Università del Piemonte Orientale

Dipartimento di Scienze e Innovazione Tecnologica

**Corso di Laurea in Informatica**

Relazione per la prova finale

# RETI NEURALI PROFONDE PER LA GENERAZIONE DI GRAFI: REVIEW DELLO STATO DELL'ARTE


**Tutore interno:**

Prof. Giorgio Leonardi

**Candidato**:

Daniele Gambera


Firmato digitalmente da Giorgio Leonardi
Data: 25.09.2024 23:57:34 CEST
Organizzazione: UNIVERSITA' DEGLI STUDI DEL PIEMONTE ORIENTALE/01943490027

Daniele Gambera

**Anno Accademico 2023/24**

# Sommario

# Abstract

I grafi sono utilizzati per la rappresentazione di dati e di conoscenza e sono ormai un formalismo di riferimento per descrivere gli oggetti e le relazioni tra questi. I recenti progressi nei modelli generativi profondi per la generazione di grafi rappresentano un importante passo avanti per migliorare la fedeltà dei grafi generati e aprono la strada a nuovi tipi di applicazioni.

Grazie ai progressi nell'apprendimento profondo basato su grafi, sono recentemente emerse nuove applicazioni che spaziano dalla scoperta di nuove strutture molecolari alla modellazione delle reti sociali.

Questo lavoro si pone l'obiettivo di fornire una panoramica estensiva della letteratura nel campo dei modelli generativi profondi per la generazione di grafi.

# 1 Introduzione

Il rapido avanzamento delle tecnologie di raccolta e archiviazione ha portato a un'enorme mole di dati da elaborare. In settori come biologia, chimica, farmaceutica, reti sociali e di conoscenza, le relazioni tra le entità di questi dati sono cruciali per ottenere previsioni più accurate e informazioni di maggiore valore. Una delle modalità più efficaci per rappresentare queste relazioni è attraverso la struttura a grafo, il che ha reso l'analisi di questi un'area di ricerca di grande interesse.

Recentemente, i progressi nell'apprendimento automatico applicato ai grafi hanno fatto passi da gigante, concentrandosi in particolare sull'apprendimento delle loro rappresentazioni. Questo processo mira a trovare embedding appropriati per nodi, archi e interi grafi in spazi continui a bassa dimensione. Questi embedding sono utili per compiti successivi come la visualizzazione, il clustering, la classificazione dei nodi e la previsione dei collegamenti.

Un altro campo emergente è la generazione di grafi, che ha l'obiettivo di creare nuove strutture di grafi con proprietà specifiche. Nonostante queste ricerche affondino le loro radici negli anni 60, solo di recente, soprattutto grazie ai progressi nel deep learning, hanno acquisito nuova rilevanza in vari domini come, ad esempio, immagini, testi e rappresentazione di conoscenza. Questi modelli mirano a generare nuovi campioni da una distribuzione simile a quella dei dati di addestramento e, a differenza dei metodi tradizionali, possono apprendere direttamente dai dati, eliminando la necessità di procedure progettate manualmente. I moderni modelli generativi profondi, infatti, superano i limiti dei metodi tradizionali, che utilizzano processi ingegnerizzati manualmente e non erano in grado di catturare le complesse dipendenze tra i grafi.

La generazione di grafi basata su deep learning, nota come *Deep Graph Generators* (**DGGs**), sta attirando sempre più l'attenzione dei ricercatori, con applicazioni che spaziano dalla scoperta di nuove strutture molecolari alla modellazione delle reti sociali.

Ho condotto un'indagine approfondita sui DGG, suddividendo gli approcci esistenti in due categorie principali: modelli generativi condizionati e non-condizionati. Per ciascuna categoria, descriverò dettagliatamente i metodi, confrontandoli da diversi punti di vista e fornendo una sintesi delle loro caratteristiche principali.

## 1.1 Notazione

Un grafo può essere rappresentato formalmente come G = (V, E), dove V è l'insieme dei nodi e E è l'insieme degli archi, con |V| = n che rappresenta il numero di nodi e |E| = m che rappresenta il numero di archi. N indica la dimensione massima del grafo nel dataset.

Dato un grafo, esistono n! possibili ordinamenti dei nodi. Se scegliamo un ordinamento specifico pi, il grafo può essere rappresentato dalla corrispondente matrice di adiacenza $(A_\pi \in \mathbb{R}^{n\times n})$.

Oltre alla rappresentazione tramite matrice di adiacenza, il grafo può anche essere rappresentato mediante sequenze dei suoi nodi o archi ordinati, indicate rispettivamente come $(S_{\mathrm{node},\pi})e(S_{\mathrm{edge},\pi})$.

## 1.2 Sfide

Lo sviluppo di modelli generativi profondi per grafi presenta diverse sfide uniche, che includono:

1) Rappresentazioni Non Univoche

   Un grafo con n nodi può essere rappresentato da un numero massimo di n! matrici di adiacenza equivalenti, ciascuna corrispondente a un diverso ordinamento arbitrario dei nodi. Questa variabilità nelle rappresentazioni rende difficile per i modelli calcolare la distanza tra i grafi generati e quelli reali durante l'addestramento. Di conseguenza, è necessario progettare un ordinamento predefinito dei nodi o una funzione obiettivo che tenga conto della loro permutazione.

2) Dipendenze Complesse

I nodi e gli archi di un grafo presentano dipendenze e relazioni complesse. Ad esempio, la probabilità di connessione tra due nodi aumenta se essi condividono vicini comuni. Pertanto, la generazione di ciascun nodo o arco non può essere trattata come un evento indipendente.

3) Grandi Spazi di Output

   Per generare un grafo con n nodi, il modello generativo potrebbe dover produrre $n^2$ valori per specificare la struttura del grafo, rendendo il compito oneroso, specialmente per grafi di grandi dimensioni. Grafi reali, come quelli utilizzati nelle reti di citazioni oppure in quelle sociali, possono contenere milioni di nodi. È quindi cruciale che i modelli generativi siano in grado di scalare e gestire la complessità degli spazi di output.

4) Oggetti Discreti per Natura

   Le tecniche standard di machine learning non sono sempre ottimali per i grafi e richiedono alcuni adattamenti, come ad esempio l'algoritmo di backpropagation che non è direttamente applicabile ai grafi. Per affrontare questo problema, è comune rappresentare i grafi mediante vettori o matrici, sebbene ricostruire i grafi da queste rappresentazioni continue sia una sfida. La decodifica degli oggetti discreti del grafo (nodi e archi) da spazi continui può richiedere metodi diversi, come la generazione sequenziale dei nodi o la generazione simultanea della matrice di adiacenza.

5) Valutazione delle Proprietà Implicite

   Valutare i grafi generati è una questione critica ma difficile, dovuta alle proprietà uniche di questi. I metodi esistenti utilizzano diverse metriche di valutazione, come il calcolo della distanza della distribuzione statistica tra i grafi nel set di test e quelli generati, o l'uso di classificatori per determinare se i grafi generati appartengono o meno alla stessa distribuzione dei grafi di addestramento. È importante rivedere e selezionare sistematicamente le metriche più appropriate in base ai loro punti di forza e limiti.

6) Diversi Requisiti di Validità

La modellazione e comprensione della generazione di grafi tramite deep learning si applica a una vasta gamma di problematiche interdisciplinari, come la progettazione di molecole, la modellazione della struttura delle proteine e il parsing semantico. Ogni applicazione ha requisiti specifici per la validità dei grafi generati.

7) Black-box con Bassa Affidabilità

   I metodi di generazione di grafi basati su deep learning spesso operano come scatole nere, con scarsa interpretabilità e affidabilità. Migliorare l'interpretabilità dei modelli generativi profondi è cruciale per aprire la "scatola nera" del processo di generazione e facilitare l'adozione in ambiti che richiedono alta sensibilità e affidabilità, come la sanità e la guida automatica.

## 1.3 Relazione con i Modelli Generativi Profondi

Questi modelli offrono un modo molto efficiente per analizzare e comprendere i dati non etichettati, basandosi sull'idea di catturare la distribuzione probabilistica interna che genera una classe di dati, al fine di generare nuovi dati simili. Approcci emergenti come le generative adversarial networks (GANs), i variational auto-encoders (VAEs), le reti neurali ricorrenti generative e le loro numerose varianti, hanno dimostrato risultati impressionanti in una vasta gamma di applicazioni.

# 2 Modelli generativi profondi non-condizionati

L'obiettivo della generazione profonda incondizionata di grafi è apprendere la distribuzione $p_{model}(G)$ basata su un insieme di grafi realistici osservati. A seconda del processo di generazione adottato, i metodi possono essere suddivisi in due categorie principali: **Generazione Sequenziale** e **Generazione One-shot**.

## 2.1 Generazione Sequenziale

La Generazione Sequenziale si basa su un approccio in cui i nodi e gli archi del grafo vengono generati in sequenza, uno dopo l'altro. Questo metodo permette di prendere decisioni locali in modo efficiente, utilizzando le informazioni generate in precedenza per influenzare le scelte successive. Tuttavia, la generazione sequenziale può incontrare difficoltà nel preservare le dipendenze a lungo termine tra i vari elementi del grafo, rendendo complicato incorporare proprietà globali.

Un'altra limitazione significativa dei metodi di generazione sequenziale risiede nella loro dipendenza da un ordine prestabilito della sequenza di generazione. Questo significa che l'ordine in cui i nodi e gli archi vengono generati può influenzare il risultato finale, lasciando aperta la questione di come gestire le permutazioni dei nodi per ottimizzare la qualità dei grafi generati.

### 2.1.1 Generazione basata su sequenze di nodi

L'approccio più intuitivo per la generazione sequenziale di grafi consiste nel rappresentare il grafo come una sequenza di nodi. In questo contesto, ogni nodo viene generato insieme ai suoi archi associati, condizionando il processo sui nodi e gli archi precedentemente generati. In pratica, se un grafo contiene N nodi, la generazione di ciascun nodo può comportare la creazione di un numero massimo di N archi, il che porta a una complessità computazionale complessiva di $O(N^2)$. Questo metodo è particolarmente adatto per grafi densi, dove la maggior parte dei nodi è connessa ad un gran numero di altri nodi.

**Metodi Basati su Sequenze di Nodi**

I metodi basati su sequenze di nodi affrontano la generazione di grafi trattando la costruzione del grafo come un processo sequenziale, in cui i nodi e i relativi archi vengono aggiunti uno alla volta, seguendo un ordine prestabilito. Il processo può essere suddiviso in due fasi principali: generazione del nodo e generazione degli archi associati.

**Quadro Generale**

In questo approccio, un grafo G viene modellato come una sequenza basata su un ordine predefinito *pi* sui nodi. Ogni unità $s_i$ nella sequenza di componenti è rappresentata come una tupla $s_i = (v_{\{pi\ i\}}, e_{\{i,j\}\{j<i\}})$. Questo significa che, in ogni passo della generazione, il metodo aggiunge un nodo $v_{\{pi\ i\}}$ e tutti i suoi archi associati $e_{\{i,j\}\{j<i\}}$, che lo collegano ai nodi già esistenti.

**Fasi della Generazione**

1. **Generazione del Nodo**:

   In questa fase, viene generato un nodo $(v_{\pi i})$, condizionato dal grafo parziale già generato $(G_i)$. Questo può essere interpretato come l'apprendimento della distribuzione condizionata $(p(v_{\pi i}|G_i))$, che rappresenta la probabilità di generare il nodo $(v_{\pi i})$ dato il grafo $(G_i)$ fino a quel punto.

2. **Generazione degli Archi Associati**:

   Dopo la generazione del nodo, viene generato il set di archi $(\{e_{i,j}\}_{j<i})$ che connettono $(v_{\pi i})$ ai nodi precedenti. Esistono due approcci principali per questa fase: **generazione collettiva** e **generazione progressiva** degli archi.

