MODUL
SISTEM CERDAS

Pegangan Mahasiswa

LEARNING

Jurusan Teknik Elektro
Fakultas Teknik
Universita Negeri Malang

Disusun oleh :
Ria Febrianti
Dr. Hakkun Elmunsyah, S.T., M.T
Dr. Eng. Anik Nur Handayani, S.T., M.T.

KATA PENGANTAR

Bissmillahirahmannirahim,

Puji syukur kehadirat Tuhan yang Maha Esa atas limpahan rahmat serta hidayahnya, sehingga penulis dapat menyelesaikan modul pembelajaran berbantuan SIPEJAR UM dan *e-collab classroom* pada mata kuliah Sistem Cerdas dengan topik materi *Learning* untuk mahasiswa Program Studi S1 Pendidikan Teknik Elektro, Jurusan Teknik Elektro, Fakultas Teknik, Universitas Negeri Malang. Penyajian materi pada modul pembelajaran *Learning* ini merujuk pada Rencana Perkuliahan Semester untuk mata kuliah Sistem Cerdas (PTEL667)

Penulisan modul *Learning* ini diharapkan dapat menumbuhkan motivasi belajar mahasiswa dan meningkatkan efektivitas pelaksanaan perkuliahan Sistem Cerdas di Jurusan Teknik Elektro, Fakultas Teknik, Universitas Negeri Malang. Selain itu, mahasiswa diharapkan mampu menguasai secara tuntas materi *Learning* yang tertulis pada modul pembelajaran terutama pada pelaksanaan pembelajaran secara daring, namun pemanfaatan modul ini juga dapat menunjang kegiatan perkuliahan secara tatap muka. Dengan demikian akan tercapai tujuan perkuliahan Sistem Cerdas yang telah ditetapkan

Penulisan modul *Learning* ini tentu tidak terlepas dari bantuan dan dukungan dari berbagai pihak. Penulis mengucapkan terimakasih kepada pihak yang telah membantu proses penyusunan modul pembelajaran *Learning* ini. Selain itu, penulis menyadari sepenuhnya bahwa isi ataupun penyajian dari modul pembelajaran ini masih jauh dari kata sempurna. Maka dari itu, penulis

mengharapkan masukan berupa kritik dan saran yang dapat membangun untuk perbaikan dan pengembangan modul selanjutnya.

Malang, Juni 2021

Penulis

DAFTAR ISI

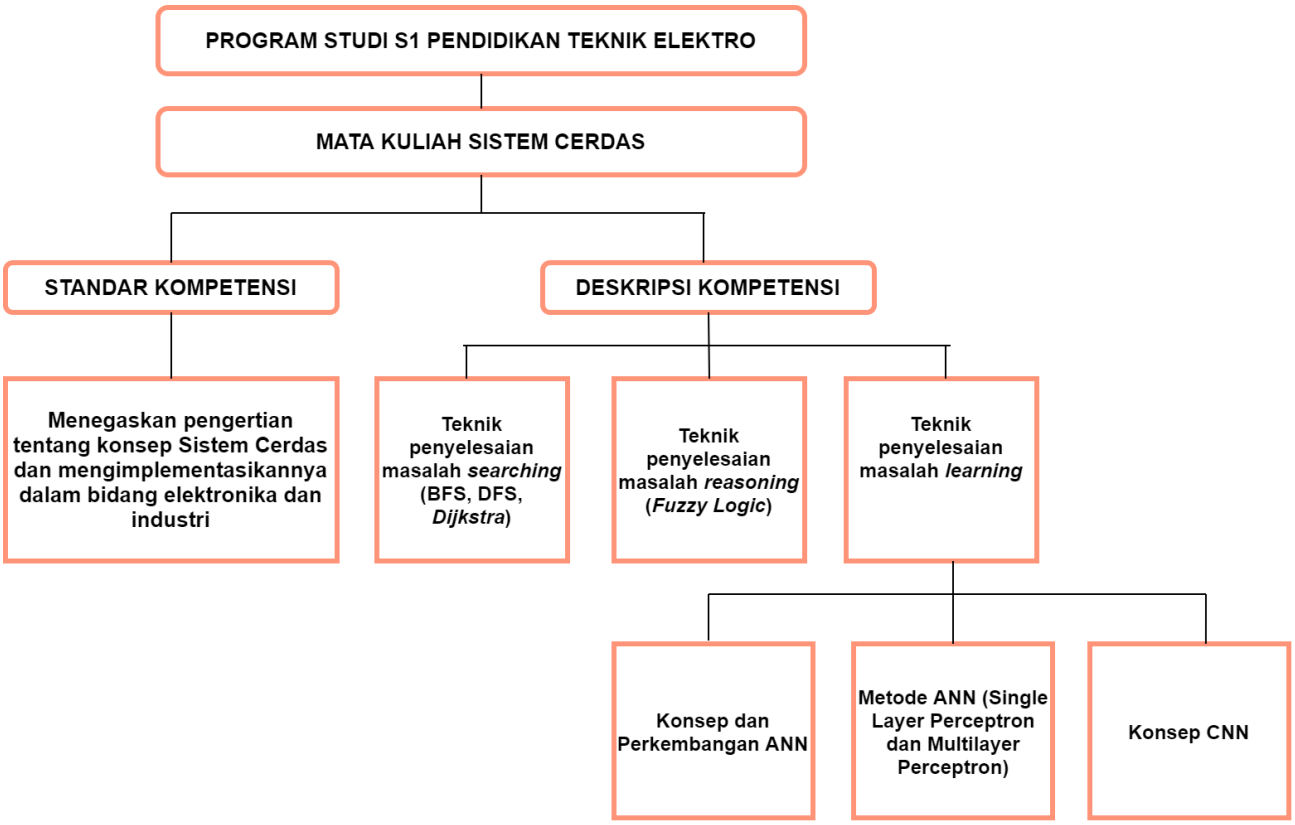
DAFTAR GAMBAR

Gambar **Halaman**

DAFTAR TABEL

Tabel **Halaman**

PETA KEDUDUKAN MODUL



1 PENDAHULUAN

- Standar Kompetensi
- Deskripsi
- Prasyarat
- Petunjuk Penggunaan Modul
- Tujuan Akhir
- Indikator Penguasaan Kompetensi

Standar Kompetensi

Menganalisis tahapan Teknik penyelesaian masalah *learning* menggunakan ANN, CNN, dan mengimplentasikan dalam bidang teknik elektronika.

Deskripsi

Modul ini merupakahan bahan ajar virtual yang dapat membantu mahasiswa untuk mempelajari salah satu jenis teknik penyelesaian masalah *learning*, yaitu dengan metode ANN dan CNN. Modul pembelajaran *learning* ini terdiri dari tiga bab yaitu pendahuluan, pembelajaran, dan evaluasi. Pada bab pertama (pendahuluan), berisi penjelasan standar kompetensi, deskripsi modul, prasyarat, petunjuk penggunaan modul, tujuan akhir, dan indikator penguasaan kompetensi.

Bab kedua (Pembelajaran) berisi tentang penyajian materi pembelajaran yang dilengkapi dengan materi pengayaan. Materi yang disajikan pada modul *learning* ini meliputi pemodelan ANN, pemodelan CNN, pemrograman ANN dan CNN pada *google colaboratory*

Bab ketiga (evaluasi), berisi tes individu sebagai bentuk evaluasi penguasaan mahasiswa terhadap materi yang disajikan. Tes individu yang disajikan berisi soal tentang penerapan *learning* (ANN dan CNN) dengan penyelesaian masalah secara analisis. Setelah menyelesaikan modul pembelajaran, diharapkan mahasiswa mampu menerapkan teknik penyelesaian masalah pada sistem cerdas menggunakan *learning* (ANN dan CNN).

Prasyarat

Untuk mendukung proses pembelajaran, mahasiswa diharapkan mampu menguasai beberapa pengetahuan seperti:

1. Mengenal materi pembelajaran penerapan *Artificial Neural Network*
2. Mengenal bahasa pemrograman *python*

Petunjuk Penggunaan Modul

1. Pelajari daftar isi dan peta kedudukan modul untuk mengetahui modul *learning* yang akan dipelajari.
2. Pelajari uraian materi yang disajikan pada setiap kegiatan pembelajaran.
3. Pahami kesulitan yang ditemukan dari uraian materi yang disajikan, tanyakan pada instrukstur/dosen pada saat kegiatan pembelajaran.
4. Pahami dan selesaikan soal evaluasi yang disajikan pada modul.

Tujuan Akhir

1. Menelaah konsep *learning* (ANN dan CNN).
2. Mengidentifikasi ANN *single layer* dan *multilayer perceptron*.
3. Menguraikan kasus kehidupan sehari-hari dengan ANN dan CNN.
4. Merepresentasikan ANN dan CNN dengan bahasa pemrograman *python* pada *google colaboratory*.

Indikator Penguasaan Kompetensi

Indikator penguasaan kompetensi dari tujuan akhir, yaitu mahasiswa dapat:

1. Mengetahui konsep *learning* (ANN dan CNN).
2. Menganalisis *learning* ANN *single layer* dan *multilayer perceptron*.
3. Megetahui langkah merepresentasikan *learning* (ANN dan CNN) pada *google colaboratory*.

