Signals in the Cells

Multimodal and Contextualized Machine Learning Foundations for Therapeutics

Alejandro "Alex" Velez-Arce

Zitnik Lab @ Harvard

alejandro_velez-arce@hms.harvard.edu

contact@tdcommons.ai

Drug discovery AI datasets and benchmarks have not traditionally included single-cell analysis biomarkers. While benchmarking efforts in single-cell analysis have recently released collections of single-cell tasks, they have yet to comprehensively release datasets, models, and benchmarks that integrate a broad range of therapeutic discovery tasks with cell-type-specific biomarkers. Therapeutics Commons (TDC-2) presents datasets, tools, models, and benchmarks integrating cell-type-specific contextual features with ML tasks across therapeutics. We present four tasks for contextual learning at single-cell resolution: drug-target nomination, genetic perturbation response prediction, chemical perturbation response prediction, and protein-peptide interaction prediction. We introduce datasets, models, and benchmarks for these four tasks. Finally, we detail the advancements and challenges in machine learning and biology that drove the implementation of TDC-2 and how they are reflected in its architecture, datasets and benchmarks, and foundation model tooling.

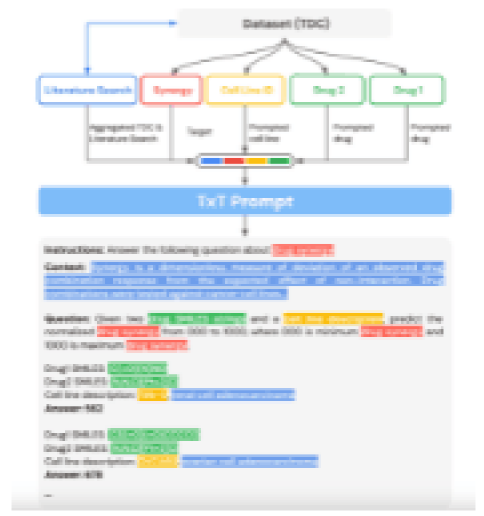
Introduction

Converging advances in therapeutics and AI

Therapeutic modalities | Foundation models | Single-cell analysis

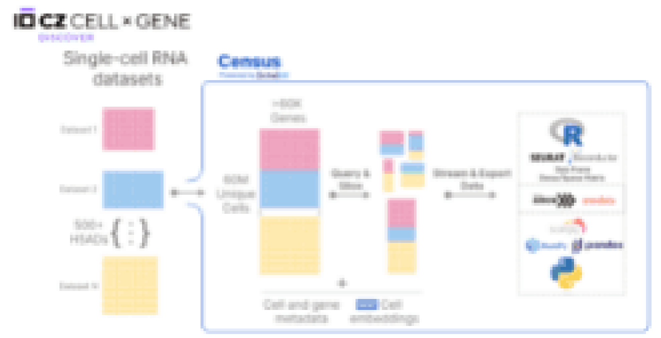
Therapeutic Foundation Models

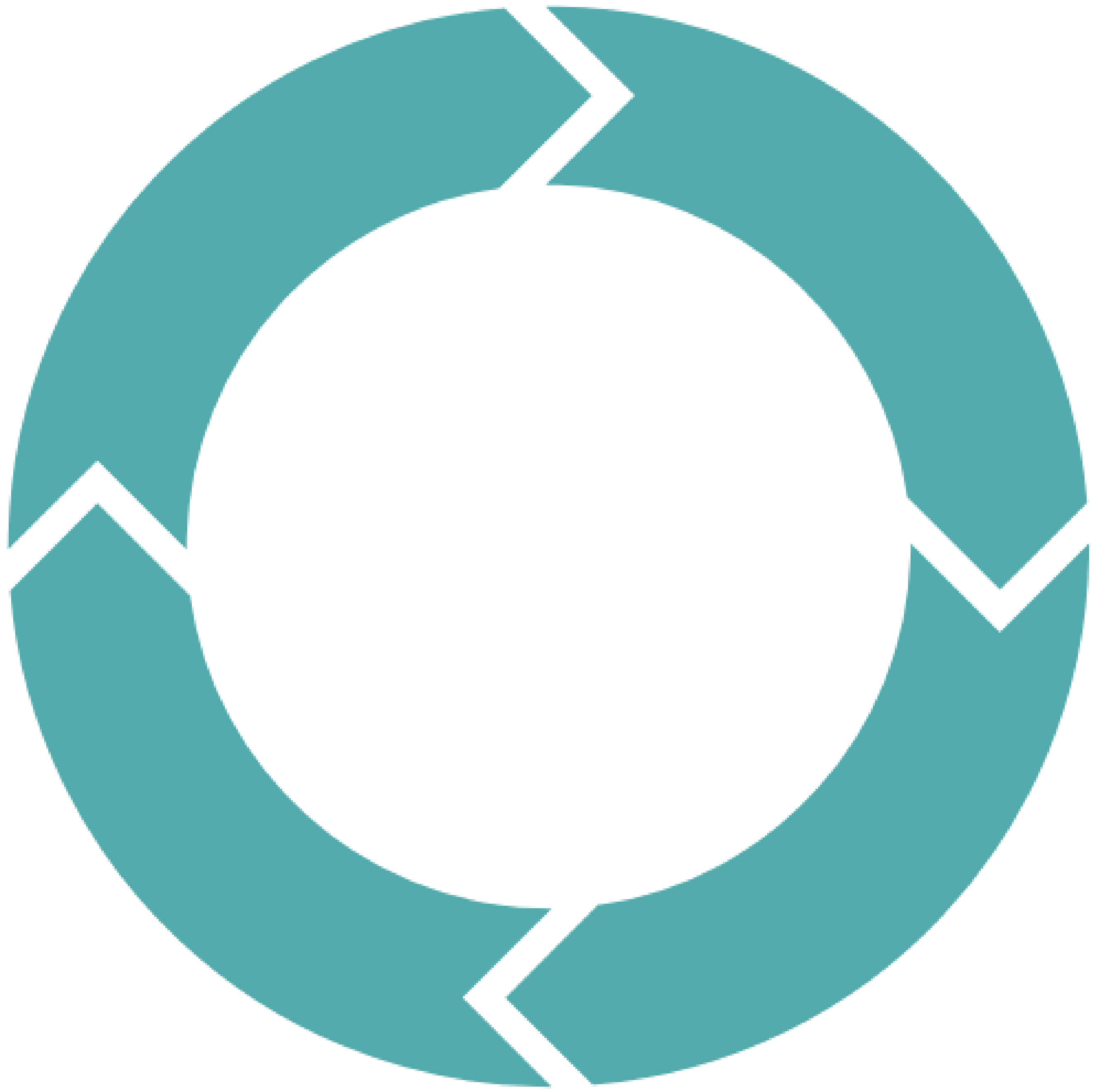
Foundation models trained on TDC have been shown to generalize across several therapeutic tasks



Single-cell Data and Machine Learning

Training foundation models on large single-cell atlases have shown a potential to advance cell type annotation and matching of healthy-disease cells to study cellular signatures of disease





LLM-based Workflows in biomedicine

LLMs succeed at using chemistry tools for tasks. LLM multi-agent frameworks have succeeded at automating single-cell analysis tasks

Contextual AI

In therapeutics, there is evidence that the effects of drugs can vary depending on the type of cell they are targeting and where specific proteins are acting

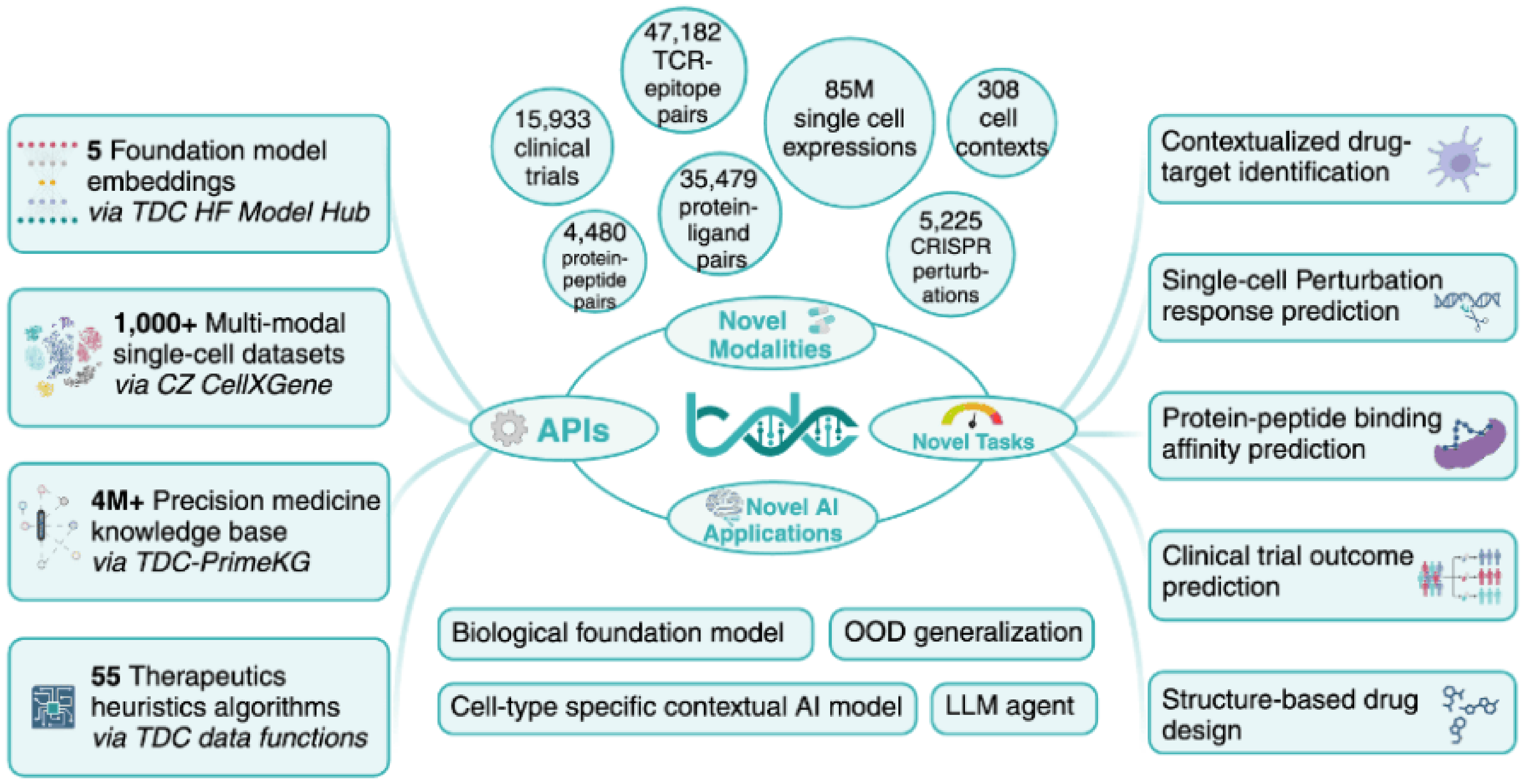

5 Foundation model embeddings via TDC HF Model Hub
1,000+ Multi-modal single-cell datasets via CZ CellXGene
4M+ Precision medicine knowledge base via TDC-PrimeKG
55 Therapeutics heuristics algorithms via TDC data functions
47,182 TCR-epitope pairs
15,933 clinical trials
85M single cell expressions
308 cell contexts
35,479 protein-ligand pairs
4,480 protein-peptide pairs
5,225 CRISPR perturbations
Novel Modalities
APIs
Novel Tasks
Novel AI Applications
Biological foundation model
OOD generalization
Cell-type specific contextual AI model
LLM agent
Contextualized drug-target identification
Single-cell Perturbation response prediction
Protein-peptide binding affinity prediction
Clinical trial outcome prediction
Structure-based drug design

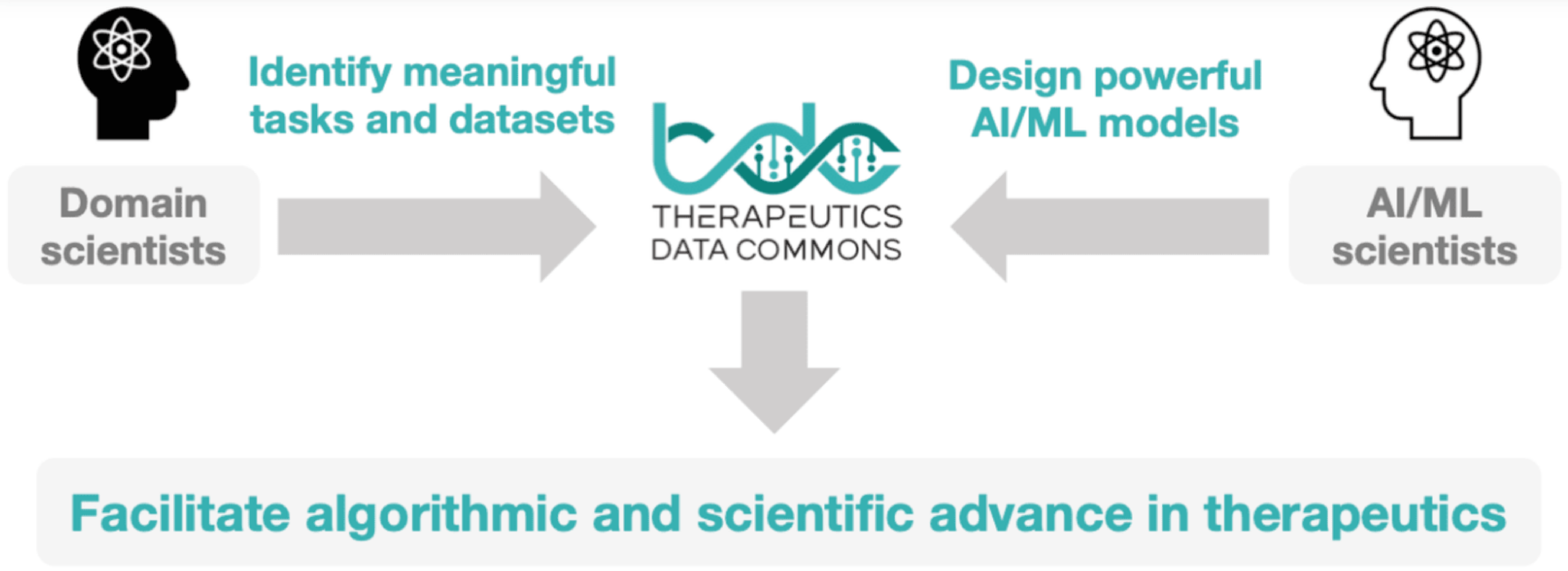
Figure 1: **Overview of TDC-2.** TDC-2 introduces a multimodal retrieval API powering ML-task-driven [11] datasets [69, 4, 67, 107, 91, 90, 14, 15, 109, 108]and benchmarks spanning 10+ new modalities and 5 state-of-the-art machine learning tasks (section 7.2), including 4 contextual AI tasks: TDC.scDTI (section 3.1), single-cell genetic perturbation response prediction (section 3.2.1), single-cell chemical perturbation response prediction (section 3.2.2), and single-cell protein-peptide interaction prediction (section 3.3). Model benchmarks highlighting biomedical AI challenges in OOD Generalization [26, 27, 120, 14] and evaluation [4, 30] of cell-type-specific contextual AI models are introduced.

Introduction

Background on TDC

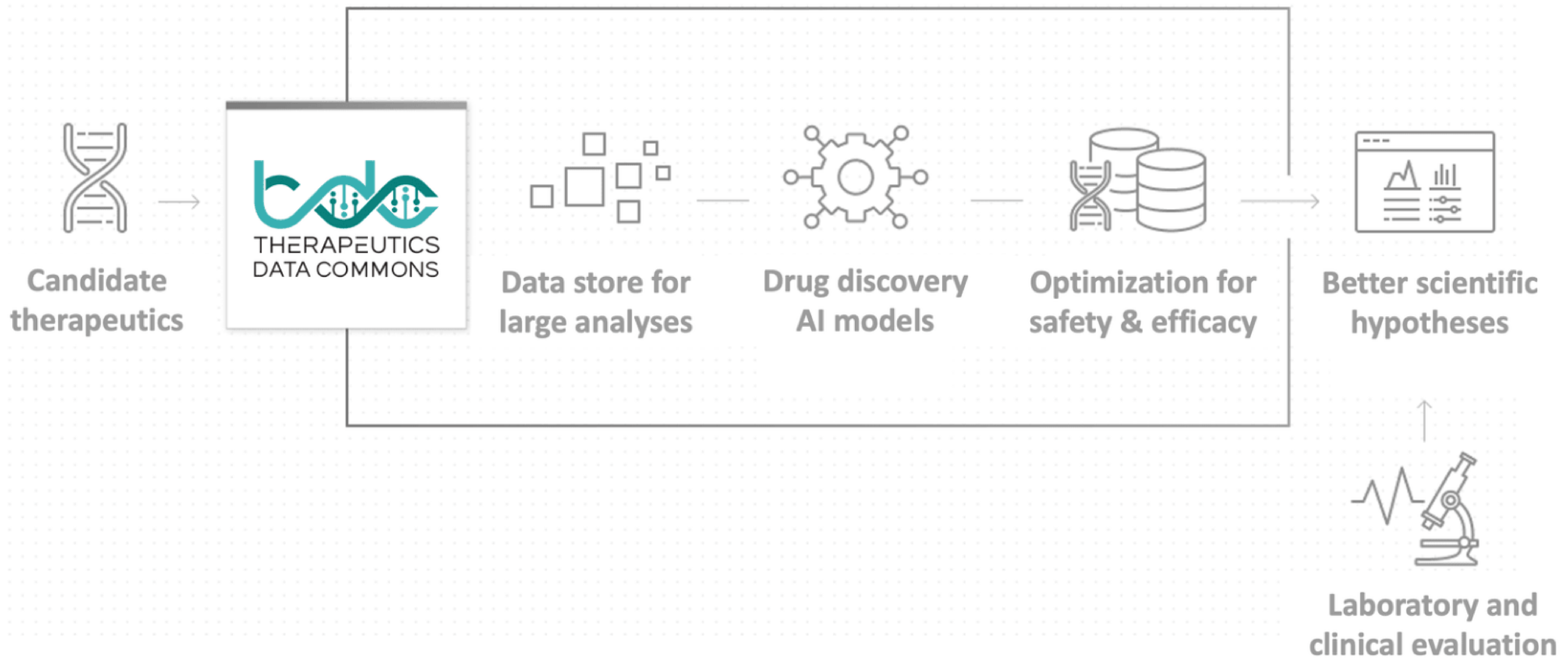
Enable algorithmic and scientific advance in therapeutic science

The Commons lies at the nexus between artificial intelligence and drug discovery. Biologists and biochemists can pose ML tasks and identify relevant datasets that are carefully processed and integrated into Commons and formulated as scientifically valid ML tasks. ML scientists can rapidly obtain these tasks and develop ML methods to advance the therapeutic task past the state of the art and open up new opportunities.


Identify meaningful tasks and datasets
Design powerful AI/ML models
Domain scientists
THERAPEUTICS DATA COMMONS
AI/ML scientists
Facilitate algorithmic and scientific advance in therapeutics