*Generazione Collettiva degli Archi Associati*

Un approccio alternativo è la generazione collettiva, tutti gli archi tra il nuovo nodo $(v_{\pi i})$ e i nodi esistenti $(v_{\pi<i})$ vengono generati simultaneamente in un singolo passo. La maggior parte dei lavori in questo ambito si concentra sulla

previsione del vettore di adiacenza $(A_{\pi i,\cdot})$, che rappresenta tutti gli archi potenziali tra il nodo $(v_{\pi i})$ e gli altri nodi già generati.

L'unità $(s_i)$ può essere così rappresentata come $(s_i = (v_{\pi i}, A_{\pi i,\cdot}))$, e l'obiettivo diventa apprendere la distribuzione:

$$p(V_\pi, A_\pi) = \prod_{i=1}^{N} p(v_{\pi i} | v_{\pi<i}, A_{\pi<i,\cdot}) \cdot p(A_{\pi i,\cdot} | v_{\pi \leq i}, A_{\pi<i,\cdot})$$

dove $(v_{\pi<i})$ rappresenta i nodi generati prima di $(v_{\pi i})$ e $(A_{\pi<i,\cdot})$ rappresenta i vettori di adiacenza generati prima di $(A_{\pi i,\cdot})$.

**Vantaggi e Limiti:**

- **Vantaggi**: Questo approccio è semplice da implementare e particolarmente adatto per grafi densi, dove ogni nuovo nodo è probabilmente connesso a molti nodi esistenti.
- **Limiti**: La complessità temporale è $(O(N^2))$, dove $(N)$ è il numero di nodi, rendendolo poco efficiente per grafi sparsi.

*Generazione Progressiva degli Archi Associati*

Per migliorare l'efficienza, soprattutto nei grafi sparsi, viene adottata la **generazione progressiva** degli archi. In questo caso, gli archi associati a un nuovo nodo $(v_{\pi i})$ vengono generati uno alla volta, in una sequenza di passi. Ogni passo include due azioni principali:

1. **addEdge**: Determina se aggiungere un arco al nodo $(v_{\pi i})$.
2. **selectNode**: Se si decide di aggiungere un arco, questa funzione seleziona il nodo $(v_{\pi j})$ a cui connettere $(v_{\pi i})$.

Ad esempio, per determinare il nodo $(v_{\pi j})$ a cui connettere $(v_{\pi i})$, si può calcolare un punteggio $(m_{\pi i,j})$ per ogni nodo esistente $(v_{\pi j})$ usando la funzione $(\mathrm{selectNode})$, che viene poi normalizzato tramite una funzione softmax per ottenere una distribuzione di probabilità sui nodi:

$$m_{\pi i,j} = f_{\text{selectNode}}(h_{\pi v_i}, h_{\pi v_j})$$

$$p(e_{i,j}|v_{\pi<i}, \{e_{<i,j}\}_{j<i}) = \text{softmax}(m_{\pi i,j})$$

La procedura viene iterata fino a quando la funzione addEdge segnala che non ci sono più archi da aggiungere per il nodo $(v_{\pi i})$.

**Vantaggi e Limiti:**

- **Vantaggi**: Riduce la complessità temporale, rendendo questo metodo più adatto per grafi sparsi, dove molti nodi non hanno connessioni.
- **Limiti**: Sebbene sia più efficiente per grafi sparsi, il processo sequenziale potrebbe essere più complesso da implementare e richiede un'accurata gestione delle dipendenze tra gli archi generati.

### 2.1.2 Generazione Basata su Sequenze di Archi

La generazione avviene creando gli archi uno alla volta, insieme ai nodi terminali associati, condizionando sempre il processo sugli archi già esistenti. La complessità di questo metodo è proporzionale al numero di archi nel grafo, ossia $(O(|E|))$, rendendolo più efficiente per la generazione di grafi sparsi, dove $(|E|)$ (numero di archi) è molto minore rispetto a $(N^2)$.

**Metodi Basati su Sequenze di Archi**

I metodi basati su sequenze di archi generano un grafo rappresentandolo come una sequenza di archi, con ogni arco creato insieme ai suoi nodi terminali in ogni passaggio. Questo approccio suddivide la generazione del grafo in unità che rappresentano coppie di nodi connessi da un arco e i rispettivi attributi.

**Quadro Generale**

In questo contesto, un grafo $(G)$ viene modellato come una sequenza di archi. Ogni unità della sequenza è rappresentata da una tupla $(s_i = (\alpha(u), \alpha(v), F_u, F_v, E_{u,v}))$, dove:

- $(\alpha(u))e(\alpha(v))$ sono indici dei nodi $(u)$ e $(v)$, rispettivamente.
- $(F_u)e(F_v)$ sono gli attributi associati ai nodi $(u)$ e $(v)$.
- $(E_{u,v})$ è l'attributo dell'arco che collega i nodi $(u)$ e $(v)$.

Questa rappresentazione consente di modellare il grafo come una sequenza di passi in cui, ad ogni passo, viene generato un arco insieme ai suoi due nodi terminali e ai relativi attributi.

**Approcci per la Generazione di un'Arco**

*Basato sull'Indipendenza*

Uno degli approcci principali per generare un'unità $(s_i)$ è basato sull'assunzione di indipendenza tra $(\alpha(u))$ e $(\alpha(v))$. In questo caso, si assume che gli indici dei nodi $(u)$ e $(v)$ siano indipendenti l'uno dall'altro e si utilizza un

algoritmo di ricerca in profondità (DFS) come funzione di ordinamento $(\alpha(\cdot))$ per costruire un indice canonico dei nodi del grafo.

La distribuzione condizionale per generare ogni arco del grafo $(G)$ può essere formalizzata come:

$$p(s_i|s_{<i}) = p((\alpha(u), \alpha(v), F_u, F_v, E_{u,v})|s_{<i}) = p(\alpha(u)|s_{<i})p(\alpha(v)|s_{<i})p(F_u|s_{<i})p(F_v|s_{<i})p(E_{u,v}|s_{<i})$$

dove $(s_{<i})$ si riferisce agli archi e ai nodi già generati.

*Basato su LSTM*

Per implementare questo modello, viene utilizzata una LSTM (Long Short-Term Memory) personalizzata, che prevede:

1. **Funzione di Transizione** $(f_{\text{trans}})$: Trasferisce lo stato nascosto dall'ultimo passo a quello corrente.
2. **Funzione di Embedding** $(f_{\text{emb}})$: Incorpora il grafo già generato in rappresentazioni latenti.
3. **Cinque Funzioni di Output**: Queste modellano le cinque componenti della distribuzione condizionale $(\alpha(u))$, $(\alpha(v))$, $(F_u)$, $(F_v)$, $(E_{u,v})$.

$$h_G^{(i)} = f_{\text{trans}}(h_G^{(i-1)}, f_{\text{emb}}(s_{i-1}))$$

$$\alpha(u) \sim \text{Cat}(\theta_{\alpha(u)}); \quad \theta_{\alpha(u)} = f_{\alpha(u)}(h_G^{(i)})$$

$$\alpha(v) \sim \text{Cat}(\theta_{\alpha(v)}); \quad \theta_{\alpha(v)} = f_{\alpha(v)}(h_G^{(i)})$$

$$F_u \sim \text{Cat}(\theta_{F_u}); \quad \theta_{F_u} = f_{F_u}(h_G^{(i)})$$

$$F_v \sim \text{Cat}(\theta_{F_v}); \quad \theta_{F_v} = f_{F_v}(h_G^{(i)})$$

$$E_{u,v} \sim \text{Cat}(\theta_{E_{u,v}}); \quad \theta_{E_{u,v}} = f_{E_{u,v}}(h_G^{(i)})$$

dove $(s_{i-1})$ rappresenta la tupla generata al passo precedente ed è rappresentata dalla concatenazione di tutte le componenti nella tupla. $(h_G^{(i)})$ è uno stato nascosto a livello di grafo della LSTM che codifica lo stato del grafo generato fino al passo $(i)$.

Le uscite delle cinque funzioni $(f_{\alpha(u)})$, $(f_{\alpha(v)})$, $(f_{F_u})$, $(f_{F_v})$, $(f_{E_{u,v}})$ modellano la distribuzione delle cinque componenti della nuova tupla di arco, che sono parametrizzate rispettivamente da cinque vettori $(\theta_{\alpha(u)})$, $(\theta_{\alpha(v)})$, $(\theta_{F_u})$, $(\theta_{F_v})$, $(\theta_{E_{u,v}})$. Infine, le componenti della nuova tupla di arco vengono campionate dalle cinque distribuzioni apprese.

**Vantaggi e Limiti**

- **Vantaggi**: Questo approccio è particolarmente efficace nel gestire grafi complessi e sparsi, dove gli archi hanno attributi significativi e le relazioni tra i nodi devono essere modellate accuratamente.
- **Limiti**: L'assunzione di indipendenza tra $(\alpha(u))$ e $(\alpha(v))$ potrebbe non essere realistica in tutti i contesti, specialmente quando esistono forti dipendenze tra i nodi.

### 2.1.3 Generazione Basata su Sequenze di Motivi

Sebbene gli approcci basati su nodi e archi siano efficaci nel catturare le relazioni binarie, essi spesso non riescono a modellare adeguatamente le relazioni di ordine superiore, come i triangoli nelle reti sociali o i gruppi funzionali nei grafi molecolari. Per affrontare questa limitazione, sono stati sviluppati metodi che rappresentano i grafi come sequenze di motivi, ovvero piccoli sottografi ricorrenti che catturano relazioni complesse tra i nodi. In ogni passaggio della generazione, un intero motivo, composto da più nodi e archi, viene generato simultaneamente. Questo approccio non solo migliora l'efficienza computazionale, ma consente anche di preservare le strutture locali complesse all'interno del grafo.

**Metodi Basati su Sequenze di Motivi**

I metodi basati su sequenze di motivi rappresentano un grafo $(G)$ come una sequenza di motivi grafici $(\mathrm{Seq}(G) = \{C_1, \ldots, C_M\})$, dove ogni blocco di nodi e archi che costituisce ciascun motivo grafico $(C_i)$ viene generato in ogni passaggio. A ogni passo, un nuovo motivo grafico $(C_i)$ viene generato in base al grafo attuale $(G_i)$ e successivamente collegato a $(G_i)$.

**Quadro Generale**

Il concetto chiave di questo approccio è la rappresentazione di un grafo come una sequenza di motivi, dove ogni motivo può essere considerato come un sottografo che viene generato e collegato al grafo esistente in un singolo passaggio. Questo tipo di rappresentazione offre un modo più compatto e naturale di catturare le relazioni complesse tra gruppi di nodi e archi rispetto ai metodi che si concentrano sulla generazione di singoli nodi o archi.

Uno dei principali problemi che emergono nei metodi basati sui motivi è il collegamento del nuovo motivo grafico $(C_i)$ al grafo esistente $(G_i)$. Esistono infatti numerose modalità per collegare due sottografi, e la scelta della strategia di collegamento dipende fortemente dalla definizione dei motivi grafici stessi.

*Grafi Domain-Agnostic*

Sono grafi che non richiedono conoscenze specifiche del dominio. In questo contesto, gli archi associati a tutti i nodi in $(C_i)$ e il loro collegamento a $(G_i)$ avviene sulla base di previsioni, che seguono una logica più generica e adattabile a diversi tipi di grafi.

*Grafi Domain-Specific*

Per i grafi che richiedono conoscenze specifiche del dominio, i motivi grafici sono definiti e collegati sulla base di tali conoscenze pregresse. Un esempio tipico è la generazione di strutture molecolari, dove i motivi chimici, come gli anelli o gruppi funzionali, sono definiti e utilizzati per costruire la struttura molecolare in modo coerente con le regole chimiche.