2 PEMBELAJARAN

Konsep ANN

Metode ANN (*Single Layer Perceptron* dan *Multilayer Perceptron*)

Konsep CNN

Pemrograman ANN dan CNN dengan *Google Colaboratory*

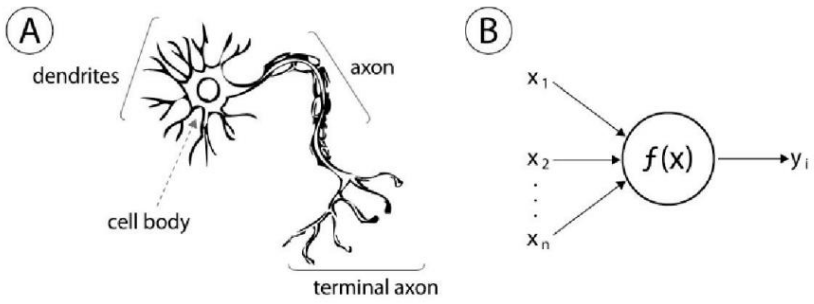
LEARNING

Jabaran Materi

1. Konsep ANN
2. Metode ANN (*single layer perceptron* dan *multilayer perceptron*)
3. Konsep CNN
4. Pemrograman ANN dan CNN pada *google colaboratory*

1. Konsep ANN

Artificial Neural Network (ANN) atau dalam bahasa Indonesia Jaringan Syaraf Tiruan (JST) merupakan salah satu algoritma *machine learning*. ANN termasuk algoritma sistem cerdas yang digunakan untuk mengolah informasi dari perkembangan generalisasi model matematika. Prinsip kerja ANN terinspirasi dari sistem jaringan saraf (*neural network*) manusia (Gershenson, 2003). Para ilmuan menciptakan algoritma matematis yang bekerja menyerupai pola kerja saraf (*neuron*) tersebut, maka digunakanlah nama *Artificial Neural Network*. Gambar 1 berikut menunjukkan kemiripan arsitektur ANN dengan jaringan saraf pada tubuh manusia:



Gambar 1 Jaringan Saraf Manusian vs ANN

Sumber: Ryandhi (2017)

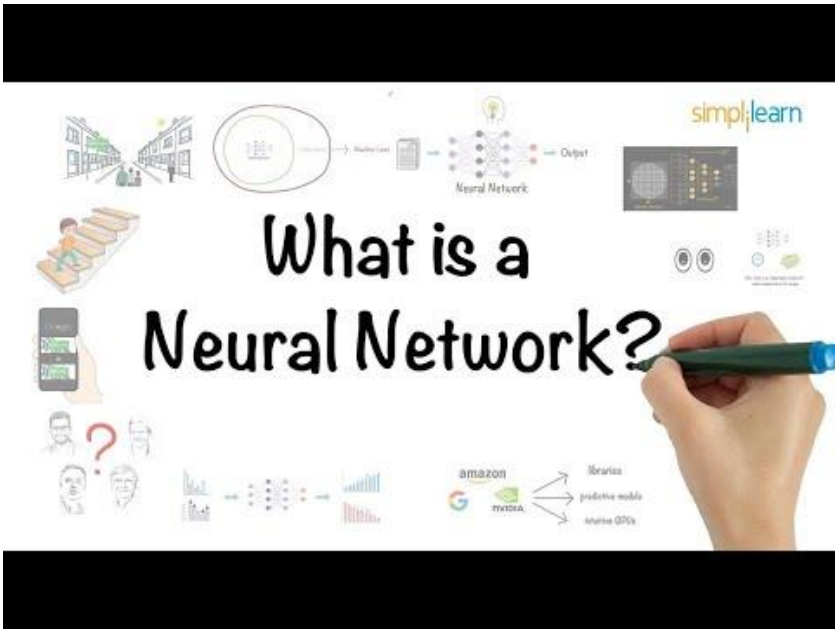
Label A pada Gambar 1 merupakan struktur susunan sel *neuron* pada tubuh manusia. Fungsi dari sel *neuron* yaitu sebagai pengantar informasi dari satu sel ke sel lainnya dengan urutan sebagai berikut:

- Dendrit adalah bagian yang berfungsi menerima rangsangan atau informasi.
- Badan sel berfungsi untuk menerima dan mengakumulasikan rangsangan dari dendrit, kemudian memproses informasi tersebut dan meneruskANNya ke akson.
- Akson bertugas meneruskan rangsangan yang telah diproses badan sel ke *neuron* lain.

Label B pada Gambar 1 merupakan struktur ANN yang juga mempunyai tiga bagian didalamnya, yaitu:

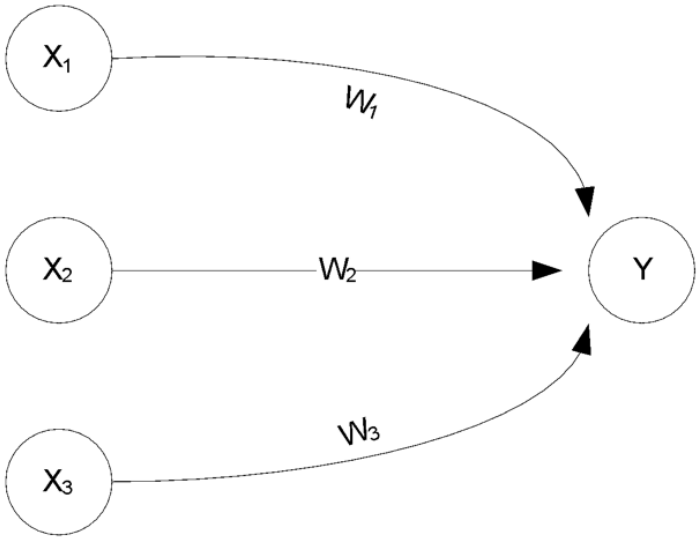
- *Input layer* (x): lapisan yang membawa data masuk ke dalam sistem dan menerima informasi menggunakan bobot yang ditentukan, kemudian diproses pada *layer* berikutnya.
- *Neuron*: pada lapisan ini informasi dengan bobot yang telah diperoleh, kemudian dikumpulkan dan diakumulasikan. Hasil penjumlahan tersebut dibandingkan dengan *threshold* yang ditentukan sebagai nilai aktifasi.
- *Output layer* (y): lapisan terakhir dari *neuron* yang menghasilkan *output* sistem apabila informasi yang masuk memenuhi syarat.

Untuk mengetahui lebih jelas tentang cara kerja ANN dapat dilihat pada Video 1.



Video 1 Cara Kerja ANN

ANN merupakan sistem adaptif yang dapat mengubah struktur dalam memecahkan suatu masalah berdasarkan informasi internal maupun eksternal (Nurhikmat, 2018). Gambar 2 merupakan contoh ANN, dimana Y menerima *input* dari *neuron* X_1, X_2, dan X_3 dengan bobot hubungan masing-masing W_1, W_2, dan W_3. Ketiga *impuls neuron* yang ada dijumlahkan **net = X_1W_1 + X_2W_2 + X_3W_3**

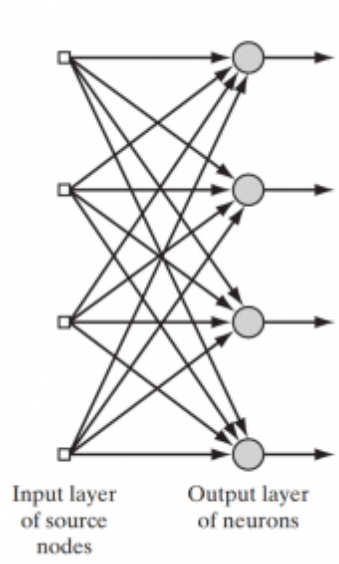


Gambar 2 Contoh ANN

Perancangan ANN ditentukan pada 3 hal yaitu: (1) fungsi aktivasi, (2) pola hubungan antar *neuron* (disebut arsitektur jaringan), dan (3) metode untuk menentukan bobot penghubung/metode training atau leraning algoritma. Adapun arsitektur jaringan pada ANN dibagi menjadi tiga macam yaitu *single layer, multilayer,* dan *recurrent network*. Penjabaran tentang arsitektur jaringan ANN adalah sebagai berikut:

a. Single Layer

Pembentukan ANN terdiri dari *neuron* yang disusun dalam bentuk lapisan (*layer*). Pemodelan ANN yang paling sederhana yaitu *single layer,* dimana *input layer* yang berasal dari sumber *node* diproyeksikan langsung ke *output layer* dari *neuron*. Pemodelan ini merupakan jenis jaringan *feedforward* yang dapat dilihat pada Gambar 3, gambar tersebut menunjukkan nilai *input* dan *output* yang memiliki 4 *node*. Adapun yang dimaksud dengan *single layer* yaitu *output* dari jaringan, sedangkan *input* tidak berpengaruh karena pada saat proses *learning* tidak terjadi proses komputasi. Contoh penerapan *single layer* dapat ditemukan pada jaringan *McCulloch-Pitts, hebb, dan perceptron*. Implementasi *single layer* pada modul ini hanya berfokus pada jaringan *perceptron*.

