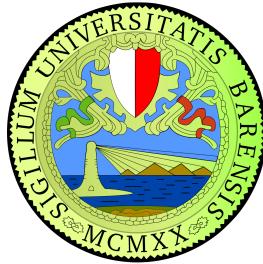


University of Bari Aldo Moro
Department of Computer Science

One-For-All: Un prober per riconoscerli tutti



Emanuele Fontana

December 1, 2025

Abstract

TODO: scrivere l'abstract

Contents

List of Figures

List of Tables

Chapter 1

Introduzione

1.1 Scopi e Obiettivi

Il rilascio di GPT-3 aperto al pubblico da parte di OpenAI ha segnato una svolta significativa nel NLP (Natural Language Processing), aprendo la strada a una nuova era di modelli di linguaggio di grandi dimensioni (LLM) in grado poter mantenere una conversazione in maniera fluida e naturale come una persona. Il rilascio del ChatBot è stato solo l'inizio: ora gli LLM sono utilizzati in una vasta gamma di applicazioni e sono ormai uno strumento irrinunciabile sia in ambito privato che accademico che industriale, il loro utilizzo è pressocchè sconfinato.

Tuttavia, uno dei principali ostacoli all'adozione diffusa e sicura di queste tecnologie in ambiti critici (e.g. medico, legale) è il fenomeno delle *allucinazioni*. Nonostante la coerenza sintattica, gli LLM tendono occasionalmente a generare informazioni fatualmente errate o inventate con un alto grado di confidenza. Questo comportamento deriva dalla natura stocastica degli LLM: questi ultimi non sanno contare o effettuare un ragionamento logico basato su regole e fatti: la loro abilità nel produrre il testo dipende esclusivamente dalle statistiche apprese durante la fase di addestramento su grandi corpora testuali. Si parla di *next token prediction*: un LLM dunque, dato il contesto (cioè l'insieme dei token precedenti), predice il token successivo più probabile in base alle distribuzioni apprese (possiamo pensare a un token come a una parola o parte di essa). Un esempio classico di allucinazione presente nei primissimi LLM è:

Utente: How many r's are there in the word strawberries?

LLM: 2

Attualmente, le tecniche per mitigare questo problema si basano spesso sulla verifica fattuale esterna (*retrieval-augmented generation*), sul *fine-tuning* specifico o sulle *CoT*: *Chain of Thought*, ma queste tecniche non sono infallibili ed è dunque cruciale capire quando un modello sta allucinando e quando no.

Il *probing* delle attivazioni interne (o *probing classifiers*) rappresenta una soluzione innovativa a questo problema. Questa metodologia consente di analizzare lo "stato interno" del modello durante la generazione, ipotizzando che l'informazione sulla veridicità

ità di un'affermazione sia codificata nello spazio latente dei neuroni, indipendentemente dall'output testuale finale il quale viene generato stocasticamente in quanto influenzato da molteplici fattori (temperature, top-k sampling, ecc.). Diversi studi recenti hanno dimostrato che è possibile addestrare classificatori lineari semplici per distinguere le affermazioni vere da quelle false osservando le attivazioni dei layer interni e che in genere le attivazioni relative alle allucinazioni e quelle relative alle risposte corrette tendono a essere facilmente separabili **TODO: citare paper**.

La maggior parte delle ricerche si è concentrata sulla creazione di probe specifici per un singolo modello o una specifica architettura, lasciando relativamente inesplorato il problema della trasferibilità. Questo lavoro si propone di studiare se esiste una rappresentazione universale delle allucinazioni nei LLM, indipendentemente dall'architettura sottostante. In particolare, si mira a sviluppare un *prober* generale capace di rilevare le allucinazioni in diversi modelli senza la necessità di un addestramento specifico per ciascuno di essi (al più con un allineamento preliminare tra gli spazi latenti, essendo questi ultimi diversi tra modelli differenti).

Gli obiettivi principali di questa progetto sono:

- Analizzare e confrontare le attivazioni interne di diverse famiglie di LLM (es. Qwen, Falcon) per identificare pattern comuni associati al fenomeno dell'allucinazione.
- Sviluppare e valutare metodologie di allineamento (lineari e non lineari) tra gli spazi latenti di un modello “insegnante” e un modello “studente”.
- Costruire un *prober* universale capace di trasferire la capacità di rilevamento delle allucinazioni da un modello all'altro senza supervisione aggiuntiva sul modello target.

1.2 Panoramica del documento

Il resto del progetto è strutturato come segue:

- **Capitolo 2: Background e Lavori Correlati** - Una panoramica delle tecniche esistenti per il rilevamento delle allucinazioni nei LLM.
- **Capitolo 3: Metodologia** - Descrizione dettagliata delle architetture utilizzate, delle tecniche di allineamento e del design del prober universale.
- **Capitolo 4: Esperimenti e Risultati** - Presentazione dei risultati ottenuti, analisi delle prestazioni del prober e discussione sui pattern comuni identificati.
- **Capitolo 5: Conclusioni e Lavori Futuri** - Sintesi dei contributi del progetto e suggerimenti per ricerche future in questo ambito.

Chapter 2

Background

In questo capitolo verranno introdotti i concetti di base necessari per comprendere il lavoro svolto. Verranno dunque presentati il concetto di Deep Learning, il concetto dell’attenzione, i Large Language Models (LLM), il fenomeno delle allucinazioni e il probing.

2.1 Deep Learning

Negli ultimi anni il *Deep Learning* ha rivoluzionato il campo dell’intelligenza artificiale, permettendo di raggiungere risultati straordinari in molteplici ambiti, tra cui il riconoscimento delle immagini, l’elaborazione del linguaggio naturale e i giochi. Il Deep Learning non è altro che una branche del Machine Learning (a sua volta branca dell’Intelligenza Artificiale) ed è basato sull’utilizzo di reti neurali profonde (Deep Neural Networks, DNN) per apprendere rappresentazioni complesse dei dati. La sostanziale differenza tra il Deep Learning e il Machine Learning tradizionale è nella rappresentazione dei dati da fornire all’algoritmo: nel Machine Learning tradizionale (Decision-Trees, SVM, etc.) è necessario fornire delle feature ingegnerizzate a mano, mentre nel Deep Learning le reti neurali sono in grado di apprendere automaticamente le rappresentazioni più utili per il compito da svolgere direttamente dai dati grezzi.

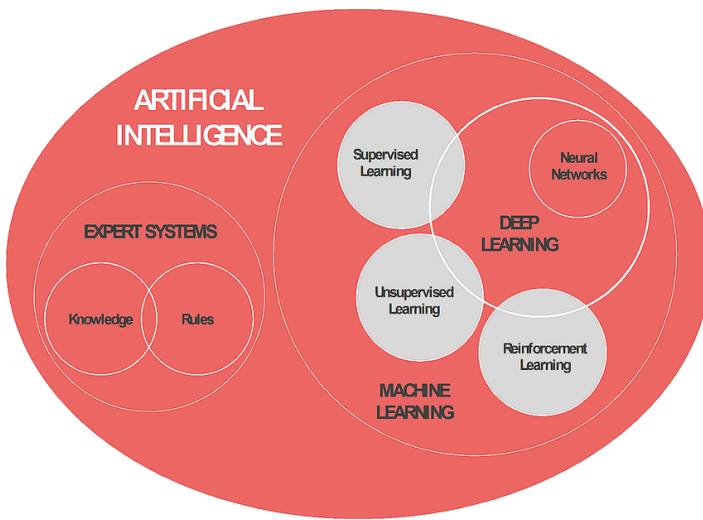


Figure 2.1: Rappresentazione delle branche dell’Intelligenza Artificiale

2.1.1 Cenni storici

Molti pensano che il Deep Learning sia una tecnologia estremamente recente, fantascienza fino a prima di ChatGPT. In realtà le basi teoriche del Deep Learning risalgono agli anni ’40 con i primi modelli come il Perceptron di Rosenblatt. Tuttavia a causa di limitazioni computazionali, teoriche (Vanishing Gradient Problem) e di disponibilità dei dati, il Deep Learning non ha potuto esprimere tutto il suo potenziale fino agli anni 2000. Con l’avvento delle Graphics Processing Units (GPU), nate per il rendering grafico e il gaming, è stato possibile accelerare notevolmente i calcoli necessari per l’addestramento delle reti neurali profonde. Le GPU, a differenza delle CPU, sono progettate per eseguire un set limitato di operazioni semplici in parallelo, rendendole ideali per le operazioni matriciali e vettoriali tipiche del Deep Learning. Inoltre, la disponibilità di grandi dataset, come ImageNet, ha permesso alle reti neurali di apprendere rappresentazioni più robuste e generalizzabili. Infine, i progressi teorici, come l’introduzione di nuove funzioni di attivazione (ReLU), tecniche di regolarizzazione (Dropout) e algoritmi di ottimizzazione (Adam), hanno ulteriormente migliorato le prestazioni delle reti neurali profonde.

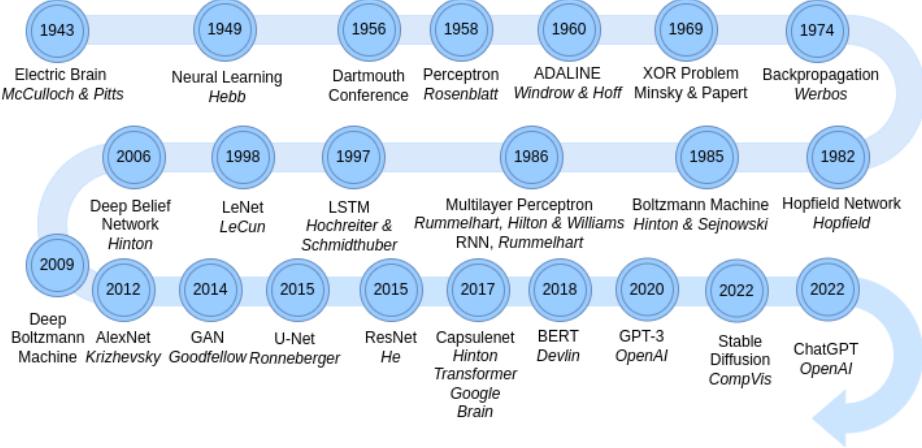


Figure 2.2: Timeline dei principali eventi storici nel campo del Deep Learning

2.1.2 Architettura e addestramento delle DNN

Le reti neurali profonde sono costituite da molti strati (layer) di neuroni artificiali collegati tra loro (come fossero neuroni di un cervello biologico). Il primo layer è l'input layer, che riceve i dati grezzi, mentre l'ultimo layer è l'output layer, che produce le predizioni finali: la configurazione dell'output layer dipende dal tipo di compito da svolgere (classificazione, regressione, etc.). Tra l'input e l'output layer ci sono diversi hidden layer, che trasformano progressivamente i dati attraverso operazioni lineari (pesi e bias) e non lineari (funzioni di attivazione come ReLU, Sigmoid, Tanh). L'addestramento delle DNN avviene in 2 fasi principali:

- **Forward propagation:** i dati di input vengono propagati attraverso la rete, calcolando le attivazioni di ogni neurone fino a produrre l'output finale. Questo output viene confrontato con il valore target (ground truth) utilizzando una funzione di perdita (loss function) che misura l'errore della predizione. I dati scorrono dunque dall'input layer all'output layer.
- **Backward propagation:** utilizzando l'algoritmo di backpropagation, l'errore calcolato viene propagato all'indietro attraverso la rete. Utilizzando la regola della catena, si calcolano i gradienti rispetto alla loss function e questi vengono propagati all'indietro per aggiornare i pesi e i bias della rete utilizzando un algoritmo di ottimizzazione (come Stochastic Gradient Descent, Adam, etc.). I dati scorrono dunque dall'output layer all'input layer.