**Vantaggi e Limiti**

- **Vantaggi**: I metodi basati su sequenze di motivi offrono una rappresentazione più naturale e efficiente per catturare relazioni complesse all'interno di un grafo, specialmente in contesti dove tali motivi sono ricorrenti o strutturati (es. chimica molecolare).
- **Limiti**: La definizione e il collegamento dei motivi possono risultare complessi, soprattutto in contesti domain-agnostic, dove la mancanza di una chiara struttura o conoscenza specifica può rendere difficile la scelta delle strategie di collegamento e la definizione dei motivi stessi.

### 2.1.4 Generazione Basata su Sequenze di Regole

Infine, per specifiche applicazioni dove è fondamentale rispettare determinati vincoli o grammatiche (come nei linguaggi di programmazione o nella modellazione molecolare), sono stati sviluppati metodi che basano la generazione su sequenze di regole predefinite. In questi approcci, un grafo viene costruito seguendo una sequenza di regole che determinano la struttura e le proprietà del grafo. Questo assicura che il grafo generato sia valido secondo i criteri specifici del dominio applicativo.

**Metodi Basati su Sequenze di Regole**

I metodi basati su sequenze di regole si concentrano sulla generazione di grafi attraverso l'applicazione sequenziale di regole di produzione o comandi. Questo approccio è particolarmente utile quando il grafo da generare deve rispettare vincoli stringenti o seguire una grammatica predefinita, come accade nella generazione di molecole o altri oggetti strutturati che richiedono la conservazione di proprietà fondamentali.

**Quadro Generale**

L'idea alla base di questi metodi è quella di convertire il processo di generazione del grafo in una sequenza di regole che guidano la costruzione del grafo stesso. Ad esempio, nel caso della generazione di molecole, è essenziale rispettare proprietà chimiche fondamentali, come la conservazione della carica, che influiscono sulla validità dei nodi e degli archi all'interno del grafo molecolare. Per garantire la correttezza e la validità dei grafi generati, si utilizza una grammatica libera dal contesto (Context-Free Grammar, CFG) che descrive formalmente le regole e le restrizioni della struttura molecolare.

In questo contesto, il processo di generazione del grafo può essere visto come la costruzione di un albero di parsing che rappresenta la struttura molecolare discreta. L'albero di parsing viene generato seguendo una serie di regole predefinite, e può essere rappresentato come una sequenza di regole ordinate in modo specifico.

**GrammerVAE**

Uno degli approcci più noti in questa categoria è il Grammar Variational Auto Encoder (GrammerVAE), che si focalizza sulla generazione di alberi di parsing per rappresentare oggetti discreti, come molecole o espressioni aritmetiche. Il processo avviene nel seguente modo:

- **Generazione dell'Albero di Parsing**: Viene creato un albero di parsing che descrive l'oggetto discreto utilizzando una grammatica specifica, come la grammatica SMILES per le molecole.
- **Codifica delle Regole**: L'albero di parsing è scomposto in una sequenza di regole di produzione seguendo un attraversamento pre-ordinato dei rami dell'albero, da sinistra a destra.
- **Vettorizzazione One-Hot**: Le regole di produzione vengono convertite in vettori one-hot, dove ogni dimensione del vettore rappresenta una regola specifica nella grammatica.
- **Codifica e Decodifica**: Un autoencoder viene utilizzato per mappare l'albero di parsing in un vettore continuo $(z)$. Durante la fase di decodifica, il vettore $(z)$ viene passato attraverso una rete neurale ricorrente (RNN) che produce una serie di vettori logit. Ogni logit corrisponde a una probabilità non normalizzata per una regola di produzione specifica. Il modello genera gli alberi di parsing in direzione top-down, espandendo progressivamente l'albero attraverso le regole di produzione.

**SD-VAE (Syntax-Directed Variational Autoencoder)**

Mentre la grammatica libera dal contesto (CFG context free grammar) garantisce la generazione di oggetti sintatticamente validi, essa non può assicurare la validità semantica dell'oggetto generato. Per risolvere questo problema, è stato introdotto il Syntax-Directed Variational Autoencoder (SD-VAE):

- **Vincoli Sintattici e Semantici**: SD-VAE integra una componente di restrizione semantica nel processo di generazione, oltre ai vincoli sintattici imposti dalla CFG. Questo permette al modello di produrre oggetti che sono validi sia dal punto di vista sintattico che semantico.

- **Generazione dell'Albero Sintattico**: Come nel GrammarVAE, il modello genera un albero sintattico, ma con ulteriori restrizioni per garantire che ogni passo del processo di generazione rispetti le regole semantiche richieste.

**Vantaggi e Limiti**

- **Vantaggi**: I metodi basati su sequenze di regole offrono un controllo fine sulla generazione del grafo, garantendo che i grafi risultanti siano validi e rispettino vincoli sintattici e semantici. Questo è particolarmente utile in applicazioni che richiedono conformità a regole strutturali rigorose, come nella chimica computazionale o nella generazione di linguaggi formali.
- **Limiti**: Il principale svantaggio di questi metodi è la complessità intrinseca nella definizione e gestione delle regole, che può richiedere una profonda conoscenza del dominio specifico e un'implementazione accurata delle grammatiche utilizzate.

### 2.1.5 Confronto delle Diverse Sotto-categorie

In questa sottosezione, verranno confrontate le tre categorie principali di metodi di generazione sequenziale di grafi sotto tre aspetti fondamentali: **Scalabilità**, **Espressività**, e **Scenari di Applicazione**.

**Scalabilità**

- **Complessità Temporale**: La complessità temporale dei diversi metodi è un fattore cruciale per determinare la loro scalabilità, specialmente quando si lavora con grafi di grandi dimensioni.

    - **Metodi basati su sequenze di nodi**: Questi metodi tendono a presentare una complessità temporale dell'ordine di $(O(N^2))$, dove $(N)$ rappresenta il numero di nodi. Ciò li rende meno scalabili, soprattutto per grafi con un gran numero di nodi.

    - **Metodi basati su sequenze di archi**: La complessità temporale per questi metodi è generalmente $(O(|E|))$, dove $(|E|)$ rappresenta il numero di archi. Per grafi sparsi, in cui $(N^2 \gg |E|)$, i metodi basati su sequenze di archi risultano essere più scalabili rispetto a quelli basati su sequenze di nodi.

    - **Metodi basati su sequenze di motivi**: La complessità di questi metodi varia, da $(O(N^2))$ per grafi generici a $(O(N \cdot |C|))$ per grafi specifici per dominio, dove $(|C|)$ rappresenta il numero di motivi grafici. La scalabilità dipende quindi dal tipo di grafi e dall'approccio specifico utilizzato.

    - **Metodi basati su sequenze di regole**: Questi metodi hanno tipicamente una complessità lineare rispetto al numero di regole utilizzate per generare un grafo, rendendoli altamente scalabili, soprattutto quando il numero di regole è significativamente inferiore al

numero di nodi o archi.

**Espressività**

- **Capacità di Modellare Dipendenze Complesse**: L'espressività di un modello di generazione dipende dalla sua capacità di catturare e modellare le dipendenze tra gli elementi costitutivi del grafo, come nodi, archi e strutture più complesse.

    - **Metodi basati su sequenze di nodi e di archi**: Questi metodi sono altamente espressivi, in quanto possono modellare complesse dipendenze tra nodi e archi, incluse le dipendenze nodo-nodo, arco-arco e nodo-arco. Sono adatti a catturare la complessa interazione tra gli elementi del grafo.

    - **Metodi basati su sequenze di motivi**: Questi metodi sono in grado di catturare dipendenze tra motivi di grafo, che spesso rappresentano relazioni di ordine superiore o schemi globali. Ciò li rende particolarmente adatti per rappresentare strutture complesse a livello di sottografo.

    - **Metodi basati su sequenze di regole**: Questi metodi offrono un'elevata espressività nel modellare dipendenze tra le regole operative, permettendo di catturare modelli semantici complessi nella costruzione di grafi. Sono particolarmente potenti nel generare grafi che devono rispettare vincoli sintattici e semantici stringenti, difficili da apprendere solo dalla topologia del grafo.

**Scenari di Applicazione**

- **Adeguatezza ai Contesti Applicativi**: La scelta del metodo di generazione sequenziale dipende dalla necessità di rispettare vincoli specifici,

dall'importanza della validità del grafo generato e dell'accessibilità delle regole o motivi.

- **Metodi basati su sequenze di nodi e di archi**: Questi metodi sono ideali per la generazione di grafi in contesti generali, dove non è richiesta una conoscenza specifica del dominio. Esempi tipici includono la generazione di reti sociali, reti di traffico, o altri tipi di grafi realistici che non devono necessariamente seguire regole specifiche predefinite.

- **Metodi basati su sequenze di motivi**: Sono particolarmente utili quando si desidera garantire parzialmente la validità del grafo generato attraverso la selezione di motivi predefiniti. Trovano applicazione in contesti dove i motivi grafici possono essere derivati da una base di conoscenza esistente, come nella generazione di strutture complesse in domini specifici.

- **Metodi basati su sequenze di regole**: Questi metodi sono preferibili in applicazioni dove è fondamentale garantire la validità sintattica e semantica del grafo generato, come nella generazione di molecole chimiche o nella modellazione di programmi. L'utilizzo di regole di produzione predefinite e di grammatiche assicura che il grafo risultante sia conforme a vincoli rigorosi, rendendoli ideali per applicazioni sensibili alla validità.

La scelta del metodo di generazione sequenziale di grafi dipende dal trade-off tra tutti gli elementi che abbiamo parlato fino ad ora. Mentre i metodi basati su sequenze di nodi e archi offrono un elevata espressività e sono adatti per scenari generici, i metodi basati su sequenze di motivi e regole sono preferibili in contesti dove è necessario rispettare vincoli specifici e garantire la validità del grafo generato.

## 2.2 Generazione One-Shot

In contrapposizione alla generazione sequenziale, i metodi di generazione one-shot mirano a mappare l'intero grafo in una rappresentazione unificata che segue una distribuzione probabilistica nello spazio. Questo approccio consente la generazione di un intero grafo in un solo passo, campionando direttamente da questa distribuzione.

I metodi di generazione one-shot hanno il vantaggio di poter modellare le proprietà globali del grafo, generando e raffinando l'intera struttura (sia i nodi che gli archi) in modo sincrono attraverso diverse iterazioni. Questo approccio consente di catturare meglio le caratteristiche strutturali del grafo, offrendo una maggiore capacità di rappresentare correttamente la distribuzione sottostante del dataset di grafi.

Tuttavia, la generazione one-shot presenta sfide significative in termini di scalabilità. La complessità temporale associata a questi metodi è generalmente superiore a $(O(N^2))$, poiché è necessario modellare collettivamente le relazioni globali tra tutti i nodi. Questo rende difficile applicare la generazione one-shot a grafi di grandi dimensioni, come quelli che si trovano frequentemente in applicazioni reali, come le reti sociali o le reti di citazioni.

La sfida principale in questi metodi consiste nel generare congiuntamente sia la **topologia del grafo** sia gli **attributi di nodi e archi**.