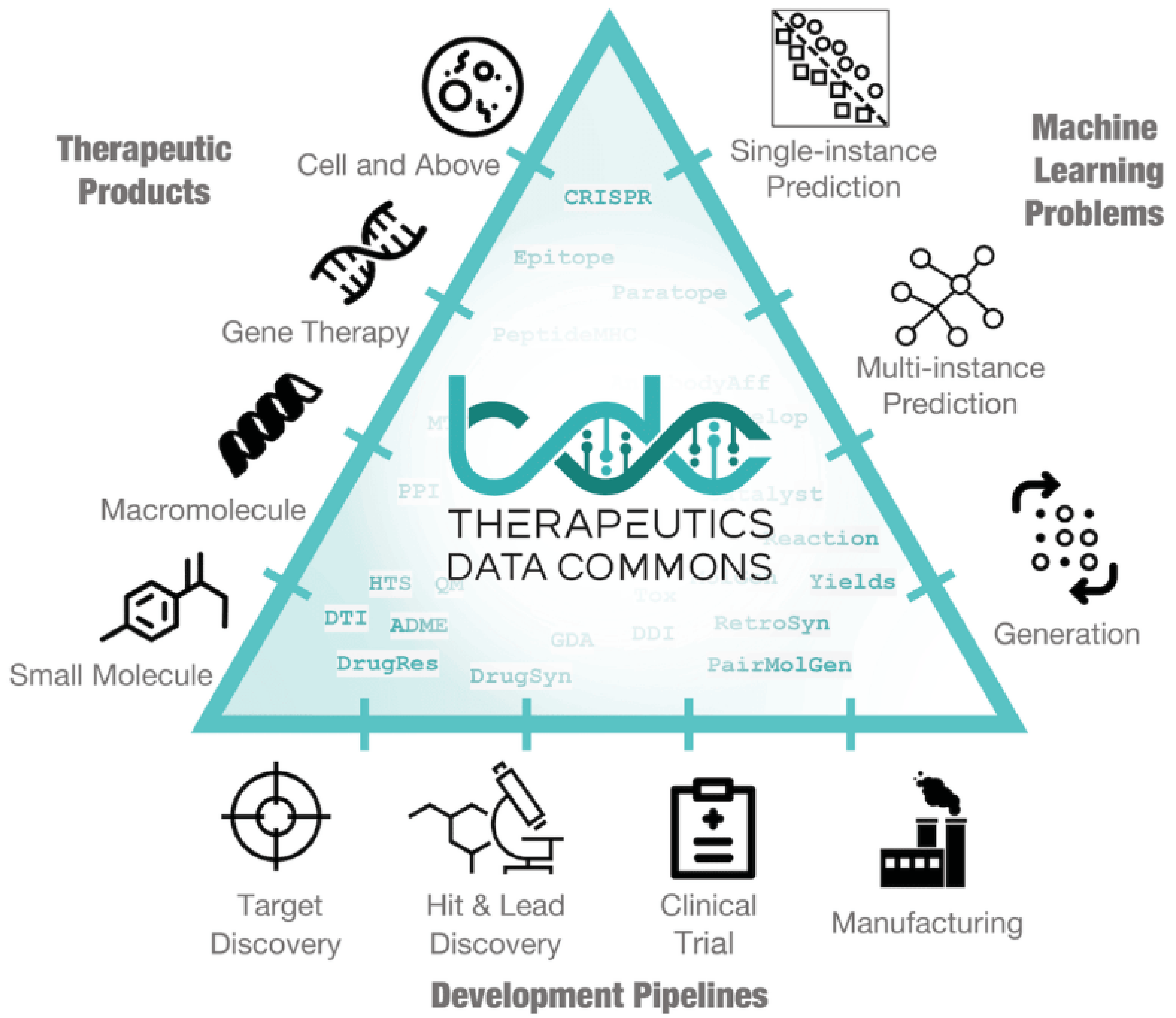
Introduction

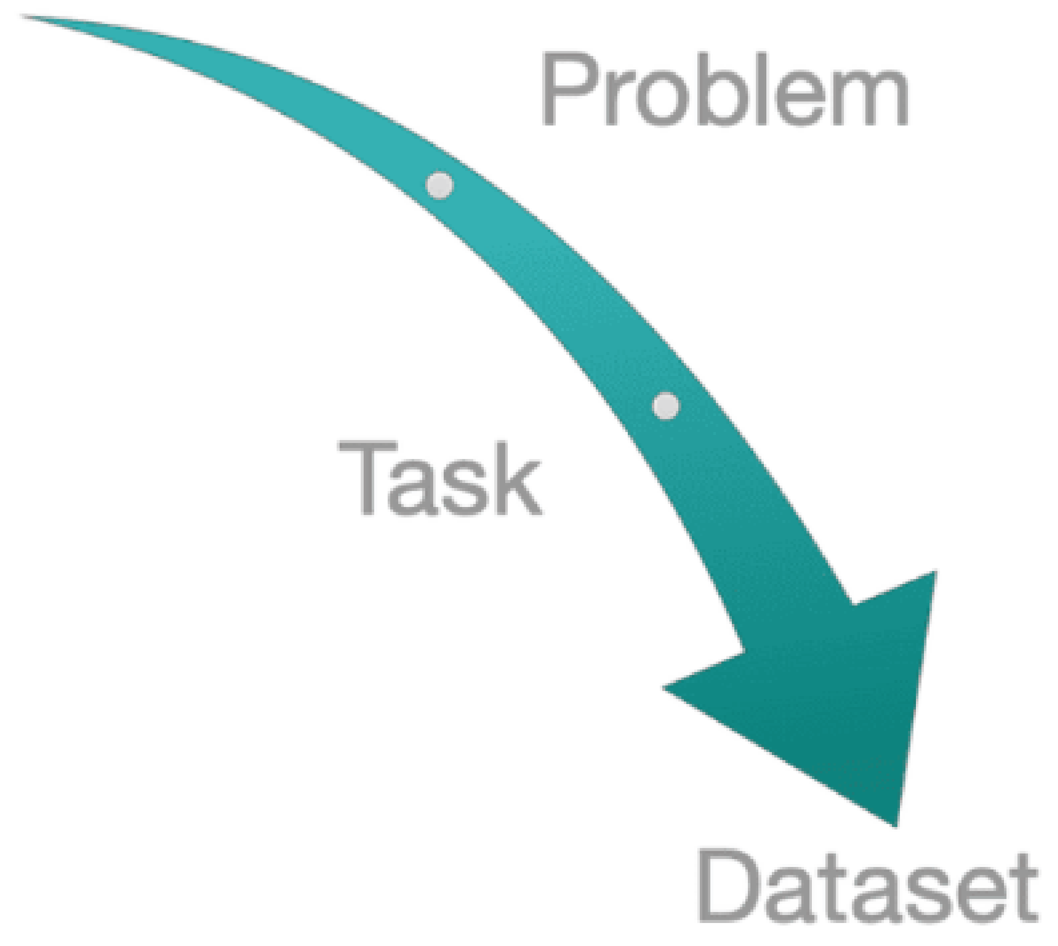
TDC in the drug discovery pipeline

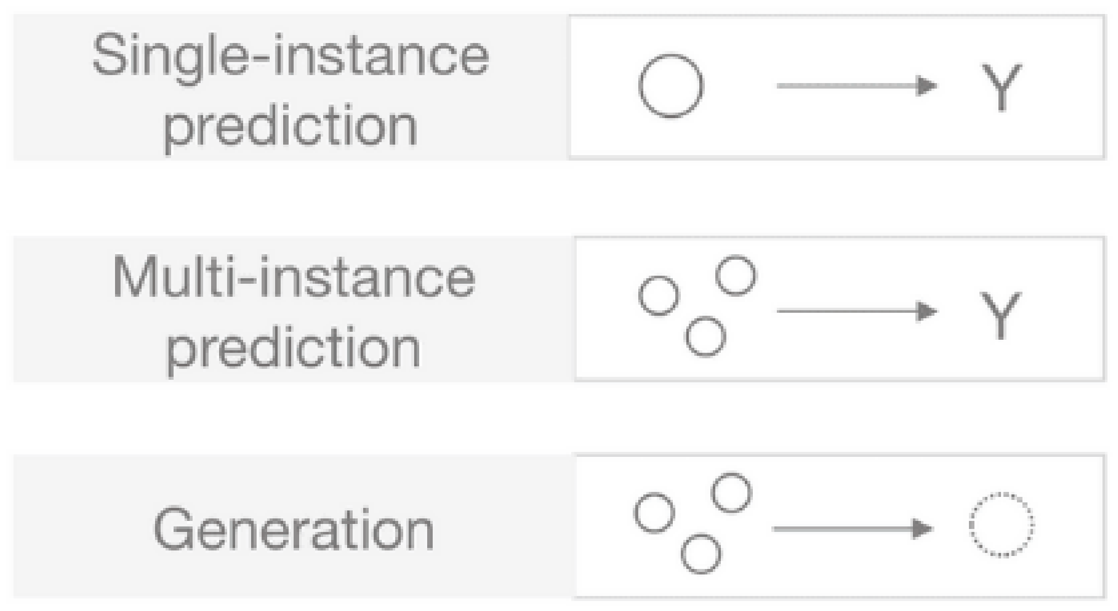

Candidate therapeutics
THERAPEUTICS DATA COMMONS
Data store for large analyses
Drug discovery AI models
Optimization for safety & efficacy
Better scientific hypotheses
Laboratory and clinical evaluation

Introduction

TDC at the intersection of problems, products, and pipelines

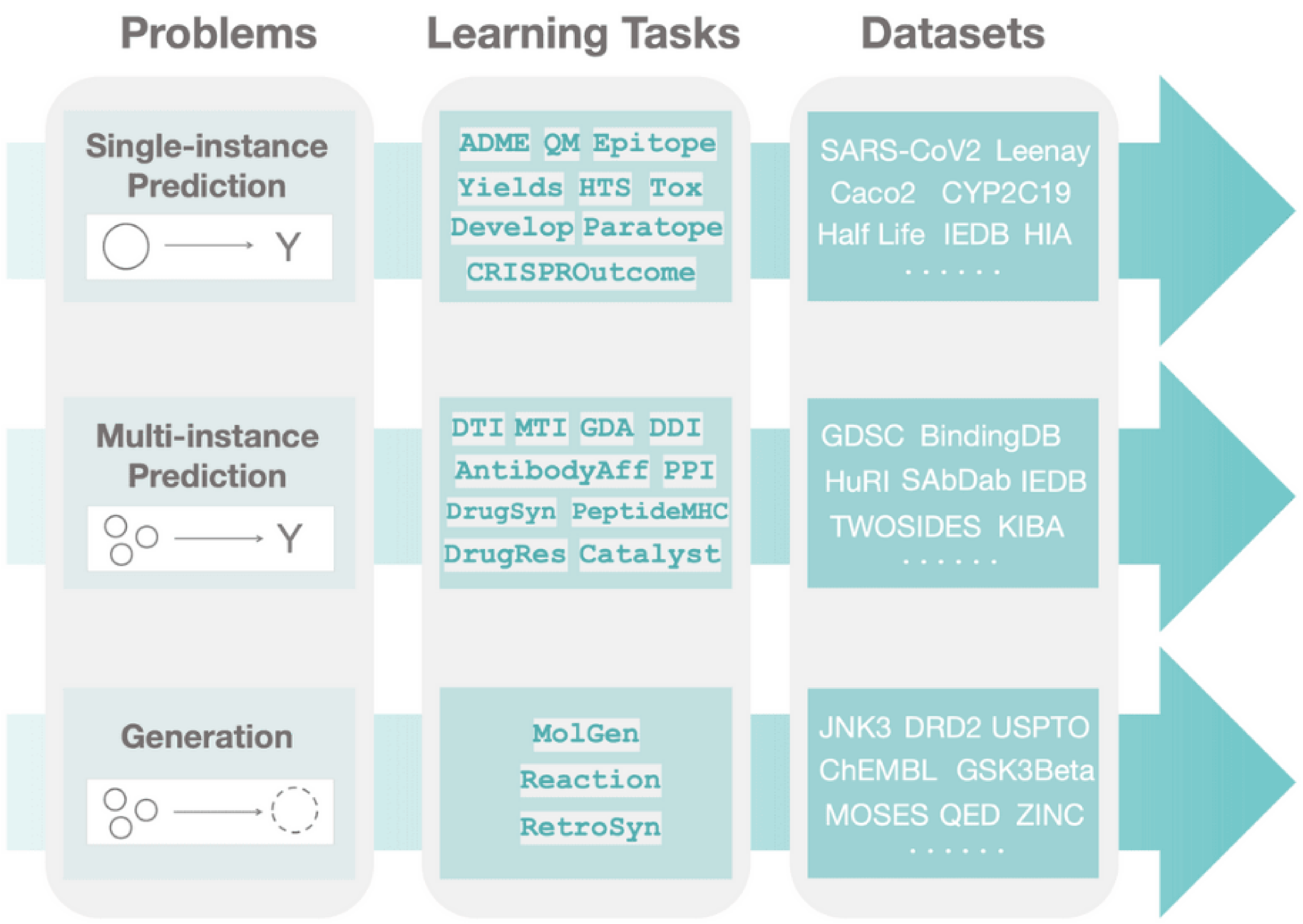






Introduction

TDC Datasets

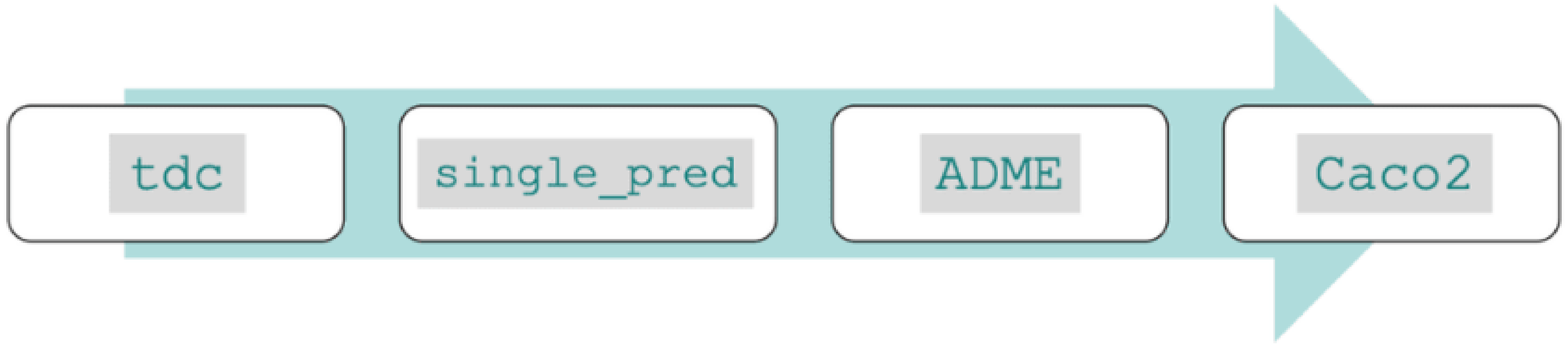


Introduction

TDC API

Data Loaders

TDC provides intuitive, high-level APIs for both beginners and experts to create ML models in Python. Building off the modularized "Problem--ML Task--Dataset" structure, TDC provides a three-layer API to access any ML task and dataset.



As an example, to obtain the `Caco2` dataset from `ADME` task in the `single-instance prediction` problem do as follows:

```
from tdc.single_pred import ADME
data = ADME(name = 'Caco2_Wang')
df = data.get_data()
splits = data.get_split()
```

The variable `df` is a Pandas object holding the entire dataset. By default, the variable `splits` is a dictionary with keys `train` , `val` , and `test` whose values are all Pandas DataFrames with Drug IDs, SMILES strings and labels. For detailed information about outputs, see Datasets documentation.

The user only needs to specify "Problem -- ML Task -- Dataset." TDC then automatically retrieves the processed ML-ready dataset from the TDC server and generates a `data` object, exposing numerous data functions that can be directly applied to the dataset.

Introduction

Converging advances in therapeutics and AI

Therapeutic modalities | Foundation models | Single-cell analysis

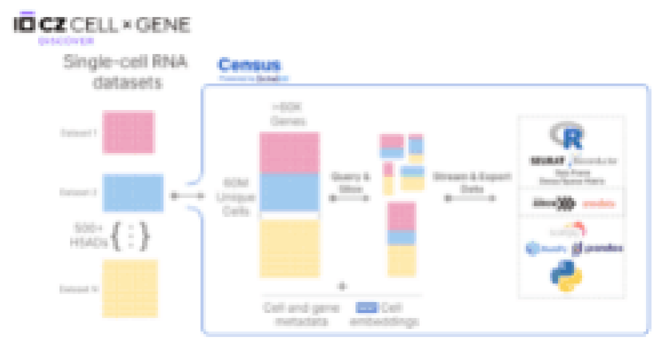
Therapeutic Foundation Models

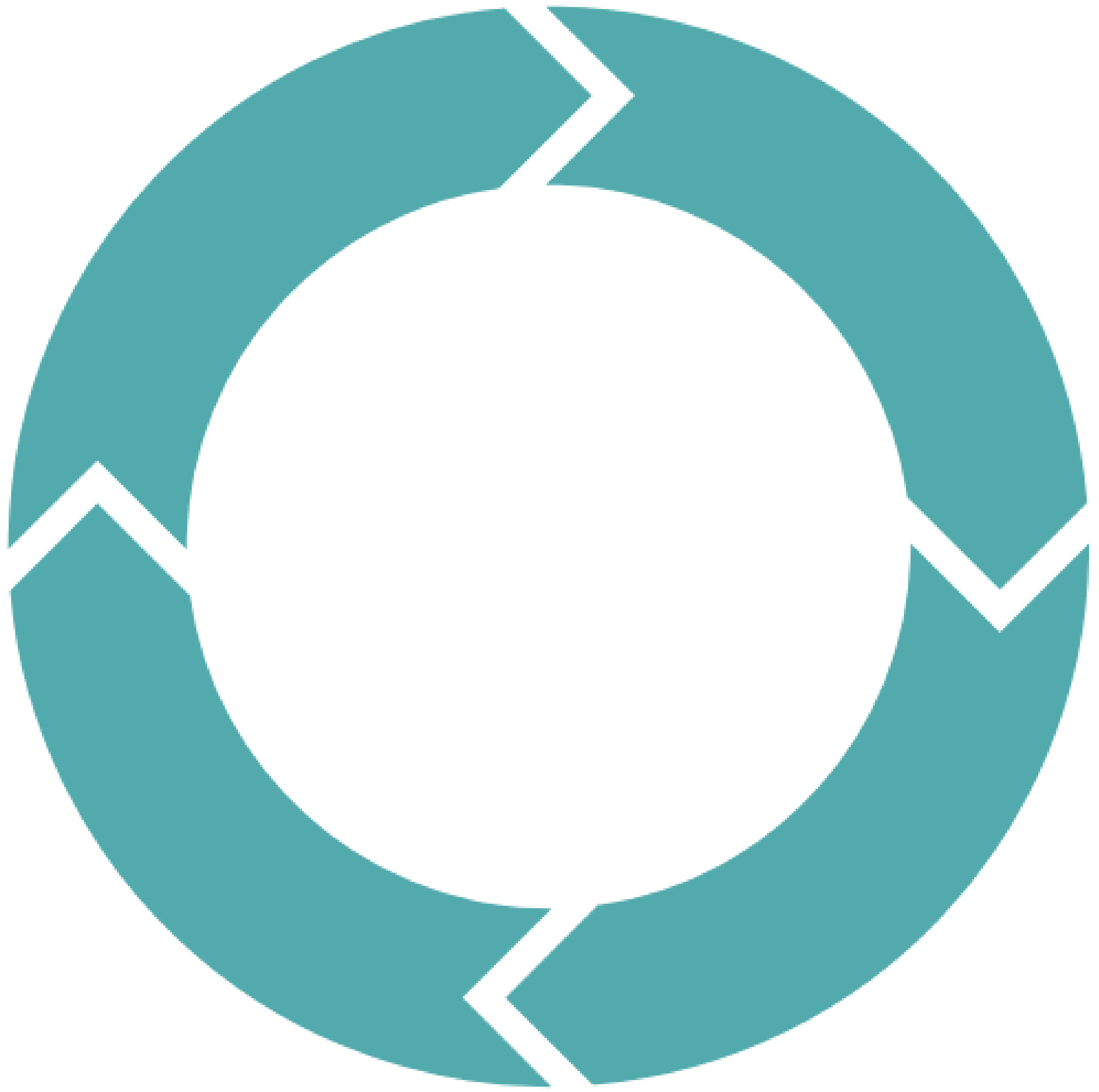
Foundation models trained on TDC have been shown to generalize across several therapeutic tasks

Dataset (TDC)
TxT Prompt

Single-cell Data and Machine Learning

Training foundation models on large single-cell atlases have shown a potential to advance cell type annotation and matching of healthy-disease cells to study cellular signatures of disease





LLM-based Workflows in biomedicine

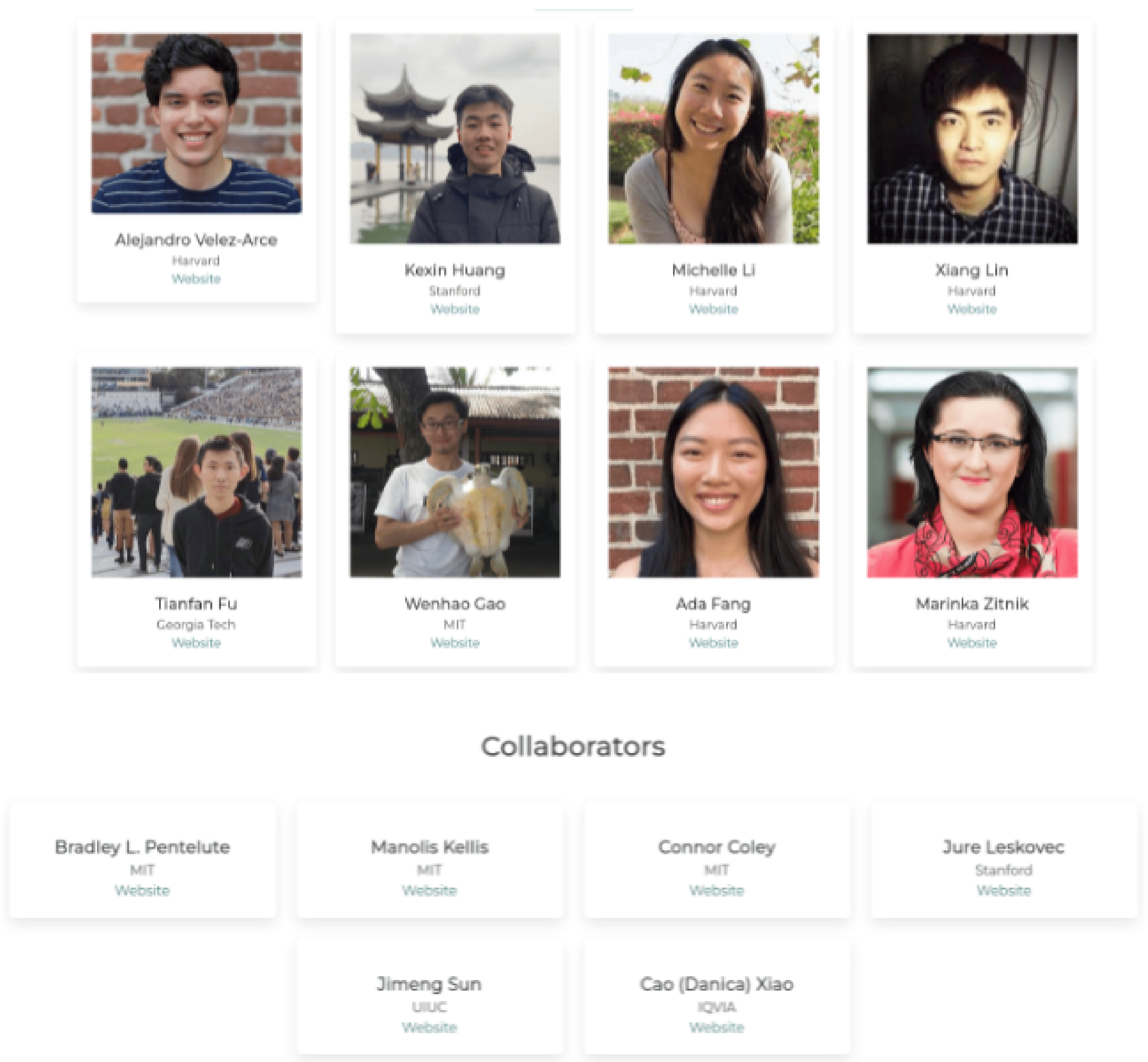
LLMs succeed at using chemistry tools for tasks. LLM multi-agent frameworks have succeeded at automating single-cell analysis tasks

Contextual AI

In therapeutics, there is evidence that the effects of drugs can vary depending on the type of cell they are targeting and where specific proteins are acting

Therapeutics Commons

TDC-2



Alejandro Velez-Arce
Harvard
Website

Kexin Huang
Stanford
Website

Michelle Li
Harvard
Website

Xiang Lin
Harvard
Website

Tianfan Fu
Georgia Tech
Website

Wenhao Gao
MIT
Website

Ada Fang
Harvard
Website

Marinka Zitnik
Harvard
Website

Collaborators

Bradley L. Pentelute
MIT
Website

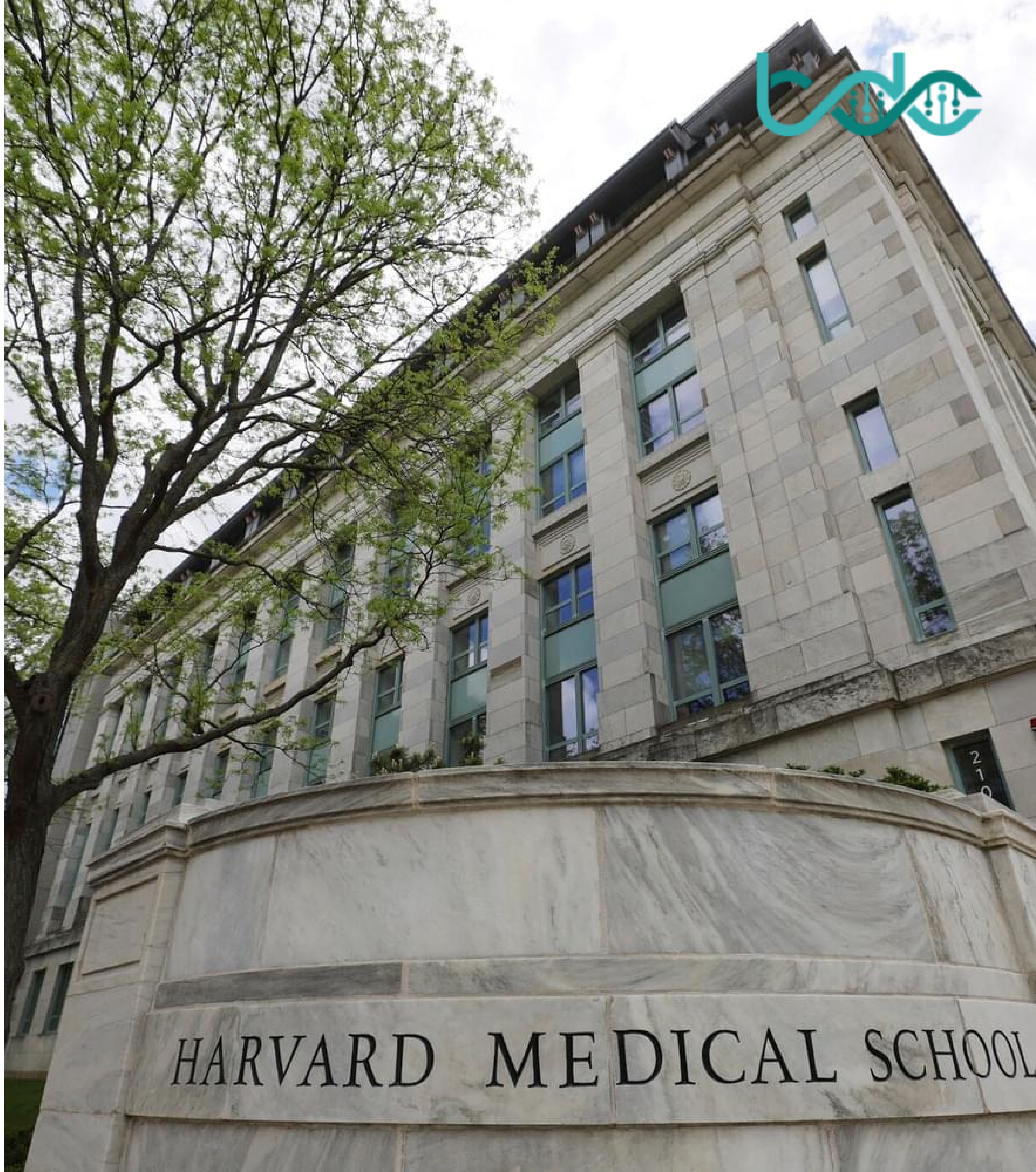
Manolis Kellis
MIT
Website

Connor Coley
MIT
Website

Jure Leskovec
Stanford
Website

Jimeng Sun
UIUC
Website

Cao (Danica) Xiao
IQVIA
Website



Outline

Models, datasets, benchmarks, architecture, ML Platform, and future directions

Single-cell Therapeutics

- Integrated single-cell analysis and therapeutics ML tasks.
- Contextual AI, metrics, and benchmarking.
- Multimodal single-cell retrieval API

API-first-dataset Architecture

- Challenges with continually updated datasets, heterogeneous data sources, and retrieval APIs.
- Model-View-Controller Design.
- Resource modules (knowledge graphs, scFM embedding retrieval, etc.)

