BAB 2
LANDASAN TEORI

2.1 Pengolahan Citra

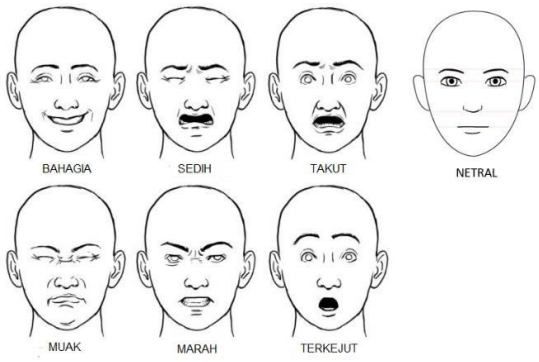
Citra didefinisikan sebagai fungsi dari dua variabel misalnya *a(x,y)* dimana *a* sendiri sebagai amplitudo (misalnya kecerahan) citra pada kordinat *(x,y)*[5]. Menurut Ian T. Young dkk, citra digital *a*[*m*,*n*] merupakan citra dalam ruang diskrit 2D yang berasal dari citra analog *a(x,y)* di ruang kontinyu 2D melalui proses sampling yaitu yang biasa kita sebut digitalisasi [8].

Sedangkan menurut Maria citra digital adalah citra *f(x,y)* yang telah didiskritkan pada kordinat spasial dan kecerahan. Citra digital direpresentasikan oleh *array* dua dimensi atau sekumpulan *array* dua dimensi dimana setiap *array* merepresentasikan satu kanal warna. Nilai kecerahan yang didigitalkan dinamakan nilai tingkat keabuan, setiap elemen *array* tersebut dinamakan piksel atau pel yang diambil dari istilah *'picture element'* [9]. Secara umum, pengolahan citra dapat didefinisikan sebagai pemrosesan sebuah gambar dua dimensi secara digital.

Pengolahan citra digital pada umumnya bertujuan memperbaiki kualitas pada suatu gambar sehingga dapat dengan mudah diinterpretasikan oleh mata manusia dan oleh komputer untuk mengolah informasi yang terdapat pada suatu gambar. Proses ini mempunyai data masukan dan informasi keluaran yang berbentuk citra [10].

2.2 Ekspresi Wajah

Ekspresi wajah manusia merupakan bentuk respon alami manusia yang menggambarkan perasaan manusia saat berinteraksi pada suatu hal tertentu. Ekspresi wajah merupakan bagian dari komunikasi manusia. Ekspresi wajah mengacu pada komunikasi nonverbal yang sangat kuat [2]. Oleh karena itu ekspresi wajah manusia merupakan bagian terpenting manusia dalam proses komunikasi. Ada Tujuh ekspresi dasar manusia yaitu Bahagia, Sedih, Terkejut, Takut, Marah Muak dan Netral.



Gambar 2.1 Eskpresi Dasar Manusia

Ciri-ciri ekspresi wajah manusia pada Gambar apabila dideskripsikan sebagai berikut:

Tabel 2.1 Ciri-ciri Ekspresi Wajah

Emosi	Ciri-ciri ekspresi wajah
Bahagia	Terlihat kerutan pada bagian bawah mata dan mata menyipit, serta pada bagian mulut dan bibir melebar, kadang-kadang gigi terlihat.
Sedih	Pada bagian alis terangkat, kemudian dahi berkerut dan bagian mulut tertarik kebawah
Terkejut	Pada bagian mata membesar, dahi berkerut serta bagian alis terangkat secara keseluruhan dan bagian bibir ditarik atau sedikit terbuka.
Takut	Bagian kelopak mata bagian atas terangkat kemudian bagian putih mata terlihat jelas sedangkan kelopak mata bagian bawah menegang dan terangkat kemudian bagian dahi berkerut dan bagian bibir ditarik kebawah.
Marah	Pada Bagian alis ditarik ke dalam sedangkan mata menyipit kemudian bagian bibir tertutup rapat.
Muak	Pada bagian kelopak mata bagian bawah terangkat dan berkerut sedangkan mulut merapat dan kedua bibir terangkat.
Netral	Seluruh otot wajah dalam kondisi rileks kemudian kelopak mata bersinggungan dengan retina dan pada bagian bibir atas dan bawah saling bersentuhan dengan retina dan pada bagian bibit tertutup.

2.3 *Grayscale*

Citra *grayscale* adalah citra yang hanya memiliki 1 kanal sehingga yang ditampilkan hanyalah nilai intensitas atau dikenal juga dengan istilah derajat keabuan.

Jenis citra *grayscale* ini disebut juga sebagai 8-bit *image* karena untuk setiap nilai pikselnya memerlukan penyimpanan sebesar 8 bit [10]. Proses *grayscale* bertujuan untuk merubah citra berwarna menjadi citra berskala keabuan. Secara umum perubahan citra berwarna menjadi citra *grayscale* menggunakan rumus:

$$I = 0.2989\ x\ R + 0.5870\ x\ G + 0.1141\ x\ B \tag{2.1}$$

2.4 *Image Smoothing*

Image Smoothing atau penghalusan citra merupakan proses perbaikan kualitas citra. Metode yang digunakan pada penelitian kali ini adalah *Gaussian smoothing. Gaussian Smoothing* adalah metode yang menggunakan fungsi *gaussian* yang beroperasi dengan cara mengonvolusikan citra dengan kernel *gaussian* dengan ukuran tertentu dari pojok kiri atas sampai pojok kanan bawah citra [10]. Konvolusi adalah perkalian total dari dua buah fungsi f(x) dan g(x). Contoh konvolusi satu dimensi dapat didefinisikan sebagai berikut.

$$h(x) = f(x) * g(x) = \int_{-\infty}^{\infty} f(u)g(x-u)du = \tag{2.2}$$

Dalam hal ini, f adalah filter yang diterapkan, g adalah sinyal input, h adalah hasil konvolusi antara f dan g, x dan u adalah variabel independen yang memiliki nilai kontinu serta * menyatakan operator konvolusi. Sementara itu, operasi konvolusi dua dimensi dirumuskan dengan:

$$h(x,y) = f(x) * g(x) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(u,v)g(x-u,y-v)dudv = \tag{2.3}$$