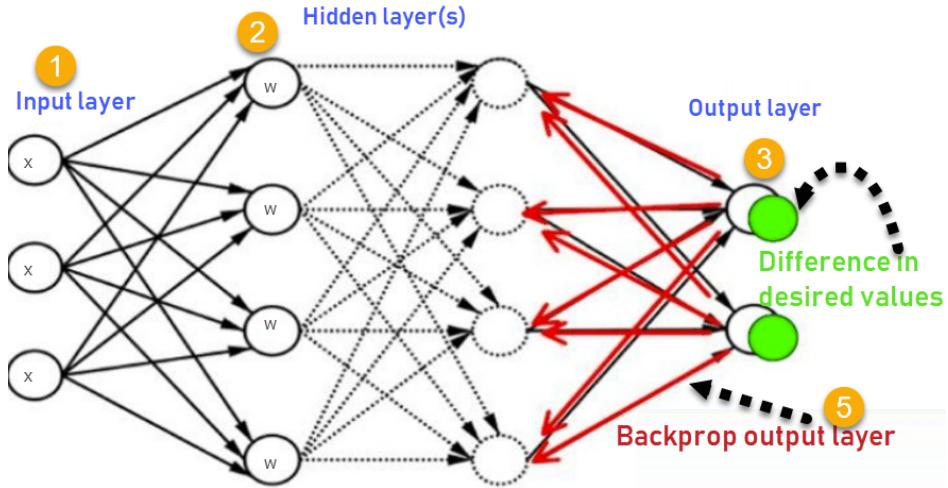


Figure 2.3: Rappresentazione della forward e backward propagation in una rete neurale

2.2 Panoramica sui Transformer

Il meccanismo della self-attention è stato introdotto per la prima volta nel 2017 da Vaswani et al. nel paper "Attention is All You Need" [TODO:cita](#) ed è alla base dell'architettura Transformer, i quali hanno rivoluzionato il campo dell'elaborazione del linguaggio naturale (NLP) e non solo, grazie alla loro capacità di catturare relazioni a lungo raggio nei dati sequenziali in modo più efficiente rispetto ai modelli precedenti come RNN e loro varianti. La differenza cruciale tra i transformer e i modelli precedenti risiede nell'elaborazione dell'input: le RNN elaborano i dati in modo sequenziale, mantenendo uno stato nascosto che cattura le informazioni precedenti, mentre i transformer utilizzano il meccanismo di self-attention per elaborare l'intera sequenza contemporaneamente, permettendo di catturare relazioni tra parole indipendentemente dalla loro distanza nella sequenza. Con un input molto lungo le RNN tendono a dimenticare le informazioni iniziali mentre i transformer non hanno questo problema.

2.2.1 Architettura dei Transformer

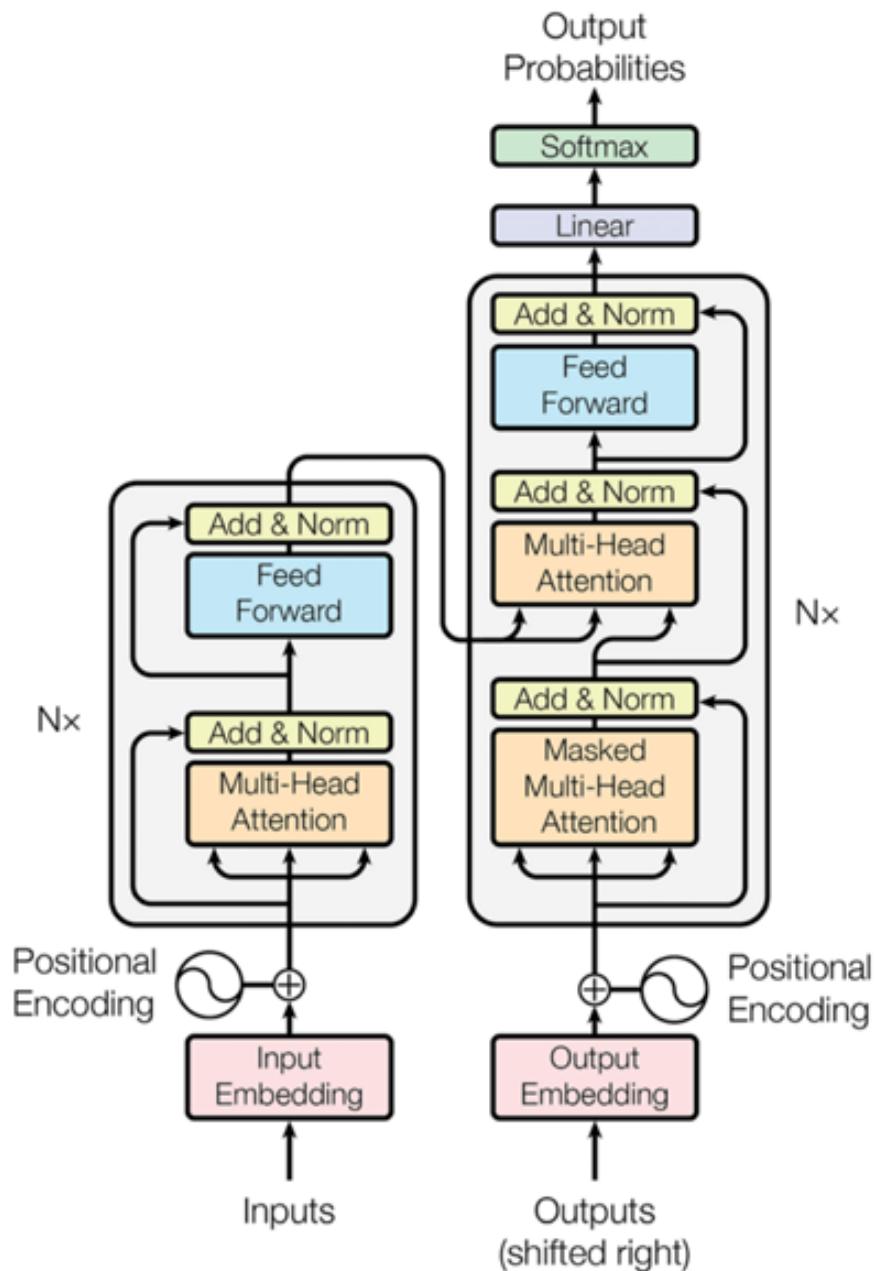


Figure 2.4: Architettura di un Transformer

Un transformer è composto da due blocchi principali: *Encoder* e *Decoder*.

Encoder

L'encoder è costituito da una serie di strati identici posti in sequenza, ciascuno dei quali contiene:

- **Multi-Head Self-Attention:** questo meccanismo consente al modello di focalizzarsi su diverse parti della sequenza di input contemporaneamente, catturando relazioni complesse tra le parole.
- **Layer Normalization e Residual Connections:** per migliorare la stabilità dell'addestramento e facilitare il flusso del gradiente, ogni sottostrato è seguito da una normalizzazione e da una connessione residua.
- **Feed-Forward Neural Network (FFNN):** ogni strato dell'encoder contiene una rete neurale feed-forward completamente connessa che elabora ulteriormente le rappresentazioni ottenute dalla self-attention.
- Un altro strato di Layer Normalization e Residual Connections.

Il ruolo principale dell'encoder è quello di trasformare la sequenza di input in una rappresentazione interna (embedding) che cattura le relazioni semantiche tra le parole.

Decoder

Il decoder è anch'esso costituito da una serie di strati identici posti in sequenza, ciascuno dei quali contiene:

- **Masked Multi-Head Self-Attention:** simile al meccanismo nell'encoder, ma con una maschera che impedisce al modello di "guardare avanti" nella sequenza di output durante l'addestramento, garantendo che la generazione delle parole avvenga in modo autoregressivo.
- **Layer Normalization e Residual Connections:** analogamente all'encoder, per migliorare la stabilità dell'addestramento.
- **Multi-Head Attention sull'Output dell'Encoder:** questo meccanismo consente al decoder di focalizzarsi sulle parti rilevanti della rappresentazione dell'input generata dall'encoder.
- Un altro strato di Layer Normalization e Residual Connections.
- **Feed-Forward Neural Network (FFNN):** come nell'encoder, per elaborare ulteriormente le rappresentazioni.
- Un ulteriore strato di Layer Normalization e Residual Connections.

2.2.2 Meccanismo di Self-Attention

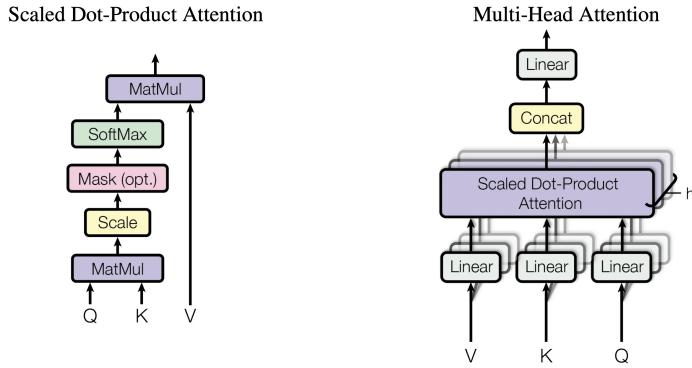


Figure 2.5: Rappresentazione del meccanismo di Self-Attention

Il meccanismo di self-attention consente al modello di pesare l'importanza delle diverse parole nella sequenza di input quando si elabora una particolare parola. Il calcolo della self-attention avviene attraverso tre matrici distinte: Query (Q), Key (K) e Value (V). La formula per calcolare l'attenzione è la seguente:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (2.1)$$

dove d_k è la dimensione delle chiavi. La softmax normalizza i punteggi di attenzione, permettendo al modello di concentrarsi sulle parole più rilevanti. Il meccanismo di multi-head attention estende questo concetto eseguendo più calcoli di attenzione in parallelo, permettendo al modello di catturare diverse tipologie relazioni tra le parole.

2.3 Large Language Models (LLM)

I Large Language Models (LLM) rappresentano l'evoluzione naturale delle architetture Transformer applicate su scala massiva. Un LLM è essenzialmente un modello probabilistico addestrato su enormi corpora di testo (nell'ordine di trilioni di token) con l'obiettivo fondamentale di prevedere il token successivo in una sequenza, dato il contesto precedente. Formalmente, data una sequenza di token $x = (x_1, x_2, \dots, x_t)$, il modello cerca di stimare la distribuzione di probabilità condizionata $P(x_{t+1}|x_1, \dots, x_t)$.

