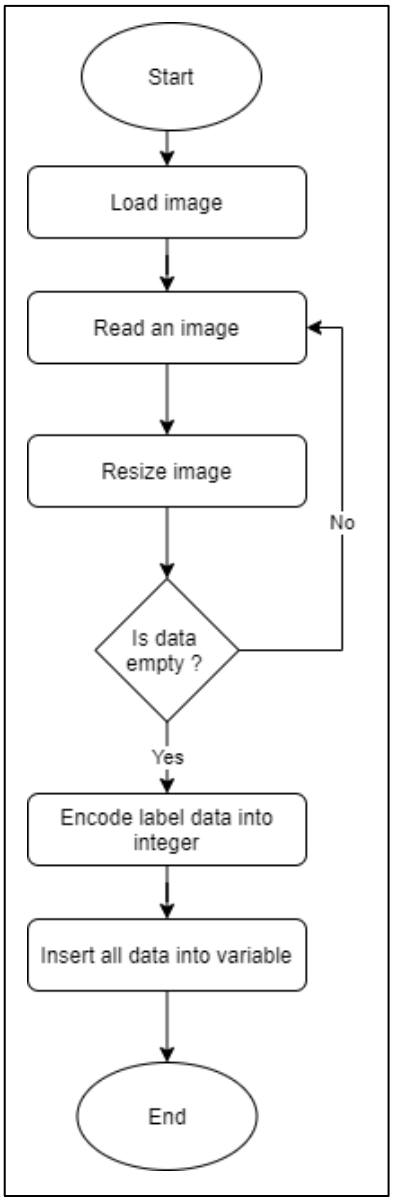


Gambar 3.2 *Flowchart* Read Dataset

3.2.2 Preprocessing

Setiap data pada *dataframe* yang telah dibuat pada proses *read dataset* akan melalui tahap *preprocessing.* Tahap *preprocessing* dilakukan dengan cara mengubah ukuran gambar dataset atau melakukan *resize image*. Data gambar yang terdapat dalam *dataframe* akan dilakukan penyeragaman ukuran spasial menjadi

224x224 dengan memanfaatkan *library* OpenCv. Penyeragaman gambar dilakukan untuk menyesuaikan *input shape* dari arsitektur EfficientNet dan diharapkan mendapat akurasi serta performa yang baik. Pada tahap ini dilakukan *looping* terhadap atribut path pada *dataframe*. Kemudian setiap gambar pada setiap kelas dilakukan penyeragaman ukuran. Data label kelas akan diubah menjadi data numerik dengan menggunakan LabelEncoder yang merupakan bagian dari *library* Sklearn. Hasil penyeragaman ukuran gambar dan data label kelas diubah menjadi array dan disimpan ke dalam sebuah variabel. *Flowchart preprocessing* dapat dilihat pada Gambar 3.3.

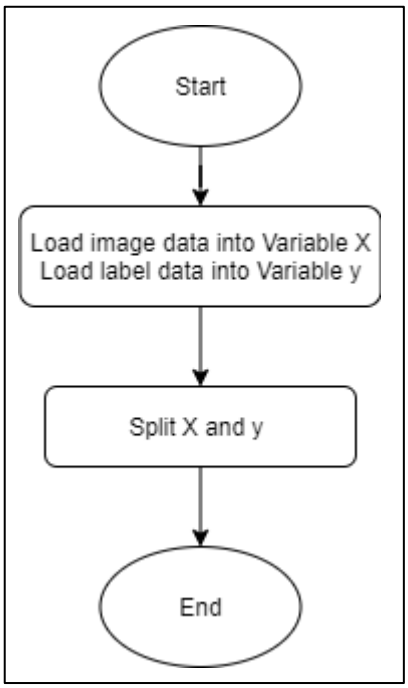


Gambar 3.3 *Flowchart Preprocessing EfficientNet*

3.2.3 Split Data

Setelah tahap *preprocessing* dilakukan, selanjutnya adalah pembagian data ke dalam *train set* dan *test set* dengan rasio yang telah ditentukan sesuai dengan skenario yang dilakukan. Hasil yang telah didapat dari tahap *preprocessing* akan dimasukkan ke dalam variabel. Variabel X merepresentasikan data gambar, sedangkan variabel y merepresentasikan data label. Pembagian data dilakukan dengan menggunakan *library* Sklearn Train Test Split. Dataset dibagi menjadi dua

