BAB 2
TINJAUAN PUSTAKA

2.1. Pengenalan Optik

Pengenalan karakter optik atau biasa disebut sebagai *Optical Character Recognition* (OCR) merupakan metode untuk melakukan konversi secara elektronik dari citra teks cetakan ataupun tulisan tangan menjadi teks yang dikenali dalam pengolahan komputasi. Secara umum bagian pemrosesan pada OCR dibagi menjadi tiga bagian proses utama yaitu *pre-processing*, pengenalan (*recognition*), dan *post-processing* [6].

2.2. Citra

Murni menyatakan bahwa citra merupakan hsil dari suatu sistem pengambilan data yang bersifat optik berupa foto. Sifat citra berbentuk sinyal analog seperti tampilan gambar pada monitor televisi, atau sifatnya digital sehingga dapat disimpan pada pita magnetik [7]. Ian T. Young berpendapat bahwa citra digital *a[m,n]* adalah gambar dalam ruang dimensi dua berasal dari citra sinyal analog *a(x,y)* pada ruang dimensi dua yang telah melewati proses sampling atau dengan istilah sekarang adalah digitalisasi [8].

Maria berpendapat bahwa citra digital merupakan citra *f(x,y)* bentuknya telah terpisah pada beberapa titik dan kecerahan. Citra dapat dimodelkan dalam bentuk larik dua dimensi atau kumpulan larik dua dimensi. Setiap larik menghasilkan satu kanal warna. Tingkat keabuan dijelaskan sebagai tingat nilai kecerahan suatu citra. Setiap elemen larik citra tersebut dinamakan pel merupakan istilah '*picture element*', sekarang lebih dikenal dengan istilah piksel [9]. Umumnya proses pengolahan citra dapat didefinisikan sebagai gambar berdimensi dua secara digital.

Kebutuhan dalam mengolah citra umumnya dilakukan untuk membantu menyesuaikan kualitas suatu gambar agar akan lebih mudah diimplementasikan oleh manusia dan mudah diolah dengan komputasi sesuai kebutuhan. Implementasi dari pengolahan citra dapat berguna untuk mendapatkan informasi yang terkandung didalam pada suatu gambar. Proses pengolahan ini memilikidata masukan dan informasi keluaran yang berbentuk citra [10].

2.3. Jenis Citra

Jenis citra yang umum ditemukan meliputi tiga tipe citra. Ketiga tipe citra tersebut adalah jenis citra berwarna, citra berskala keabuan, dan citra biner [11].

2.3.1. Citra Berwarna

Citra jenis ini disebut berwarna karena biasanya menggunakan campuran warna merah (R), hijau (G), dan biru (B), atau biasa disingkat dengan RGB. Ketiga komponen warna memiliki tingkat nilai antara 0 sampai dengan 255. Jika nilai suatu warna R, G, atau B condong kearah dekat dengan nol, maka warna yang dihasilkan akan sedikit. Sebaliknya jika tingkat nilai warna R, G, atau B mendekati angka 255, maka warna yang dihasilkan akan semakin jelas. Setiap komponen warna menggunakan 8 bit sehingga warna yang mungkin dapat menghasilkan nilai 255 x 255 x 255 atau 16.581.375 warna [12].

Gambar 2.1 Citra Sertifikat Berwarna

2.3.2. Citra Berskala Keabuan

Berbeda dengan citra berwarna, citra dengan skala keabuan hanya memiliki gradasi warna hitam dan putih, warna yang disajikan pada citra akan terlihat

menjadi berwarna abu-abu. Skala keabuan dinyatakan menggunakan intensitas nilai yang berkisar diantara 0 sampai dengan 255 [12].

Gambar 2.2 Citra Sertifikat Berskala Keabuan

2.3.3. Citra Biner

Citra biner merupakan citra yang hanya memiliki nilai biner pada setiap piksel. Nilai biner merupakan bilangan yang hanya memiliki dua nilai domain berupa angka nol dan satu. Maka pada citra biner nilai setiap piksel hanya memungkinkan nilai nol atau satu. Nilai nol akan menyatakan warna piksel tersebut berwarna putih sedangkan nilai satu akan menyatakan warna putih atau bisa didefinisikan sebaliknya. Citra biner biasa digunakan dalam pemrosesan citra, umumnya digunakan untuk menemukan batasan tepi suatu objek [12].

Gambar 2.3 Matriks *Pixel* pada Citra Biner

2.4. Praproses

Tahap praproses atau biasa disebut *preprocessing* merupakan penjelasan tahap awal dalam alur kerja untuk menyiapkan citra sebagai data input yang siap diolah lebih lajut oleh sistem. Proses pada praproses yang digunakan pada penelitian ini sebagai berikut:

2.4.1. *Grayscale*

Grayscale adalah suatu proses perubahan nilai warna pada suatu citra menjadi nilai keabuan sehingga citra akan menjadi lebih sederhana. Rumus penyederhanaan pada tahap grayscale adalah sebagai berikut.

$$I = 0.2989 * R + 0.5870 * G + 0.1141 * B$$

Pada rumus tersebut R merupakan nilai warna merah (*Red*), G berarti nilai dari warna hijau (*Green*), dan B merupakan nilai warna biru (*Blue*) pada suatu pixel citra [11].

Gambar 2.4 Citra Hasil *Grayscale*

2.4.2. *Maximally Stable Extremal Regions*

Proses *Maximally Stable Extremal Regions* atau MSER akan melakukan pendeteksian wilayah-wilayah objek pada suatu citra dengan melakukan metode *blob* atau penggumpalan pada wilayah tertentu untuk mendapatkan objek perhatian yang diharapkan mendapatkan karakter yang dibutuhkan. Untuk dapat mendeteksi wilayah-wilayah yang diharapkan tersebut tahapan yang akan dilakukan adalah dengan melakukan transformasi variasi-variasi kawasan dengan cara

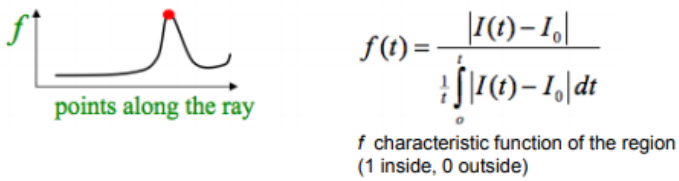
mempertimbangkan tingkat perubahan penerangan, penerjemahan nilai piksel citra, rotasi, skala, dan transformasi *affine*. Transformasi *affine* merupakan transformasi secara geometris yang mempertahankan garis dan paralelisme. Proses deteksi wilayah akan dilakukan berulang-ulang dan secara stabil, sehingga mampu membedakan antar wilayah.

Algoritma yang digunakan dalam menentukan intensitas *affine* invarian berbasis ekstrem (*Extrema-based*). Yang pertama adalah dimulai dari titik ekstrem intensitas lokal. Selanjutnya pergi ke segala arah sampai titik ekstrem dari suatu fungsi f(x). Kurva yang akan menghubungkan titik-titik adalah batasan wilayah. Berikutnya kalkulasikan momen geometris yang didapatkan pada batasan tersebut. Lalu ubah daerah dengan tanda elips [13].

