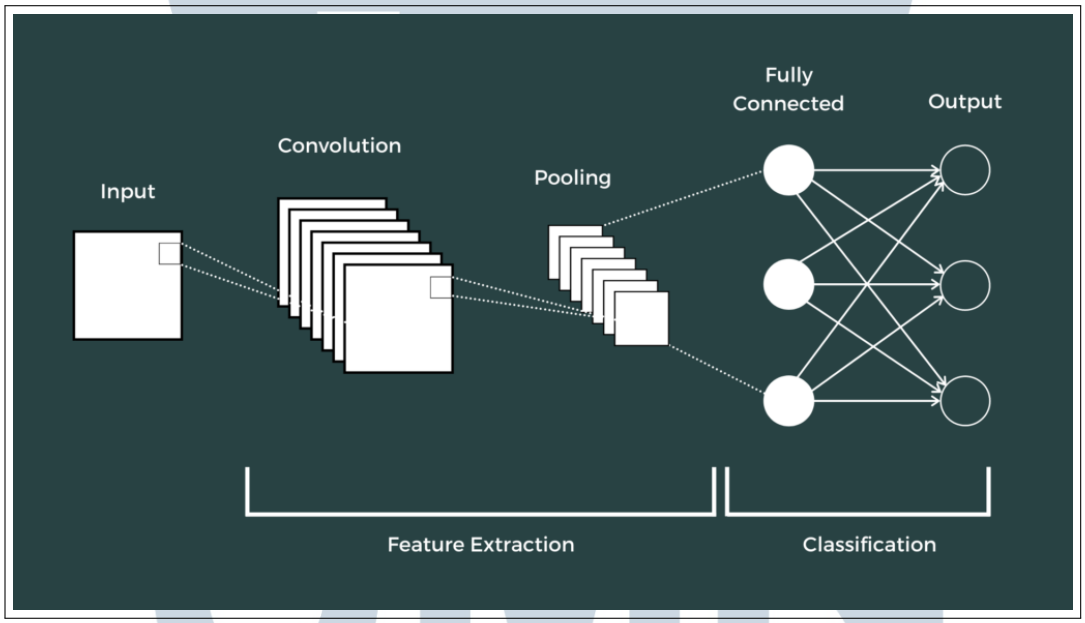
# BAB 2
# LANDASAN TEORI

## 2.1 *Convolutional Neural Network*

*Convolutional Neural Network* (CNN) merupakan implementasi dari *Artificial Neural Network* (ANN) yang lebih khusus dan dianggap sebagai metode terbaik untuk pengenalan gambar. Arsitektur CNN terdiri dari tiga lapisan, yaitu lapisan *convolution*, *pooling* dan *fully connected* [16]. Berikut adalah arsitektur CNN yang menggunakan dua lapisan konvolusi dan pooling, serta dua lapisan terhubung penuh.




Gambar 2.1. *CNN Architecture*

Sumber: [17]

Dapat dilihat pada Gambar 2.1 bahwa terdapat dua komponen utama dari metode CNN, diantaranya ekstraksi fitur dan pengklasifikasi. Ekstraksi fitur dilakukan di lapisan *convolution* dan *pooling*, sedangkan pengklasifikasi berada di lapisan *fully connected*. Fitur-fitur spesifik dari sebuah objek dapat dikenali dalam ekstraksi fitur, sementara pengklasifikasi digunakan untuk pembelajaran model dan menemukan label yang sesuai untuk setiap gambar uji.

