BAB III
METODE PENELITIAN

3.1 Metode Pengumpulan Data

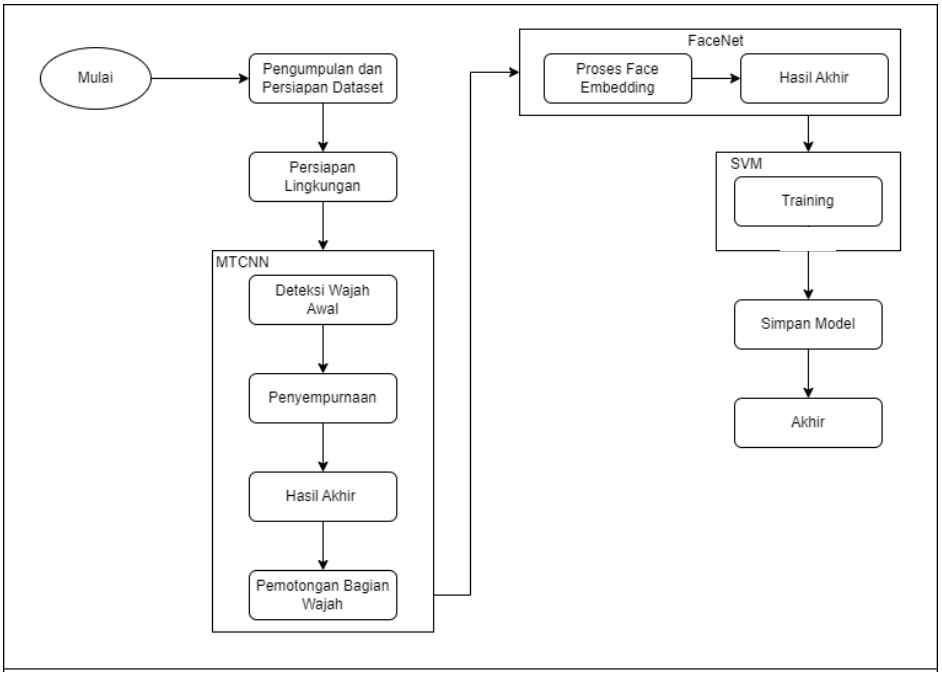
Untuk membuat suatu model yang dapat memprediksi wajah orang, diperlakukan suatu *dataset*, maka dari itu dilakukan pengumpulan data yang bersifat *self-build dataset*. *Self-build dataset* sendiri adalah *dataset* yang peneliti kumpulkan atau dapatkan dengan mengambil foto pribadi / manual tanpa mengambil *dataset* tambahan dari sumber *external* kecuali *dataset* bawaan model. Beberapa cara yang bisa dilakukan untuk mengumpulkan data yang berisi gambar dari wajah-wajah orang menjadi suatu *dataset* :

1. Memasukkan data gambar wajah orang kedalam suatu folder, dan memberi nama folder tersebut sesuai dengan nama gambar wajah. Hal ini dilakukan sebelum dimulainya aplikasi. Kegiatan memberi nama folder sesuai dengan nama pemilik wajah juga biasa disebut sebagai *labeling*, yaitu memberi tau model bahwa seluruh gambar yang ada di folder tersebut merupakan gambar dari wajah orang tertentu.
2. Mengambil gambar langsung dari aplikasi. Hal ini dilakukan setelah dimulainya aplikasi. Pada aplikasi nantinya akan memiliki kemampuan untuk mengambil *screenshot* pengambil kuis dan membuat folder otomatis sesuai dengan nama yang diinputkan. Hal ini meningkatkan kemudahan dalam hal membangun *dataset*.

Untuk setiap folder memiliki minimal 15 gambar yang berformatkan .jpeg, .png, .jpg. Semakin banyak gambar maka akan semakin akurat model dalam mengenali wajah.

