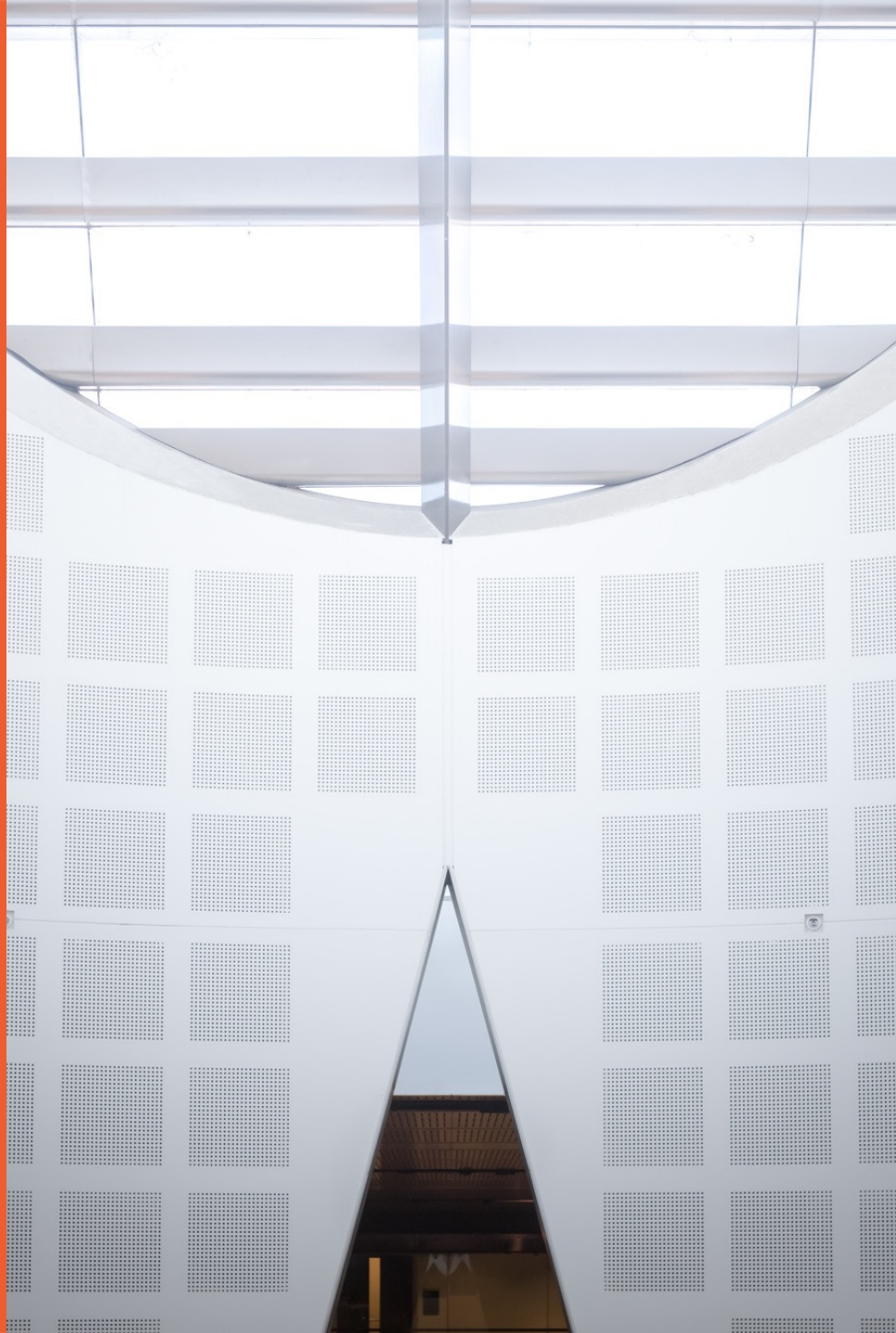
# Week 8 Drug Discovery



THE UNIVERSITY OF SYDNEY

# Overview

**Data:** structured and unstructured biomedical and biological data

- Sources: *hospitals*, *laboratories*, *pharmaceutical companies*, etc.
- Categories: *drug molecular structures*, *protein-drug interaction networks*, etc.

**Approach:** data analysis algorithms

- Categories: *pre-marketing*, and *post-marketing*;
- Targets: *limit the search space* and *provide recommendations to domain experts* for hypothesis generation and further analysis and experiments.

# Overview

## 1. Chemical and Biological Data

# Chemical and Biological Data

## What kind of data do we need?

- Information about **drugs**;
- Information about their **targets**;
- The **side effects** of drugs;
- The **chemical interactions** between
  - a drug and its target(s) (i.e., **drug-target** interactions)
  - a drug and other drug(s) (i.e., **drug-drug** interactions).
- …

# Chemical and Biological Data

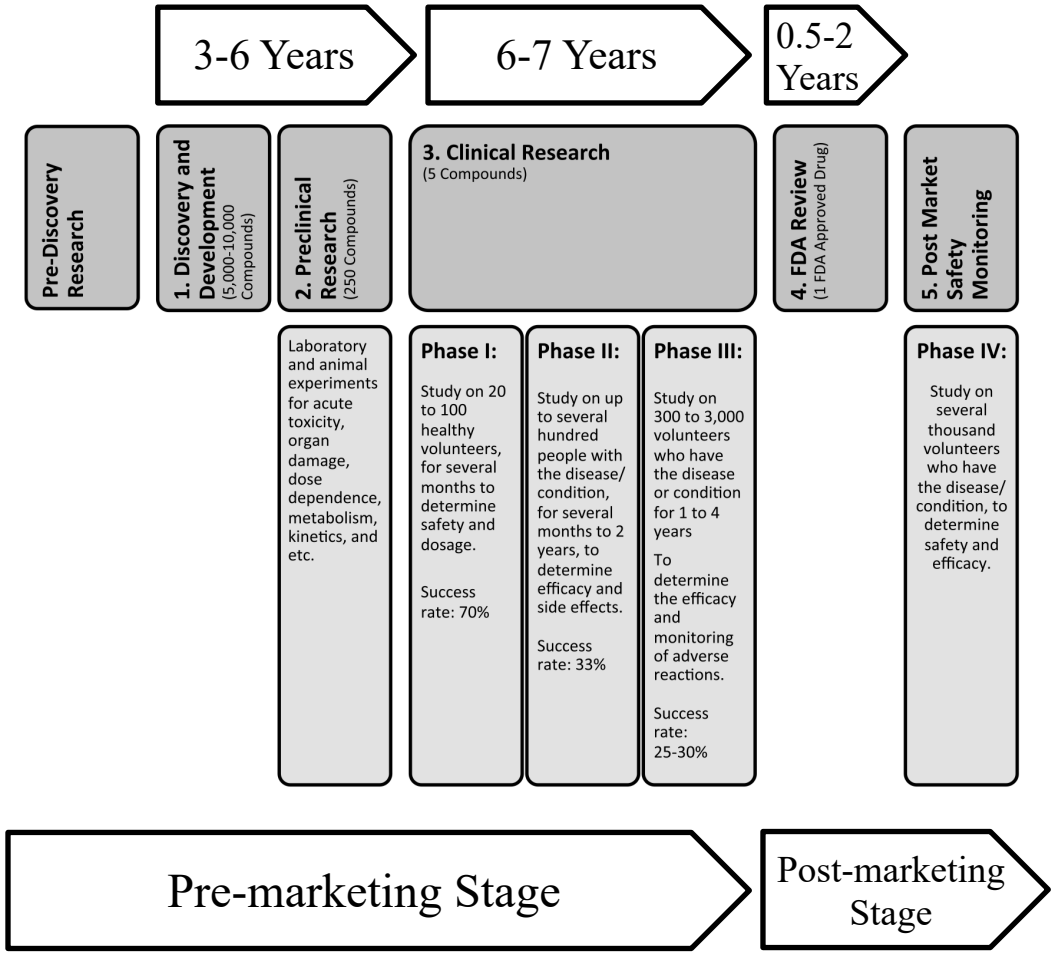
## Examples of Databases

| Type | Database | Content |
|---|---|---|
| Drug | PubChem | Structures and activities of chemicals |
| Protein | PDB | Structures of proteins |
| Target Protein | TTD | Target, disease, pathway and corresponding drugs |
| ADR protein | DITOP | Drug-induced toxicity related proteins |
| Drug-target | STITCH | Known and predicted interactions of drug–target |
| Genetic profile | CMAP | Gene expression, diseases and bioactive small molecules |
| Cellular profile | NCI60 | Gene expression and drug activity against 60 cancer cells |
| Pathway | KEGG | Pathways of biological substances and xenobiotics |

# Overview

1. Chemical and Biological Data
2. **Approaches**

# Approaches




Creating a new drug involves a lengthy process that typically spans a decade or more.

Utilizing data analysis algorithms can serve as valuable tools to expedite the drug development timeline and lower associated expenses.

