BAB II

TINJAUAN PUSTAKA

2.1 Citra Digital

Citra digital adalah representasi digital dari suatu gambar atau visual yang dikodekan dalam bentuk angka atau bit. Citra digital terdiri dari kumpulan piksel (elemen gambar terkecil) yang dikelompokkan menjadi baris dan kolom dalam matriks, di mana setiap piksel memiliki nilai intensitas yang menggambarkan tingkat kecerahan atau warna pada posisi tertentu dalam gambar.

> Citra digital adalah representasi dua dimensi dari suatu objek yang diwakili dalam bentuk diskret, terdiri dari kumpulan elemen gambar kecil yang disebut piksel. Setiap piksel memiliki nilai intensitas yang menggambarkan tingkat kecerahan atau warna pada posisi tertentu dalam citra. (Gonzalez et al., 2019)

Pengolahan citra digital adalah proses mengambil, memanipulasi, dan menganalisis citra digital menggunakan teknik-teknik komputasi untuk memperoleh informasi yang berguna atau menghasilkan suatu output yang diinginkan. Pengolahan citra digital dapat dilakukan dalam berbagai domain, seperti pengenalan pola, pengolahan medis, pengenalan wajah, deteksi objek, serta dalam berbagai aplikasi lainnya.

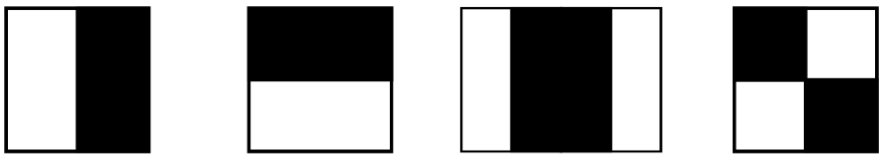
2.2 Deep Learning

Teknologi *deep learning* telah digunakan dalam berbagai aspek kehidupan. Contohnya, teknologi pengenalan wajah dan sidik jari yang digunakan pada sistem keamanan *smartphone* untuk mengenali dan membedakan ciri biometrik dari individu yang berbeda. *Deep learning* juga digunakan pada sistem penerjemahan bahasa seperti Google Translate untuk menerjemahkan bahasa secara otomatis dari satu bahasa ke bahasa lain dengan akurasi yang tinggi.

Deep learning adalah teknologi pembelajaran mesin yang menggunakan model jaringan saraf tiruan yang terdiri dari banyak lapisan untuk memproses data dan membuat prediksi. Jaringan saraf tiruan yang digunakan pada *deep learning* memiliki kemampuan untuk belajar secara mandiri dari data yang diberikan, tanpa harus diprogram secara eksplisit. Dengan menggunakan *deep learning*, mesin dapat mengenali pola-pola yang rumit dan tersembunyi dalam data, sehingga dapat digunakan dalam berbagai aplikasi seperti pengenalan wajah, pengenalan suara, analisis citra, dan sebagainya. (Goodfellow et al., 2016)

Secara umum, cara kerja *deep learning* adalah dengan mempelajari representasi data yang kompleks melalui jaringan saraf tiruan *(artificial neural network)* yang terdiri dari banyak lapisan *(layer)*. Proses ini dilakukan dengan cara memasukkan data ke dalam jaringan dan membiarkan jaringan mengenali pola-pola yang tersembunyi dalam data secara otomatis.

2.3 Haar-Cascade



Gambar 2.1 Fitur *Haar-like* pada *Haar-Cascade*

Haar-Cascade adalah metode deteksi objek yang populer dan efisien dalam bidang pengenalan objek komputer. Metode ini dikembangkan oleh Paul Viola dan Michael Jones pada tahun 2001. *Haar-Cascade* menggunakan algoritma pembelajaran mesin untuk mendeteksi objek berdasarkan fitur-fitur *Haar*.

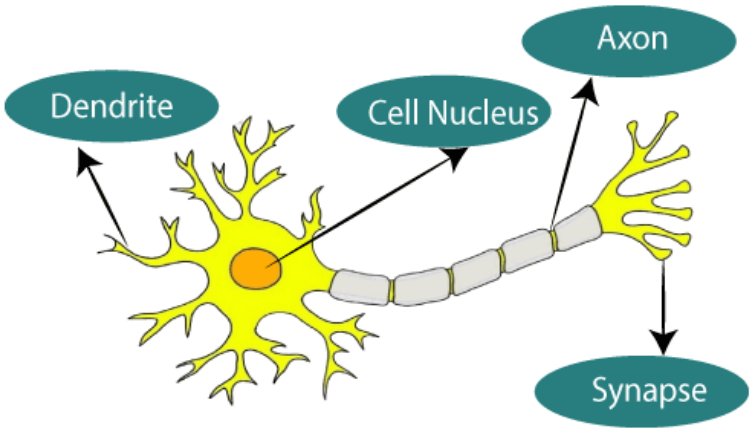
Haar-like feature akan memproses gambar dalam kotak-kotak, dimana dalam satu kotak tersebut terdapat beberapa piksel. Per kotak itu kemudian akan diproses dan akan menghasilkan perbedaan nilai yang menandakan adanya daerah gelap dan terang. Nilai-nilai inilah yang nantinya akan dijadikan dasar dalam pemrosesan gambar. (Ramadini et al., 2022)

Algoritma Viola-Jones digunakan untuk mendeteksi wajah menghadap ke depan berdasarkan identifikasi mata kiri, mata kanan, hidung, dan mulut. Jika salah satu komponen tidak teridentifikasi, maka wajahnya tidak akan dapat dikenali. Dalam kehidupan nyata, setiap orang selalu melakukan sejumlah gerakan yang membuat posisi wajah berubah saat beraktivitas. Terkadang, wajah menghadap ke depan atau menoleh ke samping. Misalnya saat wajah menoleh ke samping atau ke belakang, wajah tersebut tidak tertangkap kamera depan sehingga menyebabkan informasi pengenalan wajah tidak lengkap. (Fauzi et al., 2019)

2.4 Jaringan Saraf Tiruan

Jaringan Saraf Tiruan atau *Artificial Neural Network* adalah model komputasi yang terinspirasi oleh struktur dan fungsi jaringan saraf biologis di otak. Jaringan Saraf Tiruan terdiri dari simpul atau neuron yang saling berhubungan, yang memproses data masukan dan belajar dari contoh untuk membuat prediksi atau keputusan. (Bishop, 2006)

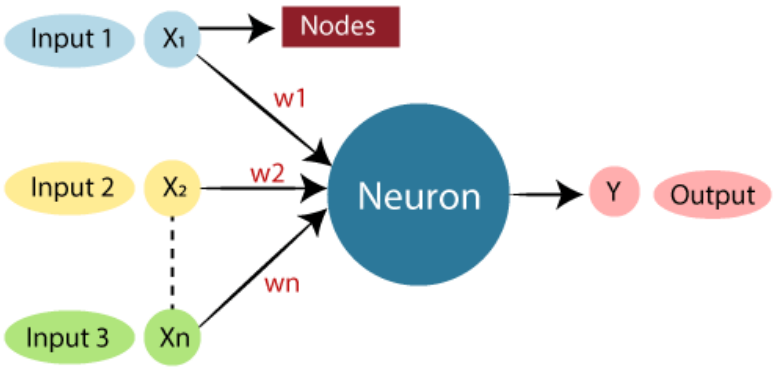
Jadi, *Artificial Neural Network* atau yang biasa dikenal dengan istilah Jaringan Saraf Tiruan adalah algoritma *deep learning* yang prinsip kerjanya dikembangkan dari jaringan saraf biologis yang membentuk struktur otak manusia. Berikut ilustrasi dari jaringan saraf biologis.



Gambar 2.2 Jaringan saraf yang terdapat pada manusia

Sama halnya dengan otak manusia yang memiliki neuron-neuron yang saling berhubungan satu sama lain, Jaringan Saraf Tiruan juga memiliki neuron-neuron yang saling berhubungan satu sama lain dalam berbagai lapisan jaringan.