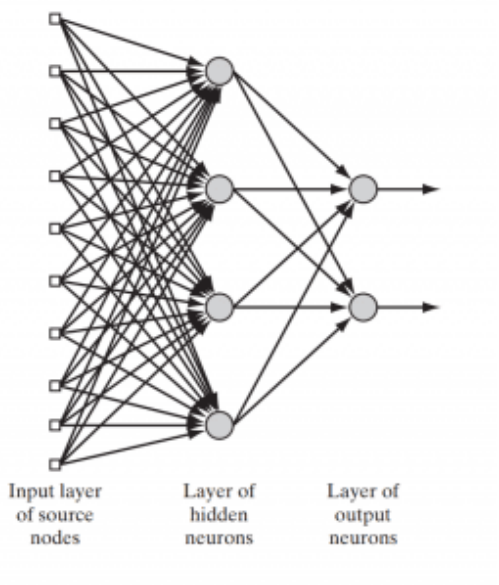


Gambar 3 Single *Layer*

Sumber: Yanuar (2018)

b. *Multilayer*

Pada pemodelan *single layer* apabila terdapat tambahan satu atau dua *hidden layer* maka jaringan akan terganggu, karena *input* dan *output* dari jaringan tidak dapat melihat *hidden layer* yang dimasukkan. Berdasarkan hal tersebut, maka diperlukan suatu jaringan yang dapat melihat dan menampung data tambahan yang bernama *multilayer*. Cara kerja dari *multilayer* yaitu *input layer* menyuplai *input vektor* pada jaringan, kemudian *input* tersebut melakukan komputasi pada *layer* yang kedua, selanjutnya *output* dari *layer* yang kedua digunakan sebagai *input* dari *layer* yang ketiga dan seterusnya. Ilustrasi pemodelan *multilayer* dapat dilihat pada Gambar 4.

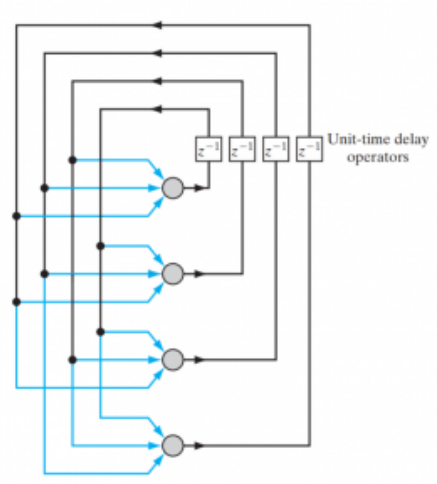


Gambar 4 *Multilayer*

Sumber: Yanuar (2018)

c. *Recurrent Network*

Recurrent network terbentuk karena pada jaringan single *layer* dan *multilayer* memiliki *feedback* untuk dirinya sendiri pada setiap *loop* (perulangan) jaringanya. Jaringan pada *recurrent network* tidak memerlukan *feedback* dari dirinya sendiri melainkan *feedback* dari *input* yang digunakan. Gambar 5 merupakan ilustrasi dari *recurrent network.*



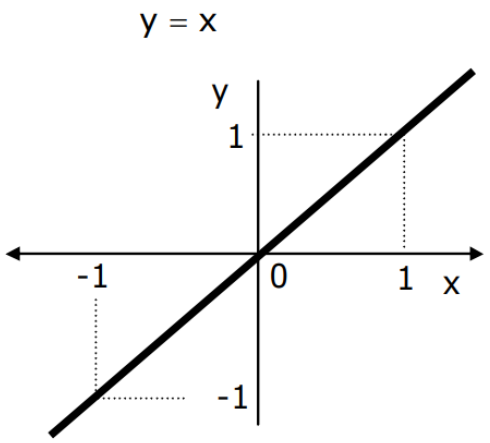
Gambar 5 *Recurrent Network*

Sumber: Yanuar (2018)

Untuk menentukan keluaran *neuron* maka diperlukan suatu fungsi aktivasi. Fungsi aktivasi diartikan sebagai net masukan (kombinasi linier masukan dan bobotnya). ANN menggunakan fungsi aktivasi untuk membatasi keluaran Y agar sesuai dengan batasan sinyal *output*nya. Selain itu fungsi ini bertujuan untuk menentukan apakah *neuron* diaktifkan atau tidak. Berikut merupakan jenis-jenis fungsi aktivasi pada ANN:

a. Fungsi Linear

Fungsi aktivasi linear memiliki nilai *output* sama dengan nilai *input*. Rumus dari fungsi aktivasi ini yaitu: y = x (Gambar 6). Fungsi linear dianggap tidak menggunakan fungsi aktivasi karena fungsi ini tidak didapat perhitungan apapun yang dilakukan pada nilai keluaran.



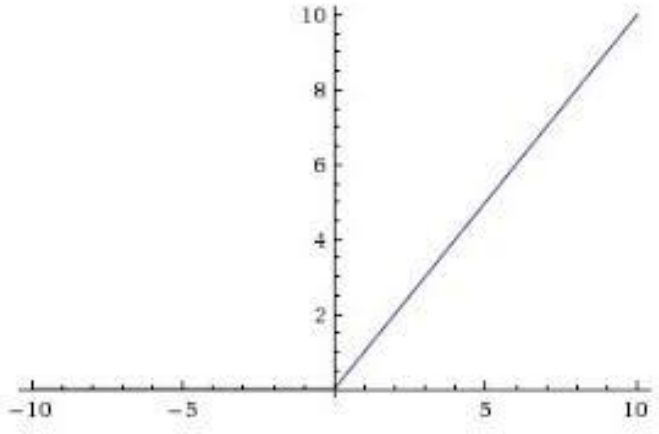
Gambar 6 Fungsi Aktivasi Linear

Sumber: Yanuar (2018)

b. Fungsi *Rectified Linear Unit* (ReLU)

Pada fungsi ini *input* dari *neuron* berupa bilangan negatif, kemudian fungsi tersebut diterjemahkan ke dalam nilai 0 dan jika *input* bernilai positif maka *output* dari *neuron* adalah nilai aktivasi itu sendiri. Kelebihan dari fungsi ReLU yaitu dapat mempercepat proses konfigurasi yang dilakukan dengan *Stochastic Gradient Descent* (SGD). Namun fungsi ini juga memiliki kelemahan yaitu

dapat membuat unit mati apabila *learning rate* yang diinisialisasi terlalu tinggi, jika *learning rate* yang diinisialisasi secara tepat maka hal tersebut tidak menjadi masalah. fungsi aktivasi ReLU ditunjukkan pada Gambar 7.



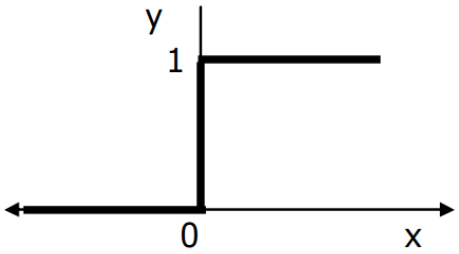
Gambar 7 Fungsi Aktivasi ReLU

Sumber: Aina (2018)

c. Fungsi Undak Biner (*Layer*)

Fungsi aktivasi *layer* (undak biner) sering digunakan pada jaringan dengan lapisan tunggal. Fungsi ini sering dipakai untuk mengkonversikan *input* dari suatu variabel yang bernilai kontinu ke suatu *output* biner (0 atau 1) (Gambar 8). Fungsi *layer* dirumuskan sebagai berikut:

$$y = \begin{cases} 0, jika\ x\ \leq 0 \\ -1, jika\ x > 0 \end{cases}$$



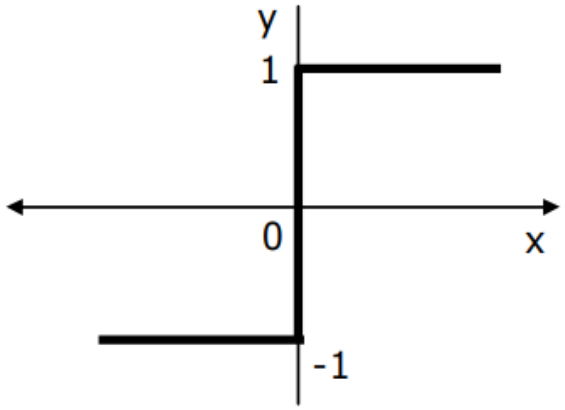
Gambar 8 Fungsi Aktivasi *Layer*

Sumber: Aina (2018)

d. Fungsi *Bipolar* (Symmetrical *Layer*)

Fungsi *bipolar* menggunakan nilai ambang atau sering juga disebut dengan nama fungsi nilai ambang (*threshold*). Adapun *output* yang dihasilkan berupa 1, 0, atau -1 (Gambar 9). Fungsi *Bipolar* dirumuskan sebagai berikut:

$$y = \begin{cases} 1, jika\ x > 0 \\ 0, jika\ x\ =\ 0 \\ -1, jika\ x\ <\ 0 \end{cases}$$



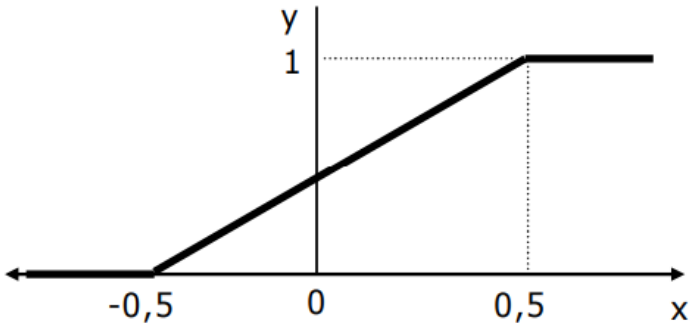
Gambar 9 Fungsi Aktivasi *Bipolar* (Symmetrical *Layer*)

Sumber: Aina (2018)

e. Fungsi *Saturating Linear*

Fungsi aktivasi ini akan bernilai 0 jika *input*nya kurang dari -½ dan bernilai 1 jika *input*nya lebih dari ½. Sedangkan jika nilai *input* terletak antara -½ maka *output*nya bernilai sama dengan nilai *input* ditambah ½ (Gambar 10). Fungsi aktivasi Saturating Linear dapat dirumuskan sebagai berikut:

$$y = \begin{cases} 1; & \text{Jika } x \geq 0{,}5 \\ x\ +\ 0{,}5; & \text{Jika } -0{,}5\ x \leq 0{,}5 \\ 0; & \text{Jika } x \leq -0{,}5 \end{cases}$$