Gambar 2.5 Deteksi Wilayah

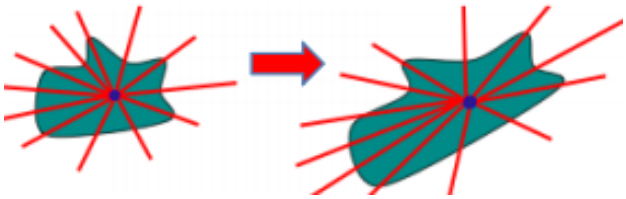
Untuk mendeteksi daerah ekstrem dapat dilakukan dengan langkah berikut

1. Mendeteksi titik *anchor* (jangkar), dapat menggunakan funsi detektor Harris untuk mendapatkan sudut-sudutnya. Titik jangkar yang terdeteksi pada berbagai skala adalah ekstrem intensitas lokal.
2. Menjelajahi gambar di sekitar setiap pencahayaan dari setiap titik jangkar. Telusuri setiap cahaya dimulai dari titik ini hingga mencapai suatu fungsi f(x).



Gambar 2.6 Fungsi Deteksi Titik Jangkar

3. Semua titik membuat beberapa wilayah berbentuk tidak teratur. Kira-kira daerah yang sesuai diperoleh untuk daerah yang mengalami transformasi *affine*.



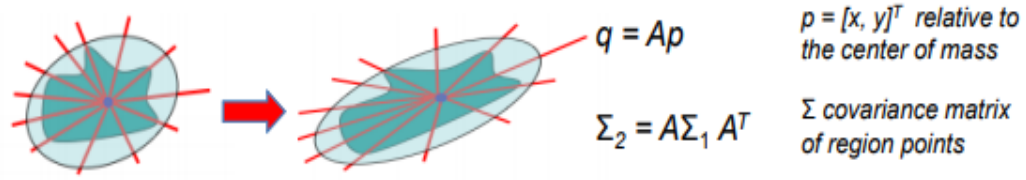
Gambar 2.7 Deteksi Suatu Daerah

Selanjutnya untuk lebih mendapatkan wilayah tersebut lebih tepat dapat melakukan tahapan berikut.

1. Dengan menggunakan fungsi sebelumnya, dicari momen kedua (integral) untuk kemungkinkan mendapatkan kawasan daerah yang sama pada satu elips.

$$m_{pq} = \int x^p y^q f(x,y)\, dxdy$$

2. Lingkaran dari hasil daerah transformasi *affine* sesuai dengan elips wilayah asli di bawah transformasi yang sama.



Gambar 2.8 Penyesuaian Elips Wilayah Asli

3. Penerangan berbasis elips ke lingkaran untuk invarian *affine* dijadikan sebagai daerah.

Algoritma MSER mengekstrak dari gambar daerah *co-varian*, yaitu sebuah MSER adalah komponen yang terhubung secara stabil dari beberapa set gambar tingkat abu-abu. MSER didasarkan pada gagasan untuk mengambil daerah yang tetap hampir sama melalui berbagai ambang batas sebagai berikut:

1. Semua piksel di bawah ambang yang diberikan berwarna putih dan semua yang di atas atau yang sama berwarna hitam.
2. Jika kita ditunjukkan urutan gambar yang sudah berada pada ambang (*thresholded images*) dengan bingkai t yang sesuai dengan ambang t, kita akan melihat gambar hitam pertama, kemudian bintik-bintik putih sesuai

dengan wilayah secara lokal. Intensitas minima akan muncul kemudian tumbuh lebih meluas.

3. Bintik-bintik putih ini akhirnya akan bergabung, sampai seluruh gambar putih.
4. Himpunan semua komponen yang terhubung dalam urutan adalah himpunan semua wilayah ekstrem

Secara opsional, bingkai elips dilampirkan ke MSER dengan memasang elips ke setiap daerah. Deskripsi setiap wilayah tersebut disimpan sebagai fitur.

Kata "*extremal*" merujuk pada properti yang semua piksel di dalam MSER memiliki intensitas lebih tinggi (wilayah dengan tingkat kecerahan tinggi) atau lebih rendah (wilayah ekstrem gelap) daripada semua piksel pada batas luarnya [29].

Operasi ini dapat dilakukan dengan terlebih dahulu mengurutkan semua piksel berdasarkan nilai abu-abu dan kemudian secara bertahap menambahkan piksel ke setiap komponen yang tersambung saat ambang diubah. Setiap area akan diperhatikan, sedemikian rupa sehingga variasi mereka mencapai ambang minimal dapat didefinisikan secara maksimal stabil [13].

2.4.3. *Binarization*

Proses binarization yaitu proses pengolahan suatu citra yang bertujuan untuk menghilangkan objek-objek yang tidak diperlukan seperti latar belakang dari citra atau Background dengan memberi warna hitam. Untuk mendapatkan suatu citra biner maka dibutuhkan sebuah citra *grayscale* yang dilakukan tresholding terhadapnya berdasarkan nilai ambang batas (*treshold*) yang ditentukan. Jika nilai piksel pada citra *grayscale* melebihi atau menyamai nilai treshold, maka nilai piksel tersebut dikonversi menjadi 255 namun jika nilai piksel kurang dari *threshold*, maka nilai piksel tersebut dikonversi menjadi 0 [14].

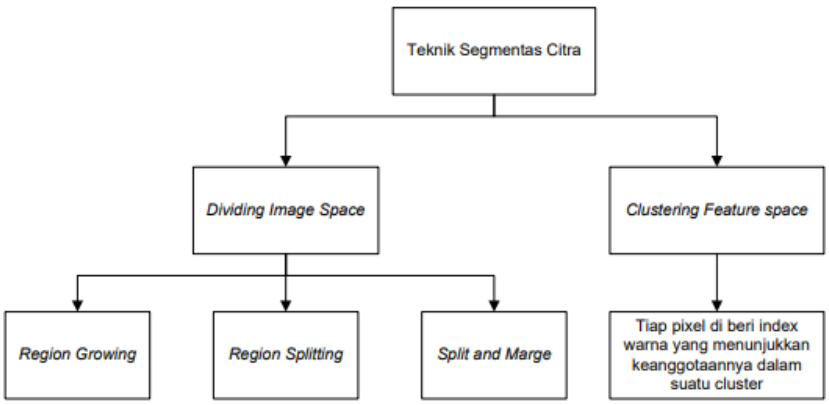
Nilai satu akan mewakili warna putih dan satu akan mewakili warna hitam, sehingga nilai yang terdapat pada setiap pixel citra hanya memiliki dua domain. Cara menentukan suatu pixel menjadi satu atau nol berdasarkan nilai keabuan dari suatu pixel dibandingkan dengan nilai ambang batas yang ditentukan yaitu 127. Berikut rumus penentuan nilai biner suatu pixel B(x,y) ditentukan oleh nilai hasil keabuan sebelumnya G(x,y) dibandingkan dengan nilai ambang batas T.

$$B(x,y) \begin{cases} a1, & \text{jika } G(x,y) >= T \\ a2, & \text{jika } G(x,y) < T \end{cases}$$

Gambar 2.9 Ambang binerisasi

2.4.4. *Image Segmentation*

Segmentasi citra adalah suatu proses membagi suatu citra menjadi wilayah-wilayah yang homogen. Menurut Jain, segmentasi citra dapat dibagi dalam beberapa jenis, yaitu *dividing image space* dan *clustering feature space*. Jenis yang pertama adalah teknik segmentasi dengan membagi image manjadi beberapa bagian untuk mengetahui batasannya, sedangkan teknik yang kedua dilakukan dengan cara memberi index warna pada tiap piksel yang menunjukkan keanggotaan dalam suatu segmentasi. Teknik segmentasi citra, 11 menurut Jain dapat dilihat pada diagram berikut [15].