# Approaches

## Pre-marketing Stage

- **Discovery and Development:** predicting ***drug-target interaction*** between chemical compounds (e.g., drugs) and biological targets (e.g., proteins) or generating drug-like molecules;
- **Predicting Adverse Side Effects/Unintentional Therapeutic Effects:**
  - **Multi-target interaction:** Most drugs affect multiple targets, which potentially results in adverse side effects or unintentional therapeutic effects (e.g., unacceptable toxicities). They are the main cause in the high failure rate of drug candidates in clinical trials.
  - **Drug-drug interaction:** This may account for up to 30% of unexpected adverse drug events and close to 50% in hospitalized patients [10]. A survey shows over 76% of elderly Americans are taking two or more drugs each day [12]. Another study estimated 29.4% of elderly patients are taking six or more drugs [13].
- **Drug-repositioning/repurposing:** predicting drug-target interaction to find new therapeutic effects of pre-approved drugs.

# Approaches

## Post-marketing Stage

Finding **patterns** that indicate potential **drug-related adverse events**:

- Consequences of such events (the U.S. as an example):
  - causes more than two millions hospitalizations and injuries each year;
  - causes more than 700,000 emergency visits each year;
  - leads to potential 100,000 deaths each year;
  - costs about $75 billion annually.
- Difficulties to identify such events:
  - Only a limited number of patient characteristics are studied in clinical trials and for a limited duration.

Data analysis algorithms are crucial to **narrow the search space** and **detect the hidden patterns**.

# Approaches

In this lecture, we are going to focus on three kinds of **drug discovery** methods:

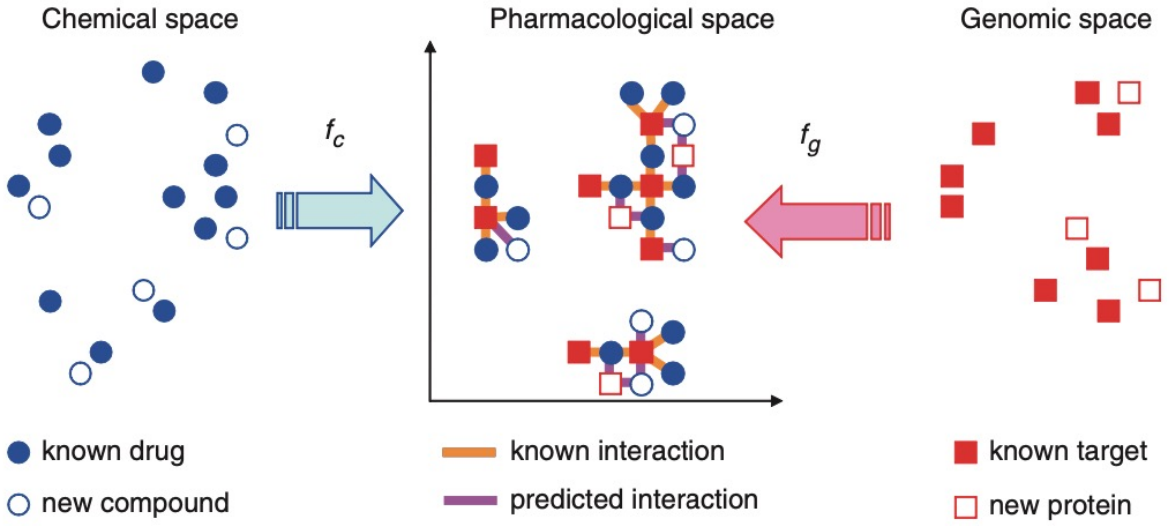
1. Drug-target Interaction Prediction;
2. Deep Graph Generative Modeling;
3. Large Language Models.

# Drug-target Interaction Prediction

# Drug-target Interaction Prediction

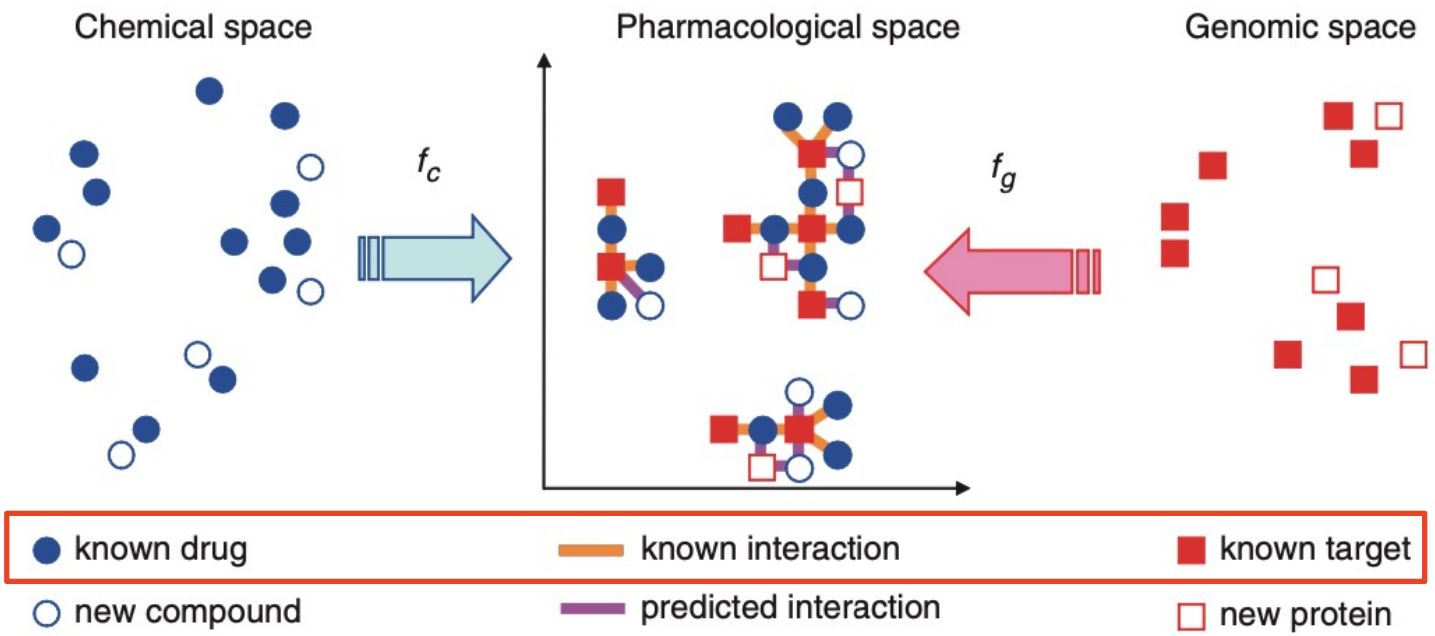
## Pharmacological Space

- Investigate the **relationship** between drug chemical structure, target protein sequence and drug-target network topology;
- Develop a new supervised method to infer unknown **drug-target interactions** by integrating chemical space and genomic space into a unified space called "pharmacological space".



# Pharmacological Space

**Step 1:** Known compounds and proteins on a known interaction network are embedded into a unified **pharmacological space**.



# Pharmacological Space

## Interaction network embedding

Both compounds and proteins are represented by sets of q-dimensional feature vectors: $\{\boldsymbol{u}_{c_i}\}_{i=1}^{n_c}$ and $\{\boldsymbol{u}_{g_i}\}_{j=1}^{n_g}$, where $c$ is a compound and $g$ is a protein.

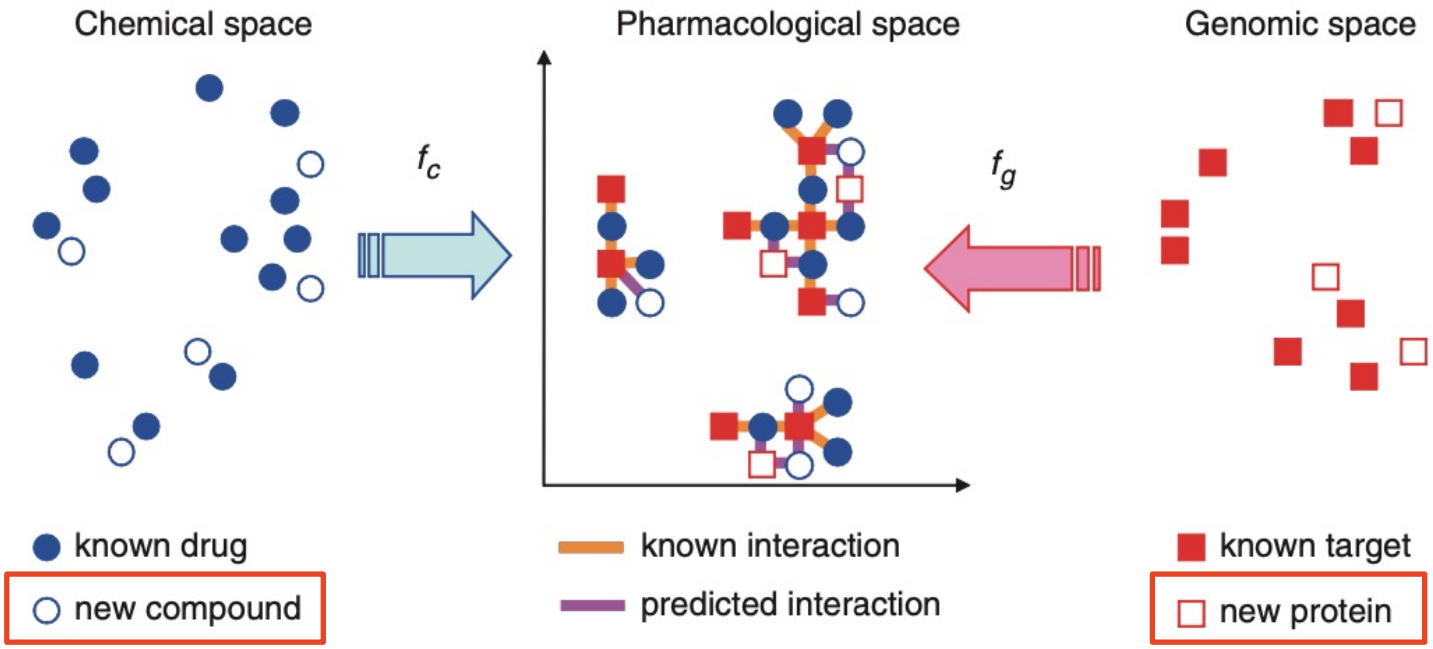
All drugs and target proteins are then represented by the row vectors of the matrix

