Meeting 1: Evolusi AI — Dari Deep Learning ke Generative AI

AI-40X: Generative AI & Large Language Models

Hendri Karisma, M.T.
Dosen Teknik Informatika STMIK Tazkia
VP Engineering at jejakin.com, 2026

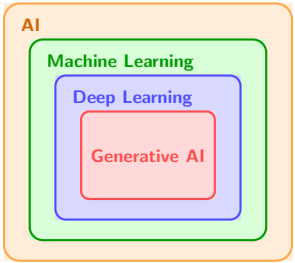
Semester 4 — 2025/2026

Outline

Perjalanan Kita Sejauh Ini

- **Semester 2:** Probabilitas & Statistika — distribusi, Bayes' theorem.
- **Semester 3:** Machine Learning & Deep Learning — regresi, klasifikasi, CNN, RNN dasar.
- **Semester 4 (sekarang): Generative AI & LLM** — model yang *menghasilkan* data baru.

Pertanyaan kunci: Apa bedanya model yang *membedakan* kucing vs anjing, dengan model yang *membuat gambar* kucing baru?



Definisi: AI
Sistem yang meniru kecerdasan manusia.

Discriminative vs Generative Models

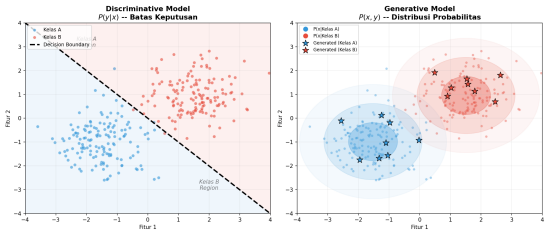
Definisi: Discriminative Model

Mempelajari batas keputusan $P(y|x)$ — "Ini kucing atau anjing?"

Definisi: Generative Model

Mempelajari distribusi data $P(x)$ — "Seperti apa kucing itu?" lalu bisa membuat kucing baru.

Aspek	Discriminative	Generative
Tujuan	Klasifikasi	Generasi
Memodelkan	$P(y\|x)$	$P(x)$ atau $P(x\|y)$
Contoh	SVM, Logistic Reg, CNN classifier	VAE, GAN, GPT, Diffusion
Output	Label/kategori	Data baru
Analogi	Hakim (menilai)	Seniman (membuat)



Timeline: Evolusi Arsitektur AI

Tahun	Model	Terobosan
1958	Perceptron	Neuron buatan pertama (Rosenblatt)
1986	Backpropagation	Training multi-layer networks (Rumelhart et al.)
1998	LeNet (CNN)	Konvolusi untuk pengenalan digit (LeCun)
2012	AlexNet	CNN dalam di ImageNet — era Deep Learning dimulai
2014	GAN	Generative Adversarial Network (Goodfellow)
2014	Seq2Seq + Attention	Bahdanau Attention untuk NMT
2017	**Transformer**	"Attention Is All You Need" (Vaswani et al.)
2018–2023	GPT-1 → GPT-4	Era Large Language Models
2023–now	LLaMA, Mistral	Open-source LLM revolution

Tema utama: Dari arsitektur sederhana → model yang semakin besar & mampu menghasilkan konten.

Mengapa Kita Butuh RNN?

Definisi: Data Sekuensial

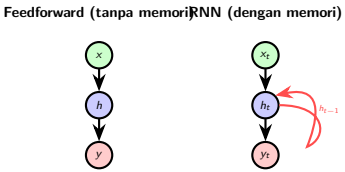
Data di mana **urutan** elemen penting untuk maknanya. Contoh: kalimat, audio, time series. "Saya makan nasi" $\neq$ "Nasi makan saya".

Masalah: Feedforward NN memproses input **independen** — tanpa ingatan.

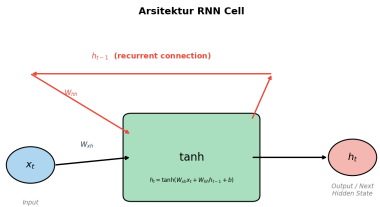
Contoh data sekuensial:

- **Teks:** "Saya suka AI" — urutan = makna
- **Audio:** Nada berurutan = musik
- **Time series:** Harga saham hari ini ← kemarin

Analogi: Membaca buku kata per kata — otak menggabungkan konteks. RNN bekerja seperti ini!