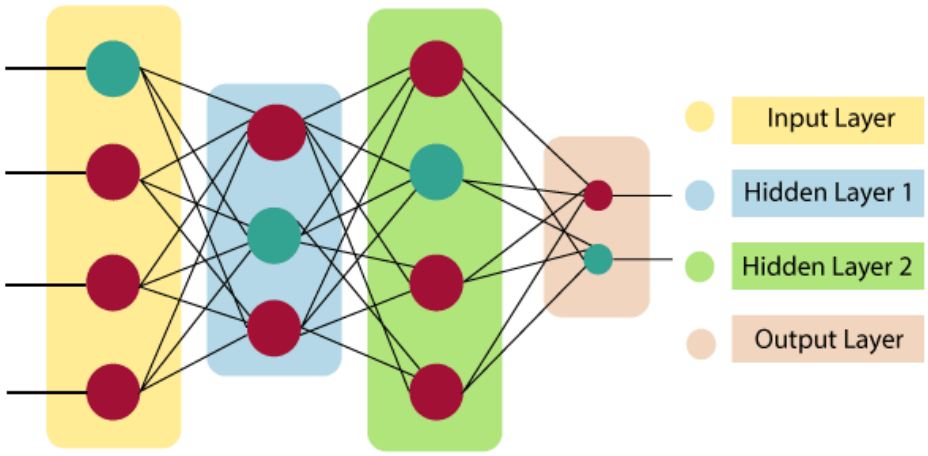
Adapun ilustrasi dari Jaringan Saraf Tiruan dapat dilihat pada gambar di bawah ini.



Gambar 2.3 Jaringan saraf pada Jaringan Saraf Tiruan

2.4.1 Arsitektur Jaringan Saraf Tiruan

Arsitektur Jaringan Saraf Tiruan menggunakan berbagai lapisan pemrosesan matematis untuk memahami informasi yang diberikan. Biasanya, jaringan saraf tiruan memiliki puluhan hingga jutaan neuron buatan yang disebut sebagai unit dan tersusun dalam beberapa lapisan atau *layer*.



Gambar 2.4 Lapisan pada arsitektur Jaringan Saraf Tiruan

Adapun lapisan pada jaringan saraf tiruan terdiri dari tiga lapisan utama yakni *input layer*, *hidden layer*, dan *output layer*.

a. Input layer

Seperti namanya, lapisan ini menerima input dalam beberapa format berbeda yang disediakan oleh programmer. Input yang terima adalah data yang ingin diproses atau dipelajari oleh jaringan saraf tiruan.

b. Hidden layer

Lapisan tersembunyi atau *hidden layer* menjembatani lapisan input dan output. Dari unit input *(input layer)*, data dapat melewati satu atau lebih *hidden layer*. Tugas *hidden layer* adalah mengubah input menjadi sesuatu yang dapat digunakan unit output. Selain itu, lapisan ini melakukan semua perhitungan untuk menemukan fitur dan pola tersembunyi. Mayoritas jaringan saraf sepenuhnya terhubung dari satu lapisan ke lapisan lainnya. Koneksi ini diberi bobot; semakin tinggi angkanya, semakin besar pengaruh satu unit terhadap unit lain, mirip dengan otak manusia. Saat data melewati setiap unit, jaringan belajar lebih banyak tentang data.

c. Output layer

Masukan atau input melewati serangkaian transformasi menggunakan *hidden layer*, pada akhirnya menghasilkan output yang dibawa menuju lapisan ini. Jaringan saraf tiruan mengambil input dan menghitung jumlah bobot dari input dan bias. Jaringan saraf tiruan mengambil input dan menghitung jumlah bobot dari input dan bias. Perhitungan ini direpresentasikan dalam bentuk fungsi transfer.

$$\sum_{i=1}^{n} W_i \times x_i + b$$

Keterangan:

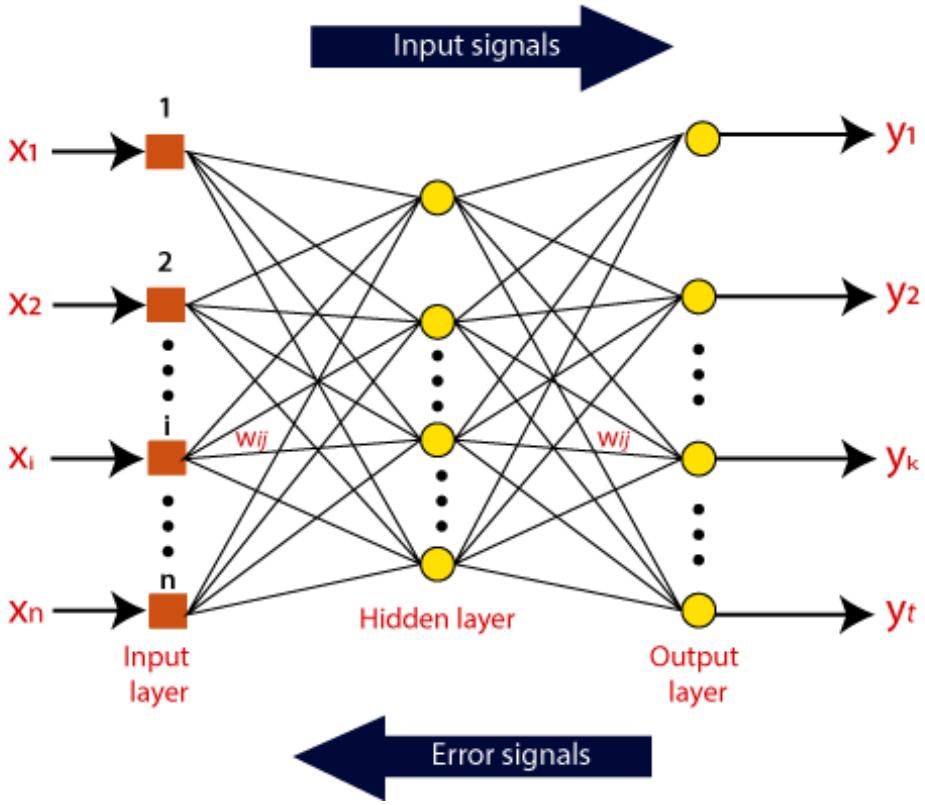
W = bobot yang dihubungkan dengan input b = bias pada neuron

x = input yang diberikan pada jaringan

Jaringan saraf tiruan menentukan bobot yang dilewatkan sebagai input ke sebuah fungsi aktivasi untuk menghasilkan keluaran *(output)*. Fungsi aktivasi memilih apakah sebuah node akan dibangkitkan atau tidak. Hanya node yang berhasil dibangkitkan yang akan dibuat sebagai *output layer*.

2.4.2 Cara Kerja Jaringan Saraf Tiruan

Jaringan Saraf Tiruan paling baik direpresentasikan sebagai graf berarah berbobot, di mana neuron buatan membentuk simpul atau node. Hubungan antara output neuron dan input neuron dapat dilihat sebagai tepi berarah dengan bobot. Jaringan syaraf tiruan menerima sinyal masukan dari sumber luar, berupa pola atau gambar yang direpresentasikan dalam vektor. Input ini kemudian secara matematis ditetapkan dengan notasi x(n) untuk setiap n jumlah input.