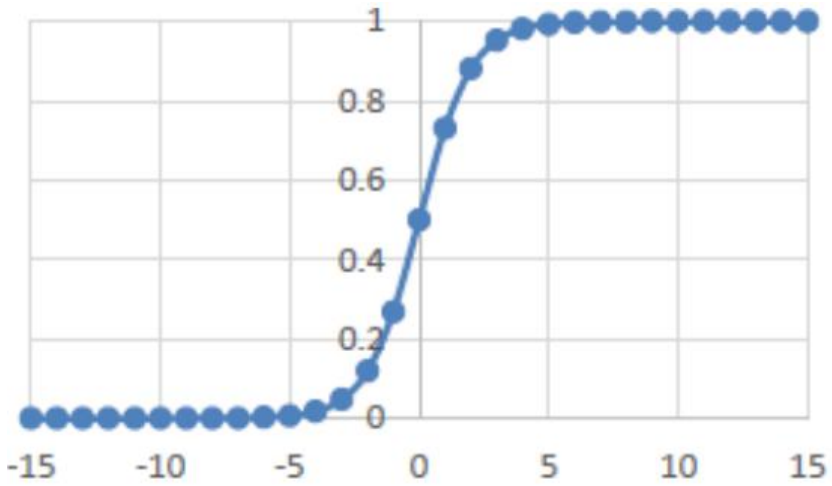


Gambar 10 Fungsi Aktivasi Saturating Linear

Sumber: Aina (2018)

f. Fungsi Aktivasi *Sigmoid*

Fungsi aktivasi *sigmoid* merupakan fungsi *non linear*. Grafik fungsi aktivasi *sigmoid* dapat dilihat pada Gambar 11.



Gambar 11 Fungsi Aktivasi *Sigmoid*

Sumber: Aina (2018)

Fungsi aktivasi *sigmoid* mentransformasikan *range* nilai dari *input* x menjadi antara 0 dan 1 dalam bentuk distribusi fungsi. Sehingga fungsi *sigmoid* dapat dituliskan sebagai berikut:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Pada dasarnya fungsi *sigmoid* memiliki kelemahan yaitu *range* nilai *output* dari fungsi *sigmoid* tidak terpusat pada angka nol. Hal tersebut menyebabkan terjadinya proses *backpropagation* yang

tidak ideal, selain itu *weight* (bobot) pada JST tidak terdistribusi rata antara nilai positif dan nilai negatif. Sehingga nilai bobot akan banyak mendekati nilai ekstrim 0 atau 1. Komputasi nilai propagasi pada fungsi *sigmoid* menggunakan perkalian, maka nilai ekstrim tersebut akan menyebabkan efek *saturating gradients*. Jka nilai bobot cukup kecil, maka lama kelamaan nilai bobot akan mendekati salah satu ekstrim sehingga memiliki gradien yang mendekati nol. Apabila hal tersebut terjadi, maka *neuron* tidak dapat mengalami *update* yang signifikan dan akan nonaktif.

2. Metode ANN

Metode algoritma yang baik dan sesuai dalam melakukan pengenalan pola-pola gambar adalah algoritma *Perceptron* dan B*ackpropagation*. *Perceptron* merupakan salah satu jaringan syaraf tiruan yang memiliki bentuk paling sederhana dan digunakan dalam mengklasifikasikan pola khusus yang biasa disebut dengan *linearly separable*. Teknik *perceptron* ditemukan oleh psikolog yang bernama Frank Rosenblatt di penghujung tahun 1950-an. Pembagian arsitektur jaringan pada *perceptron* dibagi menjadi dua yaitu *single layer perceptron* dan *multilayer perceptron*.

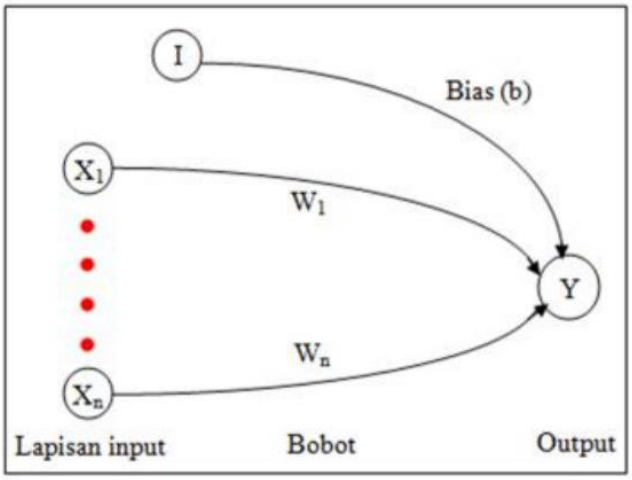
a. *Single Layer Perceptron*

Single layer perceptron merupakan bentuk terkecil dari sebuah ANN. Penyelesaian masalah menggunakan *single layer perceptron* hanya mampu digunakan untuk permasalahan yang bersifat *lineary separable* dimana batas *output* didefinisikan jelas dan hanya memiliki dua kemungkinan. *Single layer perceptron* termasuk *supervised learning* yaitu metode pembelajarANNya

dilakukan melalui contoh-contoh. Pembagian arsitektur jaringan pada *perceptron* dibagi menjadi dua yaitu *single layer perceptron* dan *multilayer perceptron*.

1. **Arsitektur jaringan single *layer perceptron***

Single layer perceptron dapat dikatakan sebagai salah satu Teknik jaringan saraf tiruan yang sederhana. Teknik tersebut hanya mempunyai sebuah lapisan *input* dan sebuat unit *output* seperti pada Gambar 12, pada gambar tersebut terdapat bias (b) yaitu unit yang aktivasinya selalu satu dan mempunyai bobot (W). Jadi arsitektur pada jaringan pada *single layer perceptron* terdiri dari *input layer* dan *output layer*.



Gambar 12 Arsitektur Jaringan Single *Layer Perceptron*

Sumber: Puspitorini (2012)

- **Fungsi aktivasi (*Layer*)**

Fungsi aktivasi yang digunakan pada *single layer perceptron* adalah fungsi aktivasi *layer*. Arsitektur *single layer perceptron* sering menggunakan fungsi undak untuk mengkonversi *input* dari suatu *variabel* yang bernilai kontinu ke suatu *output* biner (Gambar 8).

- **Algoritma pelatihan**

Penggunaan algoritma pelatihan pada *single layer perceptron* bertujuan untuk melatih jaringan syaraf tiruan, yaitu dengan cara mengajarinya contoh-contoh kasus/pola sampai jaringan syaraf tiruan berhasil mengenali pola tersebut. Apabila *output* yang dihasilkan jaringan tidak sesuai dengan target yang diharapkan maka bobotnya di-update. Hal tersebut terus-menerus dilakukan sampai tidak ada lagi bobot yang berubah pada setiap pasangan latihan sensor dan terget. Bobot-bobot terakhir yang diperoleh pada saat pelatihan jaringan syaraf tiruan yang akan digunakan pada saat pengaplikasian (dengan menggunakan algoritma aplikasi *perceptron*).

Adapun algoritma pelatihan *perceptron* sebagai berikut:

- Melakukan inisialisasi semua bobot dan bias (untuk sederhananya semua bobot dan bobot bias diset sama dengan nol). Set *learning* rate: α = 1, ($0 < \alpha \leq 1$)
- Selama kondisi berhenti dan bernilai false, selanjutnya dilakukan langkah-langkah sebagai berikut:
  1. Pada setiap pasangan pembelajaran s – t, dilakukan:
     a. Set *input* dengan nilai sama dengan vektor *input*, yaitu $x_i = s_i$
     b. Menghitung respon untuk unit *output*, menggunakan persamaan:

$$y\_in = b + \sum_i x_i w_i$$

$$y = \begin{cases} 1, & \text{jika } y\_in > \theta \\ 0, & \text{jika } -\theta \leq y\_in \leq \theta \\ -1, & \text{jika } y\_in < \theta \end{cases}$$

     c. Melakukan perbaikan bobot dan bias jika terjadi *error*:
        Jika $y \neq t$ maka:

$$w_i(\text{baru}) = w_i(lama) + \alpha * t * x_i$$

b(baru) = $b(lama) + \alpha * t$

Jika tidak maka :

w_i(baru) = $w_i(lama)$

b(baru) = $b(lama)$

2. Tes kondisi berhenti: apabila tidak terdapat perubahan bobot pada (i) maka kondisi berhenti *TRUE*, namun apabila masih terdapat perubahan maka kondisi berhenti *FALSE*.