Recurrent Neural Network: Arsitektur



Definisi: Hidden State (h_t)

"Ingatan" RNN pada waktu t. Merangkum semua informasi dari input $x_1, x_2, \ldots, x_t$ yang sudah diproses.

Persamaan inti:

$$h_t = \tanh(W_{xh}\, x_t + W_{hh}\, h_{t-1} + b)$$

Komponen:

- x_t: Input pada waktu t (embedding kata)
- h_{t-1}: Hidden state sebelumnya ("ingatan")
- W_{xh}: "Seberapa penting input baru?"
- W_{hh}: "Seberapa penting ingatan lama?"

Kata kunci: RNN menggabungkan **input baru** + **ingatan lama** $\rightarrow$ ingatan yang diupdate.

Masalah Vanishing Gradient

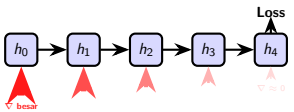
Definisi: Vanishing Gradient

Fenomena di mana gradien menjadi **sangat kecil** (mendekati nol) saat backpropagation melalui banyak time step, membuat model gagal belajar dependensi jarak jauh.

Contoh: "Saya lahir di **Indonesia**, tumbuh besar di Jakarta, kuliah di ITB, dan saya fasih berbahasa ___"

Gradien dikalikan berulang dengan W_{hh}:

$$\frac{\partial L}{\partial h_0} \propto W_{hh}^T$$



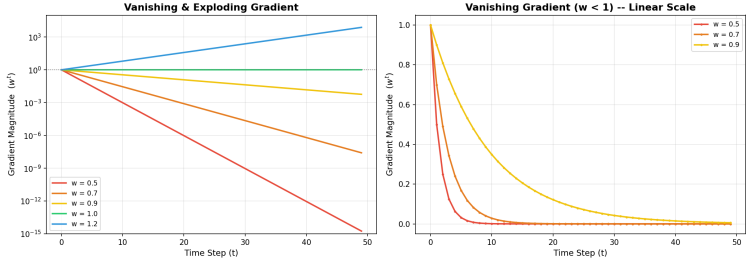
Gradien semakin kecil (vanishing) dari Loss ke h_0

Contoh numerik:

- $0.7^3 = 0.343$
- $0.7^{10} = 0.028$
- $0.7^{50} \approx 0$!

Analogi: Telepon rusak — pesan semakin kabur setelah banyak orang.

Simulasi Vanishing Gradient



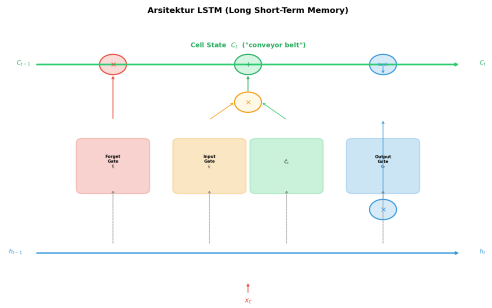
Grafik menunjukkan besaran gradien yang menyusut secara eksponensial seiring bertambahnya jumlah time steps. Dengan faktor 0.7, setelah 50 langkah gradien sudah hampir nol.

Solusi: LSTM — Ide Utama

Definisi: LSTM (Long Short-Term Memory)
Varian RNN yang mengatasi vanishing gradient dengan menambahkan **Cell State** (C_t) — jalur informasi terpisah yang dikontrol oleh 3 **gates** (forget, input, output). Diperkenalkan oleh Hochreiter & Schmidhuber (1997).

Analogi Notebook:

- **Cell State** = Halaman notebook (catatan jangka panjang)
- **Hidden State** = Papan tulis (ringkasan saat ini)
- **Gates** = Tiga aksi kontrol notebook



Tiga gerbang (gates):

1. **Forget**: "Apa yang *dilupakan*?"
2. **Input**: "Info baru apa yang *disimpan*?"
3. **Output**: "Bagian mana jadi *output*?"