Konvolusi dua dimensi inilah yang banyak digunakan pada operasi pengolahan citra. Operasi yang dilakukan untuk pengolahan citra, adalah operasi diskrit karena nilai koordinat *pixel* merupakan nilai yang diskrit. Kemudian, filter atau *mask* yang digunakan juga terbatas. Secara matematis filter rata-rata dapat dituliskan sebagai berikut

$$h(x,y) = \frac{1}{mn}\sum_{x=0}^{m-1}\sum_{y=0}^{n-1} \tag{2.4}$$

Keterangan:

h(x,y) = filter *h* (filter rata-rata)
n = jumlah baris pada filter *h* (filter rata-rata)
m = jumlah kolom pada filter *h* (filter rata-rata)
x = *koordinat letak citra pada titik x*
y = *koordinat letak citra pada titik y*

Operasi pada filter rata-rata yang akan digunakan adalah metode konvolusi yaitu perkalian fungsi diskrit antara citra *f(x,y)* dan filter *g(x,y)* (filter *h(x,y)* dimisalkan sebagai *g(x,y)*).

$$h(x,y) = f(x,y) \times g(x,y) \tag{2.5}$$

Pada filter *h(x,y)* yang akan digunakan adalah karnel berbasis 3×3 sebagai proses penghalusan citra dan disebut sebagai konvolusi dari *f(x,y)* dengan respon *h(x,y)*, berikut adalah karnel yang digunakan dalam proses perhalusan citra:

$$h(x,y) = 1/9 \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.6}$$

f(x,y) adalah piksel yang dikenai operasi beserta tetangganya, maka *h(x,y)* adalah hasil dari perhitungan adalah sebagai berikut:

$$\begin{aligned} h(x,y) = {} & (A \times P1) + (B \times P2) + (C \times P3) + (D \times P4) \\ & + (E \times P5) + (F \times P6) + (G \times P7) \\ & + (H \times P8) + (I \times P9) \end{aligned} \tag{2.7}$$

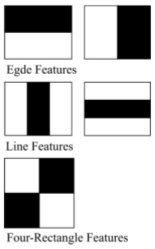
2.5 Segmentasi Citra

Segmentasi citra merupakan teknik membagi suatu citra ke wilayah berbeda yang lebih spesifik dengan mengkonversikan ke dalam bentuk matrix. Pada proses pengenalan wajah manusia, proses segmentasi dibutuhkan untuk memisahkan wajah

manusia terhadap *background* atau terhadap bagian tubuh yang lain sehingga didapatkan gambar wajah yang akan dikenali. Untuk proses pengenalan jenis objek, proses segmentasi diperlukan untuk pemisah masing-masing objek terhadap *background* sehingga pada saat proses pengenalan, bagian *backgound* tidak ikut terproses. Pada penelitian ini proses segmentasi digunakan untuk memisahkan bagian objek mata, objek hidung dan objek mulut dengan memanfaatkan fitur *Haar like Feature* pada metode *viola and jones* dan akan dibantu dengan metode *Thresholding.* Tahapan yang akan dilakukan adalah sebagai berikut:

2.5.1 *Haar Like Feature*

Haar Like Feature secara umum digunakan untuk mendeteksi objek *digital.* Istilah *Haar* menunjukkan suatu fungsi matematika (*Haar Wavelet*) yang berbentuk kotak, prinsipnya sama seperti pada fungsi fourier. *Haar Like Feature* memproses gambar dalam kotak-kotak, dimana dalam satu kotak terdapat beberapa pixel. Per kotak itupun kemudian diproses dan menghasilkan perbedaan nilai yang menandakan daerah gelap dan terang. Nilai-nilai inilah yang nantinya dijadikan dasar dalam pemrosesannya [11]. Dalam algoritma *Viola and Jones,* ada beberapa jenis fitur yang bisa digunakan seperti *Edge-feature, Line feature,* dan *Four-rectangle feature.* Beberapa contoh fitur tersebut adalah sebagai berikut :



Gambar 2.2 Variasi Fitur *Haar*

3 tipe kotak feature adalah:

1. Tipe *two-rectangle feature* (*horisontal/vertikal*) adalah perbedaan antara jumlah nilai piksel pada kotak terang dengan nilai piksel pada kotak gelap dan

untuk mendapatkan fitur dengan cara menjumlahkan nilai piksel pada kotak terang dan kotak gelap.

2. Tipe *three-rectangle feature* adalah dengan mempertimbangkan tiga nilai piksel pada kotak terang piksel pada kotak terang pertama dijumlahkan dengan nilai piksel pada kotak terang kedua dan kemudian ditambahkan dengan nilai piksel kotak gelap antara kotak terang kesatu dan kotak terang kedua.
3. Tipe *four-rectangle feature* adalah dengan mempertimbangkan nilai piksel kotak gelap sebesar 2x2 yang tersusun secara diagonal dan menjumlahkan seluruh nilai piksel kotak terang dan kotak hitam.

Pada proses pemilihan fitur *Haar,* fitur-fitur tersebut digunakan untuk mencari fitur wajah seperti mata, hidung, dan mulut pada citra. Pada setiap kotak-kotak fitur tersebut terdiri dari beberapa piksel dan akan dihitung selisih antara nilai piksel pada kotak terang dengan nilai piksel pada kotak gelap. Apabila nilai selisih antara daerah terang dengan daerah gelap di atas nilai ambang (*threshold*), maka daerah tersebut dinyatakan memiliki fitur. Dalam algoritma *viola and jones* nilai-nilai piksel yang didapat didalam kotak-kotak tersebut akan dilakukan perhitungan *integral image.*

2.5.2 *Integral Image*

Integral Image digunakan pada algoritma untuk pendeteksian objek dimana proses perhitungan dengan menggunakan intgral image memerlukan waktu yang singkat atau cepat dan hasil yang akurat. Proses untuk menghitung hasil penjumlahan nilai piksel pada daerah yang dideteksi oleh fitur haar. Nilai-nilai piksel yang akan dihitung adalah nilai-nilai piksel dari sebuah citra masukan yang dilalui oleh fitur haar pada saat pencarian fitur wajah. Pada setiap jenis fitur yang digunakan, pada setiap kotak-kotaknya terdiri dari beberapa piksel [12].

