BAB 2

LANDASAN TEORI

2.1 Klasifikasi Wajah

Sistem Klasifikasi Wajah adalah sistem yang membedah karakteristik atau fitur-fitur pada wajah seseorang dan memprosesnya pada suatu mesin database. Sistem ini digunakan untuk mendeteksi wajah dan klasifikasi wajah. Secara umum, ada 4 tahapan dalam klasifikasi wajah yaitu.

1. Perolehan citra wajah.
2. Pra-proses *data.*
3. Ekstraksi fitur wajah.
4. Klasifikasi citra dari wajah.

Tahapan yang paling penting disini adalah tahap ketiga dan tahap keempat, yaitu ekstraksi fitur wajah dan klasifikasi citra (Fandiansyah et al., 2017). Ekstraksi fitur wajah dinilai sangat penting karena untuk dapat mengklasifikasikan data tersebut, mesin harus bisa mengidentifikasikan fitur-fitur penting pada wajah. Klasifikasi citra dinilai penting karena itu adalah output dari sistem klasifikasi, artinya klasifikasi citra wajah ini nantinya akan dapat mendeteksi keberadaan wajah pada suatu citra.

2.2 Ekspresi Wajah

Ekspresi wajah adalah refleksi dari pengalaman dan pesan dari suatu *endogroup* terhadap *exogroup*. Istilah yang digunakan untuk menyebut emosi sangatlah unik, dimana pesan terhadap suatu ekspresi dapat tertangkap secara spontan. Ada beberapa pergerakan otot muka yang dapat menghasilkan beberapa

kelompok ekspresi. Contohnya, ekspresi positif seperti senyuman dapat di observasi dengan mudah di dalam suatu waktu, intensitas, dan juga suatu konteks. Ekspresi negatif juga dapat terdeteksi dengan mudah dan mempunyai morfologi yang khas, contohnya di ujung mulut atau di alis mata (Ramachandran, Ramachandran and Ramachandran, 2012). Ekspresi wajah mempunyai banyak kegunaan sebagai metode untuk pengukuran keadaan mental. Dengan akses terhadap perangkat lunak pendeteksi wajah secara *real time*, hal ini bisa menjadi wadah untuk memberikan informasi tentang ekspresi wajah pelajar yang ada, mengukur level emosi pelajar tersebut, lalu membuat suatu *prompt* untuk mendukung keadaan emosinya sekarang (Tettegah and Martin, 2016).

2.3 Pembelajaran Mesin

Pembelajaran Mesin adalah suatu bidang studi yang mempelajari algoritma komputasi untuk mengubah data empirikal menjadi suatu model yang bisa digunakan (Edgar and Manz, 2017). Pembelajaran mesin sendiri berkembang dari bidang statistika tradisional dan kecerdasan buatan.

Proses pembelajaran mesin biasanya melalui empat tahap berikut.

1. Mempelajari masalah digital dengan mengumpulkan data dari masalah tersebut yang nantinya akan diteliti
2. Merangkum pemahaman masalah tersebut secara mendasar dalam bentuk model
3. Memprediksi nilai dari masalah yang ada melalui model yang dihasilkan dari proses yang ada
4. Mendeteksi anomali yang ditunjukan oleh masalah yang sedang diamati

Bidang pembelajaran mesin telah menjadi salah satu topik yang terpopuler dalam topik ilmu komputer berkat usaha perusahaan besar seperti *Google*, *Miccrosoft*, *Facebook*, *Amazon*, dan lainnya yang mempopulerkan bidang ini. Bisnis proses yang dimiliki perusahaan tersebut memungkinkan adanya pengumpulan data secara masif yang memungkinkan adanya pendekatan statistik dan komputasi untuk merancang model yang berguna dari data tersebut.

Umumnya, ada dua jenis skenario pembelajaran yaitu *supervised learning,* dan *unsupervised learning*.

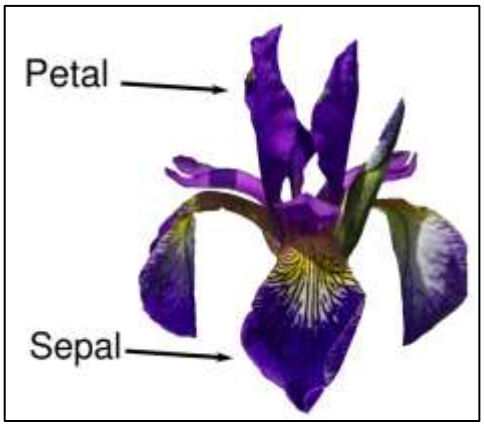
2.3.1 Supervised Learning

Supervised learning adalah pembelajaran yang terarah atau terawasi (Putra, 2020). Ibaratnya, ada guru yang mengarahkan dan siswa yang diajarkan oleh guru tersebut. Manusia yang membuat algoritma pembelajaran mesin berperan sebagai guru sedangkan mesin berperan sebagai murid. Proses belajar ini mengoptimalkan nilai *w* sebagai proses latihan atau *training*. Semakin lama proses *training*, konsep ini akan makin jelas juga.

Ketika melalui proses *supervised learning*, data latihan akan terdiri dari pasangan *input* dengan *output* yang benar. Disaat proses latihan, algoritma akan mencari pola-pola di dalam data yang berhubungan dengan keluaran yang diinginkan. Setelah proses latihan terjadi, algortima akan mengambil masukan atau *input* yang belum terdeteksi dan menentukan klasifikasi data tersebut melalui model yang sudah dilatih sebelumnya. (Wilson, 2019)

Terdapat dua tipe permasalahan dalam *supervissed learning* yaitu klasifikasi dan regresi. Pada masalah klasifikasi, tujuan dari model adalah untuk memprediksi suatu *class label* yang merupakan pilihan dari kemungkinan yang sebelumnya telah

didefinisikan. Contohnya adalah klasifikasi bunga Iris seperti yang terlihat pada di Gambar 2.1, di mana bunga Iris diklasifikasikan menjadi tiga spesies berdasarkan *petal* dan *sepal*. Terdapat dua jenis klasifikasi yaitu *binary classification* yang mengklasifikasikan tepat dua kelas, dan *multiclass clasification* yang mengklasifikasikan lebih dari dua kelas. Contoh permasalahan bunga Iris yang dijelaskan tadi adalah contoh dari *multiclass classification* (Müller and Guido, 2016).



