BAB II

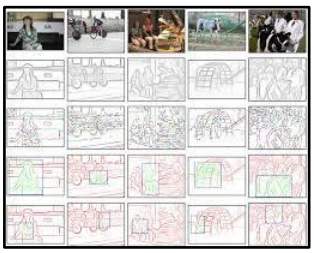
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Ada beberapa penilitiaan terdahulu yang menjadi bahan acuan dalam melakukan penelitian ini.

2.1.1 C Lawrence Zitnick dan Poitr Dollar, 2014.

Pada penelitian terdahulu yang dilakukan oleh (Zitnick & Dollar, 2014) dengan judul penelitian "*Locating Object Proposal From Edge*" melakukan penelitian dan percobaan mengenai efisiensi pendekteksian objek dengan menggabungkan *Bounding box* dan *edge*. Tujuan utama penelitian yang dilakukan adalah untuk pengenalan dan deteksi terhadap bentuk serta posisi objek melalui kontur yang selanjutnya akan diberikan pembatas berupa kotak , objek yang berada didalam kotak tersebut selanjutnya akan diberikan segment garis tepian kontur dengan pada waktu yang bersamaan. Selanjutnya hal tersebut digunakan pula untuk memisahkan antara objek utama yang dituju dengan objek lain yang berada disekitar objek utama, permasalahan yang diatasi peneliti adalah semua objek yang ada akan diberi kotak sekaligus bersamaan akan terbentuk garis tepian didalam kotak yang membentuk kontur objek tersebut secara bebas, penelitian dan percobaan dilakukan dengan menggunakan dataset dari PASCAL VOC , Peneliti melakukan evaluasi terhadap metode yang dijalankan pada pembentukan skor dari kotak dan garis tepi kontur untuk meningkatkan akurasi terhadap objek utama yang dituju .



Gambar 2. 1 Ilustrasi Metode Edgeboxes

2.1.2 Wahyu Supriatin, 2020.

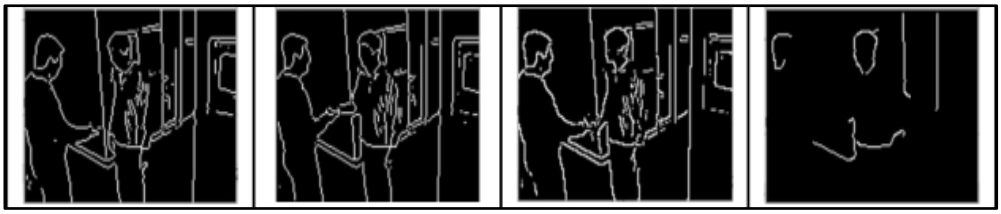
Pada penelitian terdahulu yang dilakukan oleh (Supriyatin, 2020) dengan judul penelitian "Perbandingan Metode *Sobel*, *Prewitt*, *Robert* dan *Canny* pada Deteksi Tepi Objek Bergerak" melakukan penelitian untuk menganalisis perbandingan metode deteksi tepi dengan menggunakan objek bergerak (video). Perbandingan deteksi tepi dilakukan dengan menggunakan empat buah algoritma yaitu algoritma Sobel, Prewitt, Robert dan Canny.

Deteksi tepi dilakukan sebagai tahap awal dalam pengenalan/pelacakan objek untuk dapat digunakan dalam proses pengolahan citra selanjutnya. Proses perbandingan deteksi tepi dilakukan dengan menggunakan tools Simulink Matlab. Hasil dari deteksi tepi video dibandingkan dengan menggunakan simulasi parameter dari masing-masing algoritma.

Hasil pengujian digunakan untuk melihat ketebalan dan ketepatan parameter dalam mendeteksi tepi dari objek. Sehingga bentuk objek menjadi lebih jelas sekalipun dalam kondisi bergerak.

Hasil pengujian menunjukkan bahwa variasi gambar, resolusi gambar, format gambar dan spesifikasi letak kamera mempengaruhi hasil. Penelitian ini bersifat kuantitatif karena menggunakan data objektif video yang diambil dengan menggunakan kamera. Citra bergerak jauh berbeda dengan citra diam (citra digital).

Video yang digunakan sebagai objek harus diperhatikan variasi, resolusi dan formatnya karena sangat berpengaruh terhadap ketepatan akurasi algoritma dalam mendeteksi tepi objek. (Supriyatin, 2020)



Gambar 2. 2 Illustrasi Perbandingan Metode Sobel, Prewit, Robert dan Canny

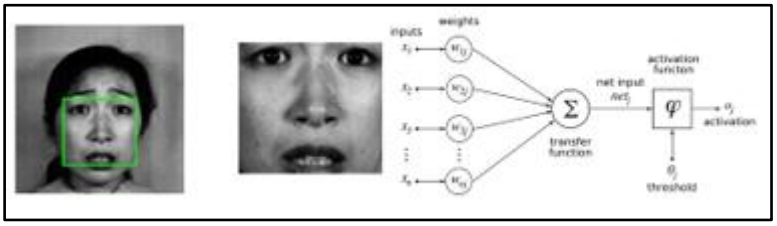
2.1.3 Johanes Cristanto dan Ken Ratri Retno Wardani, 2017.

Pada penelitian terdahulu yang dilakukan oleh (Christanto & Retno Wardani, 2017) dengan judul penelitian "Penerapan Metode *Single-layer Feed-Forward Neural Network* Menggunakan Kernel Gabor untuk Pengenalan Ekspresi Wajah" melakukan penelitian untuk menganalisis akurasi penggunaan metode ekstraksi fitur *Gabor*, *adaboost* untuk seleksi fitur, dan *Feed Forward Neural Network* untuk mengenali 7 ekspresi wajah manusia yang ada di dalam citra.

Metode ekstraksi fitur *Gabor, adaboost* digunakan sebagai pemrosesan seleksi fitur terhadap citra, selanjutnya *Neural Network* merupakan metode yang digunakan untuk mengklasifikasikan hasil dari *Adaboost*. Karena terdapat kemungkinan lebih dari 1 angka 1 yang keluar dari model *adaboost* maka *neural network* yang akan menentukan ekspresi apa yang terdapat pada citra.

Terdapat 3 jenis *layer* di dalam *Neural Network,* yaitu *layer* masukkan yang merupakan tempat memasukkan data, *hidden layer* yang *layer* yang membantu dalam proses perhitungan,

Hasil pengujian menunjukkan bahwa Metode *Single - Layer Feed Forward Neural Network* memberikan hasil yang baik dalam mempelajari pola yang dihasil oleh *AdaBoost*, namun faktor yang paling berpengaruh pada akurasi pengenalan ekspresi wajah adalah metode *Gabor*.



