Bab 10 — Deep Learning Praktis dan Arsitektur Besar

Cara membaca bab ini

Bab 10 adalah jembatan dari neural network dasar di Bab 9 menuju keluarga model modern yang dipakai di visi komputer, teks, suara, rekomendasi, deteksi objek, dan generative AI. Bab ini tetap AI-first: kita tidak memperluas terlalu jauh ke robotics atau produksi MLOps, tetapi kita cukup dalam untuk memahami mengapa arsitektur seperti CNN, LeNet, AlexNet, VGG, ResNet, YOLO, RNN, LSTM, Transformer, DBN, dan BNN menjadi tonggak penting.

Pola belajar setiap subbab:

> cerita masalah → bentuk data → operasi inti → persamaan → contoh hitung → praktik kecil → tes cepat

Jika sebuah rumus terasa menakutkan, baca perlahan dari kiri ke kanan. Di bab ini rumus bukan hiasan, tetapi alat untuk menjawab pertanyaan praktis: berapa ukuran output, berapa banyak parameter, mengapa model sulit dilatih, mengapa GPU membantu, mengapa residual connection menyelamatkan jaringan sangat dalam, dan mengapa attention dapat menggantikan recurrence pada banyak tugas urutan.

Batas cakupan: bab ini memberi fondasi deep learning yang luas dan cukup dalam. Pembahasan full computer vision, full NLP, diffusion model, deployment edge, dan produksi skala besar ditunda ke konteks bab berikutnya atau backlog edisi lanjutan.

Peta keluarga deep learning

Subbab 1 — Dari neural network ke deep learning

Inti subbab: deep learning adalah neural network berlapis banyak yang belajar representasi bertingkat dari data.

Di Bab 9 kita melihat neuron, aktivasi, loss, dan backpropagation. Deep learning memakai bahan yang sama, tetapi mengatur layer menjadi arsitektur khusus. Pada gambar, layer awal biasanya membaca pola sederhana, layer tengah menyusun pola, dan layer akhir membuat keputusan.

Contoh pada gambar produk UMKM:

> piksel → tepi → tekstur → bagian objek → kategori produk

Pada teks ulasan pelanggan:

> token → frasa → konteks kalimat → sentimen/niat → keputusan bisnis

Pada sinyal penjualan mingguan:

> angka waktu → pola naik-turun → musim → anomali → prediksi stok

Deep learning disebut “deep” bukan karena selalu misterius, tetapi karena komposisi fungsi menjadi panjang:

```
h1 = f1(x)
h2 = f2(h1)
h3 = f3(h2)
y_hat = f4(h3)
```

Cara membaca persamaan: `h1`, `h2`, dan `h3` adalah representasi perantara. Setiap `f` adalah layer atau blok. Model tidak hanya mencari jawaban akhir, tetapi juga belajar cara mengubah data mentah menjadi bentuk yang makin berguna.

Mengapa ini penting? Karena fitur manual sering mahal. Dulu sistem visi mungkin memakai fitur buatan manusia seperti SIFT/HOG. Deep learning belajar fitur itu dari data, asalkan data, komputasi, arsitektur, dan prosedur training cukup baik. AlexNet menjadi simbol kebangkitan deep learning karena menunjukkan CNN besar dapat mengalahkan metode fitur klasik di ImageNet dengan selisih besar [R2].

Contoh hitung mini: misalkan model punya tiga layer linear kecil:

```
x = [2, 1]
h1 = ReLU(W1x + b1)
h2 = ReLU(W2h1 + b2)
y = W3h2 + b3
```

Jika layer pertama belajar “kombinasi warna”, layer kedua belajar “motif”, layer ketiga belajar “kelas produk”, maka representasi tidak perlu dirancang manual satu per satu.

Tes cepat subbab 1

1. Apa perbedaan neural network biasa dan deep learning secara praktis?

2. Mengapa deep learning sering membutuhkan data dan komputasi lebih besar?

3. Berikan contoh representasi bertingkat untuk data audio pendek.

Subbab 2 — Operasi dasar deep learning: tensor, batch, dan shape

Inti subbab: sebelum memahami arsitektur, pembaca harus nyaman membaca bentuk tensor.

Deep learning hampir selalu bekerja dengan tensor. Tensor dapat dibayangkan sebagai wadah angka berdimensi banyak.

Jenis data	Contoh shape	Makna
Vektor fitur tabular	`(batch, fitur)`	banyak baris, banyak kolom fitur
Gambar grayscale	`(batch, tinggi, lebar, 1)`	satu kanal intensitas
Gambar RGB	`(batch, tinggi, lebar, 3)`	kanal merah, hijau, biru
Teks/token	`(batch, panjang_urutan)`	urutan indeks kata/token
Embedding teks	`(batch, panjang_urutan, dim_embedding)`	setiap token menjadi vektor
Video pendek	`(batch, waktu, tinggi, lebar, kanal)`	urutan frame

Cara membaca: `batch` berarti beberapa contoh diproses bersamaan. Jika shape gambar adalah `(32, 64, 64, 3)`, artinya ada 32 gambar, masing-masing 64×64 piksel, dengan 3 kanal warna.

Mini-batch membuat training lebih efisien. Loss rata-rata mini-batch sering ditulis:

```
L_batch = (1/m) × Σ_i L(y_i, y_hat_i)
```

`m` adalah jumlah contoh dalam mini-batch. Jika empat contoh punya loss `[0,2; 0,6; 0,4; 0,8]`, maka:

```
L_batch = (0,2 + 0,6 + 0,4 + 0,8) / 4 = 0,5
```

Mengapa shape penting? Banyak error deep learning bukan karena rumus salah, tetapi karena shape tidak cocok. Contoh: CNN menghasilkan tensor `(batch, 7, 7, 512)`, lalu fully connected mengharapkan vektor. Maka tensor perlu di-flatten atau di-average pooling.

Aturan praktis shape:

```
setiap layer menjawab dua pertanyaan:
1. apa bentuk input?
2. apa bentuk output?
```

Di lab bab ini, pembaca akan menjalankan kalkulator shape agar terbiasa membaca layer seperti membaca resep masakan.

Tes cepat subbab 2

1. Apa arti shape `(16, 224, 224, 3)`?

2. Mengapa mini-batch membantu GPU?

3. Jika 8 contoh punya loss rata-rata 0,75, apa maknanya terhadap training?

Subbab 3 — Convolution: jendela kecil yang menyapu gambar

Inti subbab: convolution memakai filter kecil yang digeser di atas input untuk mendeteksi pola lokal.

CNN lahir dari ide bahwa gambar punya struktur spasial. Piksel dekat biasanya lebih saling terkait daripada piksel jauh. Karena itu, alih-alih menghubungkan semua piksel ke semua neuron, CNN memakai filter kecil.

Misalkan input 4×4:

```
X = [[1, 2, 0, 1],
     [3, 1, 2, 2],
     [0, 1, 3, 1],
     [2, 2, 1, 0]]
```

Filter 2×2:

```
K = [[1, 0],
     [0,-1]]
```

Output posisi kiri atas dihitung dari patch:

```
patch = [[1,2],
         [3,1]]

nilai = 1×1 + 2×0 + 3×0 + 1×(-1) = 0
```

Posisi berikutnya bergeser satu kolom:

```
patch = [[2,0],
         [1,2]]

nilai = 2×1 + 0×0 + 1×0 + 2×(-1) = 0
```

Persamaan umum convolution 2D untuk satu kanal:

$$Y[i,j] = \Sigma_u \Sigma_v K[u,v] \times X[i+u, j+v]$$

Cara membaca persamaan: untuk setiap lokasi output `(i, j)`, ambil potongan input, kalikan elemen demi elemen dengan kernel `K`, lalu jumlahkan.