Gambar 2.1 Bagian Bunga Iris sebagai *identifier* dari *multiclass classification* (Müller and Guido, 2016)

Regression atau regresi bertujuan untuk memprediksi angka yang bersifat polinom atau *floating-point number*. Salah satu contoh dari regresi adalah prediksi gaji tahunan dari karyawan berdasarkan tingkat pendidikan yang ditempuh, umur, dan tempat tinggal. Disini jelas jika meallilam prediksi gaji, nilai prediksi ini berbentuk jumlah yang bisa terdiri dari angka berapapun (Müller and Guido, 2016).

2.3.2 Unsupervised Learning

Jika *supervised learning* diawasi oleh guru yang bertugas untuk mengajar, maka pada *unsupervised learning* tidak ada guru yang mengajar (Putra, 2020). *Unsupervised learning* sejatinya adalah algortima yang diciptakan untuk

mempelajari pola-pola pada data yang tidak memiliki label. Hal ini sering ditemukan pada permasalahan dunia nyata di mana kebanyakan data didapatkan tidak dengan label yang sudah terdefinisikan (Roman, 2019).

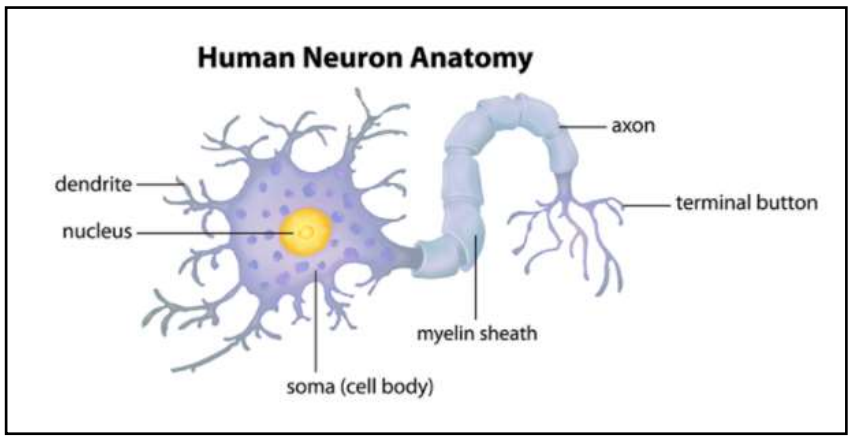
Ada dua macam *unsupervised learning* yang umum yaitu *dataset transformation* dan *clustering*. *Dataset transformation* adalah algoritma yang membuat representasi data baru yang lebih mudah dimengerti oleh manusia dan algoritma pembelajaran mesin lainnya dibandingkan dengan representasi data orisinil. Konsep ini secara umum diaplikasikan dalam *dimensionality reduction* yang mengambil representasi data dengan dimensi yang tinggi yang terdiri dari banyak fitur lalu mencari metode baru untuk representasi data yang lebih mudah dipahami dengan fitur dan karakteristik yang lebih sederhana.

Clustering disini bertujuan untuk mempartisi data menjadi kelompok-kelompok berbeda dari *item* yang serupa. Salah satu contoh penggunaan *clustering* dapat dilihat pada galeri foto. Agar bisa mengatur foto-foto yang ada, tentu galeri foto mau mengelompokan foto dengan orang-orang yang sama. Namun, galeri tidak dapat mengetahui foto yang mana yang menunjukan suatu orang secara spesifik dan galeri juga tidak bisa mengidentifikasi berapa banyak orang yang ada pada koleksi galeri. Salah satu penyelesaian masalah tersebut adalah dengan melakukan ekstraksi semua wajah yang ada dan mengelompokannya dalam kelompok dengan wajah yang mirip (Müller and Guido, 2016).

2.4 Artificial Neural Network (ANN)

Sebelumnya telah dijelaskan bahwa *machine learning* pada dasarnya meniru proses bagaimana manusia belajar. *Artificial Neural Network (*ANN) terlahir dari penelitian untuk membuat mesin dapat belajar selayaknya manusia dengan

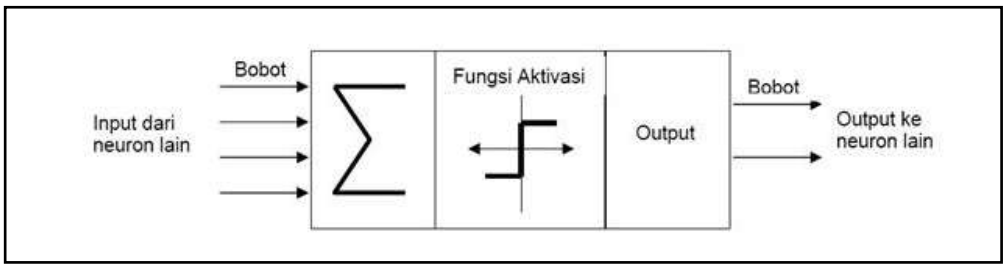
membuat simulasi jaringan saraf biologis manusia yang disebut *neural network* seperti yang terlihat pada Gambar 2.2 (Putra, 2020). *Artificial Neural network* pada dasarnya adalah sistem komputasi yang memungkinkan komputer untuk meniru koneksi saraf di dalam sistem saraf tubuh manusia (Rosebrock, 2021).



