Meeting 7: Fine-Tuning Efisien (LoRA/QLoRA)

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

Base Model: Mesin Pelengkap Dokumen

- **Base model** (GPT, Llama, Mistral) hanyalah *document completer*.
- Ia dilatih dengan satu tujuan: **memprediksi token berikutnya** berdasarkan konteks sebelumnya.

$$P(x_t \mid x_1, x_2, \ldots, x_{t-1})$$

- Jika diberi prompt "Ibu kota Indonesia adalah", model melanjutkan: "Jakarta, yang terletak di..."
- **Masalah:** Base model *tidak tahu cara menjawab pertanyaan*.
  - Prompt: "Apa ibu kota Indonesia?" $\rightarrow$ Model bisa melanjutkan: "Apa ibu kota Jepang? Apa ibu kota..." (meneruskan pola soal, bukan menjawab).
- Kita perlu mengubah *completer* menjadi *assistant*.

Instruction Tuning & SFT

- **Instruction Tuning:** Melatih model pada pasangan (*instruction*, *response*).
- Contoh data:
  - Instruksi: "Jelaskan apa itu fotosintesis dalam 2 kalimat."
  - Respons: "Fotosintesis adalah proses tanaman mengubah cahaya matahari menjadi energi..."
- **SFT (Supervised Fine-Tuning):** Training dengan data percakapan berformat chat.
- Model belajar *kapan harus menjawab* dan *bagaimana format jawaban yang baik*.

Chat Format (Template Prompt):

```
<|user|>
Jelaskan apa itu fotosintesis dalam 2 kalimat.
<|assistant|>
Fotosintesis adalah proses tanaman mengubah cahaya matahari menjadi energi kimia
dengan bantuan klorofil. Proses ini menghasilkan oksigen dan glukosa.
```

Dari Pre-training ke Deployment

Pipeline Modern LLM

1. **Pre-training** — Self-supervised pada triliunan token (mahal, dilakukan perusahaan besar).
2. **Supervised Fine-Tuning (SFT)** — Latih pada data instruksi berkualitas tinggi.
3. **Alignment (RLHF/DPO)** — Selaraskan dengan preferensi manusia (pertemuan 9–10).
4. **Deployment** — Serve model ke pengguna akhir.

Posisi Kita Hari Ini

Kita fokus pada langkah **ke-2 (SFT)** — bagaimana melakukan fine-tuning secara efisien agar bisa dilakukan di hardware terbatas.

Ukuran LLM Modern

- Model 7B parameter (Llama-2 7B, Mistral 7B):
  - FP32: $7 \times 10^9 \times 4$ bytes = **28 GB**
  - FP16: $7 \times 10^9 \times 2$ bytes = **14 GB**
- **Full fine-tuning** membutuhkan:
  - Model weights: 14 GB (FP16)
  - Gradients: 14 GB
  - Optimizer states (Adam): 28 GB
  - Activations: 10–20+ GB
  - **Total:** $>$ **70–100 GB VRAM!**

GPU	VRAM
RTX 3060	12 GB
RTX 4090	24 GB
A100	80 GB
H100	80 GB
Colab Free	15 GB

Table: VRAM GPU umum

Kesimpulan

Full fine-tuning model 7B **tidak mungkin** di GPU konsumer! Bahkan A100 pun ketat.

Solusi: PEFT (Parameter-Efficient Fine-Tuning)

- **Ide PEFT:** Jangan update *semua* parameter — cukup update *sebagian kecil* saja.
- Bekukan (freeze) model asli, tambahkan modul kecil yang bisa dilatih.
- Beberapa metode PEFT:
  1. **Prefix Tuning** — Tambah virtual tokens di awal input.
  2. **Adapters** — Sisipkan modul kecil di antara layer.
  3. **LoRA** — Dekomposisi perubahan bobot ke matriks rank rendah. ← *Fokus hari ini!*
  4. **QLoRA** — LoRA + quantization untuk efisiensi memori lebih lanjut.
- **Keuntungan:** Hanya $< 1\%$ parameter yang dilatih, memori jauh lebih hemat.

Hipotesis Low-Rank

- **Hipotesis (Hu et al., 2021):** Perubahan bobot selama fine-tuning memiliki *rank rendah*.
- Artinya: meskipun matriks bobot W berukuran besar ($d \times d$), perubahan ΔW bisa diaproksimasi oleh perkalian dua matriks kecil.

