Generative AI

L’AI generative crea nuovi dati sulla base di un modello addestrato.

I modelli generativi possono generare come output dati in formato di testo, audio, video etc.

Può essere divisa in due macroaree:

* Generative Image model
* Generative Language model

Può essere categorizzata in base al tipo di input e di output

* Text to text =>GAN
* Text to image =>LLM
* Image to text =>LLM
* Image to image =>LLM

A cat looking at the camera

Description automatically generated

## Where generative AI exists?

A white circle with black text

Description automatically generated with medium confidence

* Machine Learning è un subset dell’AI
* Deep Learning è un subset of Machine Learning
* Generative AI è un subset del Deep Learning

Genarative AI è un superset composto da

1. GAN
2. LLM

# Large Language Model

A black and white logo

Description automatically generated

* ChatGPT
* Google Bard
* Meta Llama 2

# Deep learning

## ANN (Artificial neural network)

Hidden Layer

Input Output

Le Artificial Neural Networks (ANN), o Reti Neurali Artificiali, sono modelli computazionali ispirati al funzionamento del cervello umano, utilizzati nell'ambito dell'intelligenza artificiale per simulare il modo in cui gli esseri umani apprendono. Questi modelli sono composti da unità di elaborazione chiamate neuroni artificiali o nodi, organizzati in strati.

### Componenti Principali

1. **Neuroni (Nodi)**: I neuroni sono le unità fondamentali di calcolo delle ANN, che ricevono input, eseguono una certa elaborazione e passano l'output ai neuroni successivi.
2. **Pesi (Weights)**: Ogni connessione tra i neuroni di strati consecutivi è associata a un peso, che influenza l'importanza dell'input ricevuto da un neurone nel calcolo dell'output.
3. **Bias**: Un termine aggiuntivo che si aggiunge al calcolo dell'input ponderato di un neurone per aiutare il modello a adattarsi meglio ai dati.
4. **Funzioni di Attivazione**: Le funzioni di attivazione determinano l'output di un neurone dato un insieme di input. Svolgono un ruolo cruciale nel consentire alle reti neurali di apprendere complessi pattern non lineari.

### ANN : Funzionamento

Le ANN ricevono input attraverso il primo strato di neuroni (strato di input), poi l'informazione viene elaborata attraverso uno o più strati intermedi (strati nascosti) prima di raggiungere lo strato di output, che fornisce il risultato finale. Durante l'addestramento, i pesi e i bias della rete vengono aggiustati attraverso un processo chiamato backpropagation, che utilizza un algoritmo di ottimizzazione come la discesa gradiente per minimizzare l'errore tra le previsioni della rete e i dati reali.

### Applicazioni

Le ANN trovano applicazione in una vasta gamma di campi, tra cui:

* **Riconoscimento di immagini e video**: per identificare oggetti, persone, scene, ecc.
* **Elaborazione del linguaggio naturale**: per traduzione automatica, riconoscimento vocale e generazione di testo.
* **Previsioni finanziarie**: per analizzare tendenze di mercato e valutare rischi.
* **Diagnostica medica**: per identificare pattern nei dati clinici che possono indicare diagnosi specifiche.

Le ANN sono uno strumento potente per modellare problemi complessi grazie alla loro capacità di apprendere direttamente dai dati, senza necessità di programmazione esplicita per compiti specifici.

## CNN (Convolution neural network)

Le Convolutional Neural Networks (CNN) o Reti Neurali Convoluzionali sono una categoria speciale di reti neurali artificiali che hanno trovato grande spazio nel campo della visione artificiale, riconoscimento di immagini, videoanalisi, e in applicazioni di elaborazione del linguaggio naturale (NLP) dove la struttura spaziale dei dati è cruciale.

### Caratteristiche Principali

1. **Strati Convoluzionali**: Il cuore delle CNN sono gli strati convoluzionali che applicano un'operazione matematica chiamata convoluzione. La convoluzione utilizza un filtro o kernel che passa sull'immagine di input (o sulle mappe di caratteristiche precedenti) per estrarre caratteristiche importanti come bordi, angoli, texture, ecc.
2. **Pooling (Sottocampionamento)**: Le CNN utilizzano frequentemente strati di pooling per ridurre le dimensioni spaziali delle mappe di caratteristiche, rendendo il modello più efficiente e diminuendo la sensibilità alla posizione esatta delle caratteristiche all'interno dell'immagine.
3. **Strati di Attivazione**: Come nelle altre ANN, gli strati di attivazione nelle CNN aiutano a introdurre non linearità nel modello, permettendo alla rete di apprendere complessi pattern nei dati.
4. **Strati Fully-Connected (Densamente Connessi)**: Verso il termine della rete, le CNN spesso includono uno o più strati fully-connected che prendono le caratteristiche estratte dagli strati convoluzionali e di pooling per eseguire la classificazione o altre compiti specifici.