Gambar 2.2 Anatomi dari Sistem Jaringan Saraf pada Manusia
(Rosebrock, 2021)

2.4.1 Struktur Artificial Neural Network (ANN)

Dasar dari struktur suatu *neural network* adalah dengan mengambil cara berpikir suatu sistem atau aplikasi yang mirip dengan otak manusia, baik untuk pemrosesan berbagai sinyal elemen yang diterima, *parallel processing*, dan juga batas toleransi kesalahan atau *error* pada sistem.



Gambar 2.3 Struktur dari *Artificial Neural Network*
(Suhartono, 2012)

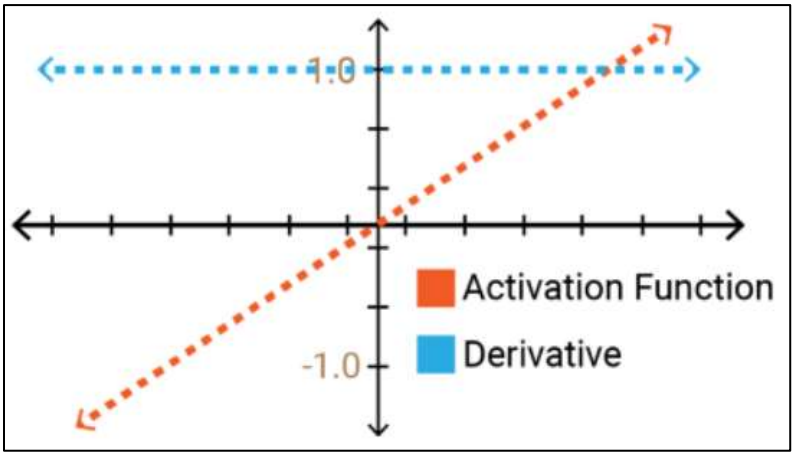
Pada Gambar 2.3, karakteristik atau struktur dari ANN dapat dilihat dari pola-pola koneksi antara satu neuron dengan neuron lainnya, metode penentuan

bobot dari tiap hubungan yang ada, dan fungsi aktivasinya. *Input* disini berperan sebagai dendrit untuk mengirimkan sinyal. *Output* berperan sebagai akson yang mengirimkan sinyal ke neuron lain. Peran fungsi aktivasi adalah menjadi sinapsis yang menghubungkan proses *input* menuju *output* (Suhartono, 2012).

Faktor bobot adalah sebuah nilai yang mendefinisikan suatu tingkatan koneksi antar *node*. Jika nilai bobot suatu koneksi membesar, maka artinya koneksi *node* tersebut makin penting. Nilai dari suatu bobot dapat berupa suatu bilangan *real* atau bilangan bulat. Bentuk nilai bobot tergantung dari jenis masalah dan penggunaan model. Setiap kali ada masalah atau *error*, sistem jaringan dapat belajar dari masalah baru tersebut dengan mengatur ulang nilai bobot yang ada untuk menyesuaikan karakter nilai. Hal ini dapat dilakukan karena nilai bobot bisa ditentukan untuk berada didalam suatu interval tertentu. Artinya, bobot dapat beradaptasi dengan pola *input* selama proses pelatihan (Puspitaningrum, 2006).

Fungsi aktivasi yang terlihat pada Gambar 2.4 berperan sebagai fungsi yang dipakai untuk mendapatkan *output* dari suatu *node*. Fungsi aktivasi juga dikenal sebagai *transfer function* (Sharma, 2017). Fungsi aktivasi ini berperan sebagai gerbang diantara *neuron* di mana fungsi aktivasi yang akan menentukan apakah suatu *neuron* harus diaktifkan atau tidak. Fungsi aktivasi ini berperan sangat penting di dalam *neural network* karena tanpa fungsi ini, *weight* dan *bias* pada *neural network* akan melakukan proses *linear transformation*. Walaupun perhitungan linear lebih mudah diselesaikan, namun perhitungan linear ini sangatlah terbatas kapasitasnya untuk menyelesaikan masalah-masalah kompleks yang dihadapi oleh ANN. Selain itu, perhitungan linear juga tidak memiliki kekuatan yang cukup untuk mempelajari *functional mapping* yang kompleks dari

suatu data. Tanpa fungsi aktivasi, *neural network* hanya akan menjadi *linear regression model* (Gharat, 2019).



Gambar 2.4 Fungsi Aktivasi Linear beserta Derivasinya
(Gharat, 2019)

Dalam penggunaannya, fungsi aktivasi dapat dibagi menjadi dua jenis.

1. *Linear* atau *identity activation function*

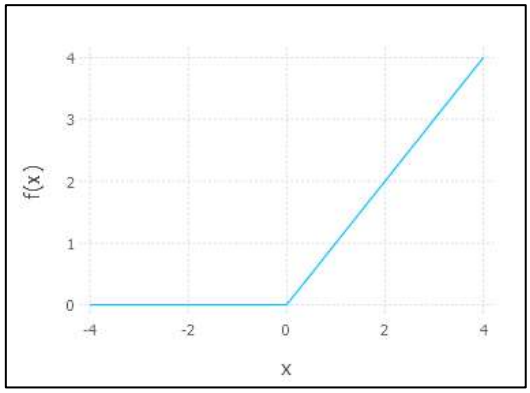
Fungsi aktivasi linear mengambil suatu *input*, mengalikannya dengan *weight* dari tiap *neuron* yang ada, lalu akhirnya membuat sinyal *output* yang sebanding dengan *input*. Fungsi aktivasi linear ini memiliki dua masalah utama. *Back-propagation* menjadi hal yang tidak mungkin dilakukan. Hal ini terjadi karena derivasi dari suatu fungsi adalah konstan dan tidak mempunyai hubungan dengan *input* X. Oleh karena itu, mustahil untuk kembali dan mencari *weight* mana yang bisa memberikan prediksi yang lebih bagus. Masalah kedua dari fungsi aktivasi linear ini adalah semua *layer* dari *neural network* akan menyatu. Tidak peduli ada seberapa banyak *layer* yang ada pada *neural network*, *layer* terakhir akan menjadi fungsi linear untuk *layer* pertama (Gharat, 2019).