Ukuran output tanpa padding:

$$out = floor((in - kernel + 2 \times padding) / stride) + 1$$

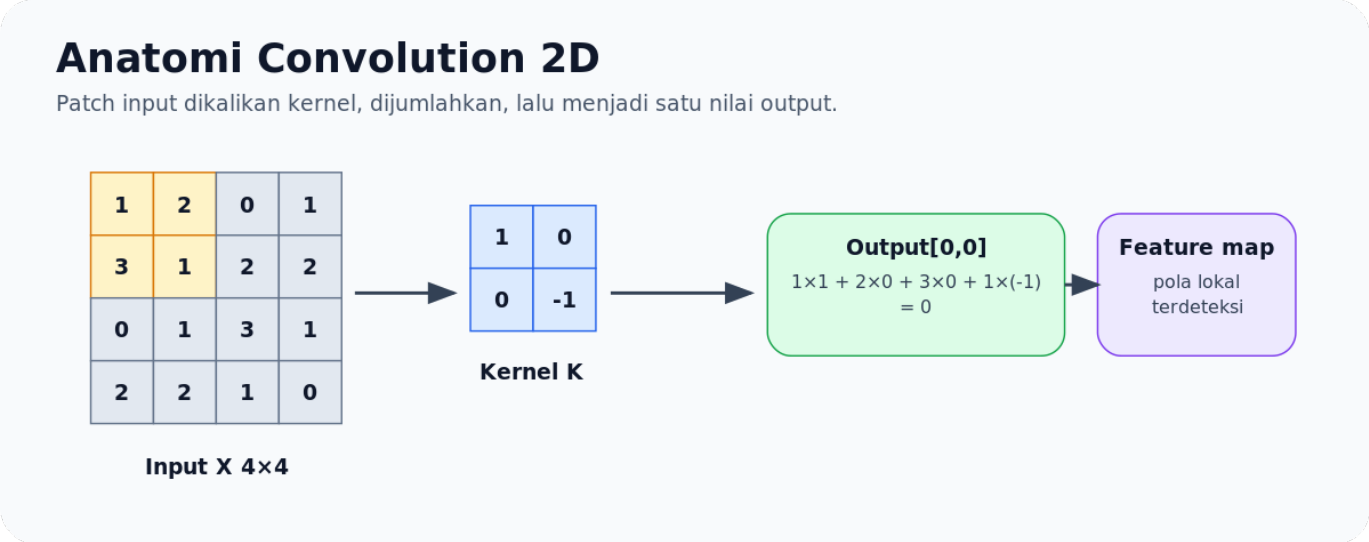
Contoh: input 32, kernel 5, padding 0, stride 1:

$$out = floor((32 - 5 + 0) / 1) + 1 = 28$$

Jika padding 2:

$$out = floor((32 - 5 + 4) / 1) + 1 = 32$$

Padding menjaga ukuran spasial. Stride memperkecil ukuran. Pooling merangkum wilayah kecil, misalnya max pooling 2×2 mengambil nilai terbesar dari setiap blok.



Anatomi convolution

Mengapa CNN hemat parameter? Jika input 224×224×3 langsung masuk fully connected 100 neuron, jumlah bobot:

$$224 \times 224 \times 3 \times 100 = 15.052.800 \text{ bobot}$$

Satu layer convolution 3×3 dengan 64 filter pada input 3 kanal hanya punya:

$$3 \times 3 \times 3 \times 64 = 1.728 \text{ bobot}$$

Perbedaannya sangat besar. CNN hemat karena filter yang sama dipakai berulang di banyak lokasi. Ini disebut weight sharing.

Tes cepat subbab 3

1. Hitung ukuran output untuk input 28, kernel 3, padding 1, stride 1.
2. Mengapa convolution lebih hemat daripada fully connected pada gambar?
3. Apa beda padding dan stride?

Subbab 4 — LeNet: CNN klasik untuk tulisan tangan

Inti subbab: LeNet menunjukkan bahwa convolution, pooling, dan backpropagation dapat bekerja untuk pengenalan digit/dokumen.

LeNet-5 dari LeCun dan kolega menjadi arsitektur klasik untuk mengenali digit dan karakter pada dokumen [R1]. Dalam konteks sejarah, LeNet penting karena memperlihatkan pola yang masih kita pakai sampai sekarang:

> convolution → pooling/subsampling → convolution → pooling → dense → output

Untuk pembaca Indonesia, bayangkan sistem membaca angka pada kode pos, nomor formulir, atau digit meter listrik. Tantangannya: bentuk angka bervariasi, posisi sedikit bergeser, dan tulisan orang berbeda-beda. CNN cocok karena filter lokal dapat mendeteksi garis, sudut, dan lengkungan tanpa harus mengetahui posisi persisnya.

Contoh alur shape sederhana untuk input 32×32 grayscale:

```
Input           : 32×32×1
Conv 5×5, 6 filter: 28×28×6
Pooling 2×2       : 14×14×6
Conv 5×5,16 filter: 10×10×16
Pooling 2×2       : 5×5×16
Flatten          : 400
Dense            : kelas digit
```

Jumlah parameter convolution pertama:

> 5×5×1×6 + 6 bias = 156 parameter

Jumlah parameter convolution kedua jika semua kanal terhubung penuh:

> 5×5×6×16 + 16 bias = 2.416 parameter

Cara membaca: kernel 5×5 melihat area kecil. Enam filter berarti model belajar enam jenis pola awal. Setelah pooling, ukuran lebih kecil sehingga layer berikutnya melihat pola yang lebih luas secara relatif.

Catatan kritis: LeNet tidak besar menurut standar modern. Tetapi ide arsitekturnya sangat tahan lama. CNN modern seperti AlexNet, VGG, ResNet, dan MobileNet masih memakai prinsip lokalitas, weight sharing, downsampling, dan hierarki fitur.

Tes cepat subbab 4

1. Mengapa LeNet cocok untuk digit/tulisan tangan?
2. Hitung parameter Conv 3×3 dengan 8 filter pada input 1 kanal.
3. Apa fungsi pooling dalam LeNet?

Subbab 5 — AlexNet: momen “deep learning bangkit”

Inti subbab: AlexNet menunjukkan bahwa CNN besar, GPU, ReLU, dropout, dan data besar dapat mengalahkan pendekatan visi klasik.

AlexNet memenangkan ILSVRC-2012 dengan top-5 error 15,3%, jauh lebih baik daripada pesaing berbasis fitur klasik yang sekitar 26,2% [R2]. Model ini dilatih pada sekitar 1,2 juta gambar ImageNet dengan 1000 kelas. Arsitekturnya memiliki lima convolutional layer dan tiga fully connected layer, sekitar 60 juta parameter, memakai ReLU, data augmentation, GPU, dan dropout.

Mengapa AlexNet penting?

1. Skala data: ImageNet membuat supervised learning pada gambar menjadi kompetisi besar.
2. Skala model: CNN cukup besar untuk belajar representasi kompleks.
3. GPU: operasi matriks/convolution dipercepat.
4. ReLU: mengurangi masalah saturasi dibanding sigmoid/tanh.
5. Dropout: mengurangi overfitting pada layer besar [R10].

Dropout secara sederhana:

```
r_j ~ Bernoulli(p)
h_drop = r ⊙ h
```

Cara membaca: `r_j` adalah saklar acak. Jika bernilai 0, unit dimatikan sementara. Jika bernilai 1, unit aktif. Simbol ■ berarti perkalian elemen demi elemen.

Contoh: aktivasi `h=[2, 5, 1, 4]`, mask dropout `r=[1, 0, 1, 0]`:

```
h_drop = [2, 0, 1, 0]
```

Dropout memaksa neuron tidak terlalu bergantung pada neuron tertentu. Analogi: tim kerja yang kadang anggotanya tidak hadir akan belajar membuat prosedur yang lebih robust, bukan mengandalkan satu orang saja.

Catatan kritis: AlexNet bukan hanya “lebih besar”. Ia adalah kombinasi arsitektur, data, komputasi, dan trik training. Meniru AlexNet tanpa data cukup atau tanpa regularisasi bisa berujung overfitting.

Timeline arsitektur CNN

Tes cepat subbab 5

1. Sebutkan tiga faktor yang membuat AlexNet sukses.

2. Apa tujuan dropout?

3. Jika `p=0,5`, kira-kira berapa unit aktif dari 1000 unit saat training?

Subbab 6 — VGG: kedalaman dari blok 3×3 sederhana

Inti subbab: VGG menunjukkan bahwa arsitektur sederhana dan sangat dalam dengan filter 3×3 dapat menghasilkan representasi kuat.