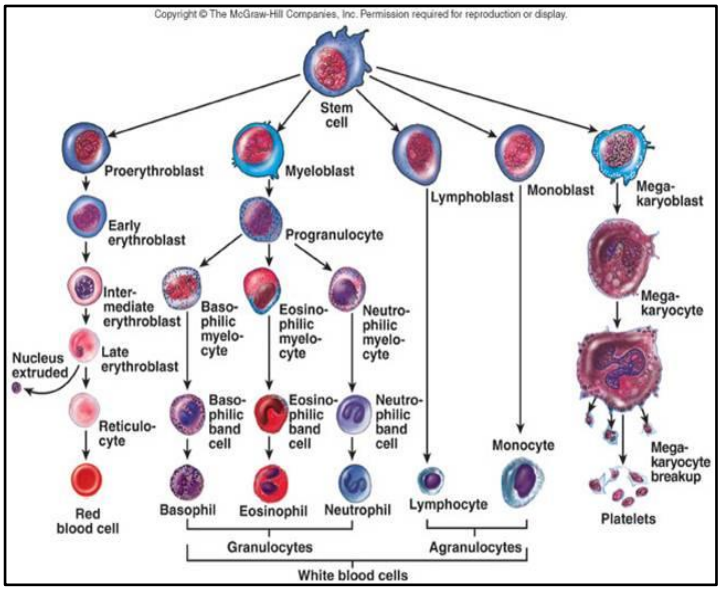
Gambar 2. 3 Illustrasi Pengenalan expresi wajah menggunakan metode Single Layer FeedForward Neural Network

2.2 Dasar Teori

Dalam penelitian tentang "Implementasi Metode Edgeboxes dan Feed Forward Neural Network Untuk Klasifikasi Leukosit Pada Citra Mikroskopis Sel Darah "

2.2.1 Leukosit

Leukosit merupakan sel darah putih yang diproduksi oleh jaringan hemopoetik untuk jenis bergranula (polimorfonuklear) dan jaringan limpatik untuk jenis tak bergranula (mononuklear), berfungsi dalam sistem pertahanan tubuh terhadap infeksi (Sutedjo, 2006). Leukosit paling sedikit dalam tubuh jumlahnya sekitar 4.000-11.000/mm3. Berfungsi untuk melindungi tubuh dari infeksi. Karena itu, jumlah leukosit tersebut berubah-ubah dari waktu ke waktu, sesuai dengan jumlah benda asing yang dihadapi dalam batas-batas yang masih dapat ditoleransi tubuh tanpa menimbulkan gangguan fungsi (Sadikin, 2002). Meskipun leukosit merupakan sel darah, tapi fungsi leukosit lebih banyak dilakukan di dalam jaringan. Leukosit hanya bersifat sementara mengikuti aliran darah ke seluruh tubuh. Apabila terjadi peradangan pada jaringan tubuh leukosit akan pindah menuju jaringan yang mengalami radang dengan cara menembus dinding kapiler (Kiswari, Astikawati, & Carollina, 2014) .

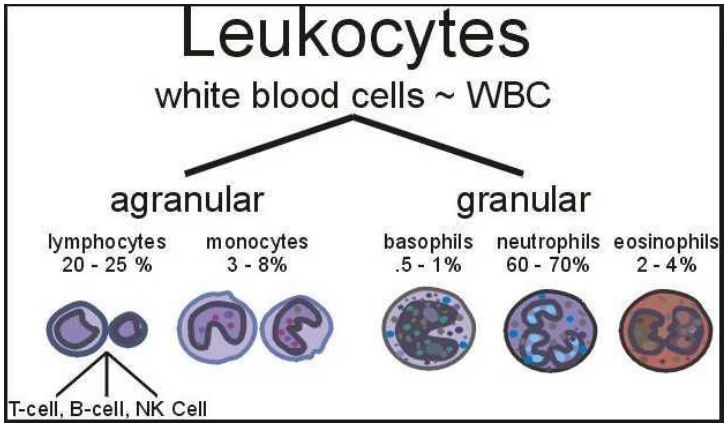


Gambar 2. 4 Illustrasi Hematopoiesis

Leukosit terdiri dari 2 kategori yaitu granulosit dan agranulosit. Granulosit, yaitu sel darah putih yang di dalam sitoplasmanya terdapat granulagranula. Granula-granula ini mempunyai perbedaan kemampuan mengikat warna misalnya pada eosinofil mempunyai granula berwarna merah terang, basofil berwarna biru dan neutrofil berwarna ungu pucat.

Agranulosit, merupakan bagian dari sel darah putih dimana mempunyai inti sel satu lobus dan sitoplasmanya tidak bergranula. Leukosit yang termasuk agranulosit adalah limfosit, dan monosit. Limfosit terdiri dari limfosit B yang membentuk imunitas humoral dan limfosit T yang membentuk imunitas selular. Limfosit B memproduksi antibodi jika terdapat antigen, sedangkan limfosit T langsung berhubungan dengan benda asing untuk difagosit (Tarwoto & Martonah, 2008).

Ada tidaknya granula dalam leukosit serta sifat dan reaksinya terhadap zat warna, merupakan ciri khas dari jenis leukosit. Selain bentuk dan ukuran, granula menjadi bagian penting dalam menentukan jenis leukosit (Nugraha, 2015). Dalam keadaan normal leukosit yang dapat dijumpai menurut ukuran yang telah dibakukan adalah basofil, eosinofil, neutrofil batang, neutrofil segmen, limfosit dan monosit. Keenam jenis sel tersebut berbeda dalam ukuran, bentuk, inti, warna sitoplasma serta granula didalamnya (Arif, 2015).



Gambar 2. 5 Illustrasi Klasifikasi Sel Darah Leukosit dalam Kelas Granula dan Agranula

2.2.2 Citra Digital

Citra (*image*) adalah kombinasi antara titik, garis, bidang dan warna untuk menciptakan suatu imitasi dari suatu objek, biasanya objek fisik atau manusia.

Citra bisa berwujud gambar (*picture*) dua dimensi, seperti lukisan, foto dan berwujud tiga dimensi, seperti patung. Citra terbagi 2 yaitu ada citra yang bersifat analog dan ada citra yang bersifat digital.

Citra analog tidak dapat direpresentasikan dalam komputer, sehingga tidak bisa diproses oleh komputer secara langsung. Citra analog harus dikonversi menjadi citra digital terlebih dahulu agar dapat diproses di komputer (Nurtantio, Sutojo, & Muljono, 2017).

Citra digital adalah citra elektronik yang diambil dari beberapa jenis dokumen, yaitu berupa foto, buku, ataupun video dan suara. Proses yang digunakan untuk merubah citra analog menjadi suatu citra digital disebut sebagai proses digitasi.

Digitasi adalah proses dimana mengubah suatu gambar, teks, atau suara yang berasal dari benda dapat dilihat ke dalam data elektronik dan dapat disimpan serta diperoses untuk keperluan yang lainnya.