2. *Non-linear activation function*

Fungsi aktivasi yang bersifat non-linear sangat sering digunakan dalam didalam *neural network*. Hal ini terjadi karena fungsi aktivasi tersebut memungkinkan *model* untuk menciptakan *mapping* yang bersifat kompleks diantara *input* dan *output* suatu jaringan. Hal ini penting karena *mapping* yang bersifat kompleks ini penting dalam proses pembelajaran suatu data yang lebih kompleks seperti, foto, video. Audio, dan set data lainnya yang mempunyai sifat non-linear atau mempunyai dimensi yang tinggi. Fungsi aktivasi non-linear ini berhasil menjawab permasalahan yang ada pada fungsi aktivasi linear yaitu *backpropagation* dan *stacking* untuk beberapa *layer* dalam suatu *neuron* (Gharat, 2019).

Berikut beberapa contoh fungsi aktivasi *non-linear*.

1. ReLU (*Rectified Linear Unit*)



Gambar 2.5 Grafik Fungsi dari ReLU
(Gharat, 2019)

Rumus dari Fungsi *ReLU* adalah sebagai berikut.

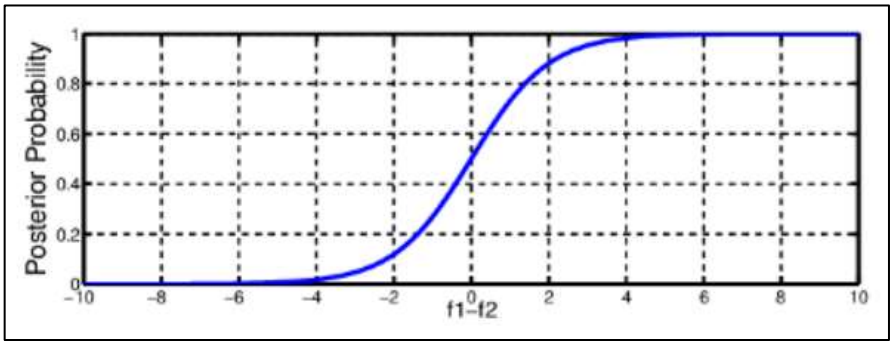
$$f(x) = a = max(0, x) \tag{2.1}$$

$$f'(x) = \{ 1 ; if\ z > 0, 0; if\ z < 0\ and\ undefined\ if\ z = 0 \} \tag{2.2}$$

Grafik fungsi dari ReLU dapat terlihat pada Gambar 2.5. Fungsi ReLU mempunyai *range* dari 0 sampai bilangan positif tak terhingga karena grafik fungsi

ReLU terus bergerak ke arah atas. Fungsi ReLU ini sangatlah efisien yang membuat proses konvergensi *network* menjadi lebih cepat. Selain itu, karena sifatnya yang non-linear, ReLU mempunyai fungsi derivasi yang memungkinkan adanya *backpropagation*. Sayangnya, ReLU masih mempunyai masalah. Ketika *input* mendekati angka 0 atau negatif, *gradient* dari fungsi tersebut menjadi 0 yang menyebabkan *network* tidak bisa melakukan *backpropagation* dan tidak bisa belajar.

2. *Softmax*



Gambar 2.6 Grafik Fungsi *Softmax*
(Gharat, 2019)

Rumus dari Fungsi *softmax* adalah sebagai berikut.

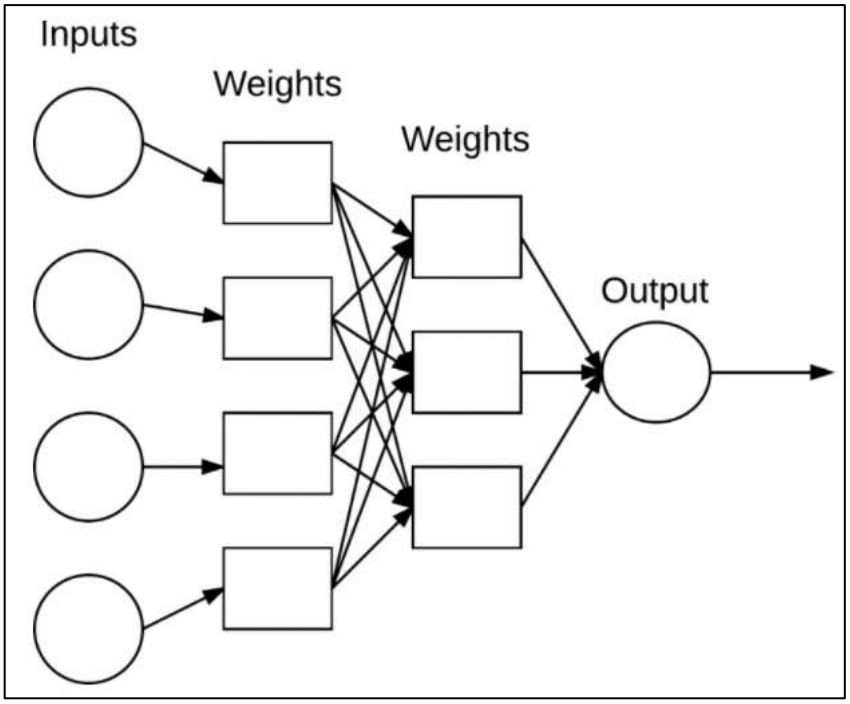
$$f(x) = e^{x}{}_{i} / (\Sigma_{j=0} e^{x}{}_{i}) \quad (2.3)$$