set, yaitu *train set* dan *test set. Train set* merupakan data yang digunakan untuk melatih model yang telah dibuat. *Test set* merupakan data yang digunakan untuk melakukan evaluasi model dan pengujian model yang sudah dilatih pada *train set.*



Gambar 3.4 *Flowchart* Split Data

3.2.4 Build Model

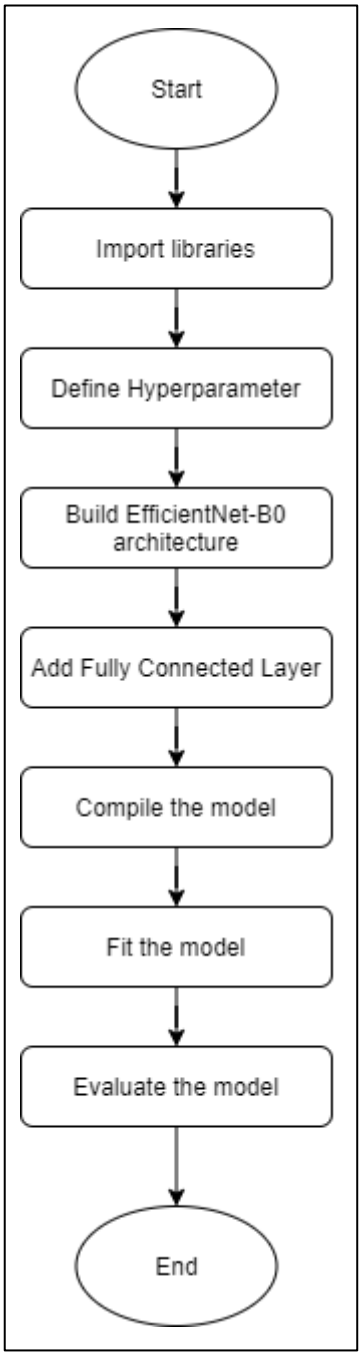
Dalam tahap ini dilakukan dua jenis pembuatan model, yaitu EfficientNet *pretrained* model dan AutoKeras.

A. EfficientNet *Pretrained* Model

Pada pembuatan model ini, tahap pertama yang dilakukan adalah *import library* yang dibutuhkan. Selanjutnya tetapkan *hyperparameter* yang akan digunakan. Kemudian data yang telah diproses pada tahap *preprocessing* dan data yang telah dibagi pada tahap *split data* akan menjadi masukan untuk model ini. Secara garis besar model yang dibuat hampir sama dengan CNN, yaitu proses *features extraction* dan *fully connected layer*. Data gambar yang terdapat dalam

data frame akan dilakukan proses ekstraksi fitur dengan menggunakan arsitektur EfficientNet. Model arsitektur EfficientNet pada penelitian ini merupakan *pretrained model*, dimana *pretrained model* ini telah dilatih dengan menggunakan data dari *repository* ImageNet. Model arsitektur EfficientNet dibangun dengan mengatur parameter weight=”imagenet” dan include_top=False. “imagetnet” digunakan karena merupakan standar untuk klasifikasi gambar. GlobalAveragePooling2D juga ditambahkan dalam model arsitektur EfficientNet. Pada penelitian ini juga dilakukan *transfer learning* dengan menggunakan ImageNet karena penggunaan *transfer learning* cenderung menghasilkan performa yang baik.

