UNA SERIE SULL'INTELLIGENZA ARTIFICIALE

Io, Gemini e altri quattro

Parte VIII

Il Fine-Tuning: Personalizzare il DNA di un Modello

ROCCO MASSIMILIANO MONDO

ABSTRACT

*Nell' episodio VII abbiamo scoperto il **RAG**: la tecnica che inietta conoscenza privata nell'LLM al momento dell'inferenza, senza toccare il modello base. Il RAG è potente, agile e immediatamente praticabile. Ma esiste un livello di personalizzazione più profondo: il **fine-tuning**, la tecnica che **modifica i pesi stessi** del modello per specializzarne il comportamento, il tono e il ragionamento su un dominio tecnico specifico. In questo ottavo episodio **analizziamo il fine-tuning** dalla teoria alla pratica: cosa sono **LoRA** e **QLoRA**, come si preparano i dati di addestramento nel formato **JSONL**, quali strumenti open source esistono oggi (**Unsloth**, **LLaMA Factory**, **Axolotl**), che tipo di hardware è effettivamente necessario e quando questo approccio conviene rispetto al RAG. Vedremo un caso d'uso concreto per comprendere meglio. Specializziamo un modello locale sulle normative CEI/UNI per creare un assistente tecnico di dominio pienamente privato e offline.*

Rocco M. Mondo, Agosto 2026

PARTE VIII

Il Fine-Tuning: Personalizzare il DNA di un Modello

PREMESSA: IL COMPLEMENTO DEL RAG

Nella settima puntata di questa serie abbiamo parlato del sistema ***RAG***: un assistente che interroga un database vettoriale di documenti tecnici e risponde citando le fonti. È uno strumento potente, immediatamente applicabile, con costi operativi minimi. Eppure al termine di quella puntata avevo lasciato aperta una questione, intenzionalmente, e cioè se la combinazione più potente è effettivamente quella del ***RAG + fine-tuning***, quindi un modello specializzato sul dominio (attraverso fine-tuning) che ha anche accesso ai documenti aggiornati (attraverso RAG)".

Il fine-tuning è l'altra faccia della personalizzazione dell'IA. Non si tratta di fornire informazioni al modello nel momento in cui risponde: si tratta di ***modificare strutturalmente i pesi del modello,*** il suo ***DNA matematico***, per specializzarne il comportamento in modo permanente.

La distinzione merita un'analogia concreta:

- Il ***RAG*** è come dare al medico il manuale aggiornato durante la visita.
- Il ***fine-tuning*** è come iscrivere il medico a una specializzazione biennale.

Entrambi hanno il loro posto. Capire quando usare l'uno e quando l'altro è la competenza che distingue chi usa l'IA superficialmente da chi la integra strategicamente nel proprio processo professionale e produttivo.

COS'È IL FINE-TUNING: ANATOMIA TECNICA

Il ciclo di vita di un LLM moderno si articola in ***tre fasi distinte***, ognuna con obiettivi e costi computazionali radicalmente diversi.

Fase	Nome tecnico	Obiettivo	Costo computazionale
1	Pre-training	Imparare il linguaggio da trilioni di token	***Enorme*** — settimane su cluster GPU
2	Instruction Tuning (SFT)	Imparare a seguire istruzioni in modo coerente	***Alto*** — giorni su GPU professionale
3	Fine-Tuning personalizzato	Specializzarsi su un dominio tecnico specifico	***Moderato*** — ore/gior su GPU consumer

PARTE VIII

Il Fine-Tuning: Personalizzare il DNA di un Modello

La fase di nostro interesse, quella che è possibile realizzare con dei mezzi quasi standard, è la terza fase.Quindi partiremo da un modello già addestrato e istruito — come ***LLaMA 3.3***, ***Mistral, Phi-4 o Gemma 4*** e lo ***addestreremo ulteriormente su un dataset*** di esempi specifici del nostro dominio di interesse, con un costo computazionale accessibile.