$$U = \left(\boldsymbol{u}_{c_1}, \dots, \boldsymbol{u}_{c_{n_c}}, \boldsymbol{u}_{g_1}, \dots, \boldsymbol{u}_{g_{n_g}}\right)^T.$$

The space spanned by features $\boldsymbol{u}_c$ and $\boldsymbol{u}_g$ is referred to as "pharmacological feature space".

# Pharmacological Space

**Step 2: Regression models** ($f_c$ and $f_g$) are learned to map new compounds and proteins from the *chemical* and *genomic space* onto the unified *pharmacological space*.



# Pharmacological Space

## Kernel regression model

- The model $f_c$ maps the new compound $c_{new}$ onto the pharmacological feature space as:

$$\mathbf{u}_{c_{\text{new}}} = f_c(c_{\text{new}}, c_i) = \sum_{i=1}^{n_c} s_c(c_{\text{new}}, c_i)\mathbf{w}_{c_i}$$

  where $s_c(\cdot,\cdot)$ is a chemical structure similarity score.

- The model $f_g$ maps the new protein $g_{new}$ onto the pharmacological feature space as:

$$\mathbf{u}_{g_{\text{new}}} = f_g(g_{\text{new}}, g_j) = \sum_{j=1}^{n_g} s_g(g_{\text{new}}, g_j)\mathbf{w}_{g_j}$$

  where $s_g(\cdot,\cdot)$ is a sequence similarity score.

# Pharmacological Space

## Chemical structure similarity score

The similarity between two compounds $c$ and $c'$ is computed as:

$$s_c(c, c') = |c \cap c'|/|c \cup c'|.$$

## Sequence similarity score

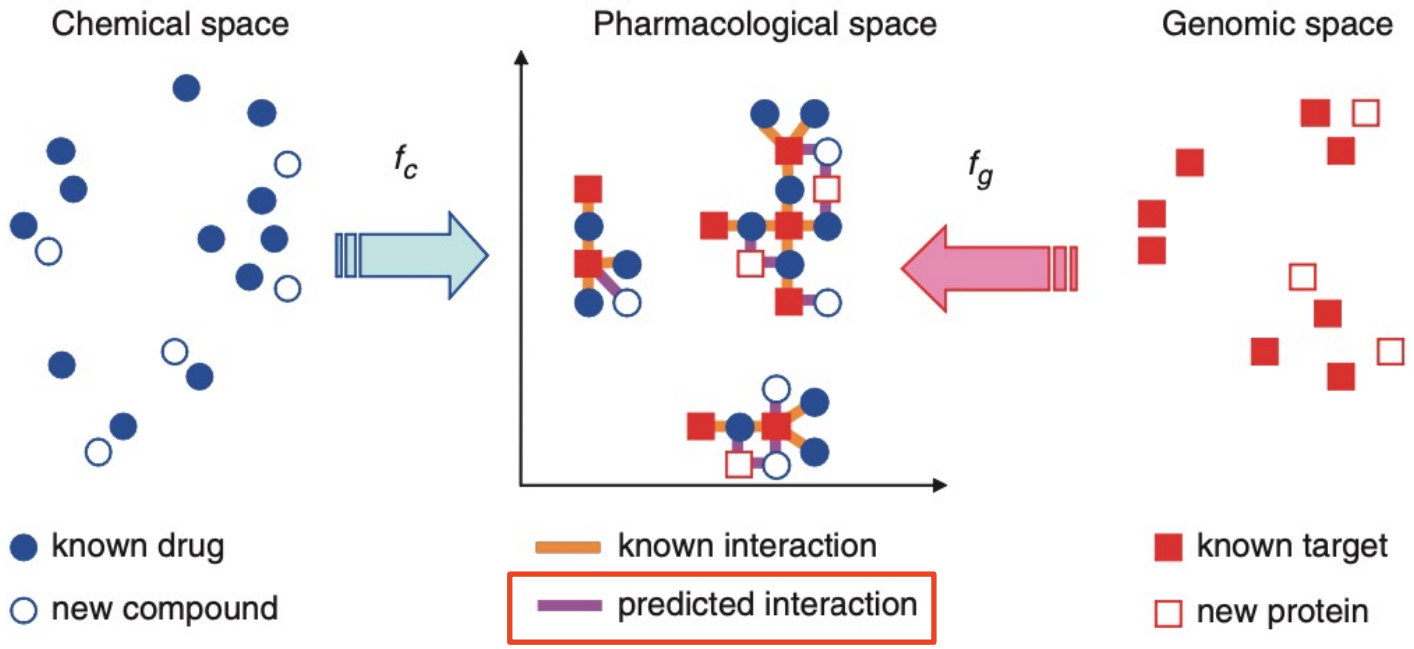
The similarity between two proteins $g$ and $g'$ is computed as:

$$s_g(g, g') = SW(g, g')/\sqrt{SW(g, g)}\sqrt{SW(g', g')},$$

where $SW(\cdot,\cdot)$ is the Smith-Water score.

# Pharmacological Space

**Step 3:** Interacting compound-protein pairs can be predicted by connecting compounds and proteins which are **closer than a threshold** in the pharmacological space.



# Pharmacological Space

## Feature-based similarity scores

Compute scores for three types of compound-protein pairs by calculating the inner product:

1) $corr(c_{new}, g_j) = \boldsymbol{u}_{c_{new}} \cdot \boldsymbol{u}_{g_j}$;
2) $corr(c_i, g_{new}) = \boldsymbol{u}_{c_i} \cdot \boldsymbol{u}_{g_{new}}$;
3) $corr(c_{new}, g_{new}) = \boldsymbol{u}_{c_{new}} \cdot \boldsymbol{u}_{g_{new}}$.

The feature-based similarity score is used as a measure of **the closeness between compounds and proteins** in the pharmacological feature space. Then, high-scoring compound-protein pairs are predicted to interact with each other.

# Deep Graph Generative Modeling

# Deep Graph Generative Modeling

## 1. Graph Neural Networks (GNNs)

# Graph Neural Networks (GNNs)

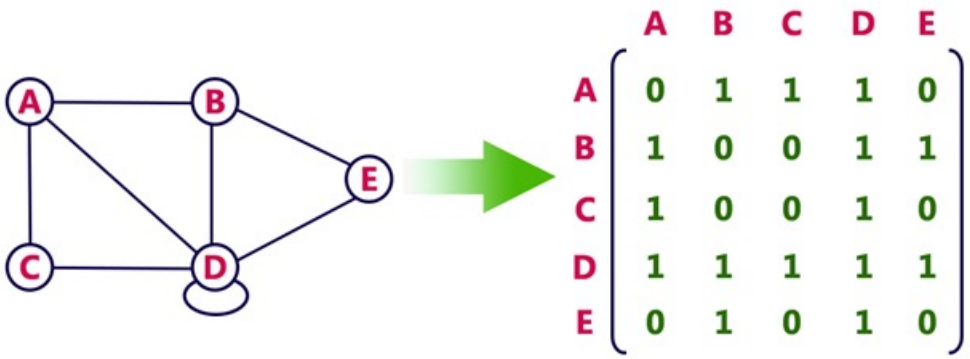
## Preliminary: graph representation

A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ typically consists of

- A set of **nodes/vertices** $\mathcal{V} = \{v_i | i = 1, \dots, N\}$;
- A set of edges $\mathcal{E} = \{e_{ij} | v_i \text{ is connected to } v_j\}$.