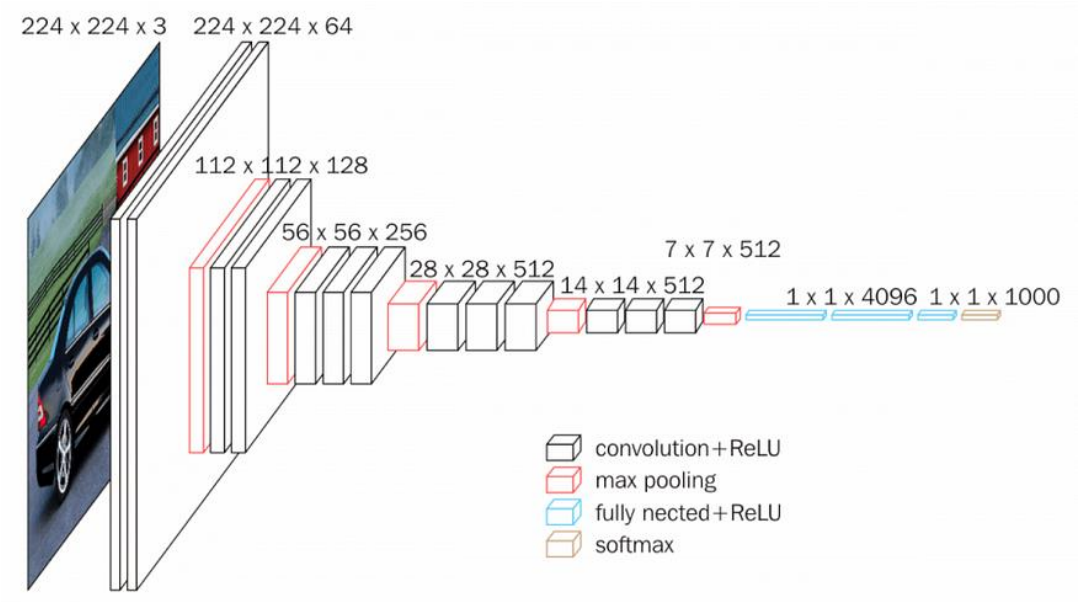


Gambar 2.5 Proses pengolahan data pada Jaringan Saraf Tiruan

Setelah itu, masing-masing input dikalikan dengan bobot yang sesuai (bobot ini adalah detail yang digunakan oleh jaringan saraf tiruan untuk memecahkan masalah tertentu). Secara umum, bobot ini biasanya mewakili kekuatan interkoneksi antar neuron di dalam jaringan saraf tiruan. Semua input berbobot diringkas di dalam unit komputasi.

Jika jumlah bobot sama dengan nol, maka bias ditambahkan untuk membuat output tidak nol atau sesuatu yang lain untuk meningkatkan respons sistem. Bias memiliki input yang sama, dan bobot sama dengan 1. Total input berbobot dapat berada dalam kisaran 0 hingga bilangan tak terhingga positif (+∞). Di sini, untuk menjaga respons dalam batas nilai yang diinginkan, nilai maksimum tertentu di-*benchmark*, dan total input berbobot dilewatkan melalui fungsi aktivasi.

2.5 Convolutional Neural Network (CNN)



Gambar 2.6 *Convolutional Neural Network (CNN)*

Convolutional Neural Network (CNN) merupakan salah satu jenis Jaringan Saraf Tiruan atau *Artificial Neural Network (ANN)* yang bertugas dalam pengenalan gambar. *Convolutional Neural Network (CNN)* adalah jenis arsitektur jaringan saraf tiruan yang dikembangkan khusus untuk memproses data dalam bentuk grid atau

matriks, seperti gambar atau data visual. CNN telah menjadi arsitektur yang sangat sukses dalam banyak tugas pengolahan gambar, seperti pengenalan gambar, deteksi objek, dan segmentasi gambar.

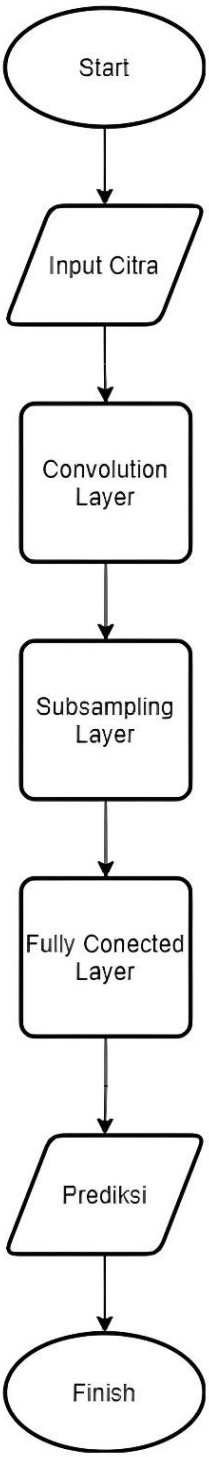
Convolutional Neural Network (CNN) adalah jenis arsitektur jaringan saraf tiruan yang didesain khusus untuk memproses data gambar, dengan menggunakan lapisan-lapisan konvolusi untuk mengekstraksi fitur dari gambar (Goodfellow et al., 2016). *Convolutional Neural Network (CNN)* merupakan jaringan saraf tiruan yang terdiri dari lapisan konvolusi dan subsampling. *Convolutional Neural Network (CNN)* secara otomatis mengekstraksi fitur dari gambar dan mengurangi dimensi data sehingga dapat dilatih dengan lebih efisien (LeCun et al., 2015).

Jadi, *Convolutional Neural Network (CNN)* adalah jenis jaringan saraf tiruan yang digunakan untuk pengolahan gambar pada *Deep Learning* melalui penggunaan lapisan konvolusi, *pooling*, dan lapisan *fully-connected*, *Convolutional Neural Network (CNN)* mampu mengekstraksi fitur-fitur penting secara hierarkis dari data gambar. Hal ini memungkinkan *Convolutional Neural Network (CNN)* untuk mempelajari representasi yang abstrak dan memahami konteks spasial dari objek yang dihadapinya. Pada beberapa penelitian terdahulu, *Convolutional Neural Network (CNN)* telah terbukti sukses dalam berbagai tugas seperti pengenalan wajah, klasifikasi gambar, deteksi objek, dan segmentasi semantik. Dengan terus berkembangnya teknologi dan penelitian di bidang ini, diharapkan *Convolutional Neural Network (CNN)* dapat terus memberikan kontribusi signifikan dalam pengolahan citra dan bidang terkait lainnya di masa depan.

2.5.1 Cara Kerja Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) yang nantinya akan digunakan sebagai metode dalam penelitian pendeteksi wajah memiliki beberapa tahap. Masing-masing tahapan memiliki peran yang penting dalam berjalannya proses CNN. Hal itulah yang nantinya akan menghasilkan model dan menjadikannya pembeda

ekspresi satu dengan yang lainnya. Dalam penggambarannya dapat dijabarkan sebagai berikut (Setiyadi, 2022):

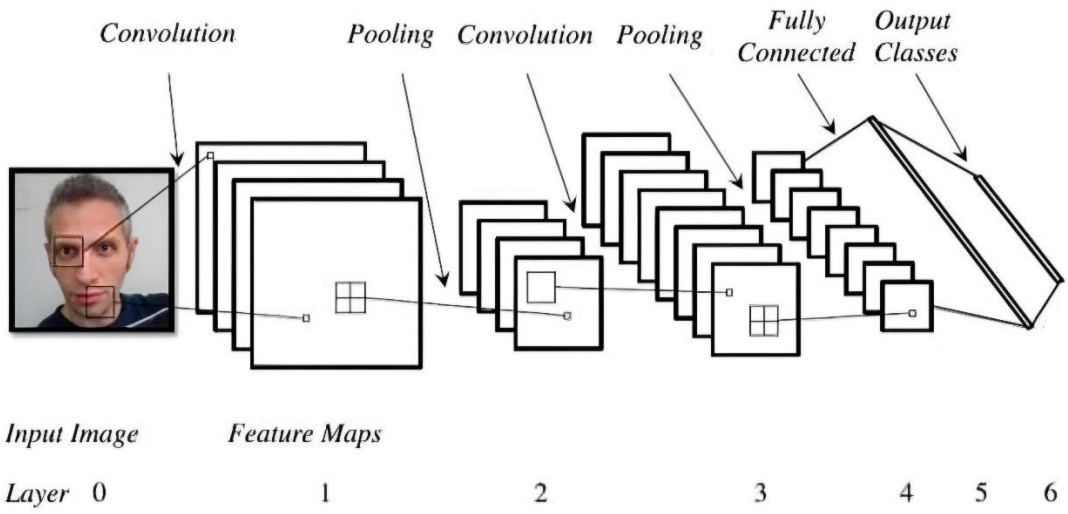


Gambar 2.7 Flowchart cara kerja *Convolutional Neural Network (CNN)*

Pada diagram tersebut digambarkan bagaimana cara kerja sebuah metode CNN. Pada tahapan pertama setelah start adalah tahapan menginput citra berupa gambar wajah yang berekspresi. Selanjutnya akan diproses menggunakan metode CNN untuk nantinya bisa diteruskan ke tahapan *Convolution Layer* dan *Subsampling Layer*, dan prediksi akan dilakukan secara *fully-connected layer*.