LSTM: Cell State Update

Langkah 1: Lupakan info lama yang tidak relevan

$$C_t^{(\text{lupa})} = f_t \odot C_{t-1}$$

Langkah 2: Tambahkan info baru yang penting

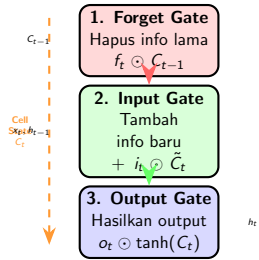
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

Langkah 3: Hasilkan output

$$h_t = o_t \odot \tanh(C_t)$$

Mengapa ini menyelesaikan vanishing gradient?
Cell state di-update via **penjumlahan** (+), bukan perkalian berulang!

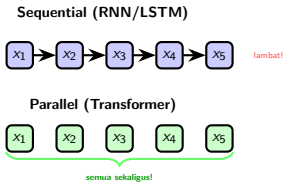


Sequential Bottleneck: Mengapa LSTM Tidak Cukup?

Definisi: Sequential Bottleneck

Keterbatasan di mana model **harus memproses data satu per satu** secara berurutan ($x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_T$), sehingga tidak bisa diparalelkan dan lambat untuk sequence panjang.

- LSTM **berhasil** atasi vanishing gradient.
- **Tapi** masih sekuensial — harus proses $x_1, x_2, \ldots, x_T$ satu per satu.
- Konsekuensi:
  - Tidak bisa paralel di GPU.
  - Waktu training $\mathcal{O}(T)$.
  - 10.000 token? Sangat lambat!



Solusi radikal: Buang rekurensi → **Transformer** (2017).

Masalah Bottleneck pada Seq2Seq

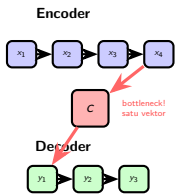
Definisi: Seq2Seq (Sequence-to-Sequence)
Arsitektur yang terdiri dari **Encoder** (membaca input menjadi representasi) dan **Decoder** (menghasilkan output dari representasi). Digunakan untuk machine translation, summarization, dll.

Seq2Seq klasik (Sutskever, 2014):

- **Encoder:** Baca kalimat input → ringkas ke **satu vektor** c.
- **Decoder:** Gunakan c untuk generate terjemahan.

Masalah: Seluruh makna "dijejalkan" ke satu vektor.

- Kalimat pendek: OK.
- Kalimat panjang (50+ kata): info hilang!



"Meringkas buku jadi satu kalimat, lalu menulis ulang buku dari kalimat itu."

Mekanisme Bahdanau Attention

Definisi: Attention

Mekanisme yang memungkinkan decoder "**melihat kembali**" ke semua hidden state encoder dan memilih informasi yang paling relevan untuk setiap langkah output. Setiap posisi output mendapat context vector yang **berbeda**.

Langkah-langkah:

1. **Alignment score:**

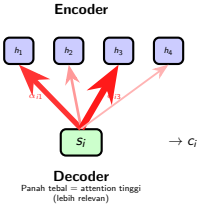
$$e_{ij} = v_a^T \tanh(W_a s_{i-1} + U_a h_j)$$

2. **Attention weights:**

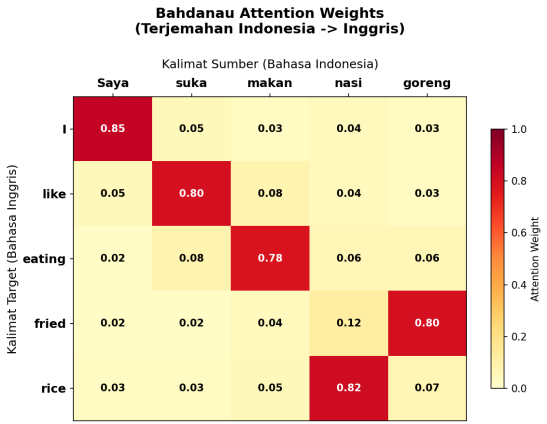
$$\alpha_{ij} = \text{softmax}(e_{ij})$$

3. **Context vector:**

$$c_i = \sum_{j=1}^{T} \alpha_{ij} \, h_j$$



Visualisasi Attention Weights



Cara membaca heatmap:

- Sumbu horizontal: kata-kata input (encoder).
- Sumbu vertikal: kata-kata output (decoder).
- Warna cerah/terang: attention weight tinggi.
- Model secara otomatis belajar **alignment** antara kata input dan output.