Dari nilai-nilai piksel yang didapatkan pada fitur tersebut, maka akan dihitung nilai *integral image* pada fitur tersebut dengan rumus sebagai berikut:

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x', y') \tag{2.8}$$

Dimana:

$ii(x,y)$ = integral image

$i(x', y')$ = citra asli

Kemudian selanjutnya adalah menghitung jumlah piksel pada daerah tertentu, adapun rumus untuk menghitung jumlah pikselnya adalah:

$$D = L1 + L4 - (L2 + L3) \tag{2.9}$$

2.5.3 *Adaptive Boosting (AdaBoost)*

AdaBoost merupakan *esemble learning* yang sering digunakan pada algoritma *boosting. Boosting* bisa dikombinasikan dengan *classifier* algoritma yang lain untuk meningkatkan performa klasifikasi. Tentunya secara intuitif, penggabungan beberapa model akan membantu jika model tersebut berbeda satu sama lain. *Adaptive Boosting* bertujuan untuk memperbaharui bobot dengan melakukan perhitungan menggunakan persamaan jumlah gambar negatif dan jumlah gambar positif. *Adaptive Boosting* mengkombinasikan *performance* banyak *weak classifier* untuk menghasilkan *strong classifier. Weak classifier* dalam hal ini adalah nilai dari *haar-like feature* [12].

Bobot awal = $wj_{1yi} = \frac{1}{2m}$, $wj_{1yi} = \frac{1}{2i}$ (2.10)

Untuk citra positif: $\in_t = \left(\sum_t^T w_{t,i}\right)|h_t(x) - (y_i)|$ (2.11)

Untuk citra negatif: $\in_j = \left(\sum_j^J w_{t,i}\right)|h_j(x) - (y_i)|$ (2.12)

Jika $\in t \vee \in j < 0$, hentikan iterasi.

Dimana:

w = *week classifier*

m = jumlah citra positif.

l = jumlah citra negatif.

t = indeks iterasi dari citra positif.

j = indeks iterasi dari citra negatif.

$h(x)$ = nilai fitur citra positif.

$h(x)$ = nilai fitur citra negatif.

Hasil akhir klasifikasi yang diharapkan pada citra positif adalah sebagai berikut:

$$H(x) = \begin{cases} 1 \sum_{j=1}^{j} a_j h_j \geq \frac{1}{2} \sum_{t=1}^{T} a_t \\ \\ 0 \qquad\qquad bukan\ objek \end{cases} \tag{2.13}$$

Dimana:

$$a_j = \log \frac{1}{\beta_j} \, , a_t = \log \frac{1}{\beta_t} \tag{2.14}$$

Kondisi:

Jika posisi $H(x)$ = Ketentuan 1 maka citra tersebut merupakan objek

Jika posisi $H(x)$ = Ketentuan 0 maka citra tersebut merupakan bukan objek

Keterangan:

$H(x)$ = Strong Classifier atau klasifikasi yang menyatakan objek atau bukan

a_j = Tingkat pembelajaran citra positif

a_t = Tingkat pembelajaran citra negatif

β_t = Nilai bobot setelah *error rate* pada citra positif

β_j = Nilai bobot setelah *error rate* pada citra negatif

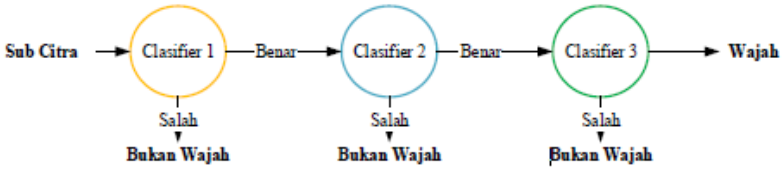
Ht = *week* atau *basic classifiers* (awal dari klasifikasi) citra positif

Hj = *week* atau *basic classifiers* (awal dari klasifikasi) citra negative

2.5.4 *Cascade Clasifier*

Pada proses *cascade classifier* yang merupakan metode untuk mengkombinasikan *classifier* yang kompleks dalam sebuah struktur yang bertingkat dan dapat meningkatkan kecepatan pendeteksian sebuah objek pada citra yang

mefokuskan pada daerah citra yang berpeluang saja, dan berikut adalah proses dari *cascade classifier* [11]:



Gambar 2.3 *Cascade Clasifier* Dengan *N Stages*

2.5.5 *Thresholding*

Proses *Tresholding* yaitu proses binerisasi pada 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 *treshold*, maka nilai piksel tersebut dikonversi menjadi 0.

Pada penelitian kali ini fungsi *THRESH_TOZERO_INV* pada pustaka *openCV*, fungsi ini menerapkan *tresholding* ke *single-channel array*. Fungsi ini biasanya digunakan untuk mendapatkan citra *bit-level* atau *binary* dari gambar *grayscale* untuk menghilangkan *noise*, yaitu dengan cara menyaring piksel yang memiliki nilai terlalu kecil atau terlalu besar. Adapun gambaran fungsi yang ada pada *THRESH_TOZERO_INV*:

$$dst(x,y) = \begin{cases} threshold & if\ src(x,y) > thresh \\ src(x,y) & otherwise \end{cases} \quad (2.15)$$

Dimana:

src = input array berupa *single-channel* antara 8-bit atau 32 bit.

dst = output array yang memiliki nilai yang sama dengan src.

tresh = nilai tresholding.

2.5.6 *Resize Image*

Resize Image atau penskalaan merupakan proses merubah ukuran citra menjadi lebih kecil. Pada proses *resize* atau pengskalaan pada penelitian ini tidak menggunakan metode khusus. Proses untuk melakukan *resize* pada penelitian ini yaitu dengan cara membandingkan ukuran citra dari segmentasi dengan target ukuran citra. *Resize Image* perlu dilakukan agar pada saat pemrosesan citra oleh komputer akan lebih cepat dan tidak banyak menghabiskan memori penyimpanan di dalam memori sementara. Skala *resize* yang akan diterapkan pada penelitian kali ini sebesar 222x222 dari ukuran citra aslinya. Rumus yang digunakan untuk merubah ukuran citra menjadi lebih kecil adalah:

$$x = \frac{pb * pp}{pa} \tag{2.16}$$

Keterangan:

x = Posisi x baru

pb = Ukuran panjang dari matriks baru.

pp = Posisi piksel x lama.

pa = Ukuran panjang dari matriks lama.

$$y = \frac{lb * pp}{la} \tag{2.17}$$

Keterangan:

y = Posisi y baru

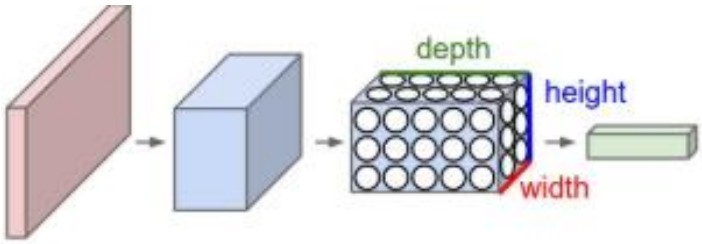
lb = Ukuran lebar dari matriks baru.

pp = Posisi piksel y lama.

la = Ukuran lebar dari matriks lama.