2.5.2 Arsitektur Convolutional Neural Network (CNN)

Setiap gambar kecil hasil dari tahap pemecahan gambar, digunakan sebagai input untuk menghasilkan sebuah output yang mewakili suatu fitur. Proses ini memberikan sebuah kemampuan terhadap CNN untuk mengenali suatu objek. Adapun gambaran dari arsitektur CNN yang dimaksud adalah sebagai berikut (Setiyadi, 2022):



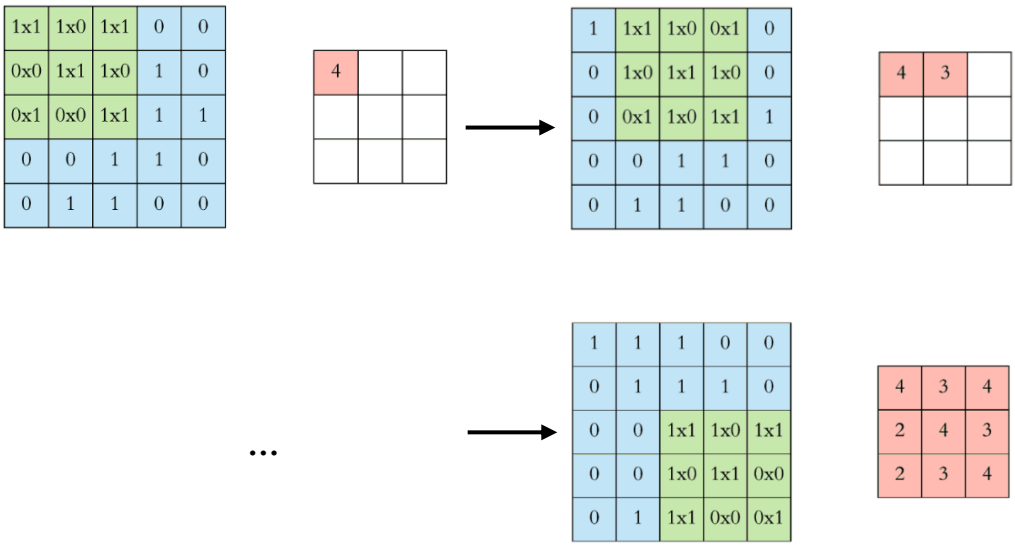
Gambar 2.8 Arsitektur *Convolutional Neural Network (CNN)*

Arsitektur di atas merupakan gambaran singkat bagaimana sebuah CNN bekerja dalam mengekstraksi sebuah gambar.

2.5.2.1 Convolution Layer

Convolution Layer merupakan salah satu jenis lapisan pada arsitektur *Convolutional Neural Network (CNN)* yang biasa digunakan pada tugas-tugas *computer vision* seperti pengenalan gambar dan deteksi objek. Pada tahapan ini terjadi *convolution layer* yaitu tahapan yang membuat sebuah citra dikonvolusi melalui gabungan filter berkonvolusional, dan konvolusi terjadi pada input citra menggunakan filter dengan kernel yang bobotnya sudah ditentukan. Kemudian setelah itu proses tersebut menghasilkan output yang disebut *activation map* atau *feature map*.

Cara kerja *convolution layer* adalah dengan melakukan operasi konvolusi pada *input layer* menggunakan filter atau kernel yang telah ditentukan. Filter ini biasanya berukuran kecil dan digunakan untuk melakukan operasi konvolusi secara bergeser-geser pada seluruh area *input layer*. Hasil dari operasi konvolusi ini kemudian dijumlahkan dan diaplikasikan fungsi aktivasi.

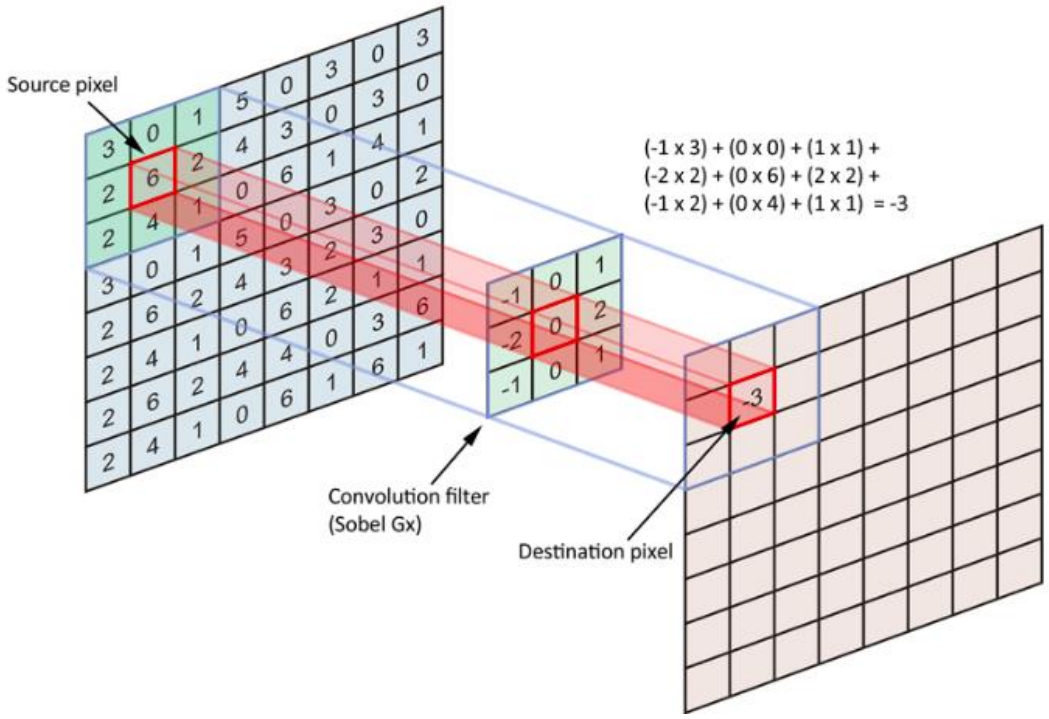


Gambar 2.9 Proses pergeseran *input layer* pada operasi konvolusi

Untuk memudahkan penjelasan, visualiasi di atas berbentuk 2 dimensi. Tetapi yang sebenarnya terjadi adalah CNN berjalan dalam 3 dimensi. Pada saat melakukan operasi konvolusi, filter akan diaplikasikan pada setiap bagian *input layer* secara bertahap dan bergeser-geser. Proses ini akan menghasilkan sebuah feature map,

yaitu representasi *input layer* yang dihasilkan dari penggunaan filter tersebut. Semakin banyak filter yang digunakan, semakin banyak juga feature map yang dihasilkan.

Tujuan dari *convolution layer* adalah untuk mengekstraksi fitur-fitur penting dari *input layer* yang kemudian akan digunakan oleh *layer* atau lapisan berikutnya pada arsitektur CNN untuk melakukan klasifikasi atau deteksi objek. Dalam proses pembelajaran, bobot atau parameter dari filter akan disesuaikan atau diperbarui secara otomatis melalui proses *backpropagation* agar menghasilkan hasil yang lebih baik.