Insight penting:

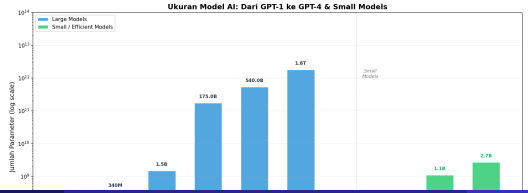
- Attention membuat model *interpretable* — kita bisa melihat "ke mana model melihat".
- Ini menjadi fondasi bagi **Transformer**.

Timeline Large Language Models

Definisi: LLM (Large Language Model)

Model bahasa dengan **miliaran parameter** yang dilatih pada teks dalam jumlah besar. Mampu memahami dan menghasilkan teks natural, melakukan reasoning, dan menyelesaikan berbagai tugas tanpa training khusus (zero/few-shot).

Model	Tahun	Parameter	Terobosan
GPT-1	2018	117M	Pre-training + fine-tuning
GPT-2	2019	1.5B	Zero-shot, “too dangerous to release”
GPT-3	2020	175B	Few-shot learning, in-context learning
ChatGPT	Nov 2022	–	RLHF, dialog yang natural
GPT-4	Mar 2023	∼1.8T*	Multimodal (teks + gambar)
LLaMA	Feb 2023	7–65B	Open-weight oleh Meta
Mistral	Sep 2023	7B	Efisien, sliding window attention



Studi Kasus: Dampak Generative AI di Indonesia

Definisi: Fine-tuning

Proses melatih ulang model pre-trained pada data spesifik (misal: bahasa Indonesia) agar performanya lebih baik untuk tugas tertentu.

Peluang bagi UMKM & industri lokal:

- **Customer service:** Chatbot bahasa Indonesia.
- **Konten:** Deskripsi produk, copywriting.
- **Analisis:** Sentimen review, prediksi tren.
- **Pendidikan:** Tutor AI personal.
- **Pertanian:** Analisis citra satelit.

Tantangan:

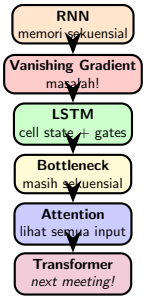
- Bahasa Indonesia *under-represented* di training data LLM.
- Infrastruktur GPU terbatas.
- Literasi AI masyarakat rendah.
- Regulasi & etika belum matang.

Definisi: Inference

Proses menggunakan model yang sudah dilatih untuk menghasilkan prediksi/output pada data baru. Berbeda dengan training, inference tidak mengubah parameter model.

Kesimpulan Pertemuan 1

1. **Discriminative vs Generative:** Model generatif mempelajari $P(x)$ untuk menghasilkan konten baru.
2. **RNN** punya memori sekuensial tapi menderita **vanishing gradient**.
3. **LSTM** mengatasi VG dengan cell state & gates, tapi masih **sekuensial**.
4. **Bahdanau Attention** mengatasi bottleneck dengan context vector dinamis.
5. Evolusi GPT-1 $\rightarrow$ open-source LLM = demokratisasi AI.



Pertemuan berikutnya: Arsitektur Transformer — Self-Attention & Multi-Head Attention.

Referensi

- Hochreiter, S. & Schmidhuber, J. (1997). “Long Short-Term Memory.” *Neural Computation*, 9(8), 1735–1780.
- Bahdanau, D., Cho, K. & Bengio, Y. (2014). “Neural Machine Translation by Jointly Learning to Align and Translate.” *arXiv:1409.0473*.
- Sutskever, I., Vinyals, O. & Le, Q.V. (2014). “Sequence to Sequence Learning with Neural Networks.” *NeurIPS*.
- Vaswani, A. et al. (2017). “Attention Is All You Need.” *NeurIPS*.
- Radford, A. et al. (2018). “Improving Language Understanding by Generative Pre-Training.” OpenAI.
- Brown, T. et al. (2020). “Language Models are Few-Shot Learners.” *NeurIPS*.
- Touvron, H. et al. (2023). “LLaMA: Open and Efficient Foundation Language Models.” *arXiv:2302.13971*.
- Olah, C. (2015). “Understanding LSTM Networks.” *colah’s blog*.