VGG dari Simonyan dan Zisserman terkenal karena prinsipnya: gunakan banyak convolution 3×3 bertumpuk [R3]. VGG-16 dan VGG-19 merujuk pada jumlah weight layer. Dibanding memakai kernel besar, VGG menyusun kernel kecil beberapa kali.

Mengapa dua convolution 3×3 bisa menggantikan satu area pandang 5×5? Jika dua layer 3×3 ditumpuk, neuron layer kedua melihat 3×3 dari feature map sebelumnya, dan setiap titik di feature map sebelumnya melihat 3×3 dari input. Efektifnya area pandang menjadi 5×5.

Perbandingan parameter untuk input/output kanal sama `C=64`:

```
Satu conv 5×5 : 5×5×64×64 = 102.400 parameter
Dua conv 3×3  : 2×3×3×64×64 = 73.728 parameter
```

Dua conv 3×3 lebih hemat dan menambah satu ReLU ekstra, sehingga model lebih non-linear.

Cara membaca: VGG seperti menyusun banyak kaca pembesar kecil. Setiap kaca sederhana, tetapi susunannya membuat model melihat pola makin luas dan kompleks.

Kelemahan VGG adalah berat. VGG-16 memiliki sekitar 138 juta parameter, terutama karena fully connected layer besar. Ini membuatnya bagus sebagai tonggak sejarah dan feature extractor, tetapi tidak selalu cocok untuk perangkat terbatas.

Kapan ide VGG masih berguna?

• Saat mengajarkan CNN karena bloknya mudah dibaca.

• Saat membuat baseline yang sederhana.

• Saat memahami mengapa kernel kecil bertumpuk bisa kuat.

• Saat membandingkan trade-off akurasi vs ukuran model.

Tes cepat subbab 6

1. Mengapa VGG memakai banyak kernel 3×3?

2. Hitung parameter satu conv 5×5 untuk `C_in=32`, `C_out=64`.

3. Apa kelemahan utama VGG untuk deployment ringan?

Subbab 7 — ResNet: jalan pintas untuk jaringan sangat dalam

Inti subbab: ResNet memakai residual connection agar jaringan sangat dalam lebih mudah dioptimalkan.

Ketika jaringan makin dalam, masalah tidak hanya overfitting. Kadang training error justru memburuk karena optimisasi sulit. ResNet memperkenalkan residual learning [R4]. Alih-alih memaksa blok belajar fungsi `H(x)` langsung, blok belajar residual `F(x)`:

```
H(x) = F(x) + x
```

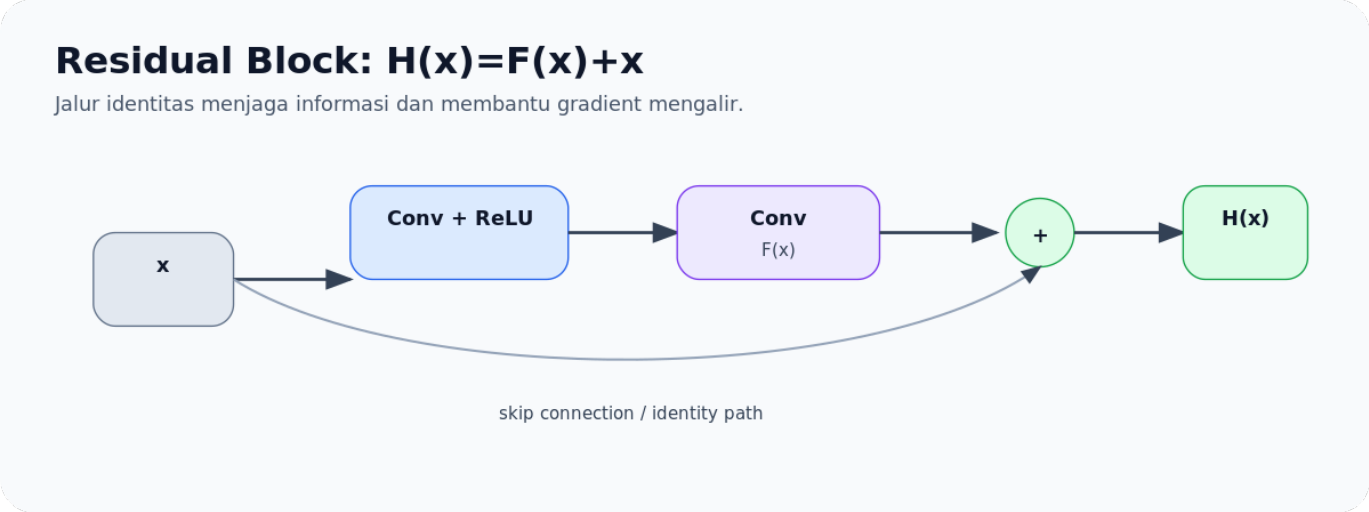
Cara membaca: output blok adalah hasil transformasi `F(x)` ditambah input asli `x`. Jika transformasi belum berguna, blok bisa belajar `F(x)≈0`, sehingga output mendekati `x`. Ini seperti memberi jalur aman agar sinyal dan gradient tidak hilang.

Contoh hitung kecil:

```
x    = [2, -1, 3]
F(x) = [0,5, 0,2, -1]
H(x) = [2,5, -0,8, 2]
```

Tanpa skip connection, output hanya `[0,5, 0,2, -1]`. Dengan skip, informasi input tetap ikut mengalir.

ResNet memungkinkan training jaringan sangat dalam, bahkan sampai 152 layer pada eksperimen ImageNet. Paper ResNet melaporkan ensemble dengan error ImageNet 3,57% dan kemenangan pada ILSVRC/COCO 2015 [R4].



Residual block

Analogi gradient: bayangkan kabar penting dikirim melalui banyak pos. Tanpa jalan pintas, setiap pos bisa melemahkan pesan. Dengan residual path, ada jalur langsung yang menjaga pesan tetap sampai.

Secara backpropagation, jika:

$$y = F(x) + x$$

maka turunan terhadap x memiliki komponen:

$$dy/dx = dF/dx + 1$$

Angka 1 dari jalur identitas membantu gradient tetap mengalir meskipun dF/dx kecil.

Tes cepat subbab 7

1. Apa bedanya $H(x)$ dan $F(x)$ dalam ResNet?

2. Mengapa jalur $+ \ x$ membantu gradient?

3. Hitung $H(x)$ jika $x=[1,2]$ dan $F(x)=[-0,2,0,5]$.

Subbab 8 — YOLO: deteksi objek sebagai prediksi sekali jalan

Inti subbab: YOLO mengubah deteksi objek menjadi regresi langsung dari gambar ke bounding box dan kelas.

Klasifikasi menjawab: “gambar ini kelas apa?” Deteksi objek menjawab: “objek apa saja dan di mana posisinya?” YOLO, singkatan dari *You Only Look Once*, memandang deteksi sebagai satu model terpadu yang memprediksi bounding box dan probabilitas kelas dari seluruh gambar dalam satu forward pass [R5].

Ide sederhana YOLO:

gambar → grid S×S → setiap sel memprediksi box + confidence + kelas

Bounding box sering direpresentasikan sebagai:

(x, y, w, h, confidence)

x, y adalah pusat box relatif terhadap sel/gambar, w, h adalah lebar dan tinggi, $confidence$ mengukur keyakinan bahwa box berisi objek dan lokasinya baik.

YOLO grid

Contoh hitung IoU sederhana: misalkan box A dan B punya luas masing-masing 20 dan 30. Luas irisan 10. Maka luas gabungan:

```
union = 20 + 30 - 10 = 40
IoU = intersection / union = 10 / 40 = 0,25
```

Cara membaca IoU: semakin dekat ke 1, box makin tumpang tindih. Jika IoU 0, tidak overlap.