Il meccanismo: back-propagation sui propri dati

Durante il fine-tuning, il modello elabora gli esempi del nostro ***dataset*** e aggiorna i propri ***pesi*** (***weights***) attraverso il processo di ***back-propagation***, lo stesso algoritmo usato nel pre-training, ma su scala molto minore e con obiettivi molto più specifici. Il risultato è un modello che ragiona secondo i pattern del vostro dominio, usa la terminologia tecnica corretta senza doverla specificare nel prompt e produce output nel ***formato*** e nel ***tono*** che avete mostrato negli esempi; è meno incline ad ***allucinare*** su argomenti per cui ha ricevuto esempi corretti e validati da un esperto.

Il concetto di "base model" e "adapter"

Un'analogia utile: ***immaginate il modello base*** come il sistema operativo del computer, e il ***fine-tuning*** come l'installazione di un'applicazione specializzata. Il sistema operativo non cambia nella sua struttura fondamentale, ma il comportamento del sistema in quel dominio specifico diventa radicalmente più preciso e affidabile.

IL RISCHIO: IL CATASTROPHIC FORGETTING

Prima di procedere, è doveroso affrontare un argomento di fondamentale importanza, ossia il ***limite principale del fine-tuning tradizionale***: il ***catastrophic forgetting*** (dimenticanza catastrofica). Quando si addestra un modello su un ***dataset specifico***, i pesi vengono modificati per ottimizzare le prestazioni su quel dominio. Questa ottimizzazione può ***degradare*** le prestazioni su altri compiti, perché ***il modello "dimentica"*** ciò che aveva imparato durante il pre-training per fare spazio alle nuove informazioni.

[Esempio pratico]
*Un modello a cui è stato applicato un processo di **fine-tuning aggressivo su normative CEI** in italiano potrebbe peggiorare nel ragionamento matematico generale, nella comprensione di altri contesti, o nella capacità di generare codice Python. **Il fine-tuning aggressivo** può anche limitare le capacità generali.*
*Questo è precisamente il motivo per cui le tecniche **PEFT** (Parameter-Efficient Fine-Tuning) come **LoRA** e **QLoRA** sono diventate lo standard del settore: permettono di specializzare il modello senza modificare la maggior parte dei pesi originali, preservando le capacità generali acquisite durante il **pre-training** di base.*

LE TECNICHE EFFICIENTI: LoRA E QLoRA

La svolta pratica nel fine-tuning è arrivata nel 2021 con il paper ***"LoRA: Low-Rank Adaptation of Large Language Models" (Microsoft Research)***. La tecnica ha reso il fine-tuning accessibile su hardware consumer, abbattendo i requisiti di memoria VRAM di un fattore 10–20x rispetto al ***full fine-tuning***.

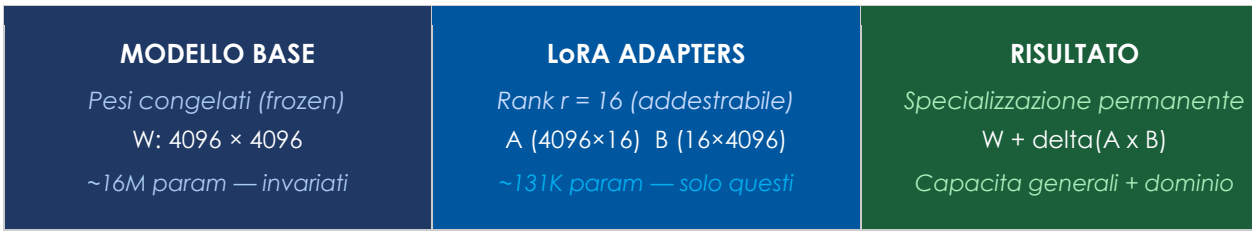
LoRA: Low-Rank Adaptation

Invece di modificare tutti i miliardi di pesi del modello, LoRA "***congela***" i pesi originali e introduce piccole matrici aggiuntive chiamate ***adapter in ogni layer critico***. Solo questi adapter vengono addestrati durante il fine-tuning.