### 2.2.1 Generazione Basata su Matrice di Adiacenza

**Quadro Generale**

I metodi di generazione di grafi basati su **matrice di adiacenza** si concentrano sul mappare direttamente un embedding latente $(z)$ al grafo di output in termini di una matrice di adiacenza, spesso integrata con matrici o tensori degli attributi dei nodi e degli archi. L'obiettivo principale di questi metodi è realizzare una mappatura che sia al tempo stesso espressiva ed efficiente. Tuttavia, esiste un compromesso intrinseco tra queste due caratteristiche, e le tecniche esistenti adottano diverse strategie per bilanciarlo. Alcune di queste strategie includono modelli basati su MLP (Multilayer Perceptron), approcci basati su message-passing, trasformazioni invertibili, e convoluzioni trasposte. I modelli basati su MLP tendono ad essere altamente end-to-end, mentre gli approcci basati su message-passing e convoluzioni trasposte possono modellare esplicitamente le correlazioni di ordine superiore nei grafi. Le tecniche basate su trasformazioni invertibili, d'altra parte, cercano di modellare mappature più rigorosamente invertibili, ma con limitazioni aggiuntive sull'espressività.

**Metodi Basati su MLP**

Nei metodi basati su MLP, la generazione di grafi avviene attraverso un **decoder di grafi** $g(z)$, costruito utilizzando un MLP. Questo modello prende in ingresso una rappresentazione latente del grafo $(z \sim p(z))$, campionata da una distribuzione statistica (come la distribuzione normale standard), e produce simultaneamente una matrice di adiacenza $(A^{\pi})$ e una matrice di attributi dei nodi $(F^{\pi})$.

**Fasi della Generazione**:

1. $g(z)$ prende un vettore latente $(z)$ di dimensione $(D)$ e genera due oggetti continui e densi: $(\tilde{A}^{\pi})$, che definisce gli attributi degli archi, e $(\tilde{F}^{\pi})$, che rappresenta gli attributi dei nodi.
2. Questi attributi sono interpretati in termini probabilistici, con ogni nodo e arco rappresentato da una distribuzione categoriale di tipi.

3. Per ottenere il grafo finale, gli oggetti discreti $(A^{\pi})$ e $(F^{\pi})$ vengono derivati dai corrispondenti $(\tilde{A}^{\pi})$ e $(\tilde{F}^{\pi})$.

**Approcci di Discretizzazione**:

1. **Attivazione Sigmoid**: Durante l'addestramento, la funzione di attivazione sigmoid calcola $(A^{\pi})$ e $(F^{\pi})$.
2. **Ricampionatura Categoriale con Gumbel-Softmax**: Consente di campionare da una distribuzione categoriale durante il passaggio in avanti e mantenere valori continui nel passaggio indietro.

**Metodi Basati su Message-Passing**

I metodi basati su **message-passing** generano grafi raffinando iterativamente la topologia del grafo e le rappresentazioni dei nodi, partendo da una matrice di adiacenza inizializzata $(A^0)$ e da rappresentazioni latenti dei nodi $(H^0)$. Questi metodi utilizzano reti neurali basate su message-passing (MPNN) per aggiornare progressivamente la topologia del grafo e gli stati dei nodi attraverso più strati.

**Fasi della Generazione**:

1. A partire da un $(z)$ campionato, si genera una matrice di adiacenza iniziale $(A^0)$ e rappresentazioni latenti dei nodi $(H^0)$.
2. Queste vengono aggiornate tramite MPNN per produrre $(A^1, H^1)$, continuando iterativamente fino a raggiungere lo strato finale $(A^T, H^T)$.
3. I valori finali vengono utilizzati per parametrizzare le distribuzioni categoriali, dalle quali vengono campionati i nodi e gli archi del grafo finale.

**Metodi Basati su Trasformazioni Invertibili**

I metodi basati su **trasformazioni invertibili**, noti anche come **metodi basati su flow**, realizzano la generazione one-shot creando una funzione invertibile tra il grafo $(G)$ e il latente $(z)$. Un esempio di questo approccio è il modello **GraphNVP**.

**Fasi della Generazione**:

1. Nella trasformazione forward, le variabili discrete $(A)$ e $(F)$ vengono convertite in variabili continue $(A^0)$ e $(F^0)$, tramite dequantizzazione.
2. Successivamente, vengono applicati **reversible affine coupling layers** per trasformare $(A^0)$ e $(F^0)$ in rappresentazioni latenti $(z_A)$ e $(z_F)$.
3. La trasformazione backward inverte questo processo per ottenere il grafo finale, eseguendo l'argmax sui nodi per ottenere la matrice di caratteristiche discrete $(F)$.

**Metodi Basati su Convoluzione Trasposta**

I metodi basati su **convoluzione trasposta** utilizzano network di convoluzione trasposta per decodificare la matrice di adiacenza del grafo a partire dai vettori di rappresentazione latente dei nodi.

**Fasi della Generazione**:

1. Lo strato di convoluzione trasposta per i nodi decodifica le rappresentazioni degli archi del grafo a partire dall'embedding dei nodi.
2. Diversi strati di convoluzione trasposta per gli archi vengono applicati ricorsivamente per decodificare le rappresentazioni latenti degli archi.

Questi approcci permettono una generazione efficiente e scalabile di grafi, bilanciando l'espressività del modello con la necessità di gestione della complessità computazionale.

**Vantaggi e Limiti**

- **Vantaggi**: (Espressività) L'apprendimento della distribuzione della matrice di adiacenza permette al modello di catturare in modo esplicito la topologia del grafo, compresi i pattern complessi di connessioni tra i nodi. (Generazione Congiunta) La generazione congiunta delle matrici di adiacenza e degli

attributi assicura che la topologia del grafo e i suoi attributi siano coerenti tra loro.

- **Limiti**: (Inefficienza) Questo approccio è generalmente inefficiente sia in termini di memoria che di tempo, specialmente per grafi di grandi dimensioni. La matrice di adiacenza può essere molto grande, richiedendo una notevole quantità di memoria per essere rappresentata e manipolata. (Scalabilità) A causa della necessità di manipolare grandi matrici, i metodi basati su matrice di adiacenza sono difficili da scalare per grafi molto grandi.

### 2.2.2 Generazione Basata su Liste di Archi

Per affrontare le limitazioni della generazione basata su matrice di adiacenza, alcuni metodi si concentrano sull'apprendimento di **pattern locali** e si basano su rappresentazioni più compatte come le **liste di archi**. Questi metodi sono più efficienti nella gestione di grafi più grandi, soprattutto quando i pattern globali sono relativamente semplici.

**Quadro Generale**

I metodi basati su lista di archi si concentrano sulla generazione di grafi attraverso l'apprendimento delle probabilità di esistenza degli archi, trattando ogni arco come un'entità indipendente. Questi metodi sono particolarmente utili quando si desidera generare un nuovo grafo a partire da uno esistente, mantenendo la struttura dei nodi invariata. Il processo generale di questi metodi può essere diviso in due fasi principali:

**Fasi della Generazione**

1. **Calcolo del punteggio degli archi**: per ogni possibile coppia di nodi, viene stimata la probabilità di esistenza di un arco che li collega.
2. **Campionamento degli archi**: basandosi sui punteggi calcolati, gli archi vengono campionati per costruire il grafo finale.

I metodi esistenti per la generazione di probabilità degli archi si dividono principalmente in due categorie: basati sul random walk e basati su similarità tra nodi.

*Basati su random walk*

Questi metodi determinano la probabilità di un arco calcolando quanto spesso esso appare in un insieme di random walk generate sul grafo di partenza. Un esempio rappresentativo è **NetGAN**, che si propone di simulare reti su larga scala attraverso un processo in due fasi:

**Fasi della Generazione**

1. **Generazione di random walk**: viene utilizzato un modello GAN per apprendere la distribuzione delle random walk sul grafo osservato, e successivamente generare nuove random walk.
2. **Costruzione della matrice di punteggi**: viene costruita una matrice $(S \in \mathbb{R}^{N \times N})$, dove ogni elemento $(S_{i,j})$ rappresenta il numero di volte in cui l'arco $(i, j)$ appare nell'insieme di random walk generate.

La probabilità di esistenza di un arco $(\tilde{A}_{i,j})$ è quindi calcolata come:

$$\tilde{A}_{i,j} = \frac{S_{i,j}}{\sum_{u,v} S_{u,v}}$$

Infine, utilizzando questa matrice di probabilità degli archi, il grafo viene generato attraverso un processo di campionamento.

**Varianti e miglioramenti**

Alcuni lavori successivi hanno proposto variazioni a NetGAN, come la modifica del criterio per la scelta del nodo di partenza nelle random walk o l'introduzione di camminate spaziali-temporali per la generazione di grafi con proprietà spazio-temporali. Un esempio è la generalizzazione di NetGAN attraverso l'uso di GANs basati su motivi, dove la probabilità dell'arco è calcolata utilizzando matrici di punteggio generate da tre diversi GANs. Il processo di campionamento dell'arco può così basarsi su una delle tre matrici di punteggio selezionate casualmente.

*Basati su Similarità tra Nodi*

Questi metodi stimano la probabilità di un arco basandosi sulla similarità tra le rappresentazioni dei nodi. Il punto chiave è calcolare una matrice di probabilità di adiacenza $(\tilde{A})$ utilizzando le rappresentazioni latenti dei nodi $(Z \in \mathbb{R}^{N \times L})$, dove $(Z_i)$ rappresenta la rappresentazione del nodo $(v_i)$.

**Vantaggi e Limiti**

- **Vantaggi**: (Efficienza) Le liste di archi richiedono meno memoria rispetto alle matrici di adiacenza, rendendo questi metodi più adatti per grafi di grandi dimensioni. (Scalabilità) Grazie all'apprendimento di pattern locali, questi metodi possono gestire grafi più grandi senza compromettere significativamente l'efficienza.
- **Limiti**: (Espressività Limitata) Sebbene i metodi basati su liste di archi siano efficienti, possono non essere altrettanto espressivi nel catturare pattern topologici complessi come quelli basati su matrici di adiacenza. (Pattern Locali) Questi metodi sono più adatti a grafi con pattern globali semplici, il che può limitare la loro applicabilità in contesti in cui la topologia del grafo è altamente complessa e interconnessa.

### 2.2.3 Confronto delle Diverse Sotto-categorie

Confrontiamo ora le due categorie di metodi one-shot su due aspetti principali: **complessità temporale** e **scenari di applicazione**.

**Complessità Temporale**

Entrambe le categorie di metodi, sia quelli basati su matrice di adiacenza che quelli basati su similarità tra nodi per la generazione di liste di archi, presentano una complessità temporale di $(O(N^2))$. Questo è dovuto al fatto che entrambi i metodi devono considerare ogni possibile coppia di $(N)$ nodi nel grafo per stimare la probabilità dell'esistenza di un arco.

- **Metodi basati su matrice di adiacenza**: richiedono il calcolo delle probabilità di connessione per ogni coppia di nodi, che comporta una complessità quadratica rispetto al numero di nodi nel grafo.

- **Metodi basati su similarità tra nodi**: analogamente, la complessità rimane $(O(N^2))$ poiché viene valutata la similarità tra tutte le possibili coppie di nodi.

D'altro canto, i **metodi basati su camminate casuali** per la generazione di liste di archi sono più scalabili. In questi metodi, gli archi sono campionati in base alla probabilità che un arco esista, determinata dalla frequenza con cui esso appare in un insieme di camminate casuali generate. Questo approccio riduce la necessità di considerare esplicitamente ogni coppia di nodi, permettendo una maggiore efficienza temporale, soprattutto in grafi di grandi dimensioni.

**Scenari di Applicazione**

Le differenze nella complessità temporale e nella struttura dei metodi rendono ciascuna categoria più adatta a specifici scenari di applicazione:

- **Metodi basati su matrice di adiacenza**: Questi metodi sono particolarmente adatti per grafi di piccole dimensioni, dove la modellazione globale del grafo è

cruciale. Esempi di tali grafi includono strutture molecolari e proteine, dove la precisione nella rappresentazione delle connessioni globali è essenziale. La loro alta espressività e il ridotto consumo di tempo li rendono ideali per contesti in cui la comprensione dettagliata della struttura globale del grafo è più importante della scala.