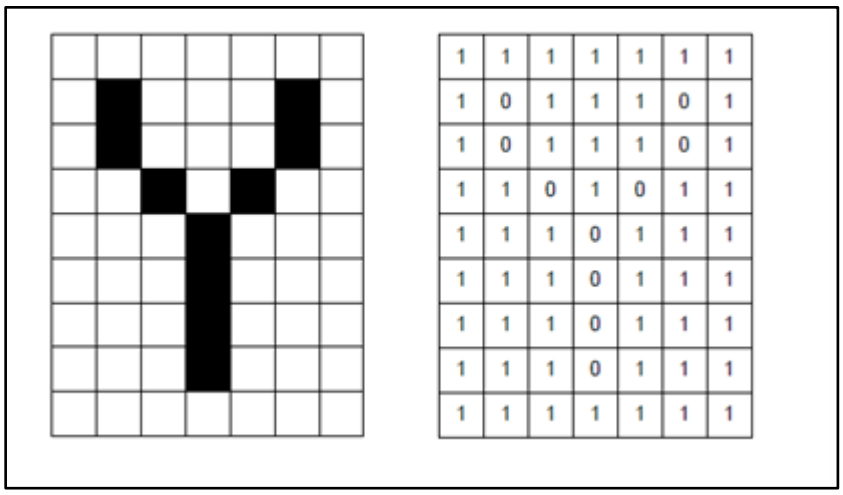
Citra digital merupakan sebuah representasi numerik (mayoritas biner) dari gambar 2 dimensi.

Sebuah gambar dapat didefinisikan sebagai fungsi 2 dimensi f(x,y) di mana x dan y merupakan titik koordinat bidang datar, dan harga dari fungsi f dari setiap pasangan titik koordinat (x,y) yang disebut dengan intensitas atau level keabuan (*grey level*) dari suatu gambar.

Ketika nilai titik x,y dan nilai intensitas f terbatas dengan nilai diskrit, ma gambar tersebut akan dapat dikatakan sebagai sebuah citra digital (Gonzales, Rafael, & Woods, 2002).

2.2.2.1 Citra Biner

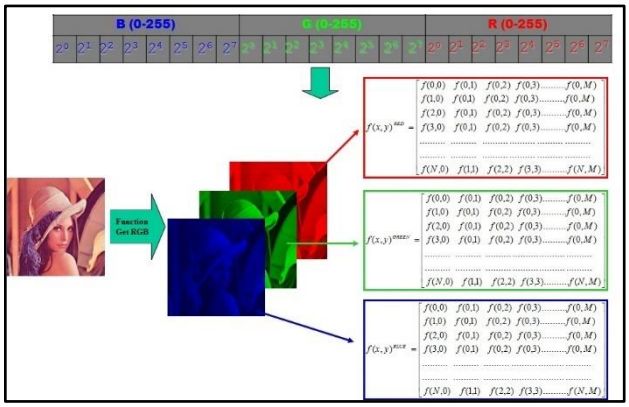
Citra Biner merupakan citra yang hanya memiliki warna atau dua nilai saja, yaitu hitam atau putih dan 0 atau 1 (Sudaryanto, 2018).



Gambar 2. 6 Illustrasi Citra Biner

2.2.2.2 Citra RGB (Red Green Blue)

Citra RGB (Red Green Blue) adalah citra yang terbentuk atas 3 model warna yaitu Red (Merah), Green (Hijau), Blue (Biru). Komponen warna ini ditambahkan berbagai macam cara yang selanjutnya akan menghasilkan berbagai macam warna.

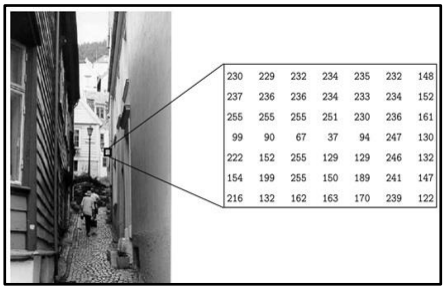


Gambar 2. 7 Ilustrasi Citra RGB

2.2.2.3 Citra Grayscale

Citra Grayscale atau citra keabuan adalah sebuah citra dengan matriks yang memiliki nilai intensitas dari setiap piksel yakni minimum 0 dan maksimum 255, nilai minimum dan maksimum tersebut kemungkinan banyak bergantung pada jumlah bit yang digunakan, pada umumnya menggunakan 8 bit. (Maria et al, 2018). Berikut adalah rumus mengubah citra RGB menjadi Grayscale dengan persamaan:

$$Gray(1,1) = 0{,}299 \times R + 0.587 \times G + 0.114 \times B$$



Gambar 2. 8 Illustrasi Citra *Grayscale*

2.2.3 Pengolahan Citra Digital

Pengolahan citra digital (*image processing*) merupakan proses mengolah piksel piksel dalam citra digital untuk suatu tujuan tertentu. Sebuah gambar disebut dengan citra digital apabila gambar yang dihasilkan berasal dari proses sebuah komputer, kamera, scanner atau perangkat elektronik lainnya.

Pengolahan citra digital diproses oleh komputer dengan menggunakan algoritma. Citra digital direpesentasikan dengan matriks, sehingga pengolahan pada citra digital pada dasarnya memanipulasi elemen-elemen matriks yang berupa piksel (A'la, 2016).

Secara matematis, citra merupakan fungsi kontinyu (*continue*) dengan intensitas cahaya pada bidang dua dimensi. Agar dapat diolah dengan komputer digital, maka suatu citra harus dipresentasikan secara numerik dengan nilai-nilai diskrit.

Repersentasi dari fungsi kontinyu menjadi nilai-nilai diskrit disebut digitalisasi citra. Sebuah citra digital dapat diwakili oleh sebuah matriks dua dimensi f(x,y) yang terdiri dari M kolom dan N baris, dimana perpotongan antara kolom dan baris disebut piksel (*pixel = picture element*) atau elemen terkecil dari sebuah citra (Kusmanto & Tompunu, 2011).

$$f(x,y) \approx \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix} \quad (1)$$

Suatu citra f(x,y) dalam fungsi matematis dapat dituliskan sebagai berikut:

$$0 \leq x \leq M\text{-}1$$

$$0 \leq y \leq N\text{-}1$$

$$0 \leq f(x,y) \leq G\text{-}1$$

dimana: M = jumlah piksel baris (*row*) pada array citra

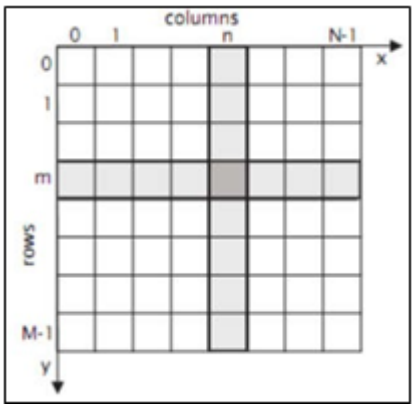
N = jumlah piksel kolom (*column*) pada array citra

G = nilai skala keabuan (*gray level*)

Besarnya nilai M, N dan G pada umumnya merupakan perpangkatan dari dua.

$$M = 2^m ; N = 2^n; G = 2^k$$

Dimana nilai m, n dan k adalah bilangan bulat positif. Interval (0,G) disebut skala keabuan (*grayscale*). Besar G tergantung pada proses digitalisasinya. Biasanya keabuan 0 (nol) menyatakan intensitas hitam dan 1 (satu) menyatakan intensitas putih. Untuk citra 8 bit, nilai G sama dengan $2^8 = 256$ warna (derajat keabuan) (Kusmanto & Tompunu, 2011).



Gambar 2. 9 Illustrasi Citra 2 Dimensi

Tujuan pengolahan citra antara lain dapat memperbaiki kualitas gambar dilihat dari aspek radiometrik (peningkatan kontras, transformasi warna, restorasi citra) dan dari aspek geometrik (rotasi, translasi, skala, transformasi geometrik).