La matematica di base (semplificata):

- **Matrice originale W (es. 4096 × 4096 = 16M parametri)** — congelata, non viene mai modificata
- **Due matrici adapter A (4096×r) e B (r×4096)** — solo queste vengono addestrate, con ***rank r*** tipicamente tra 4 e 64
- **Con r = 16:** solo 4096×16 + 16×4096 = 131.072 parametri, invece di 16M — riduzione del 99%

Il rank r controlla ***il trade-off*** tra capacità di adattamento e dimensione dell'adapter. Valori bassi (r = 8 o r = 16) sono sufficienti per la maggior parte dei casi di specializzazione tecnica su un dominio ben definito come quello normativo.



Schema concettuale: i pesi originali rimangono invariati, gli adapter LoRA aggiungono specializzazione.

QLoRA: la democratizzazione del fine-tuning

QLoRA (Dettmers et al., 2023) combina la ***quantizzazione a 4-bit del modello base*** con gli ***adapter LoRA***. Il modello base viene caricato in memoria in formato ***NF4*** (Normal Float 4-bit), occupando circa 1/4 della VRAM normale. Solo gli ***adapter LoRA*** vengono addestrati in piena precisione ***BF16***.

Il risultato pratico è straordinario: è possibile fare fine-tuning di un modello da 13B parametri con una singola GPU consumer da 24 GB di VRAM. Per modelli da 7B, bastano 10–12 GB.

Metodo	Parametri modificati	VRAM richiesta (mod. 70B)	Qualità risultante
Full Fine-Tuning	100% — ~70B parametri	~560 GB — non praticabile	Massima teorica
LoRA (r=16)	~0,1% — ~70M parametri	~40 GB — GPU professionale	Ottima
QLoRA 4-bit + LoRA (r=16)	~0,1% su base quantizzata	~24 GB — RTX 4090	Ottima / Molto buona
QLoRA 4-bit + LoRA (r=8) 13B	~0,05% — ~7M parametri	~12 GB — RTX 3060	Buona per dominio specifico

GLI STRUMENTI OPEN SOURCE

L'ecosistema del fine-tuning open source è maturato notevolmente negli ultimi due anni. Tre strumenti meritano attenzione particolare per chi vuole avvicinarsi a questa tecnica in modo pratico e operativo nel 2026.

Unsloth — La scelta per chi inizia

Unsloth è la libreria di riferimento per il fine-tuning efficiente con ***QLoRA***. I suoi vantaggi concreti: velocità di training 2–5x superiore rispetto a implementazioni naive; riduzione del 60–70% nel consumo di VRAM grazie a ***kernel CUDA*** ottimizzati manualmente; supporto nativo per LLaMA 3.3, Mistral, Gemma, Phi-4, Qwen2.5; interfaccia Python semplice, pienamente compatibile con ***l'ecosistema Hugging Face***. Installazione come libreria python: ***pip install unsloth***.

Filosofia: "***Fast, memory efficient, and easy-to-use finetuning***" — ideale per il professionista che vuole risultati senza immergersi nei dettagli dei kernel CUDA.

LLaMA Factory — Per chi preferisce la GUI

LLaMA Factory è un framework completo con interfaccia web (WebUI) che permette di avviare, monitorare e valutare sessioni di fine-tuning senza scrivere codice. Supporta oltre 100 modelli di base e include strumenti integrati per la valutazione post-training ***(benchmark MMLU, calcolo delle perdite, confronto modelli)***. La curva di apprendimento è bassa: ideale per team misti tecnici/non tecnici che vogliono prototipare rapidamente.

Axolotl — Per la massima configurabilità

Axolotl è lo strumento preferito da chi cerca massima flessibilità. Gestisce il fine-tuning tramite file di configurazione ***YAML*** dettagliati, supporta dataset multipli in formato eterogeneo, e ha ottime integrazioni ***per DeepSpeed e FSDP*** (training multi-GPU distribuito). Curva di apprendimento più alta, ma offre il controllo granulare necessario per ricerca avanzata.