Selanjutnya adalah menambahkan *layer* dengan fungsi aktivasi yang berperan sebagai *fully connected layer* serta *dropout*. Penggunaan *dropout* dimaksudkan untuk mencegah *overfitting* pada model. Model yang telah dibuat akan di-*compile* dengan *Adam Optimizer*. Hasil dari meng-*compile* model akan dimasukkan ke dalam beberapa variabel yang sudah ditentukan. Proses selanjutnya adalah *training* model, model akan menerima masukan berupa *train set* yang digunakan sebagai data latih untuk model, *epochs* yang menentukan berapa kali model akan bekerja untuk seluruh data latih, *batch size* yang merupakan jumlah data yang harus dikerjakan dalam satu *epochs*, dan *validation data* yang digunakan untuk menentukan data validasi. Hasil keluaran dari *training* model berupa nilai *loss, acc, precision, dan recall.* Hasil tersebut akan dievaluasi untuk mencari nilai terbaik pada model.



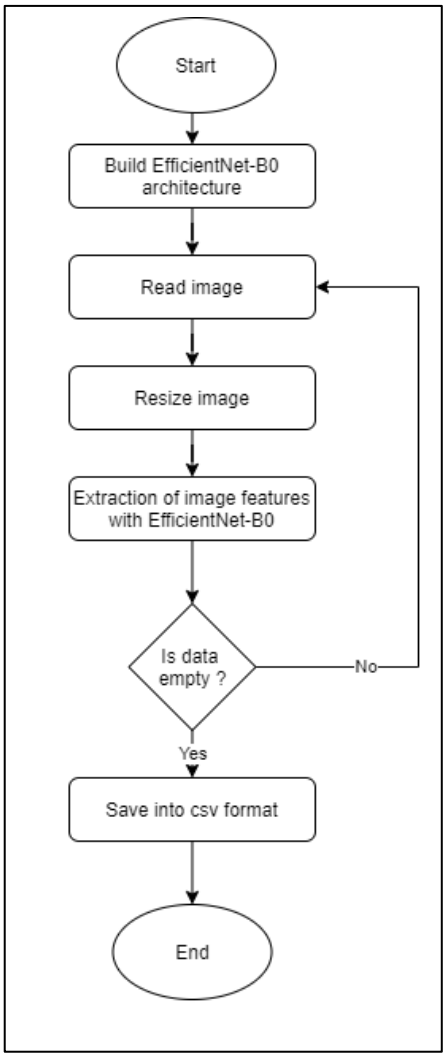
Gambar 3.5 *Flowchart* Build Model EfficientNet

B. AutoKeras

Tahap pertama yang dilakukan adalah *import library* yang dibutuhkan termasuk *library* AutoKeras. Selanjutnya dilakukan pencarian model dan *hyperparameter* terbaik. Dataset yang telah dikumpulkan dilakukan proses

ekstraksi fitur dengan menggunakan arsitektur EfficientNet-B0. Proses awal yaitu melakukan looping terhadap atribut path pada data frame, selanjutnya setiap gambar pada setiap kelas diubah menjadi 224x224. Kemudian melakukan ekstraksi fitur dari setiap gambar.