Gambar 2.10 *Convolution Layer*

Setiap gambar direpresentasikan sebagai matriks 3D dengan dimensi *width*, *height*, dan *depth*. Depth merupakan dimensi karena adanya warna yang digunakan dalam gambar (RGB). Beberapa operasi *convolution* pada data input akan dilakukan menggunakan filter yang berbeda. Hasilnya juga akan disimpan dalam peta fitur yang berbeda. Namun pada akhirnya, semua peta fitur ini akan diambil dan digabungkan sebagai hasil akhir dari *convolution layer*.

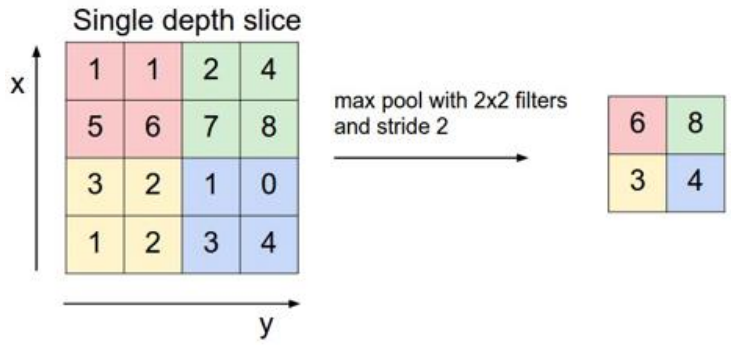
Sama seperti *Neural Network* lainnya, kita menggunakan fungsi aktivasi *(activation function)* untuk membuat output kita menjadi non-linear. Dalam kasus *Convolutional Neural Network*, output dari operasi *convolution* akan dilewatkan melalui fungsi aktivasi. Salah satunya menggunakani fungsi aktivasi *Rectified Linear Unit (ReLU)*. *Rectified Linear Unit (ReLU)* merupakan fungsi aktivasi yang memperkenalkan non-linearitas ke dalam jaringan saraf, memungkinkannya untuk belajar dan mendekati pola dan hubungan yang kompleks dalam data.

Definisi matematika dari fungsi aktivasi ReLU cukup sederhana, yaitu : f(x) = maks (0, x). Dalam persamaan ini, "x" mewakili nilai masukan ke fungsi aktivasi, dan "f(x)" adalah hasil keluaran dari fungsi tersebut. Fungsi ReLU yaitu mengembalikan nilai masukan jika nilainya lebih besar atau sama dengan nol, dan mengembalikan nilai nol untuk nilai masukan negatif. Dengan kata lain, ReLU mempertahankan nilai positif tanpa perubahan dan mengatur nilai negatif menjadi nol (Goodfellow et al., 2016).

Setelah *convolution layer*, biasanya ditambahkan lapisan penyatuan *(pooling layer)* di antara *layer* CNN. Fungsi *pooling* adalah untuk mengurangi dimensi secara terus menerus serta mengurangi jumlah parameter dan komputasi dalam jaringan. Hal ini akan mempersingkat waktu training dan mengontrol terjadinya *overfitting*.

2.5.2.2 Max-Pooling

Pada langkah sebelumnya, gambar sudah dipecah menjadi bagian yang kecil akan tetapi array yang dihasilkan masih terbilang cukup besar. Oleh karena itu, pada proses ini dilakukannya sebuah *downsampling* yang mana penggunaannya dinamakan *max-pooling* (Setiyadi, 2022). Pooling ini mengambil nilai maksimum di setiap *window*. Ukuran *window* ini perlu ditentukan sebelumnya. Operasi ini mengurangi ukuran peta fitur dan hanya akan menyimpan informasi penting dan signifikan dalam proses klasifikasi nantinya. Yaitu mengambil nilai piksel paling besar di setiap *pooling* kernel untuk mengambil informasi terpenting dengan tidak mengurangi jumlah parameter.



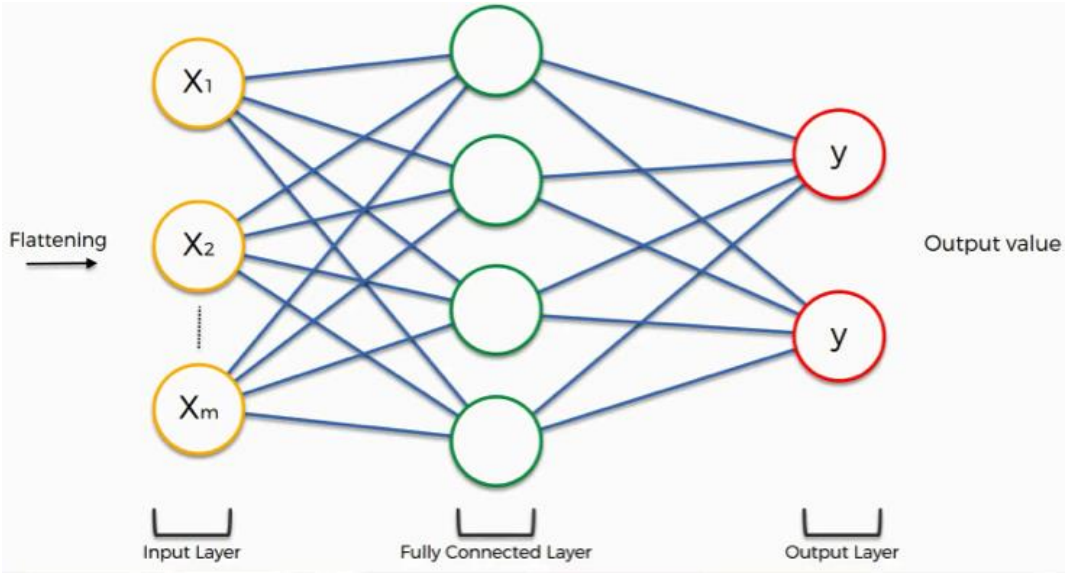
Gambar 2.11 Cara kerja *max-pooling*

Cara kerja *max-pooling* adalah dengan membagi setiap *feature map* yang dihasilkan oleh *convolution layer* menjadi *sub-region* yang lebih kecil, lalu mencari nilai maksimum dari setiap *sub-region* tersebut. Kemudian, nilai maksimum tersebut akan diambil dan disimpan pada *output layer* (LeCun et al., 2015). Proses ini akan membantu untuk mengurangi dimensi dari feature map sehingga akan mempercepat proses komputasi selanjutnya pada arsitektur CNN.

Pada saat proses pembelajaran, *max-pooling* tidak memiliki parameter yang perlu di-update. Namun, dimensi *sub-region* yang digunakan pada *max-pooling* dapat disesuaikan agar cocok dengan karakteristik dari dataset yang digunakan. Semakin besar dimensi *sub-region* yang digunakan, semakin banyak juga informasi yang akan hilang, namun semakin kecil dimensinya maka akan semakin banyak informasi yang dipertahankan. Oleh karena itu, pilihan dimensi *sub-region* pada *max-pooling* harus disesuaikan dengan tujuan dan karakteristik dari tugas yang ingin diselesaikan.

2.5.2.3 Prediksi

Setelah melakukan tahapan sebelumnya, langkah terakhir yang dilakukan adalah membuat prediksi. Tahapan ini merupakan proses akhir dari CNN. Proses ini diawali dengan menginputkan array kecil yang sudah di downsampling ke dalam jaringan saraf lain. Langkah ini biasa disebut dengan *fully-connected network* (Setiyadi, 2022).