L'evoluzione dagli scorsi modelli di NLP agli attuali LLM (come la serie GPT, Llama, e Claude) è caratterizzata dal fenomeno dello *scaling law* **TODO: citare Kaplan et al.**, secondo cui le prestazioni del modello migliorano in modo prevedibile all'aumentare del numero di parametri, della quantità di dati di addestramento e della potenza di calcolo.

Il ciclo di vita tipico di un LLM si divide in fasi distinte:

1. **Pre-training:** È la fase più costosa computazionalmente. Il modello viene addestrato in modo auto-supervisionato su vasti dataset (web crawl, libri, codice) per apprendere la struttura del linguaggio, la logica e una vasta gamma di conoscenze generali. L'obiettivo è minimizzare la *Cross-Entropy Loss* tra la distribuzione prevista e il token reale successivo.
2. **Supervised Fine-Tuning (SFT):** Il modello pre-addestrato (detto "Base") viene ulteriormente addestrato su un dataset più piccolo e curato di istruzioni e risposte (formato prompt-response). Questo permette al modello di imparare a seguire le istruzioni dell'utente e di adottare un formato conversazionale.
3. **Alignment (RLHF/DPO):** Per garantire che il modello sia sicuro e allineato con i valori umani, vengono applicate tecniche come il *Reinforcement Learning from Human Feedback* (RLHF) o la *Direct Preference Optimization* (DPO). In questa fase, il modello viene penalizzato se genera contenuti tossici, bias o non utili.

Una caratteristica emergente degli LLM è la capacità di *In-Context Learning*, ovvero la capacità di apprendere nuovi task semplicemente osservando alcuni esempi nel prompt, senza aggiornare i pesi del modello.

2.4 Allucinazioni nei LLM

Nonostante le straordinarie capacità generative, gli LLM soffrono di una limitazione critica nota come "allucinazione". Nel contesto dell'intelligenza artificiale, un'allucinazione si verifica quando un modello genera un output che è sintatticamente corretto e fluido, ma fattualmente errato, logicamente incoerente o non fedele alla sorgente di input fornita.

Le allucinazioni possono essere categorizzate principalmente in due tipologie:

- **Allucinazioni di Fattualità (Factuality Hallucinations):** Il modello genera informazioni che contraddicono la conoscenza del mondo reale. Un esempio tipico è l'attribuzione di una citazione alla persona sbagliata o l'invenzione di eventi storici mai accaduti. Questo fenomeno è spesso legato alla natura probabilistica del modello, che tende a completare pattern linguistici piuttosto che recuperare fatti deterministici.
- **Allucinazioni di Fedeltà (Faithfulness Hallucinations):** Si verificano in task come la summarization o la Question Answering basata su documenti (RAG), quando il modello genera informazioni che non sono presenti nel testo sorgente o le contraddicono, pur essendo magari vere nel mondo reale (allucinazione estrinseca) o false (allucinazione intrinseca).

Le cause delle allucinazioni sono molteplici e includono: la compressione con perdita della conoscenza durante il training, dati di addestramento rumorosi o contraddittori,

e la tendenza del modello a privilegiare la fluidità del testo rispetto all'accuratezza fattuale (spesso definita come *sycophancy*, ovvero la tendenza a compiacere l'utente confermando i suoi bias). Rilevare e mitigare le allucinazioni è attualmente una delle sfide più aperte nella ricerca sugli LLM.

2.5 Probing

Il *Probing* (o *Probing Classifiers*) è una tecnica di analisi utilizzata nel campo dell'*Explainable AI* (XAI) per interpretare le rappresentazioni interne delle reti neurali profonde. L'idea fondamentale è che, sebbene gli LLM siano spesso considerati delle “black box”, le loro attivazioni interne (i valori degli hidden states e degli output dei vari moduli a ogni layer) contengano informazioni ricche e strutturate riguardanti l'input elaborato.

La metodologia standard di probing prevede i seguenti passi:

1. **Estrazione delle attivazioni:** Si fornisce un input al modello (frozen, ovvero con i pesi congelati) e si registrano i vettori di attivazione h_l generati in uno specifico layer l .
2. **Addestramento del Prober:** Si addestra un classificatore supervisionato semplice (spesso una Regressione Lineare o un Perceptron Multistrato semplice) che prende in input le attivazioni h_l e cerca di predire una specifica proprietà y dell'input (ad esempio: la parte del discorso, la lunghezza della frase, o, nel caso di questo lavoro, la veridicità dell'affermazione).
3. **Valutazione:** Le prestazioni del classificatore indicano quanto l'informazione relativa alla proprietà y sia codificata linearmente (o non linearmente) in quel particolare layer del modello.

Nel contesto delle allucinazioni, il probing viene utilizzato per investigare se esiste una “direzione della verità” nello spazio latente del modello. L'ipotesi è che il modello, internamente, “sappia” se sta generando un fatto vero o un'allucinazione, e che questa informazione possa essere decodificata analizzando gli stati interni prima che venga generato l'output testuale finale.

2.6 Large Language Models utilizzati

Per gli esperimenti condotti in questo lavoro, sono stati selezionati due modelli open-weights che rappresentano lo stato dell'arte per le rispettive dimensioni e famiglie architetturali.

2.6.1 Qwen2.5-7B

Qwen2.5-7B è un modello sviluppato da Alibaba Cloud ed è parte della serie Qwen2.5. È un modello *Decoder-only* basato su Transformer con 7 miliardi di parametri. Rispetto ai suoi predecessori, Qwen2.5 introduce diverse ottimizzazioni architetturali:

- **Grouped Query Attention (GQA)**: Utilizza GQA invece della standard Multi-Head Attention per ottimizzare l'uso della memoria della cache KV durante l'inferenza, permettendo contesti più lunghi e una generazione più veloce.
- **SwiGLU Activation**: Sostituisce la classica funzione di attivazione GeLU con SwiGLU, che ha dimostrato empiricamente di migliorare le prestazioni di convergenza.
- **RoPE (Rotary Positional Embeddings)**: Utilizza embeddings posizionali rotativi per gestire meglio le posizioni relative dei token e supportare finestre di contesto molto ampie (fino a 128k token).

Il modello è stato pre-addestrato su un corpus massivo e multilingua, dimostrando eccellenti capacità non solo nel ragionamento e nella comprensione del linguaggio, ma anche nel coding e nella matematica, spesso superando modelli di dimensioni simili come Llama-3-8B in vari benchmark.

2.6.2 Falcon3-7B-Base

Falcon3-7B-Base è l'ultima iterazione della famiglia di modelli sviluppati dal Technology Innovation Institute (TII) di Abu Dhabi. Come Qwen, è un modello *causal decoder-only* da 7 miliardi di parametri, ma si distingue per scelte progettuali orientate all'efficienza. Falcon3 continua la tradizione della serie Falcon concentrandosi sulla qualità dei dati di pre-training (basati sul dataset RefinedWeb, un dataset web rigorosamente filtrato e deduplicato). A livello architettonico, Falcon3-7B adotta soluzioni moderne per massimizzare il throughput in inferenza. Sebbene le specifiche esatte possano variare tra le versioni (es. l'uso di *Multi-Query Attention* o *FlashAttention*), l'obiettivo principale di Falcon è fornire un modello performante che sia facilmente deployabile su hardware di consumo. Nel contesto di questo lavoro, Falcon3 viene utilizzato come termine di paragone (o modello “studente”) per verificare la generalizzabilità dei proverbi addestrati su architetture diverse come Qwen.

Chapter 3

Methodology

3.1 Introduzione

In questo capitolo viene esplorata la metodologia di lavoro adottata per lo sviluppo del progetto. Per la costruzione del prober universale è stato fatto uso del dataset “Belief Bank” [TODO:citazione](#), che fornisce un insieme di affermazioni e la loro veridicità in diversi contesti. Sono stati presi in considerazione cinque diversi approcci metodologici per allineare lo spazio latente di un modello “studente” a quello di un modello “insegnante”. La prima fase del lavoro ha riguardato la preparazione del dataset e l'estrazione delle attivazioni interne dei modelli Qwen2.5-7B e Falcon3-7B-Base su tutti i layer e ogni tipologia di componente di un layer (attention, MLP, hidden states). Successivamente sono stati effettuati studi preliminari per valutare la capacità di ogni singolo componente di layer di discriminare tra affermazioni vere e allucinazioni, al fine di identificare i layer e le componenti più rilevanti per la costruzione del prober universale. Sono state dunque calcolate alcune statistiche relative alle allucinazioni dei due modelli e tutte le attivazioni sono state preprocessate per essere utilizzate nei vari approcci metodologici. In una prima fase è stato usato un approccio completamente lineare come baseline, per poi esplorare metodi più complessi che includono adattamenti non lineari tramite AdapterMLP, Encoder e AutoEncoder. Ogni metodologia è stata valutata in termini di prestazioni e capacità di trasferimento tra modelli diversi.

3.2 Dataset

Il lavoro si basa su *BeliefBank* [TODO: cita](#), un dataset strutturato progettato per analizzare la consistenza e l'accuratezza delle credenze nei modelli linguistici pre-addestrati.

3.2.1 BeliefBank

Struttura del Dataset Originale Nella sua formulazione originale, BeliefBank si compone di due elementi distinti ma interconnessi:

- **Vincoli (Constraints):** Un insieme di regole logiche formali, comprendenti implicazioni positive (es. $A \implies B$) e mutue esclusioni (es. $A \implies \neg B$). Questi vincoli sono stati estratti semi-automaticamente da knowledge base esterne quali ConceptNet e WordNet **TODO: cita**.
- **Fatti (Facts):** Un corpus di asserzioni vere o false generate istanziando i vincoli su un insieme di 85 entità (animali e piante). Le etichette di verità (“silver labels”) sono state ottenute annotando manualmente i nodi foglia del grafo e propagando la verità attraverso la rete dei vincoli.

