BAB 2
LANDASAN TEORI

2.1 *Face Recognition*

Face Recognition adalah teknologi yang dapat mengenali wajah seseorang untuk berbagai keperluan. Wajah kemudian dianalisis secara detail menggunakan metode teknologi pencitraan 2D atau 3D. Beberapa informasi yang diambil adalah informasi tentang bentuk wajah, mata, hidung, dan lain sebagainya. Lalu, berbagai data wajah diubah menjadi data atau informasi digital, dimana nantinya akan dicocokkan data wajah lainnya dalam database atau sebuah sistem. Untuk menentukan hasil akhir, sistem kemudian menggunakan algoritma yang membandingkan data wajah yang baru saja diambil dengan data wajah yang sudah tersimpan di dalam sistem [7].

2.2 Face-api

Face-api.js adalah sebuah library JavaScript untuk face recognition yang dapat digunakan untuk mengenali dan memanipulasi wajah pada gambar maupun video. Library ini dikembangkan oleh Vincent Mühler dan diluncurkan pada tahun 2018 [8]. Lalu dikembangkan oleh tim pengembang open-source yang terdiri dari beberapa kontributor di GitHub, dan tersedia secara gratis untuk digunakan.

Kegunaan dari face-api.js adalah untuk membantu pengembang dalam membangun aplikasi yang memerlukan pengenalan wajah secara otomatis, seperti sistem presensi mahasiswa atau pengenalan wajah untuk keperluan keamanan. Library ini memiliki fitur yang lengkap, seperti deteksi wajah, pengenalan emosi, identifikasi wajah, dan pengenalan landmark [8].

Berikut akan dijelaskan masing masing model yang sudah disediakan oleh Face-api.js

1. *Face Detection Models*
   Model *face detection* digunakan untuk mendeteksi wajah dalam gambar atau video. Model ini mengidentifikasi posisi, ukuran, dan bentuk wajah yang ada dalam gambar atau frame video. Dengan menggunakan model ini, dapat dilakukan pengenalan keberadaan wajah dan mengambil tindakan lebih lanjut berdasarkan informasi tersebut. Model yang disediakan ada sebagai berikut.

(a) *SSD Mobilenet V1*

(b) *Tiny Face Detector*

(c) MTCNN *(Multi-task Cascaded Convolutional Neural Networks)*

2. *Face Landmark Models*

   Model *face landmark* digunakan untuk menentukan titik-titik penting pada wajah, seperti mata, hidung, dan mulut. Model ini membantu dalam memahami struktur dan posisi fitur wajah yang lebih rinci. Dengan menggunakan model ini, dapat melakukan tugas seperti estimasi deteksi mata tertutup, atau analisis bentuk wajah.

   Untuk model yang dapat digunakan adalah face_landmark_68_model, model ini sudah dilakukan pelatihan kurang lebih 35 ribu gambar yang sudah dilakukan pemberian label dengan titik penting pada wajah [8].

3. *Face Recognition Model*

   Model *face recognition* digunakan untuk mengenali dan membandingkan wajah berdasarkan fitur-fitur yang unik. Model ini memungkinkan identifikasi individu berdasarkan gambar wajah yang diberikan. Dengan menggunakan model ini, dapat membangun aplikasi pengenalan wajah, sistem kehadiran, atau sistem keamanan yang berbasis pengenalan wajah. Untuk model yang dapat digunakan adalah faceRecognitionNet, model ini sudah dilakukan pelatihan menggunakan dataset ”300 faces In-the-wild” [9] dan berhasil menghasilkan akurasi sebesar 99.38%[8].

2.3 *Convolutional Neural Network*(CNN)

CNN merupakan jenis model deep learning yang digunakan untuk memproses data dengan pola grid, seperti gambar. Struktur model ini terinspirasi oleh organisasi korteks visual pada hewan[10, 11].

Cara kerja CNN adalah dengan menggunakan proses konvolusi dengan menerapkan kernel (filter) konvolusi dengan ukuran tertentu pada gambar, dan komputer memperoleh informasi representasi baru dengan mengalikan bagian gambar dengan filter yang digunakan [12]. Berikut akan dijelaskan untuk tahapan pada CNN

1. *Convolutional Layer*

   CNN menggunakan lapisan konvolusi yang menerapkan operasi konvolusi

pada input citra. Pada setiap lapisan konvolusi, beberapa filter (juga dikenal sebagai kernel) diterapkan pada input untuk mengekstraksi fitur-fitur visual yang relevan.