$$S_{j} = P(y = j|x) \quad (2.4)$$

Gambar 2.6 menujukkan grafik fungsi Softmax. Fungsi *Softmax* mengkalkulasikan distribusi probabilitas dari *event* dari ‘n’ *event* yang berbeda. Fungsi *softmax* ini mampu menangani *multiple classes* di mana hanya ada satu kelas di fungsi aktiasi lain yang menormalisasikan *output* dari tiap-tiap kelas dari 0 sampai 1, dan membaginya dari jumlahnya, memberikan probabilitas nilai *input* di dalam suatu kelas spesifik. Selain itu, fungsi aktivasi *softmax* juga berfungsi untuk *output neuron* dimaan biasanya *layer* ini digunakan untuk *output layer*, dan untuk

neural networks yang ingin mengklasifikasikan *input* ke beberapa kategori berbeda (Gharat, 2019).

Agar suatu sistem dapat ditandai sebagai *neural network*, sistem tersebut harus mempunyai struktur graf yang terlabel di mana tiap *node* didalam graf tersebut melakukan perhitungan atau komputasi sederhana. Gambar 2.6 menunjukan contoh dari graf suatu sistem arsitektur *neural network* di mana tiap *node* melakukan komputasi sederhana dan membawa sinyal perpindahan dari satu *node* ke *node* yang lain. Perpindahan ini juga ditandai dengan suatu label *weight* untuk menentukan ukuran amplifikasi atau reduksi dari sinyal tersebut (Rosebrock, 2021).



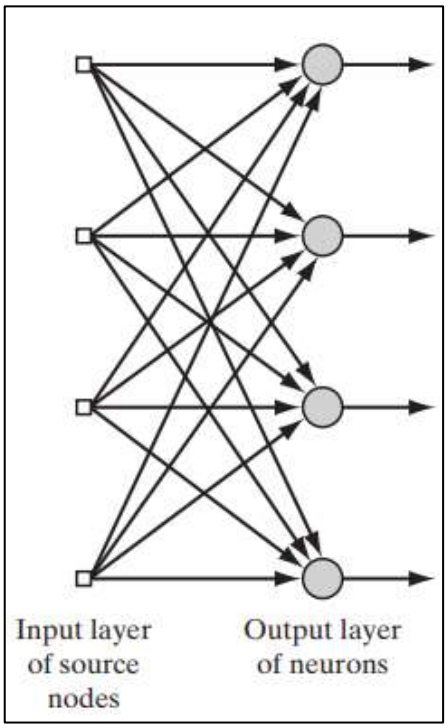
Gambar 2.7 Contoh *neural network* sederhana
(Rosebrock, 2021)

2.4.2 Arsitektur Artificial Neural Network (ANN)

Secara umum, terdapat tiga jenis arsitektur dari *neural network*.

A. Single-Layer Feedforward Network

Dalam *neural network* yang berlapis, *neuron* diatur dalam bentuk *layer*. Dengan bentuk *layered network* paling sederhana, terdapat *input layer* dari sumber *node* yang diproyeksikan langsung ke lapisan keluaran *neuron*, tetapi tidak sebaliknya. Dengan kata lain, *neural network* ini merupakan tipe *feedforward*. Gambar 2.7 mengilustrasikan kasus dari 4 *node* yang terdapat pada *input* dan *output layer*. Jaringan terseut dinamakan *single-layer network,* dengan sebutan *single-layer* mengacu pada lapisan keluaran dari *computation nodes*. Lapisan input dari *source node* tidak dihitung karena tidak ada perhitungan yang dilakukan pada *node* tersebut (Haykin, 2008).



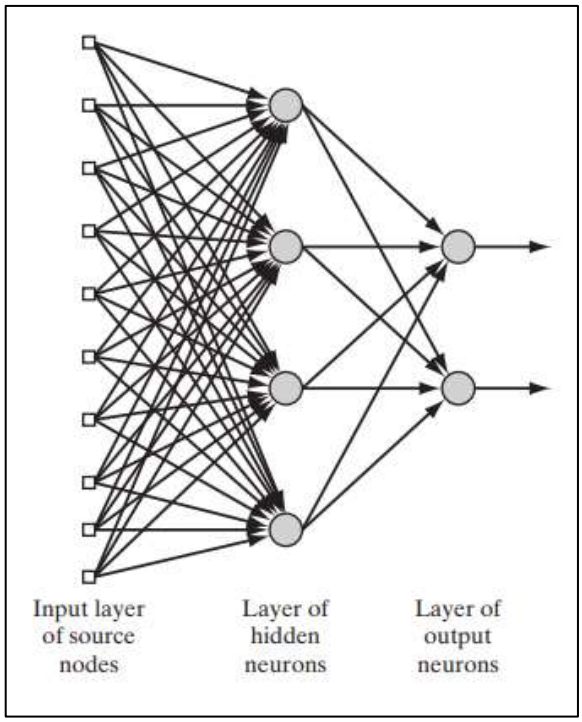
Gambar 2.8 Ilustrasi *Feedforward Network* dengan *Single Layer Neuron* (Haykin, 2008)

B. Multilayer Feedforward Network

Kelas kedua dari arsitektur ANN adalah *multilayer feedforward network* yang menggunakan *hidden layer* yang disematkan dianatara *input layer* dan *output layer*. *Hidden layer* terdiri dari *neuron hidden* yang melakukan beberapa perhitungan dari *input layer* untuk kemudian dilanjutkan kepada *output layer*, Pada suatu arsitektur *multilayer,* total dari *layer hidden* yang digunakan bisa melebihi dari satu, relatif terhadap permasalahan yang dihadapi.