### Funzionamento

Il processo di una CNN inizia con l'immagine di input che passa attraverso una serie di strati convoluzionali, di attivazione (come ReLU), e di pooling. Questi strati lavorano in sequenza per estrarre progressivamente caratteristiche ad alto livello dall'immagine. Dopo aver passato attraverso questi strati, l'output viene poi appiattito e alimentato attraverso strati fully-connected per la classificazione finale o altre decisioni.

### Applicazioni

* **Riconoscimento di Immagini**: Identificazione di oggetti, persone, segnali stradali, ecc., in immagini statiche.
* **Riconoscimento di Pattern in Video**: Come il riconoscimento di attività umane, il tracciamento di oggetti, ecc.
* **Analisi di Testo e Sequenze**: Anche se più comunemente associate a immagini, le CNN sono state applicate con successo anche in NLP, per esempio, nella classificazione di documenti o nell'analisi di sentimenti.
* **Diagnostica Medica**: Aiutare i medici nell'interpretazione di immagini mediche come raggi X, MRI, ecc.

Le CNN hanno rivoluzionato il campo della visione artificiale e continuano a essere al centro della ricerca per migliorare ulteriormente la precisione e l'efficienza nel riconoscimento e nell'analisi di immagini e video.

## RNN (Recurrent neural network)

FeedBack Loop

Hidden Layer

Input Output

Le RNN sono progettate per elaborare dati sequenziali, come serie temporali o sequenze di parole. A differenza delle reti neurali tradizionali, le RNN hanno connessioni ricorrenti che permettono di mantenere uno "stato" che cattura le informazioni delle sequenze di input precedenti. Questo le rende particolarmente adatte per:

* **Elaborazione del Linguaggio Naturale (NLP)**: Compiti come la traduzione automatica, la generazione di testo e il riconoscimento vocale.
* **Previsioni di Serie Temporali**: Ad esempio, prevedere il prezzo delle azioni o il consumo energetico.

Un problema comune delle RNN standard è la difficoltà nel catturare dipendenze a lungo termine a causa del problema del gradiente che svanisce o esplode. Questo ha portato allo sviluppo di varianti più sofisticate, come le Long Short-Term Memory (LSTM) e le Gated Recurrent Unit (GRU), che sono più efficaci nel mantenere informazioni per periodi più lunghi.

In sintesi, mentre le CNN eccellono nell'analizzare dati spaziali come le immagini, le RNN sono superiori nell'elaborazione di dati sequenziali, consentendo loro di catturare dinamiche temporali o sequenziali all'interno dei dati.

## RL (Reinforcement learning)

Reinforcement Learning (RL) è un tipo di apprendimento automatico in cui un agente impara a prendere decisioni ottimizzando le sue azioni in un ambiente specifico al fine di massimizzare una ricompensa cumulativa. A differenza dell'apprendimento supervisionato, dove l'agente apprende da un set di dati contenente le risposte corrette, in RL l'agente apprende esclusivamente tramite l'esperienza, provando diverse strategie e ricevendo feedback sotto forma di ricompense o punizioni.

### Componenti Chiave dell'RL:

1. **Agente**: L'entità che prende le decisioni basandosi sulla sua politica.
2. **Ambiente**: Il contesto o il dominio in cui l'agente opera e prende decisioni.
3. **Stato**: Una rappresentazione dell'ambiente in un dato momento, basata sulle percezioni dell'agente.
4. **Azione**: Un'operazione o un'attività che l'agente può scegliere di eseguire.
5. **Ricompensa**: Un segnale immediato dato all'agente in risposta alle sue azioni, che mira a valutare l'efficacia di quella azione rispetto all'obiettivo dell'agente.

### Processo di Apprendimento:

L'apprendimento per rinforzo si basa sull'interazione tra l'agente e l'ambiente. L'agente osserva lo stato dell'ambiente, decide e compie un'azione, e poi riceve una ricompensa e osserva il nuovo stato risultante dall'azione. Questo ciclo aiuta l'agente a costruire o aggiornare la sua politica, che è una mappa da stati a azioni che massimizza la ricompensa totale attesa lungo il tempo.

### Metodi di RL:

* **Metodi basati su valore**: L'agente cerca di ottimizzare una funzione di valore che stima il valore futuro delle azioni in uno stato dato.
* **Metodi basati sulla politica**: L'agente ottimizza direttamente la politica di azione senza fare affidamento su una funzione di valore.
* **Metodi Actor-Critic**: Combinano gli approcci basati su valore e sulla politica, utilizzando due modelli: uno per la politica (actor) e uno per il valore (critic).