- **Algoritma Aplikasi/Pengujian**

Algoritma aplikasi/pengujian *perceptron* sebagai berikut:

1. Menerapkan algoritma pelatihan untuk mengeset bobot-bobot
2. Pada setiap vektor *input* x yang ingin diklasifikasikan, lakukan langkah 3
3. Melakukan set aktivasi unit-ubit *input* (x_i), I = 1,…, n yaitu $x_i = s_i$
4. Menghitung respons unit *output* (y):

$$y_{in} = b + \sum_i x_i w_i$$

$$y = \begin{cases} 1 \; jika \; yin > \theta \\ 0 \; jika \; -0 \leq yin \leq \theta \\ -1 \; jika \; yin < \theta \end{cases}$$

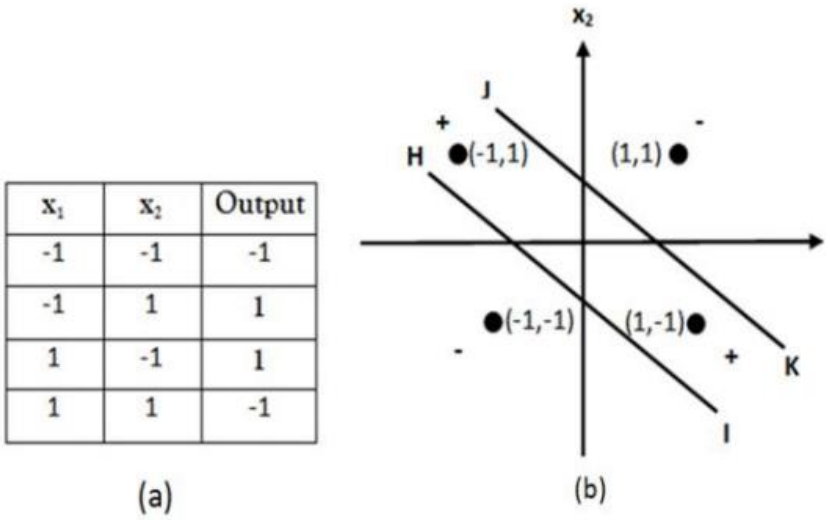
Keterangan:

s = sensor
t = target
x_i = unit inpu ke-i
s_i = unit sensor ke-i
w_i = bobot ke-i
b = bias
y = unit respons (*output*)
α = angka pembelajaran
Θ = nilai ambang
i = 1,…,n dimana nilai n adalah banyaknya unit *input*

B. *Multilayer Perceptron*

Multilayer perceptron merupakan algoritma yang mengadopsi cara kerja jaringan syaraf pada makhluk hidup. Penelitian yang dilakukan oleh Marvin Minsky dan Seymour memaparkan tentang kelebihan dan keterbatasan yang dimiliki

oleh *single layer perceptron*. Keterbatasan yang paling utama adalah ketidakmampuan *single layer perceptron* dalam membedakan pola secara linear yang tidak dapat dipisahkan. Hal tersebut sangat membatasi bidang aplikasi yang bisa diselesaikan menggunakan arsitektur *single layer perceptron*. Salah satu permasalahan sederhANNa yang tidak bisa diselesaikan dengan menggunakan arsitektur *single layer perceptron* adalah masalah pada fungsi XOR (Gambar 13).



Gambar 13 (a) Tabel fungsi logika XOR (dalam bentuk vipolar) **(b)** terdapat 2 buah garis pemisah (HI dan JK), bidang pada masalah XOR terbagi menjadi 3 bidang solusi

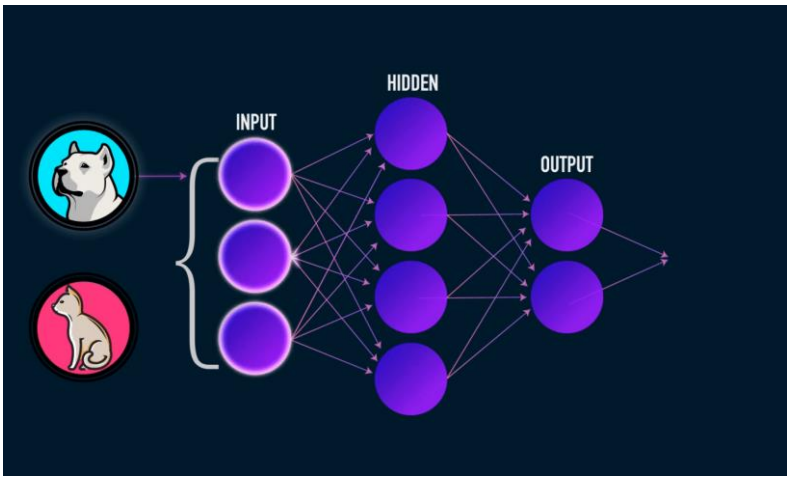
Sumber: Jaya (2018)

- **Arsitektur Jaringan Multi*layer Perceptron***

Berdasarkan (Gambar 13a) tabel fungsi logika XOR merupakan pemisah antara daerah dengan respons -1 dan +1. Sedangkan (Gambar 13b) merupakan bidang x_1x_2 dengan 4 titik *input* yaitu (1.1), (1,0), (0,1), dan (0,0). Simbol (+) dan (–) masing-masing menggambarkan daerah respons +1 dan -1. Pada gambar tersebut tidak ada sebuah garis yang memisahkan antara daerah (+) dan daerah (-). Kedua daerah tersebut membutuhkan pemisah setidaknya dua buah baris.

Ketidakmampuan *single layer perceptron* untuk menambahkan lapisan tengah/tersembunyi tidak mampu memecahkan masalah XOR.

Penambahan lapisan tengah (hidden *layer*) untuk memecahkan masalah XOR memberikan fleksibilitas pada jaringan. *Hidden layer* tersebut memberikan 2 garis yang bisa memisahkan sebuah bidang ke dalam 3 buah daerah. Sehingga arsitektur pada *multilayer perceptron* terdiri dari 3 yaitu *input, hidden,* dan *output layer.* Ilustrasi cara kerja *multilayer perceptron* ditampilan pada Gambar 14.



Gambar 14 Ilustrasi *Multilayer Perceptron*

Sumber: Puspitorini (2012)

Berikut penjelasan masing-masing *layer* (Vercellis, 2009):

- *Input layer*

Lapisan *input layer* bertugas menerima nilai masukan dari tiap *record* pada data. Jumlah simpul pada *input* sama dengan jumlah variabel predictor.

- Hidden *layer*

Lapisan hidden *layer* bertugas mentransformasikan nilai *input* didalam *network.* Pada tiap simpul hidden *layer* terhubung dengan simpul hidden *layer* sebelumnya atau dari simpul pada

input layer ke simpul hidden *layer* berikutnya atau ke simpul *output layer*.

- *Output layer*

Garis yang terhubung dengan *output layer* berasal dari *hidden layer* atau *input layer* dan mengembalikan nilai keluaran yang sesuai dengan variabel prediksi. Keluaran dari *output layer* biasanya merupakan nilai floating antara 0 sampai 1 (Kusrini & Luthfi, 2009).

- **Fungsi aktivasi**

Fungsi aktivasi yang digunakan pada *multilayer perceptron* adalah fungsi aktivasi *sigmoid*. Arsitektur *multilayer perceptron* menggunakan fungsi *sigmoid* untuk nilai *output* yang terletak pada interval 0 sampai 1. Namun, fungsi ini bisa juga digunakan oleh jaringan syaraf yang nilai *output*nya 0 atau 1 (Gambar 11).

- **Algoritma pelatihan**

Pembelajaran algoritma ini dilakukan dengan peng*updet*an bobot balik (*back propagation*). Penetapan bobot yang optimal akan berujung pada hasil prediksi yang tepat. Algoritma *multilayer perceptron* dapat dibagi ke dalam 2 bagian:

a. Algoritma pelatihan

Terdiri dari 3 tahap yaitu tahap umpan maju pola pelatihan *input*, tahap pemropagasibalikan *error*, dan tahap pengaturan bobot. Langkah pelatihan dalam algoritma *backpropagation* adalah sebagai berikut (Myatt, 2007):

1. Inisialisasi bobot jaringan secara acak (biasanya antara -0.1 sampai 1.0)
2. Pada setiap data training, *input* dihitung untuk simpul berdasarkan nilai *input* dan bobot jaringan saat itu, menggunakan rumus:

$$input_j = \sum_{i=1}^{n} O_i Wi_j + \theta_j$$

Keterangan:

Oi = *output* simpul i dari *layer* sebelumnya

w_{ij} = bobot relasi dari simpul i pada *layer* sebelumnya ke simpul j

θ_j = bias (sebagai pembatas)

3. Berdasarkan *input* pada langkah dua, selanjutnya membangkitkan *output* untuk simpul menggunakan fungsi aktivasi *sigmoid*:

$$Output = \frac{1}{1+e^{-input}}$$

4. Menghitung nilai *error* antara nilai yang diprediksi dengan nilai yang sesungguhnya menggunakan rumus:

$$Error_j = Output_j \,.\, (1 - Output_j) \,.\, (Target_j - Output_j)$$

Keterangan:

$Output_j$ = *output* aktual dari simpul j

$Target_j$ = nilai target yang sudah diketahui pada data *training*

5. Setelah nilai *error* dihitung, selanjutnya dibalik ke *layer* sebelumnya (back propagated). Perhitungan nilai *error* pada hidden *layer* dapat menggunakan rumus:

$$Error_j = Output_j\,(1 - Output_j) \sum_{k=1}^{n} Error_k\, W_{jk}$$

Keterangan:

$Output_j$ = *Output* aktual dari simpul j

$Error_k$ = *error* simpul k

wj_k = Bobot relasi dari simpul j ke simpul k pada *layer* berikutnya

6. Nilai *error* yang dihasilkan dari langkah sebelumnya digunakan untuk memperbarui bobot relasi, dengan menggunakan rumus:

$$W_{ij} = W_{ij} + 1\,.\, Error_j.Output_i$$

Keterangan:

wi_j = bobot relasi dari unit i pada *layer* sebelumnya, ke unit j

I = *learning rate* (konstanta, nilainya antara 0 sampai dengan 1)