Pre-elaborazione e Implementazione Per l'utilizzo nel progetto, i dati grezzi sono stati processati attraverso una pipeline dedicata che gestisce la trasformazione da rappresentazioni simboliche a linguaggio naturale e il bilanciamento delle classi. I dati originali sono archiviati sotto forma di triple strutturate (soggetto, relazione, oggetto). La pipeline implementa un modulo di conversione che utilizza template predefiniti per trasformare queste triple in frasi di senso compiuto in lingua inglese. Per prevenire bias verso una specifica classe di risposta, il dataset viene esteso tramite una procedura di *negation augmentation* generando sinteticamente la controparte negata di ogni fatto o implicazione presente, invertendo la label di verità associata. Questo approccio raddoppia efficacemente la dimensione del dataset e assicura che il modello sia esposto a una distribuzione bilanciata di etichette positive (yes) e negative (no). Ad ogni esempio è associato un identificativo univoco (`instance_id`) per facilitare il tracciamento e l'analisi dei risultati. Esempio di elementi del dataset dopo la pre-elaborazione:

- `instance_id`: 38
- **Question:** An albatross has a talon
- **Answer:** Yes

3.2.2 Creazione dataset di attivazioni

Una volta preparato il dataset di affermazioni e relative etichette di veridicità, il passo successivo è stato l'estrazione delle attivazioni interne dai modelli Qwen2.5-7B e Falcon3-7B-Base. Ogni esempio è preceduto da un prompt che viene fornito al modello di linguaggio per guidarne la risposta. *Answer the following question with just the essential information, without explanations* Quindi il modello di linguaggio riceve in input il prompt seguito dalla domanda e deve rispondere con “yes” o “no”. Per ogni affermazione nel dataset, sono state registrate le attivazioni corrispondenti a tutti i layer e componenti (attention, MLP, hidden states) dei modelli. Queste attivazioni costituiscono la base dati su cui sono stati addestrati i vari prober per il rilevamento delle allucinazioni. Per verificare se il modello risponde correttamente o allucina è stata usata una semplice logica di matching tra la risposta generata e l'etichetta di veridicità

associata all'affermazione nel dataset: se la risposta reale (“yes” o “no”) è una substring (cioè è contenuta) nella risposta generata dal modello, allora l'affermazione è considerata vera (non allucinazione); altrimenti è considerata falsa (allucinazione).

3.3 Studi preliminari

3.3.1 Statistiche sulle allucinazioni

Prima di procedere con la costruzione del prober universale, sono state calcolate alcune statistiche relative alla frequenza delle allucinazioni nei modelli Qwen2.5-7B e Falcon3-7B-Base sul dataset Belief Bank. Analizzando l'output dei modelli sul dataset Belief Bank, sono state calcolate le seguenti statistiche relative alle allucinazioni:

- **Qwen2.5-7B**: su un totale di 27416 affermazioni, il modello ha generato 15728 allucinazioni
- **Falcon3-7B-Base**: su un totale di 27416 affermazioni, il modello ha generato 16499 allucinazioni
- **Allucinazioni comuni**: i due modelli condividono 12599 allucinazioni.

I risultati suggeriscono che i modelli si comportano in modo molto simile, il che indica che le allucinazioni potrebbero essere legate a caratteristiche intrinseche nei modelli di linguaggio (e dunque la costruzione di un prober universale potrebbe essere possibile).

3.3.2 Studio delle singole componenti di layer

Per valutare la capacità discriminativa delle diverse componenti di layer (attention, MLP, hidden states) dei modelli Qwen2.5-7B e Falcon3-7B-Base nel rilevamento delle allucinazioni, sono stati condotti esperimenti sistematici su ciascun layer e tipologia di attivazione. In particolare, per ogni layer e componente, sono state estratte le attivazioni corrispondenti e utilizzate come input per addestrare un classificatore Logistic Regression, con l'obiettivo di distinguere tra affermazioni vere e allucinazioni.

La pipeline sperimentale prevede:

- Estrazione delle attivazioni per ogni layer e componente (attention, MLP, hidden) dai modelli su tutto il dataset.
- Suddivisione del dataset in training e test set, mantenendo la stessa suddivisione per tutti gli esperimenti.
- Normalizzazione delle attivazioni tramite StandardScaler.
- Addestramento di un classificatore Logistic Regression per ciascuna combinazione layer/componente.

- Valutazione delle prestazioni tramite metriche di accuratezza e F1-score.

- Ordinamento dei layer in base alle performance ottenute, per identificare quelli più rilevanti.

I risultati mostrano che alcune componenti e layer sono particolarmente efficaci nel discriminare le allucinazioni, evidenziando la presenza di pattern informativi specifici nelle attivazioni interne dei modelli. Tutti i risultati sono stati salvati e ordinati in formato JSON per facilitare l'analisi comparativa tra modelli e componenti.

Table 3.1: Risultati completi per Qwen2.5-7B (ordinati per performance decrescente)

Rank	Attention (Attn)			MLP			Hidden		
	Lyr	Acc	F1	Lyr	Acc	F1	Lyr	Acc	F1
1	16	0.9866	0.9885	20	0.9886	0.9902	18	0.9904	0.9918
2	18	0.9861	0.9881	16	0.9884	0.9901	20	0.9894	0.9909
3	19	0.9859	0.9879	18	0.9883	0.9900	19	0.9892	0.9907
4	15	0.9858	0.9878	19	0.9878	0.9896	16	0.9883	0.9900
5	21	0.9850	0.9872	21	0.9875	0.9893	21	0.9882	0.9899
6	11	0.9847	0.9868	12	0.9872	0.9891	17	0.9877	0.9895
7	12	0.9846	0.9867	17	0.9871	0.9889	22	0.9875	0.9893
8	10	0.9841	0.9863	14	0.9861	0.9881	23	0.9874	0.9891
9	8	0.9838	0.9861	15	0.9861	0.9881	14	0.9871	0.9889
10	17	0.9838	0.9861	10	0.9860	0.9880	15	0.9871	0.9889
11	13	0.9832	0.9856	9	0.9855	0.9876	24	0.9869	0.9887
12	14	0.9831	0.9855	11	0.9853	0.9874	13	0.9864	0.9883
13	20	0.9831	0.9855	13	0.9853	0.9874	11	0.9863	0.9882
14	25	0.9830	0.9854	8	0.9850	0.9872	12	0.9863	0.9882
15	6	0.9825	0.9849	22	0.9850	0.9872	25	0.9860	0.9880
16	3	0.9822	0.9848	23	0.9848	0.9869	7	0.9859	0.9879
17	24	0.9820	0.9845	6	0.9843	0.9865	26	0.9859	0.9879
18	22	0.9815	0.9841	7	0.9835	0.9858	10	0.9852	0.9873
19	23	0.9808	0.9835	24	0.9835	0.9858	9	0.9847	0.9869
20	7	0.9801	0.9829	27	0.9832	0.9856	27	0.9847	0.9868
21	26	0.9799	0.9828	25	0.9831	0.9855	6	0.9840	0.9862
22	4	0.9795	0.9823	26	0.9819	0.9845	8	0.9840	0.9862
23	9	0.9793	0.9823	4	0.9808	0.9835	5	0.9829	0.9853
24	5	0.9782	0.9813	5	0.9791	0.9820	4	0.9824	0.9849
25	2	0.9774	0.9806	3	0.9776	0.9807	3	0.9809	0.9836
26	27	0.9770	0.9803	0	0.9734	0.9771	2	0.9780	0.9811
27	1	0.9731	0.9769	2	0.9712	0.9752	0	0.9743	0.9779
28	0	0.9725	0.9764	1	0.9653	0.9701	1	0.9736	0.9773

Table 3.2: Risultati completi per Falcon3-7B-Base (ordinati per performance decrescente)

Rank	Attention (Attn)			MLP			Hidden		
	Lyr	Acc	F1	Lyr	Acc	F1	Lyr	Acc	F1
1	2	0.8118	0.8334	10	0.7897	0.8155	3	0.7938	0.8189
2	12	0.8108	0.8324	12	0.7880	0.8125	19	0.7903	0.8153
3	7	0.8091	0.8310	11	0.7876	0.8134	2	0.7900	0.8157
4	17	0.8084	0.8310	6	0.7867	0.8120	5	0.7887	0.8135
5	27	0.8083	0.8306	7	0.7849	0.8121	8	0.7886	0.8141
6	13	0.8069	0.8294	5	0.7842	0.8104	13	0.7886	0.8134
7	20	0.8063	0.8287	21	0.7840	0.8088	20	0.7883	0.8132
8	10	0.8061	0.8283	4	0.7836	0.8095	14	0.7878	0.8123
9	1	0.8049	0.8267	20	0.7835	0.8090	18	0.7871	0.8103
10	25	0.8044	0.8283	15	0.7832	0.8090	4	0.7869	0.8118
11	22	0.8041	0.8276	25	0.7829	0.8094	26	0.7866	0.8131
12	24	0.8029	0.8257	13	0.7827	0.8084	9	0.7863	0.8112
13	8	0.8023	0.8250	16	0.7825	0.8080	15	0.7859	0.8117
14	19	0.8018	0.8246	18	0.7815	0.8067	12	0.7858	0.8107
15	3	0.8016	0.8244	14	0.7809	0.8063	25	0.7852	0.8108
16	4	0.8009	0.8240	9	0.7807	0.8070	17	0.7850	0.8105
17	9	0.8005	0.8233	27	0.7805	0.8056	10	0.7846	0.8101
18	11	0.8000	0.8223	19	0.7803	0.8069	21	0.7844	0.8094
19	0	0.7989	0.8188	2	0.7799	0.8053	23	0.7844	0.8091
20	16	0.7988	0.8216	3	0.7796	0.8051	24	0.7844	0.8103
21	14	0.7987	0.8210	24	0.7795	0.8054	22	0.7843	0.8099
22	21	0.7981	0.8210	8	0.7791	0.8051	6	0.7840	0.8097
23	23	0.7981	0.8221	23	0.7786	0.8036	1	0.7830	0.8082
24	6	0.7979	0.8214	17	0.7778	0.8042	11	0.7830	0.8085
25	15	0.7970	0.8194	1	0.7711	0.7966	27	0.7819	0.8085
26	5	0.7959	0.8186	22	0.7697	0.7960	7	0.7818	0.8081
27	18	0.7947	0.8181	26	0.7675	0.7948	16	0.7810	0.8056
28	26	0.7929	0.8164	0	0.7584	0.7860	0	0.7627	0.7898

Successivamente è stata effettuata un'analisi relativa alla distribuzione di allucinazioni e risposte corrette nello spazio delle attivazioni. Affinchè fosse possibile visualizzare le attivazioni in uno spazio bidimensionale, è stata applicata la tecnica di riduzione della dimensionalità PCA (Principal Component Analysis).