Model Server and TDC ML Platform

- TDC Huggingface Model Hub.
- Model server design.
- Models, demos, and code

Future Directions

- Expand multimodal benchmarking
- Multimodal molecular encoders and modality fusion
- Arenas for molecular property prediction

Single-cell Therapeutic AI Tasks

drug-target nomination, genetic/chemical perturbation response prediction, protein-peptide binding affinity

TDC.scDTI Single-cell Drug-Target Identification (Nomination)

Motivation

In therapeutics, there is evidence that the effects of drugs can vary depending on the type of cell they are targeting and where specific proteins are acting.

Definition

The predictive task is defined as learning an estimator for a **disease-specific function of a target protein and cell type** outputting whether the candidate protein is a therapeutic target in that cell type.

Evaluation

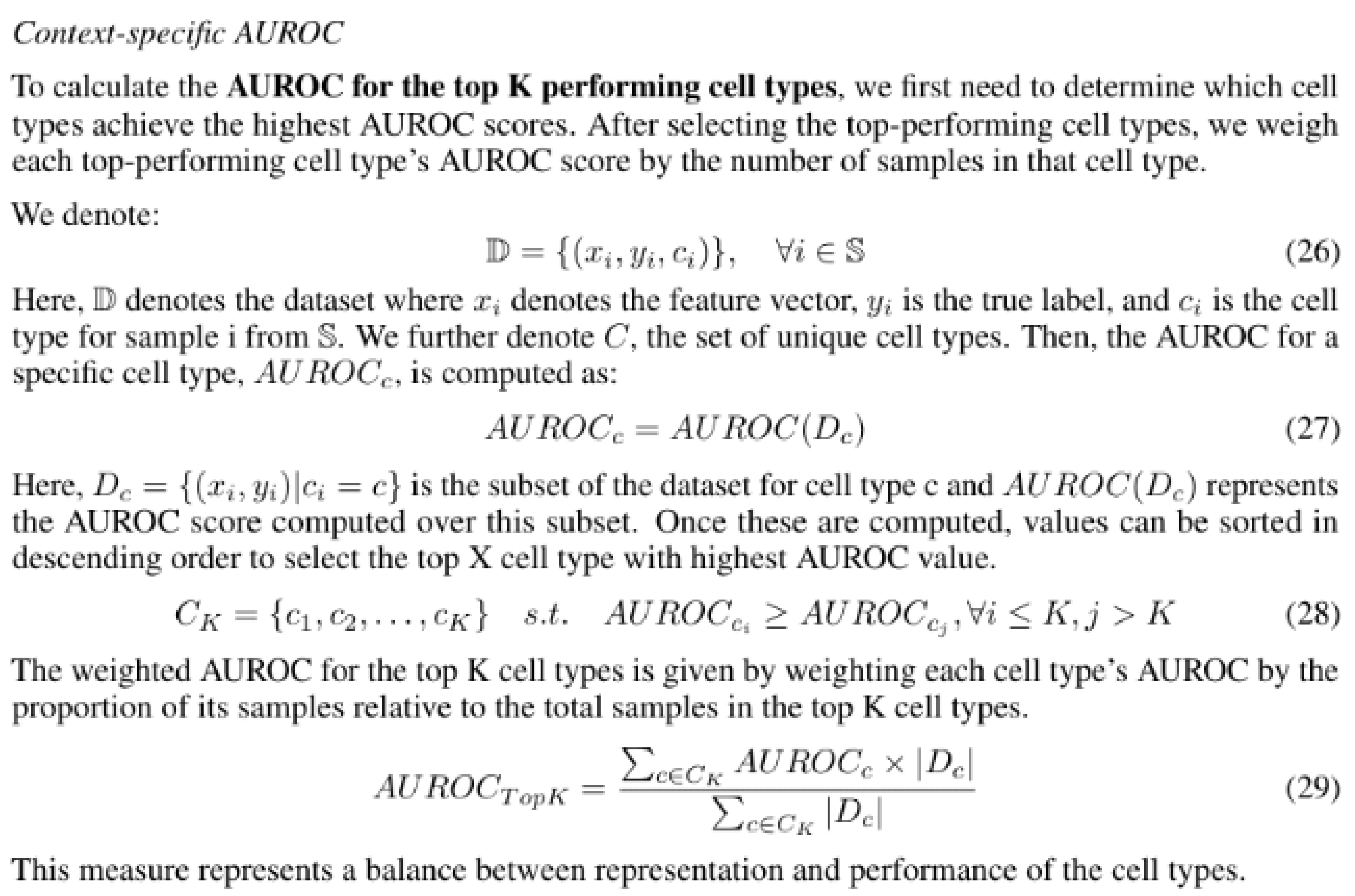
Models' performance is measured across sets of disease-specific proteins and cells. We compute **contextualized metrics** at top-performing cell types.

$$\hat{y} = f_\theta(t \in \mathbb{T}, c \in \mathbb{C}).$$

TDC.scDTI task formulation

Contextualized Metrics

Context-specific metrics are defined to **measure model performance at critical biological slices**, with our benchmarks focused on measuring cell-type-specific model performance. For single-cell drug-target nomination, we measure model performance at top-performing cell types.



Context-specific AUROC

To calculate the **AUROC for the top K performing cell types**, we first need to determine which cell types achieve the highest AUROC scores. After selecting the top-performing cell types, we weigh each top-performing cell type's AUROC score by the number of samples in that cell type.

We denote:

$$\mathbb{D} = \{(x_i, y_i, c_i)\}, \quad \forall i \in \mathbb{S} \tag{26}$$

Here, $\mathbb{D}$ denotes the dataset where x_i denotes the feature vector, y_i is the true label, and c_i is the cell type for sample i from $\mathbb{S}$. We further denote C, the set of unique cell types. Then, the AUROC for a specific cell type, $AUROC_c$, is computed as:

$$AUROC_c = AUROC(D_c) \tag{27}$$

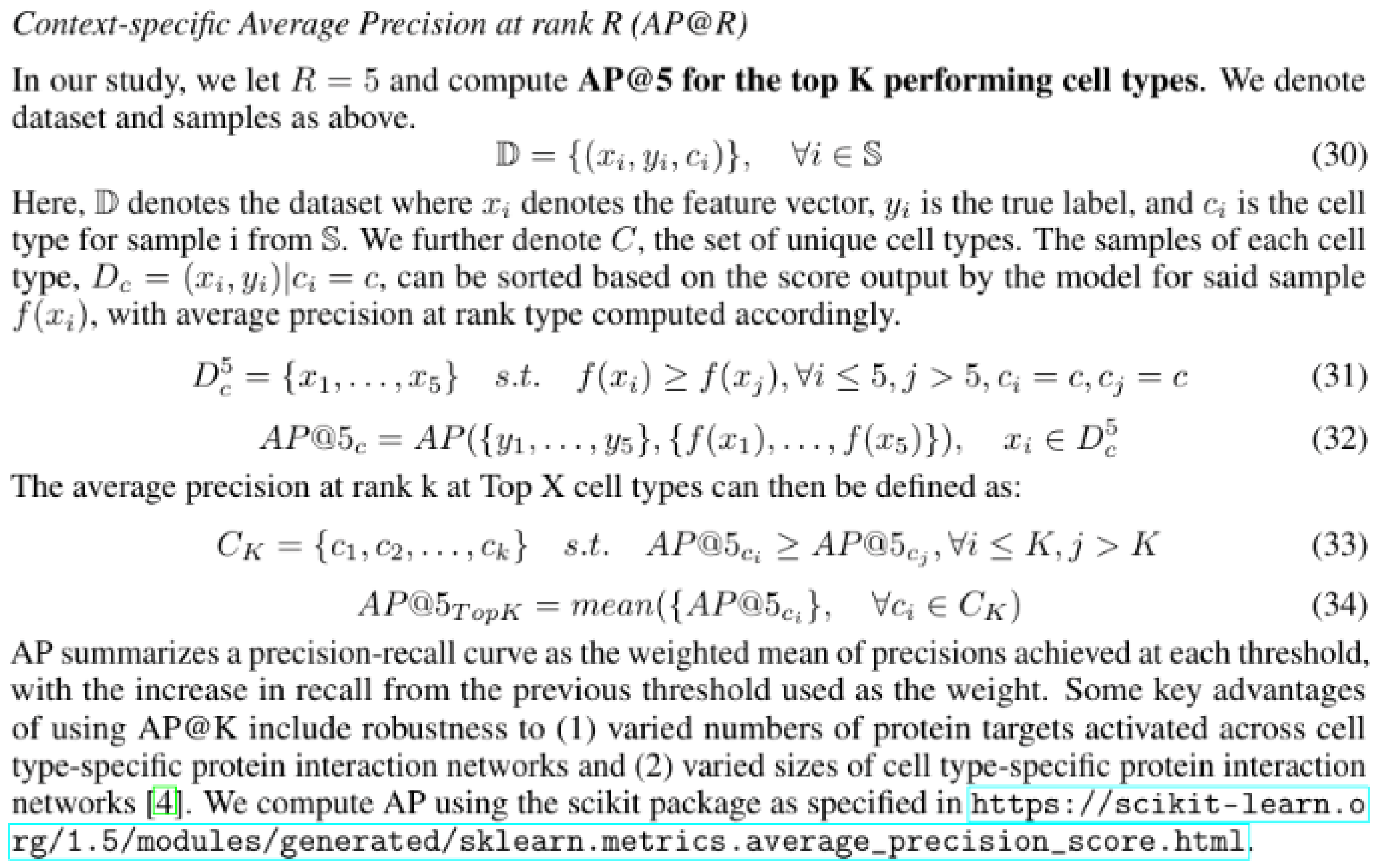
Here, $D_c = \{(x_i, y_i) | c_i = c\}$ is the subset of the dataset for cell type c and $AUROC(D_c)$ represents the AUROC score computed over this subset. Once these are computed, values can be sorted in descending order to select the top X cell type with highest AUROC value.

$$C_K = \{c_1, c_2, \ldots, c_K\} \quad s.t. \quad AUROC_{c_i} \geq AUROC_{c_j}, \forall i \leq K, j > K \tag{28}$$

The weighted AUROC for the top K cell types is given by weighting each cell type's AUROC by the proportion of its samples relative to the total samples in the top K cell types.

$$AUROC_{TopK} = \frac{\sum_{c \in C_K} AUROC_c \times |D_c|}{\sum_{c \in C_K} |D_c|} \tag{29}$$

This measure represents a balance between representation and performance of the cell types.



Context-specific Average Precision at rank R (AP@R)

In our study, we let $R = 5$ and compute **AP@5 for the top K performing cell types**. We denote dataset and samples as above.

$$\mathbb{D} = \{(x_i, y_i, c_i)\}, \quad \forall i \in \mathbb{S} \tag{30}$$

Here, $\mathbb{D}$ denotes the dataset where x_i denotes the feature vector, y_i is the true label, and c_i is the cell type for sample i from $\mathbb{S}$. We further denote C, the set of unique cell types. The samples of each cell type, $D_c = (x_i, y_i) | c_i = c$, can be sorted based on the score output by the model for said sample $f(x_i)$, with average precision at rank type computed accordingly.

$$D_c^5 = \{x_1, \ldots, x_5\} \quad s.t. \quad f(x_i) \geq f(x_j), \forall i \leq 5, j > 5, c_i = c, c_j = c \tag{31}$$

$$AP@5_c = AP(\{y_1, \ldots, y_5\}, \{f(x_1), \ldots, f(x_5)\}), \quad x_i \in D_c^5 \tag{32}$$

The average precision at rank k at Top X cell types can then be defined as:

$$C_K = \{c_1, c_2, \ldots, c_k\} \quad s.t. \quad AP@5_{c_i} \geq AP@5_{c_j}, \forall i \leq K, j > K \tag{33}$$

$$AP@5_{TopK} = mean(\{AP@5_{c_i}\}, \quad \forall c_i \in C_K) \tag{34}$$

AP summarizes a precision-recall curve as the weighted mean of precisions achieved at each threshold, with the increase in recall from the previous threshold used as the weight. Some key advantages of using AP@K include robustness to (1) varied numbers of protein targets activated across cell type-specific protein interaction networks and (2) varied sizes of cell type-specific protein interaction networks [4]. We compute AP using the scikit package as specified in `https://scikit-learn.org/1.5/modules/generated/sklearn.metrics.average_precision_score.html`.

(Li, Michelle, et al.)

Dataset Description: To curate target information for a therapeutic area, we examine the drugs indicated for the therapeutic area of interest and its descendants. The two therapeutic areas examined are rheumatoid arthritis (RA) and inflammatory bowel disease. Positive examples (i.e., where the label y = 1) are proteins targeted by drugs that have at least completed phase 2 of clinical trials for treating a specific therapeutic area. As such, a protein is a promising candidate if a compound that targets the protein is safe for humans and effective for treating the disease. We retain positive training examples activated in at least one cell type-specific protein interaction network. We define negative examples (i.e., where the label y = 0) as druggable proteins that do not have any known association with the therapeutic area of interest according to Open Targets. A protein is deemed druggable if targeted by at least one existing drug. We extract drugs and their nominal targets from Drugbank. We retain negative training examples activated in at least one cell type-specific protein interaction network.

Task Description: Classification. Given the protein and cell-context, predict whether the protein is a therapeutic target.

Dataset Statistics: The final number of positive (negative) samples for RA and IBD were 152 (1,465) and 114 (1,377), respectively. In PINNACLE, this dataset was augmented to include 156 cell types.

Dataset Split: Cold Protein Split We split the dataset such that about 80% of the proteins are in the training set, about 10% of the proteins are in the validation set, and about 10% of the proteins are in the test set. The data splits are consistent for each cell type context to avoid data leakage.

```
from tdc.resource.dataloader import DataLoader
data = DataLoader(name="opentargets_dti")
df = data.get_data()
```

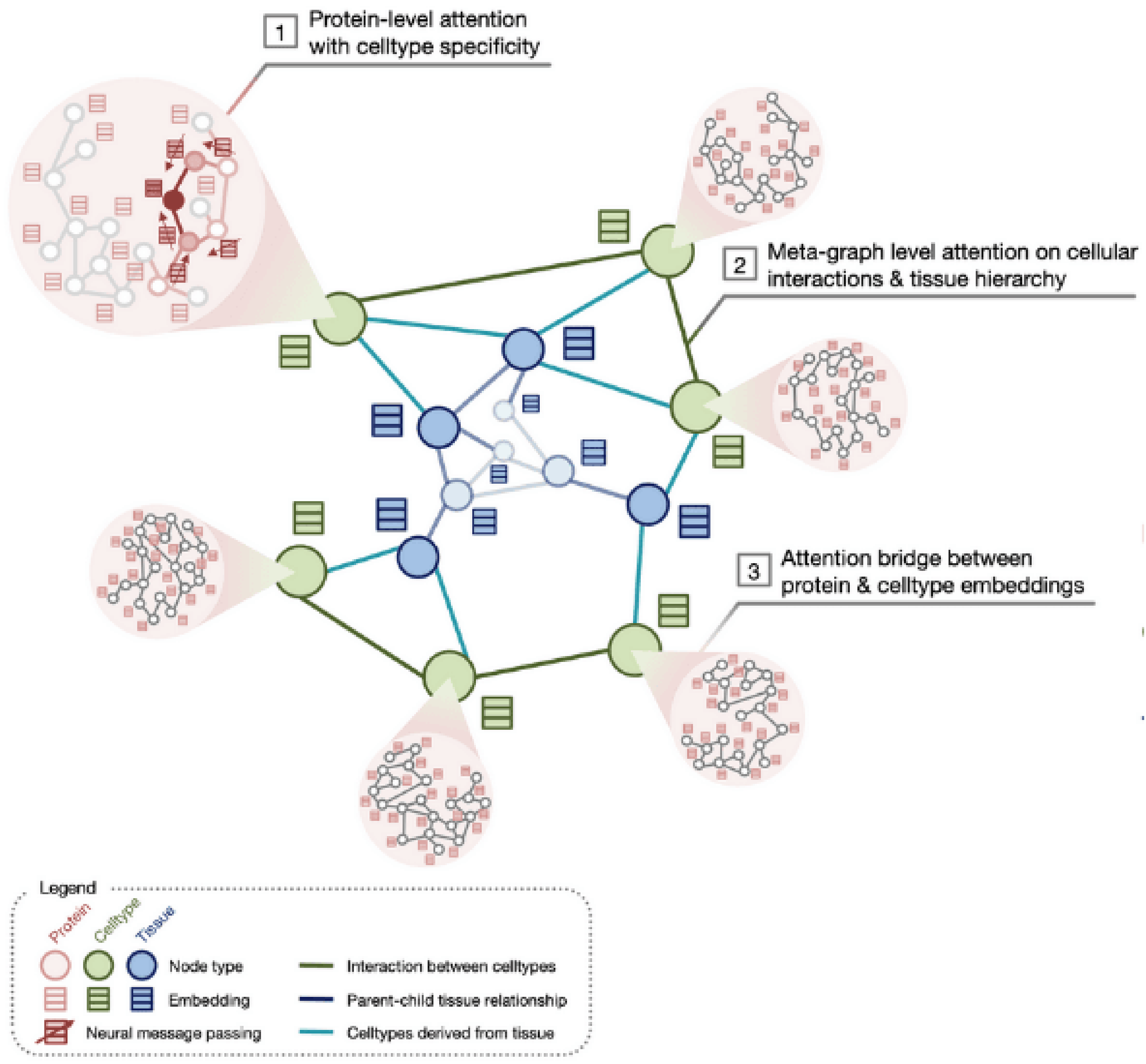
References:

[1] Li, Michelle, et al. “Contextualizing Protein Representations Using Deep Learning on Protein Networks and Single-Cell Data” bioRxiv (2023)

Dataset License: CC BY 4.0 US.

Benchmarked Models

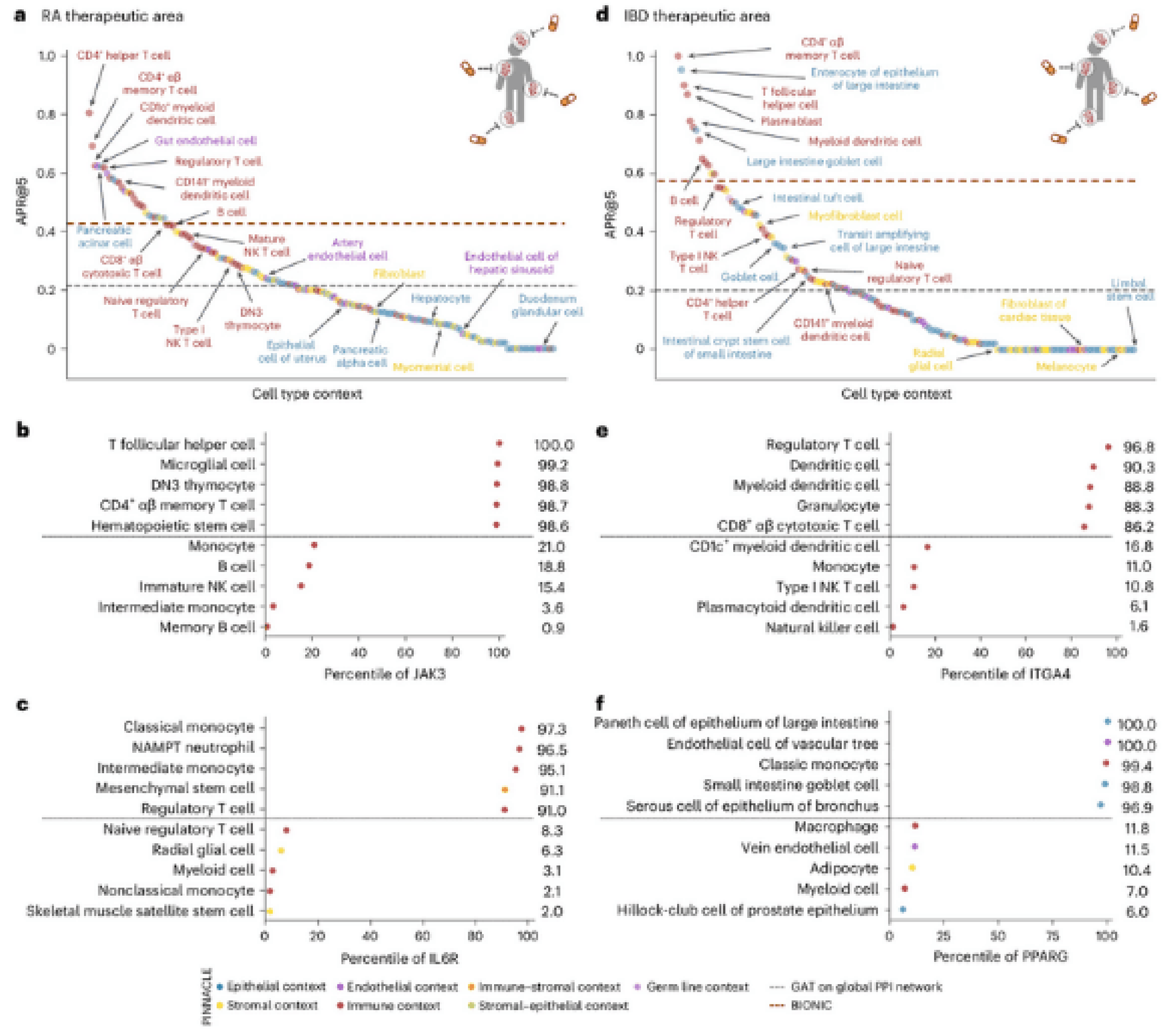
PINNACLE (Li et al.), GAT (baseline)



1 Protein-level attention with celltype specificity
2 Meta-graph level attention on cellular interactions & tissue hierarchy
3 Attention bridge between protein & celltype embeddings
Legend
Protein
Celltype
Tissue
Node type
Embedding
Neural message passing
Interaction between celltypes
Parent-child tissue relationship
Celltypes derived from tissue

Fig. 5: Performance of contextualized target prioritization for RA and IBD therapeutic areas.