- **Metodi basati su lista di archi**: Questi metodi sono invece più efficienti per la generazione di grafi di grandi dimensioni, dove l'interesse principale risiede nei modelli locali piuttosto che nella struttura globale. Esempi tipici includono reti sociali e reti di citazioni, dove le connessioni locali tra i nodi giocano un ruolo più rilevante rispetto alla visione complessiva del grafo. Grazie alla loro capacità di gestire grandi volumi di dati in modo scalabile, questi metodi sono preferiti in contesti che richiedono l'analisi di reti estese e complesse.

## 2.3 Confronto tra Generazione Sequenziale e One-shot

Entrambi i metodi di generazione presentano vantaggi e svantaggi distinti:

- **Generazione Sequenziale:** Eccelle nella gestione delle decisioni locali e nel trattamento delle dipendenze immediate tra i nodi e gli archi. Tuttavia, la sua capacità di catturare proprietà globali e dipendenze a lungo termine è limitata.

- **Generazione One-shot:** Offre una migliore capacità di catturare le proprietà globali del grafo, consentendo una modellazione più accurata delle strutture complesse. Tuttavia, la sua applicabilità a grafi di grandi dimensioni è limitata dalla complessità computazionale.

La scelta tra generazione sequenziale e one-shot dipende quindi dal tipo di grafi da generare e dalle specifiche esigenze applicative, come la necessità di scalabilità o la fedeltà alla struttura globale del grafo.

La generazione sequenziale di grafi rappresenta un approccio metodologico in cui la costruzione del grafo avviene attraverso una sequenza di decisioni, generando i nodi e gli archi uno alla volta o in piccoli gruppi, condizionando ogni passaggio al sottografo già generato. Questo approccio permette di catturare le complesse dipendenze locali tra i componenti del grafo durante il processo di generazione.

# 3 Modelli generativi profondi condizionati

L'obiettivo della generazione condizionale profonda di grafi è apprendere una distribuzione condizionale $(p_{\text{model}}(G|y))$, basata su un insieme di grafi realistici osservati $(G)$ e le relative informazioni ausiliarie o condizioni $(y)$. Queste condizioni possono includere etichette di categoria, contesti semantici, o grafi provenienti da altri spazi di distribuzione.

Rispetto alla generazione incondizionale di grafi, la generazione condizionale comporta una sfida aggiuntiva: l'estrazione e l'integrazione delle caratteristiche della condizione fornita nel processo di generazione dei grafi. Di conseguenza, i modelli generativi profondi condizionali vengono generalmente classificati in base a come gestiscono queste condizioni, le quali possono essere suddivise in tre principali categorie: grafi, sequenze, e contesto semantico.

## 3.1 Condizionamento sui Grafi

Il condizionamento sulla base di altri grafi, noto anche come trasformazione profonda di grafi o traduzione profonda di grafi, mira a tradurre un grafo di input $(G_S)$ dal dominio sorgente nel corrispondente grafo di output $(G_T)$ nel dominio di destinazione. A seconda delle entità che vengono trasformate, si possono identificare due categorie principali di approcci:

### 3.1.1 Trasformazione degli Archi

Il problema della trasformazione degli archi riguarda la generazione della topologia e degli attributi degli archi di un grafo di destinazione, condizionata su un grafo di input. In questo contesto, l'insieme degli archi $(E)$ e i loro attributi $(F_E)$ cambiano, mentre l'insieme dei nodi e i loro attributi $(F_V)$ rimangono invariati. Questo processo di traduzione può essere formalizzato dalla funzione di trasformazione $(T : G_S(V, E_S, F_V, F_{E_S}) \rightarrow G_T(V, E_T, F_V, F_{E_T}))$. Il problema della trasformazione degli archi ha molte applicazioni nel mondo reale, come la modellazione delle

reazioni chimiche, il ripiegamento delle proteine, e la sintesi delle reti informatiche malevole.

- **Approcci Basati su Insegnamento Supervisionato**: Utilizzano reti neurali per apprendere come modificare gli archi basandosi su dati di addestramento. Questi metodi impiegano spesso rappresentazioni latenti che mappano gli archi del grafo sorgente a quelli del grafo di destinazione, preservando la struttura globale.
- **Modelli Generativi**: Modelli come i Variational Autoencoders (VAE) e le Generative Adversarial Networks (GAN) vengono adattati per generare nuovi archi a partire da quelli esistenti, mantenendo l'integrità della struttura del grafo sorgente.
- **Approcci Basati su Encoder-Decoder** Uno degli approcci principali per affrontare il problema della trasformazione degli archi è l'utilizzo di un framework encoder-decoder. In questo contesto, l'encoder apprende una rappresentazione latente del grafo di input, che viene poi utilizzata dal decoder per generare il grafo di destinazione.
- **GT-GAN**: Questo modello utilizza una Generative Adversarial Network (GAN) per trasformare la topologia dei grafi. Il GT-GAN è composto da due componenti principali: un traduttore di grafi e un discriminatore condizionale di grafi. Il traduttore di grafi è suddiviso ulteriormente in un encoder e un decoder di grafi. L'encoder utilizza una rete neurale convoluzionale su grafi per incorporare il grafo di input in rappresentazioni latenti a livello di nodo. Successivamente, una rete di deconvoluzione su grafi viene utilizzata come decoder per generare la topologia del grafo di destinazione.

### 3.1.2 Co-trasformazione Nodo-Arco

Il problema della co-trasformazione di nodi e archi consiste nel generare sia gli attributi dei nodi che quelli degli archi di un grafo di destinazione, condizionata sul grafo di input. In questo caso, sia i nodi che gli archi possono variare durante il processo di trasformazione, come descritto dalla funzione di trasformazione $(T : G_S(V_S, E_S, F_{V_S}, F_{E_S}) \rightarrow G_T(V_T, E_T, F_{V_T}, F_{E_T}))$.

- **Modelli Basati su Reti Neurali Convoluzionali su Grafi (GCN)**: Questi modelli apprendono rappresentazioni latenti per nodi e archi attraverso convoluzioni su grafi, utilizzandole poi per generare il grafo di destinazione.

- **Modelli Basati su Messaggi (Message Passing)**: Questi metodi utilizzano meccanismi di passaggio di messaggi per aggiornare iterativamente le rappresentazioni dei nodi e degli archi, riflettendo i cambiamenti strutturali tra il grafo sorgente e quello di destinazione.

**Tecniche Basate su Embedding**

Le tecniche basate su embedding mirano a mappare i nodi e gli archi del grafo sorgente in uno spazio latente, da cui viene poi generato il grafo di destinazione. Questi approcci si suddividono principalmente in due categorie:

- **Modelli Basati su Reti Neurali (GAN)**: Alcuni modelli utilizzano reti generative avversarie per apprendere una rappresentazione latente del grafo sorgente. Il generatore della GAN utilizza poi questa rappresentazione per produrre il grafo di destinazione. Un esempio di questo approccio è il **Graph Translator GAN (GT-GAN)**, che integra un encoder di grafi e un decoder di grafi per trasformare un grafo di input in un grafo di output.

- **Graph Neural Networks (GNNs)**: Altri approcci utilizzano reti neurali su grafi, come le Graph Convolutional Networks (GCN) o le Graph Attention Networks (GAT), per apprendere embedding dei nodi e degli archi. Questi embedding vengono poi decodificati per generare il grafo di destinazione, compresa la sua topologia e i relativi attributi.

- **Modelli DAG-to-DAG:**

    - **Kaluze et al. (2020):** Questo modello propone la traduzione DAG-to-DAG utilizzando reti neurali, dove sia il dominio che l'immagine della funzione target sono spazi DAG (Directed Acyclic Graphs). Un

encoder basato su Deep-Gated DAG Recursive Neural Network (DG-DAGRNN) generalizza le RNN impilate per sequenze su strutture DAG. L'encoder utilizza unità ricorrenti gate (GRUs) per ogni nodo, producendo un embedding che serve come input per il decoder DAG. Il decoder genera il grafo di destinazione seguendo un processo sequenziale, predicendo prima il numero di nodi e poi generando nodi e archi basandosi sullo stato nascosto.

- **Modelli Graph-to-Graph:**
    - **You et al. (2018):** Propongono un modello che formalizza il grafo come un DAG e utilizza un modello ricorrente per la traduzione. L'encoder, denominato topology-flow encoder, incorpora la topologia del grafo di input nelle rappresentazioni dei nodi. Il decoder segue un processo di generazione basato sui nodi, predicendo la struttura del grafo di destinazione nodo per nodo.
- **Motivi Grafici:**
    - Alcuni approcci rappresentano il grafo come un insieme di motivi grafici predefiniti, particolarmente utili in compiti come l'ottimizzazione molecolare. Questi metodi sfruttano la combinazione di motivi grafici per generare il grafo di destinazione.

**Tecniche Basate su Editing**

Le tecniche basate su editing si concentrano sulla modifica diretta della struttura del grafo sorgente per ottenere il grafo di destinazione. Questo viene fatto attraverso operazioni di editing su nodi e archi:

- **Modelli Basati su Operazioni di Editing**: Questi modelli utilizzano operazioni strutturali come aggiungere, rimuovere o modificare nodi e archi per trasformare il grafo sorgente nel grafo di destinazione. Ad esempio, un algoritmo di editing dei grafi potrebbe essere adattato per generare un grafo

di destinazione basandosi sulle differenze con il grafo sorgente.

- **Reti Neurali per Editing di Grafi**: Alcuni approcci si avvalgono di reti neurali per apprendere le operazioni di editing necessarie per trasformare il grafo sorgente in quello di destinazione. Queste reti possono predire le modifiche necessarie a livello di nodi e archi per ottenere il grafo di destinazione desiderato.

- **Modelli Basati su Reinforcement Learning:**

    - **You et al. (2018):** Introducono il Graph Convolutional Policy Network (GCPN), che utilizza reti neurali convoluzionali per grafi e reinforcement learning per ottimizzare proprietà specifiche del grafo molecolare sorgente. Il modello opera in uno spazio di azioni fisso, dove ogni azione prevede la creazione di collegamenti tra nodi o l'aggiunta di sotto-grafi predefiniti al grafo target.

- **Modelli Basati su MPNN:**

    - **Guo et al. (2020):** Propongono un metodo basato su Message Passing Neural Networks (MPNN) per l'editing iterativo del grafo sorgente. Il modello genera attributi di nodi e archi in più stadi, producendo grafi intermedi. L'influenza di ciascun nodo sui nodi adiacenti è calcolata tramite una funzione basata su MLP (Multi-Layer Perceptron), mentre un'altra funzione aggiorna gli attributi dei nodi. Il processo di trasformazione degli archi segue una logica simile, mantenendo la coerenza tra nodi e archi.

**Considerazioni Aggiuntive**

- **Integrazione di Condizioni Supplementari**: Alcuni metodi includono condizioni supplementari, come etichette di classe o altre informazioni contestuali, per migliorare la qualità della traduzione del grafo.

- **Applicazioni Pratiche**: La trasformazione di grafi trova applicazione in vari campi, tra cui la bioinformatica (per la traduzione di strutture molecolari), la visione artificiale (per la traduzione di scene grafiche), e i social network (per la modellazione delle dinamiche di rete).

Questa categorizzazione offre un quadro chiaro su come i modelli generativi profondi affrontano la sfida della generazione di grafi basata su condizioni specifiche, variando dall'alterazione delle connessioni fino alla trasformazione simultanea di nodi e archi.

### 3.1.3 Confronto delle Diverse Sotto-categorie

**Pattern Catturati dai Grafi di Input:**

- **Tecniche Basate su Embedding:** Questi metodi catturano pattern globali, come densità o energia molecolare, utilizzando rappresentazioni latenti che sintetizzano informazioni complesse e globali dal grafo di input. Questo approccio è utile per modellare trasformazioni sostanziali e sofisticate tra il grafo di input e quello di destinazione.