3.2 Perancangan Sistem *Face Recognition*

Berikut merupakan rancangan sistem dalam bentuk *FlowChart*, lebih lengkapnya dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Flowchart Sistem Face Recognition

3.3 Persiapan Lingkungan

Pembuatan model dilakukan dengan menggunakan bantuan code editor yaitu *Visual Studio Code* v.1.86.1 dan *Sublime Text* 4. Adapun agar *code editor* kita dapat melakukan *image processing* dengan MTCNN, *Facenet*, dan SVM, maka diperlukan meng-*install* beberapa *library* yang dibutuhkan, khususnya *library* yang bisa digunakan dengan bahasa pemrograman *Python*. *Library* dalam *Python* sendiri merupakan kumpulan kode yang telah dioptimalkan, sehingga membantu analyst dalam menghindari penulisan rutin dan fokus pada penciptaan solusi yang lebih inovatif.

Berikut beberapa *library* yang digunakan penulis serta penjelasannya secara singkat:

1. OS
2. *pickle*
3. PIL
4. *numpy*
5. *Pytorch*
6. MTCNN
7. *FaceNet*
8. *scikit-learn*

3.3.1 OS

OS merupakan salah satu modul bawaan yang menyediakan fungsi-fungsi untuk berinteraksi dengan sistem operasi. Modul ini memungkinkan untuk

melakukan berbagai operasi terkait sistem operasi, seperti manipulasi file dan direktori, mengakses variabel lingkungan, dan lainnya.

3.3.2 *Pickle*

Pickle merupakan modul yang digunakan untuk melakukan serialisasi dan deserialisasi objek *Python*. Serialisasi adalah proses mengubah objek *Python* menjadi format *byte*, sedangkan deserialisasi adalah proses mengembalikan objek *Python* dari format *byte* tersebut. Singkatnya modul *Pickle* dalam penelitian ini akan digunakan untuk menyimpan hasil akhir berupa *file classifier* yang dapat mengklasifikasikan suatu gambar wajah.

3.3.3 PIL

PIL atau Python Imaging Library merupakan modul yang menyediakan fungsionalitas untuk memanipulasi berbagai jenis gambar, termasuk membuka, menyimpan, dan memanipulasi gambar dalam berbagai format.

3.3.4 *Numpy*

Numpy atau *Numerical Python* merupakan salah satu *library Python* yang menyediakan dukungan untuk *array* dan operasi *array* berbasis matriks, serta berbagai fungsi matematika yang efisien untuk bekerja dengan data numerik.

3.3.5 Pytorch

Pytorch merupakan salah satu library selain *Tensorflow* yang berfungsi untuk *machine learning* dan *deep learning*. *Pytorch* sendiri dikembangkan oleh *Facebook* sedangkan *Tensorflow* dikembangkan oleh *Google*.

3.3.6 MTCNN

MTCNN atau *Multi-Task Cascaded Convolutional Networks* merupakan sebuah *library Python* yang digunakan untuk mendeteksi wajah dalam gambar. MTCNN dirancang untuk mendeteksi wajah manusia dalam gambar dan mendapatkan kotak pembatas (*bounding box*) untuk setiap wajah yang terdeteksi.

3.3.7 FaceNet

FaceNet adalah model deep learning yang dirancang untuk mengekstrak representasi vektor wajah yang seragam dari gambar wajah, sehingga memungkinkan pengenalan wajah yang akurat.

3.3.8 scikit-learn

Scikit-learn merupakan *library* yang menyediakan berbagai algoritma *machine learning* yang mencakup klasifikasi, regresi, klastering, reduksi dimensi, seleksi fitur, dan lainnya. Ini mencakup algoritma populer seperti *Support Vector Machines* (SVM), *Decision Trees*, *Random Forests*, *k-Means*, dan yang lainnya.

Dalam penelitian ini, penulis menggunakan SVC atau *Support Vector Classification* yang digunakan untuk melatih model *Support Vector Machine* (SVM) untuk tugas klasifikasi.

3.4 *Face Detection* menggunakan MTCNN

MTCNN menggunakan proses bertahap untuk memperbaiki deteksi wajah dari tingkat kasar hingga tingkat yang sangat akurat. Terdapat 3 proses utama dalam proses deteksi wajah sampai menghasilkan hasil akhir, proses tersebut mencakup:

1. Deteksi Kasar

   Pada tahap ini MTCNN menerima input awalan berupa gambar asli dan mengidentifikasi area yang kemungkinan terdapat wajahnya. Hasilnya berupa prediksi area yang kemungkinan terdapat wajahnya, dan juga skor yang mengindikasikan seberapa yakin algoritma yakin bahwa area tersebut mengandung wajah.