Selain itu pengolahan citra juga dilakukan untuk proses penarikan informasi atau deskripsi objek atau juga untuk pengenalan objek yang terkandung pada citra tersebut (Hermawati, 2013). Beberapa system dilakukannya pengolahan citra pada citra digital antara lain adalah:

1. Untuk mendapatkan citra asli dari suatu citra yang buruk karena pengaruh derau. Proses pengolahan bertujuan mendapatkan citra yang diperkirakan mendekati citra sesungguhnya.
2. Untuk memperoleh citra dengan karakteristik tertentu dan cocok secara visual yang dibutuhkan untuk tahap yang lebih lanjut dalam pemrosesan analisis citra.

Pada pengolahan citra yang diharapkan adalah terbentuknya suatu system yang dapat memproses citra masukan sehingga citra tersebut dapat dikenali cirinya. Pengenalan ciri inilah yang sering diapliasikan dalam kehidupan sehari-hari. Secara umum operasi pengolahan citra dapat dipilih menjadi beberapa kelompok berikut :

1. Perbaikan citra (*image restoration*)
2. Peningkatan kualitas citra (*image enhancement*)
3. Registrasi citra (*image registration*)

4. Pemampatan data citra (*image data compaction*)

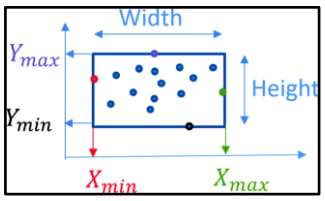
5. Pemilahan citra (*image segmentation*)

2.2.3.1 Preprocessing Citra

Preprocessing Citra adalah sebuah proses untuk memberikan sebuah peningkatan kualitas pada citra untuk lebih baik lagi. Salah satu contoh preprocessing adalah dengan mengubah ukuran citra, mengubah citra RGB menjadi citra Grayscale dan lain lain. (Purnamasari, 2017) .

2.2.4 Metode Edgeboxes

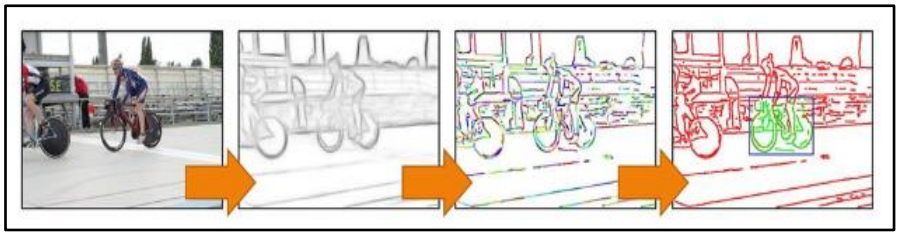
Metode Edgeboxes adalah metode segmentasi yang menggunakan pendekatan *Bounding box* dan algoritma canny edge untuk memberi garis sisi dan kontur pada objek. Penggabungan *Bounding box* dan *edge* untuk pengenalan dan deteksi terhadap bentuk serta posisi objek melalui kontur yang selanjutnya akan diberikan pembatas berupa kotak , objek yang berada didalam kotak tersebut selanjutnya akan diberikan segment garis tepian kontur. Selanjutnya hal tersebut digunakan pula untuk memisahkan antara objek utama yang dituju dengan objek lain yang berada disekitar objek utama (Zitnick & Dollar, 2014)



Gambar 2. 10 Illustrasi Canny dan Boundingbox

Pendeteksian objek utama dengan proses pemisahan dan pengelompokan objek yang berdasarkan jumlah, posisi, bentuk, warna dengan pendekatan setiap objek yang terdeteksi akan langsung dipisah masing masing oleh kotak, selanjutnya objek didalam kotak akan diberi garis tepi kontur sehingga meningkatkan akurasi terhadap onjek yang dituju(Jing Liu, Tongwei Ren, Jia Bei, 2016)

Garis tepian kontur merupakan kunci pengukuran dan pengenalan bentuk utama objek, dengan adanya garis tepian kontur yang membentuk kontur model terhadap objek, digunakan untuk memprediksi objek dan model/ bentuk dari objek tersebut setelah dikotaki dan dikelompokan(Xiang Bay,2017)



Gambar 2. 11 Illustrasi Metode Edgeboxes dengan kombinasi Algoritma Canny edge dan Boundingbox

2.2.4.1 Algoritma Canny Edge

Canny adalah algoritma deteksi tepi yang banyak digunakan dalam berbagai penelitian karena dinilai sebagai algoritma deteksi tepi yang paling optimal. Langkah awal pada algoritma Canny adalah mengimplementasikan tapis Gaussian pada citra untuk menghilangkan derau.

Kemudian dilanjutkan dengan melakukan deteksi tepi pada citra dengan salah satu algoritma deteksi tepi yang ada, misalnya Sobel atau Prewitt.

Langkah berikutnya adalah membagi garis-garis yang ada menjadi 4 warna terpisah dengan sudut masing-masing, lalu memperkecil masing-masing garis tepi agar menjadi tipis (non maximum surpression).

Langkah terakhir adalah melakukan proses binerisasi berdasarkan nilai low & high threshold yang diberikan. Deteksi tepi merupakan metode yang telah banyak diimplementasikan dalam berbagai penelitian dan pada aspek yang bermacam-macam (Edoh, 2011)

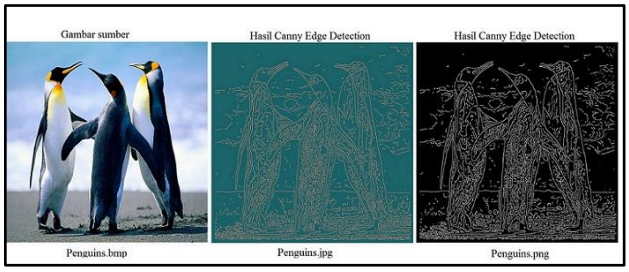
Terdapat banyak jenis algoritma deteksi tepi, seperti Canny, Sobel, Prewitt, Robert, dan Laplacian of Gaussian (LoG). Setiap algoritma deteksi tepi memiliki kelebihan dan kekurangan masing-masing. Algoritma Sobel

memiliki kelebihan yaitu peka terhadap garis diagonal dibanding garis horizontal dan vertikal, sebalikanya algoritma Prewitt lebih peka terhadap garis horizontal dan vertikal daripada garis diagonal

Diantara semua algoritma deteksi tepi yang ada, Canny merupakan algoritma deteksi tepi memberikan hasil yang paling optimal untuk digunakan pada hampir segala macam kondisi

Hal tersebut dikarenakan algoritma Canny dapat memberikan garis hasil deteksi tepi yang tipis dengan ketebalan 1 piksel sehingga dapat dinyatakan paling akurat dibandingkan algoritma deteksi tepi yang lain