2.6 *Convolutional Neural Network*

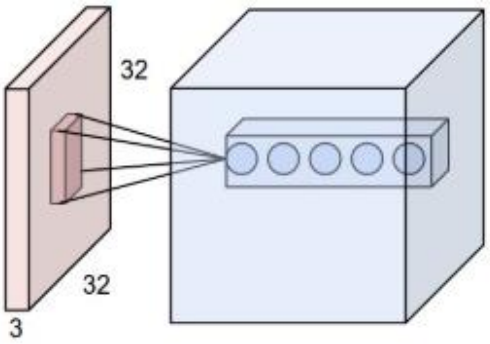
Convolutional Neural Networks, adalah salah satu kelas *Deep Learning, Deep Learning* adalah bagian dari *Artificial Neural Network* [13], yang banyak diaplikasikan pada analisis citra. CNN terdiri atas satu lapis masukan (*input layer*), suatu lapis keluaran (*Output Layer*), dan sejumlah lapis tersembunyi (*hidden layer*). Lapis tersembunyi umumnya berisi *convolutional layers, pooling layers, normalization layers, ReLu layer, fully connected layers,* dan *loss layer.* Semua lapisan tersebut disusun secara bertumpuk-tumpuk. Setiap lapisan CNN mentransformasikan volume masukan tiga dimensi (3D *input volume*) ke dalam volume keluaran tiga dimensi aktivasi-aktivasi sel saraf (3D *output volume of neuron activations*). Masukan berupa citra berwarna, di mana lebar dan tinggi menyatakan dimensi citra tersebut sedangkan dalam (*depth*) nya adalah 3 yang menyatakan kanal *Red, Green, Blue* (RGB) [7].



Gambar 2.4 Arsitektur CNN Secara Umum

2.6.1 *Convolutional Layer*

Convolutional layer terdiri dari *neuron* yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi. Sebagai contoh misalkan volume input memiliki ukuran 32 x 32 x 3. Jika bidang reseptif atau ukuran filter adalah 5 x 5, maka setiap neuron pada lapisan *convolutional* akan memiliki bobot ke wilayah 5 x 5 x 3 = 75 bobot dan +1 parameter bias.



Gambar 2.5 *Convolutional Layer*

CNN pada umumnya menggunakan lebar langkah atau *stride* dengan *zero padding* sebesar

$$P = \frac{(f-1)}{2} \tag{2.18}$$

Di mana P adalah ukuran *padding* dan F adalah ukuran bidang reseptif atau tingkat spasial yang tentu saja sama dengan ukuran filter. Konvolusi pada dasarnya hanya berupa *dot product* antara filter dengan sebuah bidang reseptif kecil pada citra masukan yang berukuran sama dengan ukuran filter, dimana filter dilambangkan $k \times k$. Adapun persamaannya adalah sebagai berikut [15]:

$$Convj_{x,y} = \left(I \otimes C + B_j\right), \tag{2.19}$$

$$dimana \otimes = \sum_{u=1}^{3}\sum_{v=1}^{3} I_{x+u,y+v} * C_{u,v} \tag{2.20}$$

Keterangan:

$I_{x,y}$ = Nilai pixel dari input ke-x,y.

$C_{u,v}$ = Nilai pixel dari filter ke-u,v.

B_j = Nilai bias ke-j.

Persamaan 2.20 hanya berlaku untuk operasi konvolusi pada layer pertama saja, untuk melakukan operasi konvolusi ke tahap layer selanjutnya persamaan yang digunakan akan sedikit berbeda, dimana inputan dari layer sebelumnya sebanyak

jumlah filter masing-masing akan dikonvolusi dan kemudian akan dijumlahkan. Persamaan tersebut adalah sebagai berikut

$$Convj = \sum_{i=1}^{filter} pool_i \otimes C_{i,j} + B_j \tag{2.21}$$

Keterangan:

$Convj$ = Layer konvolusi ke-j.

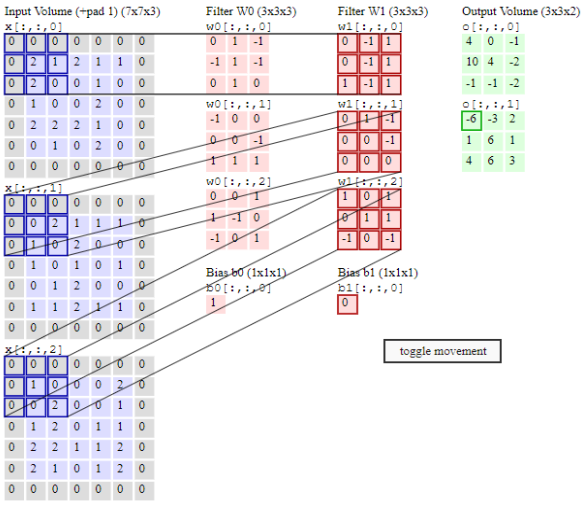
$pool_i$ = Layer *pooling* ke-i.

$C_{i,j}$ = Kernel index ke-j dan filter ke-i.

Bj = Bias ke-j.

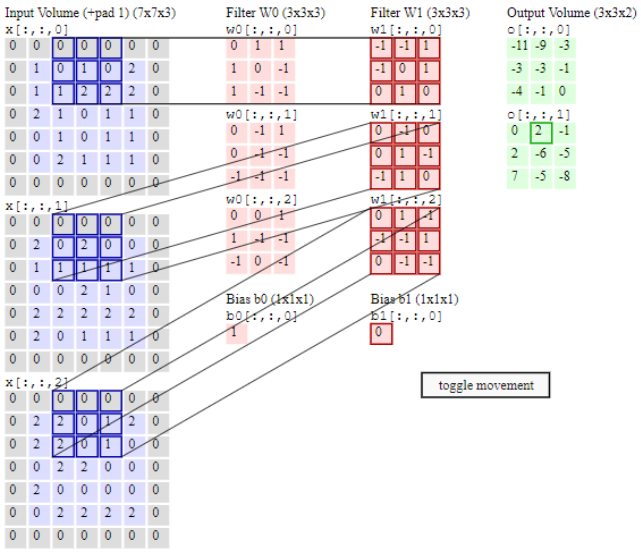
$\otimes$ = Operasi konvolusi [8].