$Error_j$ = *Error* pada *output layer* simpul j

$Output_i$ = *Output* dari simpul i

7. Algoritma aplikasi/pengujian

b. Akgoritma pengujian/aplikasi

Pada algoritma ini, tahap pengujian dilakukan melalui *feedforward* dengan langkah-langkah sebagai berikut:

1. Melakukan inisialisasi bobot (hasil pelatihan)
2. Pada setiap vektor *input*, mengerjakan langkah 2 – 4
3. Untuik *I = I,…,n : set aktivasi unit input* X_I
4. Untuik *j = I,…,p:*

$$z\_in_j = v_{oj} + \sum_{i=1}^{n} . x_i . v_{ij}$$

$$z_j = f(z_{inj})$$

5. Untuk *k - I,…,p:*

$$y\_in_k = w_{ok} + \sum_{i=1}^{n} . z_i . w_{jk}$$

$$y_k = f(y_{in\,k})$$

Selanjutnya melakukan denomarlisasi *testing*. Setelah melakukan proses training dan testing pola yang dilatih, maka akan diperoleh hasil bahwa nilai pengujian terhadap pola tersebut telah benar/akurat atau sebaliknya. Rata-rata *Error* (RMSE) jaringan dapat dihitung menggunakan rumus (Andrijasa, dkk., 2010):

$$RMSE = \sum_{i=1}^{n} . \frac{(yi - yn)^2}{N}$$

Keterangan:

y_i = Nilai aktual data (target)

y_n = Nilai hasil prediksi (*actual output*)

N = Jumlah data yang diujikan

Sedangkah untuk proses denormalisasi atau pengembalian nilai hasil prediksi jaringan ke bentuk data semula (sebelum dilakukan normalisasi), dapat dihitung menggunakan rumus sebagai berikut:

$$x_i = y_n(x_{max} - x_{min}) + x_{min}$$

Keterangan:

x_i = Nilai X yang akan dilakukan denormalisasi

y_n = Nilai hasil prediksi (actual *output*) yang sesuai dengan xi

x_{max} = Nilai maksimum pada barisan X

x_{min} = Nilai minimum pada barisan X

Cabang ilmu kecerdasan buatan yang cukup luas memberikan tuntutan untuk mampu memproses data secara maksimal. Penggunaan *multilayer perceptron* memiliki beberapa kelemahan dalam mendeteksi suatu *objek,* sehingga dibuatlah versi regularisasi *multilayer perceptron* yang tergolong dalam *deep feedforward artificial neural network* yang disebut dengan *Convolutional Neural Network* (CNN). Pada dasarnya CNN terinspirasi dari pola konektivitas antar *neuron* yang menyerupai *visual cortex* pada binatang, maka dari itu CNN banyak digunakan pada data *image*/analisis citra.

3. Konsep CNN

Convolutional Neural Network (CNN) merupakan versi regulasi *Multilayer Perceptron* dan tergolong dalam *deep feed-forward artificial neural network*. *Convolutional layer* juga terdiri dari *neuron* yang tersusun sedemikian rupa sehingga membentuk sebuah filter dengan panjang dan tinggi (*pixels*). Secara garis besar, CNN memanfaatkan proses konvolusi dengan menggerakan sebuah kernel konvolusi (filter) berukuran tertentu

ke sebuah gambar, komputer mendapatkan informasi representatif baru dari hasil perkalian bagian gambar tersebut dengan filter yang digunakan. CNN memiliki banyak istilah dalam bidang pemrosesan gambar. Implementasi CNN dari segi pemrosesan gambar dapat diilustrasikan pada Gambar 8. Gambar tersebut memiliki berbagai macam posisi. Selain tantangan variasi posisi objek, masih ada tantangan lain seperti rotasi objek dan perbedaan ukuran objek (*scaling*). Pada dasarnya *neural network* mampu mengenali objek pada gambar dari berbagai posisi (*translation invariance*).

Gambar 15 Ilustrasi CNN dari Segi Pemrosesan

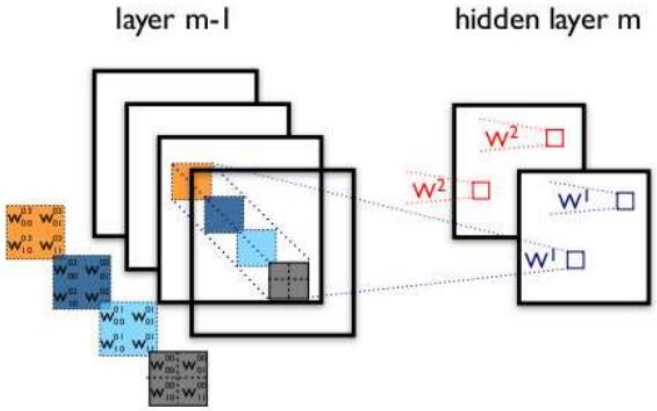
Sumber: https://wiragotama.github.io

Cara kerja CNN memiliki kesanmaan dengan Mult*layer Perceptron*, namun dalam CNN setiap *neuron* dipresentasikan dalam bentuk dua dimensi. Berbeda dengan multi*layer perceptron* yang setiap *neuron* hanya berukuran satu dimensi. Arsitektur pada multi*layer perceptron* telah diilustrasikan pada Gambar 14. Berbeda dengan CNN, data yang dipropagasikan pada jaringan adalah data dua dimensi. Hal tersebut berpengaruh pada operasi linear dan parameter bobot pada CNN berbeda. Operasi linear pada CNN menggunakan operasi

konvolusi, sedangkan bobot tidak lagi satu dimensi saja, namum berbentuk empat dimensi yang merupakan kumpulan kernel konvolusi seperti pada Gambar 15. Dimensi bobot pada CNN adalah:

neuron input x neuron output x tinggi x lebar

CNN hanya dapat digunakan pada data yang memiliki struktur dua dimensi, seperti citra dan suara. Hal tersebut dikarenakan CNN menggunakan proses konvolusi (Gambar 16).



Gambar 16 Proses Konvolusi pada CNN

Sumber: https://wiragotama.github.io

- **Arsitektur Jaringan CNN**

Pada *multilayer perceptron*, sebuah jaringan tanpa *hidden layer* dapat memetakan persamaan linear apapun, sedangkan jaringan dengan satu atau dua hidden *layer* dapat memetakan sebagian besar persamaan pada data. Namun pada data yang lebih kompleks, *multilayer perceptron* memiliki keterbatasan. Pada suatu permasalahan dengan menggunakan jumlah *hidden layer* dibawah tiga *layer*, terdapat pendekatan untuk menentukan jumlah *neuron* pada maisng-masing *layer* untuk mendekati hasil optimal. Namun, penggunaan *layer* diatas dua pada umumnya tidak direkomendasikan, karena dapat menyebabkan *overfitting* serta kekuatan *backpropagation* berkurang secara signifikan.

Kekurangan *multilayer perceptron* dalam menangani data kompleks, dapat diatasi menggunakan suatu fungsi dalam mentransformasikan data *input* menjadi bentuk yang lebih mudah dimengerti. Hal tersebut memicu berkembangnya model neural network dengan jumlah diatas tiga *layer*. Secara umum jenis *layer* pada CNN dibedakan menjasi dua yaitu:

a. *Layer* ekstraksi fitur gambar

Layer ekstraksi fitur gambar terletak pada awal arsitektur tersusun atas beberapa *layer* dan setiap *layer* tersusun dari *neuron* yang terkoneksi pada daerah lokal (*local religion*) *layer* sebelumnya. *Layer* jenis pertama yaitu *convolutional layer* dan *layer* kedua adalah *pooling layer*. Setiap *layer* tersebut, diberlakukan fungsi aktivasi, posisinya berselang-seling antara jenis pertama dengan jenis kedua. *Layer* ini menerima *input* gambar secara langsung dan memprosesnya hingga menghasilkan keluaran berupa vektor untuk diolah pada *layer* berikutnya.

b. *Layer* klasifikasi

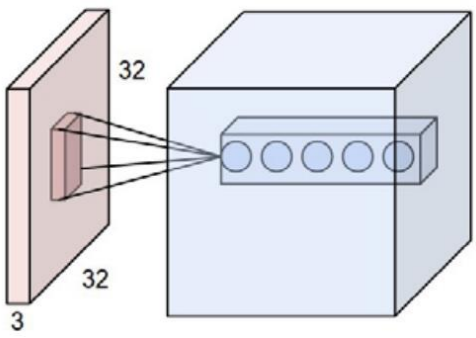
Layer klasifikasi tersusun dari beberapa *layer* dan setiap *layer* tersusun atas *neuron* yang terkoneksi secara penuh (*fully connected*) dengan *layer* yang lain. *Layer* ini menerima *input* dari hasil keluaran *layer* ekstrasi fitur gambar berupa vektor kemudian ditransformasikan seperti *multilayer perceptron* dengan tambahan beberapa *hidden layer*. Hasil keluaran berupa skoring kelas untuk klasifikasi. Dengan demikian CNN merupakan suatu metode yang digunakan untuk mentransformasikan gambar original pada *layer* per *layer* dari nilai piksel gambar kedalam nilai skoring kelas untuk klasifikasi. Pada setiap *layer* ada yang mempunyai *hyperparameter* dan ada yang tidak memiliki parameter (bobot dan bias pada *neuron*).

