BAB 2
LANDASAN TEORI

2.1 Kepribadian

Pada umumnya kepribadian adalah interaksi antara individu yang satu dengan individu yang lain atau interaksi antara individu dan lingkungannya. Dan setiap individu akan memiliki sikap, ekspresi dan perasaan yang berbeda terhadap interaksi tersebut. Kepribadian merupakan suatu yang terorganisasi dan terpola, namun kepribadian bukanlah sebuah organisasi yang statis, tetapi selalu tumbuh dan berubah. Berdasarkan Pervin dan John kepribadian mewakili karakteristik dari individu yang terdiri dari pola pikiran, perasaan dan perilaku yang konsisten. Karakteristik kepribadian tersebut didapat dari apa yang diterima dari lingkungan setiap individu. Contohnya adalah karakteristik yang didapat dari keluarga dan karakteristik lainnya yang didapat ketika masih kecil. Oleh karena itu kepribadian adalah campuran dari hal – hal yang bersifat psikologis, kejiwaan dan bersifat fisik [6].

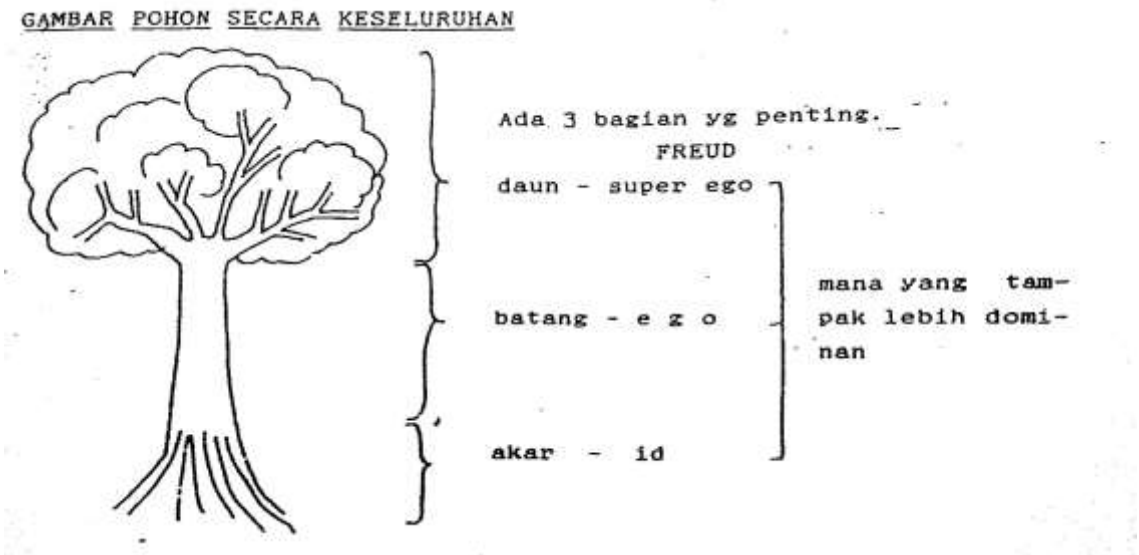
Menurut Sigmund Freud pembentukan kepribadian dalam diri manusia terdiri dari tiga komponen utama yaitu *das es*, *das ich*, dan *das uber ich*, atau dalam arti lainnya yaitu *id*, *ego*, dan *superego* [6].

Id adalah kepribadian asli yang dimiliki ketika manusia dilahirkan. *Id* dikendalikan oleh prinsip kenikmatan dan proses primer. *Id* berkembang dimulai pada usia bayi, yang sudah dimiliki sejak dilahirkan dan memiliki dorongan dasar seperti kebutuhan makan, minum, eliminasi, menghindari rasa sakit dan memperoleh kenikmatan. *Id* adalah kondisi bawah sadar yang didalamnya terdapat naluri bawaan dan keinginan yang tidak sesuai dengan norma yang ada. *Id* akan menuntut untuk segera dipuaskan dan termasuk dalam tindakan refleks dan proses berpikir primer [6].

Ego erat hubungannya dengan dunia realitas yang ada diluar dirinya. *Ego* sudah dimiliki semenjak lahir, tetapi berkembang dengan hubungannya antara dirinya dengan lingkungan sekitarnya. *Ego* bertindak untuk mengatur, memerintah dan mengendalikan kepribadian, yang mengontrol jalannya *id*, *superego*, dan dunia luar. *Ego* berada pada tingkat prasadar dan menjalankan fungsi dengan proses berpikir sekunder (rasional) [6].

Superego memiliki aspek keadilan terhadap kedua sistem kepribadian yang mengevaluasi nilai – nilai atau aturan – aturan yang baik atau buruk. *Superego* berkembang dari usia 4 – 6 tahun. *Superego* terdiri dari hati nurani dan ego ideal, yang sesuai dengan norma moral masyarakat. *Superego* memiliki fungsi sebagai pengendalian *id*, mengarahkan ego pada tujuan yang sesuai dengan moral daripada kenyataan, dan mendorong individu ke arah kebenaran [6].

Tes Baum atau lebih dikenal dengan *tree test* merupakan salah satu tes untuk mengenali kepribadian yang ditemukan oleh Emil Jucker dan dikembangkan oleh Karl Koch. Hal ini dapat dilihat melalui bentuk gambar, kelengkapan gambar, kerapihan, dan aspek lainnya. Tiga bagian pada pohon diantaranya bagian mahkota (daun), batang dan akar yang masing masing menggambarkan komponen dari kepribadian, yang dapat dilihat pada gambar 2.1.