Gambar 2.10 Teknik Segmentasi Citra

Adapun teknik segmentasi tersebut dapat dilakukan dengan beberapa pendekatan sebagai berikut :

1. Pendekatan *Edge-Based*

   Pendekatan ini melakukan proses deteksi sisi dengan operator gradient. Masukannya berupa citra *gray* level dan keluarannya berupa citra *edge* (biner). Selanjutnya dilakukan proses region growing dengan masukan citra asli (*gray-level*) dan citra edge. Proses pembentukan suatu wilayah akan berhenti bila menjumpai piksel *edge*. Kekurangan dari pendekatan

ini adalah belum tentu menghasilkan edge yang kontinu, mengakibatkan terjadinya kebocoran wilayah (wilayah-wilayah yang tidak tertutup).

2. Pendekatan *Region-Based*

   Pendekatan ini memerlukan *criteria of uniformity*, memerlukan penyebaran *seeds* atau dapat juga dengan pendekatan *scan line*, kemudian dilakukan proses *region growing*. Kekurangan dari pendekatan ini adalah belum tentu menghasilkan wilayah-wilayah yang bersambungan.

3. Pendekatan *Hybrid*

   Pendekatan ini melakukan proses deteksi sisi untuk menghasilkan citra sisi (piksel edge dan piksel *non-edge*), melakukan pemisahan wilayah dengan metode *connected region*. (*Connected regions* adalah set piksel 4-tetangga yang bukan piksel *edge*), dan selanjutnya dilakukan proses merging regions. Pendekatan ini bertujuan untuk mendapatkan hasil segmentasi dengan wilayah-wilayah yang tertutup dan bersambungan. Sedangkan menurut Sutoyo segmentasi citra bertujuan untuk membagi wilayah-wilayah yang homogen. Segmentasi adalah salah satu metode penting yang digunakan untuk mengubah citra input ke dalam citra output berdasarkan atribut yang diambil dari citra tersebut. Segmentasi membagi citra ke dalam daerah intensitasnya masing-masing sehingga bisa membedakan antara objek dan backgroundnya. Pembagian ini tergantung pada masalah yang akan diselesaikan. Segmentasi harus dihentikan apabila masing-masing objek telah terisolasi atau terlihat dengan jelas. Tingkat keakurasian segmentasi tergantung pada tingkat keberhasilan prosedur analisis yang dilakukan dan diharapkan proses segmentasi memiliki tingkat keakuratan yang tinggi. Algoritma dari segmentasi terbagi dalam dua macam yaitu [16]:

   1. Diskontinuitas

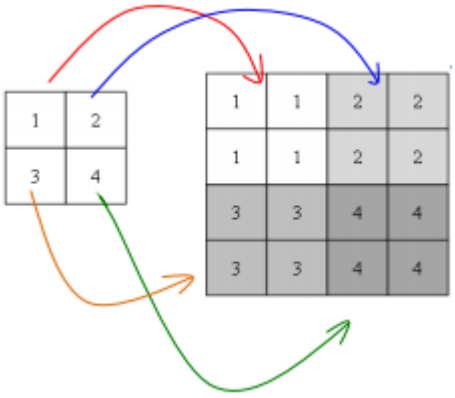
      Pembagian citra berdasarkan perbedaan dalam intensitasnya. Contohnya titik, garis, dan tepi.

   2. Similaritas

Pembagian citra berdasarkan kesamaan-kesamaan criteria yang 13 dimilikinya, contohnya thresholding, region growing, region splitting, dan region merging.

2.4.5. *Resize*

Dalam bahasa yang paling sederhana *resize* berarti mengubah resolusi atau ukuran horizontal dan vertikal suatu citra (gambar). *Resize* dapat membatasi dan atau meninggikan serta melebarkan citra yang sudah ditampilkan. Pasokan maksimal, terdapat pada pixel, untuk satu atau kedua dimensi. Seperti tercantum dalam preferensi , aspek rasio (Aspek rasio adalah rasio panjang dan lebar sebuah foto. Jika panjang dan lebar foto sama, maka aspek rasionya 1:1) gambar itu dipertahankan . Jika membatasi kedua tinggi dan lebar, beberapa aplikasi pengolahan citra akan mempertahankan aspek rasio dengan menggunakan maksimum yang ditetapkan untuk satu dimensi dan sesuatu yang kurang maksimal untuk dimensi lainnya. Hal ini diperlukan untuk meminimalisir besar image, sehingga process dapat berjalan dengan baik dan cepat. Pada dasarnya resize image hanya ada dua, yaitu memperbesar dan memperkecil citra. Dalam perbesaran citra, maka akan dilakukan dengan membuat setiap piksel menjadi beberapa pixel. Gambar 2.11 memberikan contoh cara memperbesar citra.



Gambar 2.11 Teknik *Resize*

Pada contoh di atas pembesaran pada arah vertikal dan horizontal sebesar 2 kali. Ada juga perbesaran citra yang berbeda antara vertical dan horizontalnya. Jadi tergantung algoritma yang digunakan untuk mengatur perbesaran citra tersebut. Secara prinsip, pengecilan citra berarti mengurangi jumlah pixel. Algoritma yang

digunakan untuk mewujudkan perbesaran citra dapat digunakan untuk melakukan pengecilan citra dengan mengganti nilai pembanding menjadi nilai pecahan seperti 1/2, ¼, 1/8, dan seterusnya. Proses penentuan koordinat baru menggunakan rumus sebagai berikut [11].

$$x = \frac{pb * pp}{pa} \quad \text{..... (2.1)}$$

Keterangan:

x = Nilai piksel baris baru

pb = Ukuran panjang matriks baru

pp = Posisi piksel baris

pa = Ukuran panjang matriks lama

$$y = \frac{lb * lp}{la} \quad \text{..... (2.2)}$$

Keterangan:

y = Nilai piksel kolom baru

lb = Ukuran lebar matriks baru

lp = Posisi piksel kolom

la = Ukuran lebar matriks lama

2.5. *Artificial Intelligence (AI)*

Artificial intelligence (AI) merupakan salah satu metode untuk menciptakan mesin dengan kemampuan kecerdasan, terutama dalam pembuatan aplikasiatau program komputer cerdas. Paada umumnya AI bertujuanuntuk untuk menciptakan komputer, robot, atau aplikasi atau program yang memiliki kecerdasan, layaknya seperti manusia [17].

Tujuan diciptakannya AI itu sendiri untuk:

a. Menciptakan sistem pakar, yakni suatu sistem yang memuliki sifat dan berperilaku cerdas, mampu belajar, mendemonstrasikan, menjelaskan, dan menyarankan pengguna.
b. Melakukan implementasikan kecerdasan ke dalam mesin, sehingga menciptakan suatu sistem kecerdasan yang mampu memahami, berpikir, belajar, dan berperilaku menyerupai manusia manusia.

AI diantaranya adalah kontribusi dari bidang Ilmu Komputer, Biologi, Psikologi, Bahasa, Matematika, dan Teknik. Salah satu kemajuan besar dalam menciptakan komputer yang dapat memiliki kecerdasan adalah, berpikir dengan logika, belajar, dan menyelesaikan permasalahan. Metode yang digunakan oleh AI yaitu dengan menyelesaikan permasalahan serta mengolah suatu informasi dan pengetahuan sehingga dapat diakses dan dipahami dengan mudah oleh pengguna, mudah dimanipulasi dalam perbaikan *error*, dan dapat berguna di berbagai situasi walaupun masih belum sempurna atau akurat.

2.6. *Machine Learning*

Machine learning merupakan teknik-teknik yang dapat mendukung dalam memberikan penanganan dan memelakukan prediksi data yang sangat besar dengan mengkaji data-data tersebut dengan suatu algoritma pembelajaran [18]. Istilah *machine learning* pertama kali didefinisikan oleh Arthur Samuel pada tahun 1959. Menurut Arthur Samuel, *machine learning* adalah suatu bidang ilmu komputer yang memberikan kemampuan pembelajaran kepada komputer untuk mengetahui sesuatu tanpa pemrograman yang jelas.