Graphs can be stored as **adjacency matrices** in computer:

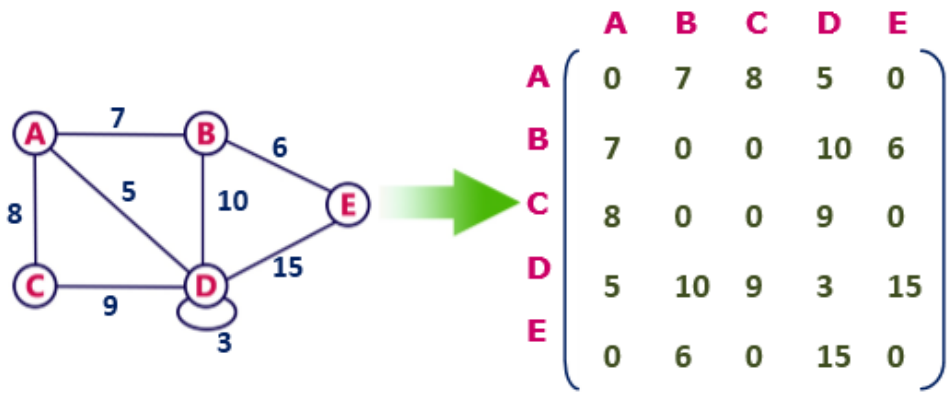
- 1 represents there is a connection, and 0 represents no connection.



# Graph Neural Networks (GNNs)

## Preliminary: weighted graphs

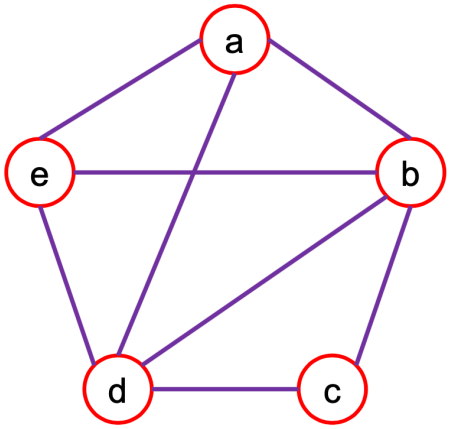
Edges can also correspond to **weights** to represent different kinds of connections (i.e., different chemical bonds in molecules).



# Graph Neural Networks (GNNs)

## Preliminary: degree matrix

- is usually denoted by $D$;
- stores how many edges connect to a node;
- is always a diagonal matrix.



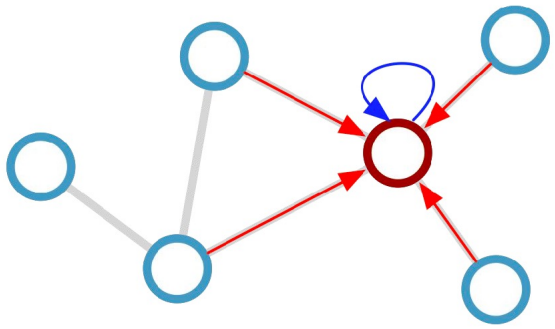

$D=$

| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | 3 | 0 | 0 | 0 | 0 |
| b | 0 | 4 | 0 | 0 | 0 |
| c | 0 | 0 | 2 | 0 | 0 |
| d | 0 | 0 | 0 | 4 | 0 |
| e | 0 | 0 | 0 | 0 | 3 |

# Graph Neural Networks (GNNs)

## A message passing view of GNNs

- In a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, each **node** (or vertex) $v_i \in \mathcal{V}$ carries some feature $h_i$, and each **edge** $e_{i,j} \in \mathcal{E}$ represents a connection between $v_i$ and $v_j$;
- A GNN passes node features as **messages** along edges;
- For instance, the node in red aggregates messages from its neighbors via edges connected to it (red arrows). An optional self-loop (the blue arrow) can also be added to it.



$$h_i' = \sigma\left(W_0 h_i + \sum_{j \in \mathcal{N}_i} W_1 h_j\right)$$

# Deep Graph Generative Modeling

1. Graph Neural Networks (GNNs)
   - **Graph Convolutional Networks (GCNs)**

# Graph Convolutional Networks (GCNs)

## The limitations of a minimal model

We can start from a minimal model with only the **matrix production** and **non-linear activation function**:

$$H^{(l+1)} = f\left(H^{(l)}, A\right) = \sigma\left(AH^{(l)}W^{(l)}\right),$$

where:

- $H^{(l)}$ is the hidden state of layer $l = 1, \ldots, L$ with $H^{(0)} = X$;
- $A$ is an adjacency matrix,
- $W^{(l)}$ is the weight matrix, and
- $\sigma(\cdot)$ is a non-linear activation function.

# Graph Convolutional Networks (GCNs)

## The limitations of a minimal model

$$H^{(l+1)} = f\left(H^{(l)}, A\right) = \sigma\left(AH^{(l)}W^{(l)}\right),$$

This model has two major limitations:

1) For every node, we sum up all the feature vectors of all neighbouring nodes but not the node itself (unless there is a self-loop);
2) The adjacency matrix $A$ is typically not normalized, and therefore the multiplication with $A$ will completely change the scale of $H^{(l)}$.

# Graph Convolutional Networks (GCNs)

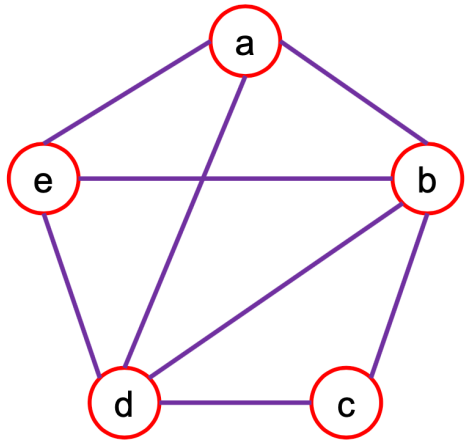
## The limitations of a minimal model

**Limitation 1:** For every node, we sum up all the feature vectors of all neighbouring nodes but not the node itself.

**Solution:** We can enforce self-loops in the graph by adding an identity matrix to $A$ and use $\tilde{A} = A + I_N$ in GCNs.

$A=$

| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | 0 | 1 | 0 | 1 | 1 |
| b | 1 | 0 | 1 | 1 | 1 |
| c | 0 | 1 | 0 | 1 | 0 |
| d | 1 | 1 | 1 | 0 | 1 |
| e | 1 | 1 | 0 | 1 | 0 |




$\hat{A}=$

| | a | b | c | d | e |
|---|---|---|---|---|---|
| a | 1 | 1 | 0 | 1 | 1 |
| b | 1 | 1 | 1 | 1 | 1 |
| c | 0 | 1 | 1 | 1 | 0 |
| d | 1 | 1 | 1 | 1 | 1 |
| e | 1 | 1 | 0 | 1 | 1 |

# Graph Convolutional Networks (GCNs)

## The limitations of a minimal model

**Limitation 2:** The adjacency matrix $A$ is typically not normalized, and therefore the multiplication with $A$ will completely change the scale of $H^{(l)}$.

**Solution:** We can normalize $A$ such that all rows in it sum to one. This can be reached by production with the inverse of its degree matrix: $D^{-1}A$. In practice, the **symmetric normalization** form, $D^{-1/2}AD^{-1/2}$, generates better results.

# Graph Convolutional Networks (GCNs)

## GCNs' layer propagation rule

Combining the two solutions, for layer $l = 1, \dots L$, the layer-wise propagation rule of GCNs is formally defined as

$$H^{(l+1)} = f\left(H^{(l)}, A\right) = \sigma\left(\widetilde{D}^{-1/2}\tilde{A}\widetilde{D}^{-1/2}H^{(l)}W^{(l)}\right),$$

where:

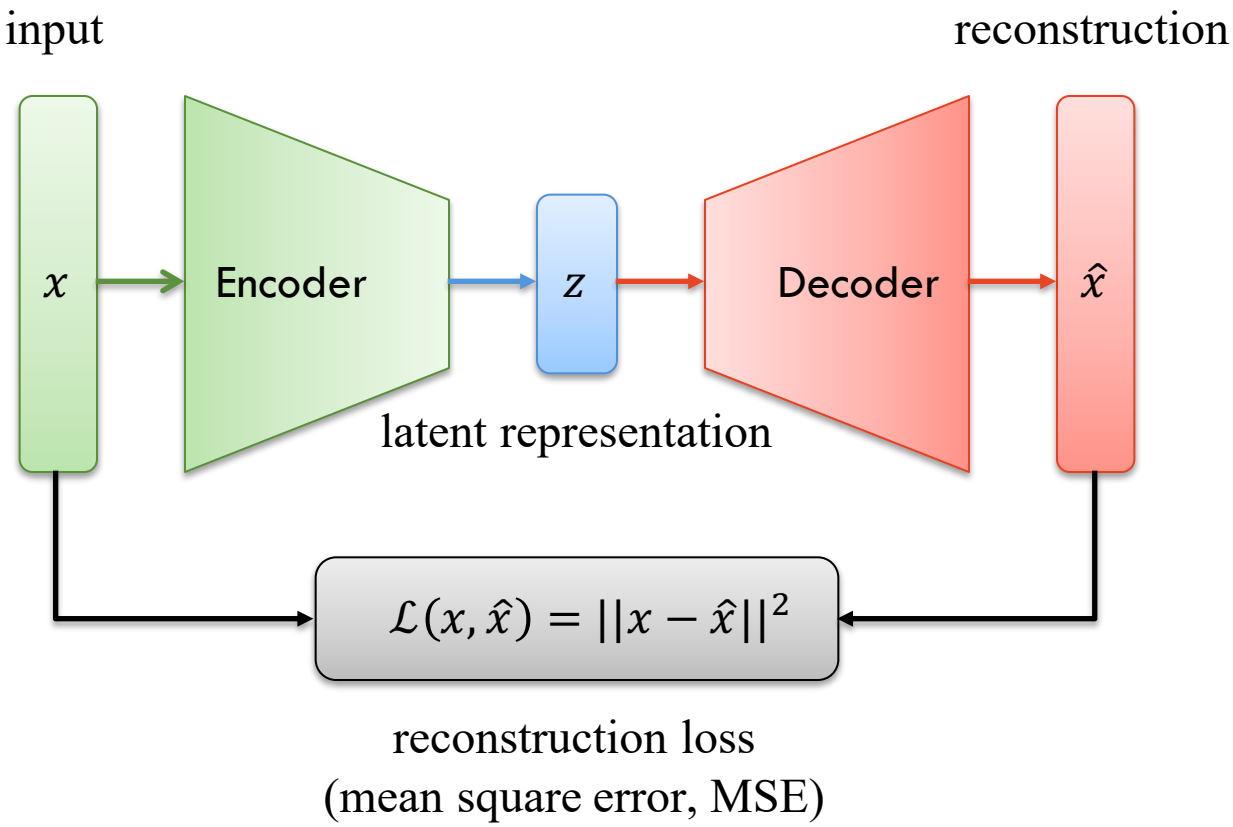
- $\tilde{A} = A + I_N$ is an adjacency matrix with self-loops,
- $\widetilde{D} = \sum_j \tilde{A}_{ij}$ is a diagonal degree matrix of $\tilde{A}$,
- $W^{(l)}$ is the weight matrix, and
- $\sigma(\cdot)$ is a non-linear activation function.

# Deep Graph Generative Modeling

1. Graph Neural Networks (GNNs)
   - Graph Convolutional Networks (GCNs)
2. **Variational Graph Autoencoders (VGAEs)**

# Variational Graph Autoencoders (VGAEs)

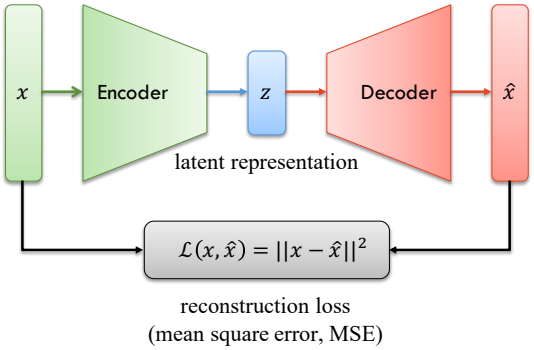
## Preliminary: autoencoders (AEs)



# Variational Graph Autoencoders (VGAEs)

**Can we use AEs to generate molecules?**

- An autoencoder aims to **reconstruct** its inputs;
- The MSE loss implies the output ($\hat{x}$) is expected to be **identical** to the input ($x$);
- However, we want to generate **unknown** molecules;
- Therefore, we need to slightly modify the autoencoder.



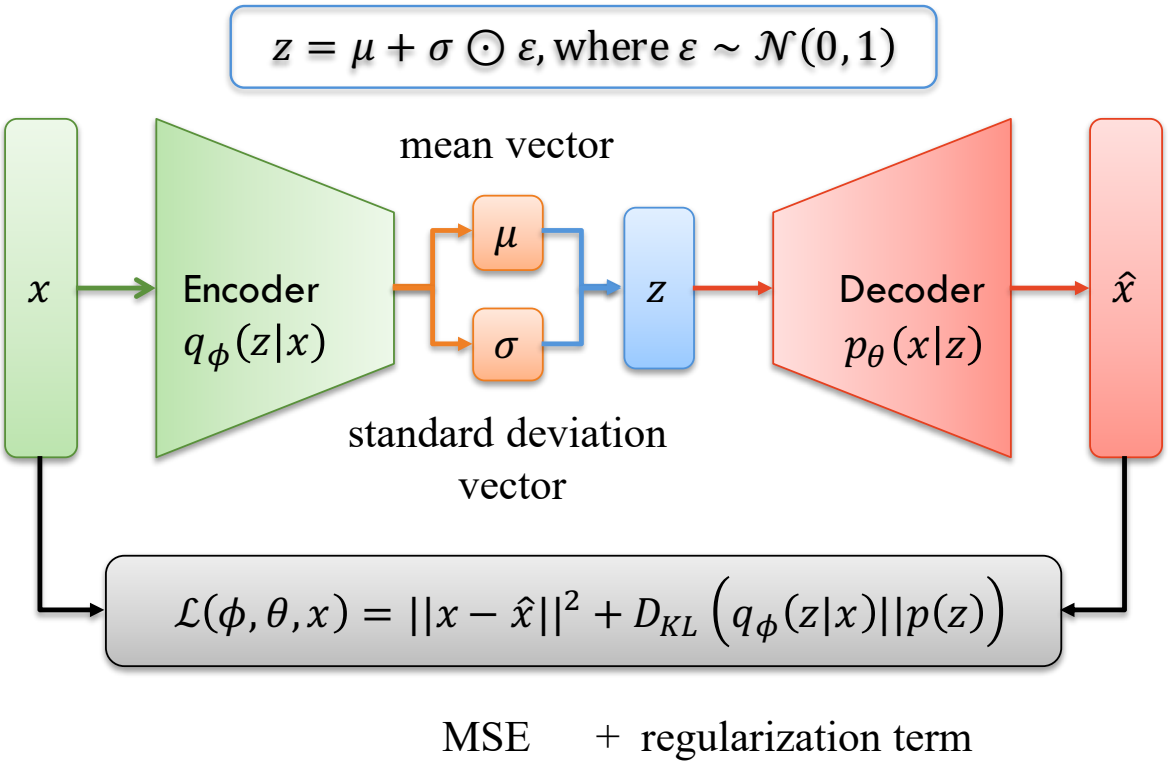
# Variational Graph Autoencoders (VGAEs)

**Variational Autoencoders (VAEs)**

- Instead of learning a fixed latent representation for each input, VAEs learn the **mean** and **standard** deviation of the latent distribution;
- Then, we can draw latent samples from the latent distribution to generate **unknown** outputs.

# Variational Graph Autoencoders (VGAEs)

## Variational Autoencoders (VAEs)




VAEs are a **probabilistic twist** on AEs.

# Variational Graph Autoencoders (VGAEs)

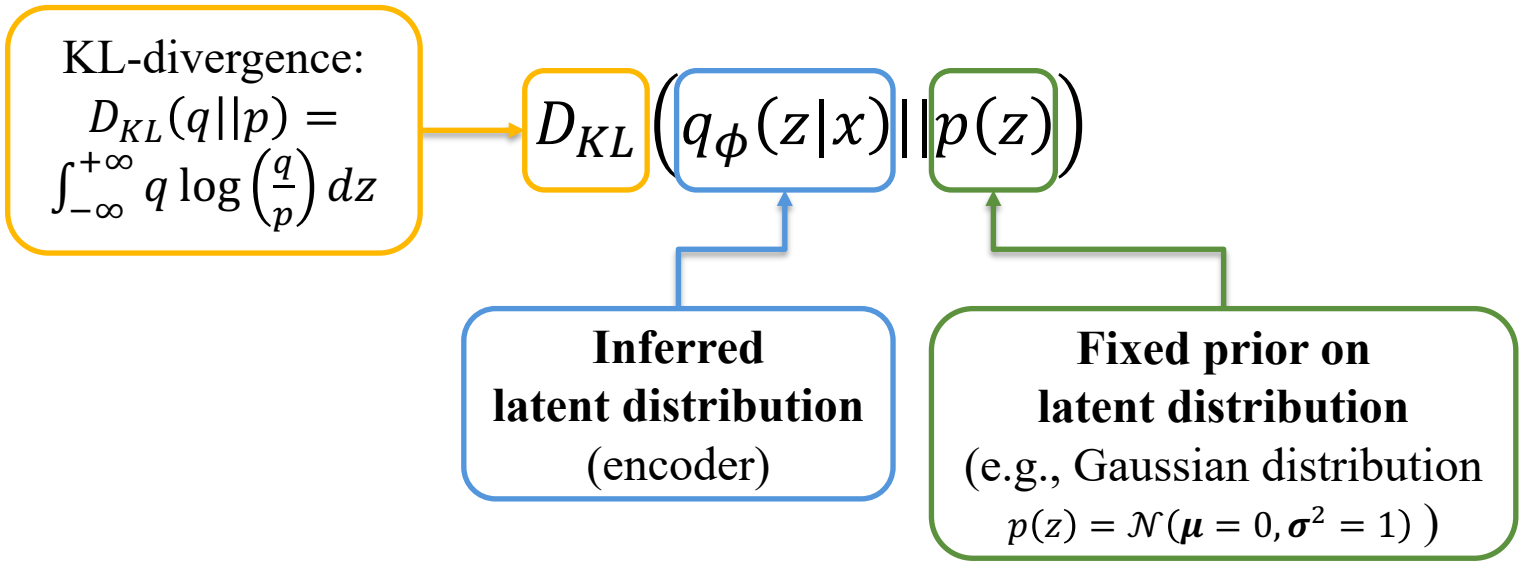
## Variational Autoencoders (VAEs)

The MSE term is identical to the reconstruction loss in AEs.