YOLO terkenal cepat karena tidak memisahkan proposal region dan klasifikasi seperti pipeline lama. Namun versi awal YOLO juga punya kelemahan: objek kecil dan objek berdekatan bisa sulit karena setiap sel grid punya tanggung jawab terbatas. Versi-versi baru YOLO memperbaiki banyak hal, tetapi bab ini fokus pada ide fondasionalnya.

Catatan adaptasi Indonesia: deteksi objek relevan untuk inspeksi produk UMKM, hitung kendaraan di CCTV, sortir komoditas pertanian, membaca kerusakan jalan, atau audit stok gudang. Tetapi dataset lokal, variasi cahaya, privasi, dan bias lokasi harus diperhatikan.

Tes cepat subbab 8

1. Apa perbedaan klasifikasi dan deteksi objek?

2. Hitung IoU jika luas A=50, luas B=70, irisan=20.

3. Mengapa YOLO bisa cepat?

Subbab 9 — Efficient CNN: Inception, MobileNet, U-Net, dan trade-off nyata

Inti subbab: tidak semua model harus makin besar; banyak arsitektur modern mencari keseimbangan akurasi, ukuran, dan latency.

Selain LeNet, AlexNet, VGG, ResNet, dan YOLO, ada beberapa ide penting yang membantu pembaca membaca peta CNN modern.

Inception/GoogLeNet: memakai beberapa ukuran filter paralel dalam satu blok agar model menangkap pola multi-skala. Ide praktisnya: objek bisa terlihat kecil atau besar, maka model perlu jalur berbeda.

U-Net: memakai jalur contracting dan expanding untuk segmentasi, sangat populer di biomedis [R13]. Contracting path menangkap konteks; expanding path mengembalikan resolusi; skip connection menggabungkan detail lokal dengan konteks global.

MobileNet: memakai depthwise separable convolution untuk model ringan [R12]. Convolution standar memfilter dan mencampur kanal sekaligus. MobileNet memisahkan:

1. depthwise conv: filter per kanal
2. pointwise 1×1 conv: mencampur kanal

Biaya convolution standar:

$$D_K \times D_K \times M \times N \times D_F \times D_F$$

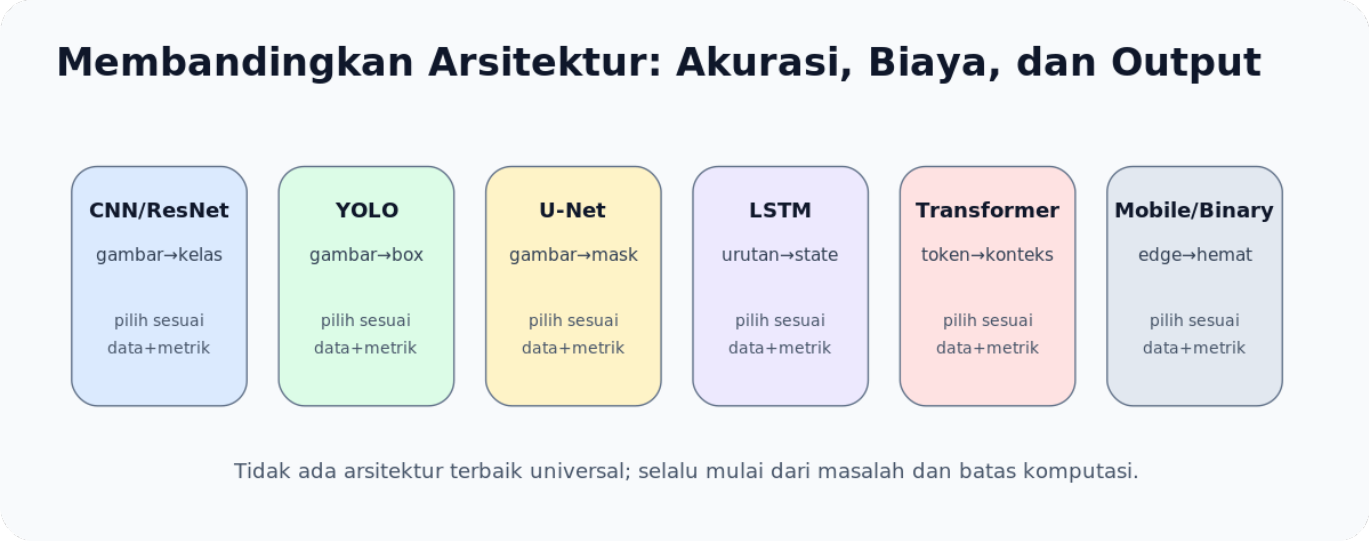
Biaya depthwise separable:

$$D_K \times D_K \times M \times D_F \times D_F + M \times N \times D_F \times D_F$$

Contoh $D_K=3$, $M=32$, $N=64$, $D_F=112$:

standar = 3×3×32×64×112×112 = 231.211.008 operasi
separable = 3×3×32×112×112 + 32×64×112×112
= 3.612.672 + 25.690.112
= 29.302.784 operasi
rasio ≈ 7,89× lebih hemat

Cara membaca: MobileNet bukan sihir. Ia mengurangi perkalian mahal dengan memisahkan "mencari pola dalam kanal" dan "mencampur kanal". Paper MobileNet menyebut convolution 3×3 depthwise separable dapat memakai sekitar 8–9 kali lebih sedikit komputasi daripada convolution standar, dengan penurunan akurasi kecil pada banyak setting [R12].



Perbandingan arsitektur

Tes cepat subbab 9

1. Apa ide utama U-Net?

2. Mengapa MobileNet cocok untuk perangkat terbatas?

3. Hitung biaya standar conv untuk $D_K=3$, $M=16$, $N=32$, $D_F=32$.

Subbab 10 — RNN: neural network untuk urutan

Inti subbab: RNN memproses data berurutan dengan state yang dibawa dari langkah sebelumnya.

Tidak semua data berupa gambar tetap. Teks, suara, transaksi, cuaca, dan harga harian adalah urutan. RNN memakai hidden state:

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b)$$

```
y_t = W_y h_t + c
```

Cara membaca: `x_t` adalah input pada waktu ke-`t`; `h_{t-1}` adalah memori dari langkah sebelumnya; `h_t` adalah memori baru; `y_t` adalah output.

Contoh cerita: prediksi permintaan es teh di warung. Input harian bisa berisi suhu, hari libur, dan promo. Hidden state menyimpan pola beberapa hari terakhir.

Contoh hitung satu dimensi:

```
x_t = 2
h_{t-1} = 0,5
W_x = 0,4
W_h = 0,6
b = -0,1

z = 0,4×2 + 0,6×0,5 - 0,1 = 1,0
h_t = tanh(1,0) ≈ 0,762
```

RNN klasik punya masalah vanishing/exploding gradient. Jika gradient dikalikan matriks berulang kali, nilainya bisa mengecil drastis atau membesar tidak stabil.

Jika faktor gradient rata-rata 0,8 selama 20 langkah:

```
0,8^20 ≈ 0,0115
```

Sinyal dari masa lalu menjadi sangat kecil. Jika faktor 1,2:

```
1,2^20 ≈ 38,34
```

Sinyal bisa meledak.

Tes cepat subbab 10

1. Apa fungsi hidden state pada RNN?

2. Mengapa RNN klasik sulit mengingat dependensi panjang?

3. Hitung `0,9^10` kira-kira dan jelaskan maknanya untuk gradient.

Subbab 11 — LSTM: gerbang untuk memori panjang

Inti subbab: LSTM memakai cell state dan gate untuk mengatur apa yang dilupakan, disimpan, dan dikeluarkan.

Long Short-Term Memory diperkenalkan untuk mengatasi masalah error/gradient yang menghilang pada urutan panjang [R6]. LSTM memiliki jalur memori yang disebut cell state `c_t` dan beberapa gate.

Rumus umum LSTM:

```
f_t = sigmoid(W_f [h_{t-1}, x_t] + b_f)      forget gate
i_t = sigmoid(W_i [h_{t-1}, x_t] + b_i)      input gate
g_t = tanh(W_g [h_{t-1}, x_t] + b_g)         candidate memory
c_t = f_t ⊙ c_{t-1} + i_t ⊙ g_t              new cell state
o_t = sigmoid(W_o [h_{t-1}, x_t] + b_o)      output gate
h_t = o_t ⊙ tanh(c_t)                        hidden state
```

Cara membaca:

- `f_t` memutuskan berapa banyak memori lama dipertahankan.
- `i_t` memutuskan berapa banyak memori baru ditulis.
- `g_t` adalah kandidat isi memori baru.
- `o_t` memutuskan berapa banyak memori terlihat sebagai output.