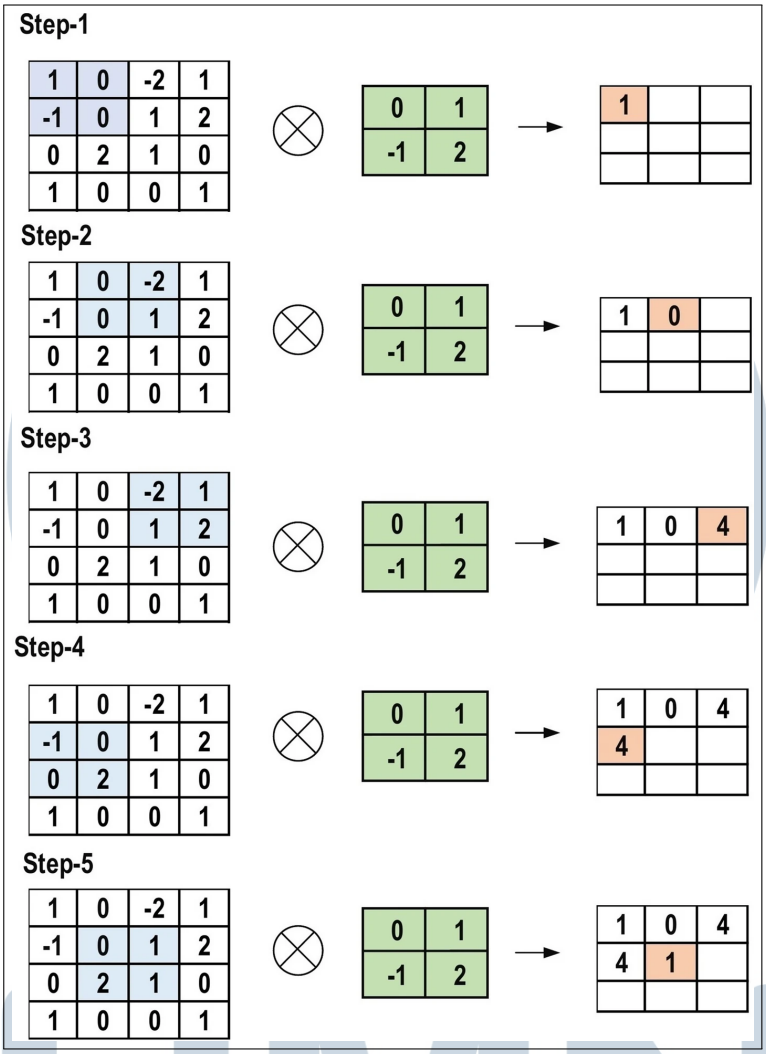
### 2.1.1 *Convolution Layer*

*Convolution Layer* merupakan bagian utama dari *Convolutional Neural Network* (CNN). Salah satu pembaruan yang diusulkan adalah dengan menambahkan *depth-wise convolution* pada *convolution layer*, di mana tiap saluran input dikonvolusi dengan pola yang berbeda dalam gambar 2 dimensi.

Gabungan kedua jenis konvolusi ini membentuk *over-parameters*, karena menambahkan parameter yang dapat dipelajari. Walaupun begitu, operasi matematis yang dihasilkan dapat disederhanakan menjadi hanya satu *convolution layer*. *Convolution Layer* baru ini disebut sebagai *DO-Conv*, dan merupakan pendekatan baru dalam penggunaan *over-parameter*.

Hasil eksperimen menunjukkan bahwa penggantian *Convolution Layer* konvensional dengan *DO-Conv* saja sudah meningkatkan kinerja CNN pada banyak tugas pengenalan gambar yang umum, seperti klasifikasi, deteksi, dan segmentasi. Selain itu, pada tahap inferensi, *depth-wise convolution* digabungkan ke dalam konvolusi konvensional, sehingga mengurangi kompleksitas komputasi menjadi sama dengan *convolution layer* biasa.

*DO-Conv* memberikan peningkatan kinerja tanpa menambah kompleksitas komputasi saat pengenalan, sehingga disarankan sebagai alternatif untuk lapisan konvolusi biasa [18].




Gambar 2.2. Proses *Convolution Layer*

Sumber: [19]

Gambar 2.2 menggambarkan secara grafis perhitungan utama yang dilakukan pada setiap langkah. Pada gambar ini, warna hijau muda mewakili kernel 2 x 2, sedangkan warna biru muda mewakili area ukuran yang sama dari gambar masukan. Keduanya dikalikan lalu hasil akhir setelah menjumlahkan nilai produk yang dihasilkan (ditandai dengan warna oranye muda) mewakili nilai entri ke peta fitur keluaran.

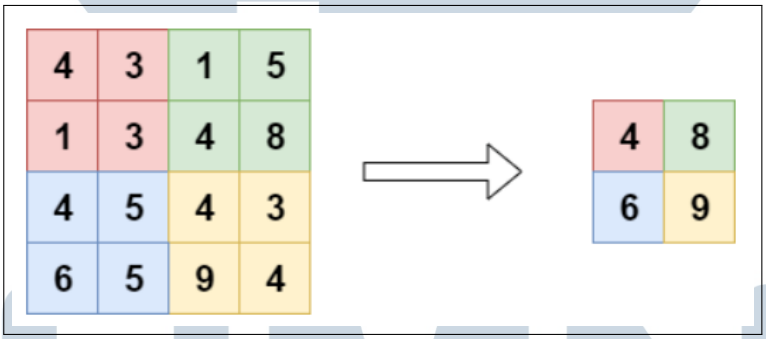
### 2.1.2 *Pooling Layer*

*Pooling Layer* merupakan elemen dalam struktur jaringan yang menggunakan fungsi yang menerima Peta Fitur (*Feature Map*) sebagai input,