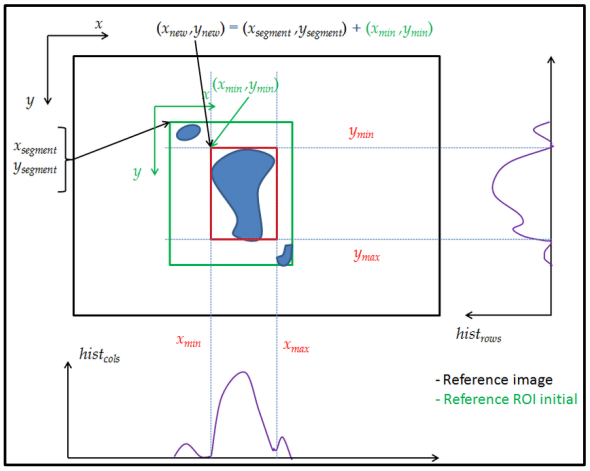
Namun algoritma Canny memiliki sebuah kekurangan, yaitu proses komputasi deteksi tepi yang dilakukan pada 11 cukup kompleks sehingga memakan waktu cukup lama



Gambar 2. 12 Illustrasi Penerapan Algoritma Canny pada Citra

2.2.4.2 Bounding Box

Bounding box merupakan kotak imajiner yang mengelilingi objek yang teridentifikasi. Bounding box sendiri berbentuk kotak yang dimana besarnya sama seperti besar objek yang teridentifikasi tersebut. Untuk membuat bounding box sendiri menggunakan koordinat piksel objek upperleft (UL), upper-right (UR), lower-left (LL), dan lower-right (LR). (Mandyartha & Fatichah, 2016).



Gambar 2. 13 Illustrasi Skema Boundingbox

2.2.5 Machine Learning

Machine Learning adalah cabang dari kecerdasan buatan, merupakan disiplin ilmu yang mencakup perancangan dan pengembangan algoritma yang memungkinkan komputer untuk mengembangkan perilaku yang didasarkan kepada data empiris, seperti dari sensor data pada basis data.

Sistem pembelajaran dapat memanfaatkancontoh (data) untuk menangkap ciri yang diperlukan dari probabilitas yang mendasarinya (yang tidak diketahui).

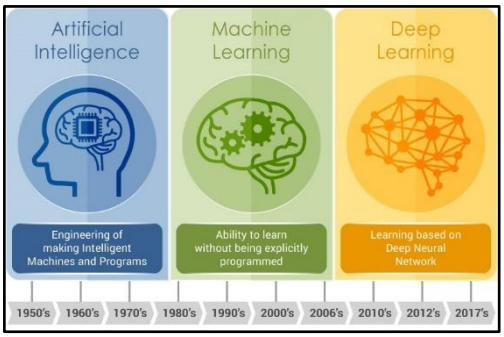
Data dapat dilihat sebagai contoh yang menggambarkan hubungan antara variabel yang diamati. (Purnamasari, 2017).

Salah satu tugas dari *Machine Learning* adalah klasifikasi. Misalkan ingin menentukan suatu hal dengan menggunakan sistem pakar, maka hal-hal yang bisa dilakukan, antara lain seperti menimbang dengan timbangan jika perlu tahu mengenai berat, atau menggunakan *computer vision* jika ingin mengenali suatu bentuk, dan hal-hal lainnya untuk mengumpulkan informasi.

Jika semua informasi penting sudah terkumpul, hal terakhir yang akan dilakukan pastinya adalah proses klasifikasi, yang nantinya akan menghasilkan *output* berupa jenis klasifikasi yang kita inginkan.

Klasifikasi pada *Machine Learning* biasanya dilakukan dengan menggunakan algoritma yang sudah sangat baik untuk digunakan dalam proses tersebut. Jika telah menggunakan algoritma *Machine Learning* untuk klasifikasi, maka

selanjutnya harus melatih algoritma yang digunakan tersebut, dan untuk melatih algoritma, maka hal yang dilakukan adalah dengan memberi algoritma tersebut data yang berkualitas (dikenal dengan istilah training set). Setiap training mempunyai fitur dan variabel target. Target variabel adalah apa yang akan diprediksikan oleh algoritma Machine Learning, yang selanjutnya *Machine Learning* akan mempelajari hubungan antara fitur dan variabel target. (Purnamasari, 2017).



Gambar 2. 14 Illustrasi Perkembangan *Artificial Intelegent, Machine Learning dan Deep Learning*

Algoritma dalam *Machine Learning* dapat dikelompokkan berdasarkan keluaran yang diharapkan dari algoritma:

1. Pembelajaran terarah (*supervised learning*) membuat fungsi yang memetakan masukan ke *output* yang dikehendaki. Misalnya pengelompokan (klasifikasi). Contoh: *k-Nearest Neighbors, Naive Bayes, Support Vector Machines, Decision Trees*.
2. Pembelajaran tak terarah (*unsupervised learning*) memodelkan himpunan masukan, seperti penggolongan (*clustering*). Contoh: *k- Means, DBSCAN*

Terdapat langkah - langkah dasar yang digunakan untuk melakukan tugas *Machine Learning* dan tugas ini sangat penting dalam mempersiapkan solusi untuk segala bentuk permasalahan dalam *Machine Learning* maupun *Deep Learning*:

1. Mengumpulkan Data Proses

Pengumpulan data dilakukan dengan mengambil contoh dari berbagai sumber informasi, seperti di Internet dan media cetak. Data yang dikumpulkan adalah data yang disebarkan secara bebas ke publik.

2. Mempersiapkan Data Masukan

   Pada hal ini data masukan yang disiapkan adalah data masukan yang sesuai dengan format yang dibutuhkan untuk analisis.

3. Menganalisis Data Masukan

   Setelah proses pertama dan kedua dilakukan, maka hal selanjutnya yang harus dilakukan adalah menganalisis data masukan dan untuk menganalisis dapat dilakukan dengan melihat pola data dan juga dengan memisahkan data berdasarkan dimensi masing-masing data.

4. Mengikutsertakan Keterlibatan Manusia

5. Melatih Algoritma

   Pada langkah ini pengguna "memberi makan" algoritma dengan data yang berkualitas, dan nantinya algoritma akan mengolah data tersebut menjadi informasi serta menyimpannya.

6. Menguji Algoritma

   Pada langkah ini hal yang dilakukan adalah melihat seberapa baik kualitas algoritma yang telah dilatih pada tahap sebelumnya.

7. Menggunakannya

   Langkah ini merupakan langkah akhir untuk algoritma yang diterapkan dalam suatu program, sehingga dapat melakukan suatu hal. Kemudian dilakukan pengecekan ulang terhadap langkah-langkah sebelumnya. (Harrington, 2012).

2.2.6 Deep Learning

Deep Learning adalah cabang dari *Machine Learning* yang dapat mengajarkan komputer untuk melakukan pekerjaan selayaknya manusia, seperti komputer dapat belajar dari proses *training* (Deng & Yu, 2013).

Dalam *Deep Learning*, sebuah komputer dapat belajar mengklasifikasi secara langsung dari gambar, suara, teks, atau video sekalipun.