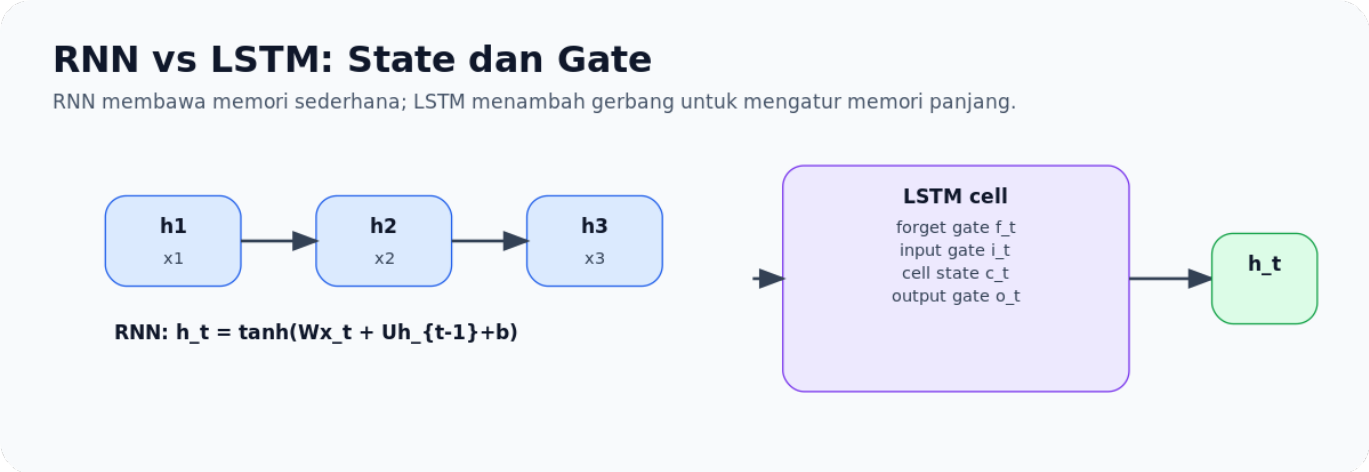
Contoh hitung satu dimensi:

```
c_{t-1} = 2
f_t = 0,8
i_t = 0,3
g_t = 0,5

c_t = 0,8×2 + 0,3×0,5 = 1,6 + 0,15 = 1,75
```

Artinya memori lama masih dominan, tetapi ada sedikit informasi baru. Inilah kekuatan LSTM: tidak setiap input harus menimpa memori.

Paper LSTM melaporkan bahwa arsitektur ini dapat menjembatani time lag lebih dari 1000 langkah diskrit pada beberapa eksperimen dan memiliki kompleksitas per time step dan bobot `O(1)` [R6].



RNN dan LSTM

Tes cepat subbab 11

1. Apa fungsi forget gate?

2. Hitung `c_t` jika `c_{t-1}=4`, `f=0,25`, `i=0,5`, `g=2`.

3. Mengapa LSTM lebih cocok daripada RNN klasik untuk dependensi panjang?

Subbab 12 — Transformer: attention menggantikan recurrence

Inti subbab: Transformer memakai self-attention agar setiap token dapat melihat token lain secara langsung dan paralel.

Transformer diperkenalkan dalam paper *Attention Is All You Need* [R7]. Ide besarnya: untuk banyak tugas urutan, kita tidak harus membaca token satu per satu seperti RNN. Kita bisa menghitung hubungan antar-token dengan attention.

Self-attention memakai tiga proyeksi:

```
Q = query
K = key
V = value
```

Skor attention:

```
score = QK^T / sqrt(d_k)
attention = softmax(score) V
```

Cara membaca: query bertanya “saya butuh informasi apa?”, key menjawab “saya punya ciri apa?”, value membawa isi informasi. Dot product `QK^T` mengukur kecocokan. Pembagian `sqrt(d_k)` menjaga skala skor tidak terlalu besar.

Contoh kecil dua token dengan `d_k=2`:

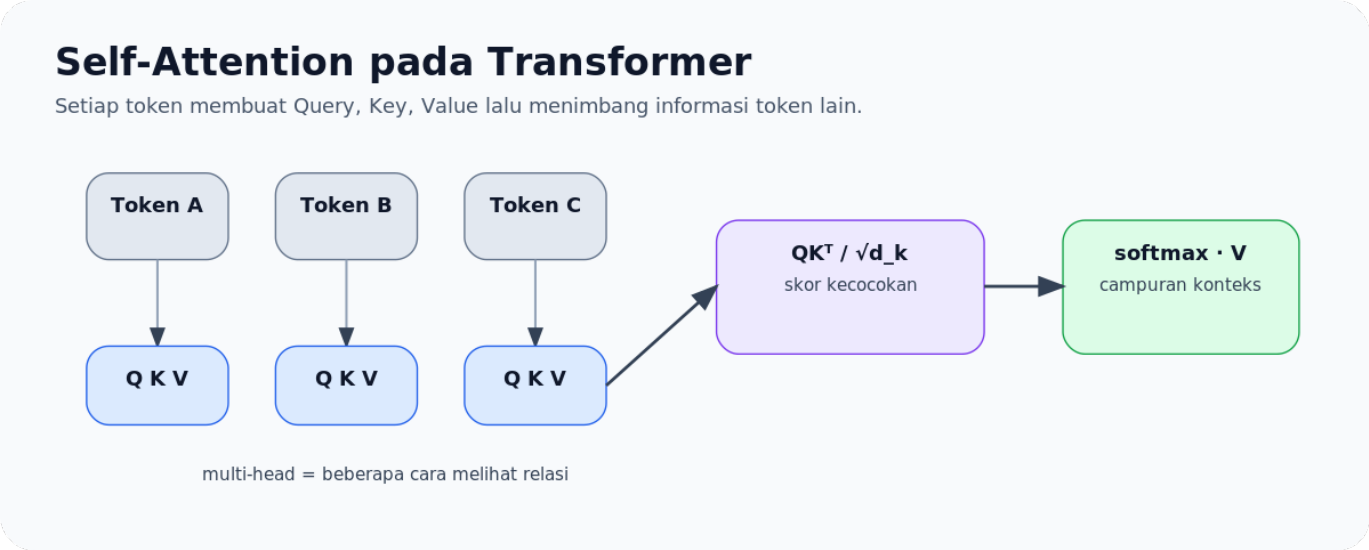
```
q = [1, 0]
k1 = [1, 0]
```

```
k2 = [0, 1]

q·k1 = 1
q·k2 = 0
skor terskala = [1/sqrt(2), 0] ≈ [0,707, 0]
softmax ≈ [0,67, 0,33]
```

Artinya token ini lebih memperhatikan token pertama, tetapi token kedua masih punya kontribusi.

Transformer lebih paralel daripada RNN karena semua token dapat diproses bersamaan dalam layer attention. Paper aslinya melaporkan hasil 28,4 BLEU untuk WMT 2014 English-to-German dan 41,0 BLEU untuk English-to-French, dengan training 3,5 hari pada delapan GPU [R7].



Transformer attention

Komponen penting Transformer:

1. Token embedding: mengubah token menjadi vektor.

2. Positional encoding: memberi informasi urutan karena attention sendiri tidak tahu posisi.

3. Multi-head attention: beberapa attention head melihat relasi berbeda.

4. Feed-forward network: transformasi non-linear per token.

5. Residual + normalization: menstabilkan training.

Catatan kritis: Transformer kuat, tetapi biaya attention standar tumbuh terhadap panjang urutan secara kuadratik:

```
biaya attention ~ O(n²)
```

Jika panjang konteks naik dua kali, matriks attention naik kira-kira empat kali. Karena itu muncul banyak riset efficient attention.