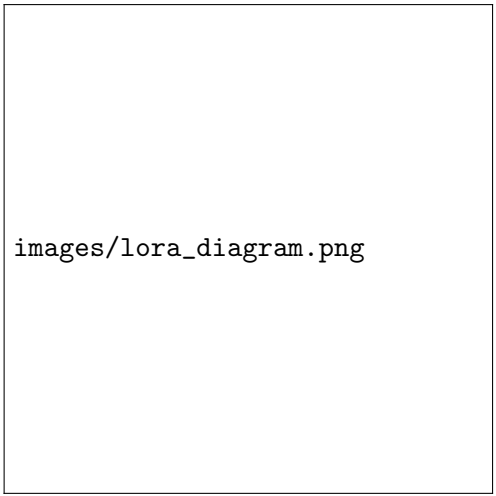
Dekomposisi LoRA

$$W_{\text{new}} = W_{\text{frozen}} + \Delta W = W_{\text{frozen}} + B \times A$$

Di mana:

- $W_{\text{frozen}} \in \mathbb{R}^{d \times d}$ — bobot asli, **tidak diubah** (frozen).
- $B \in \mathbb{R}^{d \times r}$ — matriks down-projection.
- $A \in \mathbb{R}^{r \times d}$ — matriks up-projection.
- $r \ll d$ — rank LoRA (biasanya $r = 8, 16, 32, 64$).

Visualisasi LoRA



Analogi: Sticky Notes pada Ensiklopedia

- Ensiklopedia = bobot model asli (frozen).
- Sticky notes = adapter LoRA (kecil, mudah ditempel/dilepas).
- Kita *tidak menulis ulang seluruh buku*, cukup menempel catatan kecil di halaman yang relevan.

Contoh Numerik:

- $d = 4096$, $r = 16$
- Bobot asli: $4096 \times 4096 = 16.7\text{M}$ params
- LoRA: $4096 \times 16 + 16 \times 4096 = 131\text{K}$ params
- Rasio: $< \mathbf{1}\%$ dari parameter asli!

Keunggulan LoRA

1. **Efisien Memori:** Hanya $< 1\%$ parameter yang dilatih $\rightarrow$ gradient dan optimizer state jauh lebih kecil.
2. **Adapter Portabel:** Hasil fine-tuning berukuran MB, bukan GB.
   - Model Llama-2 7B: 14 GB (FP16).
   - Adapter LoRA: 10–50 MB saja!
3. **Modular:** Satu base model bisa punya banyak adapter untuk tugas berbeda.
   - Adapter A: chatbot customer service.
   - Adapter B: penerjemah Indonesia–Inggris.
   - Adapter C: asisten coding.
4. **No Inference Overhead:** Saat deploy, ΔW bisa di-*merge* ke W $\rightarrow$ kecepatan sama dengan model asli.

Implementasi LoRA dengan PEFT

```python
from peft import LoraConfig, get_peft_model
from transformers import AutoModelForCausalLM

# Load base model
model = AutoModelForCausalLM.from_pretrained("TinyLlama/TinyLlama-1.1B-Chat-v1.0")

# Konfigurasi LoRA
lora_config = LoraConfig(
    r=16,                          # Rank LoRA
    lora_alpha=32,                 # Scaling factor
    target_modules=["q_proj", "v_proj", "k_proj", "o_proj"],
    lora_dropout=0.05,
    bias="none",
    task_type="CAUSAL_LM"
)

# Terapkan LoRA ke model
model = get_peft_model(model, lora_config)
model.print_trainable_parameters()
# Output: trainable params: 4,194,304 || all params: 1,100,048,384
#         || trainable%: 0.38%
```

Quantization: Kompresi Presisi

- **Quantization:** Mengurangi presisi bilangan dari floating-point ke integer.
- Mengubah representasi bobot: FP32 (4 byte) $\rightarrow$ FP16 (2 byte) $\rightarrow$ INT8 (1 byte) $\rightarrow$ INT4 (0.5 byte).

Presisi	Ukuran per Parameter	Model 7B
FP32	4 bytes	28 GB
FP16 / BF16	2 bytes	14 GB
INT8	1 byte	7 GB
INT4 (NF4)	0.5 byte	3.5 GB

- **NormalFloat4 (NF4):** Tipe data kuantisasi yang dioptimalkan untuk distribusi bobot neural network (distribusi normal). Diperkenalkan oleh Dettmers et al. (2023).
- Degradasi performa? Sangat kecil! Model 4-bit hampir setara dengan FP16 untuk kebanyakan tugas.

QLoRA: Quantized + LoRA

Ide QLoRA (Dettmers et al., 2023)

Kombinasikan **quantization 4-bit** pada model dasar dengan **adapter LoRA** yang dilatih dalam FP16.