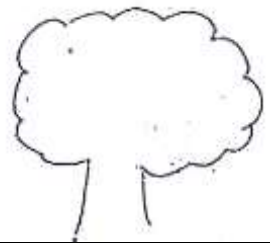
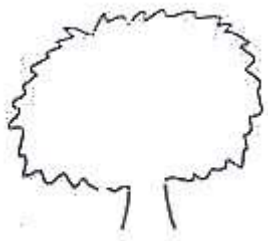


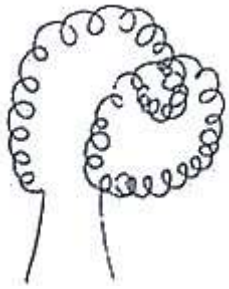
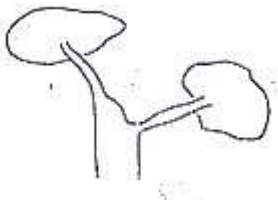
Gambar 2. 1 Teori Freud pada Gambar Pohon

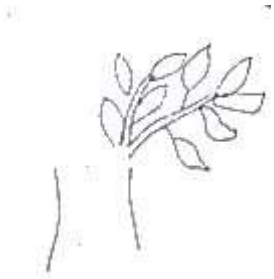
Kecenderunngan dari daun (mahkota) yaitu super ego yang berkuasa, intelektuil, ide – ide, fantasi dan norma – norma. Kecenderungan pada batang yaitu prinsip realita, mengakui yang nampak nyata (didominir). Sedangkan pada akar kecenderungannya adalah id dan drive yang berkuasa.

Pada penelitian ini penulis hanya akan menggunakan 3 fitur yaitu bagian mahkota (daun), batang dan buah. Tidak digunakannya fitur akar dikarenakan pohon yang tidak disertai akar adalah hal yang normal. Terdapat 37 fitur pada bagian mahkota (daun), 8 fitur pada batang dan 1 fitur pada buah. Namun, berdasarkan dataset yang didapat penulis maka hanya didapatkan 8 kelas fitur pada mahkota, 3 kelas fitur pada batang dan 1 fitur pada buah, yang sudah mewakili dan yang paling umum berdasarkan dataset yang didapat. Berikut adalah tabel fitur – fitur yang digunakan diantaranya.

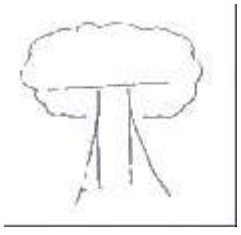
Tabel 2. 1 Tabel Kepribadian berdasarkan Fitur Mahkota Pohon

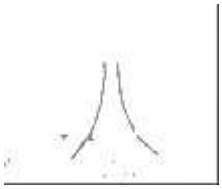
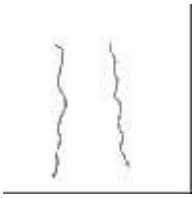
No	Gambar	Fitur	Kepribadian
1		Seperti awan / berombak	- Cenderung menutup diri - Mempunyai suasana hati yang hidup - Menyenangkan - Lemah - Mudah bergaul
2		Mahkota yang digambarkan bergetar	- Mudah grogi - Mudah terganggu perasaannya - Mudah ragu ragu - Mudah takut

3		Mahkota terlalu ruwer	- Jiwa tidak bergolak - Tidak aturan - Tak mempunyai kemauan - Pikiran kacau - Kurang sistematis - Tidak stabil - Konsentrasi juga kurang - Tidak senang bergantung - Mempunyai kecerdasan tinggi / psikopat - Retardasi
4		Mahkota yang keriting	- Vitalitas yang cukup - Dorongan yang cukup - Cepat menyesuaikan diri - Cenderung suka humor yang tidak realistik - Lebih mengutamakan hal hal lahiriah - Sering menonjolkan diri - Mudah untuk improvisasi
5		Mahkota yang tersebar	- Cukup dapat memisahkan antara rasio dan emosi - Takut akan realistas - Kurang prinsip - Pendirian mudah berubah - Selalu menyembunyikan sesuatu - Kurang dapat bertindak agresif pada saat tertentu
6		Mahkota yang centripental (banyak lingkaran dalam)	- Tendesi konsentrasi yang baik - Cepat dalam mengambil keputusan - Mempunyai satu tujuan yang pasti - Keadaan diri yang tertutup - Tabah dan ulet - Sukar kontak atau cenderung menolak - Sukar dipengaruhi - Kemampuan berdiri sendiri

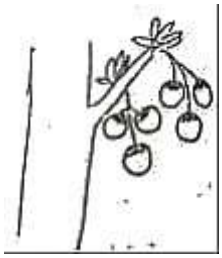
7		Mahkota seperti benang kusut	- Fleksibel dalam hidup - Dorongan kuat tetapi tak diimabngi kemauan - Ada keinginan untuk berproduksi banyak - Dalam orientasi kurang baik. Sehinngga mudah menimbulkan kesalahpahaman dalam penyesuaian - Konsentrasi yang lemah
8		Mahkota yang digambar dengan daun yang nyata	- Berbakat dekoratif - Tajam dalam pengamatan - Senang hal yang lahiriah - Cari perhatian - Suka dipuji - Kurang riil dalam menghadapi sesuatu - Sering suka menyenangkan hati orang lain - Pergaulah lincah dan kekanak kanakan.