*Machine learnin*g dapat didefinisikan sebagai metode komputasi berdasarkan pengalaman untuk meningkatkan performa atau membuat prediksi yang akurat. Definisi pengalaman disini ialah informasi sebelumnya yang telah tersedia dan bisa dijadikan data pembelajar. II-2 Dalam pembelajaran *machine learning*, terdapat skenario-skenario seperti berikut [19].

1. *Supervised Learning*

   Penggunaan skenario *supervised learning*, pembelajaran menggunakan masukan data pembelajaran yang telah diberi label. Setelah itu membuat prediksi dari data yang telah diberi label.

2. *Unsupervised Learning*

   Penggunaan skenario *unsupervised learning*, pembelajaran menggunakan masukan data pembelajaran yang tidak diberi label. Setelah itu mencoba untuk mengelompokan data berdasarkan karakteristik-karakteristik yang ditemui.

3. *Reinforcement learning*

Pada skenario reinforcement learning fase pembelajaran dan tes saling dicampur. Untuk mengumpulkan informasi pembelajar secara aktif dengan berinteraksi ke lingkungan sehingga untuk mendapatkan balasan untuk setiap aksi dari pembelajar.

2.7. *Deep Learning*

Deep Learning merupakan salah satu bidang dari *machine learning* yang memanfaatkan jaringan syaraf tiruan untuk implementasi permasalahan dengan *dataset* yang besar. Teknik deep learning memberikan arsitektur yang sangat kuat untuk *supervised learning*. Dengan menambahkan lebih banyak lapisan maka model pembelajaran tersebut bisa mewakili data citra berlabel dengan lebih baik. Pada *machine learning* terdapat teknik untuk menggunakan ekstraksi fitur dari data pelatihan dan algoritma pembelajaran khusus untuk mengklasifikasi citra II-3 maupun untuk mengenali suara. Namun, metode ini masih memiliki beberapa kekurangan baik dalam hal kecepatan dan akurasi.

Aplikasi konsep jaringan syaraf tiruan yang dalam (banyak lapisan) dapat ditangguhkan pada algoritma *machine learning* yang sudah ada sehingga komputer sekarang bisa belajar dengan kecepatan, akurasi, dan skala yang besar. Prinsip ini terus berkembang hingga *deep learning* semakin sering digunakan pada komunitas riset dan industri untuk membantu memecahkan banyak masalah data besar seperti *Computer vision*, *Speech recognition*, dan *Natural Language Processing*. *Feature Engineering* adalah salah satu fitur utama dari *deep learning* untuk mengekstrak pola yang berguna dari data yang akan memudahkan model untuk membedakan kelas. *Feature Engineering* juga merupakan teknik yang paling penting untuk mencapai hasil yang baik pada tugas prediksi. Namun, sulit untuk dipelajari dan dikuasai karena kumpulan data dan jenis data yang berbeda memerlukan pendekatan teknik yang berbeda juga [18].

2.8. *Neural Network*

Neural Network atau istilah lainnya adalah Jaringan Saraf Tiruan (JST) merupakan model yang mengikuti dari karakteristik jaringan *neural* yang ada pada sistem biologis manusia. Jaringan ini mampu memproses sistem informasi untuk melakukan pelatihan dengan memperbaiki setiap bobot atau beban dengan

menyesuaikan data-data yang diberikan sebagai data latih oleh jaringan. Data latih yang baik akan membuat jaringan menjadi semakin mengenal setiap data yang dipelajarinya [6]. Haykin mengatakan bahwa sebuah Jaringan Saraf Tiruan adalah sebuah prosesor yang terdistribusi secara massal yang memiliki kecenderungan alami untuk menyimpan experiential knowledge dan membuatnya dapat digunakan kembali. Hal ini mencerminkan otak manusia dalam dua hal [20]:

1. *Knowledge* didapatkan dari hasil proses pembelajaran.
2. Kekuatan dari koneksi *inter-neuron* atau disebut juga *weight* sinapsis, digunakan untuk menyimpan *knowledge*.

Tujuan dari jaringan saraf tiruan adalah untuk belajar mengenali pola-pola pada data dan mensimulasikan proses belajar adaptif biologis, walau dalam skala yang sangat sederhana. Sekali jaringan saraf tiruan telah dilatih terhadap data, akan dapat membuat prediksi dengan melakukan deteksi kemiripan atau kesamaan pola-pola data masukan.

JST bukanlah duplikasi persis dari sistem biologis otak manusia, tetapi jaringan saraf tiruan ini dapat melakukan kemampuan seperti generalisasi, belajar, abstaksi, dan bahkan intuisi. Mudahnya, merupakan suatu model dari sistem saraf biologis yang disederhanakan sebagai suatu alternatif sistem komputer. Kenyataan menunujukan bahwa banyak masalah dalam kehidupan manusia yang sulit dipecahkan dengan “komputer konvensional” yang paling canggih sekalipun, namun manusia dapat menyelesaikannya dengan baik. Dengan kemampuannya untuk belajar, jaringan saraf tiruan ini diharapkan dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh komputer konvensional.

Jaringan saraf tidak diprogram dalam arti tradisional. Sebaliknya dilatih dengan contoh. Latihan itu terdiri dari banyak pengulangan input yang mengungkapkan berbagai hubungan. Dengan memperhalus bobot node sistem (neuron yang disimulasikan) secara progresif, jaringa saraf tiruan ini “menemukan” hubungan antar input. Proses penemuan ini menandakan “belajar”.

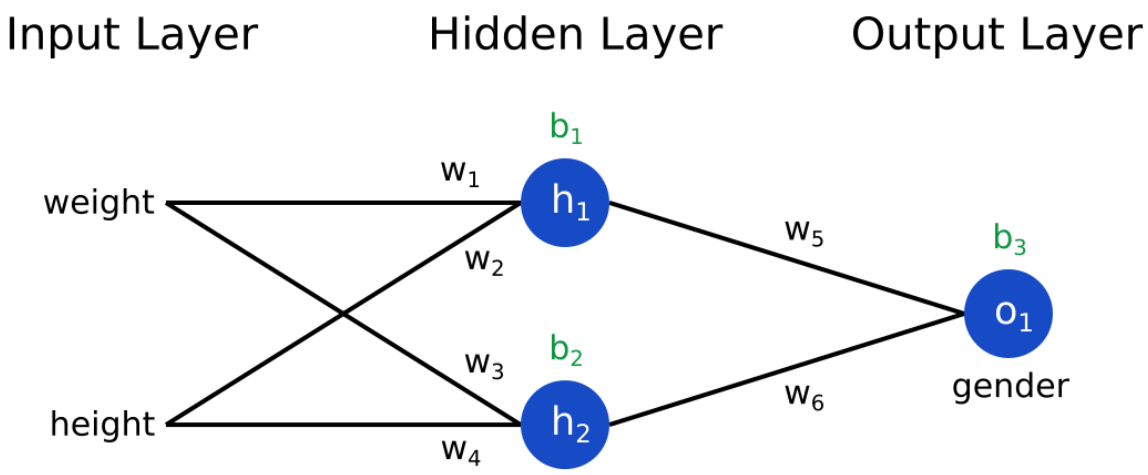
JST merupakan salah satu bentuk dari Kecerdasan Buatan. JST dipandang sebagai suatu Black Box yang dapat melakukan prediksi keluaran dari suatu pola masukan yang dia kenali. Untuk itu JST harus dilatih terlebih dahulu terhadap sejumlah pola masukan dan target yang diharapkan dari tiap pola masukan tersebut

(Supervised Learning). Sekali dilatihkan, JST akan dapat mengenali kesamaan ketika dihadapkan terhadap pola masukan baru, dan menghasilkan prediksi pola keluarannya.