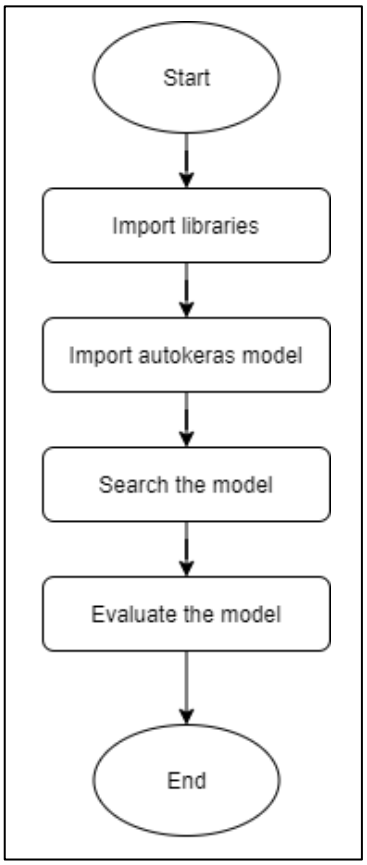
Hasil dari ekstraksi fitur berupa array dua dimensi, hasil tersebut masih akan diproses dengan flatten. Flatten berfungsi untuk mengubah hasil ekstraksi fitur menjadi array satu dimensi. Hasil akhir dari ekstraksi fitur akan disimpan ke dalam sebuah data frame. Sedangkan untuk data label kelas pada data frame akan dilakukan proses looping dimana label kelas diubah menjadi data numerik. Label dengan kelas COVID, Viral Pneumonia, dan Normal direpresentasikan dengan angka 0, 1, dan 2. Selanjutnya hasil tersebut akan disimpan ke dalam sebuah data frame. Dilakukan penggabungan data frame ekstraksi fitur dan label ke dalam sebuah data frame besar dan disimpan dengan format csv. Proses ekstraksi fitur ini dapat dilihat pada Gambar 3.6.



Gambar 3.6 *Flowchart* Ekstraksi Fitur EfficientNet-B0

Gambar 3.7 menjabarkan proses pembuatan model dengan AutoKeras. Data latih yang telah disiapkan menjadi masukan untuk mencari model terbaik. Ketika pencarian model terbaik dilakukan, API akan menerima panggilan tersebut dan melakukan *preprocessing* terhadap dataset yang diterima. Kemudian meneruskannya ke *searcher* untuk memulai pencarian model, *searcher* merupakan modul yang berisi Bayesian Optimizer dan Gaussian Process. Bayesian Optimizer di dalam *searcher* akan menghasilkan arsitektur baru, kemudian proses dilanjutkan dengan memanggil *graph modul* untuk melakukan komputasi grafik. Arsitektur tersebut disalin ke dalam *model trainer*, *model trainer* berguna untuk melatih model

dengan data latih. Model yang telah dilatih, disimpan ke dalam *model storage*. Performa dari model yang telah dilatih akan dikirim kembali ke *searcher* untuk melakukan *update* dengan menggunakan Gaussian Process. Hasil dari model terbaik disimpan ke dalam sebuah variabel untuk dilakukan evaluasi model. Model yang telah dibangun di-*export* untuk digunakan pada proses selanjutnya.

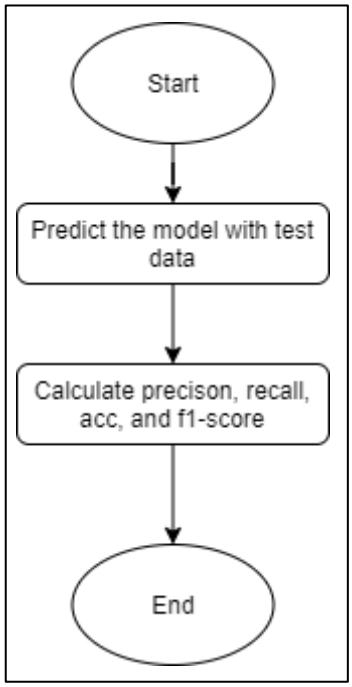


Gambar 3.7 *Flowchart* Build Model AutoKeras

3.2.5 Classify Data

Dalam tahap ini dilakukan evaluasi performa dengan menggunakan *confusion matrix* dengan memanfaatkan *library* Sklearn. Gambar 3.8 merupakan proses dalam pembagian data dimana model akan menerima masukan *test set* dan memperkirakan kemungkinan hasil. Hasil dari prediksi dan data label akan menjadi

masukan untuk *confusion matrix*. Kemudian dari *confusion matrix* dilakukan perhitungan akurasi, f1-score, *precision*, dan *recall*.



Gambar 3.8 *Flowchart* Classify Data