Tabel 2. 2 Tabel Kepribadian berdasarkan Fitur Batang Pohon

No	Gambar	Fitur	Kepribadian
1		Batang berbentuk T	- Kurang cerdas - Cenderung dikendalikan (segi naluri) - Vitalitas kuat

2		Batang kerucut	- Konkrit dalam menghadapi sesuatu - Cenderung statis - Gejala retardasi - Ada kemungkinan lambat dalam belajar - Lebih praktis tapi sangat teoritis
3		Batang berbelok belok	- *Levendig*, hidup, lincah - Dinamis - Mudah menyesuaikan diri - Mudah terpengaruh - Diplomatis

Tabel 2. 3 Tabel Kepribadian berdasarkan Fitur Buah

No	Gambar	Fitur	Kepribadian
1		Pohon Berbuah	- Tajam dalam pengamatan - Sombong - Mudah mendemonstrasikan suatu kemampuannya - Implusif dalam keputusannya - Sering membesarkan realita - Regresi ke arah pubertas - Ingin lekas mencapai tujuan - Kurang riil dalam menghadapi masalah - Butuh sanjungan - Suka melanggar peraturan - Sering membesar besarkan kenyataan

2.2 Resize

Resizing merupakan suatu proses untuk mengubah ukuran suatu objek menjadi lebih besar atau lebih kecil. sedangkan pada citra, resizing berarti mengubah ukuran tinggi atau lebar dari suatu citra menjadi lebih besar atau lebih kecil. Untuk mengubah ukuran dari sebuah citra, metode scaling menggunakan fungsi interpolasi. Pada penskalaan apabila variabel bernilai lebih besar dari 1, maka ukuran citra diperbesar. Dan apabila variabel bernilai kurang dari 1 maka ukuran citra diperkecil. Berikut rumus penskalaan :

$$x = \frac{pb * pp}{pa} \tag{2.1}$$

Keterangan :

x = Nilai *pixel* baris baru

pb = Ukuran panjang matriks baru

pp = Posisi *pixel* baris

pa = Ukuran panjang matriks lama

$$y = \frac{lb * lp}{la} \tag{2.2}$$

Keterangan :

y = Nilai *pixel* kolom baru

lb = Ukuran lebar matriks baru

lp = Posisi *pixel* kolom

la = Ukuran lebar matriks lama

2.3 Grayscale

Citra akan diolah dengan mengubah warna citra menjadi derajat keabuan. *Grayscale* dilakukan untuk nantinya diproses dengan *local thresholding* dengan

tujuan memudahkan sistem untuk memproses gambar. Pada proses *grayscale* citra RGB akan diambil *pixel*-nya dan dijumlahkan kemudian dibagi menjadi 3 dan hasilnya akan dimasukan Kembali pada posisi *pixel* yang tadi diambil.

$$grayscale = \frac{R + G + B}{3} \tag{2.3}$$

Keterangan :

Grayscale = Nilai *pixel grayscale*

R = Nilai *pixel* merah

G = Nilai *pixel* hijau

B = Nilai *pixel* biru

2.4 Thresholding

Thresholding merupakan suatu metode segmentasi untuk memisahkan antara objek dan *background* berdasarkan tingkat kecerahannya. *Thresholding* bisa saja disebut binerisasi. Maka, citra akan diubah menjadi derajat keabuan citra biner (hitam dan putih). Metode *thresholding* yang biasa dilakukan menggunakan metode *local thresholding* yang diformulasikan oleh Bensen's Technique [3].

$$T = \frac{max + min}{2} \tag{2.4}$$

Keterangan :

T = Nilai local thresholding

Max = Nilai maksimum tepi

Min = Nilai minimum tepi

Setelah nilai *threshold* didapat. Nilai matriks tersebut dimasukan pada persamaan berikut :

$$f(x,y) = \begin{cases} 0, img(x,y) \geq T(x,y) \\ 1, img(x,y) < T(x,y) \end{cases} \tag{2.5}$$

Keterangan :

f = fungsi untuk mendapatkan citra biner

img = nilai *pixel* citra grayscale

0 = Putih

1 = Hitam

2.5 Segmentasi

Pada tahap ini citra kemudian dibagi menjadi beberapa daerah. Salah satu metode segmentasi adalah *Connected Component Labeling* yang merupakan metode paling umum yang digunakan untuk pengolahan citra digital[14]. Metode ini memiliki sifat ketetanggaan, yang maksud dari tetangga adalah daerah 1 pixel sekelilingnnya. Terdapat 2 jenis ketetanggaan yaitu :

1. 4-*connectivity neighbour*

   Piksel – piksel yang terletak berdampingan secara horizontal dan vertikal. Contohnya dapat dilihat pada gambar dibawah ini.

	P(x, y-1)	
P (x-1, y)	P (x,y)	P(x+1, y)
	P(x, y+1)	

Gambar 2. 2 4-connectivity Neighbour

2. 8-*connectivity neighbour*

   Piksel – piksel yang terletak berdampingan secara horizontal, vertikal dan diagonal. Contohnya dapat dilihat pada gambar dibawah ini.