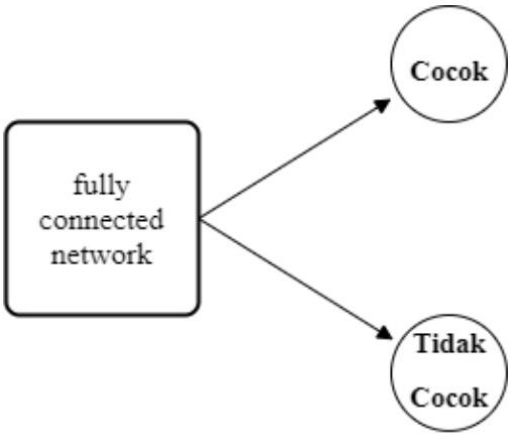
Gambar 2.12 Cara kerja *fully-connected layer*

Di atas digambarkan proses *fully-connected layer* secara rinci. Yang mana pada akhirnya sebuah citra akan diambil sebuah keputusan apakah cocok atau tidak. *Fully-connected layer* atau juga dikenal sebagai *dense layer* adalah jenis lapisan pada arsitektur *neural network* yang menghubungkan setiap neuron pada satu lapisan dengan setiap neuron pada *layer* berikutnya secara penuh atau full.

Cara kerja *fully-connected layer* adalah dengan mengubah input yang telah dihasilkan oleh lapisan atau *layer* sebelumnya menjadi sebuah vektor, kemudian melakukan operasi *dot product* antara vektor input dengan setiap vektor bobot atau *weight* yang terdapat pada lapisan tersebut. Hasil dari operasi *dot product* kemudian akan ditambahkan dengan bias dan diaplikasikan fungsi aktivasi. Proses ini akan menghasilkan output yang merupakan representasi fitur-fitur dari vektor input yang telah diubah dan diolah melalui lapisan sebelumnya. *Fully-connected layer* biasanya digunakan pada bagian akhir dari arsitektur *neural network* untuk melakukan klasifikasi atau regresi terhadap input yang telah diberikan.

Pada saat proses pembelajaran, bobot atau parameter pada *fully-connected layer* akan disesuaikan atau di-update secara otomatis melalui proses *backpropagation* untuk mengoptimalkan hasil prediksi. *Backpropagation* adalah metode untuk menentukan parameter pada setiap *layer* dari sebuah jaringan saraf tiruan *(neural*

network) dengan menghitung gradien atau perubahan nilai *loss* dari output ke input. Metode ini memungkinkan arsitektur *neural network* untuk mempelajari pola atau fitur-fitur pada data dan menyesuaikan parameter pada setiap lapisan untuk menghasilkan prediksi yang lebih baik. Proses ini akan dilakukan pada setiap iterasi pembelajaran dengan tujuan untuk meminimalkan nilai fungsi *loss*.



Gambar 2.13 Prediksi pada *fully-connected network*

Pada *output layer* digunakan fungsi aktivasi *softmax*. Tujuan dari *softmax* adalah untuk mengubah nilai numerik dari neuron output menjadi probabilitas kelas yang berbeda. Fungsi ini mengambil sejumlah nilai masukan dan mengubahnya menjadi probabilitas yang berjumlah satu. Probabilitas untuk setiap kelas dihitung berdasarkan eksponensial dari nilai masukan. Fungsi *softmax* didefinisikan sebagai (Goodfellow et al., 2016):

$$softmax(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}$$

$(x)_i$ = nilai kelas ke-i

exp (x_i) = eksponensial dari nilai kelas i

n = jumlah total kelas

exp (x_j) = eksponensial dari nilai kelas j

2.5.3 Optimizers

Optimizers adalah algoritma-algoritma yang digunakan untuk mengoptimalkan dan memperbarui parameter-parameter model pada proses pelatihan jaringan saraf *(neural networks)*. Tujuan utama *optimizers* adalah untuk mencari nilai-nilai parameter model yang meminimalkan fungsi kerugian *(loss function)* yang didefinisikan.

Algoritma Adam *(Adaptive Moment Estimation)* menggabungkan dua konsep utama: momentum (yaitu mempertimbangkan gradien sebelumnya) dan *adaptive learning rates* (yaitu menyesuaikan learning rate untuk setiap parameter). Metode ini efisien secara komputasi, memiliki sedikit kebutuhan memori, invarian terhadap penskalaan diagonal gradien, dan cocok untuk masalah yang besar dalam hal data/parameter. (Kingma et al., 2014)

Adam menggunakan momentum dari gradien sebelumnya untuk membantu memperbarui parameter lebih efisien, dan juga menggunakan *adaptive learning rates* yang mengadaptasi *learning rate* secara individu untuk setiap parameter berdasarkan perubahan historis gradien. Penggunaan Adam dalam pelatihan jaringan saraf dapat membantu meningkatkan kecepatan konvergensi dan mengatasi beberapa masalah yang sering terjadi dalam *optimizers* lain, seperti masalah *learning rate* yang terlalu besar atau terlalu kecil.

2.5.4 Losses

losses (kerugian) atau juga dikenal sebagai *loss function* adalah fungsi-fungsi yang digunakan untuk mengukur seberapa jauh prediksi model dari nilai yang sebenarnya. Tujuan dari *loss function* adalah untuk memberikan umpan balik *(feedback)* kepada model sehingga model dapat belajar untuk melakukan prediksi yang lebih baik pada tugas yang diberikan.

Cross-entropy adalah suatu metrik yang digunakan untuk mengukur perbedaan antara distribusi probabilitas yang diprediksi oleh model dan distribusi probabilitas yang sebenarnya dari data yang diamati. *Binary Crossentropy* digunakan untuk masalah klasifikasi dua kelas, sedangkan *Categorical Crossentropy* adalah salah satu *loss function* yang digunakan untuk masalah klasifikasi banyak kelas. (Chollet, 2021)

2.6 Parameter, Hyperparameter, dan Hyperparameter Tuning

Parameter adalah nilai yang mengontrol perilaku sistem (Goodfellow et al., 2016). Parameter berisi informasi yang dipelajari oleh model dari data pelatihan (Chollet, 2021). Model *machine learning* mempelajari nilai-nilai parameter ini dari data pelatihan untuk melakukan prediksi atau mengambil keputusan pada data baru (Bishop, 2006). Dalam banyak kasus, parameter ini diatur secara otomatis oleh algoritma saat model dilatih. Contoh parameter dalam model *machine learning* seperti bobot *(weights)* dalam jaringan saraf *(neural networks)*.

Hyperparameter adalah variabel yang mengontrol distribusi parameter model. *Hyperparameter* berperan dalam mengatur bagaimana proses pelatihan berlangsung. Pemilihan *hyperparameter* yang tepat dapat mempengaruhi performa model secara keseluruhan (Bishop, 2006). Contoh *hyperparameter* meliputi jumlah iterasi *(epoch)* dalam pelatihan, tingkat kecepatan pembelajaran *(learning rate)*, jumlah *hidden layer* dalam *neural network*, jumlah neuron dalam setiap lapisan, dan lain-lain. Jadi, *parameter* adalah variabel internal yang dipelajari oleh model dari data, sementara *hyperparameter* adalah variabel eksternal yang harus ditentukan sebelum pelatihan dan dapat mempengaruhi kinerja model secara keseluruhan.

Proses penyetelan *hyperparameter* pada *hyperparameter tuning* melibatkan eksplorasi secara sistematis berbagai kombinasi nilai *hyperparameter* dan mengevaluasi kinerja model menggunakan dataset validasi. Tujuannya adalah untuk menemukan kombinasi *hyperparameter* yang menghasilkan hasil terbaik dalam hal akurasi, *loss*, atau metrik evaluasi relevan lainnya.

Penyetelan *hyperparameter* adalah langkah penting dalam membangun model pembelajaran mesin yang kuat dan berkinerja tinggi, karena membantu menemukan konfigurasi terbaik yang memungkinkan model untuk mengeneralisasi data yang belum pernah dilihat sebelumnya dan menghindari *overfitting* ataupun *underfitting*.