kemudian melakukan beragam operasi statistik berdasarkan nilai *pixel* terdekat. Dalam model *Convolutional Neural Network* (CNN), *Pooling Layer* umumnya dimasukkan secara berkala setelah beberapa lapisan konvolusi.

Penyisipan *Pooling Layer* di antara lapisan-lapisan konvolusi secara berturut-turut dalam arsitektur CNN dapat secara progresif mengurangi dimensi *volume output* pada *Feature Map*, yang pada gilirannya mengurangi jumlah parameter dan komputasi dalam jaringan serta membantu mengatasi masalah *Overfitting*.

*Pooling Layer* bekerja pada setiap set *Feature Map* dan secara signifikan mengurangi dimensi mereka. Salah satu bentuk yang umum dari *Pooling Layer* adalah menggunakan filter berukuran 2x2 dengan langkah sebanyak 2 yang dapat dilihat pada Gambar 2.3, yang diterapkan pada setiap irisan dari input. Penggunaan bentuk seperti ini dapat mengurangi ukuran *Feature Map* hingga 75% dari ukuran aslinya [20].

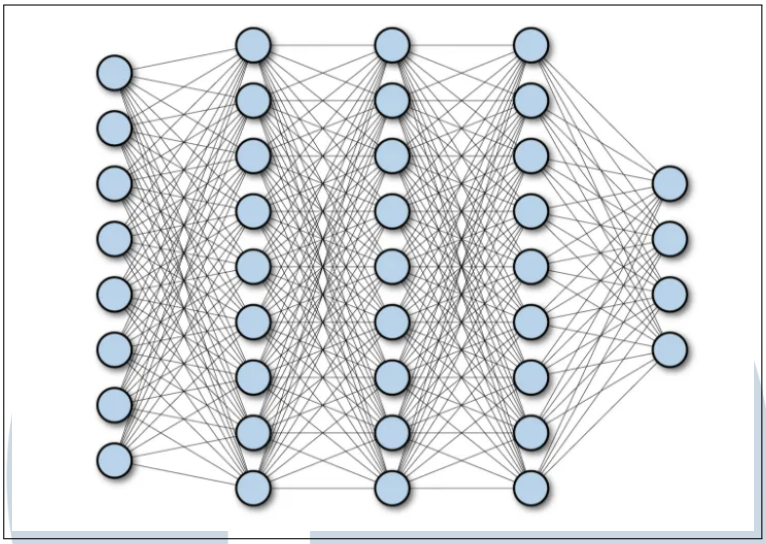



Gambar 2.3. Proses *Pooling Layer*

Sumber: [21]

### 2.1.3 *Fully Connected Layer*

*Fully Connected Layer* merupakan lapisan yang umumnya digunakan dalam *Multilayer Perceptron* dan bertujuan untuk mentransformasi setiap dimensi data sehingga data dapat diklasifikasikan secara linear. Pada Gambar 2.4 merupakan proses dari *Fully Connected Layer* yang dimana setiap neuron dalam lapisan konvolusi harus diubah menjadi data satu dimensi sebelum dimasukkan ke dalam *Fully Connected Layer*. Namun, proses ini mengakibatkan hilangnya informasi spesifik dari data yang tidak dapat dikembalikan, sehingga *Fully Connected Layer* hanya dapat diterapkan pada akhir jaringan [22].