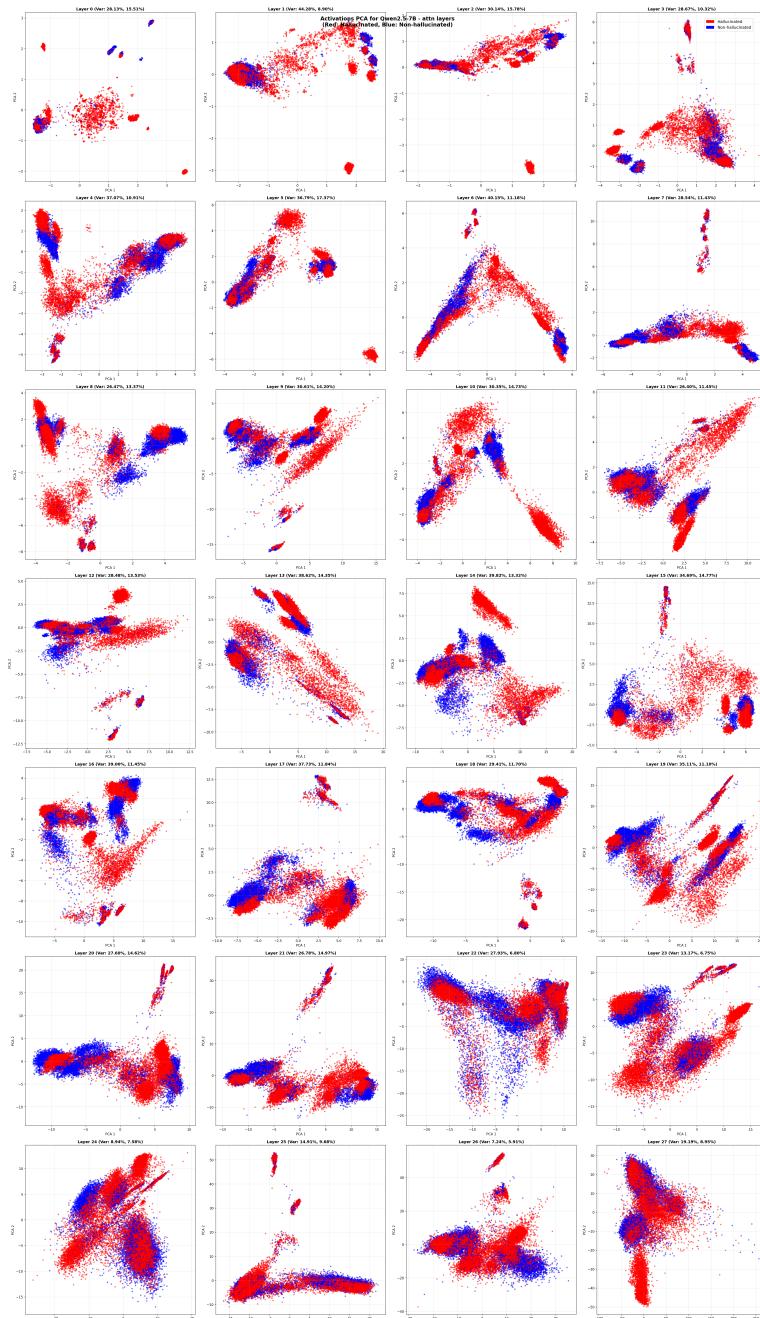


Figure 3.1: PCA delle attivazioni Attention del layer di Qwen2.5-7B

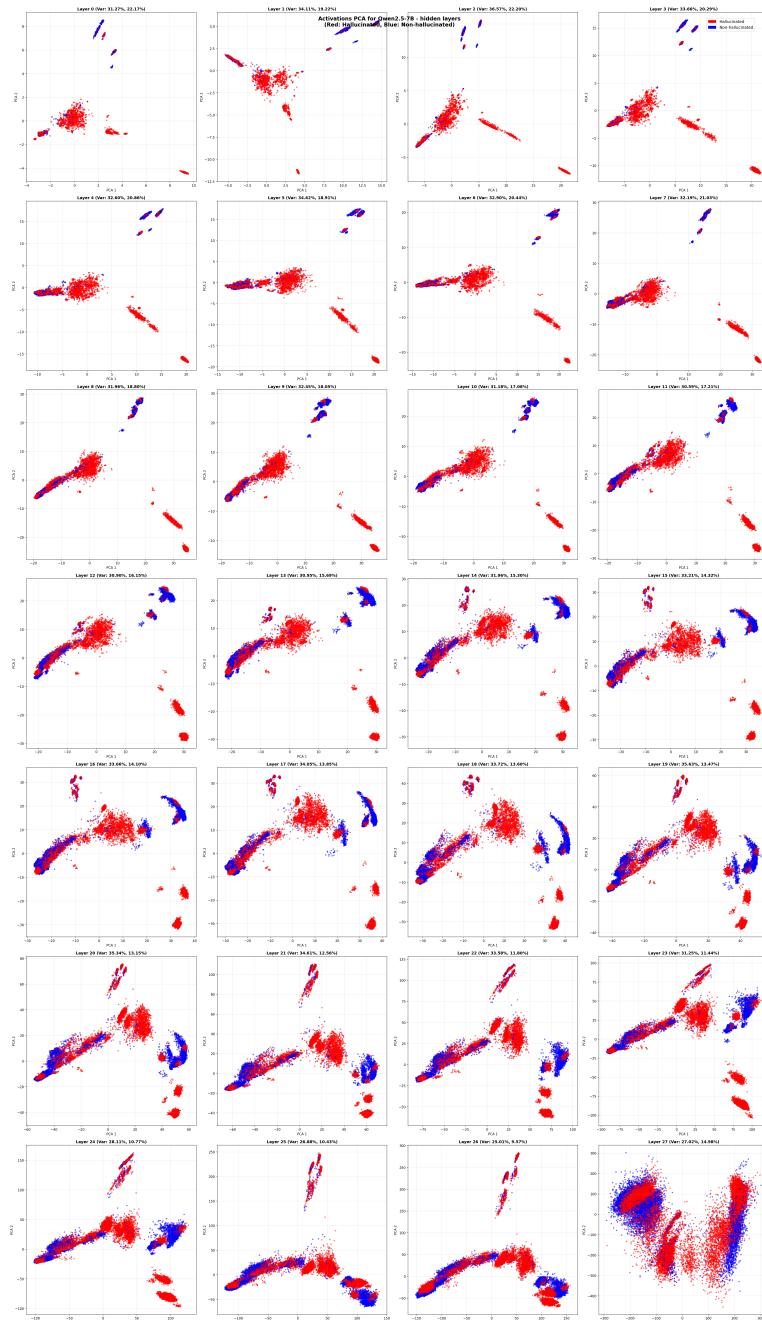


Figure 3.2: PCA delle attivazioni Hidden del layer di Qwen2.5-7B

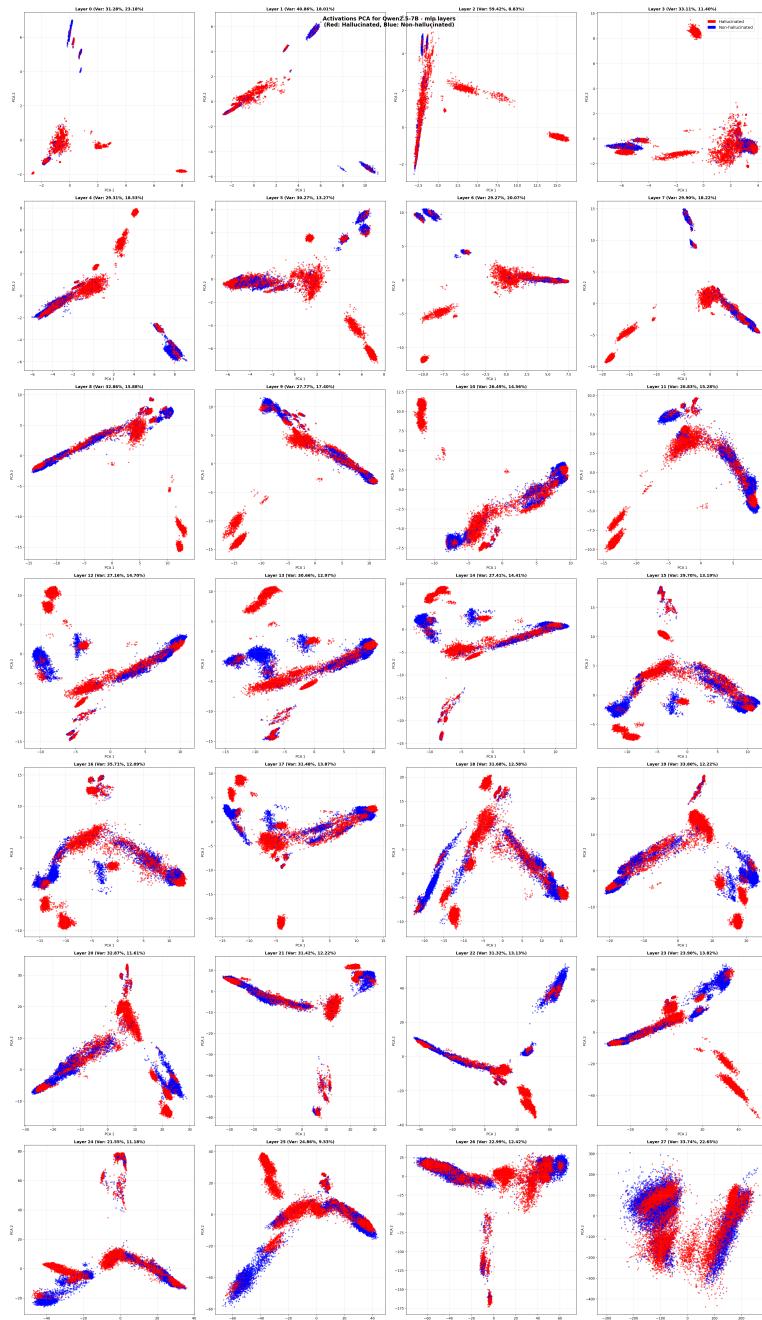


Figure 3.3: PCA delle attivazioni MLP del layer di Qwen2.5-7B

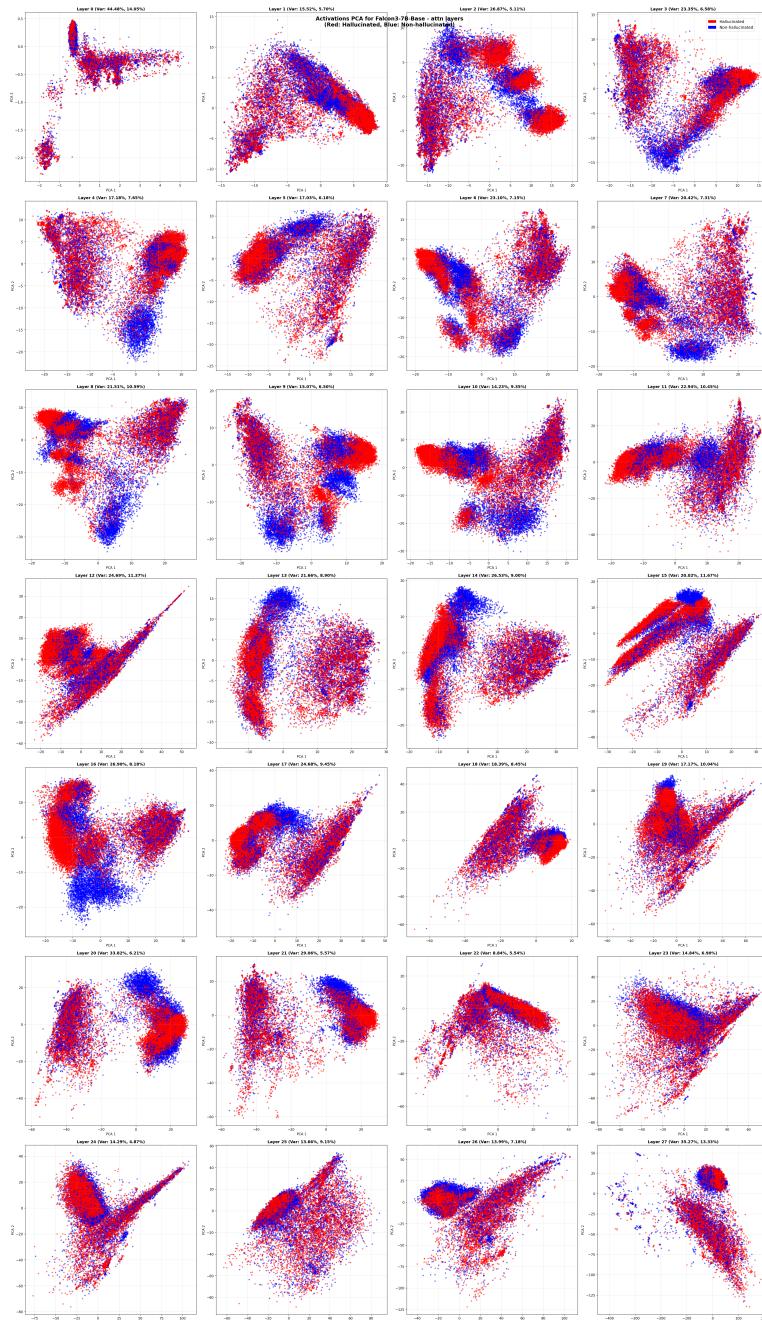


Figure 3.4: PCA delle attivazioni Attention del layer di Falcon3-7B-Base

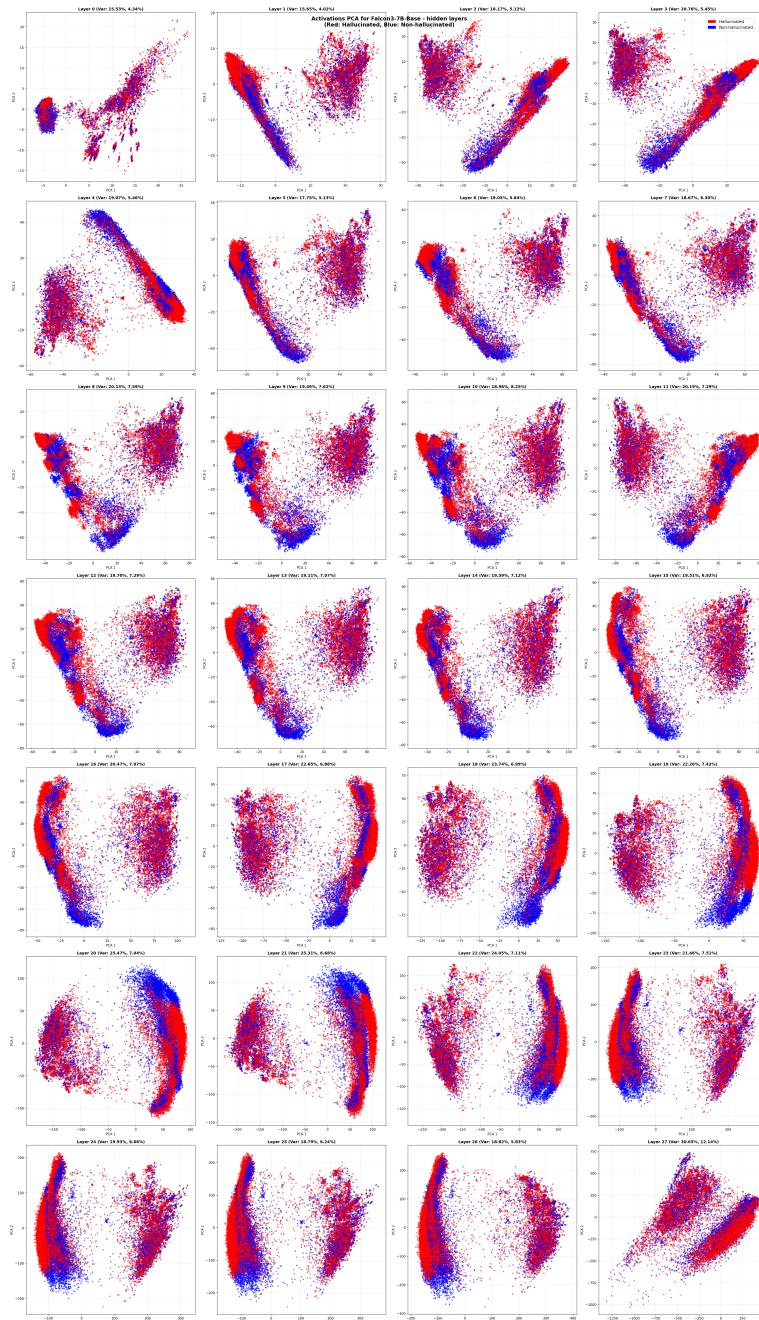


Figure 3.5: PCA delle attivazioni Hidden del layer di Falcon3-7B-Base

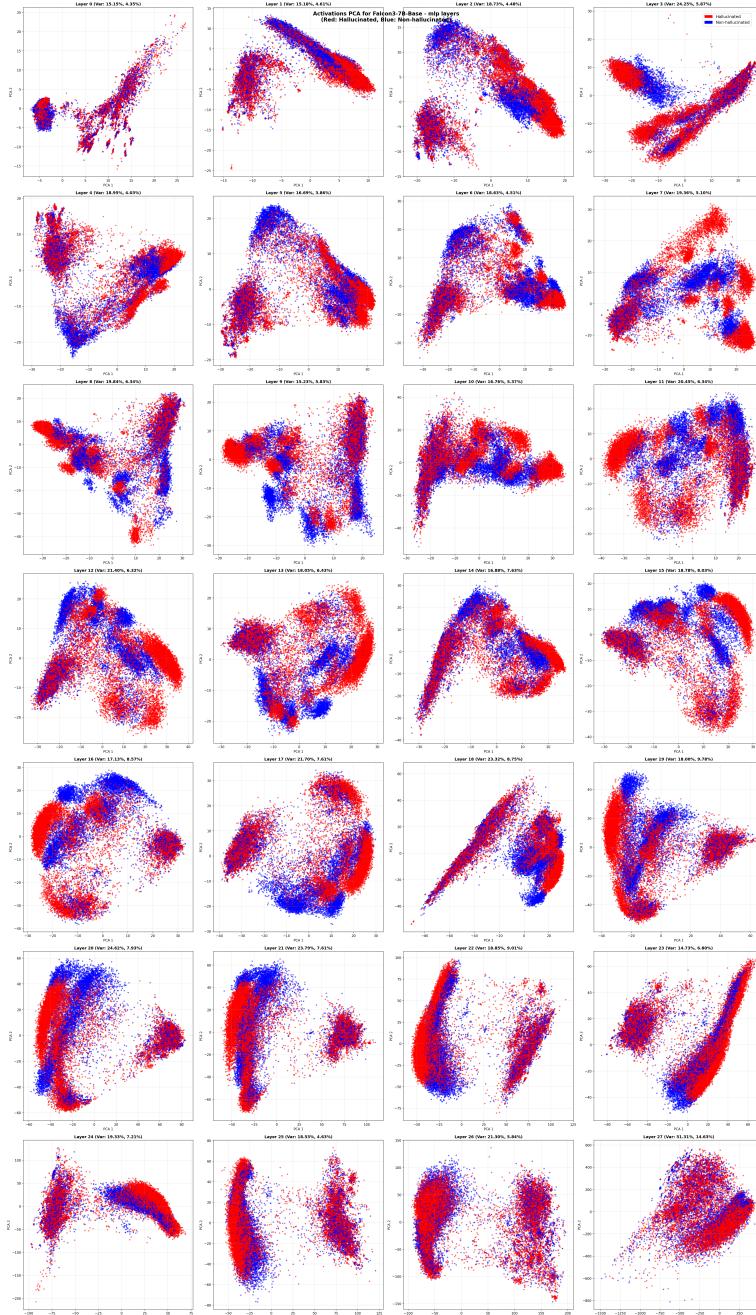


Figure 3.6: PCA delle attivazioni MLP del layer di Falcon3-7B-Base

Come si può vedere nei layer che performano meglio presi singolarmente abbiamo una più chiara separazione tra le due classi di esempi. Per queste ragioni, per gli esperimenti successivi sono stati scelti per ogni componente i 3 layer migliori in termini di performance.

Table 3.3: Configurazione dei layer selezionati per modello e componente

Modello	Attention (attn)	MLP	Hidden
Qwen2.5-7B	15, 16, 18	16, 18, 20	18, 19, 20
Falcon3-7B-Base	2, 7, 12	10, 11, 12	2, 3, 19

3.4 Metodologie per la costruzione del prober universale

In questa sezione vengono descritti i cinque approcci metodologici adottati per costruire un prober universale in grado di rilevare le allucinazioni nei modelli di linguaggio. Ogni approccio condivide la stessa pipeline di preparazione dei dati:

- Estrazione e concatenazione delle attivazioni corrispondenti per ogni esempio del dataset.
- Suddivisione del dataset in training e test set, mantenendo la stessa suddivisione per entrambi i modelli.
- Normalizzazione delle attivazioni tramite StandardScaler.