Tes cepat subbab 12

1. Apa arti Q, K, dan V?

2. Mengapa positional encoding diperlukan?

3. Jika panjang urutan naik dari 1.000 ke 2.000, kira-kira berapa kali ukuran matriks attention naik?

Subbab 13 — DBN dan RBM: jejak sejarah unsupervised pretraining

Inti subbab: Deep Belief Network penting secara historis karena menunjukkan cara melatih model dalam secara layer-wise sebelum era ReLU, batch norm, dan data/komputasi besar.

Sebelum deep learning modern stabil, melatih banyak layer dengan backpropagation sering sulit. Deep Belief Network (DBN) memakai tumpukan Restricted Boltzmann Machine (RBM) dan greedy layer-wise training [R8]. Hinton, Osindero, dan Teh memperkenalkan algoritma cepat untuk deep belief nets dengan ide complementary priors dan top layer sebagai associative memory tak berarah.

RBM adalah model probabilistik dengan visible units v dan hidden units h. Energi RBM sering ditulis:

$$E(v,h) = -b^T v - c^T h - v^T W h$$

Probabilitas konfigurasi:

$$P(v,h) = \exp(-E(v,h)) / Z$$

Cara membaca: energi rendah berarti konfigurasi lebih mungkin. Z adalah partition function, penjumlahan semua kemungkinan konfigurasi, yang sering sulit dihitung persis.

DBN secara intuitif belajar representasi bertingkat tanpa label:

input → RBM 1 belajar fitur rendah
fitur 1 → RBM 2 belajar fitur lebih abstrak
fitur 2 → classifier atau fine-tuning supervised

Mengapa DBN tidak dominan sekarang? Karena teknik modern seperti ReLU, batch normalization, residual connection, optimizer lebih baik, dataset besar, dan GPU membuat end-to-end supervised/self-supervised training lebih praktis. Namun DBN penting untuk memahami sejarah: deep learning tidak lahir tiba-tiba pada 2012; banyak ide representasi bertingkat sudah dirintis sebelumnya.

Tes cepat subbab 13

1. Apa peran RBM dalam DBN?

2. Mengapa partition function Z sulit?

3. Mengapa DBN penting secara historis walau jarang dipakai sebagai pilihan utama modern?

Subbab 14 — BNN: Bayesian Neural Network dan Binary Neural Network

Inti subbab: singkatan BNN bisa berarti dua hal berbeda; keduanya penting tetapi jangan tertukar.

Dalam literatur, BNN sering berarti Bayesian Neural Network atau Binary Neural Network. Bab ini membahas keduanya secara ringkas.

Bayesian Neural Network

Bayesian Neural Network tidak hanya mencari satu nilai bobot terbaik, tetapi distribusi atas bobot:

$$P(w \mid D) \propto P(D \mid w)\, P(w)$$

Cara membaca: posterior bobot setelah melihat data sebanding dengan likelihood data jika bobot w dipakai, dikali prior bobot sebelum melihat data.

Keunggulan Bayesian view adalah ketidakpastian. Jika data sedikit atau distribusi baru, model bisa menyatakan prediksi tidak pasti. Ini relevan untuk kesehatan, pembiayaan, dan keputusan risiko tinggi. Kekurangannya: inferensi posterior bobot neural network besar sulit dan mahal.

Binary Neural Network

Binary Neural Network memakai bobot dan/atau aktivasi biner, misalnya -1 dan $+1$, agar komputasi lebih hemat. Paper Binarized Neural Networks menekankan binary weights dan activations pada

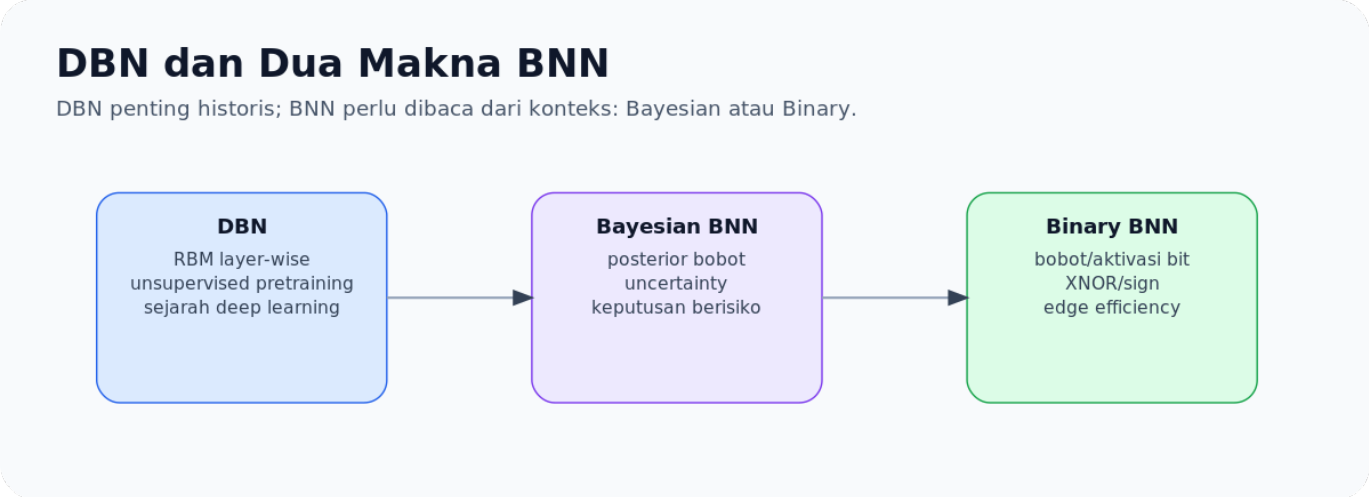
run-time, serta operasi bit-wise untuk mengurangi memori dan operasi aritmetika [R9].

Contoh binarisasi:

w = [-0,7, 0,2, 1,3]
sign(w) = [-1, +1, +1]

Dot product dengan input biner dapat diganti operasi bit tertentu. Ini menarik untuk perangkat terbatas, tetapi akurasi bisa turun jika binarisasi terlalu agresif.

Catatan istilah: saat membaca paper, selalu cek konteks “BNN”. Jika paper bicara posterior, prior, dan uncertainty, itu Bayesian. Jika bicara bit, XNOR, sign, quantization, itu Binary.



DBN dan BNN

Tes cepat subbab 14

1. Apa beda Bayesian Neural Network dan Binary Neural Network?

2. Mengapa Bayesian BNN berguna untuk ketidakpastian?

3. Mengapa Binary BNN cocok untuk efisiensi perangkat?

Subbab 15 — Batch normalization, regularization, dan optimizer

Inti subbab: arsitektur hebat tetap membutuhkan teknik training yang stabil.

Deep learning bukan hanya memilih model. Training sering gagal jika learning rate buruk, data tidak dinormalisasi, gradient meledak, atau model overfit.

Batch normalization

Batch Normalization menormalisasi aktivasi mini-batch [R11]:

$\mu_B = (1/m) \times \Sigma_i x_i$
$\sigma_B^2 = (1/m) \times \Sigma_i (x_i - \mu_B)^2$
$\hat{x}_i = (x_i - \mu_B) / \sqrt{\sigma_B^2 + \varepsilon}$
$y_i = \gamma \hat{x}_i + \beta$

Cara membaca: batch norm membuat aktivasi punya rata-rata sekitar 0 dan variansi sekitar 1, lalu belajar skala γ dan geser β agar model tetap fleksibel.

Contoh: x=[2, 4, 6]:

$\mu = 4$
$\sigma^2 = ((2-4)^2 + (4-4)^2 + (6-4)^2)/3 = 8/3 \approx 2,667$
$\hat{x} \approx [-1,225, 0, 1,225]$

Weight decay

Weight decay memberi penalti bobot besar:

$$L_{total} = L_{data} + \lambda \|w\|^2$$

Jika $L_{data}=0,4$, $\lambda=0,01$, dan $\|w\|^2=30$:

$$L_{total} = 0,4 + 0,01 \times 30 = 0,7$$

Data augmentation

Pada gambar, augmentation bisa berupa crop, flip, perubahan brightness, atau rotasi kecil. Pada teks, bisa paraphrase atau masking hati-hati. Augmentation harus menjaga label tetap benar.