2. *Activation Layer*
   Setelah operasi konvolusi, fungsi aktivasi seperti ReLU (*Rectified Linear Unit*) diterapkan untuk memperkenalkan non-linearitas ke dalam jaringan. Ini membantu dalam memodelkan hubungan yang kompleks antara fitur input.

3. *Pooling Layer*
   Lapisan *pooling* digunakan untuk mengurangi dimensi spasial dari fitur yang dihasilkan oleh lapisan konvolusi. Ini membantu dalam mengurangi dimensi secara terus menerus serta mengurangi jumlah parameter dan mengurangi kompleksitas komputasi.

4. *Fully Connected Layer*
   Setelah serangkaian lapisan konvolusi dan penggabungan, output yang dihasilkan dilewatkan ke lapisan yang sepenuhnya terhubung. Lapisan ini berfungsi sebagai klasifikasi akhir atau lapisan output, yang menghubungkan fitur-fitur input dengan kelas atau label yang relevan.

2.4 Odoo

Odoo adalah perangkat lunak manajemen lengkap yang menyediakan rangkaian modul bisnis yang membentuk rangkaian lengkap aplikasi manajemen bisnis yang ditujukan untuk bisnis dari semua ukuran [13]. Odoo juga menyediakan fitur-fitur seperti manajemen inventaris, akunting, penjualan, manajemen proyek, manajemen sumber daya manusia, dan lain-lain [14].

Modul pada Odoo merupakan komponen-komponen fungsional yang dapat ditambahkan atau dikembangkan untuk memperluas fungsionalitas Odoo. Setiap modul mewakili aspek bisnis tertentu dan terdiri dari berbagai komponen seperti model, tampilan, logika bisnis, pembuatan laporan, dan juga pembuatan suatu aksi. Modul ini dapat diinstal, diaktifkan, dan dikonfigurasi sesuai kebutuhan bisnis.

Struktur modul Odoo terdiri dari beberapa folder yang memiliki peran masing-masing, umumnya satu modul berisikan folder *data,demo,models,security,views*, *__init__.py*, dan *__manifest__.py* [15]. Berikut akan dijelaskan masing-masing peran dan folder lainnya.

1. *models*

   Merupakan tempat untuk menyimpan definisi model atau kelas yang merepresentasikan entitas dalam sistem Odoo.

2. *views*

   Berisi file-file XML yang digunakan untuk mengatur tampilan pengguna atau antarmuka modul Odoo. File-file ini mendefinisikan tampilan form, tampilan daftar (*list*), tampilan kanban, dan tampilan lainnya yang digunakan dalam modul.

3. *data*

   Digunakan untuk menyimpan *file* data yang akan dimuat ke dalam sistem Odoo saat instalasi atau pembaruan modul. *File* ini dapat berupa data statis seperti konfigurasi awal, *template*, gambar, atau *file* CSV untuk memuat data ke dalam tabel Odoo.

4. *static*

   Tempat untuk menyimpan *file* statis seperti file CSS, JavaScript, gambar, dan aset lainnya yang diperlukan oleh modul. *File* dalam *folder static* dapat digunakan untuk memodifikasi tampilan pengguna, menambahkan logika klien, atau menyediakan elemen desain tambahan.

5. *security*

   Digunakan untuk mengatur keamanan dan izin akses pada modul Odoo. Di dalam folder ini, dapat digunakan *file* csv seperti *ir.model.access.csv* untuk mendefinisikan grup-grup pengguna, hak akses, dan aturan keamanan yang berlaku pada modul.

6. *wizard*

   Digunakan untuk menampilkan jendela *popup* atau *wizard* kustom, dimana didalamnya berisikan *__init__.py* untuk melakukan inisiasi pada *file python*, file .py yang digunakan untuk mendifiniskan *transientmodel*, dan juga *file* .xml yang digunakan sebagai bentuk dari tampilan jendela *popup* atau *wizard* kustom.

7. *report*

   Digunakan untuk menyimpan *file* dengan format .xml. Pada folder ini digunakan untuk mengumpulkan *report* yang sudah dilakukan modifikasi dari bentuk awalnya ataupun *report* yang baru saja di buat dari awal, untuk

konfigurasi bisa mulai dari *action button*, format kertas yang digunakan untuk hasil keluarannya, lalu juga bentuk atau isi dari laporan tersebut.