Adapun ilustrasi dari proses konvolusi telah tergambarkan pada Gambar 2.6, 2.7 dan 2.8.



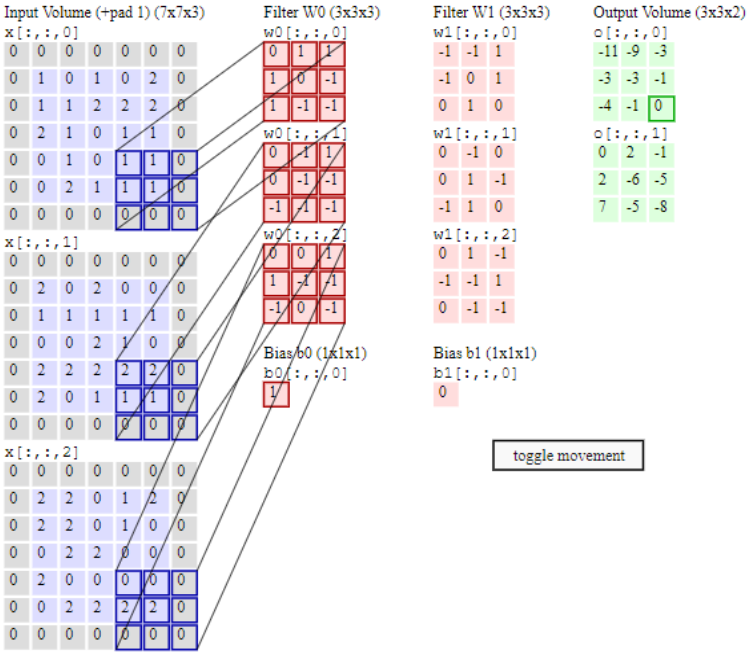
Gambar 2.6 Proses Konvolusi Pada Bagian Kiri Atas Citra Masukan

Selanjutnya bidang reseptive digeser dua piksel ke kanan yang diilustrasikan pada Gambar 2.7.



Gambar 2.7 Proses Konvolusi Setelah Pergeseran Dua Piksel Ke Kanan

Proses konvolusi terus dilanjutkan dengan menggeser bidang reseptif dua piksel ke kanan sampai pada bagian paling kanan bawah citra masukan tersebut.

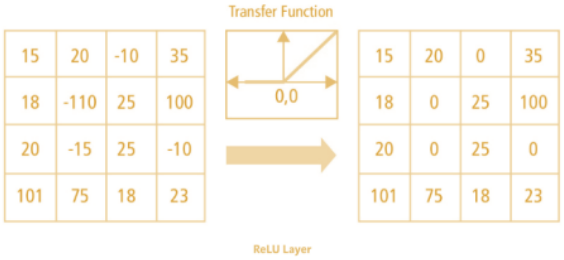


Gambar 2.8 Proses Konvolusi Setelah Pergeseran

2.6.2 *ReLu Layer*

Rectified Linear Units (*ReLU*) layer. Lapisan ini mengaplikasikan fungsi aktivasi $f(x) = \max(0, x)$. dimana nilai piksel yang memiliki nilai < 0 akan dirubah menjadi 0, fungsi ini meningkatkan sifat nonlinearitas fungsi keputusan dan jaringan secara keseluruhan tanpa mempengaruhi bidang-bidang pada *convolutional layer*. Kenapa ReLu sangat penting? Karena tujuan ReLu adalah untuk memperkenalkan nonlinearitas kepada CNN. Karena data pada dunia nyata ingin agar CNN mempelajari nilai-nilai nonnegatif. Adapun rumus pada ReLu layer adalah [14]:

$$f(x) = ReLU(x) = \begin{cases} Cj_{x,y} & jika\ (Cj_{x,y} \geq 0) \\ 0 & jika\ tidak \end{cases} \quad (2.22)$$



Gambar 2.9 Operasi ReLu

2.6.3 *Pooling Layer*

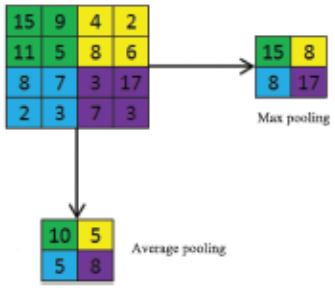
Pooling Layer berfungsi menjaga ukuran data ketika *convolution,* yaitu dengan melakukan *downsampling* atau pereduksian sampel. Dengan *pooling* kita dapat merepresentasikan data menjadi lebih kecil, mudah dikelola, dan mudah mengontrol *overfitting.* Proses *pooling* yang umum digunakan adalah *max pooling,* yaitu memilih nilai maksimum dalam suatu area tertentu, seperti diilustrasikan pada Gambar 6. Dari empat nilai di bagian kiri atas akan dihasilkan satu nilai maksimum yaitu 100, untuk mengisi data hasil *pooling.* Hal yang sama dilakukan untuk empat nilai di bagian kanan atas, kanan bawah dan kiri bawah. Teknik *max pooling* lebih praktis digunakan dan memberikan perfromansi lebih baik dibanding *average pooling* dan *L2-norm pooling.* Pada penelitian kali ini fungsi *Max Pooling* akan digunakan, adapun rumus fungsi *Max pooling* adalah sebagai berikut:

$$Pool_{x,y} = Max\left(Conv_{x,y}, Conv_{x+1,y}, Conv_{x,y+1}, Conv_{x+1,y+1}\right) \quad (2.23)$$

Keterangan:

$Pool_{x,y}$ = Hasil dari *pooling layer*.

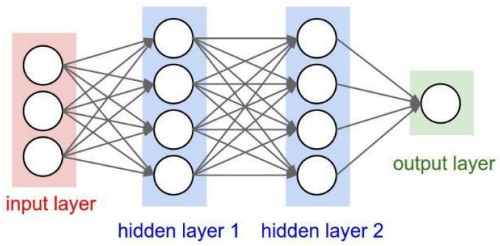
$Conv_{x,y}$ = Nilai pixel dari hasil *convolutional layer*.