Strumento	Curva apprendimento	Punto di forza	Ideale per
Unsloth	Bassa	Velocità ed efficienza VRAM ottimizzata	Chi inizia, hardware consumer limitato
LLaMA Factory	Bassa–Media	WebUI, ampia copertura modelli	Team misti, prototipazione rapida
Axolotl	Alta	Configurabilità totale, multi-GPU	Ricercatori, setup di produzione avanzati

I DATI: IL VERO CUORE DEL FINE-TUNING

Il principio fondamentale del ***machine learning*** vale nel ***fine-tuning*** con la stessa forza: un modello addestrato su dati di bassa qualità produce output di bassa qualità, indipendentemente dalla sofisticazione della tecnica. "***Garbage in, garbage out***" — vale qui più che altrove.

Il formato standard: JSONL con istruzioni

Il formato più diffuso per il fine-tuning a istruzioni è il JSONL (JSON Lines), dove ogni riga rappresenta un esempio di training nel formato istruzione-risposta. Il formato Alpaca-style, introdotto da Stanford nel 2023, è diventato lo standard de facto.

```
// Formato JSONL — Alpaca style (un oggetto JSON per riga)
{
  "instruction": "Qual e il valore di illuminamento minimo per uffici open space?",
  "input": "",
  "output": "Secondo la norma UNI EN 12464-1:2021, tabella 5.1, gli uffici open
             space richiedono Em >= 500 lx, uniformita U0 >= 0.60, UGR <= 19."
}
```

Esempio di coppia istruzione-risposta nel formato JSONL. Ogni riga del file è un esempio indipendente di training.

Quantità di dati: molto meno di quanto si pensi

Uno dei miti più diffusi sul fine-tuning è che servano migliaia di esempi. Per la specializzazione su un dominio tecnico ben definito, la realtà è molto più incoraggiante.

Obiettivo del fine-tuning	Esempi necessari	Note pratiche
Adattamento di stile e tono comunicativo	100–500	Il modello impara velocemente pattern stilistici
Conoscenza di dominio tecnico specifico	500–2.000	Copertura adeguata del dominio normativo
Comportamento molto specializzato	1.000–5.000	Richiede validazione rigorosa degli esempi
Sostituzione del pre-training	Non fattibile	In questo caso il RAG è la soluzione corretta

PARTE VIII

Il Fine-Tuning: Personalizzare il DNA di un Modello

[La qualità batte sempre la quantità]
Un dataset di 200 esempi perfettamente costruiti, verificati da un esperto di dominio, produce risultati superiori a 2.000 esempi generati automaticamente e non verificati.
Per un ingegnere elettrico: 300 coppie domanda-risposta precise sulle normative CEI, validate manualmente, valgono più di 3.000 esempi generici sull'elettrotecnica estratti dal web.
La validazione umana da parte dell'esperto di dominio non è una fase ottimizzabile: è il prerequisito per un sistema tecnicamente affidabile. Non esistono scorciatoie in questa fase.

L'HARDWARE: COSA SERVE DAVVERO

Il fine-tuning ha reputazione di essere costoso in termini di hardware. Con ***QLoRA***, la realtà è più accessibile di quanto sembri — anche se rimangono requisiti minimi significativi, specialmente per modelli di grandi dimensioni.

Modello base	VRAM minima	GPU consumer compatibile	Tempo tipico (1.000 steps)
7B parametri	10–12 GB	RTX 3060 12 GB / RTX 4070	30–60 min
13B parametri	18–24 GB	RTX 3090 24 GB / RTX 4090	1–3 ore
34B parametri	40–48 GB	2× RTX 4090 / NVIDIA A6000	4–10 ore
70B parametri	80–96 GB	4× NVIDIA A100 40 GB	12–24 ore

Il caso delle architetture a memoria unificata (Ryzen AI 395+ Max / Strix Halo)

Per chi sta valutando hardware locale con GPU integrata ad architettura unificata segnalo (ottimo rapporto prestazione/costi) il sistema ***AMD Ryzen AI Max+ 395*** (Strix Halo) con i suoi ***256 GB/s di bandwidth teorici e basato sulla tecnologia ZEN 5***. L'architettura di memoria unificata con ***96*** o ***128*** GB è teoricamente compatibile con QLoRA su modelli fino a 13–20B parametri. Il bandwidth non è ottimale per il training (che richiede throughput di scrittura più elevato rispetto all'inferenza), ma per sessioni di fine-tuning su dataset di 500–1.000 esempi il sistema è più che adeguato per uso professionale e sperimentazione approfondita.

