Parameter-Efficient Tuning (PEFT)

Fine-tuning-এর আধুনিক ও বাস্তবসম্মত বিকল্প

১) **Parameter-Efficient Tuning (PEFT)** কী?

Parameter-Efficient Tuning (PEFT) হলো এমন কিছু কৌশল, যেখানে আমরা—

- পুরো **LLM-**এর সব প্যারামিটার আপডেট করি না
- বরং খুব অল্প সংখ্যক অতিরিক্ত বা নির্বাচিত প্যারামিটার ট্রেন করি
- অথচ পারফরম্যান্স প্রায় full fine-tuning-এর কাছাকাছি পাই

সহজ ভাষায়:

> **“Model** বড়**,** কিন্তু **train** হয় ছোট অংশ।**”**

২) কেন **PEFT** দরকার হলো**?**

Full fine-tuning করতে গেলে বাস্তবে অনেক সমস্যা হয়:

Full Fine-tuning-এর সমস্যা

- বিলিয়ন প্যারামিটার আপডেট → খুব বেশি **GPU** মেমোরি
- প্রতিটা কাজের জন্য আলাদা model copy
- খরচ বেশি, সময় বেশি
- ছোট টিম / স্টার্টআপের জন্য প্রায় অসম্ভব

এই সমস্যা থেকেই PEFT এসেছে।

৩) **PEFT-**এর মূল আইডিয়া **(Core Idea)**

PEFT ধরে নেয় একটি গুরুত্বপূর্ণ কথা:

> Pre-trained foundation model ইতোমধ্যেই অনেক কিছু জানে
> আমাদের শুধু সামান্য **adjustment** দরকার

তাই:

- Base model **freeze** করা হয়
- কিছু ছোট **trainable component** যোগ করা হয়
- শুধু সেগুলো train করা হয়

৪) PEFT বনাম Full Fine-tuning

বিষয়	Full Fine-tuning	PEFT
Trainable parameters	সব	খুব অল্প
GPU memory	খুব বেশি	অনেক কম
Training cost	বেশি	কম
Multiple tasks	আলাদা model	আলাদা adapter
Industry use	সীমিত	ব্যাপক

Industry reality:
আজকাল বড় বড় কোম্পানিও PEFT ছাড়া প্রোডাকশন LLM টিউন করে না।

৫) PEFT-এর সবচেয়ে জনপ্রিয় পদ্ধতিগুলো

এখন আমরা একে একে সবচেয়ে গুরুত্বপূর্ণ PEFT কৌশলগুলো বুঝব।

৬) LoRA (Low-Rank Adaptation)

LoRA কী?

LoRA হলো সবচেয়ে জনপ্রিয় PEFT পদ্ধতি।

ধারণা:

- Transformer-এর বড় weight matrix **W**
- আমরা সরাসরি W আপডেট করি না

- বরং একটি ছোট low-rank matrix যোগ করি

গাণিতিকভাবে:

$W' = W + \Delta W$
$\Delta W = A \times B$ (A এবং B ছোট matrix)

এখানে:

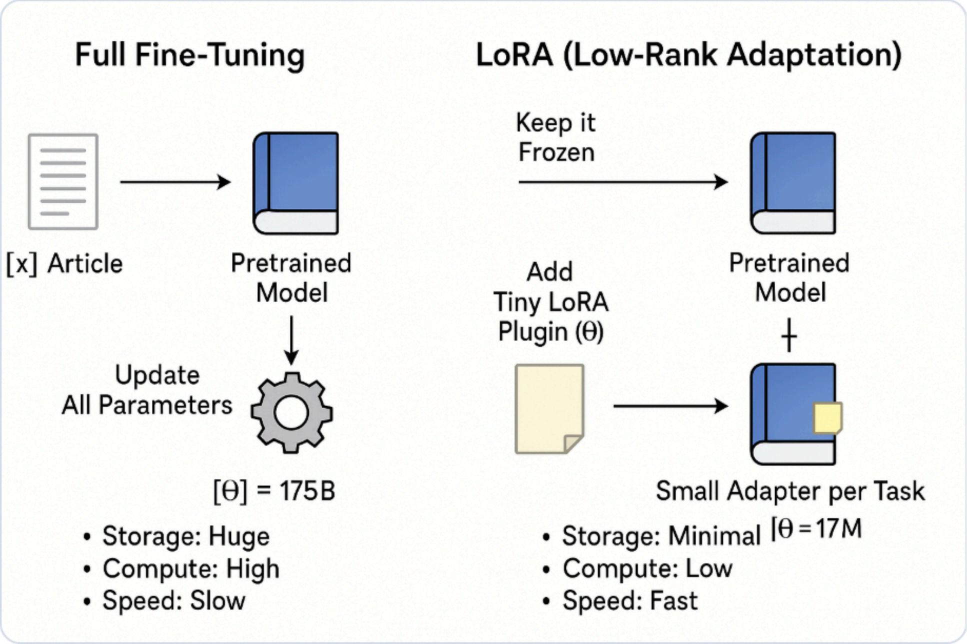
- W = frozen base weight
- A, B = trainable (খুব ছোট)

ফলে:

- Trainable parameter সংখ্যা হাজার গুণ কমে
- Performance প্রায় একই থাকে

LoRA কেন এত জনপ্রিয়?

- Easy to implement
- Memory efficient
- এক base model-এ বহু LoRA adapter লাগানো যায়



৭) Adapter Tuning

Adapter কী?

Adapter হলো ছোট neural network block যা:

- Transformer layer-এর ভেতরে বসানো হয়
- Base weights freeze থাকে
- শুধু adapter layers train হয়

স্ট্রাকচার:

Transformer Layer
 ├─ Attention
 ├─ Adapter (small FFN) ← trainable
 └─ FFN

একেকটা কাজের জন্য একেকটা adapter রাখা যায়।

Adapter-এর সুবিধা

- Task isolation (এক কাজ অন্য কাজ নষ্ট করে না)
- Multiple domain support
- Research-friendly

৮) Prefix / Prompt Tuning

Prefix Tuning কী?

এখানে:

- Model weight বদলায় না
- ইনপুটের শুরুতে কিছু **learnable virtual tokens** যোগ করা হয়

এই token গুলো:

- মডেলকে বলে দেয় “কীভাবে আচরণ করতে হবে”

একে বলা যায়:

> “Soft prompt, hard learning”

Prompt Tuning বনাম **Prefix Tuning**

- Prompt tuning: শুধু input embedding
- Prefix tuning: attention layers-এও প্রভাব ফেলে

৯) PEFT কবে ব্যবহার করবে?

PEFT ব্যবহার করো যদি:

- GPU limited
- Multiple tasks/domains
- Base model ভালো
- Behaviour/style শেখাতে চাও

PEFT যথেষ্ট নয় যদি:

- একদম নতুন জ্ঞান শেখাতে চাও
- Base model খুব দুর্বল
- Fundamental architecture বদল দরকার

তখন RAG বা full retraining দরকার।

১০) PEFT + RAG: ইন্ডাস্ট্রির সবচেয়ে জনপ্রিয় কম্বিনেশন

বাস্তবে অনেক সিস্টেম এমন:

- **PEFT (LoRA)** → tone, format, instruction habit
- **RAG** → up-to-date / private knowledge

ফলে:

- Model ছোট থাকে
- Knowledge fresh থাকে
- Cost কম থাকে