P (x-1, y-1)	P(x, y-1)	P (x+1, y-1)
P (x-1, y)	P (x,y)	P(x+1, y)
P (x-1, y+1)	P(x, y+1)	P (x+1, y+1)

Gambar 2. 3 8-connectivity Neighbour

Setelah proses *Connected Component Labeling* akan dilakukan pencarian batas, untuk menentukan batas atas, bawah, kiri dan kanan. Untuk mencari batasan tersebut menggunakan tahapan seperti berikut :

1. Pertama akan dilakukan inisialisasi awal terhadap batasan yang akan digunakan.

   $batas[0] = temp[0][x]$ //batas kanan

   $batas[1] = temp[0][y]$ //batas atas

   $batas[2] = temp[0][x]$ //batas kiri

   $batas[3] = temp[0][y]$ //batas bawah

2. Kedua akan dilakukan pencarian batasan terhadap piksel dengan cara berikut

   $$batas[0] = \begin{cases} temp[i][x], jika\ temp[i][x] > batas[0] \\ batas[0]. jika\ tidak \end{cases}$$

   $$batas[1] = \begin{cases} temp[i][y], jika\ temp[i][y] < batas[1] \\ batas[1]. jika\ tidak \end{cases}$$

   $$batas[2] = \begin{cases} temp[i][x], jika\ temp[i][x] < batas[2] \\ batas[2]. jika\ tidak \end{cases}$$

   $$batas[3] = \begin{cases} temp[i][y], jika\ temp[i][y] > batas[3] \\ batas[3]. jika\ tidak \end{cases}$$

   Pencarian ini akan dilakukan terus menerus hingga semua nilai pada matriks temp telah dibaca.

3. Terakhir, akan dilakukan pemotongan pada matriks terhadap batasan – batasan yang sebelumnya sudah ditentukan yang nantinya akan mengubah nilai dari koordinat sebelumnya.

2.6 Segmentasi Vertikal dan Horizontal

Pada proses ini citra di segmentasi menjadi dua bagian yaitu vertikal dan horizontal. Daerah ini akan digunakan untuk mengenali objek pada citra dan membedakan antara fitur citra. Segementasi yang dilakukan pada proses ini hanya segmentasi horizontal, yang membagi menjadi dua daerah yaitu bagian atas dan bawah dengan perbandingan 70% bagian atas dan 30% bagian bawah. Bagian yang diambil hanya bagian atas yaitu bagian mahkota yang akan digunakan untuk proses *processing*.

2.7 Augmentasi Data

Convolutional Neural Network (CNN) bergantung terhadap *big data*, akan tetapi *big data* belum tentu mudah untuk didapatkan, oleh karena itu data dapat di augmentasi untuk mengukur kualitas dan ukuran dari data *training* yang terbatas [12].

Augmentasi merupakan proses memodifikasi sebuah citra, sehingga citra akan diubah bentuk dan posisinya yang berbeda dengan citra asalnya. Pada kasus kasus klasifikasi penggunaan augmentasi data berhasil meningkatkan performa dari model dan membantu untuk mengatasi masalah yang umum terjadi pada deep learning yaitu *data hungry*.

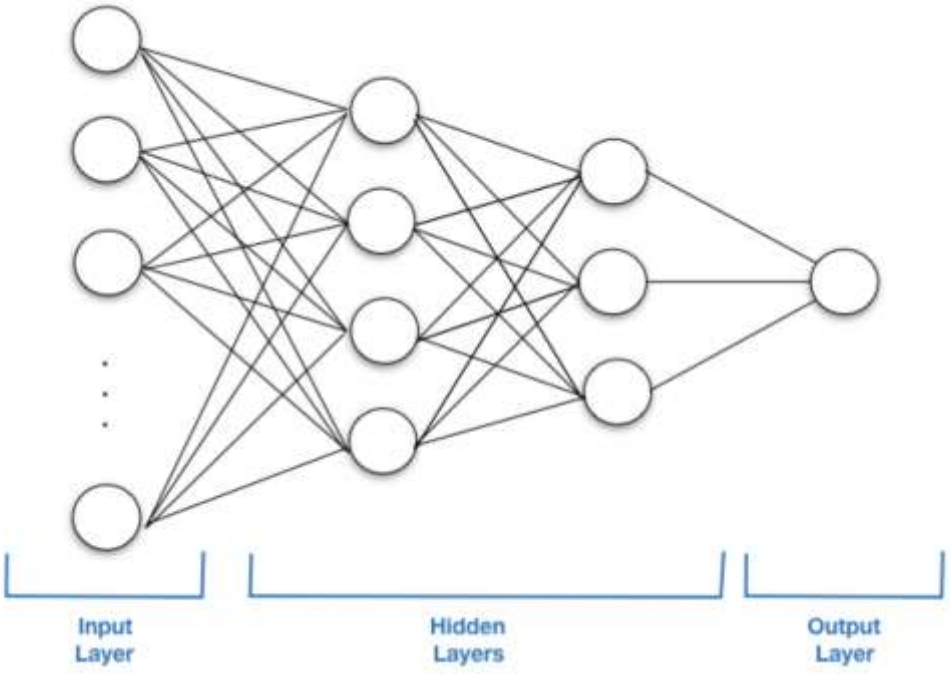
Terdapat beberapa jenis augmentasi diantaranya yaitu augmentasi rotasi, dapat digunakan searah jarum jam atau sebaliknya. Metode *flipping horizontal* merupakan metode yang terbukti efektif pada CIFAR-10 dan memberikan efek sebaliknya terhadap dataset MINTS [13]. Lalu, metode lainnya adalah metode kernel filter yaitu metode yang dapat mempertajam gambar atau membuat buram gambar. Metode selanjutnya adalah metode random erasing yang menseleksi secara acak pixel pada gambar lalu menghapusnnya. Dan metode terakhir yaitu metode *zoom* atau menskalakan gambar.