Il cloud come alternativa economicamente razionale

Servizio cloud	Costo indicativo	Hardware disponibile	Ideale per
Google Colab Pro	~15 €/mese	T4, A100 (sessioni limitate)	Sperimentazione e test rapid
Lambda Labs	~1,5–3 €/ora	A100 40/80 GB, H100	Sessioni di training programmate
Vast.ai (spot)	~0,5–2 €/ora	RTX 4090, A100	Dataset piccoli, massimo risparmio

[Costo reale — un esempio concreto]
Fine-tuning QLoRA su modello 13B, dataset 1.000 esempi: training su Lambda Labs (A100 40 GB) a 1,5 €/ora × 2 ore = circa 3 euro totali.
Il costo computazionale è irrisorio. Il vero investimento è nella costruzione e validazione del dataset, non nell'hardware.

9

UN CASO PRATICO: SPECIALIZZAZIONE SU NORMATIVE CEI

Traduco ora i concetti in un workflow concreto: come si costruisce un ***modello fine-tunato sulle normative CEI*** per applicazioni di ingegneria elettrica e illuminotecnica. Descriverò le quattro fasi operative essenziali.

Fase 1 — Costruzione del dataset

Partendo dai testi ufficiali CEI 64-8, CEI EN 61439, UNI EN 12464-1 e dalle guide applicative CEI, il dataset viene costruito attraverso tre categorie di esempi, bilanciate in proporzione circa 40 / 40 / 20:

1. **Domande fattuali:** "Qual è la corrente minima per un differenziale tipo A in ambienti con carichi non lineari?"
2. **Ragionamento normativo:** "Un impianto fotovoltaico da 5 kW in utenza domestica richiede quale tipo di protezione differenziale secondo CEI 64-8?"
3. **Applicazioni progettuali:** "Calcola il numero di proiettori LED per un ufficio open space da 300 m² con Em = 500 lx, U=0.65, Fm=0.80, flusso unitario 12.000 lm."

Fase 2 — Validazione umana (non negoziabile)

Ogni risposta del dataset viene verificata manualmente da un professionista abilitato. Questa fase non è ottimizzabile né delegabile a un'altra IA: è il prerequisito per un sistema tecnicamente affidabile. Un esempio non verificato insegna al modello a sbagliare in modo plausibile — il rischio più pericoloso in ambito normativo.

Fase 3 — Training con Unsloth (configurazione tipica)

Parametro	Valore scelto	Motivazione
Modello base	LLaMA 3.3 8B Instruct	Bilanciamento qualità/dimensione per dominio tecnico
Dimensione dataset	500 esempi validati	Sufficiente per specializzazione normativa di nicchia
LoRA rank (r)	16	Trade-off capacità/dimensione adeguato al dominio specifico
LoRA alpha	32	Valore standard consolidato: alpha = 2 × rank
Epoche di training	3	Evita overfitting su dataset di dimensioni ridott
Learning rate	2e-4	Valore consolidato per QLoRA con libreria Unsloth
Hardware utilizzato	RTX 4090 24 GB	GPU consumer compatibile con 8B QLoRA
Durata stimata	~2 ore	Stima basata su hardware indicato e dataset definito

Fase 4 — Valutazione del modello fine-tunato

Il modello fine-tunato viene testato su un set di valutazione separato di 50 esempi non visti durante il training, e confrontato con il modello base identico su un set di domande normative CEI. I parametri di valutazione per un dominio tecnico normativo: accuratezza normativa (verificata dall'esperto di dominio), completezza della risposta, correttezza delle citazioni (numero norma, articolo, tabella di riferimento), assenza di allucinazioni su dati numerici critici.

FINE-TUNING + RAG: LA COMBINAZIONE VINCENTE

Il fine-tuning e il RAG non sono alternativi: sono complementari. Ognuno risolve un problema diverso, e la loro combinazione rappresenta l'architettura ottimale per un assistente tecnico professionale di alto livello.