**But what is the regularization term?**

# Variational Graph Autoencoders (VGAEs)

## Regularization with KL-divergence

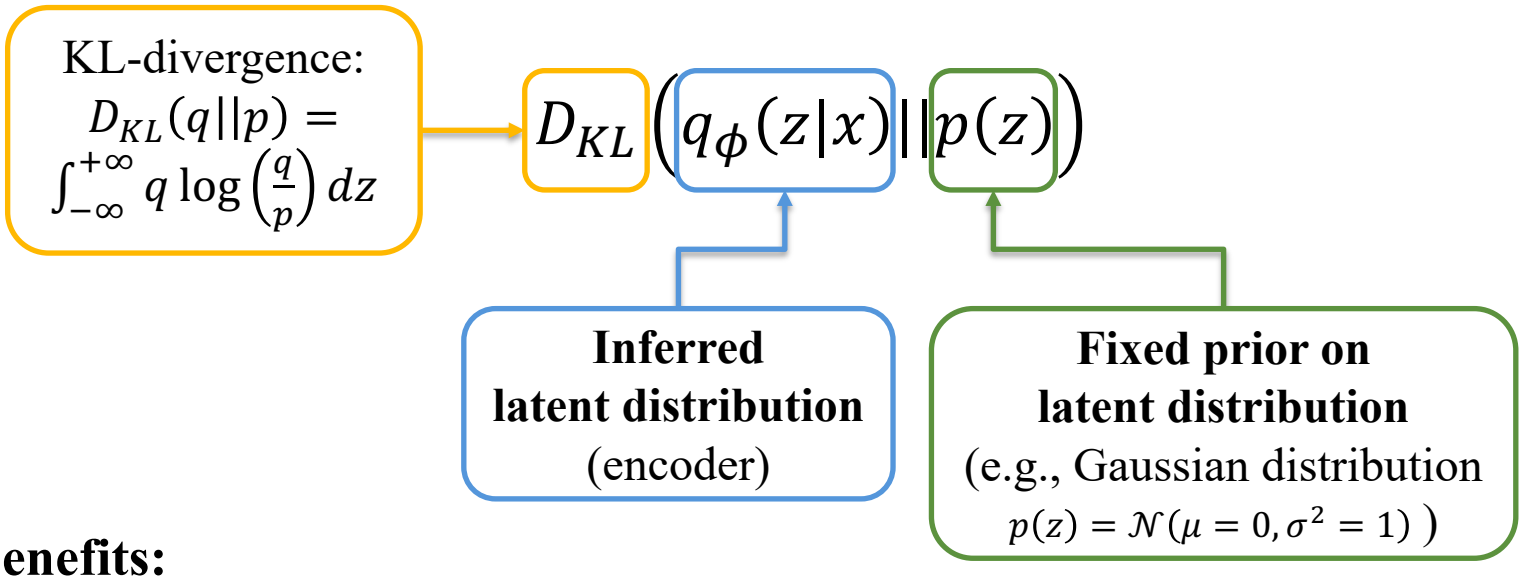



The **Kullback-Leibler (KL) divergence** is a measure of how one probability distribution ($q_\phi(z|x)$) is different from a another one ($p(z)$).

By minimizing the KL-divergence, we can enforce the inferred latent distribution to match a fixed Gaussian distribution.

# Variational Graph Autoencoders (VGAEs)

## Regularization with KL-divergence




**Benefits:**

1) It encourages the encoder to distribute its outputs evenly around the centre of the latent space;
2) It penalizes the generative model, if the model tries to "cheat" by clustering points in specific regions (i.e., by memorizing the data instead of generating new ones).

# Variational Graph Autoencoders (VGAEs)

**Encoder and decoder for graphs**

- GCN-based encoder:
  - $\boldsymbol{\mu} = \mathrm{GCN}_{\boldsymbol{\mu}}(X, A)$;
  - $\boldsymbol{\sigma} = \mathrm{GCN}_{\boldsymbol{\sigma}}(X, A)$.
- Inner-product decoder:
  - An inner-product between latent variables:

$$p(A|Z) = \prod_{i=1}^{N} \prod_{j=1}^{N} p\left(A_{ij}|z_i, z_j\right) \text{ with } p\left(A_{ij}|z_i, z_j\right) = \sigma\left(z_i^{\mathrm{T}} z_j\right).$$

# Large Language Models

# Large Language Models

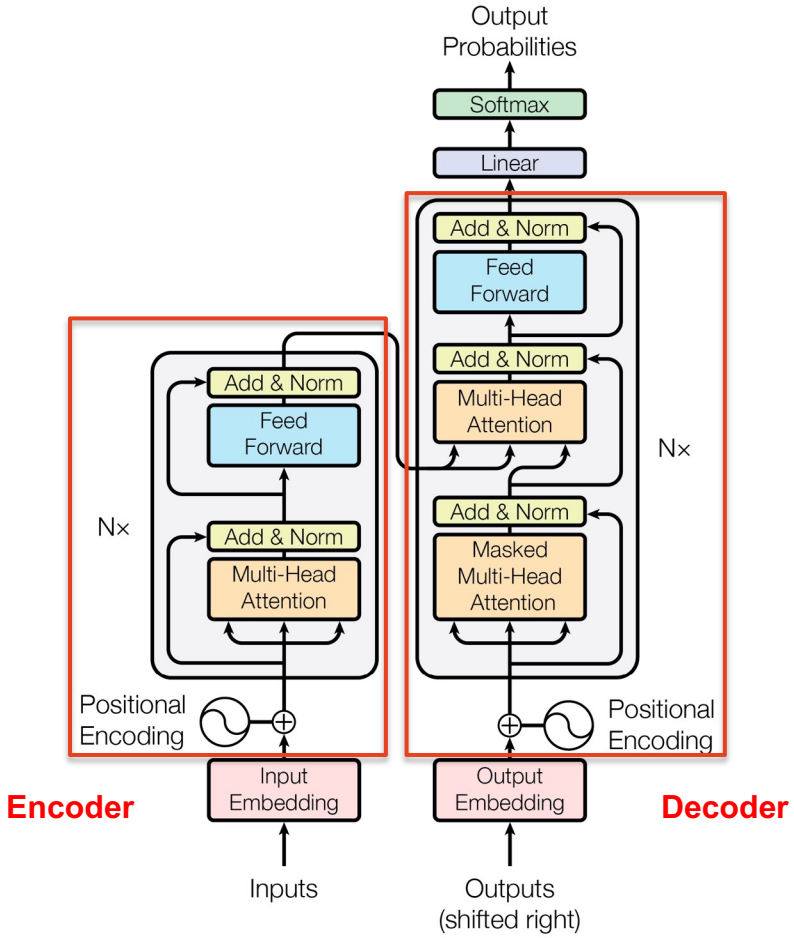
## 1. Generative Pretrained Transformer (GPT)

# Generative Pretrained Transformer (GPT)

## GPT

- GPT is a deep learning model for natural language understanding and generation.
- It is pre-trained on a large corpus of text from the internet, which helps it learn grammar, context, and the structure of language.
- After pre-training, GPT can be fine-tuned on specific tasks, such as text completion, language translation, or question answering.
- GPT's key innovation is its ability to generate coherent and contextually relevant text, making it versatile for various NLP tasks.

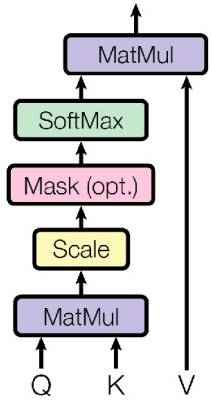
# Generative Pretrained Transformer (GPT)




- The Transformer consists of two main components: an **encoder** and a **decoder**.
- The **encoder** takes the input sequence and transforms it into a series of hidden representations using self-attention mechanisms.
- The **decoder** then generates the output sequence based on these hidden representations while attending to both the input sequence and previously generated output elements.