Sebuah komputer seperti dilatih dengan menggunakan data set berlabel dan jumlahnya sangat besar yang kemudian dapat mengubah nilai piksel dari sebuah gambar menjadi suatu representasi internal atau *feature vector* yang dimana pengklasifikasiannya dapat digunakan untuk mendeteksi atau mengklasifikasi pola pada masukan *input*

Deep Learning merupakan suatu metode pembelajaran mesin yang memungkinkan pemodelan komputasi dalam berbagai tingkat abstraksi. Harapan dari penggunaan *Deep Learning* adalah menghasilkan representasi data yang lebih baik dari abstraksi yang bertingkat. *Deep Learning* memiliki beberapa jenis metode berdasarkan pemodelan struktur layernya.

Arsitektur *Deep Learning* bisa dijelaskan tahapan proses yang nantinya akan diolah sedemikian rupa dengan bantuan *Deep Learning* yang terdiri dari :

1. Data Storage

   Data storage merupakan penyimpanan seluruh pelatihan citra digital yang nantinya akan di olah layer.

2. Layer

   Layer disini bertugas sebagai tempat proses pengolahan citra digital yang telah ada didalam data storage untuk dibandingkan sehingga mendapatkan kesimpulan apa yang terdapat dalam citra digital tersebut.

2.2.7 Jaringan Syaraf Tiruan (JST)

Jaringan saraf tiruan (feedforward) merupakan cabang dari ilmu kecerdasan buatan. Jaringan saraf tiruan biasa digunakan untuk mencari pola data atau keterkaitan antar data yang biasanya memiliki kapasitas data yang besar dan sulit untuk dianalisis (Kusumadewi, 2003).

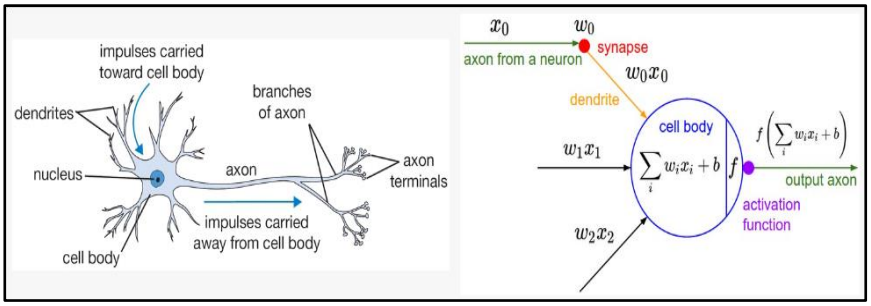
Untuk mengetahui struktur feedforward yaitu dengan melihat berapa banyak jumlah (layer) dan jumlah node di setiap lapisannya. Oleh karena itu, model feedforward ini dapat dibagi menjadi beberapa bagian, yaitu lapisan input, lapisan tersembunyi dan juga lapisan output yang saling terhubung satu dengan yang lainnya.

2.2.7.1 FFNN (*FeedForward Neural Network)*

Secara umum, proses bekerjanya jaringan neural network menyerupai cara otak manusia memproses data input sensorik, diterima sebagai neuron input. Selanjutnya neuron saling berhubungan dengan sinapsis (node), dan sinyal dari neuron bekerja secara paralel digabungkan untuk menghasilkan informasi maupun reaksi (Paul, 2018). Feed forward neural network (FFNN) merupakan salah satu model neural network yang banyak dipakai dalam berbagai bidang.

Arsitektur model FFNN terdiri atas satu lapis input, satu atau lebih lapis tersembunyi, dan satu lapis output.

Dalam model ini, perhitungan respon atau output dilakukan dengan memproses input x mengalir dari satu lapis maju ke lapis berikutnya secara berurutan. Single layer feed forward dengan satu neuron pada lapisan tersembunyi adalah jaringan syaraf yang paling dasar dan umum digunakan dalam ekonomi dan aplikasi keuangan. Kompleksitas dari arsitektur FFNN tergantung pada jumlah lapis tersembunyi dan jumlah neuron pada masing-masing lapis.



Gambar 2. 15 Illustrasi Neuron pada Manusia dan Neural Network

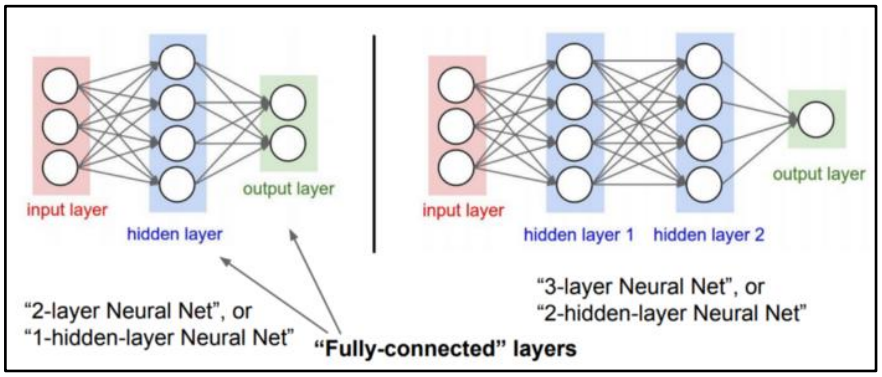
Secara umum, ada beberapa jenis arsitektur dari Jaringan Saraf Tiruan yaitu:

1. Single-Layer Feedforward Networks Di dalam Jaringan Saraf Tiruan dengan satu layer, neuron - neuron diorganisasi dalam bentuk layer - layer. Dalam bentuk paling sederhana dari Jaringan Saraf Tiruan dengan satu layer, kita mempunyai sebuah input layer dari node sumber di mana informasi diproyeksikan ke output layer dari neuron tapi tidak bisa sebaliknya. Dengan

kata lain, jaringan ini adalah tipe feedforward. Input layer dari node sumber tidak dihitung karena tidak ada perhitungan yang dilakukan.

2. Multilayer Feedforward Network Jenis kedua dari Jaringan Saraf Tiruan yang bersifat feedforward dibedakan dengan adanya keberadaan satu atau lebih hidden layer. Hidden disini berarti bagian dari Jaringan Saraf Tiruan ini secara langsung tidak terlihat dari input atau output dari jaringan tersebut. Fungsi dari hidden layer adalah untuk mengintervensi antara input eksternal dan output dari jaringan dalam cara yang berguna. Dengan menambah satu atau lebih hidden layer, jaringan dapat mengeluarkan statistik tingkat tinggi dari input.

Sumber node di input layer dari jaringan menyediakan masing-masing elemen dari pola aktivasi (vectorinput), yang merupakan sinyal input yang diaplikasikan ke neuron-neuron di layer kedua (hiddenlayer pertama). Sinyal output dari layer kedua digunakan sebagai input-input ke layer ketiga, dan seterusnya sampai ke sisa dari jaringan.



Gambar 2. 16 Illustrasi *Single layer* dan *Multi layer* Neural Network

Tabel 2. 1 Fungsi Aktivasi

FUNGSI AKTIVASI	RUMUS
SIGMOID	$f(x) = \frac{1}{(1 + e^{-x})}$
LINEAR	$H = x$
SIN	$f(x) = sin\ x$
RADIAL BEBAS	$H = e\ (-x^2)$

2.2.7.2 Algoritma Backpropagation

Algoritma pelatihan backpropagation adalah salah satu algoritma dengan multilayer perception yang pertama kali dirumuskan oleh Werbos dan dipopulerkan oleh Rumelhart dan McClelland untuk dipakai pada Neural Network. Backpropagation neural network merupakan tipe jaringan syaraf tiruan yang menggunakan metode pembelajaran terawasi (Kusumadewi, 2003).