Gambar 2.10 Teknik *Max Pool* Dan *Average Pool*

2.6.4 *Fully Connected Layer*

Pada lapisan yang terhubung secara penuh (*fully connected layer*), setiap *neurons* memiliki koneksi penuh ke semua aktivasi dalam lapisan sebelumnya. Hal ini sama persis dengan yang ada pada MLP. Model aktivasinya pun sama persis dengan MLP, yaitu komputasi menggunakan suatu perkalian matriks yang diikuti dengan bias *offset.* Sesuai dengan namanya, *Multi Layer Perceptron* memiliki lapisan ganda yang memiliki beberapa *hidden layer, activation function* dan *output layer*.



Gambar 2.11 Arsitektur MLP Secara Umum

Fully connected layer berperan untuk mengklasifikasi data masukan. Output yang dihasilkan dari *pooling layer* masih berbentuk *multidimensional array*, sehingga akan di *flatten* terlebih dahulu untuk mengubah data menjadi vektor sebelum input untuk *fully connected layer* [17]. Adapun persamaan yang akan digunakan pada tahapan ini adalah sebagai berikut:

$$Fully_i = \sum_{j}^{Length(flatten)} W_{i,j} * flatten_j + b_i \quad (2.24)$$

Keterangan:

$Fully_i$ = Hasil dari perhitungan pada *fully-connected layer*.

$W_{i,j}$ = Nilai bobot yang digunakan dari hasil *convolutional layer*.

$flatten_j$ = Nilai dari vektor ke-j.

i = Kelas ke-i ($i = 1,2,3,4,5,6,7$).

Kemudian hasil dari $Fully$ akan diaktifasi dengan menggunakan fungsi *softmax*, digunakan fungsi *softmax* karena pada penilitian ini menghasilkan *output* atau jumlah kelas lebih dari dua [15], atau biasa disebut dengan *multi-class*. Tujuan dari fungsi *softmax* sendiri adalah agar dapat diketahui prediksi yang dihasilkan dari arsitektur yang dibangun, dimana rumus untuk fungsi *softmax* ditulis pada persamaan 2.25.

$$\widehat{y_i} = \frac{e^{Fully_i}}{(\sum_{j=1}^{kelas} e^{Fully_j})} \quad (2.25)$$

Keterangan:

$\widehat{y_i}$ = Hasil dari aktifasi fungsi *softmax*.

$Fully_i$ = Hasil dari perhitungan pada *fully-connected layer* ke-i.

$flatten_j$ = Nilai dari vektor ke-j.

2.6.5 *Cross-Entropy Loss Function*

Selanjutnya akan dicari nilai *error* yang didapat dari hasil prediksi pada *fully-connected layer* sebelumnya, dimana nilai *error* ini digunakan untuk menentukan apakah hasil prediksi dari CNN sudah mencapai target atau belum, maka dari itu akan dilakukan perbandingan antara *error* dengan *target error* dimana, apabila *error* masih dibawah dari target, maka akan dilakukan *loss function*. Dari beberapa jenis *loss function* yang ada, salah satu-nya adalah *Cross-entropy Loss Function*, dimana rumus dari *cross-entropy* tersebut dituliskan pada persamaan 2.26.

$$Loss = -\sum_{i}^{kelas} t_i * Log(\hat{y}) \tag{2.26}$$

Keterangan:

$\hat{y}_i$ = Hasil dari aktifasi fungsi *softmax*.

t_i = Nilai dari target, bernilai 1 apabila target adalah kelas yang dituju, dan bernilai 0 apabila nilai target bukanlah kelas yang dituju.

$flatten_j$ = Nilai dari vektor ke-j.

i = Kelas ke-i ($i = 1,2,3,4,5,6,7$).

2.6.6 *Backpropagation*

Backpropagation adalah salah satu algoritma *supervised learneing* yang digunakan dalam *artificial neural networks. Backpropagation* mencari kombinasi bobot untuk meminimalkan kesalahan output untuk dianggap menjadi solusi yang benar [18]. Adapun tahapan *Backpropagation* adalah sebagai berikut:

1. Turunan gradien error terhadap *softmax*

   Tahapan turunan gradien *error* terhadap *softmax* berdasarkan rumus yang ditulis oleh Peter Sadowski [18].

$$\Delta\hat{y}_i = \frac{\partial Loss}{\partial \hat{y}_i} = \hat{y}_i - t_i \tag{2.27}$$

Keterangan:

$\Delta\hat{y}_i$ = Nilai turunan dari fungsi *softmax*.

t_i = Nilai dari target, bernilai 1 apabila target adalah kelas yang dituju, dan bernilai 0 apabila nilai target bukanlah kelas yang dituju.

i = Kelas ke-i ($i = 1,2,3,4,5,6,7$).

2. Turunan gradien error terhadap bobot

   Tahapan turunan gradien *error* terhadap bobot berdasarkan rumus yang ditulis oleh Zhifei Zhang [15].

$$\Delta W_{i,j} = \frac{\partial Loss}{\partial W_{i,j}} = \Delta\hat{y}_i * flatten_j \tag{2.28}$$

3. Turunan gradien error terhadap bias

   Tahapan turunan gradien *error* terhadap bobot berdasarkan rumus yang ditulis oleh Zhifei Zhang [15].

$$\Delta b_i = \frac{\partial Loss}{\partial b_i} = \Delta\hat{y}_i \tag{2.29}$$

2.6.7 *Stocastic Gradient Descent*

Stochastic Gradient Descent adalah sebuah algoritma untuk menemukan nilai minimum lokal dari sebuah fungsi, Algoritma dari *Stochastic Gradient Descent* dapat dilihat pada Gambar 2.12.

function STOCHASTIC GRADIENT DESCENT($L()$, $f()$, x, y) **returns** θ
    # where: L is the loss function
    # f is a function parameterized by θ
    # x is the set of training inputs $x^{(1)}, x^{(2)}, ..., x^{(n)}$
    # y is the set of training outputs (labels) $y^{(1)}, y^{(2)}, ..., y^{(n)}$

$\theta \leftarrow 0$
repeat T times
  For each training tuple $(x^{(i)}, y^{(i)})$ (in random order)
  Compute $\hat{y}^{(i)} = f(x^{(i)}; \theta)$ # What is our estimated output $\hat{y}$?
  Compute the loss $L(\hat{y}^{(i)}, y^{(i)})$ # How far off is $\hat{y}^{(i)}$ from the true output $y^{(i)}$?
  $g \leftarrow \nabla_\theta L(f(x^{(i)}; \theta), y^{(i)})$ # How should we move θ to maximize loss ?
  $\theta \leftarrow \theta - \eta\, g$ # go the other way instead
return θ

Gambar 2.12 Algoritma Stocastic Gradient Descent

Dimana dari algoritma tersebut, bisa juga dirumuskan nilai perbaikan bobot dan bias baru seperti pada persamaan berikut:

1. Perbaikan nilai pada bobot

$$\theta W = W - \alpha(\Delta W) \quad (2.30)$$

Keterangan:

ΔW = Nilai turunan dari bobot.