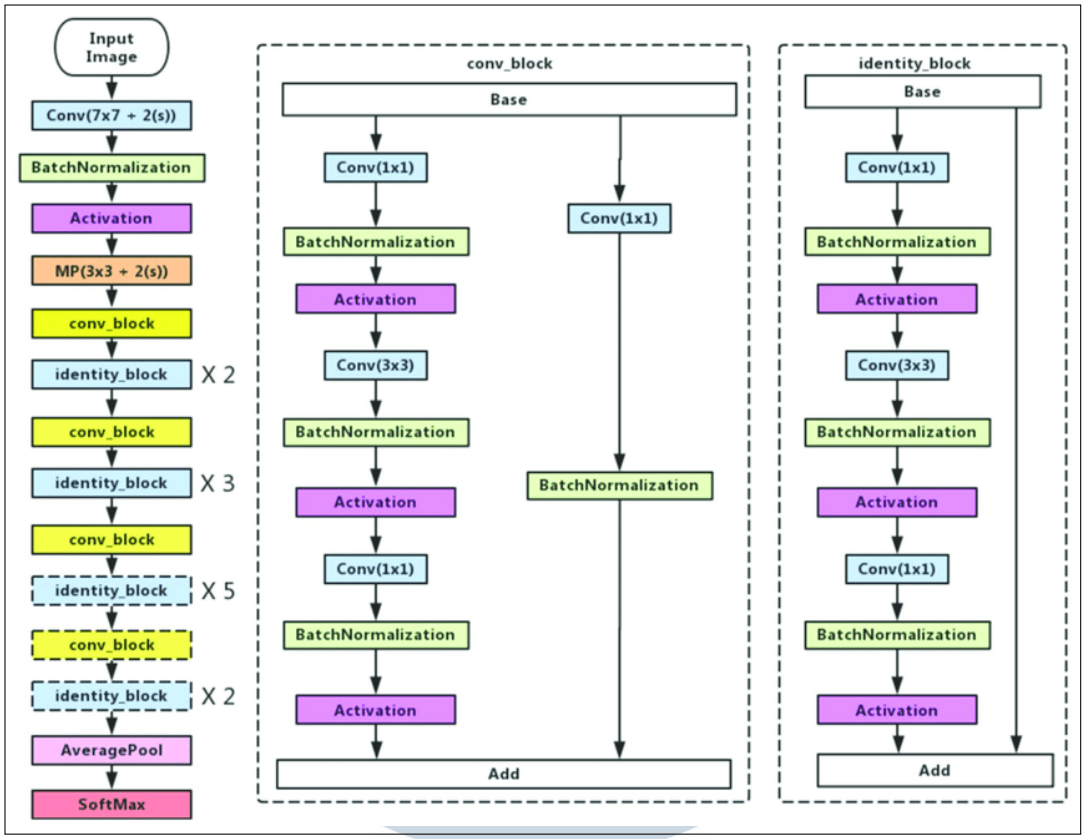


Gambar 2.4. Proses *Fully Connected Layer*

Sumber: [23]

## 2.2 *ResNet-50*

*ResNet-50* adalah salah satu struktur CNN yang mengenalkan konsep baru bernama *shortcut connections*. Penambahan konsep ini dalam arsitektur *ResNet-50* terkait erat dengan masalah *vanishing gradient* yang terjadi ketika mencoba mendalami jaringan. Membuat jaringan lebih dalam untuk meningkatkan kinerja tidak bisa hanya dengan menambahkan lapisan. Semakin dalam jaringan, masalah *vanishing gradient* dapat timbul, yang mengakibatkan gradien yang sangat kecil dan menyebabkan penurunan performa atau akurasi [24].




Gambar 2.5. Arsitektur *ResNet-50*

Sumber: [25]

Gambar 2.5 merupakan arsitektur *ResNet-50* yang terdiri dari total 50 lapisan yang terbagi ke dalam beberapa blok utama. Setiap blok terdiri dari beberapa lapisan konvolusi yang diikuti oleh *shortcut connection*. Alur utama dimulai dengan blok awal yang terdiri dari beberapa lapisan konvolusi dan operasi *max pooling* untuk mengekstrak fitur dari gambar *input*. Kemudian, terdapat empat blok utama yang masing-masing terdiri dari beberapa blok *residual*. Blok-blok ini bertujuan untuk mendalami representasi fitur secara bertahap. Pada setiap blok, *shortcut connections* memungkinkan informasi untuk "melewati" lapisan konvolusi, memperkuat aliran informasi dan mencegah penurunan tajamnya gradien. Setelah blok-blok utama, terdapat lapisan akhir yang terdiri dari *global average pooling* (GAP) untuk mereduksi dimensi spasial fitur, diikuti oleh *fully connected layer* untuk klasifikasi. Terakhir, fungsi *softmax* digunakan untuk menghasilkan distribusi probabilitas dari kelas-kelas yang mungkin. *ResNet-50* menggunakan fungsi aktivasi *ReLU* (*Rectified Linear Unit*) setelah setiap lapisan konvolusi untuk mengenonaktifkan nilai negatif [26].