Adam optimizer

Adam memakai estimasi momen pertama dan kedua gradient [R14]. Secara praktis, Adam sering menjadi baseline kuat karena adaptif terhadap skala gradient, tetapi bukan pengganti pemahaman learning rate.

Regularization dan training

Tes cepat subbab 15

1. Apa peran γ dan β pada batch normalization?
2. Hitung L_{total} jika $L_{data}=1,2$, $\lambda=0,001$, $\|w\|^2=500$.
3. Mengapa data augmentation harus menjaga label tetap benar?

Subbab 16 — Transfer learning: memakai pengetahuan model besar secara bijak

Inti subbab: transfer learning memakai model yang sudah belajar dari dataset besar lalu disesuaikan untuk tugas lokal.

Untuk banyak pembaca, melatih ResNet besar dari nol tidak realistis. Transfer learning lebih masuk akal:

$$\text{model pretrained} \rightarrow \text{ganti classifier head} \rightarrow \text{fine-tune pada dataset lokal}$$

Contoh: dataset 2.000 gambar produk lokal. Daripada melatih CNN besar dari awal, gunakan backbone pretrained pada ImageNet, bekukan beberapa layer awal, lalu latih classifier untuk kategori lokal.

Strategi umum:

Kondisi data lokal	Strategi
Sangat sedikit	freeze backbone, latih head kecil
Sedang	unfreeze beberapa blok akhir
Banyak dan mirip target	fine-tune lebih banyak layer
Domain sangat berbeda	perlu pretraining/domain adaptation tambahan

Risiko transfer learning:

1. Dataset pretrained mungkin bias budaya/geografi.
2. Kategori lokal bisa berbeda dari kategori global.
3. Data lokal bisa punya kualitas kamera berbeda.
4. Model besar bisa terlalu berat untuk perangkat target.

Praktik aman: buat baseline sederhana dulu. Bandingkan:

> baseline non-DL → CNN kecil → pretrained CNN → fine-tune

Jika peningkatan kecil tetapi biaya besar, pilih model lebih sederhana.

Tes cepat subbab 16

1. Mengapa transfer learning berguna untuk dataset kecil?
2. Apa risiko memakai model pretrained global pada data Indonesia?
3. Kapan kita sebaiknya membekukan backbone?

Subbab 17 — Membandingkan arsitektur: bukan lomba nama, tetapi cocok masalah

Inti subbab: pemilihan arsitektur harus berdasarkan bentuk data, kebutuhan output, ukuran dataset, komputasi, latency, dan risiko.

Arsitektur	Cocok untuk	Kekuatan	Kelemahan
MLP	tabular sederhana, baseline	mudah	kurang memanfaatkan struktur gambar/urutan
CNN/LeNet	gambar kecil, pola lokal	hemat parameter	konteks global perlu banyak layer
AlexNet	sejarah CNN besar	tonggak GPU+ReLU+dropout	berat dan lama menurut standar kini
VGG	baseline CNN dalam	sederhana dibaca	parameter sangat besar
ResNet	klasifikasi/fitur visual modern	gradient lebih stabil	tetap perlu data/komputasi
YOLO	deteksi objek real-time	cepat, end-to-end	objek kecil/rapat perlu desain hati-hati
U-Net	segmentasi	lokalisasi detail	butuh mask label
MobileNet	edge/mobile	ringan	akurasi bisa turun pada tugas sulit

Arsitektur	Cocok untuk	Kekuatan	Kelemahan
RNN	urutan sederhana	konsep memori mudah	sulit paralel, gradient panjang
LSTM	urutan dependensi panjang	gate memori	lebih lambat daripada attention paralel
Transformer	teks, multimodal, urutan	paralel, attention global	biaya $O(n^2)$ untuk attention standar
DBN	sejarah/unsupervised pretraining	penting historis	jarang jadi pilihan modern utama
Bayesian BNN	uncertainty	prediksi risiko	inferensi mahal
Binary BNN	efisiensi bit	hemat memori/energi	potensi penurunan akurasi

Cara memilih cepat:

```
Jika data gambar dan output kelas      → CNN/ResNet/MobileNet
Jika data gambar dan output box        → YOLO/SSD/Faster R-CNN
Jika data gambar dan output mask       → U-Net/segmentation model
Jika data urutan pendek/sedang         → LSTM atau Transformer kecil
Jika data teks modern                  → Transformer
Jika perlu ketidakpastian eksplisit    → Bayesian approach
Jika perlu perangkat sangat hemat      → MobileNet/quantization/binary
```

Catatan kritis: arsitektur populer bukan jaminan. Dataset buruk, label bocor, metrik salah, atau problem framing kabur akan merusak hasil meskipun modelnya canggih.

Tes cepat subbab 17

1. Arsitektur apa yang cocok untuk segmentasi area banjir dari citra satelit?

2. Arsitektur apa yang cocok untuk deteksi helm pada CCTV proyek?

3. Mengapa memilih Transformer besar untuk dataset kecil bisa berisiko?

Subbab 18 — Lab mental: membaca kurva loss dan gejala training

Inti subbab: praktisi deep learning harus bisa membaca gejala training, bukan hanya menunggu angka akurasi.

Kurva loss adalah alat diagnosis. Empat pola umum:

1. Train loss turun, validation loss turun: training sehat.

2. Train loss turun, validation loss naik: overfitting.

3. Train loss tidak turun: learning rate salah, model kurang kapasitas, data/label bermasalah, atau bug.

4. Loss menjadi NaN: learning rate terlalu besar, gradient exploding, numerik tidak stabil.

Contoh data:

Epoch	Train loss	Val loss	Diagnosis
1	1,20	1,25	awal normal
2	0,80	0,92	membaik
3	0,55	0,78	membaik
4	0,35	0,95	mulai overfit
5	0,20	1,30	overfit jelas

Solusi yang mungkin: data augmentation, weight decay, dropout, early stopping, model lebih kecil, atau tambah data.

Untuk RNN/LSTM, gejala khusus adalah gradient exploding. Solusi umum: gradient clipping.

```
jika ||g|| > threshold:
   g ← threshold × g / ||g||
```

Contoh: norm gradient 10, threshold 2:

```
g_baru = 2 × g / 10 = 0,2g
```

Arah gradient tetap, panjangnya diperkecil.

Tes cepat subbab 18

1. Apa diagnosis jika train loss turun tetapi validation loss naik?

2. Apa fungsi gradient clipping?

3. Sebutkan dua solusi untuk overfitting.

Subbab 19 — Loss function deep learning: apa yang sebenarnya dioptimalkan?

Inti subbab: arsitektur menentukan bentuk model, tetapi loss function menentukan perilaku yang dikejar model.

Di Bab 9 kita sudah melihat loss. Pada deep learning, pemilihan loss menjadi semakin penting karena model sangat fleksibel. Model besar akan mengejar sinyal yang kita berikan, termasuk sinyal yang salah.

Cross-entropy untuk klasifikasi

Untuk klasifikasi satu label, output biasanya softmax:

```
p_k = exp(z_k) / Σ_j exp(z_j)
```

Jika label benar adalah kelas y, cross-entropy:

```
L = -log(p_y)
```

Contoh: model memberi probabilitas untuk tiga kelas produk:

```
p = [0,70, 0,20, 0,10]
y = kelas 1
L = -log(0,70) ≈ 0,357
```

Jika model ragu dan memberi `p_y=0,20`:

```
L = -log(0,20) ≈ 1,609
```

Cara membaca: semakin kecil probabilitas pada kelas benar, semakin besar hukuman.
Cross-entropy tidak hanya menanyakan “benar atau salah”, tetapi “seberapa yakin pada jawaban benar”.