W = Nilai bobot.

α = Nilai *learning rate.*

θW = Nilai bobot baru.

2. Perbaikan nilai pada bias

$$\theta b = b - \alpha(\Delta b) \quad (2.31)$$

Keterangan:

Δb = Nilai turunan dari bias.

b = Nilai bias.

α = Nilai *learning rate.*

θb : Nilai bias baru.

2.7 *Unified Modelling Language (UML) 2.0*

Diagram UML adalah sekumpulan alat yang digunakan untuk melakukan abstraksi terhadap sebuah sistem atau perangkat lunak berbasis objek. UML merupakan singkatan dari Unified Modeling Language. Dalam UML sendiri terdapat beberapa diagram yang wajib dikuasai yaitu:

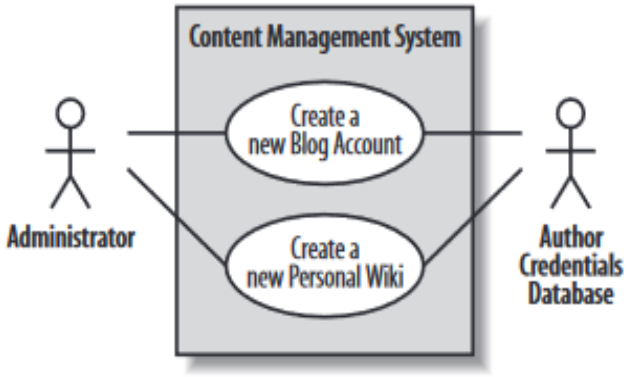
1. *Structural Diagram*
    a. *Class Diagram* ini terdiri dari *class*, *interface*, *association*, dan *collaboration*. Diagram ini menggambarkan objek - objek yang ada di sistem.

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.

2.7.1 *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 [19]. Contoh sederhana use case dapat dilihat pada Gambar 2.13.



Gambar 2.13 Contoh *Use Case*

2.7.2 *Use Case Scenario*

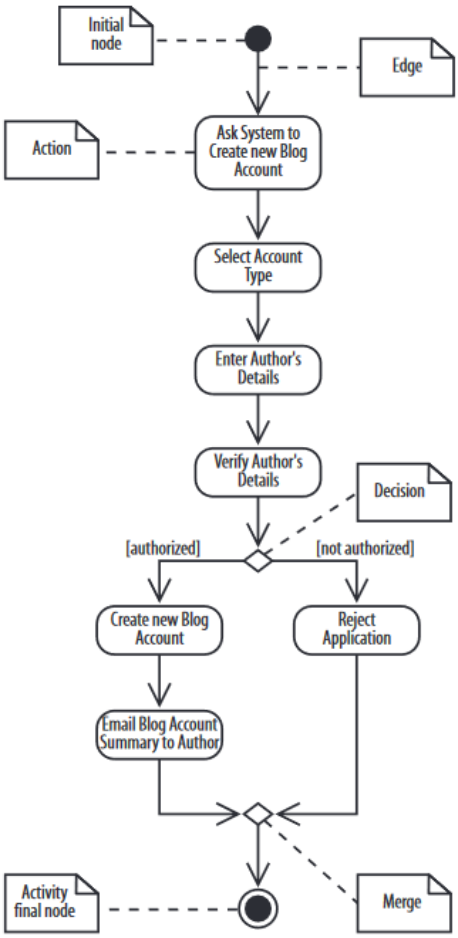
Use Case scenario adalah penjelasan secara tekstual dari sekumpulan skenario interaksi. Setiap skenario mendeskripsikan urutan aksi/langkah yang dilakukan aktor ketika berinteraksi dengan sistem, baik yang berhasil maupun gagal [19]. Kegunaan dari *use case scenario* sendiri adalah untuk memperjelas proses yang ada di dalam masing-masing *use case.* Contoh dari *use case scenario* dapat dilihat pada Gambar 2.14.

Use case name	**Create a new Personal Wiki**	
Related Requirements	Requirement A.2.	
Goal In Context	A new or existing author requests a new personal Wiki from the Administrator.	
Preconditions	The author has appropriate proof of identity.	
Successful End Condition	A new personal Wiki is created for the author.	
Failed End Condition	The application for a new personal Wiki is rejected.	
Primary Actors	Administrator.	
Secondary Actors	Author Credentials Database.	
Trigger	The Administrator asks the CMS to create a new personal Wiki.	
Main Flow	**Step**	**Action**
	1	The Administrator asks the system to create a new personal Wiki.
	2	The Administrator enters the author's details.
	3	The author's details are verified using the Author Credentials Database.
	4	The new personal Wiki is created.
	5	A summary of the new personal Wiki's details are emailed to the author.
Extensions	**Step**	**Branching Action**
	3.1	The Author Credentials Database does not verify the author's details.
	3.2	The author's new personal Wiki application is rejected.

Gambar 2.14 Contoh *Use Case Scenario*

2.7.3 *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. Contoh dari *activity* diagram telah tergambarkan pada Gambar 2.15.