2. Penyempurnaan

   Pada tahap ini dilakukan perbaikan pada tahap pertama menggunakan CNN yang lebih kompleks. Tahap ini juga berfungsi untuk meningkatkan keakuratan akurasi dalam mendeteksi area yang kemungkinan ada wajahnya.

3. Pemilihan

   Pada tahap ini menerima inputan dari tahap 2, menggunakan proses yang sama seperti yang dilakukan pada tahap 2 yaitu menggunakan CNN tetapi bedanya adalah CNN yang digunakan pada tahap ini memiliki lebih banyak lapisan guna meningkatkan akurasi hasil akhir. Hasil akhirnya berupa lokasi wajah, probabilitas lokasi wajah, dan beberapa titik kunci wajah (mata, hidung, mulut).

3.5 Pemotongan Bagian Wajah

Setelah melakukan deteksi wajah menggunakan MTCNN dilakukan pemotongan khusus pada bagian wajah, pada proses ini juga dilakukan *label-extracting* untuk mengetahui berapa banyak label yang ada dan akan dipakai untuk *training* model SVM.

Jadi hasil akhir sebelum proses *Face Recognition* dengan *FaceNet* berupa gambar yang sudah dipotong dan *label*.

3.6 Face Recognition menggunakan FaceNet (InceptionResnetV1)

InceptionResnetV1 merupakan salah satu arsitektur dalam FaceNet yang menggabungkan konsep modul *Inception* dan blok residu, yang performanya jauh lebih baik dari *FaceNet* biasa tetapi membutuhkan daya komputasi yang lebih besar.

Arsitektur *InceptionResNetV1* sudah dilatih dengan *dataset VGGFace2*, yang merupakan *dataset* besar berisi wajah-wajah dari berbagai identita, tetapi penulis akan melakukan *fine-tuning*. *Fine-tuning* sendiri adalah proses pelatihan ulang

model pada *dataset* yang lebih kecil atau khusus untuk tujuan tertentu setelah awalnya dilatih pada *dataset* besar.

Menggunakan *dataset* yang berisi gambar yang sudah dipotong, akan dilakukan juga preprocessing-image seperti mengubah nilai gambar menjadi Pytorch tensor, menormalisasi nilai gambar, menambah dimensi, setelah itu akan diproses oleh InceptionResnetV1 menghasilkan hasil akhir berupa *face embedding*. *Face embedding* adalah representasi numerik atau vektor yang mewakili fitur-fitur unik dari wajah seseorang dalam bentuk angka-angka.
Pada proses *training* SVM, nilai dari *face embedding* inilah yang akan digunakan sebagai dasar pengklasifikasian bersama *label*.

3.7 Face Classification menggunakan SVM

Training model dilakukan menggunakan bantuan *Support Vector Machine* (SVM). Untuk pengklasifikasian digunakan SVC atau *Support Vector Classification.* SVC sendiri merupakan implementasi dari SVM yang dikhususkan untuk tugas klasifikasi. Dalam SVM terdapat 4 macam kernel yaitu: *Linear, Sigmoid*, RBF, *Polynomial*. Untuk penelitian ini, penulis lebih memilih menggunakan kernel Linear dikarenakan nilai akurasi yang cenderung stabil dan tinggi [22]. Setelah proses *training* selesai maka dilakukan simpan model agar model bisa digunakan untuk proses selanjutnya.

3.8 Perancangan Sistem Database

Berikut merupakan alur / rancangan agar sistem database bisa menampilkan data dari aplikasi, lebih jelasnya dapat dilihat pada Gambar 3.2.

Gambar 3.2 Diagram *Flowchart* Alur Database

Penulis memakai *MySQL* sebagai DBMS untuk menyimpan data yang dikirim dari aplikasi dan *MySQL Workbench* sebagai aplikasi untuk mengedit data yang terdapat dalam *database*. Didalam sistem *database*, user bisa melihat semua data pengambil kuis online, mulai dari, nama mahasiswa, waktu mulai, total *frames*, ada muka, tidak ada muka/berbeda, jumlah *warning*, waktu selesai, durasi, dan *level*.