La riproducibilità degli esperimenti e l'utilizzo degli stessi set di dati per tutti i metodi sono garantiti dalla fissazione di un seed random comune.

3.4.1 Approccio Lineare (Baseline)

Come baseline per la costruzione del prober universale, è stato adottato un approccio completamente lineare. In questo scenario, le attivazioni interne dei modelli Qwen2.5-7B e Falcon3-7B-Base, estratte dai layer e componenti selezionati, sono state utilizzate per addestrare e valutare un classificatore Logistic Regression. L'obiettivo è verificare le performance utilizzando solo metodi lineari

Pipeline sperimentale

La pipeline sperimentale prevede i seguenti passaggi:

- Addestramento di un classificatore Logistic Regression per il rilevamento delle allucinazioni per il modello teacher.
- Valutazione delle prestazioni sul modello teacher tramite metriche di accuratezza e F1-score.
- Allineamento lineare tra lo spazio latente del modello studente e quello del modello insegnante tramite una proiezione lineare (Ridge Regression)

- Valutazione delle prestazioni del classificatore sui dati del modello studente dopo l'allineamento.

Per il Logistic Regressor sono stati utilizzati i seguenti iperparametri:

- **solver:** `lbfgs`
- **max_iterations:** 1000
- **class_weight:** `balanced`

3.4.2 Approccio Ibrido con AdapterMLP e Classificatore Lineare

In questo approccio si esplora una procedura di adattamento non lineare dello spazio latente del modello studente verso lo spazio del modello insegnante mediante una rete di allineamento a bassa dimensionalità (denominata AlignmentNetwork o AdapterMLP). L'obiettivo è osservare se l'introduzione di una componente non lineare nell'allineamento migliora le performance del prober universale, mantenendo un classificatore lineare (Logistic Regression) sul modello insegnante.