2.8 Deep Learning

Deep learning adalah bidang pembelajaran mesin yang menggunakan jaringan syaraf tiruan untuk mengimplementasikan masalah dengan kumpulan data besar. Teknik *deep learning* memberikan arsitektur yang sangat kuat untuk supervised learning. Dengan menambahkan lapisan memungkinkan model learning untuk lebih mewakili data gambar berlabel menjadi lebih baik. *Machine learning* memiliki teknik untuk mengekstraksi fitur dari data pelatihan dan algoritma pembelajaran khusus untuk mengklasifikasikan gambar dan mendeteksi suara. Namun, metode ini masih memiliki beberapa kekurangan baik dari segi kecepatan maupun akurasi. Penerapan konsep *deep neural network* (berlapis) dapat diinterupsi pada algoritma *machine learning* yang ada, memungkinkan komputer untuk belajar dengan cepat, akurat, dan dalam skala besar. [7]

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* [7]. Geoffrey Everest Hinton mengungkapkan bahwa *deep learning* merupakan salah satu cabang *machine learning* yang menggunakan *deep neural network* untuk menyelesaikan permasalahan pada domain machine learning. Perbedaan antara *machine learning* dan *deep learning* terletak pada *feature extraction.* Dimana pada *deep learning* sudah terdapat tahap *feature extraction* sedangkan pada *machine learning* tahap tersebut tidak ada.

Algoritma dari *deep learning* adalah jaringan yang meniru otak manusia, yang memiliki neuron – neuron yang saling berhubungan dan bekerjasama untuk mengolah informasi. Berikut adalah model dari *deep learning* :



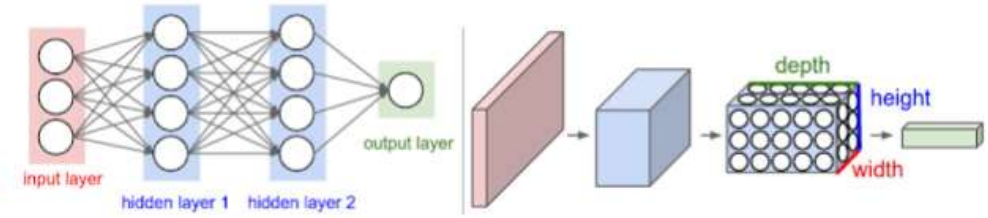
Gambar 2. 4 Diagram Multi Layer Perceptron

Diagram pada gambar 2.1 memiliki 3 komponen jaringan yaitu *input layer* yang bertugas untuk memasukan data ke dalamnya, *hidden layer* bertugas untuk memproses informasi yang diterima pada tingkatan yang berbeda dikarenakan memiliki ratusan *hidden layer* yang dapat digunakan untuk menganalisis masalah dari sudut pandang baru dan menyesuaikan dengan informasi baru yang diterimanya, dan *output layer* mengeluarkan data yang telah diolah oleh komponen jaringan sebelumnya.

2.9 Convolutional Neural Network

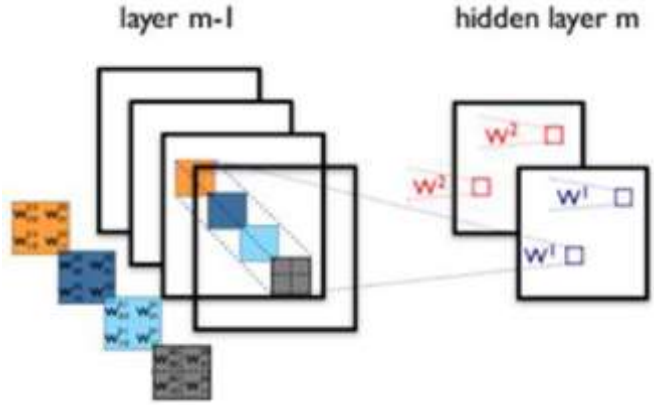
Convolutional neural network (CNN) berhasil dikembangkan oleh seorang peneliti bernama Kunihiko Fukushima dari *NHK Broadcasting Science Research Laboratories*, Kinuta, Segawa, Tokyo, Jepang, dengan nama NeoCognitron [8]. Yang digunakan sebagai metode untuk mengidentifikasi objek digital.

Cara kerja CNN mempunyai kemiripan seperti MLP (*Multi Layer Perceptron*), yang membedakan terletak pada dimensi setiap neuron pada CNN yang dipresentasikan dalam bentuk dua dimensi sedangkan pada MLP setiap neuron hanya memiliki satu dimensi.