Tiap koneksi dari *node* memiliki suatu nilai bobot yang berbeda-beda. Nilai bobot ini berperan untuk menghubungkan antara *input layer* dan *hidden layer*. Nilai bobot juga berfungsi sebagai penghubung antara *hidden layer* dan *output layer* dengan sistem umpan maju dan tidak sebaliknya (Wulandhari, 2017).

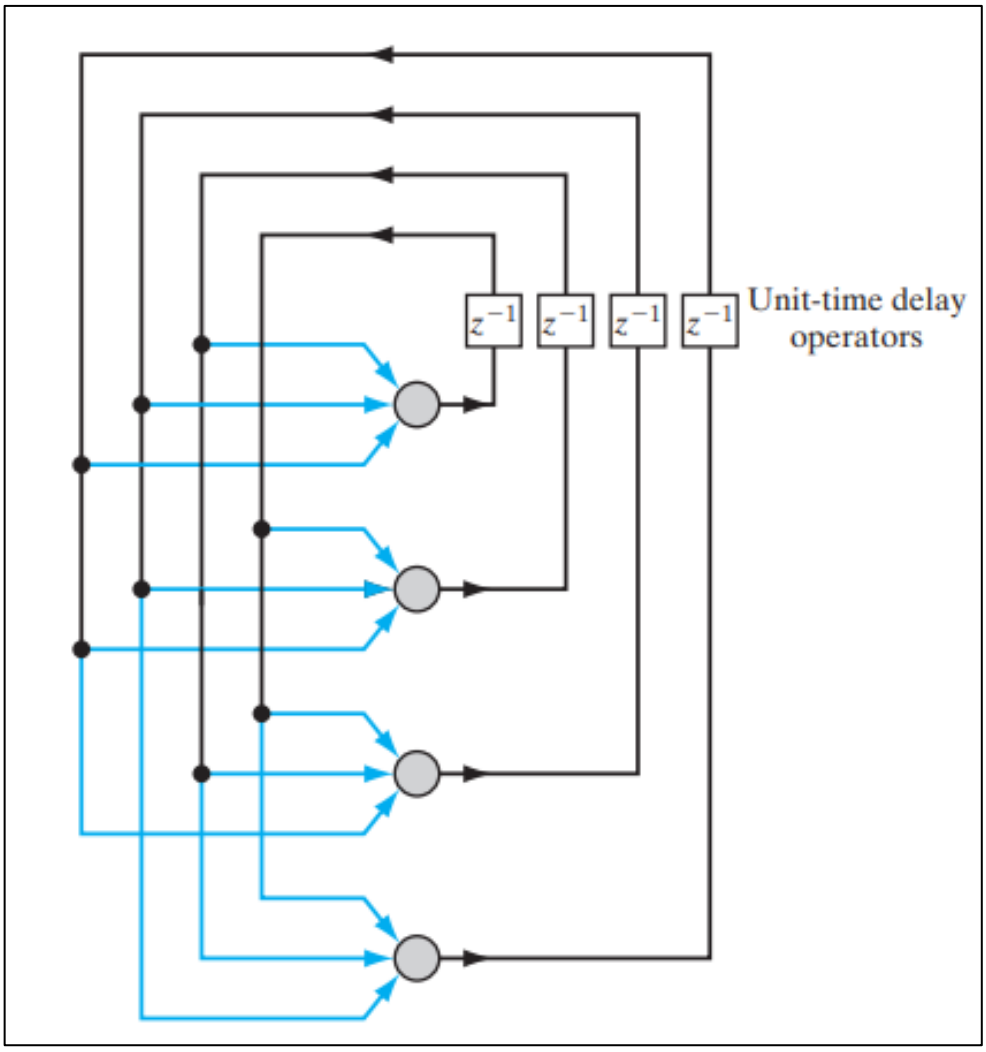
Neural network pada Gambar 2.9 mengilustrasikan *multilayer feedforward neural network* dengan satu *hidden layer*. Jaringan tersebut menggambarkan konektivitas yang utuh di mana tiap *node* yang ada didalam *layer* tersebut berhubungan satu sama lain. Ilustrasi *neural network* ini dapat disebut sebagai 10-4-2 *network* karena jaringan tersebut mempunyai 10 *source node*, 4 *hidden neuron*, dan 2 *output neuron* (Haykin, 2008).



Gambar 2.9 *Multilayer Feedforward Network* dengan satu *hidden* dan satu *output layer*
(Haykin, 2008)