8. *__init__.py*

   Sebuah *file* yang ditempatkan di dalam direktori modul Odoo. *File* ini bertanggung jawab untuk membuat modul tersebut dapat diakses dan digunakan oleh Odoo. Pada dasarnya, *__init__.py* berfungsi sebagai tanda bahwa direktori tersebut adalah sebuah paket Python. Di dalam file ini, dapat melakukan beberapa inisialisasi atau impor yang diperlukan sebelum modul dimuat.

2.5 *Convusion Matrix*

Confusion Matrix adalah tabel yang menyatakan jumlah data uji yang diklasifikasikan dengan benar dan jumlah data tes yang salah klasifikasi [16].Dengan melakukan pengukuran kinerja pada masalah klasifikasi *machine learning* dimana hasil yang diberikan dapat mencakup dua kelas atau lebih.

Confusion Matrix adalah tabel dengan 4 kombinasi berbeda dari nilai prediksi dan nilai aktual. *Confusion Matrix* Ada empat istilah yang mewakili hasil proses klasifikasi yaitu *true positive, true negative, false positive* dan *false negative* [17]. Contoh confusion matrix untuk klasifikasi dapat dilihat pada gambar

		Kelas Prediksi	
		1	0
Kelas Sebenarnya	1	TP	FN
	0	FP	TN

Gambar 2.1. Contoh *Convusion Matrix*

Sumber: [17]

Keterangan:

(a) TP (*True Positive*) = Merupakan jumlah data positif yang secara benar diklasifikasikan sebagai positif. Dalam penelitian ini, TP akan mewakili jumlah wajah yang berhasil dikenali dengan benar oleh model sebagai wajah pengguna yang sedang login.

(b) FP (*False Positive*) = Merupakan jumlah data negatif yang keliru diklasifikasikan sebagai positif. Dalam penelitian ini, FP akan mewakili jumlah wajah yang salah diklasifikasikan sebagai wajah pengguna yang sedang login, padahal sebenarnya bukan.

(c) TN (*True Negative*): Merupakan jumlah data negatif yang secara benar diklasifikasikan sebagai negatif. Dalam penelitian ini, TN akan mewakili jumlah wajah yang berhasil dikenali dengan benar oleh model sebagai wajah bukan pengguna yang sedang *login*.

(d) FN (*False Negative*): Merupakan jumlah data positif yang keliru diklasifikasikan sebagai negatif. Dalam penelitian ini, FN akan mewakili jumlah wajah pengguna yang tidak berhasil dikenali oleh model sebagai wajah pengguna yang sedang login.

Lalu pada Confusion matrix sendiri ada 4 rumus yang akan digunakan untuk mendapatkan nilai *Accuracy, Precision, Recall* dan *F-1 Score* yang akan dijelaskan sebagai berikut.

2.5.1 *Accuracy*

Akurasi *Accuracy* adalah ukuran seberapa baik model klasifikasi berhasil mengklasifikasikan data dengan tepat. Akurasi dihitung dengan membagi jumlah prediksi yang benar (*True Positives* dan *True Negatives*) dengan jumlah data. Untuk perumusannya dapat dilihat pada persamaan 2.1.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.1}$$

2.5.2 *Precision*

Presisi (*Precision*) seberapa akurat prediksi positif yang dihasilkan oleh model. Presisi dihitung dengan membagi jumlah *True Positives* dengan jumlah keseluruhan prediksi positif (*True Positives* dan *False Positives*). Untuk perumusannya dapat dilihat pada persamaan 2.2.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

2.5.3 *Recall*

Recall (Sensitivitas) mengukur sejauh mana model dapat melakukan identifikasi pada semua kasus positif yang sebenarnya. *Recall* dihitung dengan membagi jumlah *True Positives* dengan jumlah keseluruhan kasus positif yang sebenarnya (*True Positives* dan *False Negatives*). Agar lebih jelas perumusannya dapat dilihat pada persamaan 2.3.

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

2.5.4 *F-1 Score*

Skor F-1 (*F-1 Score*) menggabungkan Precision dan Recall menjadi satu hasil yang menggambarkan kualitas model secara keseluruhan. Untuk perumusan lebih jelasnnya dapat dilihat pada persamaan 2.4.

$$F1 - Score = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.4}$$