### Applicazioni:

Le tecniche di RL hanno trovato applicazione in una vasta gamma di campi, tra cui:

* **Giochi**: Superare umani in giochi complessi come gli scacchi, Go, e videogiochi.
* **Robotica**: Imparare il controllo di robot per compiti come camminare, afferrare oggetti, o navigare.
* **Sistemi di raccomandazione**: Ottimizzare le raccomandazioni personalizzate per massimizzare l'engagement degli utenti.
* **Ottimizzazione del traffico**: Gestire il flusso del traffico e i semafori per ridurre gli ingorghi.
* **Gestione delle risorse in reti di comunicazione**: Ottimizzare l'allocazione delle risorse in reti di telecomunicazioni.

Il RL rappresenta un'area di ricerca attiva e in rapida evoluzione, con nuove tecniche e applicazioni che emergono regolarmente.

## GAN (Generative Adversarial Networks)

A diagram of a sample

Description automatically generated

Generative Adversarial Networks (GAN) sono una classe di algoritmi di apprendimento automatico nell'ambito dell'intelligenza artificiale, ideate da Ian Goodfellow e i suoi colleghi nel 2014. Le GAN fanno parte della famiglia dei modelli generativi e sono composte da due reti neurali che vengono addestrate simultaneamente attraverso un processo competitivo:

* **Generatore (Generator)**: Questa rete ha il compito di generare dati nuovi che assomiglino il più possibile ai dati reali. Non ha accesso diretto ai dati reali ma impara a produrre dati simili attraverso il feedback ricevuto dal discriminatore.
* **Discriminatore (Discriminator)**: Questa rete ha il compito di distinguere i dati reali da quelli generati artificialmente dal generatore. Viene addestrata con esempi sia reali che generati, e il suo obiettivo è quello di migliorare la propria capacità di classificazione.

Il processo di addestramento delle GAN è una sorta di gioco a somma zero, dove il generatore cerca di "ingannare" il discriminatore producendo dati sempre più simili a quelli reali, mentre il discriminatore cerca di diventare sempre più abile nel distinguere i veri dati da quelli falsi. Attraverso questo processo, il generatore impara a creare dati che sono indistinguibili da quelli reali.

Le GAN hanno vasti campi di applicazione, tra cui la generazione di immagini realistiche, la creazione di opere d'arte, il miglioramento della qualità delle immagini, la generazione di modelli 3D, la sintesi vocale e molto altro. Grazie alla loro capacità di generare dati realistici da un insieme di apprendimento, le GAN hanno aperto nuove possibilità nel campo della creatività assistita dal computer, nella simulazione e in molte altre aree dell'intelligenza artificiale.

## RNN e LSTM

Le Recurrent Neural Networks (RNN) e le Long Short-Term Memory (LSTM) networks sono due tipi di architetture di reti neurali progettate per gestire sequenze di dati, come serie temporali, linguaggio naturale, o qualsiasi altro tipo di dati sequenziali. Entrambe sono particolarmente adatte per problemi in cui esiste una dipendenza temporale tra gli input.

### Recurrent Neural Networks (RNN)

Le RNN sono una classe di reti neurali che hanno connessioni ricorrenti, permettendo informazioni di persistere attraverso il tempo. In una RNN, l'output da uno strato in un certo momento viene passato insieme all'input del momento successivo, creando una sorta di "memoria" degli input passati. Questo è particolarmente utile per elaborare sequenze di dati dove il contesto è importante per comprendere l'elemento corrente della sequenza.

Tuttavia, le RNN standard soffrono di alcuni problemi significativi, come il problema del gradiente che esplode o svanisce (vanishing/exploding gradient problem), che rende difficile l'apprendimento di dipendenze a lungo termine tra gli elementi della sequenza.

### Long Short-Term Memory (LSTM) Networks

Le LSTM sono una speciale sottoclasse di RNN progettate per superare il problema del gradiente che svanisce, rendendole capaci di apprendere dipendenze a lungo termine. Sono state introdotte da Hochreiter & Schmidhuber nel 1997 e da allora sono diventate uno degli approcci più popolari per le applicazioni di RNN.

Una LSTM aggiunge una struttura chiamata cella di memoria, che può mantenere informazioni per lunghi periodi. Ogni cella ha tre porte (gate):

1. **Forget Gate**: determina quali informazioni scartare dalla cella.
2. **Input Gate**: aggiorna lo stato della cella con nuove informazioni.
3. **Output Gate**: decide quali informazioni passare al prossimo strato della rete.

Questi meccanismi di gate permettono alle LSTM di preservare informazioni rilevanti e scartare quelle non necessarie, facilitando l'apprendimento di relazioni a lungo termine senza soffrire significativamente del problema del gradiente che svanisce.