Architettura e loss

TODO:immagine architettura rete Alcune caratteristiche chiave dell'AlignmentNetwork includono:

- **Zero-init:** gli ultimi layer della decompressione sono inizializzati a zero in modo che la rete parta vicino a una trasformazione lineare identità, permettendo alla componente non-lineare di emergere solo se realmente utile.
- **Loss mista (MixedLoss):** la funzione di perdita combina Mean Squared Error (MSE) e una componente basata sulla similarità coseno:

$$\mathcal{L} = \alpha \cdot \text{MSE}(\hat{y}, y) + \beta \cdot (1 - \text{cosine_sim}(\hat{y}, y))$$

con pesi α (MSE) e β (cosine) configurabili per bilanciare fedeltà e allineamento angolare.

dove MSE =

$$\text{MSE}(\hat{y}, y) = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

e la similarità coseno è definita come:

$$\text{cosine_sim}(\hat{y}, y) = \frac{\hat{y} \cdot y}{\|\hat{y}\| \|y\|} = \frac{\sum_{i=1}^N \hat{y}_i y_i}{\sqrt{\sum_{i=1}^N \hat{y}_i^2} \sqrt{\sum_{i=1}^N y_i^2}}$$

Pipeline sperimentale

- **Teacher probing:** si addestra un probe (Logistic Regression) sullo spazio del teacher usando l'intero training set; questo rimane fisso durante la fase di alignment.
- **Training dell'alignment::** si crea una validazione interna per l'allineamento (split 90/10 dal training set del student) e si addestra l'AlignmentNetwork per proiettare le attivazioni del student nello spazio del teacher, minimizzando la loss tra le attivazioni proiettate e quelle originali del teacher.
- **Valutazione:** Le attivazioni di test del student vengono proiettate tramite l'alignment network e classificate usando il probe del teacher

Per il Logistic Regressor sono stati utilizzati i seguenti iperparametri:

- **solver:** lbfgs
- **max_iterations:** 1000
- **class_weight:** balanced

Di seguito sono riportati gli iperparametri principali utilizzati per l'addestramento dell'AlignmentNetwork:

Table 3.4: Iperparametri utilizzati nell'approccio AlignmentNetwork (AdapterMLP)

ID	Teacher	Student	Type	Hidden	Drop.	LR	W.D.	Batch	ES δ	Clip	Opt.	Sched.	α	β
1	Qwen2.5-7B	Falcon3-7B	attn	128	0.5	1e-3	1e-1	32	1e-4	1.0	AdamW	CosineAnnealingLR	0.01	1.0
2	Qwen2.5-7B	Falcon3-7B	mlp	128	0.5	1e-3	1e-1	32	1e-4	1.0	AdamW	CosineAnnealingLR	0.01	1.0
3	Qwen2.5-7B	Falcon3-7B	hidden	128	0.5	1e-3	1e-1	32	1e-4	1.0	AdamW	CosineAnnealingLR	0.01	1.0
4	Falcon3-7B	Qwen2.5-7B	attn	128	0.5	1e-3	1e-1	32	1e-4	1.0	AdamW	CosineAnnealingLR	0.01	1.0
5	Falcon3-7B	Qwen2.5-7B	mlp	128	0.5	1e-3	1e-1	32	1e-4	1.0	AdamW	CosineAnnealingLR	0.01	1.0
6	Falcon3-7B	Qwen2.5-7B	hidden	128	0.5	1e-3	1e-1	32	1e-4	1.0	AdamW	CosineAnnealingLR	0.01	1.0

3.4.3 Approccio con AlignmentNetwork e Classificatore MLP

Questo approccio segue il precedente per quanto riguarda la parte di allineamento tra i due LLM. La differenza risiede nel prober non-lineare (MLP) come classificatore sul teacher. L'obiettivo è verificare le performance utilizzando entrambe le componenti non-lineari seguendo la stessa pipeline sperimentale.

Architettura, componenti e Loss

TODo:Inserire immagine architettura rete

- **AlignmentNetwork (AdapterMLP):** Viene utilizzata la stessa architettura descritta nell'approccio precedente con la stessa loss

- **MLP Prober:** Classificatore non-lineare per rilevamento della allucinazione: rete fully-connected con layer intermedi normalizzati (LayerNorm), attivazione GELU e dropout.
- **BCEWithLogitsLoss:** combina una funzione sigmoide e la Binary Cross Entropy per stabilità numerica. Dato un input (logit) x e un target y , la perdita è definita come:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^N [y_i \cdot \log(\sigma(x_i)) + (1 - y_i) \cdot \log(1 - \sigma(x_i))]$$

dove $\sigma(x_i) = \frac{1}{1+e^{-x_i}}$ è la funzione sigmoide, N è la dimensione del batch, $y_i \in \{0, 1\}$ è il target e x_i è il logit predetto.

Pipeline sperimentale

- **Teacher probing:** si addestra il prober (qui MLP) sullo spazio del teacher. Per il prober viene effettuato un validation split interno per early stopping
- **Training Alignment:** Come nell'approccio precedente
- **Valutazione:** Le attivazioni di test del student vengono proiettate tramite l'alignment network e classificate usando il prober MLP del teacher

Di seguito sono riportati gli iperparametri principali utilizzati per l'addestramento dell'AlignmentNetwork e del Prober MLP:

Table 3.5: Parte 1: Parametri Alignment Network e Loss

ID	Teach	Stud	Typ	Alignment Network Parameters									Loss	
				Hid	Drp	LR	WD	Btc	ES	Clp	Opt	Sch	α	β
1	Qwen	Falc	attn	128	0.5	1e-3	1e-1	32	1e-4	1.0	AdamW	CosAnn	0.01	1.0
2	Qwen	Falc	mlp	128	0.5	1e-3	1e-1	32	1e-4	1.0	AdamW	CosAnn	0.01	1.0
3	Qwen	Falc	hidden	128	0.5	1e-3	1e-1	32	1e-4	1.0	AdamW	CosAnn	0.01	1.0
4	Falc	Qwen	attn	128	0.5	1e-3	1e-1	32	1e-4	1.0	AdamW	CosAnn	0.01	1.0
5	Falc	Qwen	mlp	128	0.5	1e-3	1e-1	32	1e-4	1.0	AdamW	CosAnn	0.01	1.0
6	Falc	Qwen	hidden	128	0.5	1e-3	1e-1	32	1e-4	1.0	AdamW	CosAnn	0.01	1.0

Legenda: *Hid*: Hidden Dim, *Drp*: Dropout, *WD*: Weight Decay, *Btc*: Batch Size, *ES*: Early Stopping, *Clp*: Gradient Clip, *Sch*: Scheduler (CosAnnLR).

3.4.4 Approccio con Autoencoder, AlignmentNetwork e Classificatore MLP

Questo approccio riduce la dimensionalità delle attivazioni tramite autoencoder. Per quanto riguarda il resto viene mantenuta la stessa pipeline sperimentale dell'approccio precedente, con un accortezza: l'AlignmentNetwork ora mappa tra gli spazi latenti degli autoencoder del teacher e dello student e il prober MLP opera nello spazio latente del teacher.

Table 3.6: Parte 2: Parametri Prober MLP

ID	Teach	Stud	Typ	Prober MLP Parameters									
				Hid	Drp	LR	WD	Btc	ES	Clp	Opt	Sch	
1	Qwen	Falc	attn	64	0.5	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnn	
2	Qwen	Falc	mlp	64	0.5	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnn	
3	Qwen	Falc	hidden	64	0.5	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnn	
4	Falc	Qwen	attn	64	0.5	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnn	
5	Falc	Qwen	mlp	64	0.5	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnn	
6	Falc	Qwen	hidden	64	0.5	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnn	

Legenda: *Hid*: Hidden Dim, *Drp*: Dropout, *WD*: Weight Decay, *Btc*: Batch Size, *ES*: Early Stopping, *Clp*: Gradient Clip, *Sch*: Scheduler (CosAnnLR).

Architettura e componenti principali

TODo:Inserire immagine architettura rete

- **Autoencoder (Teacher & Student)**: si addestra un autoencoder sullo spazio delle attivazioni del teacher e uno sullo spazio delle attivazioni dello student. Gli encoder producono rappresentazioni latenti di dimensione comune ai due. La loss utilizzata è la Mean Squared Error (MSE):
- **Alignment Network (latent)**: La rete utilizzata è la stessa degli approcci precedenti con la stessa Loss
- **MLP Prober**: Il classificatore MLP utilizzato è lo stesso descritto nell'approccio precedente con la stessa loss

Pipeline sperimentale

1. Addestramento Autoencoder Teacher e Student
2. Codifica nello spazio latente Z_T e Z_S delle attivazioni del teacher e dello student.
3. Addestramento MLP Prober: si allena un MLP sullo spazio latente del teacher con early stopping per validazione.
4. Addestramento Alignment Network: si allena l'Alignment Network per mappare Z_S in Z_T
5. Valutazione: il teacher prober (MLP) viene usato per classificare le rappresentazioni allineate dello student

Di seguito sono riportati gli iperparametri principali utilizzati per l'addestramento dell'Autoencoder, dell'AlignmentNetwork e del Prober MLP:

Table 3.7: Iperparametri (Parte 1 di 4): Parametri AutoEncoder (AEP)

ID	Teach	Stud	Typ	Lat	Hid	Drp	LR	WD	Btc	ES	Clp	Opt	Sch
1	Qwen	Falc	attn	128	256	0.2	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
2	Qwen	Falc	mlp	128	256	0.2	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
3	Qwen	Falc	hidden	128	256	0.2	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
4	Falc	Qwen	attn	128	256	0.2	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
5	Falc	Qwen	mlp	128	256	0.2	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
6	Falc	Qwen	hidden	128	256	0.2	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR

Legenda: *Lat*: Latent Dim, *Hid*: Hidden Dim, *Drp*: Dropout, *WD*: Weight Decay, *Btc*: Batch Size, *ES*: Early Stopping Delta, *Clp*: Grad Clip, *Sch*: Scheduler.

Table 3.8: Iperparametri (Parte 2 di 4): Parametri Alignment Network (ANP)

ID	Teach	Stud	Typ	Hid	Drp	LR	WD	Btc	ES	Clp	Opt	Sch	α	β
1	Qwen	Falc	attn	256	0.3	1e-3	1e-2	32	1e-4	1.0	AdamW	CosAnnLR	0.5	0.5
2	Qwen	Falc	mlp	256	0.3	1e-3	1e-2	32	1e-4	1.0	AdamW	CosAnnLR	0.5	0.5
3	Qwen	Falc	hidden	256	0.3	1e-3	1e-2	32	1e-4	1.0	AdamW	CosAnnLR	0.5	0.5
4	Falc	Qwen	attn	256	0.3	1e-3	1e-2	32	1e-4	1.0	AdamW	CosAnnLR	0.5	0.5
5	Falc	Qwen	mlp	256	0.3	1e-3	1e-2	32	1e-4	1.0	AdamW	CosAnnLR	0.5	0.5
6	Falc	Qwen	hidden	256	0.3	1e-3	1e-2	32	1e-4	1.0	AdamW	CosAnnLR	0.5	0.5

Legenda: *Hid*: Hidden Dim, *Drp*: Dropout, *WD*: Weight Decay, *Btc*: Batch Size, *ES*: Early Stopping Delta, *Clp*: Grad Clip, *Sch*: Scheduler.