2.7 Evaluation Metrics dan Confusion Matrix

Metrics adalah ukuran atau evaluasi kinerja model yang digunakan untuk mengevaluasi seberapa baik model bekerja pada data yang tidak digunakan dalam pelatihan (data validasi atau data uji) (Chollet, 2021). *Metrics* yang digunakan pada model yaitu *Metrics Accuracy* yang digunakan untuk menghitung persentase data yang diprediksi dengan benar dari total data yang dievaluasi.

Accuracy adalah metrik yang mengukur seberapa baik model dapat memprediksi dengan benar data pada dataset evaluasi (data validasi atau data uji) dibandingkan dengan label yang sebenarnya. Akurasi adalah rasio jumlah prediksi yang benar terhadap jumlah total sampel masukan. (Yalug et al., 2021)

Confusion matrix adalah alat yang digunakan untuk mengukur kinerja suatu model dalam masalah klasifikasi. Ini merupakan metode evaluasi yang penting untuk menggambarkan performa model dan memahami sejauh mana model tersebut mampu mengklasifikasikan data dengan benar. *Confusion matrix* menampilkan jumlah data yang diklasifikasikan dengan benar dan yang salah ke dalam kategori-kategori berikut (Karimi, 2021):

a. *True Positive (TP)*: Jumlah contoh positif yang benar diklasifikasikan sebagai positif oleh model.
b. *True Negative (TN)*: Jumlah contoh negatif yang benar diklasifikasikan sebagai negatif oleh model.
c. *False Positive (FP)*: Jumlah contoh negatif yang salah diklasifikasikan sebagai positif oleh model (sering disebut sebagai *"Type I error"* atau kesalahan tipe I).

d. *False Negative (FN)*: Jumlah contoh positif yang salah diklasifikasikan sebagai negatif oleh model (sering disebut sebagai *"Type II error"* atau kesalahan tipe II).

Secara umum, *confusion matrix* akan berbentuk seperti ini:

	Prediksi Positif	Prediksi Negatif
Aktual Positif	TP	FN
Aktual Negatif	FP	TN

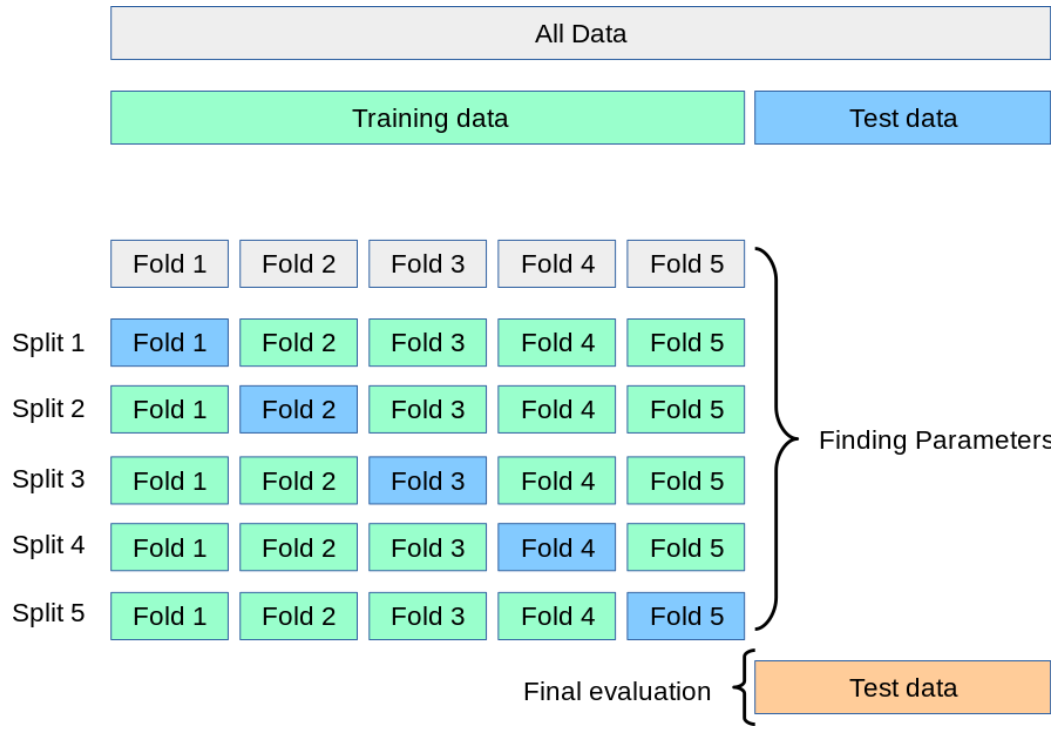
Tabel 2.1 Confusion Matrix

Dengan *confusion matrix*, kita dapat menghitung berbagai metrik evaluasi, seperti akurasi, yaitu:

$$\text{Akurasi} = \frac{(TP + TN)}{(TP + TN + FP + FN)} \quad \text{atau} \quad \frac{\text{Jumlah prediksi benar}}{\text{Jumlah keseluruhan data}}$$

2.8 Cross-Validation

Cross-validation (CV) adalah teknik yang digunakan untuk mengevaluasi kinerja model atau algoritma secara objektif. Tujuan utamanya adalah untuk memperkirakan seberapa baik model dapat generalisasi ke data yang belum pernah dilihat sebelumnya. Teknik ini melibatkan pembagian data menjadi beberapa subset atau lipatan *(folds)*, di mana setiap lipatan digunakan sebagai data pelatihan dan data validasi bergantian. Tujuan utama dari *cross-validation* adalah untuk mengukur seberapa baik model dapat umumnya menggeneralisasi pada data yang belum pernah dilihat sebelumnya. *Cross-Validation* adalah pilihan yang tepat ketika memiliki terlalu sedikit sampel untuk validasi agar dapat diandalkan. (Chollet, 2021)



Gambar 2.14 Metode *K-Fold* pada *Cross-Validation*

Proses *cross-validation* melibatkan pemisahan data menjadi beberapa subset atau *fold*. Kemudian, model dilatih pada beberapa subset data dan diuji pada subset lainnya. Secara umum, metode *cross-validation* melibatkan langkah-langkah berikut:

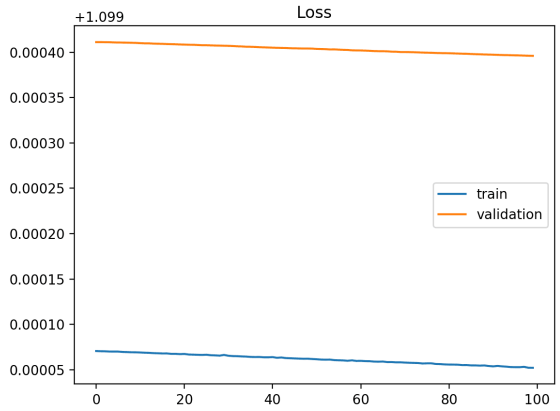
a. Pembagian data: Data dibagi menjadi k subset yang relatif seimbang. Biasanya, k = 5 atau 10, tetapi angka ini dapat berbeda tergantung pada ukuran dataset dan preferensi pengguna.
b. Pelatihan dan pengujian: Model dilatih pada k-1 subset dan diuji pada subset yang tersisa. Proses ini diulang k kali, sehingga setiap subset digunakan sebagai data pengujian tepat sekali.
c. Evaluasi kinerja: Hasil dari setiap pengujian diukur dan diakumulasi untuk memberikan nilai kinerja keseluruhan model.
d. Pengambilan rata-rata: Hasil akhir adalah rata-rata dari k hasil pengujian, yang memberikan perkiraan kinerja yang lebih andal daripada menggunakan satu pemisahan data saja.

Keuntungan dari *cross-validation* adalah mengurangi bias dan variasi dalam proses evaluasi model. Dengan menguji model pada berbagai subset data, *cross-validation* memberikan gambaran yang lebih baik tentang kinerja model secara keseluruhan dan sejauh mana model dapat digeneralisasi ke data baru.