- **Tecniche Basate su Editing:** Questi metodi si concentrano su pattern locali, come nodi "hub" o strutture ad anello, offrendo un controllo fine sulle modifiche locali. Sono ideali per trasformazioni che richiedono modifiche minori e specifiche nel grafo di input.

**Interpretabilità:**

- **Tecniche Basate su Editing:** Offrono una maggiore interpretabilità, mostrando esplicitamente le modifiche passo-passo dal grafo di input a quello di destinazione. Questo è vantaggioso in applicazioni che richiedono trasparenza e comprensione dettagliata del processo di generazione.

- **Tecniche Basate su Embedding:** La connessione tra i grafi di input e di destinazione è meno trasparente, poiché gli embedding latenti non sono facilmente spiegabili semanticamente. Questo può essere un limite quando è necessaria una comprensione chiara delle trasformazioni.

**Scenari di Applicazione:**

- **Tecniche Basate su Embedding:** Adatte per modellare trasformazioni che comportano cambiamenti significativi e complessi dal grafo di input a quello di destinazione. Sono efficaci in contesti che richiedono trasformazioni globali.

- **Tecniche Basate su Editing:** Più appropriate per scenari in cui le trasformazioni comportano piccoli cambiamenti. Questi metodi sono ideali per applicazioni dove sono richieste modifiche locali precise e minori al grafo di input.

## 3.2 Condizionamento su Sequenze

Il problema della generazione grafica condizionata su sequenze può essere visto come un problema di trasformazione da sequenza a grafo. L'obiettivo è generare un

grafo di destinazione $(G_T)$ condizionato su una sequenza di input $(X)$. Questo approccio trova applicazione in aree come l'elaborazione del linguaggio naturale (NLP) e l'analisi delle serie temporali.

**Parsing Semantico**

- **Chen et al. (2018, 2019):** Propongono un approccio chiamato Sequence-to-Action per il parsing semantico, modellato come un processo di generazione di grafi semantici end-to-end. Dato un input testuale $(X = \{x_1, ..., x_m\})$, il modello Sequence-to-Action genera una sequenza di azioni $(Y = \{y_1, ..., y_m\})$ per costruire il grafo semantico corrispondente. Le azioni includono operazioni come "Aggiungi Nodo Variabile", "Aggiungi Nodo Entità", "Aggiungi Nodo Tipo", "Aggiungi Arco", ecc. Il modello utilizza un'architettura RNN basata su meccanismi di attenzione, dove l'encoder converte la sequenza di input in vettori contestuali e il decoder genera la sequenza di azioni necessarie per costruire il grafo semantico.

**Generazione di Grafi Condizionati da Serie Temporali**

- **Yang et al. (2020):** Esplorano l'uso di GANs in un contesto condizionato e propongono il modello TSGG-GAN per la generazione di grafi condizionati da serie temporali. Il generatore utilizza Simple Recurrent Units (SRU) per estrarre informazioni rilevanti dalle serie temporali, e un MLP per generare il grafo diretto pesato. Questo metodo è efficace nel modellare le relazioni sottostanti tra le serie temporali e i nodi del grafo.

## 3.3 Condizionamento sul Contesto Semantico

La generazione di grafi condizionata sul contesto semantico si occupa di generare un grafo di destinazione $(G_T)$ in funzione di un contesto semantico rappresentato da un vettore $(C)$, che può includere categorie, etichette o altre meta-informazioni. Una

delle sfide principali è determinare come e dove incorporare il contesto semantico nel processo di generazione.

1. **Modulo di Inizializzazione dello Stato del Nodo**

    - **Yang et al.:** Propongono un modello GV-GAN (Graph Variational Generative Adversarial Networks) che incorpora il contesto semantico nel modulo di inizializzazione dello stato del nodo. Durante la generazione, ogni nodo viene inizialmente rappresentato da un embedding latente $(Z_i)$, a cui viene concatenato il vettore di contesto $(C)$ per ottenere una rappresentazione aggiornata $(\hat{Z}_i)$. La distribuzione di ogni arco $(A_{ij})$ è modellata come una distribuzione Bernoulliana, con probabilità calcolata tramite una funzione sigmoide applicata a un prodotto interno tra le rappresentazioni latenti aggiornate dei nodi.

    - **Li et al.:** Propongono un modello simile, dove il contesto semantico viene incorporato nelle rappresentazioni latenti iniziali $(Z_i)$ all'inizio del processo di decoding.

2. **Modulo di Passing dei Messaggi in Decoding Basato su MPNN**

    - **Li et al.:** Aggiungono le informazioni di contesto $(C)$ nel modulo di passing dei messaggi durante il processo di decoding basato su MPNN. Il processo di decoding viene parametrizzato come un processo Markoviano, in cui il grafo viene generato iterativamente attraverso l'aggiornamento degli stati nascosti dei nodi $(H^t)$. Ad ogni passo $(t)$, lo stato di un nodo $(h_{ti})$ viene calcolato utilizzando una combinazione delle informazioni di contesto e degli stati precedenti attraverso una rete di passing dei messaggi, influenzata dai pesi appresi $(W)$, $(\Theta)$, $(\Phi)$.

3. **Parametrizzazione della Distribuzione Condizionale per Generazione Sequenziale**

    - Nei modelli basati su MPNN e VAE, il contesto semantico può essere utilizzato per parametrizzare le distribuzioni condizionali che guidano la generazione sequenziale del grafo. Incorporando il contesto direttamente nelle rappresentazioni latenti dei nodi o nei moduli di passing dei messaggi, il contesto semantico influisce sulla generazione e sull'aggiornamento delle caratteristiche del grafo, migliorando la qualità e la coerenza del grafo generato.

# 4. Metriche di Valutazione per la Generazione di Grafi Profondi

Valutare la qualità dei grafi generati dai modelli generativi profondi è una sfida complessa, soprattutto perché richiede l'analisi sia della distribuzione appresa che della struttura intricata dei dati del grafo. Le metriche di valutazione possono essere suddivise in due categorie principali: metriche comuni utilizzate sia per la generazione incondizionata che condizionata, e metriche specifiche per la generazione condizionata.

**Metriche Comuni per la Generazione Incondizionata e Condizionata**

1. **Errori di Ricostruzione**

    - **Errore di Ricostruzione del Grafo:** Misura la discrepanza tra il grafo generato e un grafo di riferimento, considerando sia la topologia che le attribuzioni dei nodi e degli archi. Alcune delle metriche utilizzate includono:
        - **Errori sui Nodi:** Percentuale di nodi generati erroneamente o che mancano rispetto al grafo di riferimento.
        - **Errori sugli Archi:** Percentuale di archi errati o mancanti nella generazione del grafo rispetto al grafo di riferimento.
2. **Metriche di Qualità Topologica**

    - **Numero di Nodi e Archi:** Valuta se il numero di nodi e archi nei grafi generati corrisponde a quelli nei grafi di riferimento. Questo confronto può fornire un'indicazione della capacità del modello di preservare la struttura globale del grafo.
    - **Distribuzione dei Gradi:** Confronta la distribuzione dei gradi dei nodi (cioè, il numero di connessioni per nodo) nei grafi generati con quella dei grafi di riferimento. Una corrispondenza stretta indica che il modello cattura correttamente la connettività tipica della struttura del grafo.

3. **Metriche di Similarità Strutturale**

    - **Indici di Similarità:** Misure come la **Jaccard Similarity** o la **Cosine Similarity** vengono utilizzate per valutare la similarità tra la topologia del grafo generato e quella del grafo di riferimento. Questi indici quantificano quanto le strutture siano simili in termini di insiemi di nodi e archi.
4. **Metriche di Coerenza e Convergenza**

    - **Stabilità della Generazione:** Valuta la coerenza tra i grafi generati in condizioni simili. Analizza la variabilità nella generazione per determinare se il modello produce grafi coerenti o se mostra una tendenza a generare strutture diverse in situazioni simili.

**Metriche per la Generazione Condizionata**

1. **Adattamento al Condizionamento**

    - **Correlazione tra Condizione e Output:** Misura quanto efficacemente il grafo generato riflette le condizioni imposte, come etichette, categorie, o altri attributi semantici. Una forte correlazione indica che il modello è in grado di integrare con successo le condizioni nel processo di generazione.
    - **Precisione Condizionata:** Rileva quanto accuratamente il grafo generato soddisfa le specifiche fornite dal condizionamento. Ad esempio, se si specifica un certo numero di nodi o una determinata struttura, questa metrica valuta la precisione con cui queste specifiche vengono rispettate.
2. **Metriche di Qualità del Condizionamento**

    - **Errori di Condizionamento:** Misura la discrepanza tra le informazioni di condizionamento fornite e le caratteristiche effettive del grafo generato. Un basso errore di condizionamento indica che il modello ha seguito fedelmente le indicazioni di condizionamento.

- **Distanza dal Target:** Utilizza metriche come la **Graph Edit Distance** per misurare la distanza tra il grafo generato e il grafo target previsto, valutando il grado di deviazione dalla configurazione attesa.

3. **Metriche di Generazione Sequenziale Condizionata**

   - **Qualità della Sequenza di Generazione:** Nei modelli che generano grafi attraverso una sequenza di azioni (ad es., aggiunta di nodi o archi), è cruciale valutare la coerenza e la correttezza delle azioni rispetto al condizionamento. Questa metrica esamina la qualità del processo generativo in termini di precisione e sequenzialità delle operazioni.

4. **Metriche di Qualità dei Motivi**

   - **Motivi di Grafi:** In contesti specifici, come la generazione di molecole, è essenziale che i motivi (o sub-strutture) chimici presenti nei grafi generati corrispondano a quelli desiderati secondo il condizionamento. Questa metrica verifica la presenza e l'accuratezza di questi motivi rispetto alle aspettative.

## 4.1 Valutazione generale per la generazione di grafi profondi

La valutazione della qualità dei grafi generati dai modelli generativi profondi è essenziale per garantire che questi grafi rispettino le caratteristiche e le proprietà desiderate. La letteratura propone tre principali categorie di metriche per questa valutazione: statistiche, basate su classificatori e di qualità intrinseca. Le prime due categorie si concentrano sul confronto tra grafi generati e grafi reali, mentre la terza valuta direttamente le proprietà dei grafi generati.

### 4.1.1 Valutazione Basata su Statistiche

Nella valutazione basata su statistiche, la qualità dei grafi generati viene misurata confrontando la distribuzione delle statistiche dei grafi reali con quella dei grafi generati. Questo approccio permette di quantificare le differenze tra i due insiemi di grafi in modo sistematico.

**Statistiche tipiche dei Grafi**

Di seguito sono elencate le sette statistiche tipiche dei grafi utilizzate per questa valutazione:

1. **Distribuzione dei Gradi dei Nodi:** Analizza la distribuzione dei gradi dei nodi, riflettendo i modelli di connettività locali nel grafo.

2. **Distribuzione del Coefficiente di Clustering:** Misura il grado di aggregazione dei nodi. Il coefficiente di clustering di un nodo rappresenta la proporzione dei suoi vicini che sono anche connessi tra loro, formando triangoli.

3. **Distribuzione dei Conteggi degli Orbit:** Conta le occorrenze di specifiche configurazioni locali di nodi, chiamate orbit, che possono indicare la capacità del modello di catturare strutture complesse del grafo.

4. **Componente Connessa Più Grande:** Indica la dimensione della più grande sottostruttura connessa del grafo, che può riflettere la connettività globale.

5. **Conteggio dei Triangoli:** Misura il numero di triangoli nel grafo, un indicatore di densità locale e connettività.