Gambar 2. 5 Perbedaan antara MLP dan CNN

. Pengoperasian linear dan parameter bobot CNN berbeda dengan MLP. Pada CNN operasi linear yang digunakan adalah operasi konvolusi, dan bobot yang digunakan pada CNN yaitu berbentuk 4 dimensi yang merupakan kumpulan kernel kovolusi yang dapat terlihat pada gambar 2.3. Dengan sifatnya ini CNN hanya dapat digunakan untuk pengolahan data yang berstruktur 2 dimensi seperti citra dan suara [9].



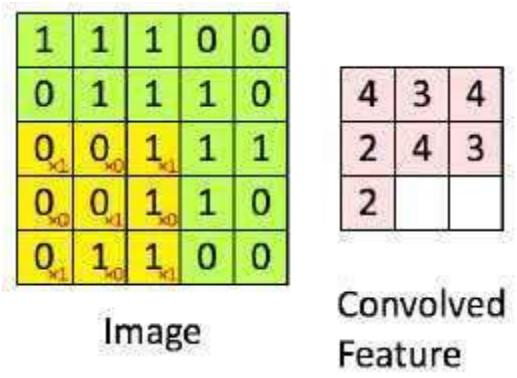
Gambar 2. 6 Proses Konvolusi CNN

Input awal dari CNN berupa balok tiga dimensi yang ditransformasikan menjadi *output* tiga dimensi yang berfungsi sebagai diferensial parameter. CNN membentuk neuron – neuron lainnya dalam bentuk tiga dimensi (panjang, lebar dan tinggi) pada masing - masing lapisannya. Arsitektur CNN terbagi menjadi

feature extraction layer dan *fully-connected layer*. Kontribusi CNN pada *feature extraction layer* terletak pada *convolution layer* dan *pooling layer* [10].

2.9.1 Convolutional Layer

Konvolusi adalah mengaplikasikan sebuah fungsi pada *output* fungsi lain secara berulang. Tujuan konvolusi adalah untuk mengekstraksi fitur dari citra yang dimasukan. Hasil konvolusi menghasilkan transformasi data yang sesuai dengan informasi spasial pada data. Bobot pada layer tersebut menspesifikasikan kernel konvolusi yang akan digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan *input* pada CNN [9].



Gambar 2. 7 Proses Konvolusi

Pada layer ini dilakukan perhitungan antara dua nilai "dot" yaitu nilai *input* dan nilai filter. Dimana ukuran dari filter adalah *k* * *k*. dengan persamaan (2.6).

$$C_{j,x,y} = f(I \otimes K_{u,v} + B_i)$$

$$dimana \otimes = \sum_{u=i}^{3} \sum_{v=i}^{3} I_{x+u,y+u} * K_{u,v} \quad (2.6)$$

Keterangan :

$C_{j,x,y}$ = Layer konvolusi ke-j dari matriks x,y

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

$K_{u,v}$ = Nilai pixel dari filter ke-u dan ke -v

B_i = Nilai bias ke-i

f = Fungsi aktivasi *non-linearity* dengan ReLU

Hasil dari persamaan diatas akan menghasilkan *activation map* untuk kemudian diaktivasi dengan fungsi aktivasi *Rectified Linear Unit* (ReLU). Fungsinya untuk pixel yang memiliki nilai < 0 akan diubah menjadi 0, fungsinya dapat dilihat sebagai berikut.

$$f(x) = ReLU(x) = \begin{cases} x(x \geq 0) \\ 0(x < 0) \end{cases}$$

Keterangan :

$ReLU(x)$ = Fungsi aktivasi ReLU

x = Hasil konvolusi pada layer konvolusi

Pada proses konvolusi di layer selanjutnya, akan menggunakan rumus yang berbeda, dimana inputan sebanyak jumlah filter masing – masing akan di konvolusi kemudian dijumlahkan, dapat dilihat pada persamaan (2.7).

$$C_{j,x,y} = f(\sum_{i=1}^{filter} S_i \otimes K_{u,v} + B_i) \quad (2.7)$$

Keterangan :

$C_{j,x,y}$ = Layer konvolusi ke-j dari matriks x,y

S_i = Layer pooling ke-i

$K_{u,v}$ = Nilai pixel dari filter ke-u dan ke -v

B_i = Nilai bias ke-i

f = Fungsi aktivasi *non-linearity* dengan ReLU

⊗ = Operasi konvolusi

2.9.2 Pooling layer

Pooling layer adalah proses mereduksi ukuran sebuah citra atau tujuan lainnya yaitu untuk meningkatkan invariansi posisi dari fitur [9]. Pooling layer terdapat dua jenis yaitu *max pooling* dan *average pooling*, *max pooling* mengekstraksi fitur yang penting seperti tepi, sedangkan *average pooling* lebih halus dibanding *max pooling* [11]. Dalam sebagian besar CNN metode yang sering digunakan adalah *max pooling*. *Max pooling* membagi output dari *convolutional layer* menjadi beberapa grid kecil lalu mengambil nilai maksimal dari setiap grid untuk menyusun citra yang direduksi. Adapun persamaan yang akan digunakan untuk mengambil nilai maksimum dari grid yang telah dibagi, persamaan tersebut dapat dilihat pada persamaan (2.8)