From: Contextual AI models for single-cell protein biology



a,**d**, Model performance (measured by APR@5) for RA (**a**) and IBD (**d**) therapeutic areas, respectively. APR@K (or Average Precision and Recall at K) is a combination of Precision@K and Recall@K (refer to 'Metrics and statistical analyses' section in Methods for more details). Each dot is the performance (averaged across ten random seeds) of PINNACLE's protein representations from a specific cell type context. The gray and dark-orange lines are the performance of the GAT and BIONIC models, respectively. For each therapeutic area, 22 cell types are annotated and colored by their compartment category. Extended Data Fig. 8 contains model performance measured by APR@10, APR@15 and APR@20 for RA and IBD therapeutic areas. **b**,**c**,**e**,**f**, Selected proteins for RA and IBD therapeutic areas, where the horizontal solid line separates the top and bottom five cell types: two selected proteins, JAK3 (**b**) and IL6R (**c**), that are targeted by drugs that have completed phase IV of clinical trials for treating RA therapeutic area; two selected proteins, ITGA4 (**e**) and PPARG (**f**), that are targeted by drugs that have completed phase IV for treating IBD therapeutic area.

Benchmarked Results

TDC.scDTI Benchmark API

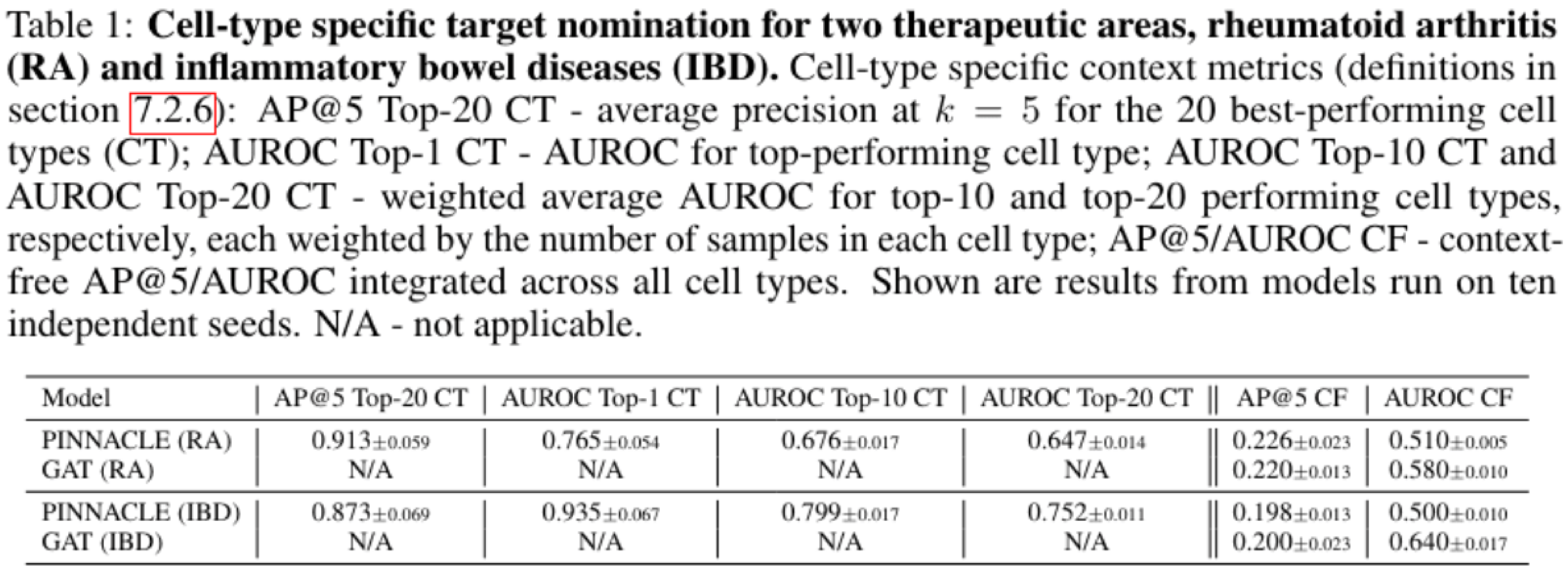
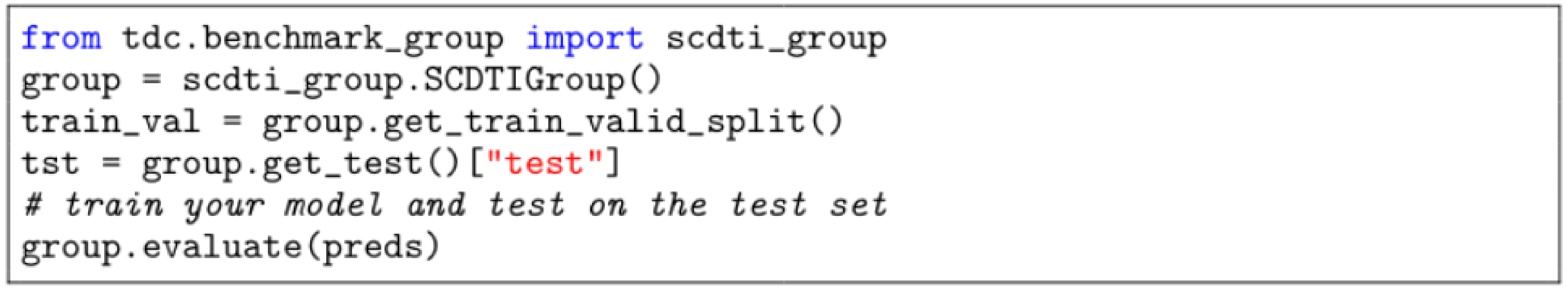


Table 1: **Cell-type specific target nomination for two therapeutic areas, rheumatoid arthritis (RA) and inflammatory bowel diseases (IBD).** Cell-type specific context metrics (definitions in section 7.2.6): AP@5 Top-20 CT - average precision at $k = 5$ for the 20 best-performing cell types (CT); AUROC Top-1 CT - AUROC for top-performing cell type; AUROC Top-10 CT and AUROC Top-20 CT - weighted average AUROC for top-10 and top-20 performing cell types, respectively, each weighted by the number of samples in each cell type; AP@5/AUROC CF - context-free AP@5/AUROC integrated across all cell types. Shown are results from models run on ten independent seeds. N/A - not applicable.

Model	AP@5 Top-20 CT	AUROC Top-1 CT	AUROC Top-10 CT	AUROC Top-20 CT	AP@5 CF	AUROC CF
PINNACLE (RA)	$0.913_{\pm 0.059}$	$0.765_{\pm 0.054}$	$0.676_{\pm 0.017}$	$0.647_{\pm 0.014}$	$0.226_{\pm 0.023}$	$0.510_{\pm 0.005}$
GAT (RA)	N/A	N/A	N/A	N/A	$0.220_{\pm 0.013}$	$0.580_{\pm 0.010}$
PINNACLE (IBD)	$0.873_{\pm 0.069}$	$0.935_{\pm 0.067}$	$0.799_{\pm 0.017}$	$0.752_{\pm 0.011}$	$0.198_{\pm 0.013}$	$0.500_{\pm 0.010}$
GAT (IBD)	N/A	N/A	N/A	N/A	$0.200_{\pm 0.023}$	$0.640_{\pm 0.017}$



```python
from tdc.benchmark_group import scdti_group
group = scdti_group.SCDTIGroup()
train_val = group.get_train_valid_split()
tst = group.get_test()["test"]
# train your model and test on the test set
group.evaluate(preds)
```

TDC.PerturbOutcome
Single-cell Perturbation Response Prediction

Genetic/Chemical

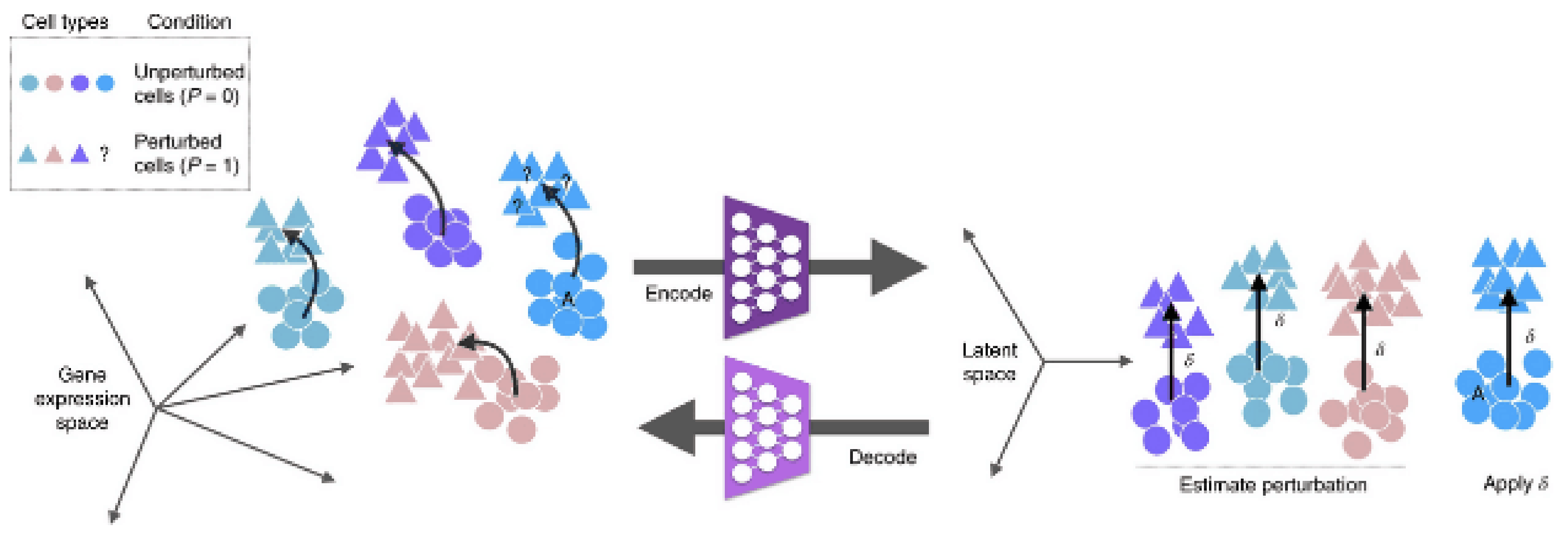
Motivation
Understanding and predicting transcriptional responses to genetic or chemical perturbations provides insights into cellular adaptation and response mechanisms.

Definition
To learn a regression model estimating the perturbation-response gene expression vector for a perturbation applied to a control cell.

Evaluation
We measure model generalization to unseen perturbations and perturbation combinations.

Fig. 1: scGen, a method to predict single-cell perturbation response.

From: scGen predicts single-cell perturbation responses



$$\hat{e_1} = f_\theta(p \in \mathbb{P}, e_0 \in \mathbb{E}_{\not p}, c \in \mathbb{C}).$$

TDC.PerturbOutcome task formulation

scPerturb

Dataset Description: The scPerturb dataset is a comprehensive collection of single-cell perturbation data, harmonized to facilitate the development and benchmarking of computational methods in systems biology. It includes various types of molecular readouts, such as transcriptomics, proteomics, and epigenomics. scPerturb is a harmonized dataset that compiles single-cell perturbation-response data. This dataset is designed to support the development and validation of computational tools by providing a consistent and comprehensive resource. The data includes responses to various genetic and chemical perturbations, which are crucial for understanding cellular mechanisms and developing therapeutic strategies. Data from different sources are uniformly pre-processed to ensure consistency. Rigorous quality control measures are applied to maintain high data quality. Features across different datasets are standardized for easy comparison and integration.

Task Description: Given cell-type-specific labels and a perturbation, predict the gene expression vector for that cell.

Dataset Statistics: 44 publicly available single-cell perturbation-response datasets. Most datasets have on average approximately 3000 genes measured per cell. 100,000+ perturbations.

Dataset Split: Random Split , Cold Cell Line Split , Cold Perturbation Split

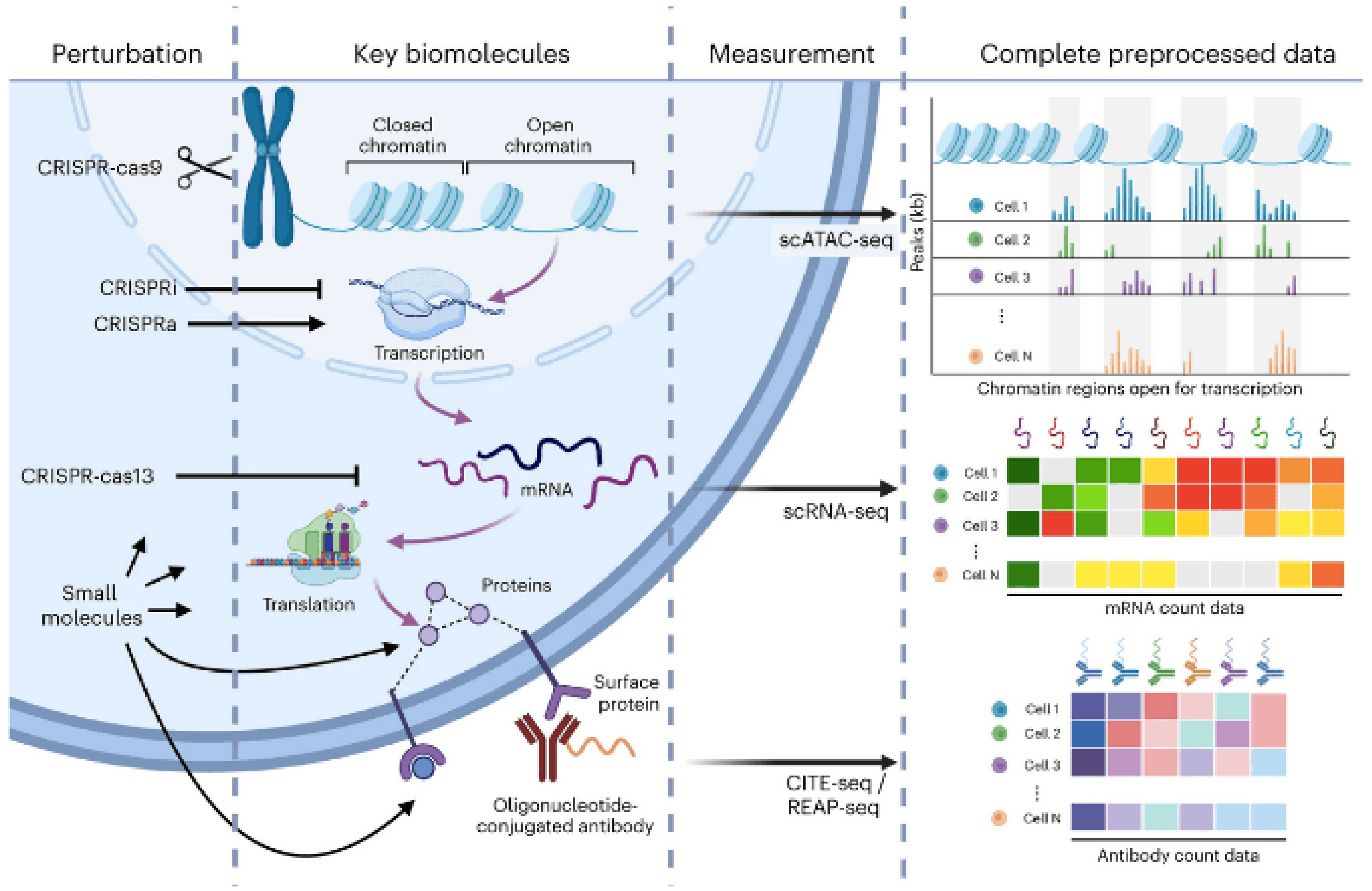
```
from tdc.multi_pred.perturboutcome import PerturbOutcome
data = PerturbOutcome(name = 'scperturb_drug_SrivatsanTrapnell2020_sciplex2')
split = data.get_split()
```

References:

Peidli, S., Green, T. D., Shen, C., Gross, T., Min, J. K., Garda, S., Yuan, B., Schumacher, L., Taylor-King, J., Marks, D., Luna, A., Blüthgen, N., & Sander, C. (2023). scPerturb: Harmonized Single-Cell Perturbation Data. https://doi.org/10.1101/2022.08.20.504663

Fig. 1: Perturbation–response profiling for single cells.

From: scPerturb: harmonized single-cell perturbation data

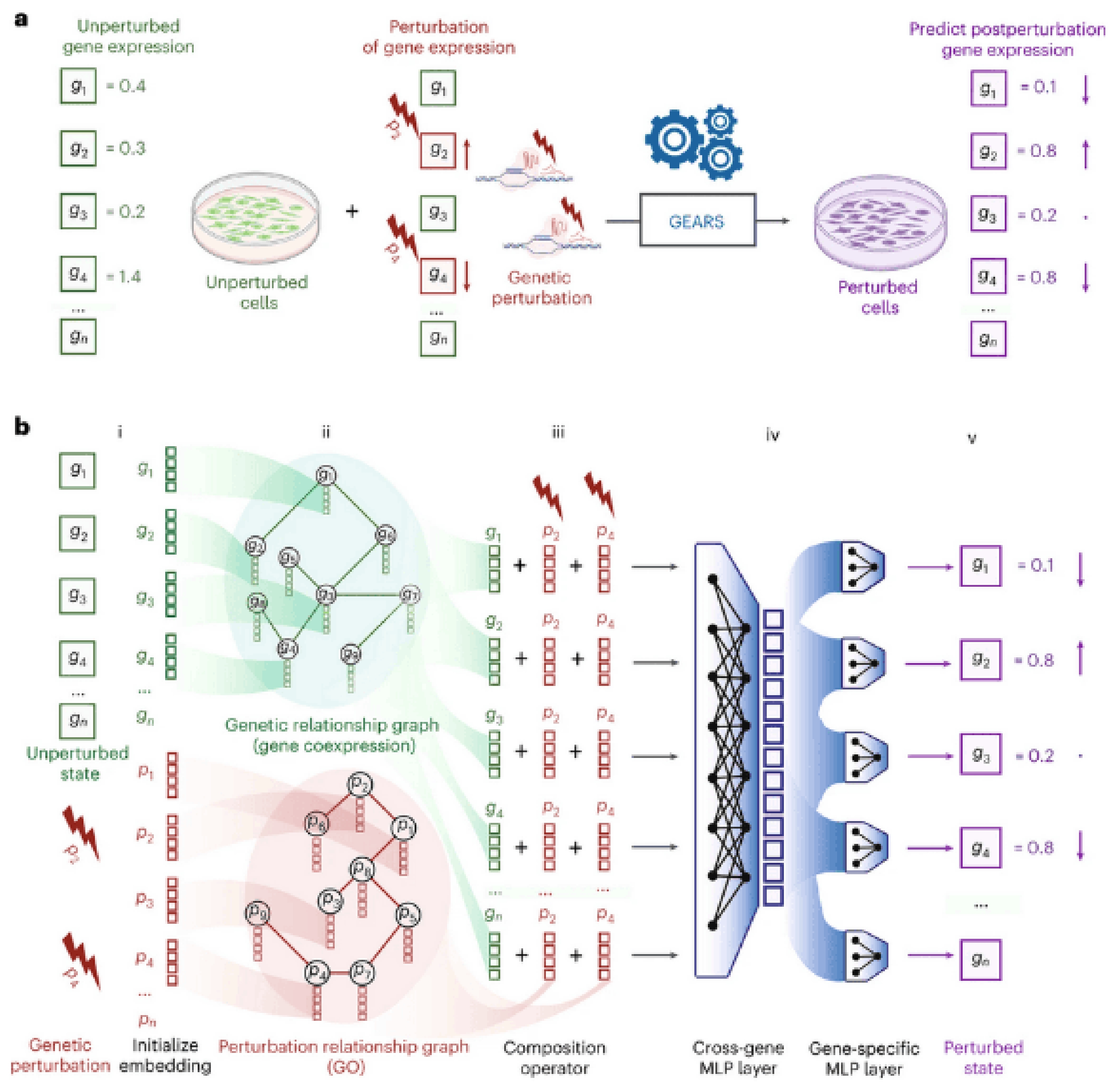


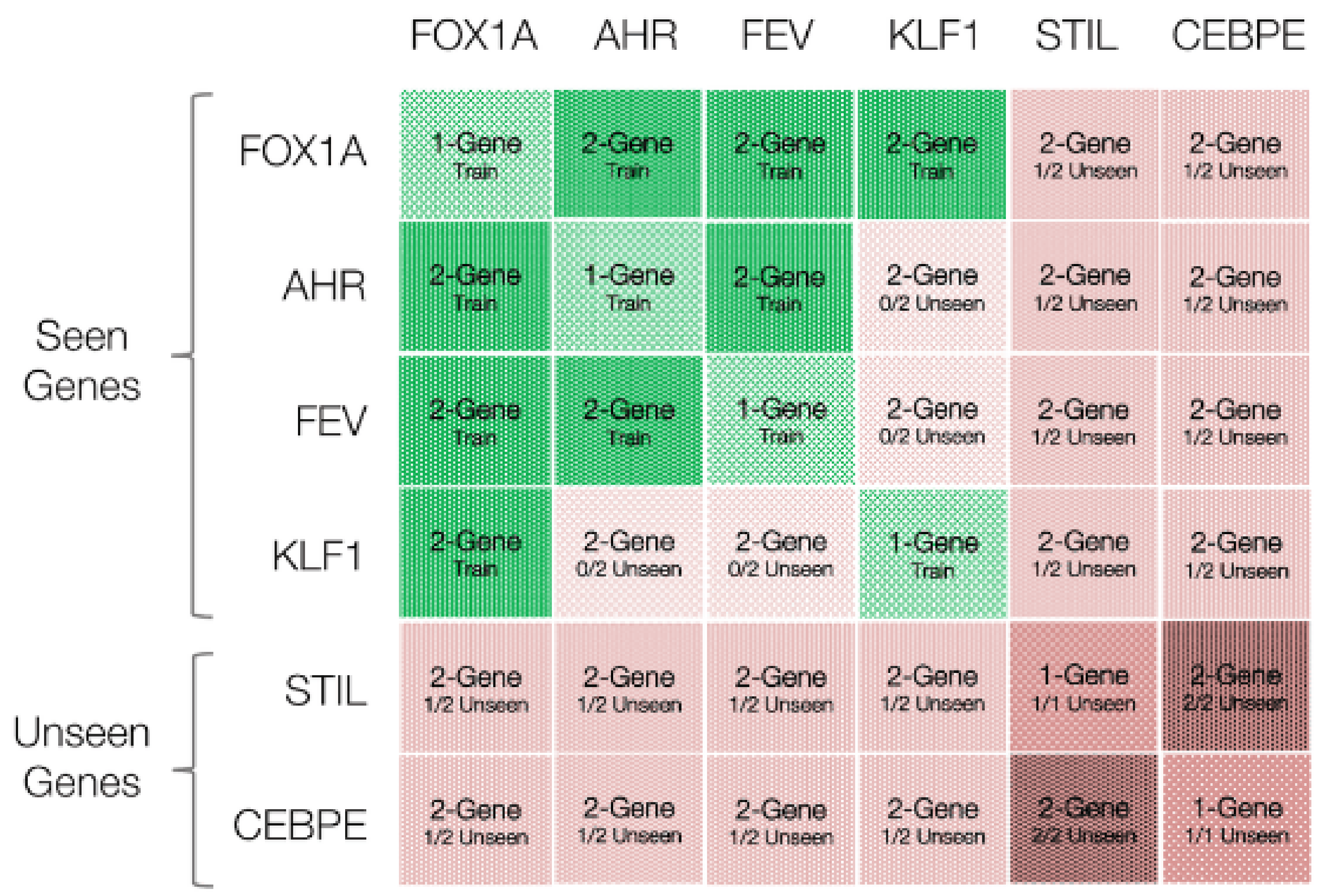
Benchmarked Models (Genetic)

GEARS (Roohani et al.), CPA (Lotfollahi et al.), no-perturb (baseline)

Fig. 1: GEARS combines prior knowledge with deep learning to predict postperturbation gene expression.