6. **Lunghezza del Cammino Caratteristico:** Calcola la lunghezza media dei cammini più brevi tra tutte le coppie di nodi, fornendo un'indicazione della distanza media tra i nodi nel grafo.

7. **Assortatività:** Esamina la tendenza dei nodi a collegarsi con altri nodi simili in termini di grado, riflettendo pattern di omofilia o eterofilia.

**Metriche per misurare la distanza tra le distribuzioni**

Per confrontare le distribuzioni delle statistiche tra grafi reali e grafi generati, vengono utilizzate due metriche principali:

1. **Divergenza Kullback-Leibler Media (KL-D):**

    - Misura la differenza tra due distribuzioni di probabilità. Per ciascuna proprietà del grafo, si calcola la distribuzione media sia per i grafi reali che per quelli generati.
    - La Divergenza KL viene poi calcolata per valutare quanto la distribuzione delle statistiche dei grafi generati si discosti da quella dei grafi reali.
2. **Massima Discrepanza Media (MMD):**

    - Utilizza una funzione kernel per misurare la differenza tra due distribuzioni di grafi. La MMD al quadrato si basa sulle statistiche dei grafi campionati dalle due distribuzioni e valuta la discrepanza tra di esse.
    - La funzione kernel, spesso basata sulla distanza di Wasserstein, permette di catturare differenze sottili tra le distribuzioni delle statistiche.

### 4.1.2 Valutazione basata sui classificatori

La valutazione basata su classificatori utilizza un classificatore di grafi per determinare se i grafi generati appartengono alla stessa distribuzione di quelli reali. Questo metodo è particolarmente utile quando si confrontano diversi tipi di grafi generati da modelli differenti.

**Metodi Basati su Graph Isomorphism Network (GIN)**

Due metodi basati su GIN sono comunemente utilizzati:

1. **Valutazione Basata sull'Accuratezza:**

- Un classificatore GIN viene addestrato su un insieme di grafi reali che rappresentano diversi tipi di grafi utilizzati nel training del modello generativo.
- Successivamente, si valuta l'accuratezza con cui il GIN classifica i grafi generati. Un'alta accuratezza indica che i grafi generati condividono le caratteristiche distintive dei grafi reali per ciascun tipo di grafo.

Questi approcci basati su classificatori forniscono un modo diretto per valutare se i grafi generati sono realistici e appartengono alla stessa distribuzione dei grafi reali. Se il GIN può correttamente classificare i grafi generati, ciò suggerisce che il modello generativo ha catturato efficacemente le proprietà rilevanti del grafo.

### 4.1.3 Valutazione basata sulla qualità intrinseca

Oltre alla valutazione basata sulla similarità tra i grafi generati e quelli reali, è fondamentale considerare anche le metriche che misurano la qualità intrinseca dei grafi generati. Queste metriche forniscono una valutazione diretta delle caratteristiche del grafo senza confrontarlo necessariamente con un set di grafi reali. Le tre principali metriche di qualità intrinseca sono: **validità**, **unicità**, e **novità**.

**Validità**

**Definizione:** La validità misura se i grafi generati rispettano determinate proprietà o regole strutturali richieste. Questa metrica è essenziale in applicazioni dove i grafi devono mantenere specifiche caratteristiche o conformarsi a regole ben definite, come nelle strutture molecolari o in topologie di rete.

**Esempi:**

- **Grafi Ciclici/Alberi:** Per strutture come cicli o alberi, la validità è definita come la percentuale di grafi generati che effettivamente rispettano le proprietà topologiche di un ciclo o di un albero.
- **Grafi di Molecole:** Nel contesto della generazione di molecole, la validità è la percentuale di molecole chimicamente valide, valutate secondo regole di chimica computazionale come il rispetto della valenza atomica.

**Importanza:** Un'alta validità indica che il modello generativo è capace di produrre grafi che non solo hanno senso strutturalmente, ma che rispettano anche le specifiche richieste.

**Unicità**

**Definizione:** L'unicità valuta la diversità all'interno dell'insieme di grafi generati. L'obiettivo è che i grafi generati siano distinti tra loro, dimostrando che il modello non ripete semplicemente lo stesso grafo, ma è in grado di generare una varietà di strutture.

**Calcolo:**

- I grafi generati vengono analizzati per rimuovere quelli che sono isomorfi come sottografi rispetto ad altri grafi generati. La percentuale di grafi non isomorfi rimanenti rappresenta l'unicità.
- **Esempio:** Se su 100 grafi generati, 10 sono distinti e non isomorfi, l'unicità sarà del 10%.

**Importanza:** Un'alta unicità è un indicatore positivo della capacità del modello di generare un insieme diversificato di grafi, anziché limitarsi a riprodurre lo stesso grafo ripetutamente.

**Novità**

**Definizione:** La novità misura la percentuale di grafi generati che non sono presenti o non sono sottografi dei grafi nel set di addestramento. Questo parametro verifica la capacità del modello di creare nuove strutture che non sono semplici ripetizioni di ciò che ha già visto durante l'addestramento.

**Importanza:** La novità è cruciale per dimostrare che il modello non solo memorizza i dati di addestramento, ma ha anche appreso a generalizzare e a produrre grafi nuovi e potenzialmente utili in applicazioni non precedentemente esplorate.

## 4.2 Valutazione per la Generazione Condizionale di Grafi Profondi

Nella generazione condizionale di grafi, oltre alle metriche generali, è importante includere metriche che valutino come i grafi generati rispondono alle condizioni specifiche imposte dal modello. Due approcci principali sono la **valutazione basata sulle proprietà del grafo** e la **valutazione basata sulla relazione di mapping**.

### 4.2.1 Valutazione Basata sulle Proprietà del Grafo

Quando i grafi generati hanno un corrispondente grafo reale come etichetta, è possibile confrontarli direttamente misurando la somiglianza o la distanza attraverso diverse proprietà o kernel dei grafi.

**Metriche Comuni:**

- **Similarità del Kernel Basata su Random-Walk:** Misura la somiglianza tra due grafi basandosi sui cammini casuali all'interno dei grafi stessi.
- **Distanze Hamming e Ipsen-Mikhailov (HIM):** Combina la distanza di Hamming e quella di Ipsen-Mikhailov per confrontare le strutture grafiche.
- **Entropie Spettrali delle Matrici di Densità:** Valuta la somiglianza spettrale tra due grafi confrontando la distribuzione degli autovalori.
- **Distanza di Centralità dell'Autovettore:** Misura quanto i nodi influenzano la rete nel suo insieme, basandosi sulla centralità degli autovettori.
- **Distanza di Centralità di Vicinanza:** Valuta la vicinanza media di un nodo rispetto a tutti gli altri nodi nel grafo.
- **Similarità del Kernel di Weisfeiler-Lehman:** Utilizza un test di isomorfismo di grafi per misurare la somiglianza tra due grafi.
- **Kernel della Distanza Pairwise del Sottografo di Vicinato:** Confronta coppie di sottografi per valutare la similarità tra grafi a livello locale.

### 4.2.2 Valutazione Basata sulla Relazione di Mapping

Questa valutazione misura se la relazione appresa tra le condizioni imposte e i grafi generati è coerente con la relazione esistente tra le condizioni e i grafi reali.

**Metodi:**

- **Relazione di Mapping Esplicita:**
    - **Condizioni di Categoria:** Valuta se i grafi generati rientrano nelle categorie condizionali utilizzando un classificatore addestrato su grafi reali.
    - **Condizioni di Grafo:** Quando si modificano proprietà specifiche del grafo di input, si confrontano le proprietà del grafo generato con quelle del grafo di input per verificare se le modifiche soddisfano i requisiti.
- **Relazione di Mapping Implicita:**
    - **Problema di Traduzione del Grafo:** In contesti dove la relazione di mapping tra grafi di input e target è complessa, si utilizza un classificatore per distinguere tra grafi target reali e grafi di input, e i risultati della classificazione vengono utilizzati come metriche di valutazione.

Queste metriche sono cruciali per verificare se il modello generativo condizionale produce grafi che rispondono accuratamente alle condizioni fornite, mantenendo la coerenza con i grafi reali.

# 5. Applicazioni

I modelli generativi profondi per la generazione di grafi rappresentano un campo di ricerca dinamico e in crescita, con applicazioni che spaziano dalla scoperta di nuovi farmaci alla modellazione delle strutture proteiche, fino alla generazione di codice e alla risoluzione di problemi complessi come quelli di soddisfacimento di vincoli (SAT). Di seguito vengono descritte alcune delle applicazioni più significative.

### 5.1 Generazione di Molecole

La generazione di molecole è una delle applicazioni più promettenti dei modelli generativi di grafi, specialmente nel contesto della scoperta di farmaci e della scienza dei materiali. Il problema principale in questo campo è progettare nuove molecole che possiedano proprietà chimiche desiderate. Anche piccole modifiche nella struttura molecolare possono causare cambiamenti significativi nelle proprietà della molecola, rendendo la generazione di molecole un compito altamente complesso.

Tradizionalmente, la scoperta di nuove molecole è stata effettuata manualmente da esperti in chimica e farmacologia. Tuttavia, il vasto spazio delle molecole valide, derivante dalle numerose combinazioni possibili di atomi e legami, richiede un approccio più efficiente. I modelli generativi di grafi, che trattano le molecole come grafi con atomi come nodi e legami chimici come archi, offrono una soluzione innovativa per esplorare questo spazio in modo più rapido e sistematico, garantendo la validità chimica delle molecole generate.

### 5.2 Modellazione della Struttura Proteica

Le proteine sono complesse molecole biologiche costituite da lunghe catene di amminoacidi. Comprendere la struttura e la funzione delle proteine è cruciale per molte applicazioni biologiche e mediche, poiché la loro forma tridimensionale determina in gran parte la loro funzione. Tuttavia, i metodi computazionali attuali per progettare nuove proteine sono lenti e spesso richiedono un intervento umano, il che può introdurre soggettività e imprecisioni.

L'uso di modelli generativi di grafi per la modellazione della struttura proteica rappresenta una nuova frontiera in questo campo. Questi modelli possono accelerare la generazione di nuove strutture proteiche funzionali, riducendo la necessità di supervisione umana e migliorando l'efficienza del processo.

### 5.3 Parsing Semantico

Il parsing semantico riguarda la conversione di frasi in linguaggio naturale in rappresentazioni logiche chiamate Rappresentazioni Astratte del Significato (AMR). I parser semantici tradizionali si basano su grammatiche manuali e lessici per il grounding semantico, un processo che può essere lungo e impreciso. I modelli di parsing semantico neurali, che utilizzano approcci sequenza-sequenza, tendono a ignorare informazioni sintattiche preziose.

Gli AMR sono strutture ad albero che rappresentano le relazioni semantiche tra le parole di una frase. I metodi basati su modelli generativi di grafi trattano il parsing semantico come un problema di generazione di grafi semantici, dove il grafo AMR viene generato direttamente a partire dalla frase.

**Lavoro Rappresentativo:** Zhang et al. hanno proposto un metodo di parsing AMR che formalizza il problema come una generazione di grafi condizionata sulla sequenza di token. In questo approccio, l'input è la sequenza di parole e l'output è il grafo AMR corrispondente. Il modello prevede prima i nodi del grafo (che corrispondono alle parole) utilizzando una rete neurale basata su RNN. Successivamente, viene appresa una matrice di punteggi per determinare la probabilità di esistenza degli archi (che rappresentano le relazioni semantiche) tra i nodi, e gli archi vengono generati campionando dalla matrice di punteggi. Questo approccio dimostra una notevole capacità di catturare le informazioni semantiche in modo automatizzato.

### 5.4 Modellazione del Codice

La modellazione del codice è un'area emergente che si occupa di generare codice sorgente valido e funzionale, considerando vincoli sintattici e semantici. I modelli

generativi di grafi possono codificare rappresentazioni significative dei programmi, migliorando la generazione di codice valido e realistico.