# Generative Pretrained Transformer (GPT)

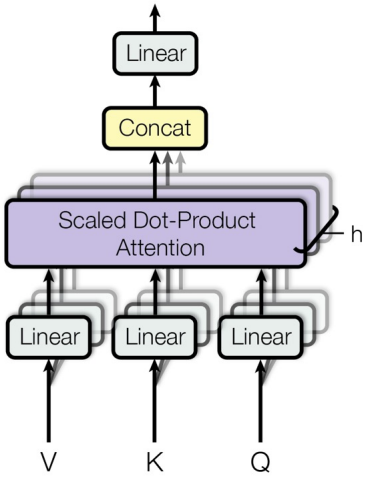
Scaled Dot-Product Attention




## Attention Mechanism

- At the core of the Transformer architecture is the attention mechanism, which allows the model to focus on different parts of the input sequence when processing it.
- The Transformer's attention mechanism can weigh the importance of each element in the input sequence for every element in the output sequence, enabling it to capture long-range dependencies effectively.

Multi-Head Attention




## Multi-Head Attention

- To capture different types of relationships in the data, the Transformer employs multi-head attention. It calculates attention scores with multiple sets of learned weight matrices (attention heads) in parallel.
- The outputs of these heads are concatenated and linearly transformed to produce the final attention output.

# Generative Pretrained Transformer (GPT)


## Self-Attention

- The Transformer uses self-attention, which means it computes attention scores within the input sequence itself.
- It calculates attention scores between each pair of input elements, producing a weighted sum that forms the output representation for each element in the sequence.
- This self-attention mechanism allows the Transformer to capture contextual information from all positions in the input sequence, making it highly parallelizable and efficient for processing long sequences.

# Generative Pretrained Transformer (GPT)

## ChatGPT

- ChatGPT is a specific implementation of the GPT model designed for conversational applications.
- It is fine-tuned on conversational data to make it more capable of understanding and generating human-like responses in a chat or dialogue format.
- ChatGPT can be used for a wide range of applications, including chatbots, virtual assistants, customer support, and more.
- OpenAI has developed ChatGPT models for different purposes and user needs, and they come in various versions and sizes.

# Generative Pretrained Transformer (GPT)

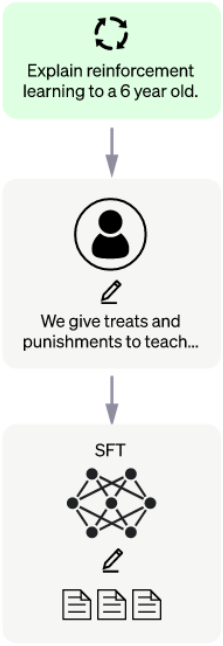
Step 1

**Collect demonstration data and train a supervised policy.**

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.



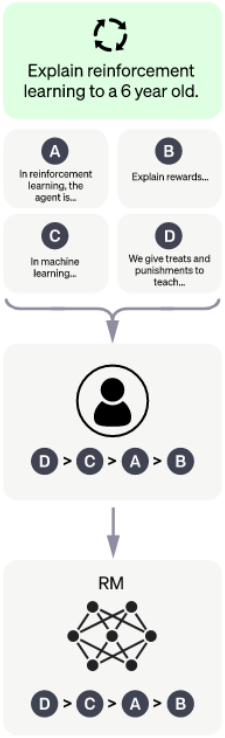

Step 2

**Collect comparison data and train a reward model.**

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.




Step 3

**Optimize a policy against the reward model using the PPO reinforcement learning algorithm.**

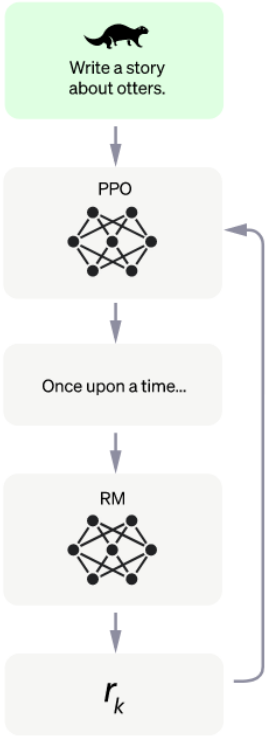
A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