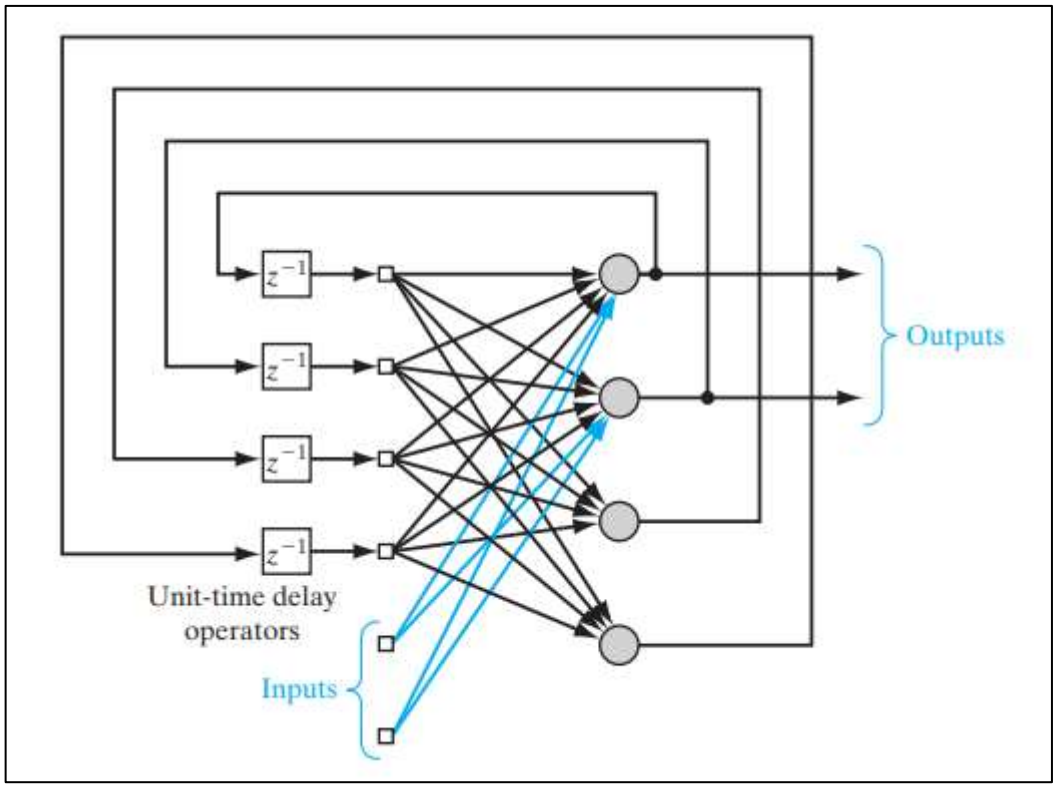
C. Recurrent Networks

Recurrent Network atau sesuai namanya *neurral network* berulang adalah *neural network* yang setidaknya memiliki satu *feedback loop*. Misalnya, *neural network* berulang ini dapat terdiri dari satu lapisan *neuron* dengan setiap *neuron* memberi sinyal *output* kembali ke *input* semua *neuron* lain seperti dengan ilustrasi pada Gambar 2.10. Dalam struktur tersebut, tidak ada sistem *self feedback loop* di dalam *neural network*.



Gambar 2.10 Recurrent network tanpa adanya *self-feedback loop* dan *hidden neuron*
(Haykin, 2008)

Gambar 2.11 mengilustrasikan *recurrent network* dengan *hidden neuron*. Koneksi *feedback* muncul dari *hidden neuron* dan *output neuron*. Kemunculakn dari *feedback loop* mempunyai dampak yang besar terhadap kemampuan latih suatu *neural network* dan performanya (Haykin, 2008).



Gambar 2.11 *Recurrent Network* dengan *Hidden Neuron*
(Haykin, 2008)

2.5 Convolutional Neural Network (CNN)

Convolutional Neural Network adalah algoritma *deep learning* yang merupakan hasil perkembangan dari *Multilayer Perceptron* (MLP) yang sama-sama diciptakan untuk mengolah data yang bersifat dua dimensi (Eka Putra, 2016). Walaupun mirip, MLP kurang cocok untuk diaplikasikan pada kasus klasifikasi citra dikarenakan ketidakmampuannya dalam menyimpan informasi spasial dari data citra. Selain itu, MLP juga mengasumsikan tiap-tiap piksel adalah suatu fitur yang bersifat independen, yang menyebabkan hasil yang tidak efisien.

2.5.1 Konsep Convolutional Neural Network (CNN)

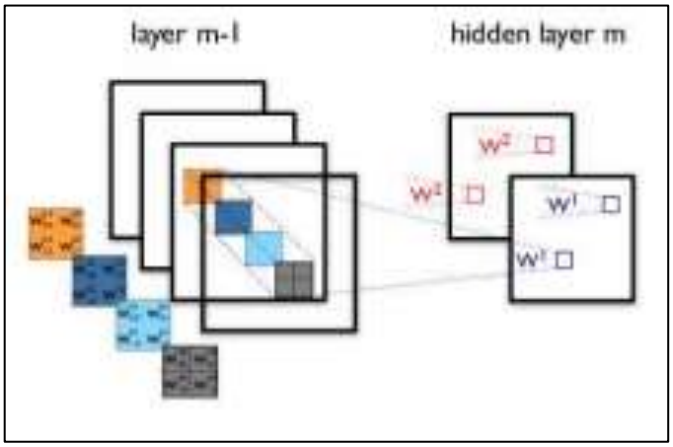
CNN mempunyai konsep yang mirip dengan MLP, hanya saja dalam CNN tiap-tiap neuron disimbolisasikan dengan bentuk dua dimensi. Di dalam CNN, data

yang terpropagasikan di dalam jaringan berbentuk dua dimensi. Artinya, operasi-operasi linear pada CNN dan parameter bobotnya akan berbeda. CNN melakukan proses konvolusi pada operasi linear, namun bobot yang ada tidak hanya satu dimensi saja tetapi berbentuk empat dimensi yang merupakan kumpulan kernel konvolusi.