From: Predicting transcriptional outcomes of novel multigene perturbations with GEARS





		FOX1A	AHR	FEV	KLF1	STIL	CEBPE
Seen Genes	FOX1A	1-Gene Train	2-Gene Train	2-Gene Train	2-Gene Train	2-Gene 1/2 Unseen	2-Gene 1/2 Unseen
	AHR	2-Gene Train	1-Gene Train	2-Gene Train	2-Gene 0/2 Unseen	2-Gene 1/2 Unseen	2-Gene 1/2 Unseen
	FEV	2-Gene Train	2-Gene Train	1-Gene Train	2-Gene 0/2 Unseen	2-Gene 1/2 Unseen	2-Gene 1/2 Unseen
	KLF1	2-Gene Train	2-Gene 0/2 Unseen	2-Gene 0/2 Unseen	1-Gene Train	2-Gene 1/2 Unseen	2-Gene 1/2 Unseen
Unseen Genes	STIL	2-Gene 1/2 Unseen	2-Gene 1/2 Unseen	2-Gene 1/2 Unseen	2-Gene 1/2 Unseen	1-Gene 1/1 Unseen	2-Gene 2/2 Unseen
	CEBPE	2-Gene 1/2 Unseen	2-Gene 1/2 Unseen	2-Gene 1/2 Unseen	2-Gene 1/2 Unseen	2-Gene 2/2 Unseen	1-Gene 1/1 Unseen

Supplementary Fig. 4: Data split matrix A sample data split illustration to describe how the different perturbation categories were defined on the basis of training set composition. Genes that were *seen* experimentally perturbed in the training data and *unseen* genes are marked on the vertical axis. 1-gene perturbations of seen genes were included in the training set (**1-Gene, Train**). 1-gene perturbations of unseen genes were included in the test set (**1-Gene, 1/1 Unseen**). 2-gene combinatorial perturbations with one gene unseen were included in the test set (**2-Gene, 1/2 Unseen**) as were those with two genes unseen (**2-Gene, 0/2 Unseen**). 2-gene combinatorial perturbations with both genes seen were randomly split between train (**2-Gene, Train**) and test (**2-Gene, 2/2 Unseen**).

Benchmarked Results

TDC.GenePerturb Benchmark API

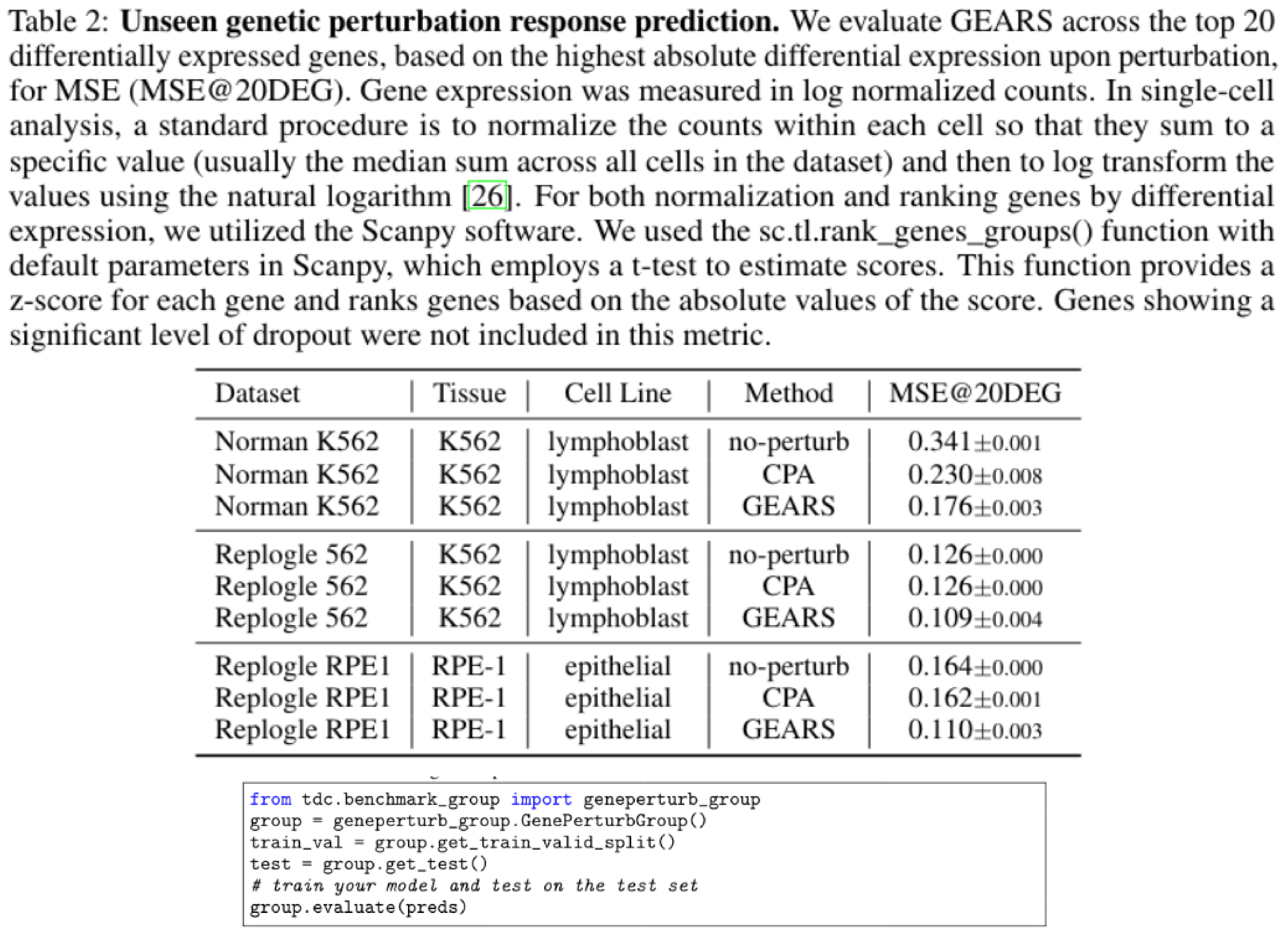


Table 2: **Unseen genetic perturbation response prediction.** We evaluate GEARS across the top 20 differentially expressed genes, based on the highest absolute differential expression upon perturbation, for MSE (MSE@20DEG). Gene expression was measured in log normalized counts. In single-cell analysis, a standard procedure is to normalize the counts within each cell so that they sum to a specific value (usually the median sum across all cells in the dataset) and then to log transform the values using the natural logarithm [26]. For both normalization and ranking genes by differential expression, we utilized the Scanpy software. We used the sc.tl.rank_genes_groups() function with default parameters in Scanpy, which employs a t-test to estimate scores. This function provides a z-score for each gene and ranks genes based on the absolute values of the score. Genes showing a significant level of dropout were not included in this metric.

Dataset	Tissue	Cell Line	Method	MSE@20DEG
Norman K562	K562	lymphoblast	no-perturb	0.341±0.001
Norman K562	K562	lymphoblast	CPA	0.230±0.008
Norman K562	K562	lymphoblast	GEARS	0.176±0.003
Replogle 562	K562	lymphoblast	no-perturb	0.126±0.000
Replogle 562	K562	lymphoblast	CPA	0.126±0.000
Replogle 562	K562	lymphoblast	GEARS	0.109±0.004
Replogle RPE1	RPE-1	epithelial	no-perturb	0.164±0.000
Replogle RPE1	RPE-1	epithelial	CPA	0.162±0.001
Replogle RPE1	RPE-1	epithelial	GEARS	0.110±0.003

```python
from tdc.benchmark_group import geneperturb_group
group = geneperturb_group.GenePerturbGroup()
train_val = group.get_train_valid_split()
test = group.get_test()
# train your model and test on the test set
group.evaluate(preds)
```

Benchmarked Models (Chemical)

ChemCPA (Hetzel et al.), Biolord (Piran et al.), scGen (Lotfollahi et al.), no-perturb-information (baseline)

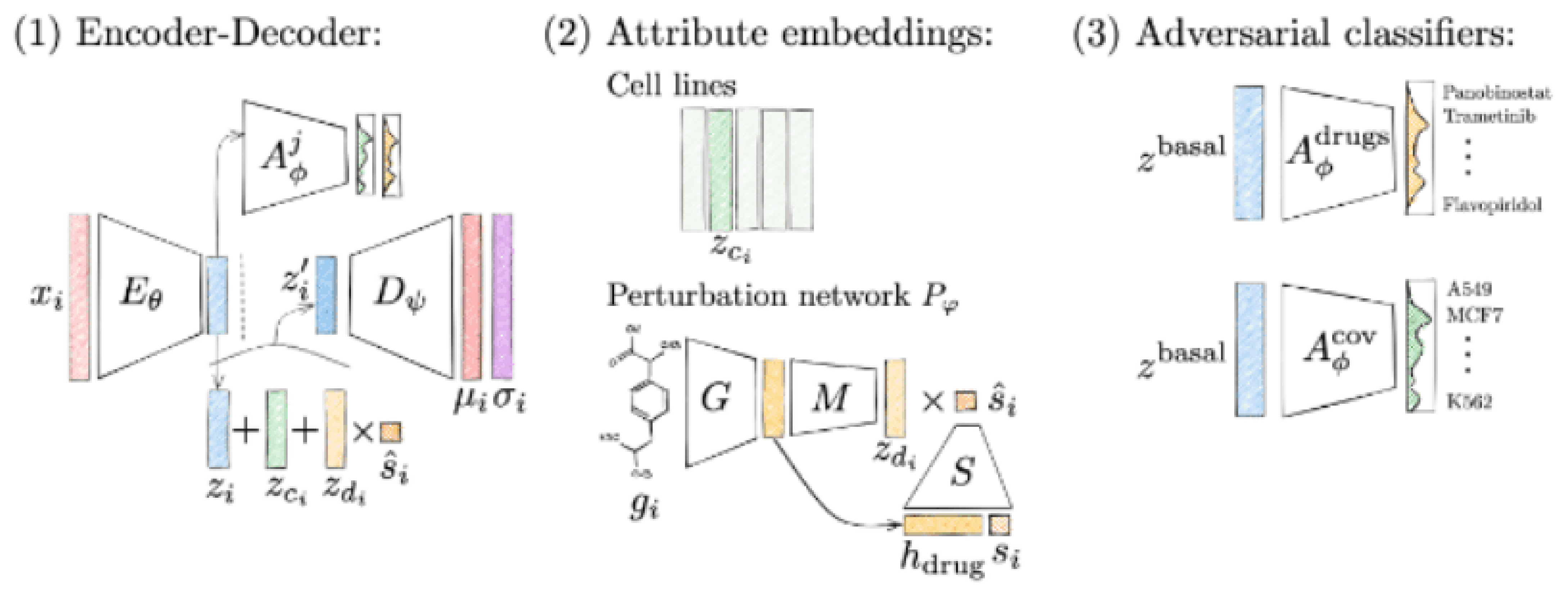
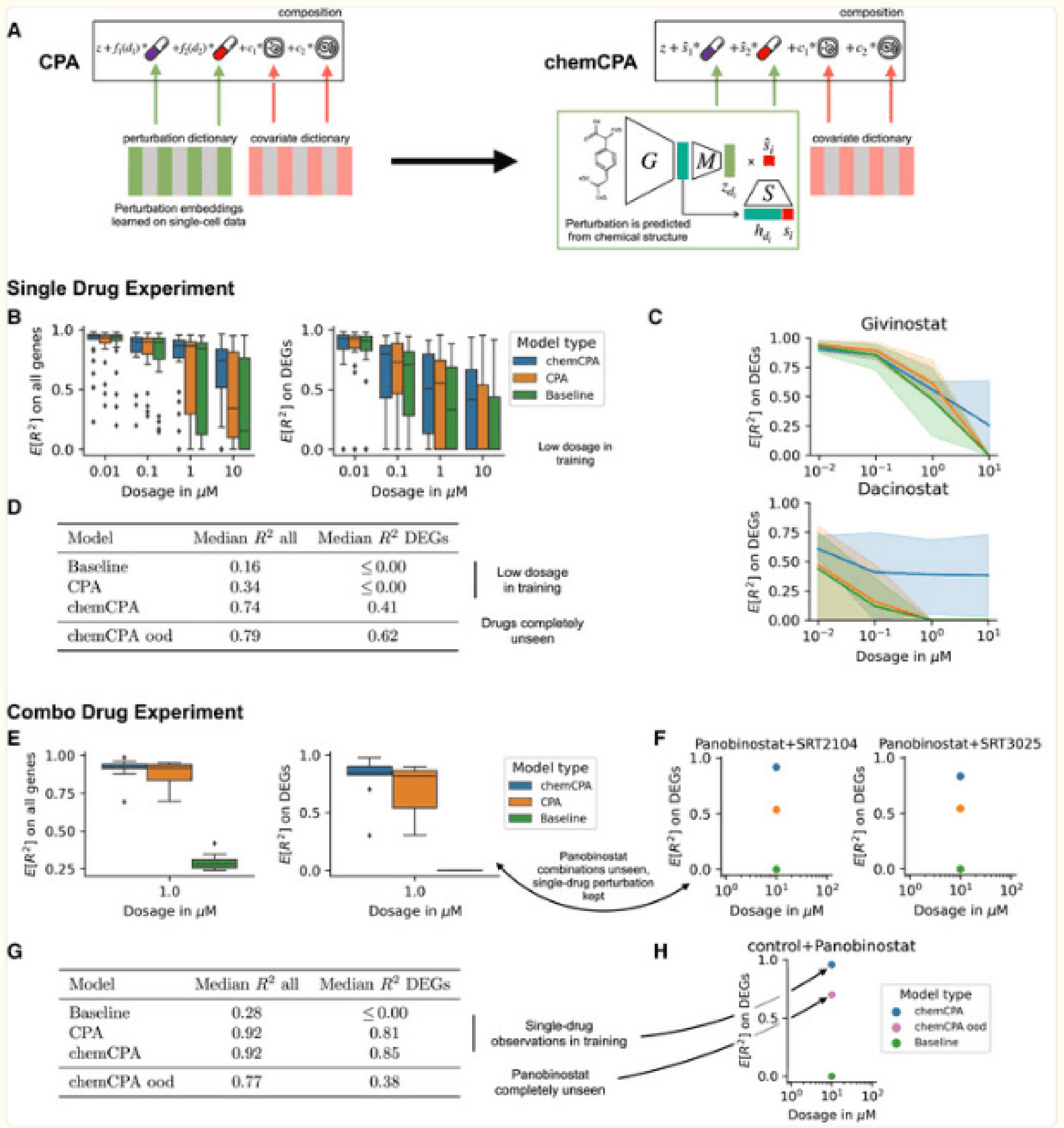


Figure 4. CPA extensibility enables predicting the response to unseen drugs.



D

Model	Median R^2 all	Median R^2 DEGs
Baseline	0.16	≤ 0.00
CPA	0.34	≤ 0.00
chemCPA	0.74	0.41
chemCPA ood	0.79	0.62

Low dosage in training

Drugs completely unseen

G

Model	Median R^2 all	Median R^2 DEGs
Baseline	0.28	≤ 0.00
CPA	0.92	0.81
chemCPA	0.92	0.85
chemCPA ood	0.77	0.38

Single-drug observations in training

Panobinostat completely unseen

Benchmarked Results

TDC.ChemPerturb Benchmark API

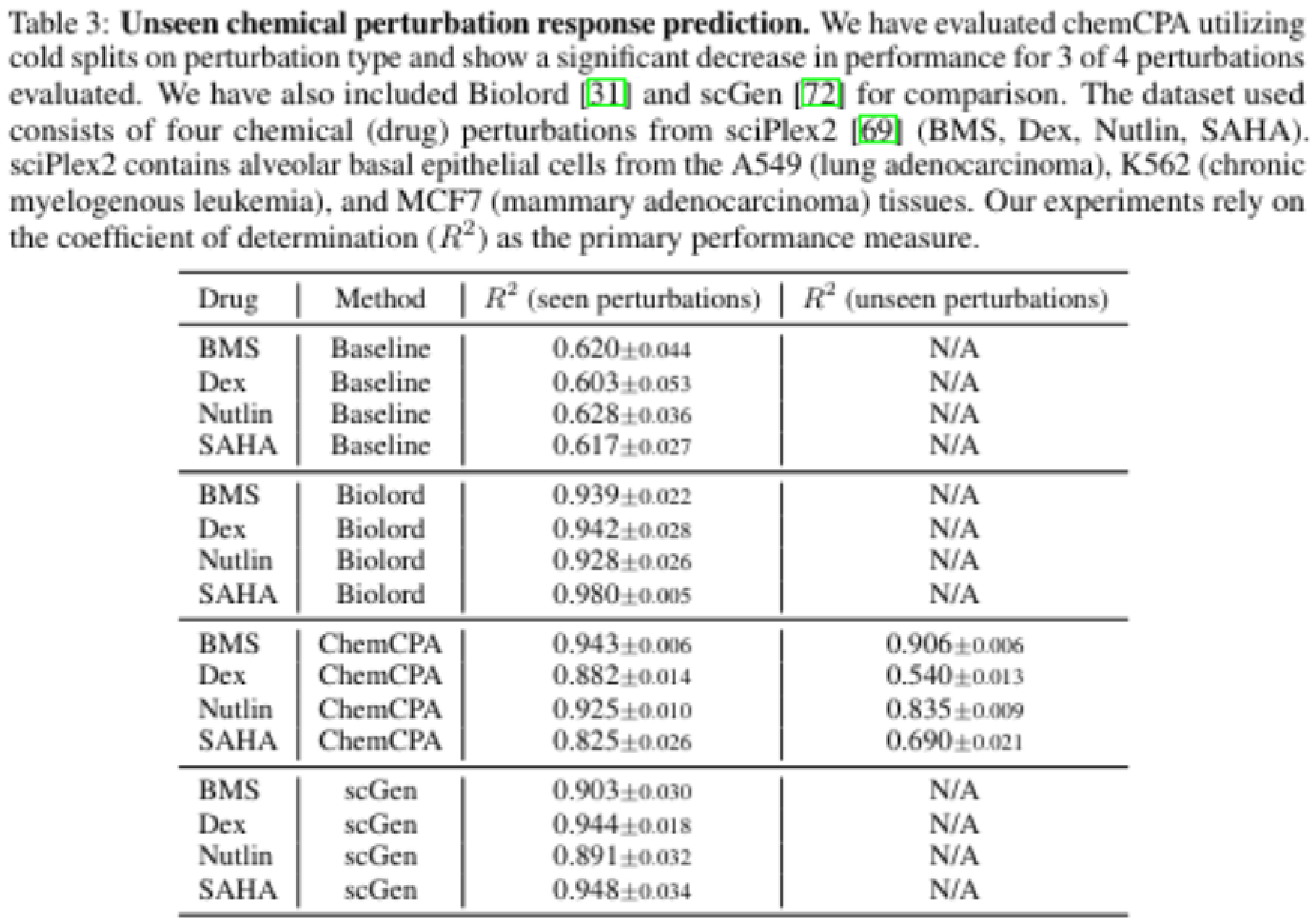
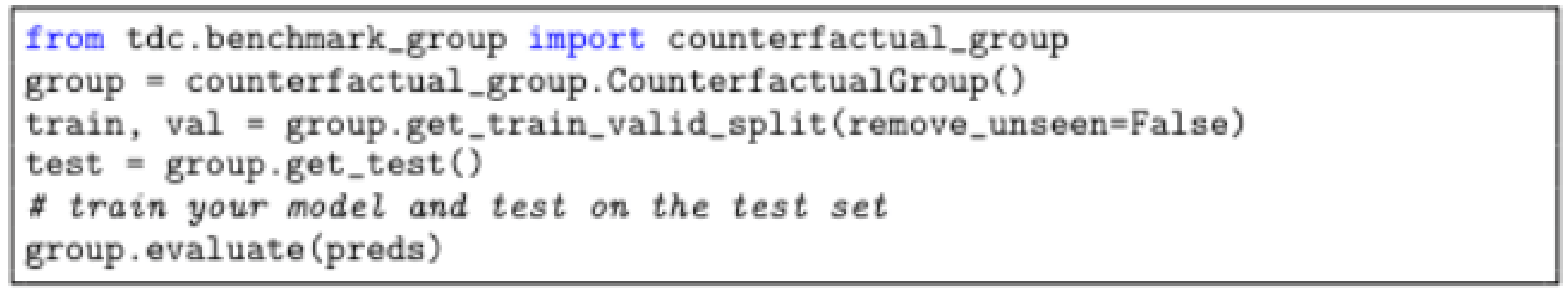


Table 3: **Unseen chemical perturbation response prediction.** We have evaluated chemCPA utilizing cold splits on perturbation type and show a significant decrease in performance for 3 of 4 perturbations evaluated. We have also included Biolord [31] and scGen [72] for comparison. The dataset used consists of four chemical (drug) perturbations from sciPlex2 [69] (BMS, Dex, Nutlin, SAHA). sciPlex2 contains alveolar basal epithelial cells from the A549 (lung adenocarcinoma), K562 (chronic myelogenous leukemia), and MCF7 (mammary adenocarcinoma) tissues. Our experiments rely on the coefficient of determination (R^2) as the primary performance measure.