JST dapat mendeteksi kesamaan masukan, bahkan sebagian masukan yang mungkin belum pernah dilatihkan atau diberikan sebelumnya. Karena JST mempunyai kemampuan interpolasi yang hebat, terutama bila data masukan tidak eksak, banyak gangguan didalamnya. Sehingga memungkinkan JST untuk digunakan sebagai subtitusi langsung bagi auto korelasi, regresi multivariabel, regresi linier, trigonometri, dan teknik regresi lainnya. Ketika data dianalisa menggunakan JST, akan memungkinkan untuk melakukan prediksi pola yang penting sebagaimana bila seorang ahli menganalisa data tersebut, karena JST dapat beraksi seperti selayaknya seorang yang ahli di bidangnya [21].

Neural Network dibagi menjadi bagian-bagian lebih kecil, masing-masing memiliki metode dan ciri-ciri tersendiri serta keunggulan yang berbeda dalam mendapatkan suatu pola kasus data. Metode yang terbagi tersebut antara lain adalah *Backpropagation, Bidirectional Memory* atau biasa dikenal dengan istilah BAM, *Hopfield Network Propagation Network* dan metode lainnya yang terus dilakukan perkembangan pada metode-metode yang berkaitan dengan jaringan saraf tiruan. Metode *Neural Network* yang yang digunakan pada skripsi ini adalah metode *Backpropagation* [21].

Neural Network umungnya dikenal dari karakteristik modelnya yang terdiri dari *layer-layer* atau bisa disebut lapisan-lapisan. Lapisan pada *Neural Network* antara lain memiliki lapisan *input*, lapisan *hidden*, dan lapisan *output*. Masing-masing lapisan memiliki sejumlah *node* atau istilah biologis nya adalah *neuron* yang sudah ditentukan di setiap lapisannya. Setiap node akan saling terhubung membentuk jaringan dimulai dari lapisan *input* menuju lapisan *hidden*, dan terakhir dari lapisan hidden menuju lapisan *output*.



Gambar 2.12 Model *Neural Network* Sederhana

2.8.1. Sejarah Singkat Neural Network

Perkembangan dari ilmu jaringan saraf biologi telah memungkinkan para peneliti untuk mengembangkan model matematika dari neuron untuk mensimulasikan tingkah laku jaringan saraf. Usaha keras untuk dapat mengenali cara kerja otak dikerjakan dan dimulai pertama kali oleh Ramon Cajal yang mengenalkan ide neuron yaitu struktur pokok dari otak manusia. Ide ini kemudian dikembangkan lagi awal tahun 1940-an ketika model abstrak pertama neuron diperkenalkan oleh MCCulloch dan Pitts. Hebb menjelaskan bagaimana jaringan saraf belajar. Peneliti yang lain mengembangkannya dalam dua dekade berikutnya, seperti Minsky dan Rosenblatt. Rosenblatt mengembangkan algoritma belajar perceptron. Pada waktu yang sama Windrow dan Hoff mengembangkan suatu variasi penting dari algoritma belajar perceptron, yang kemudian dikenal dengan Windrow-Hoff *rule*.

Kemudian Minsky dan Pert mengemukan teori batas model neural network single-layer yang ditulis pada buku mereka Perceptron. Dikarenakan rasa pesimistis terhadap proyek mengenai neural network ini, penelitian dari neural network mengalami perubahan dan kemunduran dalam dua dekade. Walapun dalam suasana yang tidak baik, beberapa peneliti masih meneruskan penelitian 7 mereka dan menghasilkan hasil yang berarti. Contohnya, Anderson dan Grossberg, yang melakukan pekerjaan penting mengenai model Psychological. Kohonen menegambangkan asosiatif model memori.

Kebangkitan neural network pada awal 1980-an. Hopfield mengenalkan ide meminimalkan energi secara fisik dalam neural network. Akibat dari tulisannya tersebut mengilhami teknologi ini dengan momentum yang dapat diperbaharui. Feldman dan Ballard pada tahun 1982 mengemukan syarat "connectionist" yang terkenal.

Pada pertengahan 1980-an, dalam buku Parallel Distibuted Processing yang ditulis oleh Rumelhart dan McClelland, menimbulkan dampak besar pada pengetahuan komputer, dan biologi. Khususnya, algoritma pembelajaran backpropagation yang dikembangkan oleh Rumelhart, Hinton, dan Williams menghasilkan suatu solusi yang sangat berati dalam melakukan suatu training terhadap multilayer neural network. Keberhasilan spektakuler dari pendekatan ini di demonstrasikan oleh system NETtalk yang dikembangkan Sejnowski dan Rosenberg, yaitu suatu sistem yang dapat menterjemahkan teks dalam bahasa Inggris kedalam bentuk suara dengan gaya bahasa yang dapat dimengerti dengan sangat jelas.

Pendekatan simbolik yang telah lama didominasi oleh bagian dari AI akhirnya telah dimenangkan oleh pendekatan neural network. Telah ada spekulasi tentang apakah pendekatan pertama seharusnya mensubstistusikan ke yang lain atau apakah pendekatan kedua seharusnya jalan bersama dan dikombinasikan. Fakta-fakta lain yang menunjukkan lebih baik memilih integrasi yang pola lowlevel kemampuannya diakui oleh pendekatan neural network dan high-level yang 8 menjadi alasan kemampuan, disediakan oleh pendekatan secara simbolik yang dikomplemen dengan lainnya. Arsitektur yang optimal pada sistem berpikir masa depan mungkin lebih baik dibangun dari integrasi antara cara yang satu dengan lainnya [21].

2.8.2. Bobot, Bias, dan Error

Jaringan yang terhubung disetiap *node* disebut sebagai bobot atau *weight*. Bobot merupakan suatu nilai yang menjadi beban perhitungan kompleks sehingga dapat mempengaruhi nilai *output* dan nilai *error* yang akan didapatkan. Bias merupakan variabel bantuan secara "bias" untuk mengantisipasi terjadinya perkalian dengan nilai nol yang memungkinkan terjadinya perubahan pola pada

jaringan. Sedangkan *Error* merupakan gambaran nilai yang menunjukkan tingkat kesalahan yang terjadi ketika proses jaringan dilakukan.

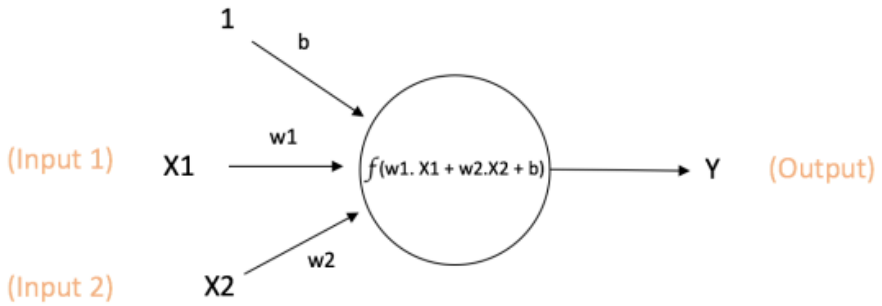
Bobot tersimpan didalam jaringan saraf tiruan atau biasa disebut sebagai bobot interkoneksi. Setiap bobot memiliki nilai awal dengan cara acak dengan nilai desimal dengan kisaran -0,1 sampai dengan 0,1 [22]. Selama jaringan dalam keadaan proses berlatih, setiap bobot akan diperbaharui berdasarkan *error* yang didapatkan. Pembaruan bobot dapat dikatakan sebagai proses pembelajaran jaringan dalam mengenali suatu pola dengan harapan jaringan dapat memberikan hasil *output* yang sesuai dengan yang diharapkan.