### Applicazioni

Sia le RNN che le LSTM sono utilizzate in una vasta gamma di applicazioni:

* **Elaborazione del linguaggio naturale (NLP)**: traduzione automatica, generazione di testo, riconoscimento vocale.
* **Serie temporali**: previsioni meteorologiche, analisi di serie storiche di mercato, monitoraggio della salute.
* **Generazione di musica**: creazione di composizioni musicali basate su pattern appresi da esempi esistenti.

Le Long Short-Term Memory (LSTM) networks sono una speciale architettura di reti neurali ricorrenti (RNN) progettata per superare i limiti delle RNN standard, in particolare il problema del gradiente che svanisce, che rende difficile per le RNN apprendere dipendenze a lungo termine tra gli elementi in una sequenza di dati. Le LSTM sono dotate di meccanismi che consentono loro di mantenere sia memorie a breve termine (short-term memory) sia a lungo termine (long-term memory), rendendole particolarmente efficaci per applicazioni che richiedono la comprensione di contesti e dipendenze temporali estesi.

### Short-Term Memory

La "short-term memory" nelle LSTM si riferisce alla capacità della rete di mantenere informazioni per brevi periodi di tempo. Questo è realizzato tramite l'utilizzo di porte (gate) all'interno delle celle LSTM, che controllano il flusso di informazioni. La porta di input (input gate) e la porta di oblio (forget gate) giocano ruoli cruciali nella gestione della short-term memory:

* **Input Gate**: Decide quali valori aggiornare nello stato della cella, permettendo alla rete di aggiungere nuove informazioni alla memoria a breve termine.
* **Forget Gate**: Decide quali valori rimuovere dallo stato della cella, permettendo alla rete di scartare le informazioni non più rilevanti per le previsioni future o il contesto attuale.

### Long-Term Memory

La "long-term memory" si riferisce alla capacità della rete di mantenere informazioni per lunghi periodi di tempo. Questo è fondamentale per applicazioni come l'elaborazione del linguaggio naturale e la previsione di serie temporali, dove la comprensione del contesto o delle dipendenze temporali estese è cruciale. La long-term memory è gestita principalmente attraverso:

* **Stato della Cellula (Cell State)**: Una struttura che attraversa l'intera rete da una cella all'altra, trasportando con sé informazioni attraverso sequenze lunghe. È modificata selettivamente dalle porte della rete, che aggiungono o rimuovono informazioni, consentendo alla LSTM di preservare informazioni rilevanti per lungo tempo.

### Meccanismo di Porte

Le porte nelle celle LSTM consentono di regolare il flusso di informazioni, aggiungendo un livello di controllo che le RNN standard non possiedono. Questo meccanismo di porte include:

* **Porta di Oblio**: Controlla la quantità di informazioni precedenti da scartare.
* **Porta di Input**: Controlla l'aggiunta di nuove informazioni allo stato della cella.
* **Porta di Output**: Decide quali informazioni nello stato della cella devono essere trasferite all'output.

1. RNN => short term memory
2. LSTM => short and long term memory

## GRU

Le Gated Recurrent Units (GRU) sono un'architettura di rete neurale ricorrente introdotta da Cho et al. nel 2014. Le GRU sono state sviluppate come un'alternativa alle Long Short-Term Memory (LSTM) networks, con l'obiettivo di semplificare il modello mantenendo la capacità di catturare dipendenze temporali a lungo termine nei dati sequenziali.

### Struttura delle GRU

Le GRU semplificano la struttura delle LSTM combinando la porta di oblio e la porta di input in una singola "porta di aggiornamento" e mescolando lo stato della cella e lo stato nascosto. Ciò riduce la complessità del modello e il numero di parametri, potenzialmente migliorando l'efficienza computazionale e facilitando l'apprendimento. Le caratteristiche principali delle GRU includono:

* **Porta di aggiornamento (Update Gate)**: Determina quanto delle informazioni passate devono essere mantenute attraverso le sequenze. Funziona in modo simile alla combinazione delle porte di oblio e di input nelle LSTM, decidendo sia la quantità di informazioni passate da scartare sia quelle nuove da aggiungere.
* **Porta di reset (Reset Gate)**: Decide quanto del passato ignorare, permettendo al modello di decidere se ignorare completamente le informazioni precedenti. Questo è utile per catturare le dipendenze a breve termine all'interno delle sequenze.

### Funzionamento delle GRU

In una GRU, l'output di ogni step temporale è calcolato usando sia lo stato nascosto precedente sia l'input corrente, modulati dalle porte di aggiornamento e di reset. Questo processo consente alla GRU di mantenere informazioni rilevanti a lungo termine e scartare quelle irrilevanti, facilitando l'apprendimento di dipendenze temporali senza incorrere nel problema del gradiente che svanisce.