Drug	Method	R^2 (seen perturbations)	R^2 (unseen perturbations)
BMS	Baseline	$0.620_{\pm 0.044}$	N/A
Dex	Baseline	$0.603_{\pm 0.053}$	N/A
Nutlin	Baseline	$0.628_{\pm 0.036}$	N/A
SAHA	Baseline	$0.617_{\pm 0.027}$	N/A
BMS	Biolord	$0.939_{\pm 0.022}$	N/A
Dex	Biolord	$0.942_{\pm 0.028}$	N/A
Nutlin	Biolord	$0.928_{\pm 0.026}$	N/A
SAHA	Biolord	$0.980_{\pm 0.005}$	N/A
BMS	ChemCPA	$0.943_{\pm 0.006}$	$0.906_{\pm 0.006}$
Dex	ChemCPA	$0.882_{\pm 0.014}$	$0.540_{\pm 0.013}$
Nutlin	ChemCPA	$0.925_{\pm 0.010}$	$0.835_{\pm 0.009}$
SAHA	ChemCPA	$0.825_{\pm 0.026}$	$0.690_{\pm 0.021}$
BMS	scGen	$0.903_{\pm 0.030}$	N/A
Dex	scGen	$0.944_{\pm 0.018}$	N/A
Nutlin	scGen	$0.891_{\pm 0.032}$	N/A
SAHA	scGen	$0.948_{\pm 0.034}$	N/A



```python
from tdc.benchmark_group import counterfactual_group
group = counterfactual_group.CounterfactualGroup()
train, val = group.get_train_valid_split(remove_unseen=False)
test = group.get_test()
# train your model and test on the test set
group.evaluate(preds)
```

TDC.ProteinPeptide
Protein-Peptide Binding Interaction Prediction

Motivation

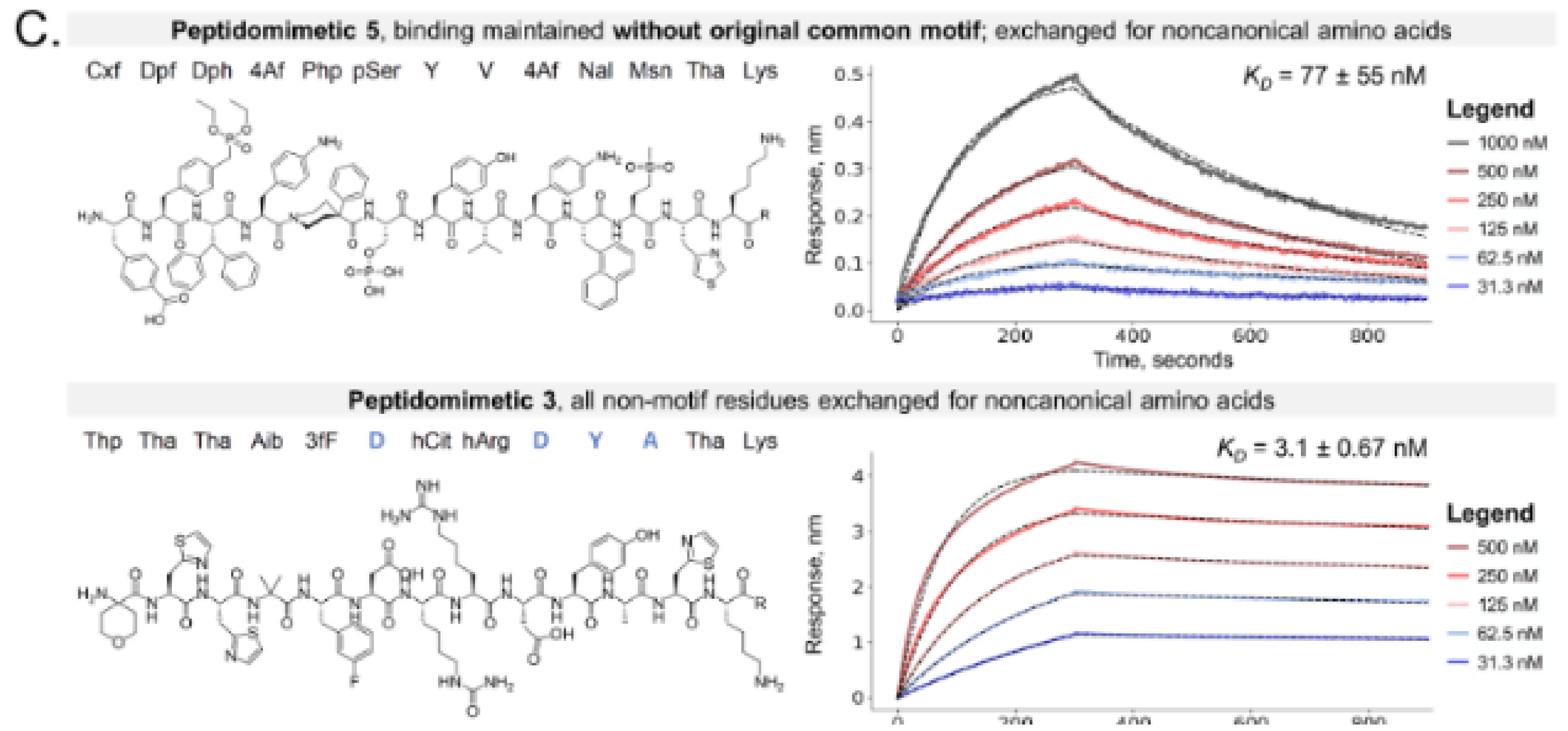
Protein-peptide binding affinity prediction and protein-protein binding affinity prediction involve similar underlying biological interactions, but they differ significantly in their complexity and the methods used to predict them (Abdin, Osana et al., 2022).

Definition

To learn a binary classification model of a protein-peptide interaction meeting specific biomarkers.

Evaluation

We measure classification performance metrics to unseen proteins and peptides.



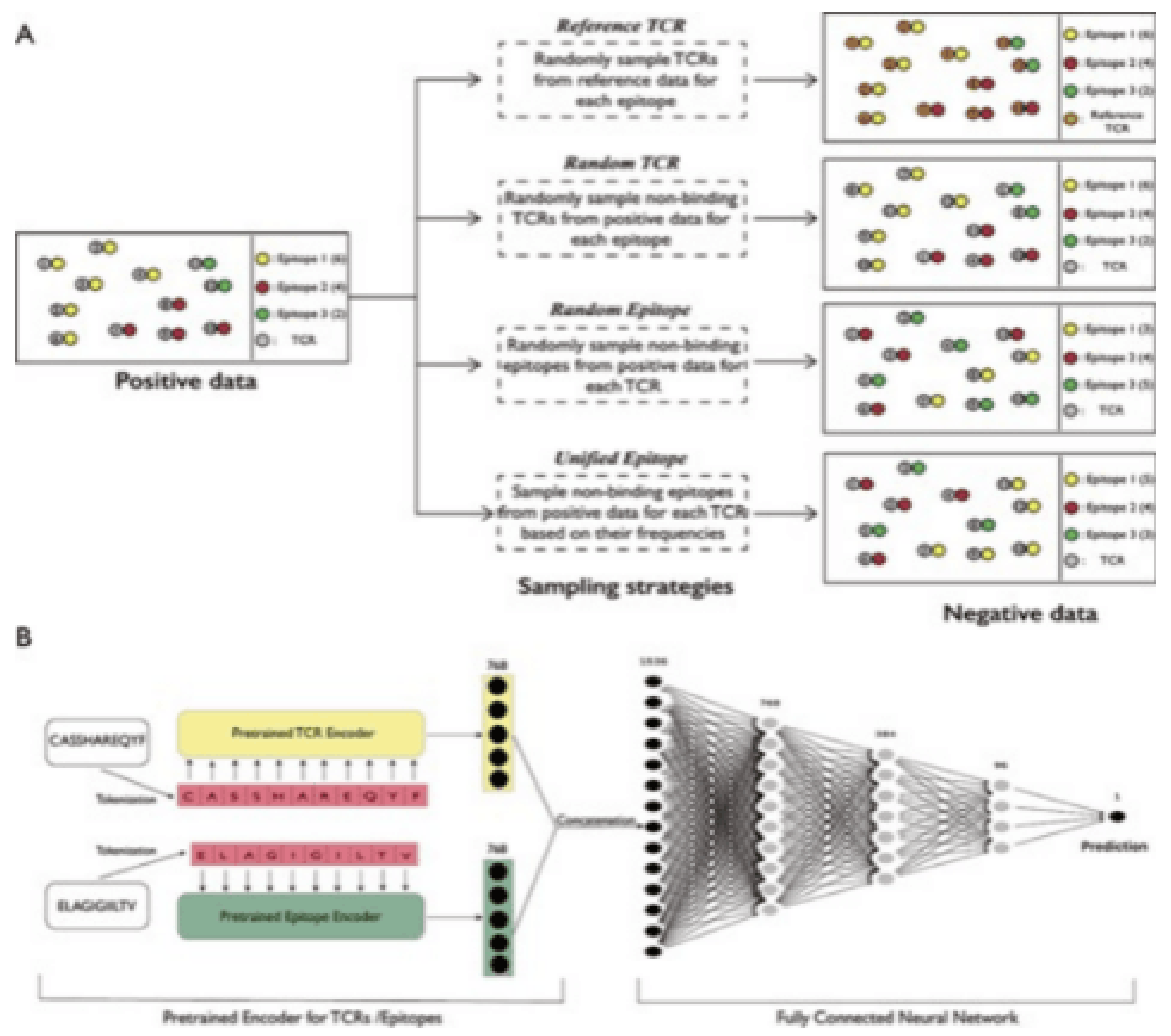
$$\hat{y} = f_\theta(p \in \mathbb{P}, s \in \mathbb{S}, a \in \mathbb{A}, i \in \mathbb{I}, c \in \mathbb{C}).$$

TDC.ProteinPeptide task formulation

TDC.TCREpitope TCR-Epitope Binding Interaction Prediction

Challenges

The critical challenge in TCR-Epitope (Peptide-MHC Complex) interaction prediction lies in creating a model that can effectively **generalize to unseen TCRs and epitopes**. While TCR-H and TEINet have shown improved performance on prediction for known epitopes, by incorporating advanced features like attention mechanisms and transfer learning, the performance considerably drops for unseen epitopes. Another challenge in TCR-Epitope interaction prediction lies in the **choice of heuristic for generating negative samples**, with non-binders often underrepresented or biased in curated datasets, leading to inaccurate predictions when generalized.


A
Positive data
Reference TCR
Randomly sample TCRs from reference data for each epitope
Random TCR
Randomly sample non-binding TCRs from positive data for each epitope
Random Epitope
Randomly sample non-binding epitopes from positive data for each TCR
Unified Epitope
Sample non-binding epitopes from positive data for each TCR based on their frequencies
Sampling strategies
Negative data
B
CASSHAREQYF
ELAGIGILTV
Tokenization
Pretrained TCR Encoder
Pretrained Epitope Encoder
Concatenation
Prediction
Pretrained Encoder for TCRs /Epitopes
Fully Connected Neural Network

Benchmarking datasets use three types of heuristics for generating negative samples: random shuffling of epitope and TCR sequences (RN), experimental negatives (NA), and pairing external TCR sequences with epitope sequences (ET).

Benchmarked Results

TDC.TCREpitope Benchmark API

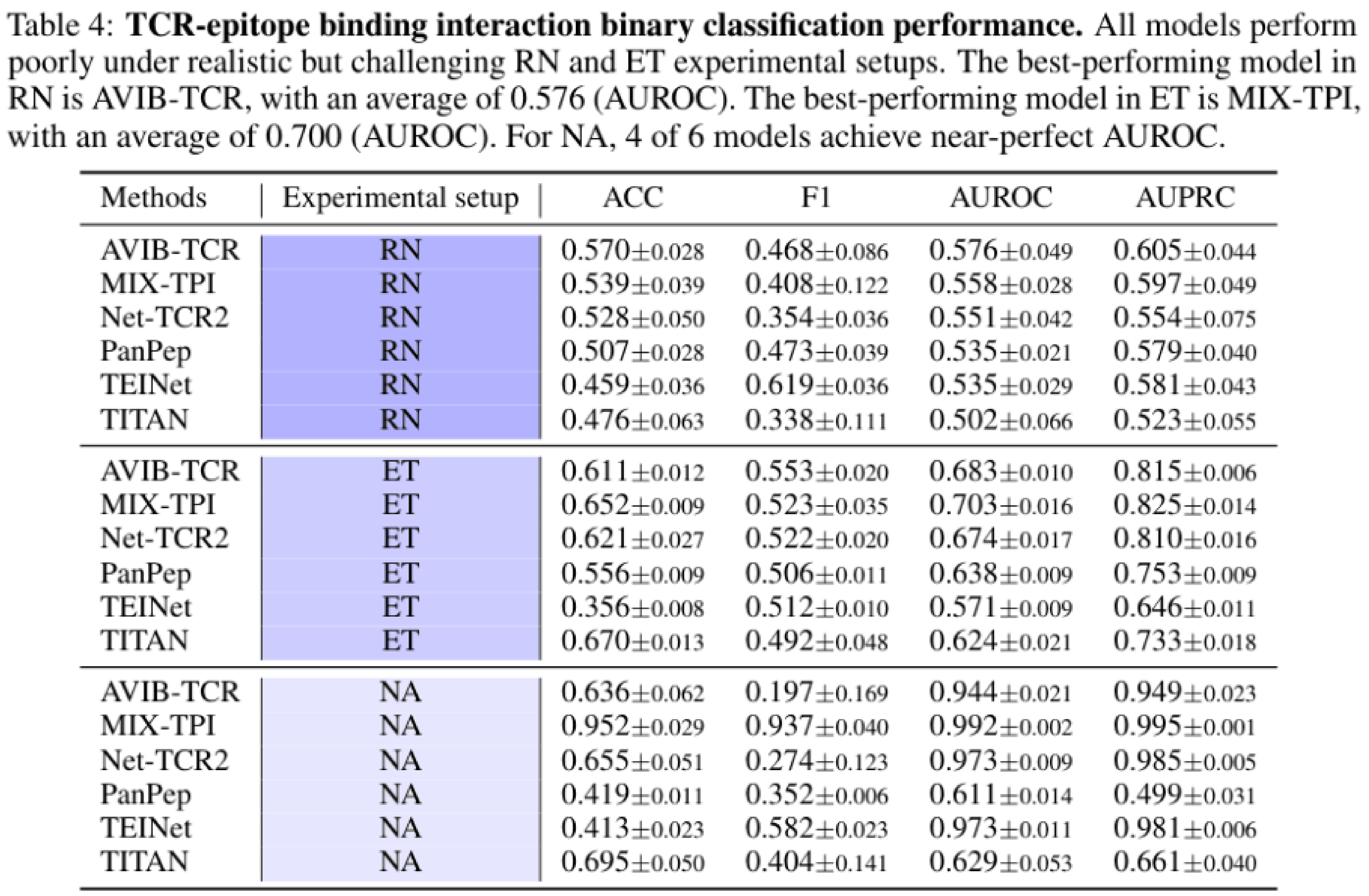
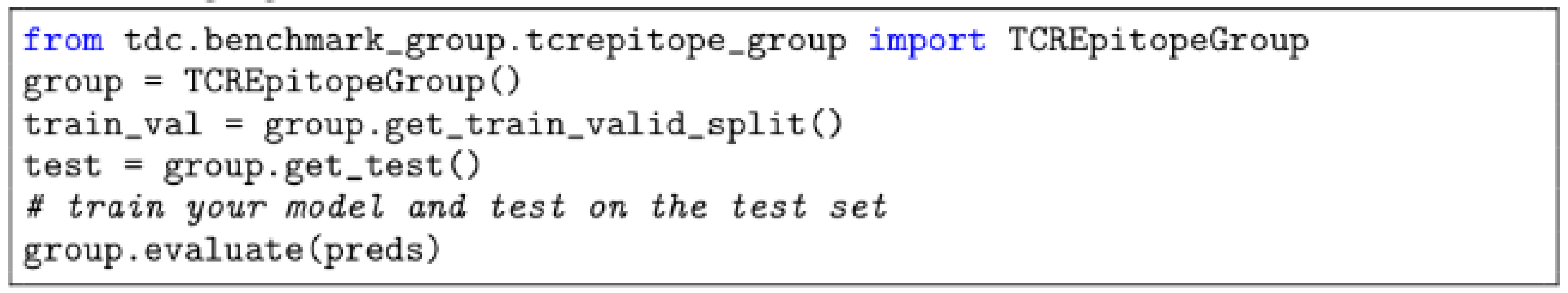


Table 4: **TCR-epitope binding interaction binary classification performance.** All models perform poorly under realistic but challenging RN and ET experimental setups. The best-performing model in RN is AVIB-TCR, with an average of 0.576 (AUROC). The best-performing model in ET is MIX-TPI, with an average of 0.700 (AUROC). For NA, 4 of 6 models achieve near-perfect AUROC.

Methods	Experimental setup	ACC	F1	AUROC	AUPRC
AVIB-TCR	RN	0.570±0.028	0.468±0.086	0.576±0.049	0.605±0.044
MIX-TPI	RN	0.539±0.039	0.408±0.122	0.558±0.028	0.597±0.049
Net-TCR2	RN	0.528±0.050	0.354±0.036	0.551±0.042	0.554±0.075
PanPep	RN	0.507±0.028	0.473±0.039	0.535±0.021	0.579±0.040
TEINet	RN	0.459±0.036	0.619±0.036	0.535±0.029	0.581±0.043
TITAN	RN	0.476±0.063	0.338±0.111	0.502±0.066	0.523±0.055
AVIB-TCR	ET	0.611±0.012	0.553±0.020	0.683±0.010	0.815±0.006
MIX-TPI	ET	0.652±0.009	0.523±0.035	0.703±0.016	0.825±0.014
Net-TCR2	ET	0.621±0.027	0.522±0.020	0.674±0.017	0.810±0.016
PanPep	ET	0.556±0.009	0.506±0.011	0.638±0.009	0.753±0.009
TEINet	ET	0.356±0.008	0.512±0.010	0.571±0.009	0.646±0.011
TITAN	ET	0.670±0.013	0.492±0.048	0.624±0.021	0.733±0.018
AVIB-TCR	NA	0.636±0.062	0.197±0.169	0.944±0.021	0.949±0.023
MIX-TPI	NA	0.952±0.029	0.937±0.040	0.992±0.002	0.995±0.001
Net-TCR2	NA	0.655±0.051	0.274±0.123	0.973±0.009	0.985±0.005
PanPep	NA	0.419±0.011	0.352±0.006	0.611±0.014	0.499±0.031
TEINet	NA	0.413±0.023	0.582±0.023	0.973±0.011	0.981±0.006
TITAN	NA	0.695±0.050	0.404±0.141	0.629±0.053	0.661±0.040



```
from tdc.benchmark_group.tcrepitope_group import TCREpitopeGroup
group = TCREpitopeGroup()
train_val = group.get_train_valid_split()
test = group.get_test()
# train your model and test on the test set
group.evaluate(preds)
```