2.8.3. *Neural System*

Sistem jaringan adalah komponen-komponen yang ada pada model suatu jaringan yang saling berhubung dalam melakukan proses. Komponen yang terdapat pada jaringan saraf tiruan adalah sebagai berikut.

1) Node

Node atau istilah biologisnya adalah sel *neuron* terletak di setiap lapisan-lapisan jaringan yang memiliki tiga unsur nilai, yaitu nilai keluaran, *error*, dan bobot. *Node* pada lapisan *hidden* dan lapisan *output* memiliki proses perhitungan untuk menghasilkan keluaran suatu *node*. Sistem yang terjadi pada *node* contohnya pada *node hidden* akan menerima bobot-bobot dan nilai-nilai dari *node* dari lapisan *input*. Selanjutnya akan dilakukan proses kalkulasi sehingga mendapatkan hasil dari suatu *node hidden* tersebut.



Gambar 2.13 Sistem pada *node*

2) Input Layer

Input layer adalah lapisan yang terletak paling kiri menjadi tempat nilai masukan akan masuk ke sistem jaringan. Pada lapisan masukan ini jumlah *node* menyesuaikan dengan jumlah nilai input yang akan diterima oleh jaringan.

3) Hidden Layer

Sedangkan *hidden layer* merupakan lapisan tersembunyi yang terletak diantara lapisan masukan dan lapisan keluaran. Lapisan tersembunyi ini dapat memiliki lebih dari satu lapisan. Semakin banyak lapisan tersembunyi pada jaringan maka akan semakin kompleks dan akan semakin baik hasil keluaran yang akan diberikan. Namun dampak menggunakan lapisan tersembunyi yang sangat banyak akan mempengaruhi waktu proses pelatihan yang menjadi semakin lama [22].

4) Output Layer

Output layer merupakan lapisan terakhir terletak paling kanan. Lapisan ini memiliki *node* sebanyak jumlah kelas *output* yang dimiliki. Setiap *node* akan mewakili satu keputusan *output* hasil dari proses jaringan. sangat banyak akan mempengaruhi waktu proses pelatihan yang menjadi semakin lama [22].

2.9. *Backpropagation Neural Network*

Metode *backpropagation* sering juga disebut dengan *generalized delta rule*. *Backpropagation* adalah metode turunan *gradien* (*gradient descent method*) untuk meminimalkan total *square error* dari *output* yang dikeluarkan oleh jaringan. JST propagasi balik merupakan algoritma pelatihan terbimbing yang mempunyai banyak lapisan. JST propagasi balik menggunakan *error output* untuk mengubah nilai bobot-bobotnya dalam arah mundur (*backward*). Untuk 40 mendapatkan error ini, tahap perambatan maju (*forward propagation*) harus dikerjakan terlebih dahulu. Syarat fungsi aktivasi dalam JST propagasi balik adalah bersifat kontinu, terdifferensial dengan mudah, dan merupakan fungsi yang tidak turun. Fungsi aktivasi yang dapat memenuhi ketiga syarat tersebut adalah *logsig*, *tansig*, dan *purelin* [2].

Metode pengenalan merupakan proses inisialisasi data yang akan diolah selanjutnya oleh JST propagasi balik. Data yang akan dikenali disajikan dalam

bentuk vektor. Masing-masing data mempunyai target yang disajikan juga dalam bentuk vektor. Target atau keluaran acuan merupakan suatu peta karakter yang menunjukkan lokasi dari vektor masukan. Sedangkan metode pelatihan merupakan proses latihan mengenali data dan menyimpan pengetahuan atau informasi yang didapat ke dalam bobot-bobot [23]. Terdapat 3 fase dalam pelatihan (*training*) JST propagasi balik, yaitu fase maju (*feedforward*), fase mundur (*backpropagation*), dan fase modifikasi bobot. Dalam fase *feedforward*, pola masukan dihitung maju dimulai dari lapisan input hingga lapisan *output*. Dalam fase back propagation, tiap-tiap unit output menerima target pola yang berhubungan dengan pola input untuk dihitung nilai kesalahan. Kesalahan tersebut akan dipropagasikan mundur. Sedangkan fase modifikasi bobot bertujuan untuk menurunkan kesalahan yang terjadi. Ketiga fase tersebut diulang secara terus menerus hingga kondisi penghentian dipenuhi [2].

2.9.1. Algoritma pada *Backpropagation Neural Network*

Proses awal yang dilakukan pada proses *training* adalah melakukan persiapan data yang akan di latih dan mempersiapkan bobot secara acak dari nilai -0.5 sampai dengan 0.5. Seperti pada penerapan inisialisasi *Nguyen – Widrow* bobot disiapkan antara -0.5 sampai dengan 0.5. Menurut Fausett, inisialisasi ini adalah penyederhanaan nilai acak berdasarkan analisa geometrid dan pendekanan transformasi *Fourier* bertujuan untuk meningkatkan kecepatan belajar jaringan [24]. Lalu setelah proses inisialisasi selanjutnya akan melakukan proses *feedforward*, pengecekan *error*, *backpropagation*, ketiga proses utama ini akan terus berulang hingga batas yang telah ditentukan. Batas yang digunakan dapat dengan membandingkan fungsi *target error*, dengan membandingkan tingkat error setiap belajar dengan *target errortarget* yang ditentukan. Atau dapat membatasi proses *training* dengan membatasi jumlah iterasi yang akan dilakukan proses *training*. Berikut langkah algoritma *backpropagation neural network* pada proses pelatihan.

Langkah 0 : Inisialisasi bobot (setiap bobot diberi nilai acak dari -0.5 sampai 0.5).

Langkah 1 : Selama kondisi berhenti belum terpenuhi, maka lakukan langkah 2-9.

Langkah 2 : Setiap nilai input lakukan langkah 3-8.

FEED FORWARD (fase maju)

Langkah 3 : Setiap masukan (X_i, i = 1, 2, …, n) menerima sinyal masukan Xi dan

meneruskannya ke seluruh unit atau *node* selanjutnya ke lapisan hidden

Langkah 4 : Setiap *node hidden* (Z_j, j = 1, 2, …, n) melakukan proses perhitungan

total sinyal masukan sebagai berikut.

$$Z\_in\_j = V_{0j} + \sum_{j=1}^{p} X_i V_{ij} \quad \text{…… (2.3)}$$

lalu menghitung nilai keluarannya menggunakan fungsi aktivasi.

$$Z_j = f(Z\_in\_j) \quad \text{…… (2.4)}$$

Selanjutnya nilai ini dikirimm ke seluruh unit lapisan *output*.