### Applicazioni delle GRU

Le GRU sono utilizzate in una varietà di applicazioni di elaborazione di dati sequenziali, simili alle LSTM, inclusi:

* **Elaborazione del linguaggio naturale (NLP)**: Traduzione automatica, riconoscimento vocale, generazione di testo.
* **Serie temporali**: Previsioni finanziarie, analisi del traffico, monitoraggio della salute.
* **Modellazione di sequenze**: Generazione di musica, video, e modellazione del linguaggio.

### Vantaggi delle GRU rispetto alle LSTM

* **Semplicità**: Le GRU hanno una struttura più semplice con meno porte e parametri, rendendole più facili da modellare e potenzialmente più veloci da addestrare rispetto alle LSTM.
* **Efficienza**: In alcuni casi, le GRU possono raggiungere o superare le performance delle LSTM con meno risorse computazionali.
* **Flessibilità**: La combinazione delle funzionalità delle porte in GRU offre una buona bilancia tra la cattura di informazioni a breve e a lungo termine.

Nonostante queste differenze, la scelta tra GRU e LSTM dipende spesso dal contesto specifico dell'applicazione e dai dati disponibili, con entrambe le architetture che mostrano prestazioni comparabili in molti compiti.

## Input e Output sequenziali

Nel contesto delle reti neurali, in particolare quando si parla di RNN (Recurrent Neural Networks) e delle loro varianti come LSTM (Long Short-Term Memory) e GRU (Gated Recurrent Units), i termini "one-to-one", "one-to-many", "many-to-one", e "many-to-many" descrivono diversi tipi di architetture di rete in base alla relazione tra input sequenziali e output sequenziali. Questi termini aiutano a categorizzare i modelli basati su come gestiscono le sequenze di dati in ingresso e in uscita.

### One-to-One

Questa è l'architettura più semplice e non implica la sequenzialità nei dati. Corrisponde a una rete neurale tradizionale dove a un singolo input corrisponde un singolo output. Un esempio di applicazione one-to-one è la classificazione di immagini dove ogni immagine è classificata in una categoria.

### One-to-Many

In un'architettura one-to-many, un singolo input genera una sequenza di output. Questo tipo di modello è utile in compiti come la generazione di testo basata su un'immagine, dove l'input è un'immagine singola e l'output è una sequenza di parole che descrivono l'immagine.

### Many-to-One

Le architetture many-to-one accettano una sequenza di dati come input e producono un singolo output. Questi modelli sono spesso usati in compiti di classificazione o regressione dove l'input è una sequenza temporale. Un esempio tipico è l'analisi dei sentimenti, dove una sequenza di parole (una frase o una recensione) viene classificata come avente un sentimento positivo o negativo.

### Many-to-Many

Le reti many-to-many elaborano sequenze di input in sequenze di output. Questo tipo di architettura può essere implementato in due modi: con lunghezze di input e output uguali o diverse. Un esempio del primo caso è la traduzione automatica, dove una sequenza di parole in una lingua viene convertita in una sequenza di parole in un'altra lingua. Un esempio del secondo caso è la generazione di testo dove, partendo da un prompt iniziale, il modello genera una sequenza di parole più lunga.

### Applicazioni e Esempi

* **One-to-One**: Classificazione di immagini.
* **One-to-Many**: Generazione di descrizioni da immagini.
* **Many-to-One**: Analisi dei sentimenti di recensioni di testo.
* **Many-to-Many**: Traduzione automatica da una lingua all'altra o tagging di parti del discorso in frasi.

Queste architetture permettono di progettare reti neurali adatte a una vasta gamma di applicazioni di apprendimento automatico e intelligenza artificiale, sfruttando la capacità delle RNN e delle loro varianti di gestire informazioni sequenziali e temporali.

## Encoder e Decoder (Seq2Seq)

Decoder

Encoder

Context vector

Nel contesto dell'apprendimento automatico e delle reti neurali, specialmente nelle architetture di rete neurale per l'elaborazione del linguaggio naturale (NLP) e la visione artificiale, gli encoder e i decoder sono componenti chiave in molti modelli avanzati, come le reti sequenza-a-sequenza (Seq2Seq) e le architetture di Transformer. Sono ampiamente utilizzati in applicazioni come la traduzione automatica, il riassunto automatico di testi, la generazione di descrizioni da immagini (image captioning) e molti altri compiti di generazione di sequenze.

### Encoder

Un encoder è una rete neurale che prende una sequenza di input (ad esempio, una frase in una lingua sorgente nella traduzione automatica) e la trasforma in una rappresentazione fissa, solitamente sotto forma di un vettore. Questo vettore cerca di catturare il contenuto semantico dell'intera sequenza di input. L'idea è che questa rappresentazione fissa, che a volte viene chiamata "contesto" o "stato latente", contenga tutte le informazioni necessarie dalla sequenza di input per permettere al decoder di generare l'output desiderato.

