Chapter 14: Deep Computer Vision dengan Convolutional Neural Networks

Pendahuluan

Convolutional Neural Networks (CNNs atau ConvNets) adalah arsitektur Jaringan Saraf Tiruan yang telah merevolusi bidang *Computer Vision*. Keberhasilan mereka berakar pada inspirasi dari cara kerja korteks visual biologis. Penelitian oleh Hubel dan Wiesel pada tahun 1950-an dan 1960-an menunjukkan bahwa banyak neuron di korteks visual memiliki *receptive field* (bidang reseptif) lokal yang kecil, artinya mereka hanya bereaksi terhadap stimulus visual di area terbatas. Mereka juga menemukan bahwa neuron-neuron ini tersusun secara hierarkis: beberapa neuron bereaksi terhadap pola sederhana seperti garis horizontal, sementara neuron di tingkat yang lebih tinggi bereaksi terhadap pola yang lebih kompleks (seperti tekstur atau bentuk) dengan menggabungkan output dari neuron di tingkat yang lebih rendah. CNN meniru arsitektur hierarkis ini untuk secara otomatis belajar mengenali pola dari yang paling sederhana hingga yang paling kompleks.

1. Blok Pembangun CNN

CNN dibangun dari beberapa jenis lapisan utama, dengan *Convolutional Layer* dan *Pooling Layer* sebagai intinya.

Convolutional Layer (Lapisan Konvolusional)

Lapisan konvolusional adalah blok pembangun paling penting dari CNN. Alih-alih menghubungkan setiap neuron ke semua neuron di lapisan sebelumnya (seperti pada *Dense layer*), neuron di lapisan konvolusional hanya terhubung ke piksel-piksel dalam bidang reseptif lokalnya.

- **Filter (atau Kernel)**: Lapisan ini bekerja dengan melewatkan satu set **filter** yang dapat dilatih di atas input. Setiap filter adalah matriks bobot kecil yang dirancang untuk mendeteksi fitur tertentu (misalnya, tepi vertikal, sudut, atau tekstur).
- **Feature Map**: Saat filter "bergeser" melintasi gambar, ia melakukan operasi konvolusi (pada dasarnya perkalian dot) antara bobotnya dan piksel di bawahnya. Hasil dari operasi ini untuk seluruh gambar disebut **feature map**. *Feature map* ini menunjukkan di mana fitur spesifik tersebut terdeteksi di dalam gambar. Sebuah lapisan konvolusional biasanya mempelajari banyak filter secara bersamaan, sehingga menghasilkan banyak *feature map*.
- **Padding**: Jika filter diterapkan tanpa *padding*, dimensi *feature map* akan lebih kecil dari gambar input, dan informasi di bagian tepi akan kurang dimanfaatkan. Untuk mengatasinya, **zero-padding** sering ditambahkan di sekitar batas input. "Valid" padding berarti tidak ada padding, sedangkan "Same" padding berarti menambahkan padding yang cukup agar dimensi output sama dengan dimensi input.
- **Stride**: *Stride* adalah jumlah piksel yang digeser oleh filter pada setiap langkah. *Stride* yang lebih besar akan menghasilkan *feature map* dengan dimensi yang lebih kecil.

Rumus Ukuran Output Konvolusional:

Ukuran output (tinggi atau lebar) dari sebuah lapisan konvolusional dapat dihitung dengan rumus:

Output_size = floor((Input_size + 2 * Padding – Filter_size) / Stride) + 1

Pooling Layer (Lapisan Pooling)

Tujuan utama dari lapisan *pooling* adalah untuk melakukan **subsampling** (yaitu, mengecilkan) representasi spasial dari *feature map*. Manfaatnya adalah:

1. Mengurangi jumlah parameter dan beban komputasi.
2. Mengurangi risiko *overfitting*.
3. Memberikan **invariance translasi** tingkat dasar, artinya model menjadi sedikit lebih toleran terhadap posisi fitur di dalam gambar.

- **Max Pooling**: Jenis *pooling* yang paling umum. Sebuah jendela (misalnya, 2x2) bergeser melintasi *feature map*, dan pada setiap posisi, ia hanya mengeluarkan nilai maksimum dari jendela tersebut.
- **Average Pooling**: Bekerja dengan cara yang sama, tetapi mengeluarkan nilai rata-rata alih-alih nilai maksimum.