1. Convolution Layer

Convolution Layer merupakan *layer* pertama yang menerima *input* gambar secara langsung pada arsitektur. Operasi pada *layer* ini sama dengan operasi konvolusi dengan melakukan operasi kombinasi linier filter terhadap daerah lokal. Filter yaitu representasi bidang reseptif dari *neuron* yang terhubung ke dalam daerah lokal (*local connectivity*) pada *input* gambar. Bentuk *layer* pada *convolution layer* direpresentasikan sebagai volume B x K x L atau *layer* ukuran B x K dengan jumlah sebanyak L (Gambar 18). *Convolution layer* mamiliki *hyperparameter* dan parameter. *Hyperparameter* pada *layer* ini menjadi acuan untuk menentukan jumlah dan ukuran hasil ekstraksi *layer* (Tabel 1).

Tabel 1 *Hyperparameter* pada *Convolution Layer*

No	*Hyperparameter*	Keterangan
1	*Depth*	Jumlah *layer* pada konvolusi atau kedalaman *layer*
2	*Stride*	Jumlah pergeseran filter pada proses konvolusi
3	*Zero-padding*	Jumlah penambahan nilai intensitasi nol di daerah sekitar *input* gambar



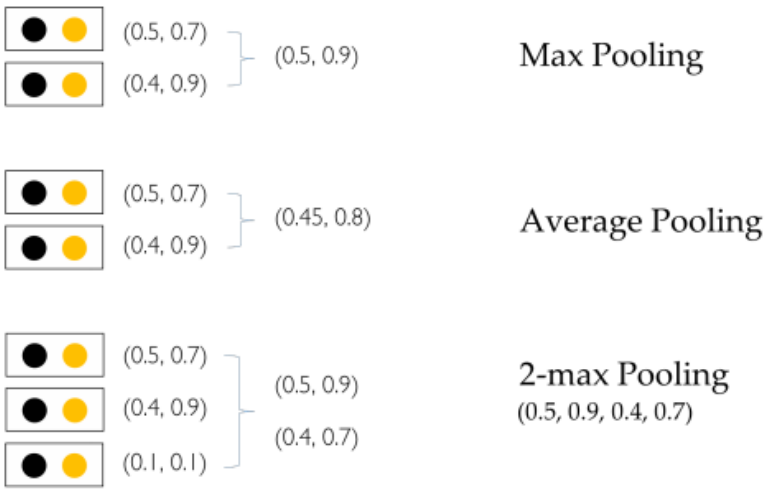
Gambar 17 *Input* Gambar pada *Convolutional Layer*

Sumber: Jaya (2018)

Convolutional layer pada Gambar 17 merupakan contoh *input* gambar berukuran 32 x 32 x 3, dengan volume *convolutiona*l *layer* pada *layer* pertama dan setiap *neuron* terhubung pada daerah *layer input*. Salah satu parameter pada *convolutional layer* yaitu *parameter sharing* yang digunakan untuk mengontrol jumlah parameter. Parameter ini merupakan himpunan *learnable filter* (suatu kernel yang nilai berupa bobot). Jika dibandingkan dengan arsitektur pada ANN, ukuran gambar adalah M x N x C x N *neuron* dan filter berukuran B x K sehingga total keseluruhan parameter yaitu M x N x C x N x B x K x L. Hal tersebut berlaku hanya jika kondisi parameter dibuat berbeda pada setiap *input* piksel gambar. Namun jumlah parameter tersebut sangat tidak efisien dan memboroskan ruang memori, sehingga model ANN tidak digunakan. Sedangkan *convolutional layer* setiap parameter disamakan pada setiap *input* gambar sehingga jumlah parameter berubah menjadi sebanyak M x N x C x B x K x L parameter.

2. *Pooling Layer*

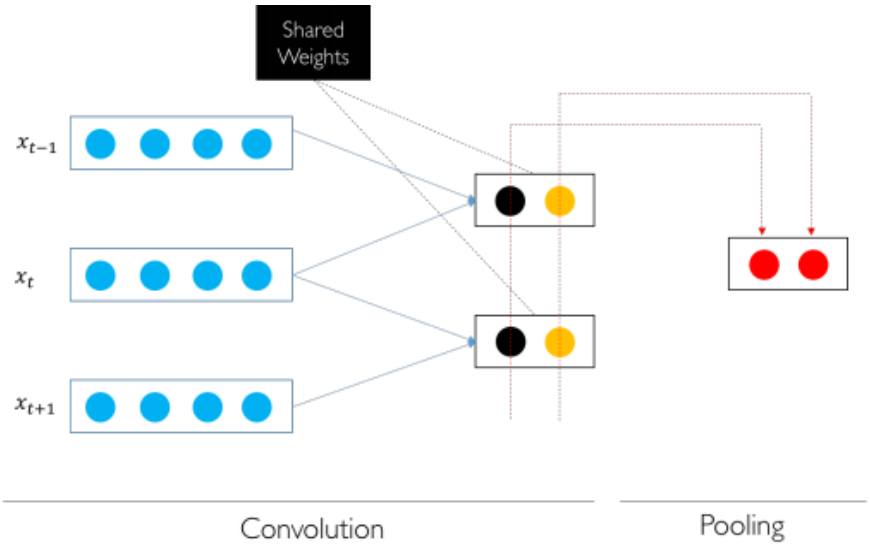
Pooling layer akan mereduksi spasial serta jumlah parameter dalam jaringan dan mempercepat komputasi dan mengontrol terjadinya *overfitting*. *Pooling layer* tidak memiliki parameter, karena parameter sudah ditentukan sebelumnya (*fixed)*. Terdapat beberapa teknik *pooling*, diantaranya yaitu: (1)*max pooling*, (2)*average pooling,* dan (3) *K-max pooling* yang diilustrasikan pada Gambar 18.



Gambar 18 Teknik *Pooling*

Sumber: https://wiragotama.github.io

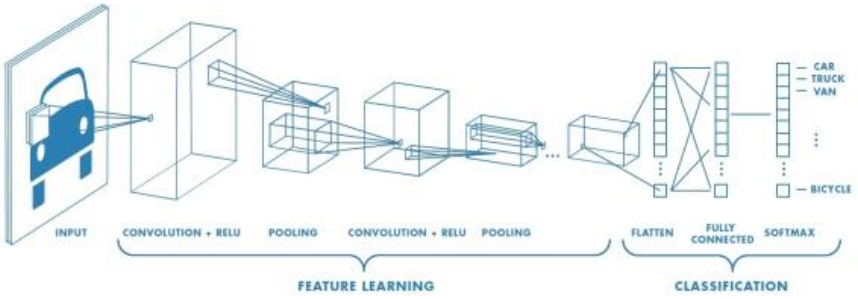
Pada *K-max pooling* dilakukan pencarian K nilai terbesar untuk setiap dimensinya (kemudian hasilnya digabungkan). Gabungan operasi *convolution* dan *pooling* secara konseptual diilustrasikan pada Gambar 19.



Gambar 19 *Convolution* dan *Pooling*

Sumber: https://wiragotama.github.io

Setelah melakukan komputasi pada *convolution* dan *pooling*, setiap vektor akan dilewatkan pada *multilayer perceptron* (*fully connected*) untuk melakukan sesuatu (tergantung permasalahan), misal klasifikasi gambar, klasifikasi sentimen, dsb.

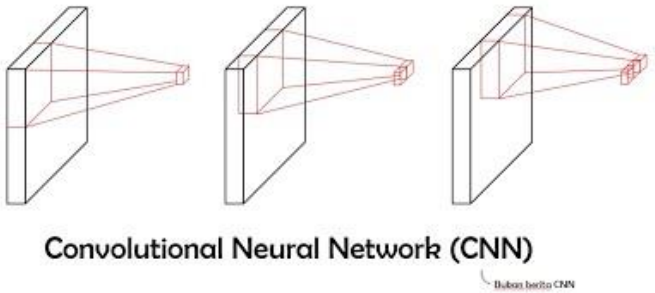


Gambar 20 *Convolution Neurak Network*

Sumber: https://wiragotama.github.io

Berdasarkan Gambar 20 tahap pertama pada arsitektur CNN yaitu konvolusi. Tahap tersebut dilakukan dengan menggunakan sebuah kernel dengan ukuran tertentu. Perhitungan jumlah kernel yang dipakai tergantung jumlah fitur yang dihasilkan. Kemudian dilanjutkan menuju fungsi aktivasi, menggunakan aktivasi ReLU (*Rectifer Linear Unit*). Setelah keluar dari proses fungsi aktivasi kemudian melalui proses *pooling*. Proses ini diulang beberapa kali sampai diperoleh peta fitur yang cukup untuk dilanjutkan pada *fully connected neural network*, dan dari *fully connected neural network* diperoleh *output class.* Untuk lebih jelasnya mengenai konsep CNN dapat dilihat penjelasan pada Video 2.

[redacted]



[redacted]

Video 2 Konsep CNN

4. Pemrograman ANN dan CNN pada Google Colaboratory

Pada modul ini pemrograman menggunakan bahasa pemrograman python. *Library* yang digunakan pada pemrograman ANN dan CNN yaitu *library keras* dan *sklearn*. Fasilitas GPU pada *google colaboratory* digunakan untuk mempercepat komputasi. Pada umumnya *library matplotlib* dan *opencv* digunakan sebagai visualisasi data. Pemrograman ANN dan CNN memiliki tahap sebagai berikut:

1. Pengumpulan *dataset* (*data acquired*)
2. Mentransformasikan data menjadi matriks
3. Pembuatan model arsitektur jaringan
4. Pelatihan dan evaluasi
5. Prediksi (*model testing*)

Penjabaran mengenai pemrograman ANN dan CNN lebih lanjut dapat diakses melalui *jobsheet* yang terdapat pada *e-collab classroom*. Pada *jobsheet* tersebut terdapat pemrograman yang terstruktur dimulai dari mount *dataset*, pembuatan model jaringan, *training data, testing data,* evaluasi, dan pemakaian model sehingga tahapan pada setiap ANN dan CNN dapat diimplementasikan.