### Decoder

Il decoder è un'altra rete neurale che prende il vettore di contesto prodotto dall'encoder e lo utilizza per generare una sequenza di output, un elemento alla volta. Per esempio, in un sistema di traduzione automatica, il decoder utilizzerebbe il vettore di contesto per generare una frase nella lingua di destinazione. Il decoder inizia la generazione dell'output basandosi su un token iniziale e continua fino a che non genera un token di fine sequenza, producendo così l'intera sequenza di output basandosi sul contesto fornito dall'encoder.

### Encoder-Decoder in Modelli Seq2Seq

Nei modelli Seq2Seq, l'encoder e il decoder lavorano insieme per trasformare una sequenza di input in una sequenza di output. Tipicamente, entrambi sono costituiti da reti neurali ricorrenti (RNN), LSTM, o GRU, anche se nelle architetture più recenti come i Transformer, l'encoder e il decoder sono costruiti usando meccanismi di attenzione senza ricorrenza.

### Meccanismo di Attenzione

Il meccanismo di attenzione è una componente che, quando aggiunta ai modelli encoder-decoder, permette al decoder di "focalizzarsi" su parti specifiche della sequenza di input durante la generazione di ogni token della sequenza di output. Questo risulta particolarmente utile per migliorare la qualità della generazione di sequenze lunghe, poiché il modello non deve fare affidamento unicamente sulla rappresentazione fissa generata dall'encoder, ma può adattare dinamicamente il suo "focus" sull'input.

### Applicazioni

* **Traduzione Automatica**: Tradurre testi da una lingua all'altra.
* **Riassunto Automatico**: Generare un riassunto conciso di un testo lungo.
* **Generazione di Descrizioni da Immagini**: Creare descrizioni testuali per immagini.
* **Riconoscimento vocale**: Trasformare l'audio parlato in testo.

L'architettura encoder-decoder ha rivoluzionato il campo dell'NLP e della visione artificiale, offrendo un framework flessibile per affrontare una varietà di compiti complessi di generazione di sequenze.

Sequence to sequence paper: <https://arxiv.org/pdf/1409.3215.pdf>

Neural machine translation with attention research paper: <https://www.researchgate.net/publication/355917108_Neural_Machine_Translation_with_Attention>

Attention is all you need research paper: <https://arxiv.org/pdf/1706.03762.pdf>

## TRANSFORMER ARCHITECTURE

A diagram of a process

Description automatically generated

Decoder

Encoder

L'architettura Transformer è un modello di deep learning introdotto nel paper "Attention is All You Need" da Vaswani et al. nel 2017, che ha rivoluzionato il modo di affrontare i problemi di elaborazione del linguaggio naturale (NLP) e oltre. Questo modello si distingue per la sua capacità di gestire sequenze di dati in modo parallelo e per la sua efficienza nel catturare dipendenze a lungo termine tra gli elementi di una sequenza, grazie al meccanismo di attenzione multi-testa.

### Componenti Principali dell'Architettura Transformer

1. **Encoder**: L'encoder del Transformer è composto da una pila di NN livelli identici. Ogni livello ha due sottolivelli principali: un meccanismo di attenzione multi-testa e una rete feed-forward posizionalmente completamente connessa. Inoltre, ogni sottolivello ha una connessione residuale attorno ad esso, seguita da una normalizzazione di livello. L'encoder trasforma l'input sequenziale in una rappresentazione continua che incorpora le informazioni di tutte le posizioni.
2. **Decoder**: Similmente all'encoder, il decoder è composto da NN livelli identici. Tuttavia, oltre ai due sottolivelli presenti nell'encoder, il decoder introduce un terzo sottolivello che esegue attenzione multi-testa sui dati dell'encoder. Questo permette al decoder di focalizzarsi su parti diverse dell'input dell'encoder durante la generazione dell'output.
3. **Mecchanismo di Attenzione Multi-Testa**: Questa componente permette al modello di concentrarsi su parti diverse dell'input per ogni operazione di attenzione, migliorando la capacità del modello di gestire dipendenze a lungo raggio e comprendere il contesto.
4. **Vettore di Contesto (Context Vector)**: Nel contesto dei Transformers, il vettore di contesto è una rappresentazione densa generata dall'encoder che cattura le informazioni di tutto l'input. I decoder possono utilizzare questi vettori per produrre l'output sequenziale.
5. **Feed-Forward Networks**: Ogni livello sia nell'encoder che nel decoder contiene una rete feed-forward che processa la sequenza posizione per posizione indipendentemente dalle altre.
6. **Embeddings e Posizional Encoding**: Gli embeddings convertono gli identificatori univoci (come le parole) in vettori densi. Il Transformer aggiunge informazioni posizionali a questi embeddings per mantenere la nozione dell'ordine delle parole nell'input.