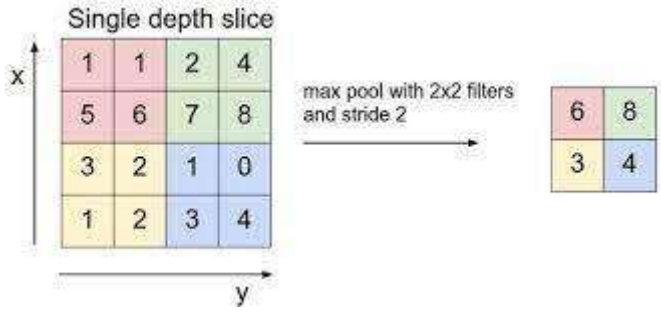
$$S_{x,y} = Max(C_{x,y}, C_{x+1,y}, C_{x,y+1}, C_{x+1,y+1},) \quad (2.8)$$

Keterangan :

$s_{x,y}$ = Hasil pooling layer

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

Mask akan bergerak dengan jarak 2 pixel sehingga nilai pixel yang sudah dicari nilai maksimumnya tidak akan dilakukan proses perhitungan kembali.



Gambar 2. 8 Max Pooling Layer

2.9.3 Fully Connected Layer

Pada layer ini dimensi data akan ditransformasikan agar data dapat diklasifikasikan secara linier. Setiap neuron pada *convolutional layer* akan ditransformasikan menjadi satu dimensi sebelum dapat dimasukan ke dalam *fully connected layer* [9]. Pada tahapan *flatten* matriks hasil dari pooling layer terakhir akan dibuat menjadi vektor, kemudian vektor tersebut dihitung dengan persamaan (2.9).

$$F_i = \sum_{j}^{lenght} W_{i,j} * flatten_j + b_i \quad (2.9)$$

Keterangan :

F_i = Hasil perhitungan pada *fully connected layer*

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

$flatten_j$ = Nilai vektor ke-j

b_i = kelas ke-i (i=1,2,3)

Kemudian hasil dari perhitungan tersebut akan diaktivasi dengan fungsi *softmax*, tujuannya agar dapat diketahui prediksi yang dihasilkan dari arsitektur yang dibangun. Rumus perhitungan *softmax* terdapat pada persamaan (2.10)

$$y_i = \frac{e^{F_i}}{(\sum_{i=1}^{kelas} e^{F_i})} \quad (2.10)$$

Keterangan :

y_i = Hasil aktifasi *softmax*

F_i = Hasil dari perhitungan pada *fully connected layer* ke-i

i = kelas ke-i (i=1,2,3)

Dikarenakan hasil transformasi satu dimensi tersebut, data kehilangan informasi spasial dan menjadi tidak reversibel, *fully connected layer* hanya dapat diimplementasikan di akhir jaringan. Dikarenakan pada *convolutional layer* dengan ukuran kernel 1 x 1 melakukan hal yang sama seperti *fully connected layer* menyebabkan penggunaan *fully connected layer* pada CNN tidak terlalu terpakai [9].

2.9.4 Cross-Entropy Loss function

Pada tahap ini akan dicari nilai error yang didapat dari proses sebelumnnya. Yang dimana nilai error ini akan digunakan untuk menentukan hasil prediksi dari CNN sudah mencapai target atau tidak. Maka dilakukan perbandingan antara error dengan target error. Bila hasil error masih dibawah target maka akan dilakukan proses *loss function*. Berikut adalah perhitungan *cross-entropy loss funciton*.

$$\text{Loss} = -\sum_{j}^{kelas} t_i * Log(y_i) \tag{2.11}$$

Keterangan :

y_i = Hasil aktifasi *softmax*

t_i = Nilai dari target bernilai 1 untuk target kelas yang dituju

i = kelas ke-i (i=1,2,3)

2.9.5 Backpropagation

Metode untuk mendapatkan turunan fungsi dari fungsi – fungsi yang digunakan. Backpropagation dapat diaplikasikan ke neural network. Neural network dapat mempelajari dan menentukan bobot dan bias menggunakan algoritma *gradient descent*. *Gradient descent* merupakan algoritma untuk optimasi orde pertama untuk menemukan nilai lokal dari sebuah fungsi [15]. Pada

backpropagation menggunakan rumus *chain rule* terhadap *gradient descent* untuk memperbaiki bobot dan bias.

2.9.5.1 Turunan gradient error terhadap softmax

Tahapan turunan gradient *error* terhadap *softmax* dirumuskan oleh Peter Sadowski[20].

$$\Delta y_i = \frac{\partial loss}{\partial y_i} = y_i - t_i \tag{2.12}$$

Keterangan :

Δy_i = Nilai turunan dari fungsi softmax

ti = Nilai dari target, bernilai 1 apabila target adalah target kelas yang dituju dan berlaku kebalikannya

i = kelas ke-i

2.9.5.2 Turunan gradient error terhadap bobot

Tahapan turunan gradient error terhadap bobot yang dirumuskan oleh Zhifei Zhang[21].

$$\Delta W_{i,j} = \frac{\partial loss}{\partial W_{i,j}} = \Delta y_i * flatten_i \tag{2.13}$$

Keterangan :

$\Delta W_{i,j}$ = Nilai turunan terhadap bobot

Δy_i = Nilai turunan dari fungsi softmax

$flatten_i$ = Nilai vektor flatten

i = kelas ke-i

2.9.5.3 Turunan gradient error terhadap bias

Turunan gradien error terhadap bias berdasarkan rumus oleh Zhifei Zhang[21].

$$\Delta b_i = \frac{\partial loss}{\partial b_i} = \Delta y_i \tag{2.14}$$

Keterangan :

Δb_i = Nilai turunan dari bias

Δy_i = Nilai turunan dari fungsi softmax

i = kelas ke-i

2.9.6 Stohastic Gradient Descent

Perhitungan bobot baru menggunakan Stohastic Gradient Descent[22]. Berikut adalah rumus perbaikan bobot dan bias baru pada persamaan dibawah ini.