**Lavoro Rappresentativo:** Brockschmidt e Al. hanno introdotto un metodo che formalizza la modellazione del codice come generazione di strutture grafiche, rappresentando il codice come un Albero di Sintassi Astratta (AST). Ogni nodo dell'AST rappresenta una costruzione del codice e gli archi denotano le relazioni semantiche. Il modello utilizza tecniche di generazione sequenziale regolate da regole per espandere un nodo alla volta, seguendo la grammatica del linguaggio di programmazione. Questo approccio semplifica il problema di generazione del codice in una serie di problemi di campionamento delle regole di produzione, facilitando la generazione di programmi sintatticamente e semanticamente validi.

### 5.5 Generazione di Istanze SAT Pseudo-Industriali

La generazione di istanze SAT (Satisfiability) artificiali è cruciale per testare e sviluppare risolutori SAT. Generare istanze che riflettano le caratteristiche delle istanze reali è difficile con metodi progettati a mano, poiché queste istanze possono presentare una varietà di caratteristiche complesse.

**Lavoro Rappresentativo: G2SAT** formalizza il problema di generazione di istanze SAT come un compito di generazione di grafi, rappresentando le formule SAT come grafi bipartiti. In questo grafo, i nodi rappresentano letterali e clausole, e gli archi indicano la presenza di un letterale in una clausola. Il processo di generazione segue uno stile basato su sequenze di motivi, dove nuovi motivi (alberi estratti dai grafi bipartiti di addestramento) vengono aggiunti al grafo parzialmente generato. Come mostrato nella figura 9 (non inclusa qui), il modello G2SAT costruisce un grafo bipartito iniziando da un insieme di motivi e combinando clausole con operazioni di congiunzione per ottenere la formula SAT finale.

Queste applicazioni dimostrano la versatilità e l'efficacia dei modelli generativi profondi per grafi nel risolvere una serie di problemi complessi e di grande rilevanza pratica. Con l'evoluzione continua di queste tecniche, si prevede che queste applicazioni diventeranno sempre più sofisticate e utili in diversi campi scientifici e ingegneristici.

# 6 Opportunità Future

Il campo della generazione di grafi con modelli generativi profondi è ancora giovane e promettente, e molte sfide rimangono aperte. Di seguito sono delineate alcune delle principali opportunità di ricerca e sviluppo future.

### 6.1 Scalabilità

**Sfida:** La scalabilità dei modelli generativi profondi per grafi è una delle sfide più critiche. Attualmente, la maggior parte dei modelli ha una complessità temporale super-lineare rispetto al numero di nodi e archi nel grafo, il che limita la loro applicabilità a reti di grandi dimensioni. Per esempio, i modelli con complessità temporale (O(N^2)) o peggiore possono gestire solo grafi relativamente piccoli, mentre le reti reali, come Internet o le reti sociali, possono contenere milioni o miliardi di nodi.

**Opportunità:** Per migliorare la scalabilità, sono necessari modelli con complessità temporale lineare o quasi lineare rispetto al numero di nodi e archi. Alcuni approcci promettenti includono l'uso di tecniche di apprendimento basate su grafi che ottimizzano la rappresentazione e la manipolazione dei dati o la progettazione di algoritmi di campionamento e riduzione dimensionale che preservano le proprietà strutturali principali.

### 6.2 Vincoli di Validità

**Sfida:** Molti grafi reali devono soddisfare vincoli specifici di validità. Per esempio, nei grafi molecolari, il numero di legami chimici deve rispettare le valenze atomiche, mentre nelle reti di interazione proteica, le connessioni devono riflettere relazioni biologiche valide. L'applicazione di questi vincoli durante l'addestramento dei modelli generativi è complessa, poiché molti vincoli sono discreti e non differenziabili.

**Opportunità:** È fondamentale sviluppare metodi per integrare vincoli di validità nei modelli generativi in modo più efficace. Questo potrebbe includere l'uso di tecniche di ottimizzazione vincolata o la progettazione di algoritmi che incorporano vincoli di validità direttamente nella funzione di perdita. Alcuni approcci potrebbero includere

l'uso di modelli ibridi che combinano tecniche di apprendimento profondo con euristiche specifiche per garantire la validità dei grafi generati.

### 6.3 Interpretabilità

**Sfida:** I modelli generativi profondi per grafi tendono a essere "scatole nere", il che rende difficile comprendere come le variabili latenti influenzano le proprietà specifiche dei grafi generati. La mancanza di interpretabilità limita la nostra capacità di capire e controllare le caratteristiche dei grafi generati.

**Opportunità:** La ricerca futura dovrebbe concentrarsi sull'aumento dell'interpretabilità dei modelli generativi. Ciò potrebbe comportare l'analisi delle variabili latenti e delle loro influenze sulle proprietà dei grafi generati, nonché lo sviluppo di metodi per decodificare e comprendere le relazioni locali tra sottografi. Tecniche come la visualizzazione delle variabili latenti e l'analisi delle dipendenze locali potrebbero offrire nuove intuizioni e migliorare il controllo sui processi generativi.

### 6.4 Oltre i Dati di Addestramento

**Sfida:** I modelli generativi sono limitati dai dati di addestramento disponibili e spesso soffrono di problemi come il collasso di modalità, dove il modello genera solo un numero ristretto di varianti di grafi. La novità dei grafi generati può essere limitata dalla variabilità dei dati di addestramento.

**Opportunità:** Per affrontare questi problemi, è necessario sviluppare tecniche che permettano ai modelli di generare grafi più vari e innovativi. Questo potrebbe includere l'uso di tecniche come la perturbazione del codice latente, l'uso di etichette aggiuntive o l'implementazione di tecniche di controllo extra nella generazione. Esplorare come queste tecniche possono essere adattate per i grafi potrebbe aprire nuove strade per la creazione di grafi più originali e diversificati.

### 6.5 Grafi Dinamici

**Sfida:** La maggior parte dei modelli generativi esistenti si concentra su grafi statici, mentre molti grafi nel mondo reale sono dinamici e cambiano nel tempo. La

generazione di grafi dinamici, che riflettono l'evoluzione temporale dei dati, è un'area poco esplorata.

**Opportunità:** La ricerca futura dovrebbe esplorare come modellare e generare grafi dinamici. Questo comporta la modellazione congiunta dei pattern temporali e strutturali nei grafi e l'integrazione dei vincoli di validità temporali. Tecniche come l'uso di modelli di serie temporali per i grafi o la progettazione di modelli che incorporano dinamiche temporali nella generazione potrebbero fornire nuovi approcci per affrontare questi problemi complessi.

# 7 Conclusioni

L'area della generazione di grafi con modelli generativi profondi è ricca di opportunità di ricerca e sviluppo. Affrontare le sfide della scalabilità, dei vincoli di validità, della interpretabilità, della novità dei dati e dei grafi dinamici non solo avanza il campo, ma può anche portare a applicazioni pratiche significative in diversi settori. Il futuro della ricerca dovrà continuare a esplorare questi aspetti al fine di sviluppare modelli più potenti e versatili che possano affrontare i problemi complessi e variegati del mondo reale.

# 8 Bibliografia

M. C. D. P. Kaluza, S. Amizadeh, and R. Yu, “A neural framework for learning dag to dag translation,” NeurIPS’2018 Workshop, 2018.

J. You, R. Ying, and X. Ren et al, “Graphrnn: generating realistic graphs with deep auto-regressive models,” in ICML’2018, 2018, pp. 5708–5717.

X. Guo, S. Tadepalli, and L. Zhao et al, “Generating tertiary protein structures via an interpretative variational autoencoder,” arXiv preprint arXiv:2004.07119, 2020.

B. Chen, L. Sun, and X. Han, “Sequence-to-action: End-to-end semantic graph generation for semantic parsing,” in ACL’2018, 2018, pp. 766–777.

Y. Wang, W. Che, and J. Guo et al, “A neural transition-based approach for semantic dependency graph parsing,” in AAAI’2018, 2018, pp. 5561–5568.

S. Yang, J. Liu, and K. Wu et al, “Learn to generate time series conditioned graphs with generative adversarial nets,” arXiv preprint arXiv:2003.01436, 2020.

Y. Li, O. Vinyals, and C. Dyer et al, “Learning deep generative models of graphs,” ICLR’2018 Workshop, 2018.

S. Zhang, X. Ma, and K. Duh et al, “Amr parsing as sequence-to-graph transduction,” in ACL’2019, 2019, pp. 80–94

M. Brockschmidt, M. Allamanis, and A. L. Gaunt et al, “Generative code modeling with graphs,” 2019.

Xiaojie Guo, Liang Zhao, "A Systematic Survey on Deep Generative Models for Graph Generation", arXiv:2007.06686v3 [cs.LG] 4 Oct 2022.

Faezeh Faez, Yassaman Ommi, Mahdieh Soleymani Baghshah, Hamid R. Rabiee, "Deep Graph Generators: A Survey", 2021.

# 9 Ringraziamenti

Al termine del mio ormai lungo percorso di studi, desidero esprimere una serie di profondi ringraziamenti che riflettono il viaggio emozionante e gratificante che ho intrapreso.

Innanzitutto, voglio rivolgere un sentito ringraziamento alla mia famiglia, pilastro insostituibile della mia vita. Per il loro costante sostegno, la loro dedizione e l'amore infinito che mi hanno donato in ogni fase di questo cammino, sono eternamente grato. Senza di loro, nulla di tutto ciò sarebbe stato possibile.

Un sentito ringraziamento va al mio stimato relatore, il professor Giorgio Leonardi, la cui guida esperta e fiducia nel mio lavoro hanno rappresentato un faro luminoso lungo il percorso della mia tesi. La sua competenza e il suo incoraggiamento hanno contribuito in modo significativo alla mia crescita accademica e personale.

Un caloroso ringraziamento va ai miei amici di sempre, coloro che hanno condiviso con me gioie, dolori e birre, e che mi hanno saputo regalare nei momenti di leggerezza e divertimento specialmente quando ero triste. Grazie per ogni giornata spesa assieme e per ogni risata condivisa. Il vostro sostegno è stato fondamentale per il mio equilibrio emotivo e la mia felicità.

Desidero esprimere la mia riconoscenza speciale alla professoressa Lavinia Egidi e al professor Daniele Codetta Raiteri, figure determinanti che hanno reso possibile la mia preziosa esperienza di studio all'estero. Il loro sostegno e la loro disponibilità hanno arricchito il mio percorso accademico in un modo che non ritenevo possibile.

Un sentito ringraziamento va anche ai miei amici, compagni di avventure e di scoperte durante il mio periodo Erasmus a Lione. Senza la loro presenza e il loro sostegno, l'esperienza all'estero non sarebbe stata altrettanto memorabile e significativa.

Non posso dimenticare di ringraziare l'associazione "Yggdra APS" e le persone straordinarie che la compongono. La loro energia positiva e la loro dedizione durante questo ultimo anno, è stata un faro di speranza e determinazione per me.

Infine, desidero ringraziare ogni singola persona che ha incrociato il mio cammino, anche per breve tempo, contribuendo in modo unico alla mia crescita e al mio sviluppo personale. Ognuno di voi ha lasciato un'impronta preziosa nel mio cuore e nella mia mente, e per questo vi sono grato.

Per concludere, un ringraziamento speciale va a me stesso (e alla mia sedia per il supporto quotidiano). Nonostante le sfide e le avversità incontrate lungo il percorso, sono giunto fino a qui con determinazione e resilienza. Queste parole che scrivo sono testimonianza del mio impegno e della mia costante ricerca di crescita e realizzazione personale. Grazie, me stesso, per non aver mai smesso di credere nei miei sogni e nel mio potenziale.