Langkah 5 : Setiap *node output* (Y_k, k = 1, 2, …, m) menghitung total nilai

masukan berbobot dengan rumus sebagai berikut.

$$Y\_in\_k = W_{0k} + \sum_{j=1}^{p} Z_j W_{jk} \quad \text{…… (2.5)}$$

Lalu dihitung nilai keluarannya dengan fungsi aktivasi.

$$Y_k = f(Y\_in\_k) \quad \text{…… (2.6)}$$

BACKPROPAGATION ERROR (fase mundur)

Langkah 6 : Setiap *node output* (Y_k, k = 1, 2, …, m) menerima pola target yang

sesuai dengan pola masukan pelatihan. *Node* tersebut akan melakukan perhitungan nilai *error* yang terjadi.

$$\delta_k = (t_k - y_k)\, f'(Y\_in\_k) \quad \text{……} \quad (2.7)$$

selanjutnya menghitung kondisi bobot untuk mengubah bobot W_{jk}

$$\Delta W_{jk} = \alpha\, \delta_k\, Z_j \quad \text{……} \quad (2.8)$$

$$\Delta W_{0k} = \alpha\, \delta_k \quad \text{……} \quad (2.9)$$

Langkah 7 : Setiap *node hidden* (Z_j, j = 1, 2, …, m) menghitung selisih *input*

$$\delta\_in\_j = \sum_{j=1}^{p} \delta_k\, W_{jk} \quad \text{……} \quad (2.10)$$

$$\delta = \delta\_in\_j\; f'(Z\_in\_j) \quad \text{……} \quad (2.11)$$

$$\Delta V_{ij} = \alpha\, \delta_j X_i \quad \text{……} \quad (2.12)$$

$$\Delta V_{0j} = \alpha\, \delta_j \quad \text{……} \quad (2.13)$$

Langkah 8 : mengubah nilai bobot menjadi nilai baru.

Setiap *node output* (Y_k, k = 1, 2, …, m; j = 0, 1, 2, … p)

$$\Delta W_{jk(new)} = W_{jk(old)} + \Delta W_{jk} \quad \text{……} \quad (2.14)$$

Setiap *node output* (Z_j, j = 1, 2, …, p; i = 0, 1, 2, …, n)

$$\Delta V_{ij(new)} = V_{ij(old)} + \Delta V_{ij} \quad \text{……} \quad (2.15)$$

Langkah 9 : cek kondisi berhenti.

Sedangkan pada proses *testing,* proses yang dilakukan adalah melakukan *feedforward* tanpa melakukan *backpropagate* lalu mencari nilai output paling

tinggi. Maka citra yang dimasukkan melalui *node-node* input akan diproses untuk dikenali sebagai suatu *class output*.

2.9.2. Fungsi Aktivasi

Fungsi aktivasi merupakan fungsi yang akan membantu proses perhitungan jaringan saraf tiruan, menurut Fausset fungsi yang dapat digunakan adalah sebagai berikut.

a. Fungsi identitas

Fungsi ini dugunakan pada umumnya pada jaringan lapis tunggal. Fungsi identitas akan menghasilkan nilai yang sama dengan masukannya. Dan grafik yang ditunjukkan akan berbentuk garis lurus dan berikut adalah persamaan fungsinya.

$$f(x) = x => \forall x \quad \text{..... (2.16)}$$

b. Fungsi tangga biner

Fungsi tangga biner adalah fungsi identitas dengan proses pembulatan yang dipengaruhi oleh parameter pembulatan $\emptyset$. Untuk $\emptyset$ = 1, fungsi ini akan mengembalikan nilai satu atau nol. Persamaan fungsi tangga biner ditulis sebagai berikut.

$$f(x) = 1, \ jika\ x \geq 0 \quad \text{..... (2.17)}$$

$$f(x) = 0, \ jika\ x < 0 \quad \text{..... (2.18)}$$

c. Fungsi *sigmoid* biner

Fungsi *sigmoid biner* akan bergantung pada *steepness* parameter (σ). Agar hasil dari proses perhitungan dengan fungsi ini hanya akan mengeluarkan nilai 0 sampai dengan 1, maka σ = 1 akan menghasilkan grafik yang tidak linier. Persamaan fungsi *sigmoid* biner adalah sebagai berikut [24].

$$f(x) = \frac{1}{1+e^{-\sigma \mathrm{x}}} \quad \text{..... (2.19)}$$

2.10. Sertifikat

Sertifikat adalah sebuah surat pernyataan atau tanda tertulis tercetak dari orang yang berwenang berupa bukti kepemilikan atau suatu kejadian. Contohnya

sertifikat tanah, pelatihan, seminar, hasil ujian, dan sertifikat lainnya yang banyak orang pasti pernah melihat salah satu contohnya [3]. Sertifikat merupakan dokumen yang sangat penting antara lain pada bidang administrasi atau yang berhubungan dengan manajemen sumber daya manusia. Badan Nasional Sertifikasi Profesi (BNSP) memberikan penjelasan mengenai sertifikat, merupakan suatu dokumen yang diberikan sebagai pengakuan dari keterampilan dan pengetahuan kerja dilakukan dengan cara yang objektif mengacu kepada standar kerja nasional maupun internasional [25]. Artinya dokumen sertifikat merupakan peran penting yang dimiliki oleh suatu individu atau organisasi sebagai tanda bukti yang akan diperlukan, dan disimpan hingga masa kedepan.

Gambar 2.14 Sertifikat

2.11. *Unified Modelling Language* (UML)

Diagram UML adalah sekumpulan alat yang digunakan untuk melakukan abstraksi terhadap sebuah sistem atau perangkat lunak berbasis objek. *Unified Modelling Language* (UML) adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML, perancang atau pemodel dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. UML lebih cocok diterapkan pada piranti berorientasi

objek seperti C++, Java, C#, dan sebagainya. Tetapi UML juga tetap dapat digunakan untuk modeling aplikasi prosedural semisal VB atau C.

UML versi 2.0 mencakup 13 macam diagram dan perangkat yang berfungsi untuk menggambarkan sistem informasi berorientasi objek dengan sangat lengkap dan rinci. Meski demikan, tidak selalu ke-13 diagram dan perangkat tersebut digunakan saat para pengembang berupaya mengemabangkan perangkat lunak berorientasi objek [11].

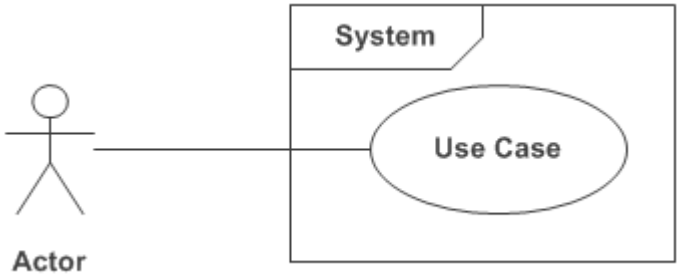
Dalam UML sendiri terdapat beberapa diagram yang wajib dikuasai yaitu:

1. *Structural Diagram*
    a. *Class Diagram* ini terdiri dari *class*, *interface*, *association*, dan *collaboration*. Diagram ini menggambarkan objek-objek yang ada di sistem.
    b. *Deployment Diagram* ini menggambarkan kumpulan node dan hubungan antar node. Node adalah entitas fisik dimana komponen di-deploy. Entitas fisik ini dapat berupa server atau perangkat keras lainnya.
2. *Behavioral Diagram*
    a. *Use case Diagram* ini menggambarkan kumpulan use case, aktor, dan hubungan mereka. *Use case* adalah hubungan antara fungsionalitas sistem dengan aktor internal/eksternal dari sistem.
    b. *Sequence Diagram*, diagram ini menggambarkan interaksi yang menjelaskan bagaimana pesan mengalir dari objek ke objek lainnya.
    c. *Statechart Diagram*, diagram ini menggambarkan bagaimana sistem dapat bereaksi terhadap suatu kejadian dari dalam atau luar. Kejadian (*event*) ini bertanggung jawab terhadap perubahan keadaan sistem.
    d. *Activity Diagram*, menggambarkan aliran kontrol sistem. Diagram ini digunakan untuk melihat bagaimana sistem bekerja ketika dieksekusi [11].