1. Perbaikan nilai pada bobot

$$\theta W_i = W_i - \alpha(\Delta W_i) \tag{2.15}$$

Keterangan :

θW_i = Nilai bobot baru

W_i = Nilai bobot

α = Nilai learning rate

ΔW_i = Nilai turunan dari bobot

i = kelas ke-i

2. Perbaikan nilai pada bias

$$\theta b_i = b_i - \alpha(\Delta b_i) \quad (2.16)$$

Keterangan :

θb_i = Nilai bias baru

b_i = Nilai bias

α = Nilai learning rate

Δb_i = Nilai turunan dari bias

i = kelas ke-i

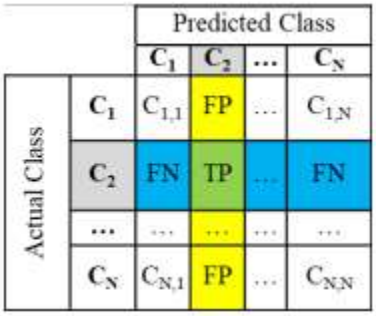
2.10 TensorFlow

Tensorflow diciptakan oleh Google yang biasanya digunakan untuk komputasi numerik dan project dari machine learning dengan skala yang besar. Tensorflow bersifat *open source* dan menggunakan beberapa algoritma *machine learning* hingga *deep learning*. Tesnsorflow digunakan untuk melatih *neural network* seperti pengklasifikasian gambar, objek atau kata.

Pada penilitan ini Tensorflow digunakan untuk melatih model Convolutional Neural Network untuk membantu mengklasifikasikan gambar pohon. Library Tesorflow yang akan digunakan adalah *ImageDataGenerator*, model *sequential*, layer CNN seperti *Conv2D, MaxPooling2D, Activation, Dense, Flatten* dan *dropout*. Optimizer yang digunakan seperti Adam, Adamax, dan RMSprop

2.11 Pengujian Akurasi

Pengujian dilakukan untuk menghitung akurasi dari model yang telah didapat pada proses training CNN. Model tersebut kemudian akan dilakukan pengujian pada tahap testing. Perhitungan akurasi akan menggunakan *confusion matrix*. *Confusion matrix* pada *multiclass* digambarkan sebagai berikut [19].



Gambar 2. 9 Multiclass Confusion Matrix

Berdasarkan dari gambar tersebut maka dapat ditentukan bahwa TP (True Positive) adalah kelas sesungguhnya dimana kelas sebenarnya sama dengan kelas prediksi, pada FP (False Positive) adalah kelas sebenarnya yang dikategorikan kelas yang lain. FN (False Negative) adalah kelas yang salah diprediksi. Dari hasil klasifikasi yang sudah dikategorikan maka hasil tersebut akan dihitung nilai *accuracy*, *precision*, dan *recall* pada rumus berikut.

$$Accuracy = \frac{\sum_{i=1}^{N} TP_{ci}}{\sum_{i=1}^{N}\sum_{j=1}^{N} C_i} * 100\% \quad (2.17)$$

$$Precision = \frac{TP_{Ci}}{TP_{ci} + FP_{ci}} * 100\% \quad (2.18)$$

$$Recall = \frac{TP_{Ci}}{TP_{ci} + FN_{ci}} * 100\% \quad (2.19)$$

Keterangan :

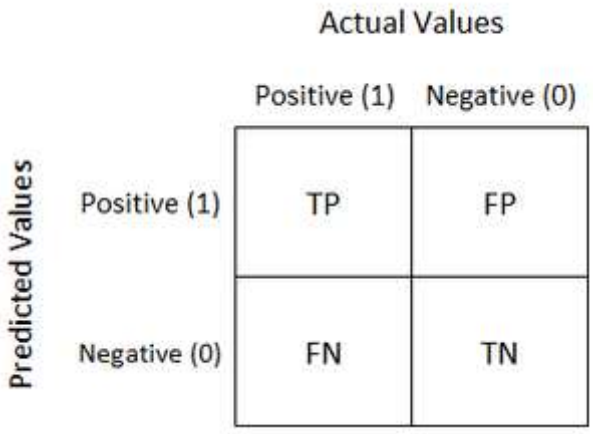
TP = True Positive

FP = False Positive

FN = False Negative

Adapun perhitungan *confusion matrix* pada kelas biner untuk perhitungan pada fitur buah seperti gambar berikut.

Gambar 2. 10 Confusion matrik biner



Dari gambar sebelumnya didapat TN (True Negative) dimana kelas prediksi dan kelas sebenarnya bernilai benar yaitu 0 atau salah. Maka rumus perhitungan untuk menghitung *confusion matriks biner* tersebut adalah sebagai berikut.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} * 100\% \quad (2.20)$$

$$Precision = \frac{TP}{TP + FP} * 100\% \quad (2.21)$$

$$Recall = \frac{TP}{TP + FN} * 100\% \quad (2.22)$$