Binary cross-entropy untuk multi-label

Jika satu gambar bisa punya beberapa label, misalnya “helm”, “rompi”, dan “sepatu”, maka gunakan sigmoid per label:

```
L = -[y log(p) + (1-y) log(1-p)]
```

Contoh satu label: `y=1`, `p=0,8`:

```
L = -log(0,8) ≈ 0,223
```

Jika y=0, p=0,8:

L = -log(1-0,8) = -log(0,2) ≈ 1,609

Loss untuk deteksi dan segmentasi

Deteksi objek biasanya menggabungkan beberapa loss:

L_total = L_class + L_box + L_objectness

Segmentasi memakai pixel-wise cross-entropy, Dice loss, atau kombinasi. Untuk dataset tidak seimbang, misalnya area banjir kecil dibanding area tidak banjir, Dice/IoU-oriented loss bisa lebih membantu.

Catatan kritis: metrik bisnis tidak selalu sama dengan loss training. Model bisa punya loss rendah tetapi tidak berguna jika metrik yang dipantau salah. Contoh: pada deteksi penyakit langka, akurasi tinggi bisa menipu karena mayoritas data negatif. Precision, recall, F1, ROC-AUC, PR-AUC, atau calibration perlu dipilih sesuai risiko.

Tes cepat subbab 19

1. Hitung cross-entropy jika kelas benar diberi probabilitas 0,5.

2. Apa perbedaan single-label dan multi-label classification?

3. Mengapa deteksi objek memakai gabungan beberapa loss?

Subbab 20 — Review kritis: kapan deep learning bukan jawaban terbaik?

Inti subbab: praktisi matang tidak memuja arsitektur; ia menguji apakah deep learning benar-benar perlu.

Deep learning sangat kuat, tetapi bukan solusi otomatis untuk semua masalah. Ada situasi ketika model sederhana lebih baik:

1. Data kecil dan tabular bersih. Random forest, gradient boosting, atau logistic regression bisa lebih cepat, lebih mudah dijelaskan, dan cukup akurat.

2. Metrik bisnis butuh interpretabilitas. Untuk kredit, kesehatan, atau kebijakan publik, model yang bisa diaudit sering lebih penting daripada kenaikan akurasi kecil.

3. Label mahal atau berisik. CNN besar akan belajar noise jika label tidak konsisten.

4. Komputasi terbatas. Model besar bisa mahal dilatih dan lambat dijalankan.

5. Distribusi mudah berubah. Model deep learning bisa rapuh terhadap domain shift: kamera berbeda, musim berbeda, bahasa daerah berbeda, atau pola pengguna berubah.

Checklist sebelum memilih deep learning

[ ] Apakah baseline sederhana sudah dibuat?
[ ] Apakah train/validation/test split bebas leakage?
[ ] Apakah label cukup konsisten?
[ ] Apakah metrik sesuai risiko?
[ ] Apakah biaya inferensi masuk akal?
[ ] Apakah ada rencana audit bias dan error?
[ ] Apakah model akan dipakai untuk keputusan sensitif?

Contoh keputusan

Kasus A: toko online kecil ingin memprediksi omzet harian dari 12 fitur tabular selama 200 hari. Deep learning kemungkinan berlebihan. Mulai dari baseline rata-rata musiman, linear regression, random forest, atau gradient boosting.

Kasus B: aplikasi ingin mengenali kerusakan daun cabai dari ribuan foto dengan variasi pencahayaan. CNN atau transfer learning masuk akal, tetapi perlu validasi pada kebun berbeda.

Kasus C: sistem membaca keluhan pelanggan dalam bahasa Indonesia informal. Transformer kecil atau embedding pretrained bisa membantu, tetapi harus diuji pada slang lokal, typo, campuran bahasa daerah, dan data terbaru.

Kesalahan umum pemula

• Memilih arsitektur karena populer, bukan karena cocok.

• Melatih model besar tanpa baseline.

• Memakai test set berkali-kali sampai bocor menjadi validation set.

• Melaporkan akurasi saja pada data tidak seimbang.

• Tidak membaca contoh error model.

• Tidak menghitung biaya parameter, latency, dan memori.

Prinsip bab ini: deep learning adalah alat. Alat yang kuat harus dipakai dengan disiplin eksperimen.

Tes cepat subbab 20

1. Sebutkan dua kondisi ketika model non-DL bisa lebih cocok.

2. Mengapa baseline sederhana wajib dibuat?

3. Apa bahaya memakai akurasi saja pada data tidak seimbang?

Praktikum utama Bab 10 — Deep learning playground dari NumPy

Pada folder `code/`, bab ini menyediakan:

```
code/deep_learning_playground.py
code/deep_learning_lab.ipynb
```

Isi praktikum:

1. Menghitung output convolution dan pooling manual.

2. Menghitung parameter dan biaya operasi CNN.

3. Membandingkan residual block dengan blok biasa.

4. Menjalankan satu langkah RNN dan LSTM kecil.

5. Menghitung self-attention mini.

6. Membandingkan convolution standar vs depthwise separable.

7. Membuat kurva loss sintetis untuk diagnosis overfitting.

Kode sengaja ringan agar bisa dijalankan di terminal, VS Code, Jupyter, Google Colab, dan Kaggle Notebook. Jika `matplotlib` tersedia, script membuat plot di folder `outputs/`. Jika tidak tersedia, perhitungan inti tetap berjalan.

Ringkasan bab

1. Deep learning adalah komposisi banyak layer yang belajar representasi bertingkat.

2. Shape tensor adalah bahasa dasar untuk membaca arsitektur.

3. CNN memakai convolution, weight sharing, pooling, dan hierarki fitur.

4. LeNet membuktikan CNN klasik untuk dokumen/digit.

5. AlexNet memicu kebangkitan deep learning modern lewat data besar, GPU, ReLU, dan dropout.

6. VGG menunjukkan kekuatan blok 3×3 sederhana tetapi berat.

7. ResNet memakai residual connection agar jaringan sangat dalam lebih mudah dilatih.

8. YOLO merumuskan deteksi objek sebagai prediksi sekali jalan.

9. MobileNet, U-Net, dan Inception menunjukkan trade-off efisiensi, lokalisasi, dan multi-skala.

10. RNN dan LSTM memodelkan urutan; LSTM memakai gate untuk memori panjang.

11. Transformer memakai self-attention dan menjadi fondasi banyak model modern.

12. DBN penting dalam sejarah unsupervised pretraining.

13. BNN bisa berarti Bayesian atau Binary; keduanya berbeda tujuan.

14. Training stabil membutuhkan normalization, regularization, optimizer, dan diagnosis kurva.

15. Pemilihan arsitektur harus dimulai dari masalah, data, metrik, dan batas komputasi.

Latihan terpadu akhir bab

1. Sebuah input 64×64×3 masuk convolution 3×3, padding 1, stride 1, 32 filter. Berapa shape output dan jumlah parameter?

2. Hitung biaya operasi convolution standar untuk `D_K=3`, `M=32`, `N=64`, `D_F=64`.

3. Hitung biaya depthwise separable untuk angka yang sama. Berapa rasio penghematannya?

4. Jelaskan mengapa residual connection membantu training jaringan sangat dalam.

5. Buat tabel pilihan arsitektur untuk tiga kasus Indonesia: klasifikasi kualitas cabai, deteksi helm, prediksi permintaan harian.

6. Hitung IoU jika box A luas 80, box B luas 50, irisan 30.

7. Hitung satu langkah LSTM: `c_old=1,5`, `f=0,7`, `i=0,4`, `g=0,8`.

8. Jelaskan mengapa Transformer lebih mudah diparalelkan daripada RNN.

9. Bedakan Bayesian BNN dan Binary BNN dengan satu contoh aplikasi masing-masing.

10. Jika train loss turun dari 1,0 ke 0,1 tetapi validation loss naik dari 0,8 ke 1,6, diagnosis dan solusi apa yang masuk akal?

Referensi inti bab

Daftar bibliografi lengkap ada di `references.md`. Penanda `[R...]` di bab ini merujuk pada referensi inti tersebut, terutama LeNet [R1], AlexNet [R2], VGG [R3], ResNet [R4], YOLO [R5], LSTM [R6], Transformer [R7], DBN [R8], Binarized Neural Networks [R9], Dropout [R10], Batch Normalization [R11], MobileNet [R12], U-Net [R13], dan Adam [R14].