Rangkuman Materi

Berdasarkan dari pemaparan materi tersebut dapat disimpulkan beberapa poin sebagai berikut:

1. ANN merupakan sistem pengolahan informasi yang terdiri dari sejumlah besar elemen pemrosesan informasi (*neuron*) yang saling terhubung dan bekerja sama untuk menyelesaikan masalah tertentu.
2. Metode pada ANN berdasarkan bentuk *layer*nya dapat dibagi menjadi dua macam yaitu single *layer perceptron* dan *multilayer perceptron*.
3. Perbedaan CNN dengan ANN terletak pada proses konvolusi yang terdapat pada setiap *layer* konvolusi. Pengolahan informasi didapat dari pergerakan kernel konvolusi (filter) berukuran tertentu pada sebuah gambar.
4. Terdapat 5 tahap dasar implementasi ANN dan CNN yang meliputi: (1) pengumpulan *dataset* (*data acquired*), (2) mentransformasikan data menjadi matriks, (3) pembuatan model arsitektur jaringan, (4) pelatihan dan evaluasi, dan (5) prediksi (model *testing*).

Materi Pengayaan

Algoritma pada teknik penyelesaian permasalahan *learning* mempunyai data didalamnya, data tersebut dinamakan sebagai "*Dataset*". Pada setiap algoritma memiliki kebutuhan berbeda satu dengan yang lainnya berkaitan dengan banyaknya data. Terdapat algoritma yang membutuhkan data yang sangat banyak dan ada juga algoritma yang cukup dengan *dataset* berukuran kecil. ANN merupakan sistem adaptif yang dapat mengubah strukturnya untuk memecahkan masalah berdasarkan informasi eksternal maupun internal yang mengalir melalui jaringan tersebut. Oleh karena sifatnya yang adaptif, ANN juga sering disebut dengan jaringan adaptif. Terdapat beberapa jenis *dataset* yang sering digunakan antara lain:

1. *Mall Customer Dataset*

   Mall Customer Dataset merupakan *dataset* yang berisi informasi tentang orang yang mengunjungi Mall di kota tertentu. *Dataset* tersebut biasanya terdiri dari berbagai kolom seperti jenis kelamin, id pelanggan, usia, pendapatan tahunan, dan skor. *Dataset* ini biasanya digunakan untuk mengklasifikasikan pelanggan berdasarkan usia, pendapatan, dan minat mereka.
2. *Iris Dataset*

   Iris merupakan *dataset* sederhana yang berisi informasi tentang kelopak bunga dan lebar sepal. *Dataset* iris dibagi menjadi tiga kelas, dengan 50 baris di setiap kelas.

Dataset ini umumnya digunakan untuk mengklasifikasi dan melakukan pemodelan regresi.

2. MNIST *Dataset*

   MNIST *dataset* merupakan database dari digit tulisan tangan. *Dataset* ini berisi 60.000 gambar pelatihan dan 10.000 gambar pengujian. *Dataset* ini adalah kumpulan data yang sempurna untuk memulai melakukan klasifikasi gambar dengan mengklasifikasikan angka dari 0 hingga 9.

3. SOCR *Dataset*

   SOCR *dataset* merupakan *dataset* yang digunakan untuk mengolah data tinggi dan bobot. *Dataset* ini hanya berisi tinggi dan berat dari 25000 manusia yang berbeda selama 18 tahun. Selain itu, *dataset* ini dapat digunakan untuk membangun model yang dapat memprediksi tinggi atau berat manusia.

5. *Titanic Dataset*

   Titanic dataset merupakan *dataset* yang berisi informasi tentang nama, usia, jenis kelamin, jumlah saudara kandung, dan informasi lain tentang 891 penumpang di set pelatihan dan 418 di set pengujian.

6. *Credit Card Fraud Detection Dataset*

   Credit Card Fraud Detection Dataset merupakan *dataset* yang berisi transaksi yang dilakukan pada kartu kredit. Pada kasus ini pengguna kartu kredit diberi label penipuan atau asli. Hal tersebut penting bagi perusahaan yang memiliki sistem transaksi untuk membangun model dalam mendeteksi aktivitas penipuan.

7. *Pima Indians Dataset*

Pima Indians Dataset yaitu *dataset* ini berasal dari National Institute of Diabetes and Digestive and Kidney Diseases, dan berisi informasi tentang 768 wanita dari suatu populasi di dekat Phoenix, Arizona, AS. Hasil yang diuji adalah Diabetes, 258 dinyatakan positif dan 500 dinyatakan negatif.

Selain *dataset* pada ANN, penerapan jaringan saraf menggunakan CNN dalam pembuatan *pipeline* dan membangun model prediktif, terkadang hasilnya tidak sempurna atau tidak sesuai yang diharapkan. Hal tersebut dapat diatasi dengan menerapkan augmentasi data. Augmentasi data merupakan strategi yang berfokus pada regenerasi lebih banyak gambar dari yang sudah tersedia. Selain itu, Teknik tersebut juga dapat memproduksi gambar dalam bentuk atau dimensi lain.

Gambar 21 Penerapan Augmentasi Data pada Gambar Anjing

Gambar seekor anjing (Gambar 16) tersebut terlihat seperti gambar yang diambil beberapa kali dari arah yang berbeda. Tapi sebenarnya gambar tersebut ditambah untuk menghasilkan bentuk atau dimensi lain menggunakan augmentasu data. Adapun langkah kerja augmentasi data yaitu:

- Mengurangi bias model terhadap kelas data tertentu ke kelas lain, hal ini dapat membantu algoritma dalam menggeneralisasi dengan baik.

- Meningkatkan jumlah sampel di kelas yang kurang terwakili dalam data (dengan menambah sampel asli menjadi lebih banyak sampel)

3 EVALUASI

Tes Individu

Referensi

Tes Individu

Untuk mengerjakan Studi Kasus 1 dan Studi Kasus 2 pada modul *Learning* pelajari jobsheet *learning* yang tertera pada fitur *google colaboratory*.

STUDI KASUS 1

Salah satu penerapan ANN yang sering digunakan pada Analisa *pima Indians diabetes dataset*. *Dataset* ini berasal dari *National Institute of Diabetes* and *Digestive and Kidney Diseases*, yang berisi informasi tentang 768 wanita dari suatu populasi di dekat Phoenix, Arizona, AS. Hasil yang diuji adalah Diabetes, 258 dinyatakan positif dan 500 dinyatakan negatif. Oleh karena itu, ada satu variabel target (dependen) dan 8 atribut (TYNECKI, 2018): *pregnancies* (kehamilan), OGTT (Tes Toleransi Glukosa Oral), *blood pressure* (tekanan darah), *skin thicknes* (ketebalan kulit), *insulin,* BMI (Indeks Massa Tubuh), usia, dan diabetes silsilah. Populasi Pima telah diteliti oleh Institut Nasional Diabetes dan Penyakit Pencernaan dan Ginjal dengan interval 2 tahun sejak 1965. Sebagai bukti *epidemiologis* menunjukkan bahwa DMT2 hasil dari interaksi faktor genetik dan lingkungan, *pima indians diabetes dataset* mencakup informasi tentang atribut yang dapat dan harus dikaitkan dengan timbulnya diabetes dan komplikasinya di masa depan.

1. Berdasarkan *dataset* pada *jobsheet* yang tertera di *google colaboratory* buatlah rancangan jaringan ANN dengan klasifikasi berikut:
   a. Deklarasi *input* variabel dan *output* variabel
   b. Arsitektur jaringan

c. Metode pembobotan

2. Bagaimana hasil pelatihan dari jaringan tersebut berdasarkan kriteria berikut:
   a. Loss dan akurasi grafik
   b. Gunakan *optimizer* lain yang terdapat pada *library* keras dan bandingkan hasilnya,

STUDI KASUS 2

Penerapan CNN dalam klasifikasi citra dan pendeteksian citra sudah banyak dilakukan, seperti mengklasifikasi citra anjing atau kucing, mendeteksi objek yang berbeda pada sebuah citra atau melakukan pengenalan wajah. Contoh implementasi CNN pada *google colaboratory* merupakan klasifikasi citra pada anjing dan kucing.

1. Berdasarkan *data augmentasi* yang ada pada *jobsheet* yang tertera di *google colaboratory* buatlah rancangan jaringan CNN dengan klasifikasi berikut:
   a. Deklarasi *input* variabel dan *output* variabel
   b. Arsitektur jaringan
   c. Metode pembobotan
2. Bagaimana hasil pelatihan dari jaringan tersebut berdasarkan kriteria berikut:
   a. Loss dan akurasi grafik
   b. Gunakan *optimizer* (Nadam, RMSprop, dan SGD) yang terdapat pada *library* keras dan bandingkan hasilnya,

Referensi

Kusrini, & Luthfi, E. T. (2009). Algoritma Data Mining. Yogyakarta: Andi Publishing, (Online), (https://books.google.co.id/books?id=-Ojclag73O8C&printsec=frontcover)

Myatt, Glenn J. (2007). Making Sense of Data: A Practical Guide to Exploratory Data Analysis and Data Mining. New Jersey: John Wiley & Sons, Inc, (Online), (https://onlinelibrary.wiley.com/)

Vercellis, C. (2009). Business Intelligent: Data Mining and Optimizzation for Decision Making. Southern Gate, Chichester, West Sussex, United Kingdom: John Wiley & Sons Ltd, (Online), (https://onlinelibrary.wiley.com/)