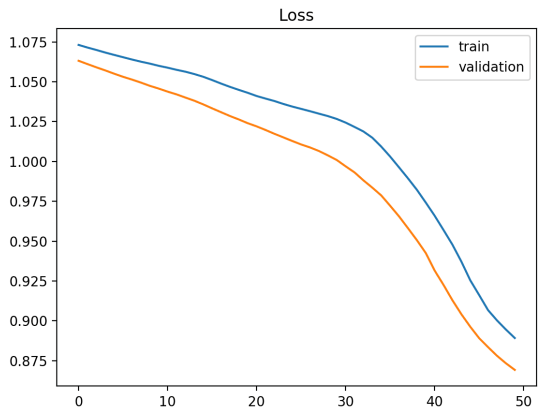
2.9 Mendiagnosa Performa Model

Dalam hal ini, diagnosa performa model dapat dilakukan dengan melalui kurva training dan validasi yang telah dilakukan pada tahapan sebelumnya. Ada tiga jenis kondisi umum yang cenderung diamati dalam kurva pembelajaran *machine learning*, yaitu (Brownlee, 2019):

a. *Underfitting*, yaitu kondisi yang terjadi ketika model tidak dapat mempelajari pola yang ada dalam data pelatihan dengan baik. Dalam hal ini, model tidak hanya memiliki performa yang buruk pada data pelatihan, tetapi juga pada data pengujian. *Underfitting* terjadi jika nilai *training loss* tetap datar pada proses pelatihan, atau nilai *training loss* tidak berkurang secara signifikan hingga akhir pelatihan.



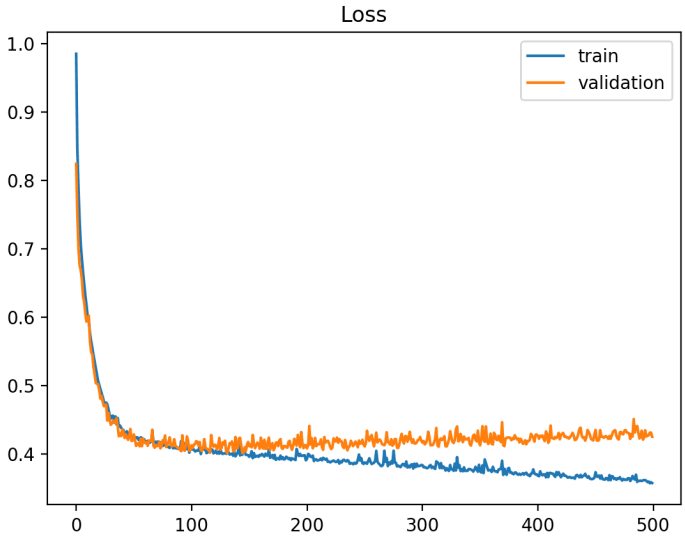
a. nilai training dan validasi tetap datar pada proses pelatihan.



b. nilai training dan validasi tidak berkurang secara signifikan hingga akhir pelatihan.

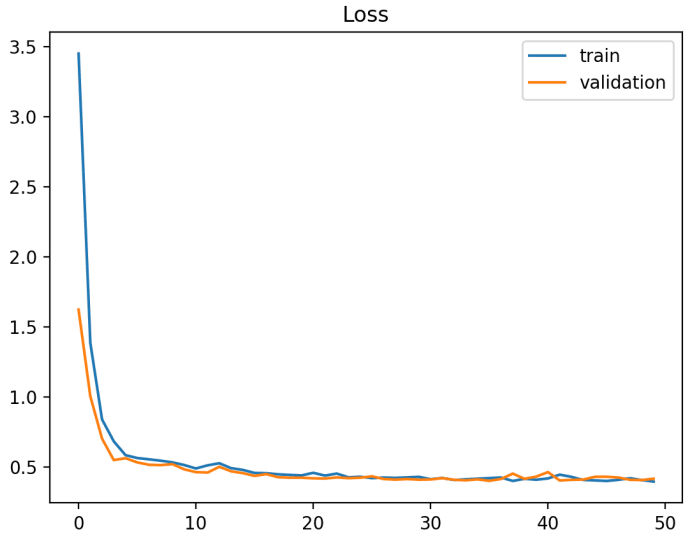
Gambar 2.15 Contoh kurva pelatihan yang mengalami *underfitting*

b. *Overfitting*, yaitu kondisi yang terjadi ketika model terlalu disesuaikan dengan data pelatihan, sehingga model hanya mengandalkan kemampuan "menghafal" data pelatihan dan memiliki kinerja yang baik pada data pelatihan. Namun, saat diterapkan pada data pengujian, model memiliki performa yang buruk karena tidak dapat menggeneralisasi dengan baik ke data yang belum pernah dilihat sebelumnya. *Overfitting* terjadi jika nilai *training loss* terus berkurang sedangkan nilai *validation loss* berkurang ke titik tertentu dan mulai meningkat melebihi nilai *training*.



Gambar 2.16 Contoh kurva pelatihan yang mengalami *overfitting*

c. *Good fit*, yaitu kondisi ketika model berhasil belajar dari data pelatihan dengan baik dan memiliki kemampuan yang baik dalam menggeneralisasi pada data pengujian yang belum pernah dilihat sebelumnya. Kondisi *good fit* terjadi jika nilai *training loss* dan nilai *validation loss* menurun ke titik stabilitas dan memiliki kesenjangan kecil antara nilai keduanya.



Gambar 2.17 Contoh kurva pelatihan *good fit*

2.10 Generalisasi

Proses generalisasi adalah ukuran sejauh mana model mampu menghindari *overfitting* (terlalu khusus untuk data pelatihan) dan *underfitting* (tidak cukup memahami pola yang ada dalam data). Idealnya, model *machine learning* harus dapat menangkap pola umum dari data pelatihan dan kemudian mengaplikasikan pola ini pada data baru untuk memberikan hasil yang akurat. Salah satu teknik yang dapat digunakan untuk generalisasi model adalah Regularisasi.

Regularisasi adalah teknik yang bertujuan untuk mengurangi kompleksitas jaringan. Cara umum teknik regularisasi yaitu dengan menerapkan *regularization* dan *dropout layer*.

a. Cara umum untuk mengurangi *overfitting* adalah dengan membatasi kompleksitas model dengan memaksa bobotnya untuk hanya mengambil nilai kecil, yang membuat distribusi nilai bobot lebih *regular* (teratur). Ini disebut regularisasi bobot, dan dilakukan dengan menambahkan ke fungsi loss dari model dengan *cost* atau biaya terkait dengan bobot yang besar. Biaya ini terdiri dari 2 jenis: Regularisasi L1 — Biaya yang ditambahkan

adalah sebanding dengan nilai mutlak dari koefisien bobot (norma L1 dari bobot); dan Regularisasi L2 — Biaya yang ditambahkan adalah sebanding dengan kuadrat dari nilai koefisien bobot (norma L2 dari bobot). Regularisasi L2 juga disebut sebagai *weight decay* dalam konteks jaringan saraf. (Chollet, 2021)

b. *Dropout* digunakan untuk secara acak "mematikan" atau "mengabaikan" neuron-neuron dengan probabilitas yang telah ditentukan sebelumnya. Ketika neuron-neuron secara acak "dimatikan" dari jaringan selama proses pembelajaran, neuron-neuron lainnya harus mengambil alih dan mengelola representasi yang diperlukan untuk membuat prediksi untuk neuron-neuron yang hilang. Dengan demikian, dengan jumlah neuron yang lebih sedikit, jaringan menjadi lebih responsif dan dapat belajar dengan lebih cepat. Pada akhir sesi pembelajaran, neuron-neuron yang sebelumnya "dimatikan" diaktifkan kembali (dengan bobot asli mereka). (Jabir et al., 2021)