### Funzionamento Generale

Il processo inizia con l'input che passa attraverso l'encoder, che lo trasforma in un vettore di contesto. Questo vettore viene poi passato al decoder, che lo utilizza per generare l'output sequenziale. Il meccanismo di attenzione in ogni livello dell'encoder e del decoder consente al modello di considerare l'intera sequenza di input (o il contesto) quando elabora ogni parola, migliorando significativamente la qualità della generazione del testo o della traduzione rispetto ai modelli precedenti.

### Impatto e Applicazioni

L'architettura Transformer ha avuto un impatto enorme sul campo dell'NLP, portando allo sviluppo di modelli come BERT, GPT, e altri che hanno stabilito nuovi standard di prestazione in una vasta gamma di compiti di NLP, come la comprensione del testo, la generazione di testo, la traduzione automatica e l'analisi del sentimento. La sua influenza si estende anche ad altri campi, come la visione artificiale, dimostrando la versatilità e l'efficacia dell'architettura Transformer

## Discriminative model vs Generative models

A diagram of a model

Description automatically generated

I modelli discriminativi e i modelli generativi rappresentano due approcci fondamentali per la modellazione dei dati e la previsione.

### Modelli Discriminativi

I modelli discriminativi, noti anche come modelli condizionali, mirano a modellare la decisione tra diverse classi. Si concentrano sulla determinazione del confine tra le classi direttamente dall'input agli output. In altre parole, un modello discriminativo impara la frontiera di decisione che distingue tra le classi di output date le caratteristiche di input. L'obiettivo è di imparare P(y∣x)P(y∣x), cioè la probabilità condizionale della classe yy dato l'input xx.

**Esempi**:

* Regressione logistica
* Support Vector Machines (SVM)
* Alberi di decisione
* Reti neurali per la classificazione

**Caratteristiche**:

* Direttamente focalizzati sulla previsione dell'output.
* Generalmente più semplici da addestrare in contesti di classificazione.
* Non modellano esplicitamente la distribuzione degli input.

### Modelli Generativi

I modelli generativi, d'altra parte, cercano di modellare come i dati vengono generati, imparando la distribuzione congiunta di input e output, P(x,y)P(x,y). Da questa distribuzione congiunta, è possibile derivare la distribuzione condizionale P(y∣x)P(y∣x), che può essere usata per la classificazione o altre inferenze. I modelli generativi non solo imparano il confine tra le classi, ma cercano anche di imparare la struttura sottostante dei dati stessi.

**Esempi**:

* Gaussian Mixture Models (GMM)
* Naive Bayes
* Hidden Markov Models (HMM)
* Generative Adversarial Networks (GAN)
* Variational Autoencoders (VAE)

**Caratteristiche**:

* Capacità di generare nuovi dati simili a quelli osservati.
* Utile per compiti come la sintesi di dati, la generazione di immagini, il completamento di immagini, etc.
* Più complessi da addestrare rispetto ai modelli discriminativi a causa della necessità di modellare la distribuzione degli input.

### Differenze Chiave

* **Obiettivo**: I modelli discriminativi mirano a distinguere tra classi di output, mentre i modelli generativi mirano a catturare come i dati vengono generati.
* **Complessità di Addestramento**: I modelli generativi sono tipicamente più complessi da addestrare poiché richiedono la modellazione della distribuzione degli input oltre alla separazione delle classi.
* **Capacità di Generazione**: Solo i modelli generativi possono essere utilizzati per generare nuovi dati che assomiglino agli input originali.

# What is LLMs?

Large Language Models (LLMs) sono dei modelli di Machine Learning che usano algoritmi di deep learning per capire e processare il linguaggio naturale. Questi modelli sono istruiti con enormi quantità di dati testuali per imparare i patterns ricorrenti e le relazioni tra entità nel linguaggio.

Con questo linguaggio è possibile eseguire task come :

* Text to text generations
* Text to image generations
* Image to text generations

A screenshot of a computer

Description automatically generated

## Milestones in LLMs

* BERT: Bidirectional Encoder Generation from Tranformers sviluppato da google
* GPT: Generative Pretrained Transformer sviluppato da OpenAI
* XLM: Cross-lingual Language Mopdel Pretrainig by Guillaume Lample, Alexis Conneau
* T5: the Text-to\_Text Transformer creato da Google AI
* Megatron:Transformer molto potentate creato dall’Applied Research team at NVIDIA
* M2M-100: Multilinhual encoder decoder (seq to seq) modello sviluppato da Facebook

## Transformer Tree

A diagram of a computer program

Description automatically generated

## OpenAI based Models

A screenshot of a computer

Description automatically generated

## Other Open source models

* Bloom
* Llama 2
* PaLM
* Falcon
* Claude
* MPT-30B
* Stablelm
* ……..