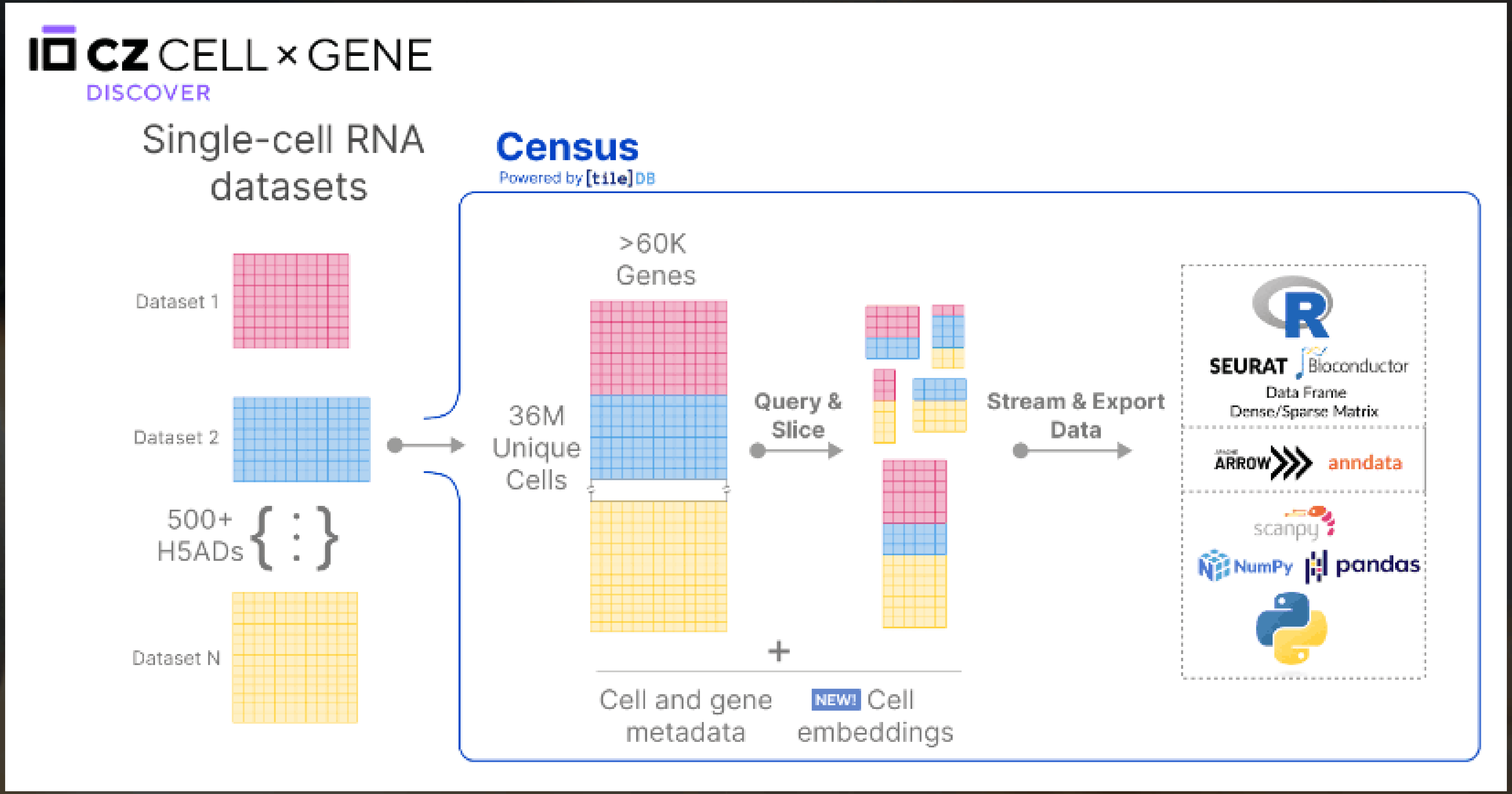
Multimodal Single-Cell Retrieval API

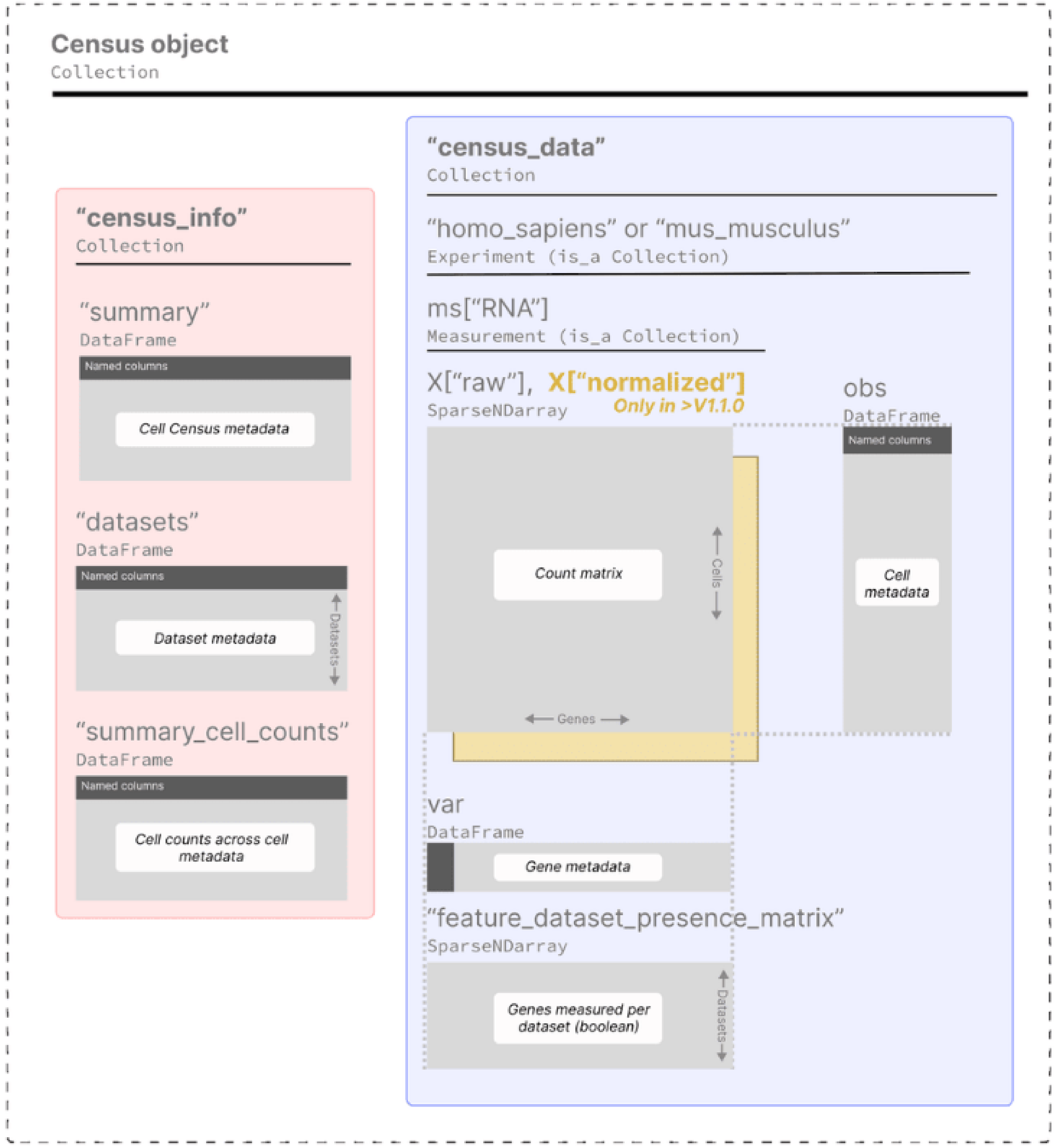
We leverage the CZ CellXGene Census to develop a TDC-2 Resource Model for constructing large-scale single-cell datasets that maps gene expression profiles of individual cells across tissues, healthy and disease state

TDC-2 <> CZ CELLxGENE Discover

Powered by TileDB

TDC-2 leverages the SOMA (Stack of Matrices, Annotated) API, adopts **TileDB-SOMA for modeling sets of 2D annotated matrices with measurements of features across observations**, and enables **memory-efficient <u>querying of multiple distinct single-cell modalities</u> (i.e., scRNA-seq, snRNA-seq), across healthy and diseased samples, with tabular annotations of cells, samples, and patients** the samples come from.





SOMA API <> CellXGene Census Discover

SOMA: Data model and API spec for annotated matrices

SOMAExperiment: Multimodal Container

obs: SOMADataFrame - observation annotations (i.e., Cell)

ms: a collection of 1 or more SOMAMeasurement (i.e., scRNA-seq)

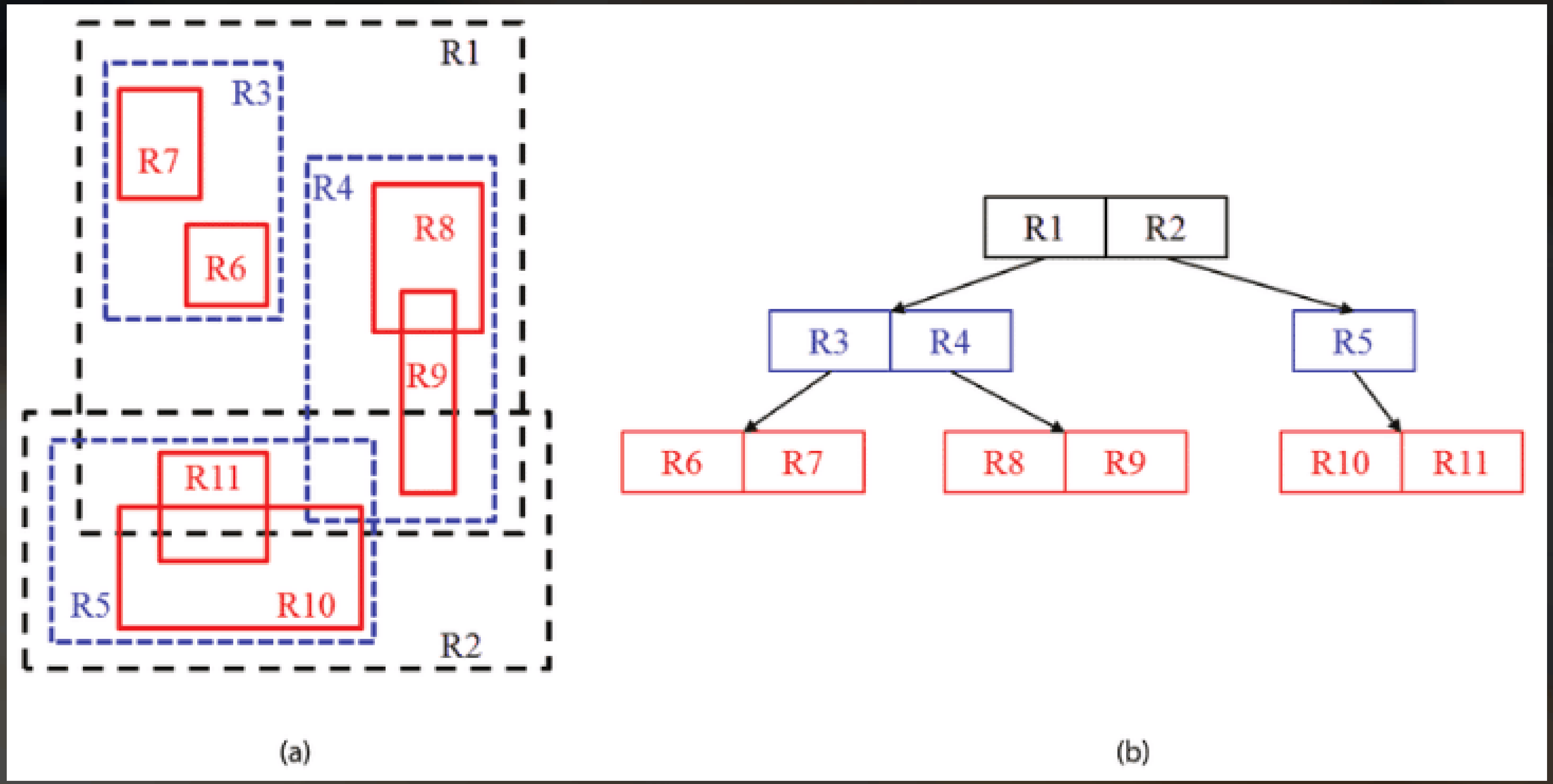
SOMAMeasurement:

- var: SOMADataFrame w/ variable annotations (i.e., Gene)
- X: Collection of SOMANdArray
- varm & obsm: derived results (i.e., embeddings)
- varp & obsp: pairwise features (i.e., "feature_dataset_presence_matrix)

TileDB

Why use TileDB-SOMA?

The key idea of TileDB is that it **<u>stores array elements into collections called fragments, which can be either dense or sparse</u>**. Each of these fragments stores data in data tiles, which are limited by number of elements for sparse arrays. TileDB implements an **R-tree as an index to implement sparse array slicing**. On array write, TileDB builds an R-tree index on the non-empty cells of the sparse array



R-tree indexing and bounding boxes

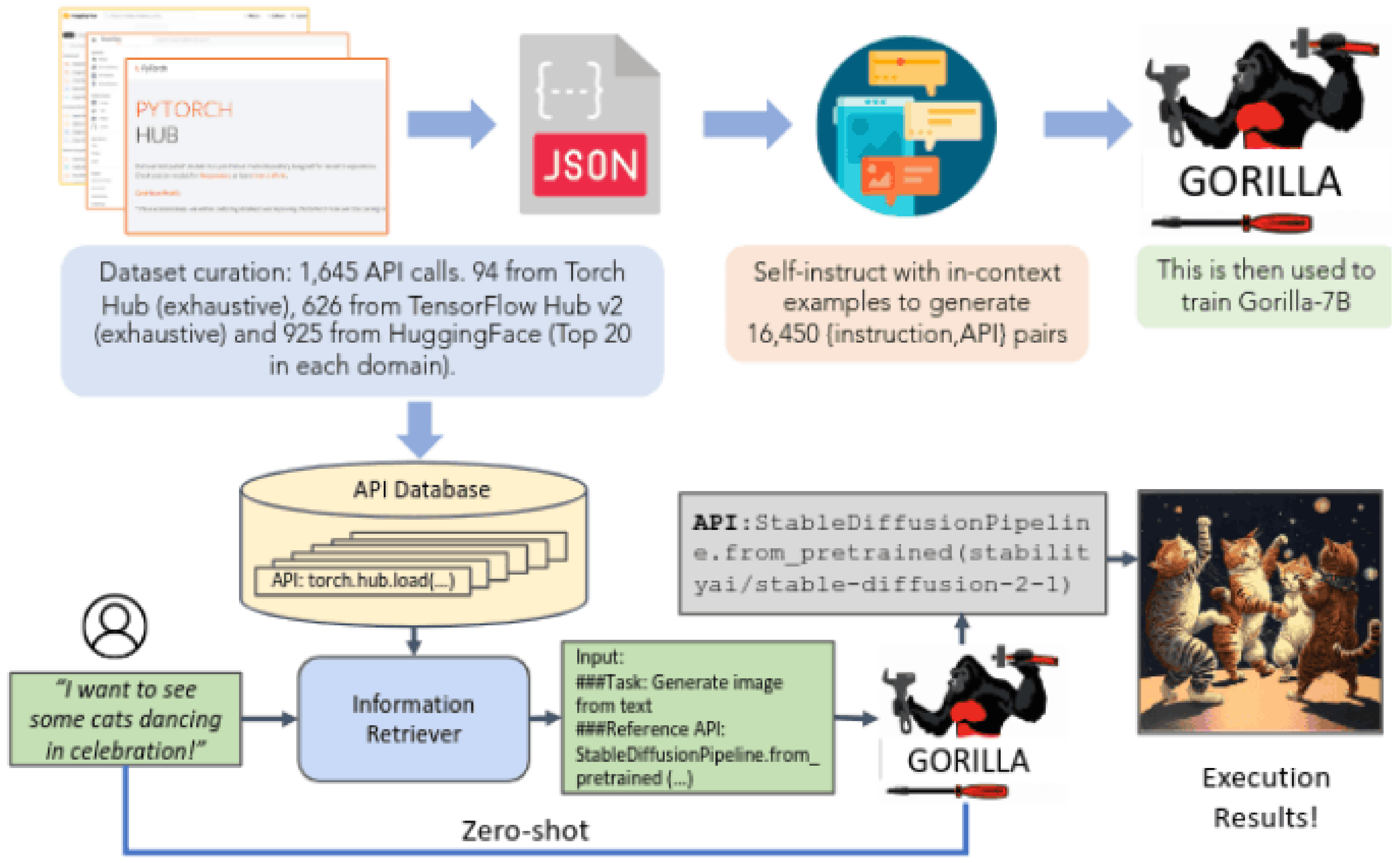
TDC-2 CellXGene API

Memory-efficient querying via TileDB-SOMA

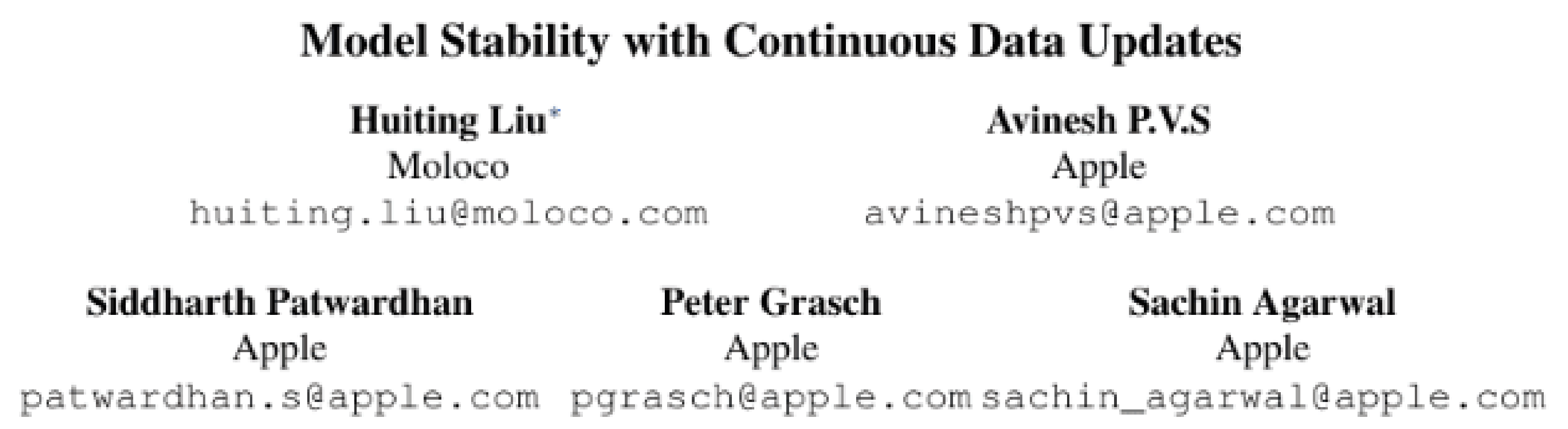
```
from tdc.multi_pred.single_cell import CellXGene
from pandas import DataFrame
dataloader = CellXGene(name="Tabula Sapiens - All Cells")
gen = dataloader.get_data(
value_filter="tissue == 'brain' and sex == 'male'"
)
df = next(gen)
```

API-first-dataset Architecture

API-integrated multimodal data-views integrating heterogeneous datasources via the Model-View-Controller design pattern


PYTORCH HUB
JSON
GORILLA
Dataset curation: 1,645 API calls. 94 from Torch Hub (exhaustive), 626 from TensorFlow Hub v2 (exhaustive) and 925 from HuggingFace (Top 20 in each domain).
Self-instruct with in-context examples to generate 16,450 {instruction,API} pairs
This is then used to train Gorilla-7B
API Database
API: torch.hub.load(...)
"I want to see some cats dancing in celebration!"
Information Retriever
Input:
###Task: Generate image from text
###Reference API: StableDiffusionPipeline.from_pretrained (...)
GORILLA
API:StableDiffusionPipeline.from_pretrained(stabilityai/stable-diffusion-2-1)
Execution Results!
Zero-shot

Patil et al. shows that integrating retrieval APIs with LLMs mitigates the issue of hallucination, commonly encountered when prompting LLMs directly. They also discuss the challenges of supporting a web scale collection of millions of changing APIs.



Model Stability with Continuous Data Updates

Huiting Liu[*]
Moloco
huiting.liu@moloco.com

Avinesh P.V.S
Apple
avineshpvs@apple.com

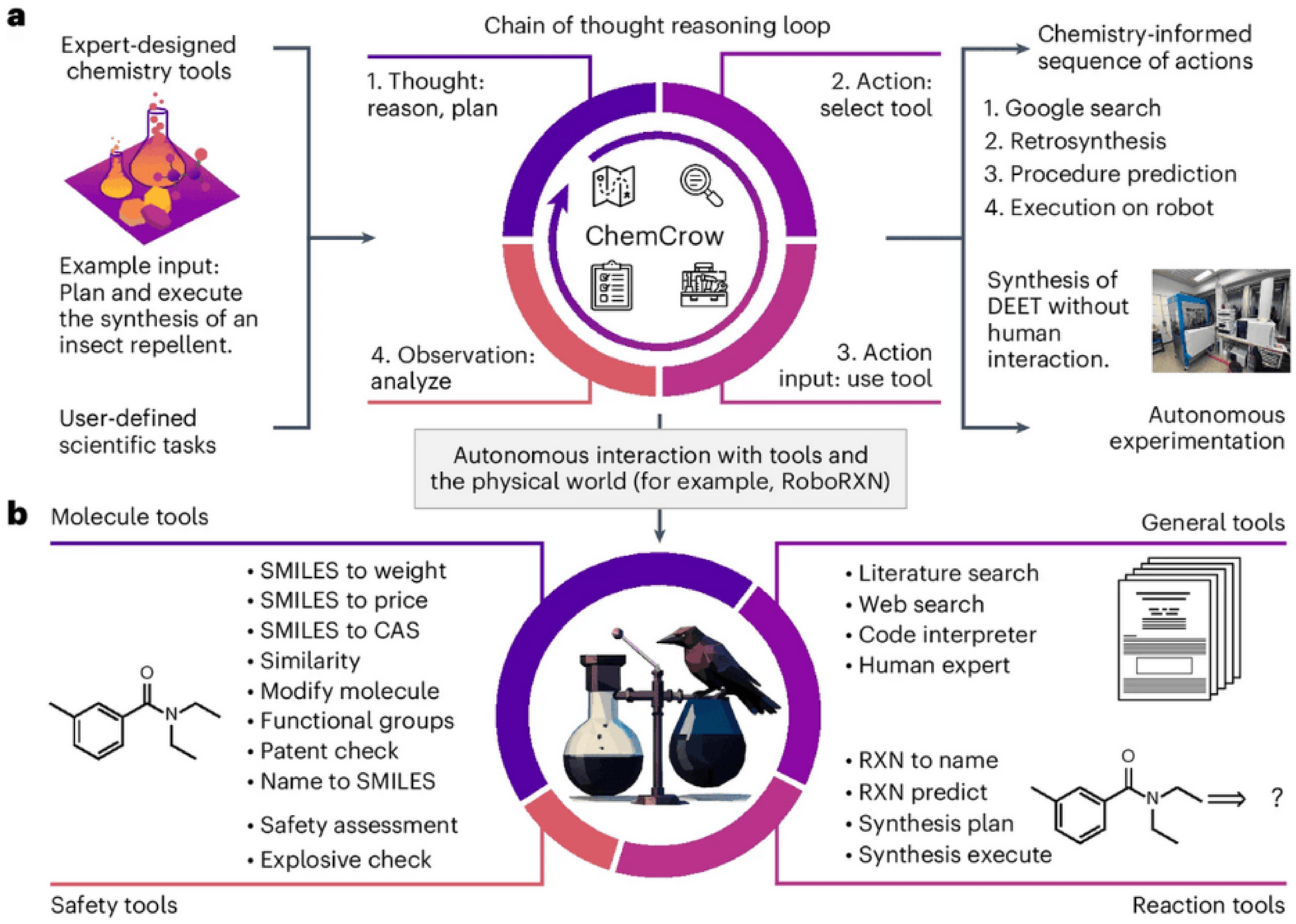
Siddharth Patwardhan
Apple
patwardhan.s@apple.com

Peter Grasch
Apple
pgrasch@apple.com

Sachin Agarwal
Apple
sachin_agarwal@apple.com

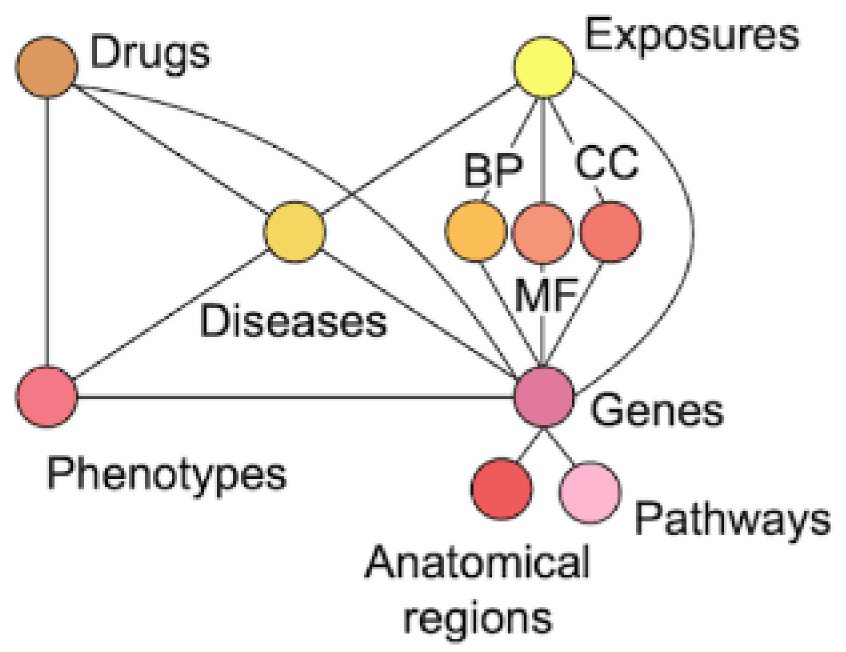
Fig. 1: Overview and toolset.

From: Augmenting large language models with chemistry tools



TDC-2 Resource Model

Enhancing contextualization across ML tasks



multimodal healthy-diseased retrieval API, vast corpus of nearly 50 million cells across 700 datasets

Harvard Dataverse + external APIs (ncbi, mygene, chembl, etc.