Table 3.9: Iperparametri (Parte 3 di 4): Parametri Prober MLP (MNP)

ID	Teach	Stud	Typ	Hid	Drp	LR	WD	Btc	ES	Clp	Opt	Sch
1	Qwen	Falc	attn	64	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
2	Qwen	Falc	mlp	64	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
3	Qwen	Falc	hidden	64	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
4	Falc	Qwen	attn	64	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
5	Falc	Qwen	mlp	64	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
6	Falc	Qwen	hidden	64	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR

Legenda: *Hid*: Hidden Dim, *Drp*: Dropout, *WD*: Weight Decay, *Btc*: Batch Size, *ES*: Early Stopping Delta, *Clp*: Grad Clip, *Sch*: Scheduler.

3.4.5 Approccio One-For-All (Frozen Head + Adapter Encoder)

In questo approccio si adotta una procedura in due fasi pensata per valutare la trasferibilità di una funzione decisionale (la *head* di classificazione) tra due modelli differenti mantenendo la stessa testa e adattando soltanto l'encoder dello studente. Si parte inizialmente addestrando un modello *teacher* completo (Encoder + Head); successivamente si congela la Head addestrata e si addestra un nuovo Encoder *student* a emettere latenti compatibili con la Head del teacher.

Architettura e componenti principali

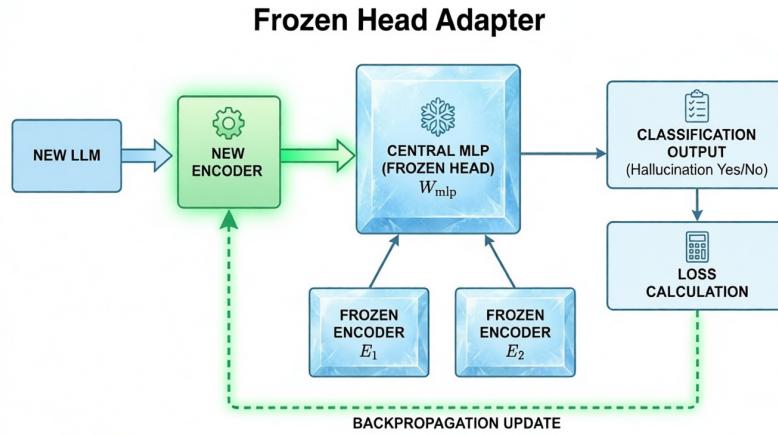


Figure 3.7: Architettura del prober One-For-All

TODO:Inserire vera immagine architettura rete,questo è un esempio generato con AI per far comprendere l'idea

- **Encoder:** rete feed-forward a più blocchi che mappa lo spazio di input in uno spazio latente di dimensione ridotta
- **Classification Head:** modulo compatto che prende in input il vettore latente e restituisce un logit binario. Implementato come una piccola MLP.

Pipeline sperimentale

1. Addestramento Teacher Encoder + Head: si addestra un encoder e una head partendo dalle attivazioni del modello teacher.
2. Congelamento Head: si congela la head addestrata per mantenere la funzione decisionale.
3. Addestramento Student Encoder: si addestra un nuovo encoder sulle attivazioni del modello student. Questo encoder è direttamente collegato alla head congelata del teacher, dunque la loss viene calcolata sui logit prodotti dalla head.
4. Valutazione: si valuta la performance del sistema Encoder student + Head teacher sulle attivazioni di test del modello student.

Di seguito sono riportati gli iperparametri principali utilizzati per l'addestramento degli Encoder e della Classification Head:

Table 3.10: Iperparametri dell' Encoder

ID	Teach	Stud	Typ	Lat	Hid	Drp	LR	WD	Btc	ES	Clp	Opt	Sch
1	Qwen	Falc	attn	256	512	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
2	Qwen	Falc	mlp	256	512	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
3	Qwen	Falc	hidden	256	512	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
4	Falc	Qwen	attn	256	512	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
5	Falc	Qwen	mlp	256	512	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
6	Falc	Qwen	hidden	256	512	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR

Legenda: *Lat*: Latent Dim, *Hid*: Hidden Dim, *Drp*: Dropout, *WD*: Weight Decay, *Btc*: Batch Size, *ES*: Early Stopping Delta, *Clp*: Grad Clip, *Sch*: Scheduler.

Table 3.11: Iperparametri della Classification Head

ID	Teach	Stud	Typ	Hid	Drp	LR	WD	Btc	ES	Clp	Opt	Sch
1	Qwen	Falc	attn	128	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
2	Qwen	Falc	mlp	128	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
3	Qwen	Falc	hidden	128	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
4	Falc	Qwen	attn	128	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
5	Falc	Qwen	mlp	128	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR
6	Falc	Qwen	hidden	128	0.3	1e-3	1e-2	64	1e-4	1.0	AdamW	CosAnnLR

Legenda: *Hid*: Hidden Dim, *Drp*: Dropout, *WD*: Weight Decay, *Btc*: Batch Size, *ES*: Early Stopping Delta, *Clp*: Grad Clip, *Sch*: Scheduler.

Chapter 4

Risultati

In questo capitolo vengono presentati i risultati ottenuti dagli esperimenti condotti per valutare l'efficacia del prober universale nel rilevamento delle allucinazioni nei modelli di linguaggio di grandi dimensioni (LLM). Per ogni approccio viene riportata una tabella con le metriche principali: accuratezza del teacher (Acc_T), accuratezza dello student dopo allineamento (Acc_S), differenza assoluta (ΔAcc) e differenza percentuale ($\Delta\%$).

4.1 Approccio Lineare (Baseline)

Table 4.1: Risultati dell'approccio Lineare (Baseline)

ID	Teacher	Student	Type	Acc_T	Acc_S	ΔAcc	$\Delta\%$
1	Qwen	Falcon	attn	0.9888	0.7455	0.2433	24.61%
2	Qwen	Falcon	mlp	0.9906	0.7375	0.2531	25.55%
3	Qwen	Falcon	hidden	0.9903	0.7397	0.2506	25.31%
4	Falcon	Qwen	attn	0.9234	0.7658	0.1576	17.07%
5	Falcon	Qwen	mlp	0.9103	0.7649	0.1454	15.97%
6	Falcon	Qwen	hidden	0.9102	0.7728	0.1374	15.10%

4.2 Approccio Ibrido con AdapterMLP e Classificatore Lineare

Table 4.2: Risultati dell'approccio Ibrido (AdapterMLP + Logistic Regression)

ID	Teacher	Student	Type	Acc_T	Acc_S	ΔAcc	$\Delta\%$
1	Qwen	Falcon	attn	0.9888	0.7387	0.2501	25.29%
2	Qwen	Falcon	mlp	0.9906	0.7174	0.2732	27.58%
3	Qwen	Falcon	hidden	0.9903	0.7371	0.2532	25.57%
4	Falcon	Qwen	attn	0.9234	0.7605	0.1629	17.64%
5	Falcon	Qwen	mlp	0.9103	0.7430	0.1673	18.38%
6	Falcon	Qwen	hidden	0.9102	0.7679	0.1423	15.63%

4.3 Approccio con AlignmentNetwork e Classificatore MLP

Table 4.3: Risultati dell'approccio AlignmentNetwork + MLP Prober

ID	Teacher	Student	Type	Acc_T	Acc_S	ΔAcc	$\Delta\%$
1	Qwen	Falcon	attn	0.9910	0.7517	0.2393	24.15%
2	Qwen	Falcon	mlp	0.9908	0.7070	0.2838	28.64%
3	Qwen	Falcon	hidden	0.9917	0.7454	0.2463	24.84%
4	Falcon	Qwen	attn	0.9297	0.7550	0.1747	18.79%
5	Falcon	Qwen	mlp	0.9087	0.7747	0.1340	14.75%
6	Falcon	Qwen	hidden	0.9112	0.7939	0.1173	12.87%

4.4 Approccio con Autoencoder, AlignmentNetwork e Classificatore MLP

Table 4.4: Risultati dell'approccio Autoencoder + AlignmentNetwork + MLP Prober

ID	Teacher	Student	Type	Acc_T	Acc_S	ΔAcc	$\Delta\%$
1	Qwen	Falcon	attn	0.9831	0.7112	0.2719	27.66%
2	Qwen	Falcon	mlp	0.9792	0.7098	0.2694	27.51%
3	Qwen	Falcon	hidden	0.9793	0.7128	0.2665	27.21%
4	Falcon	Qwen	attn	0.9054	0.7753	0.1301	14.37%
5	Falcon	Qwen	mlp	0.8902	0.7587	0.1315	14.77%
6	Falcon	Qwen	hidden	0.8850	0.7639	0.1211	13.68%

4.5 Approccio One-For-All

Table 4.5: Risultati dell'approccio One-For-All (Frozen Head + Adapter Encoder)

ID	Teacher	Student	Type	Acc_T	Acc_S	ΔAcc	$\Delta\%$
1	Qwen	Falcon	attn	0.9910	0.9257	0.0653	6.59%
2	Qwen	Falcon	mlp	0.9866	0.9139	0.0727	7.37%
3	Qwen	Falcon	hidden	0.9863	0.9067	0.0795	8.06%
4	Falcon	Qwen	attn	0.9266	0.9920	-0.0654	-7.06%
5	Falcon	Qwen	mlp	0.9091	0.9894	-0.0804	-8.84%
6	Falcon	Qwen	hidden	0.9063	0.9912	-0.0850	-9.38%

Nota: I valori negativi di ΔAcc indicano che lo student ha ottenuto prestazioni migliori del teacher. Questo fenomeno si verifica quando Falcon è il teacher e Qwen lo student, suggerendo che le attivazioni di Qwen sono più facilmente adattabili alla head addestrata su Falcon.

Chapter 5

Discussione

In questo capitolo vengono discussi i risultati ottenuti dagli esperimenti condotti per valutare i diversi approcci proposti per il rilevamento delle allucinazioni nei modelli di linguaggio di grandi dimensioni (LLM) mediante un prober universale.

Chapter 6

Conclusioni

In questo capitolo vengono riassunti i principali risultati ottenuti dal lavoro svolto e le prospettive future.

6.1 Sintesi dei Risultati

6.2 Lavori Futuri

Il lavoro svolto può essere proseguito in diverse direzioni:

- **Espansione degli esperimenti:** Ripetere gli esperimenti con diversi modelli e diversi dataset
- **Verifica cross-dataset:** Testare la capacità di generalizzazione del prober su dataset diversi da quelli utilizzati per l'addestramento.
- **Ottimizzazione delle architetture:** Capire quali sono i migliori layer da utilizzare per l'estrazione delle attivazioni e sperimentare
- **Esperimenti multimodali:** Estendere l'approccio a modelli multimodali che integrano testo e immagini.