## Casi d’uso:

* Text Classification
* Text sommarization
* Text generation
* Conversation like a chatbot, question answering
* Speech recognition and speech identification
* Spelling corrector
* ……

## Come è allenato ChatGPT?

Internamente usa un LLM come gpt-3.5 o gpt-4.

Per il suo training sono stati utilizzati un enorme quantità di dati disponibili su tutta le rete internet

1. Generative pre-training
2. Supervised fine-tuning
3. Reinforcement learning

Open source models : <https://huggingface.co/models>

Alternativa a GPT gratuita : <https://www.ai21.com/studio>

## F Tranformation e Fine tuning

Universal Language Model Fine-tuning for Text Classification paper: <https://arxiv.org/pdf/1801.06146.pdf>

La trasformazione FF e il fine tuning sono due concetti importanti nel campo del deep learning, che gioca un ruolo cruciale nello sviluppo di modelli di apprendimento automatico avanzati. Entrambi sono usati per migliorare le prestazioni dei modelli su compiti specifici, ma si applicano in maniere diverse.

### Trasformazione FF

La trasformazione FF (o trasformazione caratteristica) è un processo attraverso il quale i dati di input vengono trasformati o modificati prima di essere alimentati a un modello di deep learning. L'obiettivo è quello di rendere i dati più adatti all'apprendimento da parte del modello, migliorando così l'efficienza e l'efficacia dell'addestramento. Questa trasformazione può includere vari passaggi come la normalizzazione, la standardizzazione, l'applicazione di tecniche di augmentation dei dati (come la rotazione, il taglio, o lo zoom per le immagini), o l'estrazione di caratteristiche più significative che aiutano il modello a imparare meglio dai dati.

Esempio di codice per l'augmentation dei dati

from tensorflow.keras.preprocessing.image import ImageDataGenerator

*# Creazione di un generatore di dati con augmentation*

*datagen = ImageDataGenerator(*

*rotation\_range=20,*

*width\_shift\_range=0.2,*

*height\_shift\_range=0.2,*

*shear\_range=0.2,*

*zoom\_range=0.2,*

*horizontal\_flip=True,*

*fill\_mode='nearest'*

*)*

*# Questo generatore può essere usato per addestrare un modello*

### Fine Tuning

Il fine tuning, invece, è una tecnica di trasferimento dell'apprendimento che consiste nell'adattare un modello pre-addestrato su un nuovo compito con dati simili ma non identici a quelli usati per l'addestramento originale. In pratica, si inizia con un modello che è stato addestrato su un grande dataset (come ImageNet per la classificazione delle immagini) e si continua l'addestramento (o "fine tuning") su un dataset più piccolo specifico per il compito di interesse. Questo processo può includere l'aggiustamento di alcuni dei parametri del modello, il congelamento dei pesi di alcuni strati per prevenire la loro modifica durante l'addestramento supplementare, o la modifica dell'architettura del modello per meglio adattarsi al nuovo compito.

*from tensorflow.keras.applications import VGG16*

*from tensorflow.keras.models import Model*

*from tensorflow.keras.layers import Dense, Flatten*

*# Caricamento del modello VGG16 pre-addestrato escludendo gli strati superiori*

*base\_model = VGG16(weights='imagenet', include\_top=False, input\_shape=(224, 224, 3))*

*# Congelamento dei pesi di tutti gli strati del modello base*

*for layer in base\_model.layers:*

*layer.trainable = False*

*# Aggiunta di nuovi strati personalizzati per il nuovo compito*

*x = Flatten()(base\_model.output)*

*x = Dense(1024, activation='relu')(x)*

*predictions = Dense(10, activation='softmax')(x)*

*# Definizione del modello finale per il fine tuning*

*model = Model(inputs=base\_model.input, outputs=predictions)*

*# Ora il modello può essere compilato e addestrato sul nuovo dataset*

In entrambi i casi, l'obiettivo è massimizzare le prestazioni del modello di deep learning ottimizzando il modo in cui elabora e apprende dai dati.

# Open-AI

Anaconda

Python

Creare un virtual environement su Anacoda con il comando

-*conda create -n testingopenai python=3.11*

*-conda activate testingopenai*

*- pip list =>lista librerie installate*

*-* *pip install jupyter notebook*

*-*jupyter notebook

[*https://github.com/langchain-ai/langchain*](https://github.com/langchain-ai/langchain)

*import langchain*

*from langchain.llms import OpenAI*

*client = OpenAI(openai\_api\_key=mkey)*

*#zero short prompt*

*Prompt = “can ytou tel me total number of countries in Asia? Can you give me top 10Country name”*

*Print (client.predict(prompt).strip())*

OpenAI

request

<https://serpapi.com/dashboard>