Caratteristica	RAG (Parte VII)	Fine-Tuning (Parte VIII)
Problema risolto	Accesso a documenti aggiornati e privati	Comportamento e ragionamento specializzati
Modifica il modello?	No — il modello rimane invariato	Si — parzialmente (solo gli adapter LoRA
Aggiornabile?	Si — immediatamente (aggiungi un doc)	Solo con un nuovo ciclo di training
Cita le fonti?	Sempre — strutturale nel sistema	Solo se addestrato esplicitamente a farl
Costo operativo	Basso (storage + embedding model)	Una-tantum per il training, poi zero
Ideale per	Fatti, regole, dati aggiornati	Stile, ragionamento, terminologia di dominio

L'architettura ottimale per un assistente tecnico professionale è stratificata su due livelli:

1. **Livello base — Fine-Tuning:** un modello fine-tunato sul dominio (CEI, illuminotecnica, progettazione elettrica). Comprende la terminologia, ragiona secondo i pattern del dominio, produce output nel formato corretto. Questo strato cambia raramente.

2. **Livello superiore — RAG:** un sistema RAG con i documenti aggiornati. Conosce le ultime normative pubblicate, i calcoli progettuali recenti, le specifiche tecniche correnti. Questo strato si aggiorna continuamente e senza riaddestramento.

Il fine-tuning insegna al modello come pensare e come comunicare nel vostro dominio. Il RAG gli fornisce cosa sapere in quel momento specifico. Insieme, costruiscono un assistente che non è ottenibile con nessuno dei due approcci in isolamento.

CONCLUSIONE

Il fine-tuning era, fino a pochi anni fa, territorio esclusivo dei laboratori di ricerca con budget milionari. Oggi, grazie a ***QLoRA***, ***Unsloth*** e all'ecosistema open source maturo, è accessibile a chiunque disponga di una GPU consumer e di un dataset ben costruito. La democratizzazione tecnologica di cui parlavamo nella Parte I ha raggiunto anche questo livello avanzato della personalizzazione dei modelli linguistici.

Il fine-tuning non è la risposta a tutto. È uno strumento specifico per un problema specifico: specializzare il comportamento e il ragionamento di un modello su un dominio tecnico ben definito che funziona meglio in combinazione con un ecosistema RAG per i dati fattuali e aggiornati. ***La scelta tra i due approcci non è tecnica: è strategica.***

La vera sfida, come si è visto, non riguarda l'ambito tecnico ma la costruzione ***del dataset***. Cento esempi perfetti valgono più di mille esempi mediocri e la validazione umana da parte di un esperto nel dominio di interesse, un ingegnere che conosce davvero le normative CEI applicandole ogni giorno sul campo resta la componente più importante, quella che nessuna automazione può sostituire.

[Prossima puntata — Parte IX]

Con il prossimo episodio entreremo nella parte finale di questo affascinante percorso, quindi presenterò i frutti degli studi effettuati negli ultimi due anni sul mondo dell'intelligenza artificiale e delle possibilità che si sono prospettate nell'abbracciare questa nuova tecnologia. Il progetto ***A.D.A.M. (Adaptive Digital Assistant for Machine-learning)*** *non è un semplice LLM ma un "****Sistema AI ibrido"*** *con competenze nel campo STEM e specializzazione in fisica quantistica.* ***A.D.A.M.*** *non è il punto di arrivo ma quello di partenza per un progetto di più ampio respiro, le cui potenzialità possono aprire le porte ad un nuovo approccio allo studio e alla ricerca di confine, dandoci la possibilità di realizzare progetti che fino ad oggi avrebbero richiesto l'impegno di ingenti risorse materiali e decenni in termini di risorse temporali.*

Lo studio del campo della IA mi ha portato verso la conoscenza di nuove tecnologie e metodologie di lavoro che mi permettono di poter gestire un kologement molto vasto e soprattutto ben strutturato, finalizzato alla ricerca in campo ingegneristico. Prima di chiudere vorrei sottolineare che ***A.D.A.M. è un insieme di strumenti*** *applicati alla conoscenza che* ***convergono tutti insieme*** *verso un unico risultato, nuovi metodi per la ricerca scientifica non convenzionale.*

Rocco M. Mondo, Agosto 2026