2. Arsitektur CNN

Arsitektur CNN yang khas terdiri dari penumpukan beberapa blok, di mana setiap blok terdiri dari:

1. Satu atau lebih **Convolutional Layer**.
2. Sebuah **Fungsi Aktivasi** (biasanya ReLU).
3. Sebuah **Pooling Layer**.

Saat data mengalir lebih dalam ke jaringan, *feature map* biasanya menjadi lebih kecil secara spasial (karena *pooling*) tetapi lebih dalam secara jumlah (karena setiap lapisan konvolusional menambahkan lebih banyak filter). Hierarki ini memungkinkan jaringan untuk belajar dari fitur visual sederhana (seperti tepi di lapisan awal) hingga fitur yang sangat kompleks (seperti wajah atau objek di lapisan akhir). Setelah beberapa blok konvolusional/pooling, *feature map* akhir diratakan (*flattened*) menjadi vektor 1D dan dimasukkan ke beberapa *Dense layer* untuk melakukan tugas akhir seperti klasifikasi.

3. Evolusi Arsitektur CNN Terkenal

Sejarah *Computer Vision* modern ditandai oleh beberapa arsitektur CNN yang menjadi tonggak penting.

- **LeNet-5 (1998)**: Salah satu CNN pertama yang berhasil, dirancang oleh Yann LeCun untuk pengenalan tulisan tangan angka. Arsitekturnya sangat sederhana: Conv → Pool → Conv → Pool → Dense → Dense.
- **AlexNet (2012)**: Arsitektur yang memenangkan kompetisi ImageNet 2012 dan memicu revolusi Deep Learning. AlexNet jauh lebih besar dan lebih dalam dari LeNet-5 dan

merupakan yang pertama secara ekstensif menggunakan aktivasi ReLU dan teknik regularisasi *dropout*.

- **GoogLeNet (2014)**: Memperkenalkan **Inception Module**. Alih-alih memilih satu ukuran filter atau *pooling* pada satu lapisan, modul Inception melakukan beberapa konvolusi (1x1, 3x3, 5x5) dan *max pooling* secara paralel, lalu menggabungkan hasilnya. Ini memungkinkan jaringan untuk menangkap pola pada skala yang berbeda secara bersamaan. Arsitektur ini juga secara cerdas menggunakan konvolusi 1x1 untuk mengurangi dimensi sebelum konvolusi yang lebih mahal.
- **ResNet (Residual Network, 2015)**: Membuat terobosan yang memungkinkan pelatihan jaringan yang sangat dalam (bahkan lebih dari 100 lapisan) secara efektif.
    - **Masalah**: Jaringan yang sangat dalam sering mengalami *degradation problem*, di mana performa pelatihan justru menurun seiring bertambahnya lapisan.
    - **Solusi**: **Residual Connection** atau **Skip Connection**. Ide intinya adalah dengan menambahkan input dari sebuah blok lapisan ke output dari blok tersebut. Koneksi "jalan pintas" ini memungkinkan gradien untuk mengalir langsung melalui jaringan selama *backpropagation*, sehingga mengatasi masalah *vanishing gradient* dan membuat pelatihan menjadi lebih mudah. Jaringan pada dasarnya dilatih untuk mempelajari fungsi *residual* $F(x) = H(x) - x$, di mana $H(x)$ adalah pemetaan yang diinginkan. Lebih mudah bagi sebuah blok untuk belajar mendekati nol daripada mempelajari pemetaan identitas.

4. Tugas Computer Vision Lanjutan

Selain klasifikasi gambar, arsitektur berbasis CNN juga digunakan untuk tugas yang lebih kompleks:

- **Object Detection**: Tugas untuk menemukan lokasi (melalui *bounding box*) dan mengklasifikasikan satu atau lebih objek dalam sebuah gambar. Model populer termasuk YOLO (You Only Look Once) dan SSD.
- **Semantic Segmentation**: Tugas untuk mengklasifikasikan setiap piksel dalam sebuah gambar ke dalam kelas tertentu (misalnya, menandai semua piksel yang merupakan "jalan", "mobil", atau "pohon"). Arsitektur terkenal termasuk Fully Convolutional Networks (FCN) dan U-Net.