Algoritma backpropagation juga banyak dipakai pada aplikasi pengaturan karena proses pelatihannya didasarkan pada hubungan yang sederhana, yaitu jika keluaran memberikan hasil yang salah, maka penimbang dikoreksi supaya errornya dapat diperkecil dan respon jaringan selanjutnya diharapkan akan lebih mendekati harga yang benar. Backpropagation juga berkemampuan untuk memperbaiki penimbang pada lapisan tersembunyi (hidden layer).

Algoritma Backpropagation disebut sebagai propagasi balik karena ketika jaringan diberikan pola masukan sebagai pola pelatihan maka pola tersebut menuju ke unit-unit pada lapisan tersembunyi untuk diteruskan ke unit-unit lapisan keluaran. Selanjutnya, unit-unit lapisan keluaran memberikan tanggapan yang disebut sebagai keluaran jaringan.

Saat keluaran jaringan tidak sama dengan keluaran yang diharapkan maka keluaran akan menyebar mundur (backward) pada lapisan tersembunyi diteruskan ke unit pada lapisan masukan. Oleh karenanya mekanisme pelatihan tersebut dinamakan backpropagation.

Tahap pelatihan ini merupakan langkah bagaimana suatu jaringan syaraf itu berlatih, yaitu dengan cara melakukan perubahan penimbang (sambungan antar lapisan yang membentuk jaringan melalui masing-masing unitnya). Sedangkan pemecahan masalah baru akan dilakukan jika proses pelatihan tersebut selesai, fase tersebut adalah fase mapping atau proses pengujian/testing.

Backpropagation merupakan salah satu algoritma dalam pelatihan jaringan saraf tiruan. Algoritma ini bekerja dengan cara mundur, yaitu dari lapisan keluaran menuju lapisan masukan untuk memperbarui nilai pada lapisan tersembunyi berdasarkan nilai galat yang didapat. Berikut adalah implementasi cara kerja dari algoritma Backpropagation

1. Jaringan saraf tiruan terdiri dari tiga lapisan, yaitu lapisan masukan, lapisan tersembunyi, dan lapisan keluaran. Input Layer, Hidden Layer, Output Layer.

2. Setiap lapisan terdiri dari neuron yang saling terhubung satu sama lain.

3. Pada setiap koneksi antar neuron terdapat sebuah nilai bobot yang menggambarkan kekuatan dari koneksi tersebut. Bobot tersebut akan terus diperbarui berdasarkan nilai masukan dan nilai galat dengan cara propagasi mundur.

4. Lapisan tersembunyi berfungsi untuk memperbarui nilai bobot berdasarkan nilai masukan dan nilai galat.

5. Algoritma ini bekerja dalam dua tahap, tahap pertama adalah tahap pelatihan dengan sejumlah data pelatihan yang bertujuan untuk mendapatkan nilai-nilai bobot yang sesuai. Tahap kedua adalah tahap

pengujian, dimana jaringan saraf tiruan telah memiliki nilai-nilai bobot tertentu dan siap digunakan untuk mengenali pola.

Backpropagation adalah algoritma pembelajaran terbimbing dengan banyak lapisan perceptron untuk mengubah bobot-bobot yang terhubung dengan neuronneuron yang ada pada lapisan tersembunyi. Backpropgation merupakan pelatihan jenis terkontrol dimana pola penyesuaian bobot untuk mencapai nilai kesalahan yang minimum antara keluaran hasil prediksi dengan keluaran yang nyata.

Backpropagation adalah bentuk jaringan saraf tiruan yang terdiri dari beberapa layer. JST backpropagation melatih jaringan untuk mendapatkan keseimbangan antara kemampuan jaringan untuk mengenali pola selama pelatihan serta kemampuan jaringan untuk memberikan respons yang besar terhadap pola masukan yang serupa (tapi tidak sama) dengan pola yang dipakai selama pelatihan latih.

2.2.7.2 SLFN (*Single layer Feedforward Neural Network)*

Extreme Learning Machine adalah metode pelatihan dari Jaringan Syaraf Tiruan (JST). ELM merupakan JST *feedforward* dengan *single hidden layer* atau biasa disebut dengan *Single Hidden Layer Feedforward Neural Network* (SLFNs) (Sun, et al., 2008).

Metode pembelajaran ELM dibuat untuk mengatasi permasalahan yang disebabkan oleh jaringan syaraf tiruan *feedforward* terutama dalam bidang *learning speed.* Huang dkk pada tahun 2006 menjelaskan bahwa ada 2 alasan mengapa jaringan syaraf tiruan *feedforward* memiliki *learning speed* rendah seperti berikut.

1. menggunakan slow gradient based learning algorithm dalam melakukan training.

2. semua parameter pada jaringan ditentukan secara *iterative* dengan pembelajaran tersebut.

Pada metode pembelajaran *Conventional gradient based learning algorithm* seperti *BackPropagration* (BP) dan variannya *Lavernberg Marquadt* (LM) semua parameter pada jaringan syaraf tiruan *feedforward* harus ditentukan secara *manual.* Parameter yang dimaksud adalah *input weight* dan bias. Pada ELM parameter seperti *input weight* dan bias dipilih secara acak, sehingga ELM mempunyai *learning speed* yang baik dan dapat memberikan *good generalization performance* (Huang, et al,. 2006)

1. Normalisasi Data:

   Normalisasi data dilakukan karena rentang nilai nilai *input* tidak sama. Proses normalisasi data adalah proses *tranformasi* nilai suatu data tertentu dalam rentang nilai tertentu. *Input* data akan diproses ke nilai *output* terkecil sehingga data yang digunakan harus disesusaikan agar dapat diproses untuk mendapatkan nilai normalisasi yang kecil. Pada proses normalisasi ini digunakan metode *Min-Max*. Metode *Min-Max* merupakan teknik normalisasi sederhana yang melakukan proses tranformasi *linier* terhadap data aktual. Kelebihan metode *Min-Max* yaitu perbandingan nilai antara sebelum dan sesudah proses normalisasi akan seimbang. Persamaan 2.1 adalah proses perhitungan normalisasi data menggunakan metode *Min-Max Normalization* (Ashar, 2018) :

$$d' = \frac{d - \min}{\max - min}$$

Dengan d' adalah hasil akhir yang diperoleh dalam proses normalisasi dari d sebagai nilai aktual data dikurangi *Min* sebagai nilai terkecil pada data citra *X* dan kemudian dibagi hasil dari pengurangan *Max* sebagai nilai terbesar pada citra *X* dikurangi *Min* sebagai nilai terkecil pada data citra *X*.