2.11.1. *Use Case Diagram*

Sebuah *Use Case Diagram* menyatakan visualisasi interaksi yang terjadi antara pengguna (aktor) dengan sistem. Diagram ini menjelaskan konteks dari sebuah sistem sehingga terlihat jelas batasan dari sistem. Ada 2 elemen penting

yang harus digambarkan, yaitu aktor dan use case. Aktor dinotasikan dengan simbol gambar orang-orangan (*stick-man*) dengan nama di bagian bawah yang menyatakan peran/sistem. Aktor bisa bersifat primer, yaitu yang menginisiasi berjalannya sebuah use case, atau sekunder, yaitu yang menginisiasi berjalannya sebuah *use case*. *Use case* dinotasikan dengan simbol elisps dengan nama kata kerja aktif di bagian dalamnya yang menyatakan aktivitas dari persperktif aktor. Setiap aktor dimungkinkan berinteraksi dengan sistem dalam banyak use case [26]. Berikut merupakan gambar komponen yang digunakan dalam *Use Case Diagram.*



Gambar 2.15 Model *Use Case Diagram*

2.11.2. *Use Case Scenario*

Use Case scenario adalah penjelasan secara tekstual dari sekumpulan skenario interaksi. Setiap skenario mendeskripsikan urutan aksi/langkah yang dilakukan aktor ketika berinteraksi dengan sistem, baik yang berhasil maupun gagal [26]. Kegunaan dari *use case scenario* sendiri adalah untuk memperjelas proses yang ada di dalam masing-masing *use case.*

Use case Melihat Data Diri Mahasiswa	
Tujuan	Mengijinkan administrator untuk melihat data diri mahasiswa
Aktor	Administrator
Kondisi awal	Login tervalidasi dan valid
Skenario utama	1. Administrator memilih menu data diri mahasiswa 2. Sistem menampilkan daftar mahasiswa, untuk dilihat data dirinya
Skenario alternatif	1. Jika data diri mahasiswa yang dipilih masih kosong, maka sistem akan menunjukkan pesan “data diri masih kosong”. 2. Jika data diri mahasiswa yang dipilih masih belum lengkap, maka sistem akan menampilkan pesan “data diri masih belum lengkap”.
Kondisi akhir	Sistem menampilkan data diri mahasiswa sesuai dengan identitas mahasiswa yang dipilih.

Gambar 2.16 *Use Case Scenario*

2.11.3. *Activity Diagram*

Activity diagram digunakan untuk menjelaskan alur atau cara dari sebuah sistem mencapai tujuannya dengan diagram. *Activity* diagram sendiri menjelaskan alurnya dengan cara *actions chained* atau aksi-aksi yang saling berhubungan pada sistem tersebut.

2.11.4. *Class Diagram*

Class diagram digunakan untuk menjelaskan bagian objek-objek yang berbeda yang dibutuhkan pada sistem. *Class* diagram adalah bagian terpenting dari *object-oriented sistem* maka dari itu *class* diagram adalah yang paling populer dari UML. *Class* diagram menggambarkan hubungan antar kelas dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem [27].

2.11.5. *Sequence Diagram*

Sequence diagram digunakan untuk menjelaskan urutan hubungan atau interaksi terhadap bagian-bagian dari sistem. Dengan sequence diagram setiap interaksi dapat dijelaskan akan berjalan ketika sesuatu menjalankannya atau melakukan trigger terhadap interaksi tersebut [25].

2.12. *Python*

Python dikembangkan oleh Guido van Rossum pada tahun 1990 di CWI, Amsterdam sebagai kelanjutan dari bahasa pemrograman ABC. *Python* adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python merupakan salah satu bahasa pemrograman yang populer di dunia kerja Indonesia.

Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan dengan sintaks kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Pada umumnya *Python* mendukung multi paradigma pemrograman, seperti pemrograman berorientasi objek, pemrograman imperatif dan pemrograman fungsional. Salah satu fitur yang tersedia pada *python* adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip

meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa script. *Python* dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi [28], Saat ini kode python dapat dijalankan di berbagai platform sistem operasi.

2.13. *Python Image Library*

Python Imaging Library atau biasa dikenal dengan singkatan PIL adalah *library open source* untuk bahasa pemprograman *python* membantu menambahkan kemampuan pengolahan citra untuk interpreter python. *Library* ini mendukung banyak format file, dan memberikan kemampuan dalam mengolah gambar dan grafis. *Library* ini dapat digunakan pada sistem operasi Windows, Mac OS X dan Linux [28].

2.14. *OpenCV*

OpenCV (*Open Computer Vision*) adalah sebuah *library* yang mengkhususkan dirinya untuk pembangunan *computer vision* dan *machine learning*. *OpenCV* sendiri sudah memiliki sangat banyak algoritma yang teroptimasi dengan jumlah lebih dari 2500. *Library OpenCV* sendiri tidak hanya ada untuk bahasa pemrograman *Python* saja, melainkan ada pula untuk bahasa lain seperti C++, *Java*, dan *MATLAB* Interfaces dan sudah sangat *support* penggunaannya pada operasi sistem *Windows*, *Linux*, *Mac OS* dan *Android* [28].

2.15. *Numpy*

Numpy adalah sebuah library pada Python yang berguna untuk melakukan perhitungan scientific. Di dalamnya terdapat paket-paket untuk melakukan operasi terhadap objek array yang berupa matriks dengan N-dimensi, aljabar linear, dan banyak lagi.

2.16. *Npy* dan *Npz*

Npy dan Npz adalah sebuah file penyimpanan yang digunakan untuk penyimpanan array N-dimensi atau multi-dimensi. File Npy dan Npz sendiri dapat dibaca oleh library Numpy, sehingga akan dapat mempermudah proses-proses yang akan membutuhkan nilai dari array tersebut apabila dibutuhkan kembali.

2.17. Pengujian *Confusion Matrix*

Confusion matrix melakukan pengujian untuk memperkirakan obyek yang benar dan salah. Urutan pengujian ditabulasikan dalam confusion matrix dimana kelas yang diprediksi ditampilkan di bagian atas matriks dan kelas yang diamati dibagian kiri. Setiap sel berisi angka yang menunjukan berapa banyak kasus yang sebenernya dari kelas yang diamati untuk diprediksi [29]. Model *confusion matrix* untuk contoh 2 kelas dapat dilihat pada Tabel 2.1 sebagai berikut

Tabel 2.1 Model *Confusion Matrix*

		Nilai Prediksi	
		Positif	**Negatif**
Nilai Aktual	**Positif**	TP	FP
	Negatif	FN	TN

Keterangan:

TP = jumlah nilai positif yang diklasifikasikan positif

TN = jumlah nilai negatif yang diklasifikasikan negatif

FP = jumlah nilai positif yang diklasifikasikan negatif

FN = jumlah nilai negatif yang diklasifikasikan positif.

Perhitungan untuk mendapatkan akurasi dapat dilihat pada persamaan 2.18 berikut ini:

$$Akurasi = \frac{TP+TN}{TP+TN+FP+FN} \quad \text{..... (2.17)}$$

Untuk dapat menentukan nilai prediksi positif (PPV) menggunakan rumus sebagai berikut:

$$PPV = \frac{TP}{TP+FP} \quad \text{..... (2.18)}$$

Sedangkan untuk menentukan nilai prediksi negatif (NPV) menggunakan rumus sebagai berikut:

$$NPV = \frac{TN}{TN+FN} \quad \text{..... (2.19)}$$