Berikut adalah rumus dimensi bobot pada CNN (Eka Putra, 2016).

$$dimensi = neuron\ input\ x\ neuron\ output\ x\ tinggi\ x\ lebar \quad (2.5)$$

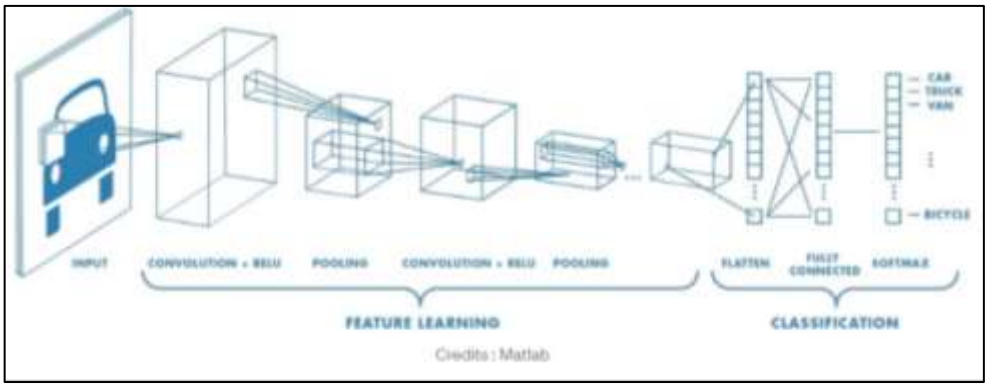
Dikarenakan adanya proses konvolusi yang terdapat pada Gambar 2.12, CNN hanya bisa diaplikasikan kedalam pengolahan data yang mempunyai struktur dua dimensi seperti suara, dan citra.



Gambar 2.12 Proses Konvolusi di CNN
(Eka Putra, 2016)

2.5.2 Arsitektur Jaringan Convolutional Neural Network (CNN)

Arsitektur jaringan CNN terdiri dari 2 bagian *layer* besar, *Feature Learning* dan *Fully Connected Layer* (MLP) yang terilustrasikan pada Gambar 2.13 (Sena, 2017).



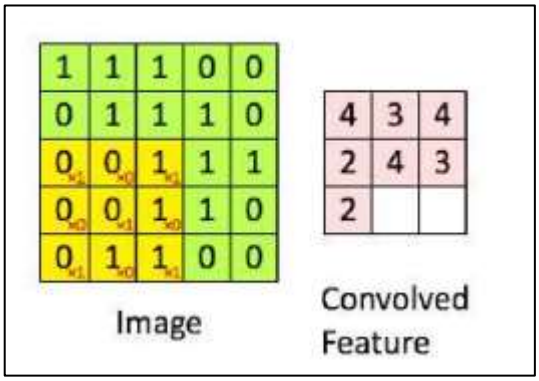
Gambar 2.13 Pembagian *Layer* pada algoritma CNN
(Sena, 2017)

A. Feature Learning Layer

Proses yang terjadi pada bagian ini ialah *encoding* citra menjadi suatu fitur berupa angka yang akan mewakilkan citra yang di *encode*. Proses ini terdiri dari dua bagian utama, *Convolutional Layer* dan *Pooling Layer*.

1. *Convolutional Layer*

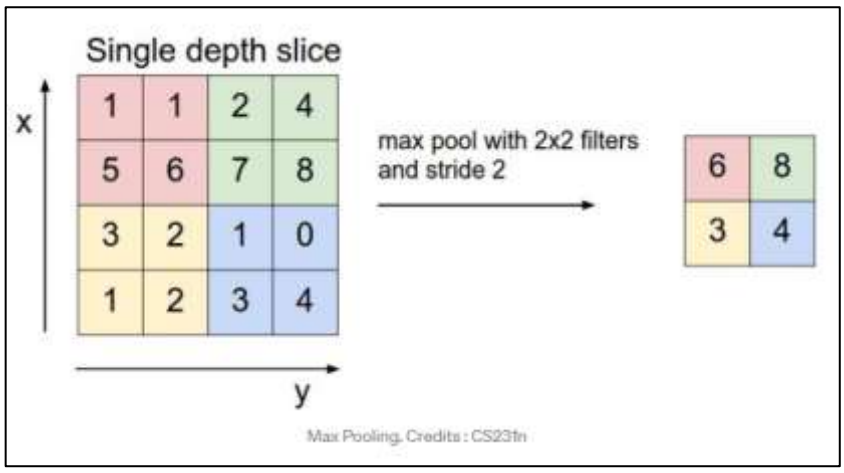
Pada proses ini, terjadi konvolusi pada *output* layer sebelumnya. Tujuan dari konvolusi ini adalah mengaplikasikan semua *kernel* pada citra seluruh *offset* yang memungkinkan untuk mengekstraksi fitur-fitur yang ada di dalam citra *input*. Proses ini terlihat pada Gambar 2.14 (Eka Putra, 2016).



Gambar 2.14 Operasi Konvolusi
(Eka Putra, 2016)

2. *Pooling Layer*

Pada proses ini, filter dengan ukuran dan *stride* (Parameter penentu jumlah pergeseran *filter)* yang akan bergeser pada seluruh kotak citra. Ide utama dari tahap ini adalah *down-sampling* yang bertujuan untuk mengurangi dimensi dari *feature map* agar mengakselerasikan perhitungan mesin karena *parameter* yang digunakan harus diperbaharui semakin sedikit untuk mengatasi *overfitting*. Hal ini bisa disamakan dengan pengurangan resolusi pada ctira (Albawi et al., 2017).



Gambar 2.15 Operasi *Pooling*
(Eka Putra, 2016)

B. Fully-Connected Layer

Feature Map yang didapatkan dari proses *feature learning* akan berbentuk sebagai *array* yang multidimensional. Oleh karena itu, terjadilah proses *flatten* yaitu proses pembentukan ulang *feature map* menjadi suatu bentuk vektor yang bisa digunakan sebagai *input* dari *fully connected layer* (Sena, 2017). Hal ini sangat penting untuk dilakukan dikarenakan tiap *neuron* pada *layer* sebelumnya harus dijadikan satu dimensi sebagai prekondisi dimasukkan ke *fully connected layer*. Hal ini dilakukan agar tidak membuat adanya *loss* pada data dan hal ini tidak *reversible*. (Eka Putra, 2016).