2. Proses Training

Proses *training* harus terlebih dahulu dilakukan sebelum proses prediksi sebagai pembelajaran awal. Proses *training* bertujuan untuk mendapatkan nilai *output weight*. Langkah-langkah yang dilakukan dalam proses *training* adalah (Ashar, 2018):

langkah pertama adalah menginisialisasi *input weight* dan bias. Nilai ini diinisialisasi secara acak dengan rentang nilai antara -1 hingga 1.

langkah selanjutnya adalah menghitung keluaran *hidden layer* (H_{init}). Persamaan 2.2 berikut untuk menghitung keluaran di *hidden layer* (Ashar, 2018).

$$H_{init_{train}} = X_{train}.W^T + b$$

Dengan $H_{init_{train}}$ adalah matriks hasil keluaran *hidden layer* untuk proses *training*, X_{train} sebagai *input* data menggunakan data *training*, W^T sebagai *transpose input weigh* dan *b* sebagai nilai bias.

Setelah nilai keluaran *hidden layer* atau H_{init} diperoleh, selanjutnya dihitung menggunakan fungsi aktivasi *sigmoid* biner, fungsi aktivasi ini sangat cocok untuk menyelesaikan permasalahan yang kompleks dan bersifat non-linier. Rumus fungsi aktivasi *sigmoid* biner ditunjukkan dengan persamaan 2.3.

$$H = \frac{1}{1+\exp\left(-H_{init_{train}}\right)}$$

Dengan H sebagai fungsi aktivasi *sigmoid* biner, exp adalah eksponential dan $H_{init_{train}}$ sebagai matriks keluaran *hidden layer* pada proses *training*.

Menghitung *output weight*. Agar mendapatkan hasil *output weight*, hal pertama yang harus dilakukan adalah mensptranspose matriks keluaran *hidden layer* dengan fungsi aktivasi. Setelah itu, matriks *transpose* tersebut dikalikan dengan matriks hasil keluaran *hidden layer* dengan fungsi aktivasi *sigmoid* biner biasa disebut matriks H. Setelah itu, langkah berikutnya adalah menghitung nilai *invers* dari matriks yang diperoleh sebelumnya. Setelah itu menghitung matriks *Moore-Penrose Generalized Inverse* dari hasil keluaran *hidden layer* dengan fungsi aktivasi. Persamaan 2.4 berikut ini untuk mencari matriks *Moore-Penrose Generalize Inverse.*

$$H^+ = (H^T H)^{-1} H^T$$

Dengan H^+ sebagai Moore-Penrose Generalize Inverse, H^T sebagai matriks H transpose, H sebagai matriks H keluaran hidden layer

Persamaan 2.5 berikut ini untuk menghitung nilai *output weight*.

$$\beta = H^{+}.Y$$

Dengan β sebagai matriks *output weight*, H^{+} sebagai matriks *Moore-Penrose Generalize Invers* dari matriks H dan Y sebagai matriks target.

3. Proses Testing

Pada proses ini bertujuan untuk mengevaluasi metode ELM dari hasil proses *training* sebelumnya. Proses *testing* dilakukan *input weight*, bias dan *output weight* yang didapatkan dari proses *training*. Berikut langkah-langkah proses *testing*.

Langkah pertama adalah menginisialisasi *input weight* dan bias yang telah didapatkan dari proses *training*.

Keluaran di *hidden layer* dihitung menggunakan persamaan 2.6 dengan data *testing* dan dihitung menggunakan fungsi aktivasi menggunakan fungsi aktivasi menggunakan persamaan 2.7 berikut.

$$H_{init_{test}} = X_{test}.W^{T} + b$$

Dengan $H_{init_{test}}$ sebagai matriks hasil keluaran *hidden layer* untuk proses *Training*, X_{test} sebagai *input* data menggunakan data *training*, W^{T} sebagai *Transpose Input weight* dan b sebagai nilai bias.

$$H = \frac{1}{1+\exp\left(-H_{init_{test}}\right)}$$

Dengan H sebagai fungsi aktivasi *sigmoid* biner, exp sebagai eksponential dan $H_{init_{test}}$ sebagai matriks keluaran *hidden layer* pada proses *testing*.

Nilai *output weight* yang telah didapatkan pada proses *training* digunakan untuk menghitung keluaran *output layer* yang merupakan hasil prediksi. Persamaan 2.8 digunakan untuk menghitung nilai *output layer*.

$$\hat{Y} = H.\beta$$

Dengan Ŷ sebagai *output layer* yang merupakan hasil prediksi, β sebagai nilai *output weight* yang didapatkan dari proses *training* dan H sebagai keluaran di *hidden layer* dihitung dengan fungsi aktivasi.

Langkah terakhir adalah menghitung nilai *error* semua *output layer* nilai *error* ini menunjukkan nilai kesalahan dari hasil prediksi yang didapatkan.

4. Proses Denormalisasi

Pada proses ini bertujuan untuk mengevaluasi metode ELM dari hasil proses *training* sebelumnya. Proses *testing* dilakukan *input weight*, bias dan *output weight* yang didapatkan dari proses *training*. Berikut langkah-langkah proses *testing*.

Langkah pertama adalah menginisialisasi *input weight* dan bias yang telah didapatkan dari proses *training*.

Keluaran di *hidden layer* dihitung menggunakan persamaan 2.6 dengan data *testing* dan dihitung menggunakan fungsi aktivasi menggunakan fungsi aktivasi menggunakan persamaan 2.7 berikut.

$$H_{init_{test}} = X_{test}.W^T + b$$

Dengan $H_{init_{test}}$ sebagai matriks hasil keluaran *hidden layer* untuk proses *Training*, X_{test} sebagai *input* data menggunakan data *training*, W^T sebagai *Transpose Input weight* dan b sebagai nilai bias.

$$H = \frac{1}{1+\exp\left(-H_{init_{test}}\right)}$$

Dengan H sebagai fungsi aktivasi *sigmoid* biner, exp sebagai eksponential dan $H_{init_{test}}$ sebagai matriks keluaran *hidden layer* pada proses *testing*.

3. Nilai *output weight* yang telah didapatkan pada proses *training* digunakan untuk menghitung keluaran *output layer* yang merupakan hasil prediksi. Persamaan 2.8 digunakan untuk menghitung nilai *output layer*.

$$\hat{Y} = H.\beta$$

Dengan $\hat{Y}$ sebagai *output layer* yang merupakan hasil prediksi, β sebagai nilai *output weight* yang didapatkan dari proses *training* dan H sebagai keluaran di *hidden layer* dihitung dengan fungsi aktivasi.

Langkah terakhir adalah menghitung nilai *error* semua *output layer* nilai *error* ini menunjukkan nilai kesalahan dari hasil prediksi yang didapatkan.

5. Proses MSE Mean Error Square

Mean Square Error (MSE) digunakan untuk mengevaluasi prediksi. MSE memiliki kesederhanaan dalam perhitungannya. Persamaan 2.10 berikut ini untuk menghitung nilai *error* pada hasil prediksi.

$$MSE = \frac{\sum_{i=1}^{n} e_i^2}{n} = \frac{\sum_{i=1}^{n} (y_i - t_i)^2}{n}$$

Dengan *n* sebagai jumlah data, *i* sebagai [1,2,…, n], *n* adalah keseluruhan jumlah data, *e* adalah *error, y* sebagai nilai *output* (prediksi) dan *t* sebagai nilai aktual.