Komponen kunci QLoRA:

1. **4-bit NormalFloat** — Kuantisasi bobot model ke 4-bit NF4.
2. **Double Quantization** — Kuantisasi juga konstanta kuantisasi → hemat memori tambahan.
3. **Paged Optimizers** — Gunakan RAM CPU saat GPU VRAM penuh (via unified memory).
4. **LoRA adapters** — Hanya adapter yang dilatih dalam FP16/BF16.

Hasil

Fine-tuning model **7B parameter** kini bisa dilakukan di **GPU 8–16 GB VRAM** (RTX 3060, RTX 4060, Google Colab Free)!

Implementasi QLoRA

```
from transformers import AutoModelForCausalLM, BitsAndBytesConfig
from peft import LoraConfig, get_peft_model
import torch

# Konfigurasi quantization 4-bit
bnb_config = BitsAndBytesConfig(
    load_in_4bit=True,
    bnb_4bit_quant_type="nf4",            # NormalFloat4
    bnb_4bit_compute_dtype=torch.bfloat16, # Komputasi dalam BF16
    bnb_4bit_use_double_quant=True,        # Double quantization
)

# Load model dalam 4-bit
model = AutoModelForCausalLM.from_pretrained(
    "mistralai/Mistral-7B-v0.1",
    quantization_config=bnb_config,
    device_map="auto"
)

# Tambahkan LoRA adapter
lora_config = LoraConfig(r=16, lora_alpha=32, target_modules=["q_proj","v_proj"],
                         lora_dropout=0.05, bias="none", task_type="CAUSAL_LM")
```

Bias dalam Data & Model

- **Prinsip dasar:** Data masuk → model belajar → output mencerminkan data.
- Jika data training mengandung **bias**, maka model juga akan **bias**.
- Contoh bias:
  - Data dominan bahasa Inggris → model buruk di bahasa Indonesia.
  - Data mengandung stereotip gender → "dokter" selalu diasosiasikan laki-laki.
  - Data dari sumber tertentu → model cenderung ke sudut pandang tertentu.
- **Etika anotasi data:** Siapa yang melabeli data?
  - Pekerja kontrak di negara berkembang dengan upah rendah?
  - Apakah mereka terpapar konten berbahaya tanpa dukungan kesehatan mental?
  - Ref: *"OpenAI Used Kenyan Workers on Less Than $2/hr"* — TIME, 2023.

Tanggung Jawab Pengembang AI

- **Responsible Development:**
  1. Uji model sebelum deploy — *red teaming*, evaluasi bias.
  2. Dokumentasikan limitasi model (*model card*).
  3. Sediakan mekanisme feedback dan pelaporan.
- **Pertanyaan untuk refleksi:**
  - Siapa yang dirugikan jika model memberikan output salah?
  - Apakah model digunakan untuk konteks yang tepat?
  - Bagaimana mencegah penyalahgunaan?

Preview Pertemuan 9–10

Kita akan membahas lebih dalam tentang:

- **Alignment:** RLHF, DPO — bagaimana menyelaraskan model dengan nilai manusia.
- **Safety & Security:** Prompt injection, jailbreaking, guardrails.

Kesimpulan

1. **Base model** hanyalah pelengkap dokumen; perlu **instruction tuning / SFT** agar menjadi assistant.
2. **Full fine-tuning** LLM membutuhkan VRAM sangat besar (>100 GB) — tidak praktis.
3. **LoRA** mendekomposisi perubahan bobot menjadi matriks rank rendah (< 1% parameter trainable).
4. **QLoRA** menambahkan quantization 4-bit sehingga fine-tuning 7B model bisa di GPU konsumer.
5. **Etika AI:** Bias dalam data menghasilkan bias dalam model — kita bertanggung jawab.

Untuk Pertemuan Berikutnya (Meeting 8: UTS)

Persiapkan Tugas Besar 1: NanoGPT from scratch **atau** Fine-tune model kecil dengan LoRA/QLoRA.

Referensi

- Hu, E.J., et al. (2021). "LoRA: Low-Rank Adaptation of Large Language Models." *arXiv:2106.09685*.
- Dettmers, T., et al. (2023). "QLoRA: Efficient Finetuning of Quantized Language Models." *arXiv:2305.14314*.
- Ouyang, L., et al. (2022). "Training language models to follow instructions with human feedback (InstructGPT)." *NeurIPS*.
- **Library:** Hugging Face PEFT, BitsAndBytes.
- Perrigo, B. (2023). "OpenAI Used Kenyan Workers on Less Than $2 Per Hour." *TIME*.