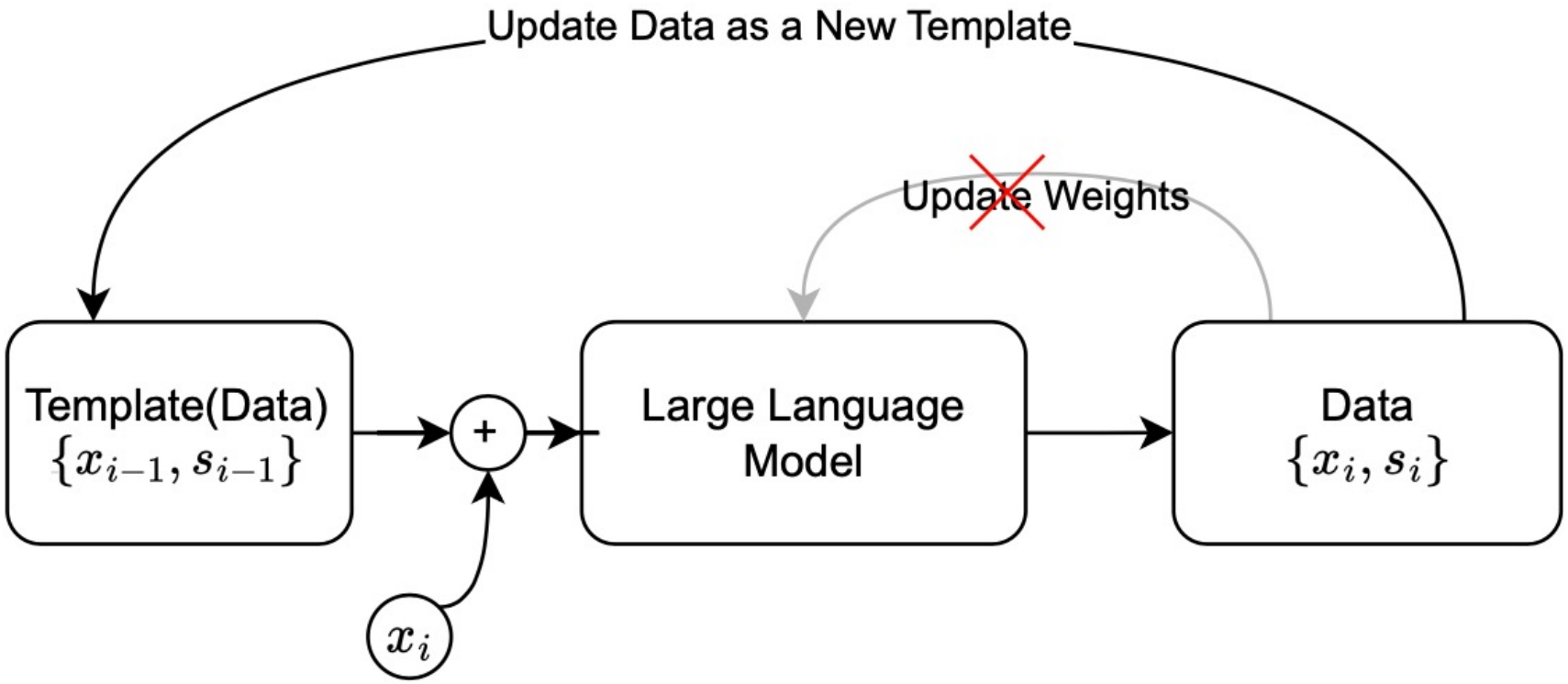


# Large Language Model

1. Generative Pretrained Transformer (GPT)
2. **Self-supervised Molecule Annotation***

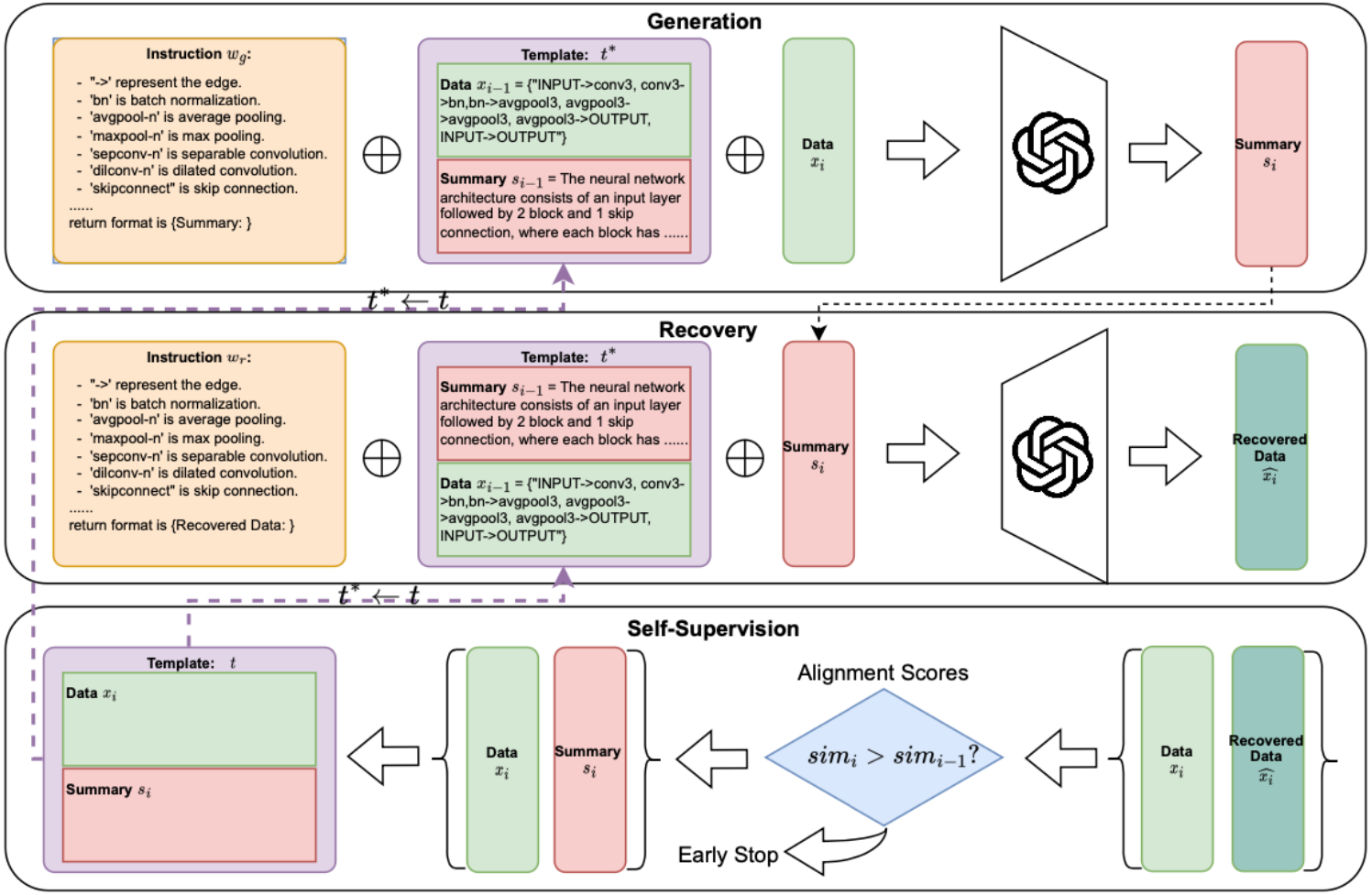
* GPT Self-Supervision for a Better Data Annotator. https://arxiv.org/abs/2306.04349

# Self-supervised Molecule Annotation

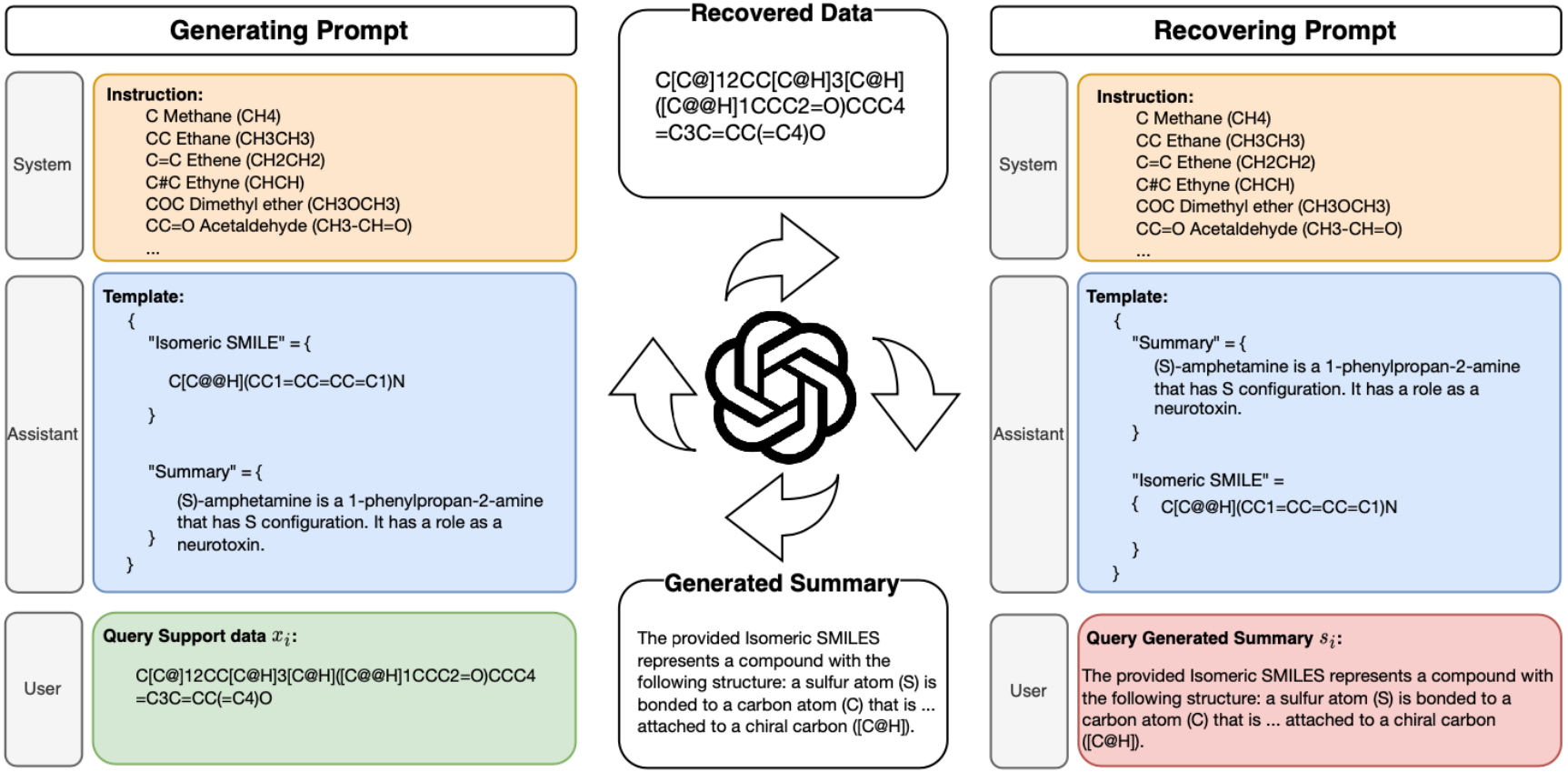



A data-centric generating-recovering paradigm.

# Self-supervised Molecule Annotation



# Self-supervised Molecule Annotation



# Self-supervised Molecule Annotation

**Instruction for generating summary on the PubMed dataset.**

You are a professional annotator of Isomeric SMILES based on the organic compound description. In SMILES, atoms are represented by their atomic symbols.

The second letter of two-character atomic symbols must be entered in lower case. Each non-hydrogen atom is specified independently by its atomic symbol enclosed in square brackets, [ ] (for example, [Au] or [Fe]). Square brackets may be omitted for elements

in the "organic subset" (B, C, N, O, P, S, F, Cl, Br, and I) if the proper number of "implicit" hydrogen atoms is assumed. "Explicitly" attached hydrogens and formal charges are always specified inside brackets. A formal charge is represented by one of the symbols + or -. Single, double, triple, and aromatic bonds are represented by the symbols, -, =, #, respectively. Single and aromatic bonds may be, and usually are, omitted.

C Methane (CH4)

CC Ethane (CH3CH3)

C=C Ethene (CH2CH2)

C#C Ethyne (CHCH)

COC Dimethyl ether (CH3OCH3)

CCO Ethanol (CH3CH2OH)

CC=O Acetaldehyde (CH3-CH=O)

C#N Hydrogen Cyanide (HCN)

[C-]#N Cyanide anion

I will provide **an Isomeric SMILES** you return **a summary**. Your return format is a json dict: **{Summary: }**

# Self-supervised Molecule Annotation

**Instruction for recovering data on the PubMed dataset.**

You are a professional annotator of Isomeric SMILES based on the organic compound description. In SMILES, atoms are represented by their atomic symbols.

The second letter of two-character atomic symbols must be entered in lower case. Each non-hydrogen atom is specified independently by its atomic symbol enclosed in square brackets, [ ] (for example, [Au] or [Fe]). Square brackets may be omitted for elements

in the "organic subset" (B, C, N, O, P, S, F, Cl, Br, and I) if the proper number of "implicit" hydrogen atoms is assumed. "Explicitly" attached hydrogens and formal charges are always specified inside brackets. A formal charge is represented by one of the symbols + or -. Single, double, triple, and aromatic bonds are represented by the symbols, -, =, #, respectively. Single and aromatic bonds may be, and usually are, omitted.

C Methane (CH4)

CC Ethane (CH3CH3)

C=C Ethene (CH2CH2)

C#C Ethyne (CHCH)

COC Dimethyl ether (CH3OCH3)

CCO Ethanol (CH3CH2OH)

CC=O Acetaldehyde (CH3-CH=O)

C#N Hydrogen Cyanide (HCN)

[C-]#N Cyanide anion

I will provide **a description**, you return **an Isomeric SMILES**. The return format is json dict: **{Isomeric SMILES: }**

Thank you.

THE UNIVERSITY OF
SYDNEY