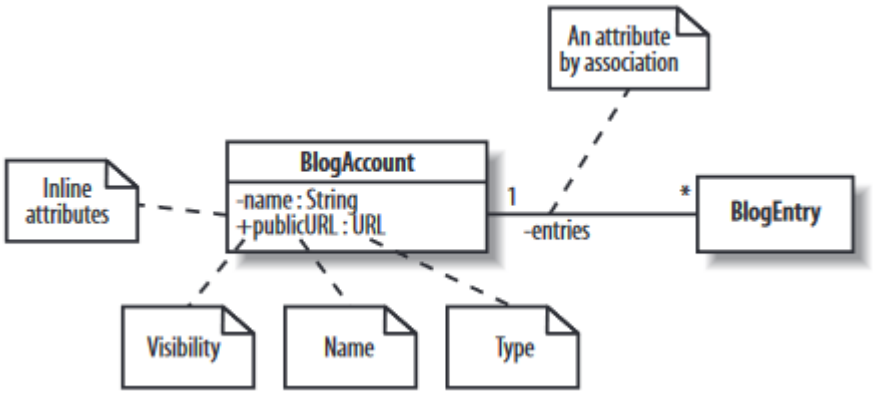


Gambar 2.15 Contoh Activity Diagram

2.7.4 *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

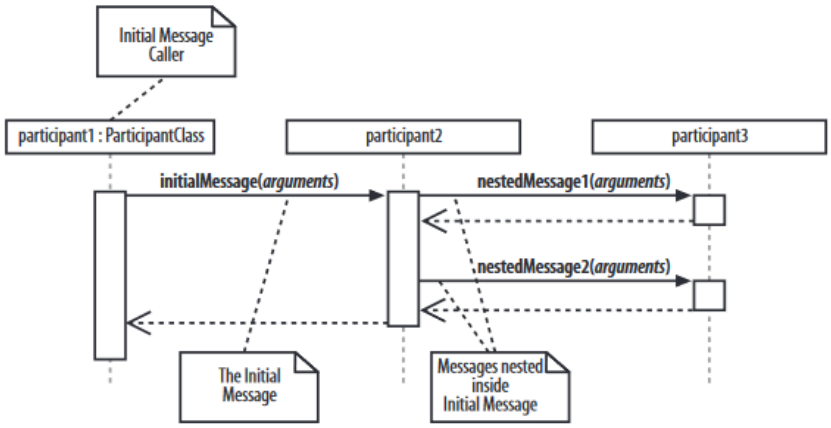
sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem [20]. Adapun contoh dari *Class* diagram telah tergambarkan padaGambar 2.16.



Gambar 2.16 Contoh Dari *Class Diagram*

2.7.5 *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 [20]. Adapun contoh dari *Sequence* diagram telah tergambarkan pada Gambar 2.17.



Gambar 2.17 Contoh *Sequence Diagram*

2.8 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.

2.9 PyQt

PyQt adalah sebuah *library* pada *python* untuk membangun *Graphic User Interface* (GUI) dengan lebih mudah. Di dalam PyQt sudah terdapat paket bernama QtDesigner untuk mempermudah pembangunan GUI hanya dengan *drag & drop* desain saja. PyQt sendiri sudah memiliki lebih dari 35 ekstensi modul yang siap digunakan dan mampu membuat *Python* sebagai bahasa alternatif dari C++. Dalam penggunaannya yang harus dilakukan adalah dengan memanggil *function* dengan meng-*import library* seperti berikut:

```
from PyQt5 import QtCore, QtGui, QtWidgets
```

2.10 OpenCV

OpenCV 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. Dalam penggunaannya yang harus dilakukan adalah dengan meng-*import library* seperti berikut:

```
import cv2
```

2.11 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. Untuk menggunakan *library* ini hal yang harus dilakukan adalah dengan meng-*import library* seperti berikut:

```
import numpy
```

2.12 NumpyCNN

NumpyCNN adalah sebuah *library* pada *Python* yang berdiri dari *library* Numpy. Kegunaan dari NumpyCNN ini adalah untuk melakukan perhitungan pada layer-layer yang ada pada CNN, seperti *convolutional layer*, *relu*, dan *max pooling*. Untuk menggunakan *library* ini hal yang harus dilakukan adalah dengan memanggil *library* tersebut dengan meng-*import* seperti berikut:

```
import numpycnn
```

2.13 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. Untuk menyimpan sebuah array ke file Npy dan Npz hal *code* yang dibuat dalam penelitian ini adalah seperti berikut.

```
np.savez("bobot/filter.npz", f1=C1_filter, f2=C2_filter, f3=C3_filter,
f4=C4_filter, f5=C5_filter)
np.save("bobot/weight.npy", weight)
np.save("bobot/bias.npy", bias)
```

2.14 PyCharm

PyCharm adalah sebuah IDE yang secara spesifik digunakan untuk penulisan kode dari bahasa pemrograman *Python*. Fitur-fitur di dalamnya sudah cukup banyak, mulai dari *auto-complete code, coding assistance and analysis, syntax and error highlighting*, dan banyak lagi. Sejarahnya PyCharm dibangun oleh Czech Company JetBrains, versi beta pertama muncul pada juli tahun 2010. Kini PyCharm sudah memilih 2 versi, yaitu PyCharm Professional Edition yang membutuhkan lisensi untuk menggunakannya dan PyCharm Community Edition yang sebagai *open-source*-nya.

2.15 Metode Pengujian *Confusion Matrix*

Confusion Matrix merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode klasifikasi. Pada dasarnya *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan klasifikasi yang seharusnya [21]. Pada pengukuran kinerja menggunakan *confusion matrix* , terdapat 4 istilah sebagai representasi hasil proses klasifikasi. Keempat istilah ini adalah *True Positive* (TP),*True Negative* (TN), *False Positive* (FP) dan *False Negative* (FN).

1. *True Positive* (TP) merupakan data positif yang terdeteksi dengan benar
2. *False Negative* (FN) merupkan kebalikan dari *True Positive*, yaitu data positif namun yang terdeteksi sebagai data negative
3. Setiap kolom dalam matriks merepresentasikan kelas yang diprediksi, Sedangkan setiap baris merepresentasikan kelas yang sebenarnya.

Adapun pengujian *confusion matrix* yang digunakan pada penelitian ini adalah sebagai berikut:

Tabel 2.2 Contoh *Confusion Matrix*

Kelas Sebenarnya	Hasil Prediksi						
	Bahagia	Marah	Muak	Netral	Sedih	Takut	Terkejut
Bahagia	TP	FN	FN	FN	FN	FN	FN
Marah	FN	TP	FN	FN	FN	FN	FN
Muak	FN	FN	TP	FN	FN	FN	FN
Netral	FN	FN	FN	TP	FN	FN	FN
Sedih	FN	FN	FN	FN	TP	FN	FN
Takut	FN	FN	FN	FN	FN	TP	FN
Terkejut	FN	FN	FN	FN	FN	FN	TP

Kemudian persamaan yang akan digunakan untuk menguji metode klasifikasi yang digunakan pada penelitian ini adalah sebagai berikut:

$$Akurasi = \frac{TP}{(TP + FN_1 + \cdots + FN_7} * 100\% \quad (2.34)$$