Querying API for precision medicine-oriented knowledge graph (PrimeKG)

PrimeKG

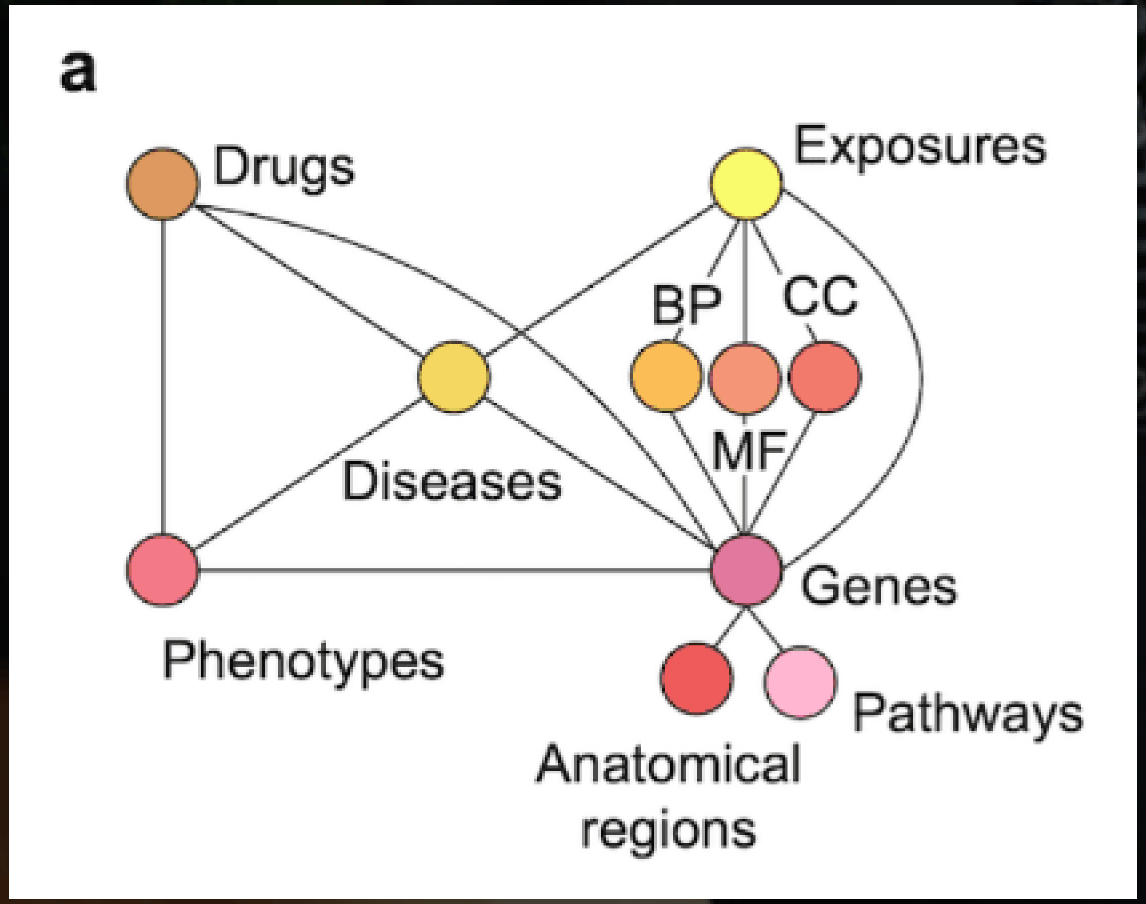
Precision Medicine Oriented Knowledge Graph

PrimeKG integrates **20 high-quality resources to describe 17,080 diseases with 4,050,249 relationships representing ten major biological scales**, including disease-associated protein perturbations, biological processes and pathways, anatomical and phenotypic scale, and the entire range of approved and experimental drugs with their therapeutic action.

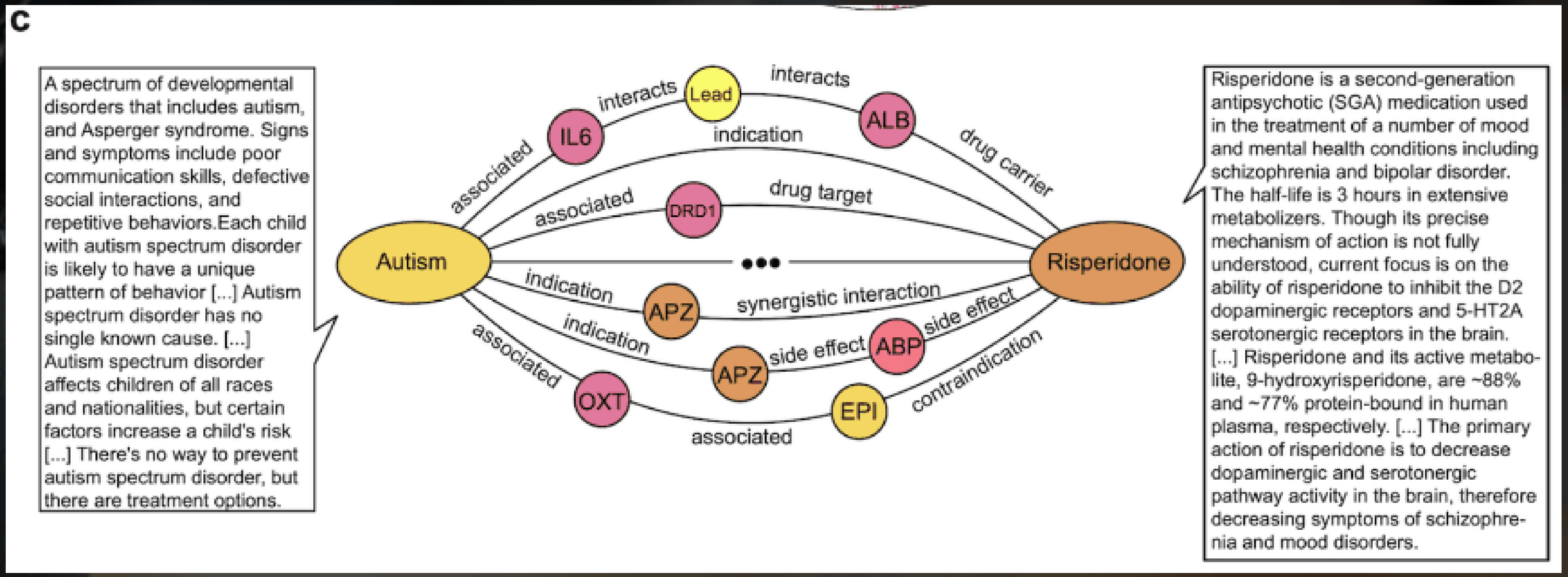
PrimeKG **supports drug-disease prediction by including an abundance of ’indications’, ’contradictions’ and ’off-label use’ edges**, which are usually missing in other knowledge graphs. We accompany PrimeKG's graph structure with text descriptions of clinical guidelines for drugs and diseases to enable multi-modal analyses.

PrimeKG

Nodes, relations, example paths



Abbreviations - MF: molecular function, BP: biological process, CC: cellular component, APZ: Apiprazole, EPI: epilepsy, ABP: abdominal pain, + / - associations: positive and negative associations.



A spectrum of developmental disorders that includes autism, and Asperger syndrome. Signs and symptoms include poor communication skills, defective social interactions, and repetitive behaviors.Each child with autism spectrum disorder is likely to have a unique pattern of behavior [...] Autism spectrum disorder has no single known cause. [...] Autism spectrum disorder affects children of all races and nationalities, but certain factors increase a child's risk [...] There's no way to prevent autism spectrum disorder, but there are treatment options.

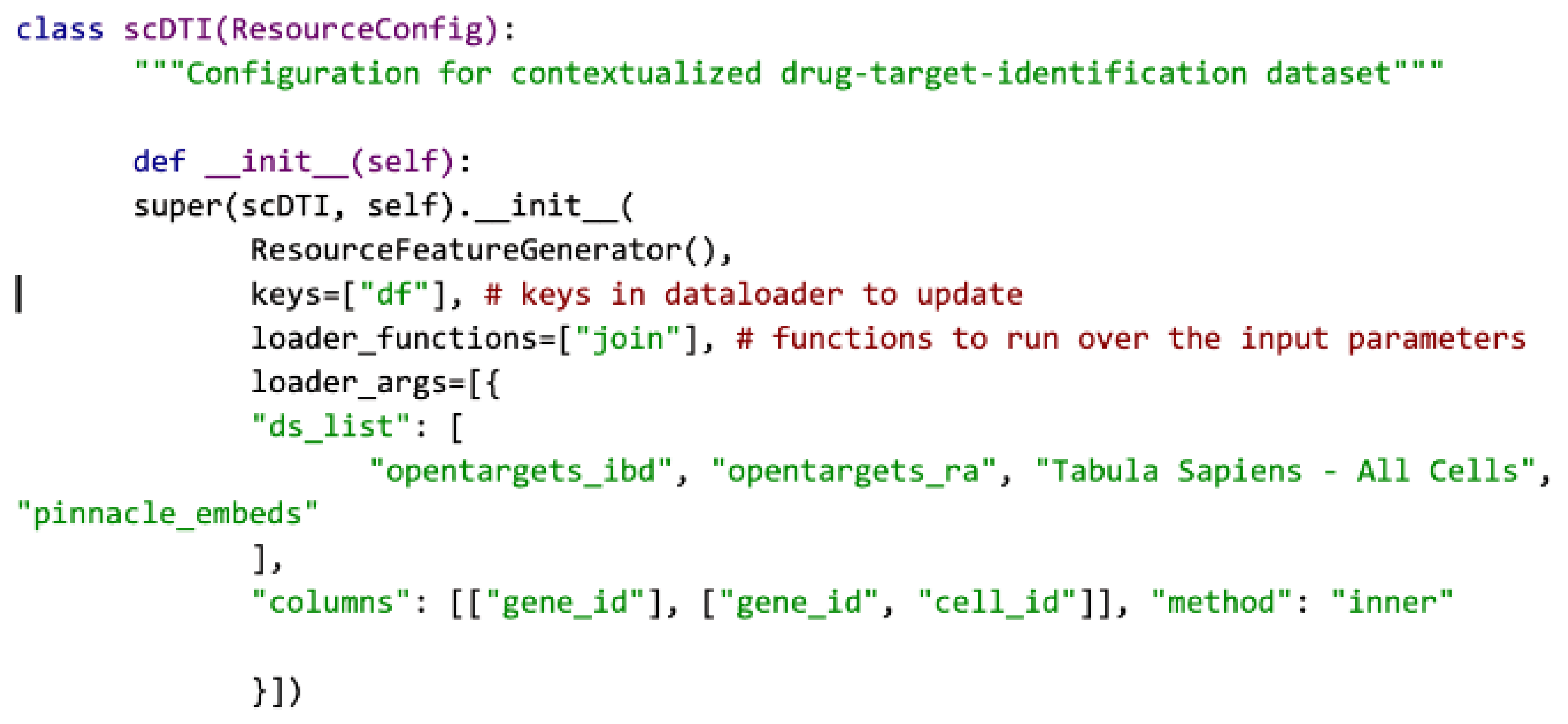
Risperidone is a second-generation antipsychotic (SGA) medication used in the treatment of a number of mood and mental health conditions including schizophrenia and bipolar disorder. The half-life is 3 hours in extensive metabolizers. Though its precise mechanism of action is not fully understood, current focus is on the ability of risperidone to inhibit the D2 dopaminergic receptors and 5-HT2A serotonergic receptors in the brain. [...] Risperidone and its active metabolite, 9-hydroxyrisperidone, are ~88% and ~77% protein-bound in human plasma, respectively. [...] The primary action of risperidone is to decrease dopaminergic and serotonergic pathway activity in the brain, therefore decreasing symptoms of schizophrenia and mood disorders.

example of paths in PrimeKG between the disease node ‘Autism’ and the drug node ‘Risperidone’

TDC-2 DSL for multimodality

Contextualizing datasets with resources, for ML-ready multimodal data views

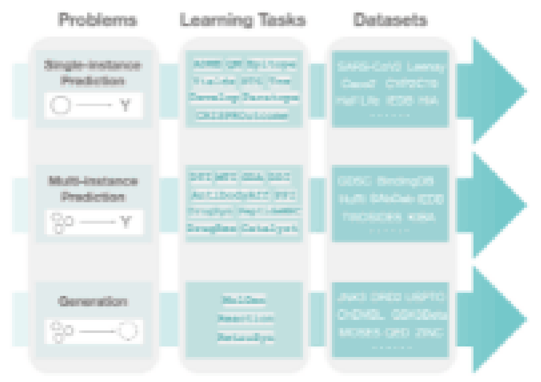
TDC-2's **Application-Embedded Domain-Specific Data Definition Programming Language**. Can be used to, for example, create a contextualized view of drug-target interaction datasets using single-cell-resolution embeddings



```
class scDTI(ResourceConfig):
    """Configuration for contextualized drug-target-identification dataset"""

    def __init__(self):
    super(scDTI, self).__init__(
        ResourceFeatureGenerator(),
        keys=["df"], # keys in dataloader to update
        loader_functions=["join"], # functions to run over the input parameters
        loader_args=[{
        "ds_list": [
            "opentargets_ibd", "opentargets_ra", "Tabula Sapiens - All Cells",
"pinnacle_embeds"
        ],
        "columns": [["gene_id"], ["gene_id", "cell_id"]], "method": "inner"

        }])
```

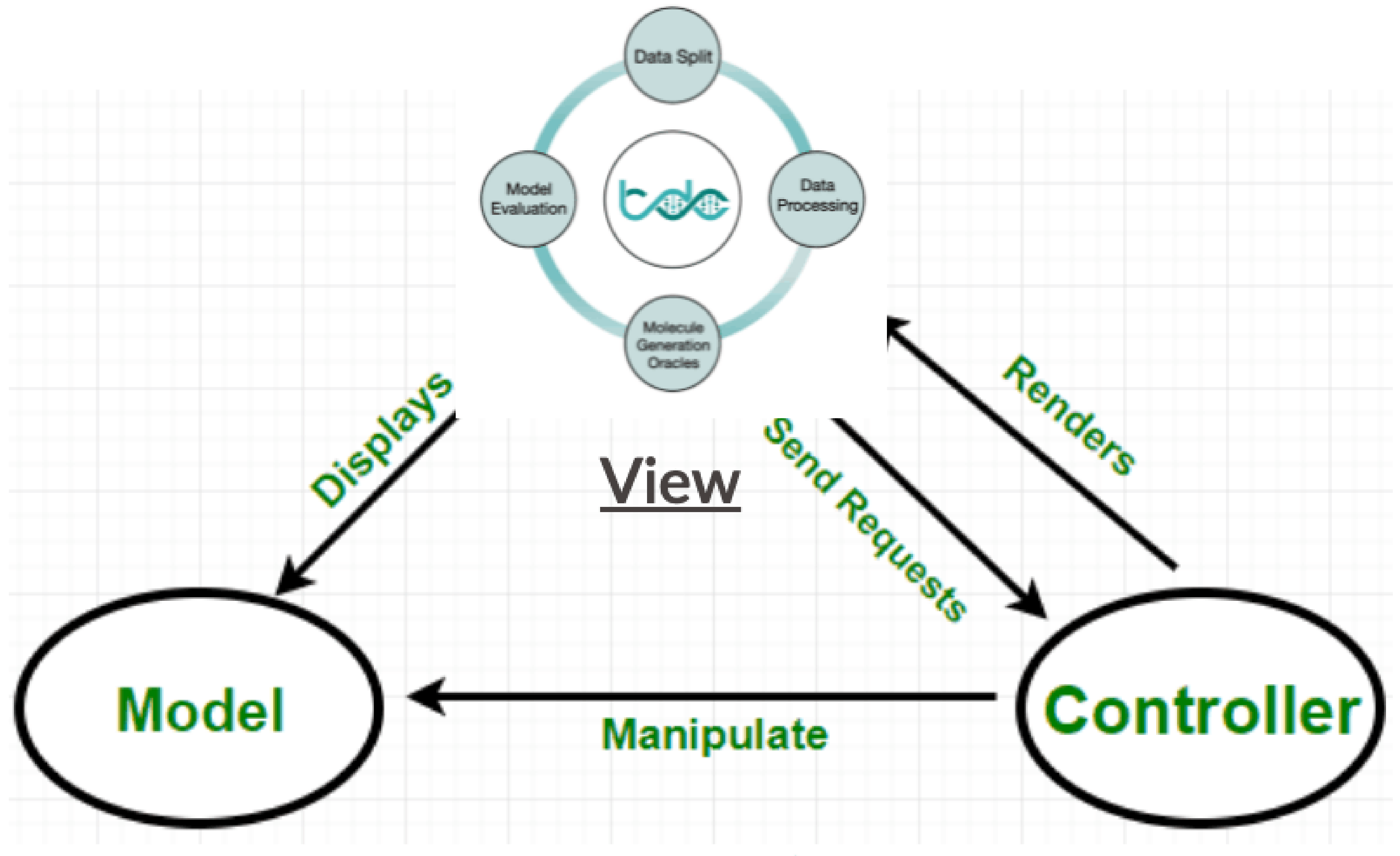
TDC-2 Model-View-Controller

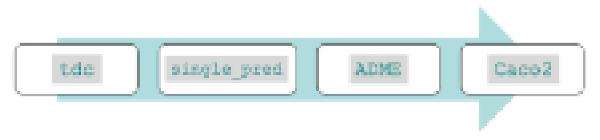
enhanced dataset retrieval with contextualization for diverse ML tasks



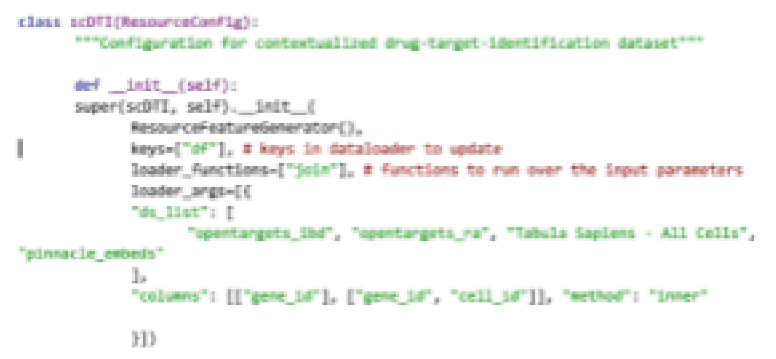
TDC Datasets

TDC-2 Resources





DataLoader



```
class scDTI(ResourceConfig):
    """Configuration for contextualized drug-target-identification dataset"""

    def __init__(self):
    super(scDTI, self).__init__(
        ResourceFeatureGenerator(),
        keys=["df"], # keys in dataloader to update
        loader_functions=["join"], # functions to run over the input parameters
        loader_args=[{
        "ds_list": [
            "opentargets_ibd", "opentargets_ra", "Tabula Sapiens - All Cells",
"pinnacle_embeds"
        ],
        "columns": [["gene_id"], ["gene_id", "cell_id"]], "method": "inner"

        }])
```

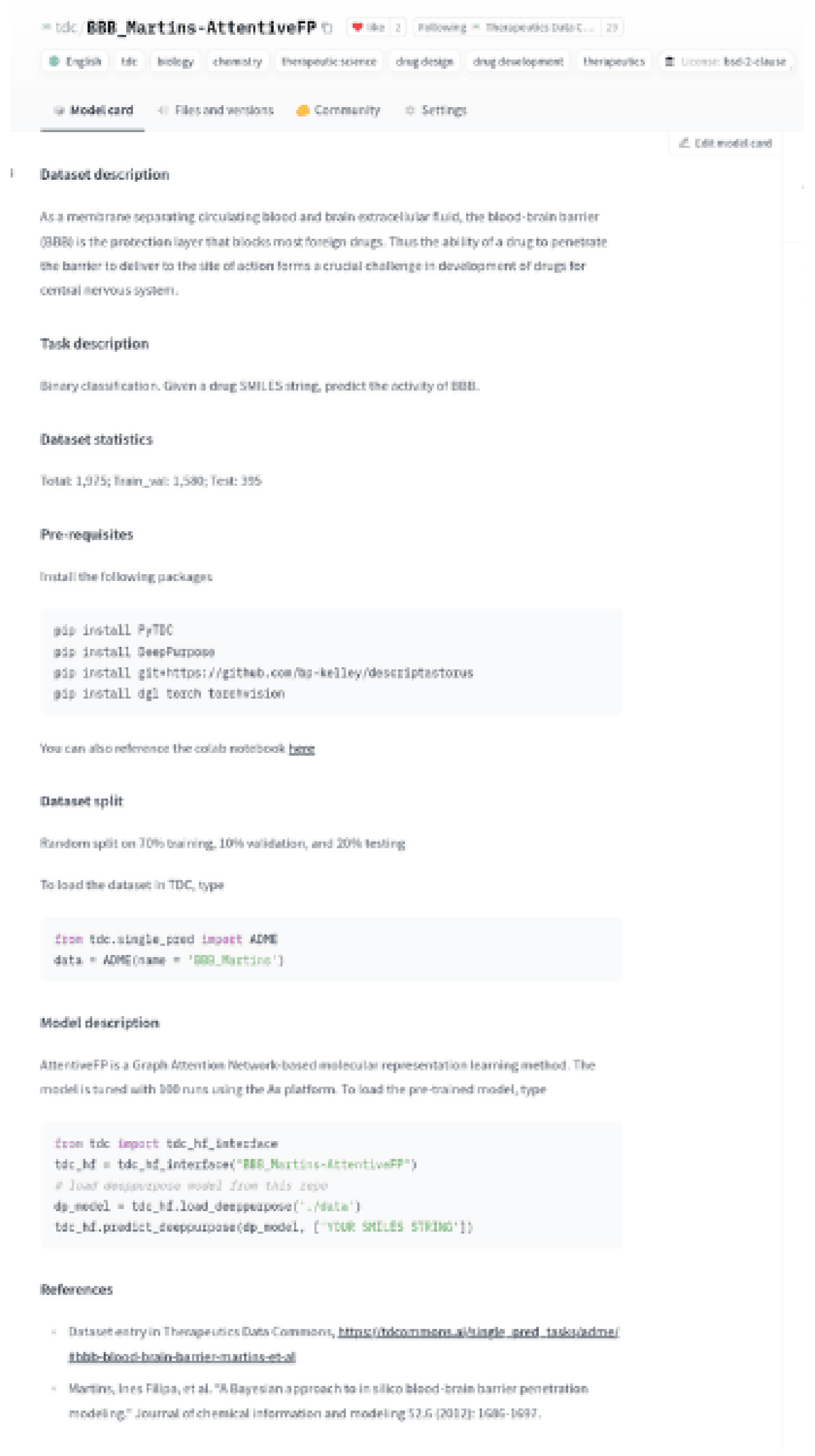
TDC-2 DSL

Model Server and TDC ML Platform

Embedding retrieval + Training and inference on pre-trained therapeutic ML models

TDC Model Server

TDC Huggingface Model Hub



tdc / **BBB_Martins-AttentiveFP**   like 2   Following Therapeutics Data C… 23

English · tdc · biology · chemistry · therapeutic science · drug design · drug development · therapeutics · License: bsd-2-clause

Model card · Files and versions · Community · Settings

Edit model card

Dataset description

As a membrane separating circulating blood and brain extracellular fluid, the blood-brain barrier (BBB) is the protection layer that blocks most foreign drugs. Thus the ability of a drug to penetrate the barrier to deliver to the site of action forms a crucial challenge in development of drugs for central nervous system.

Task description

Binary classification. Given a drug SMILES string, predict the activity of BBB.

Dataset statistics

Total: 1,975; Train_val: 1,580; Test: 395

Pre-requisites

Install the following packages

```
pip install PyTDC
pip install DeepPurpose
pip install git+https://github.com/bp-kelley/descriptastorus
pip install dgl torch torchvision
```

You can also reference the colab notebook here

Dataset split

Random split on 70% training, 10% validation, and 20% testing

To load the dataset in TDC, type

```
from tdc.single_pred import ADME
data = ADME(name = 'BBB_Martins')
```

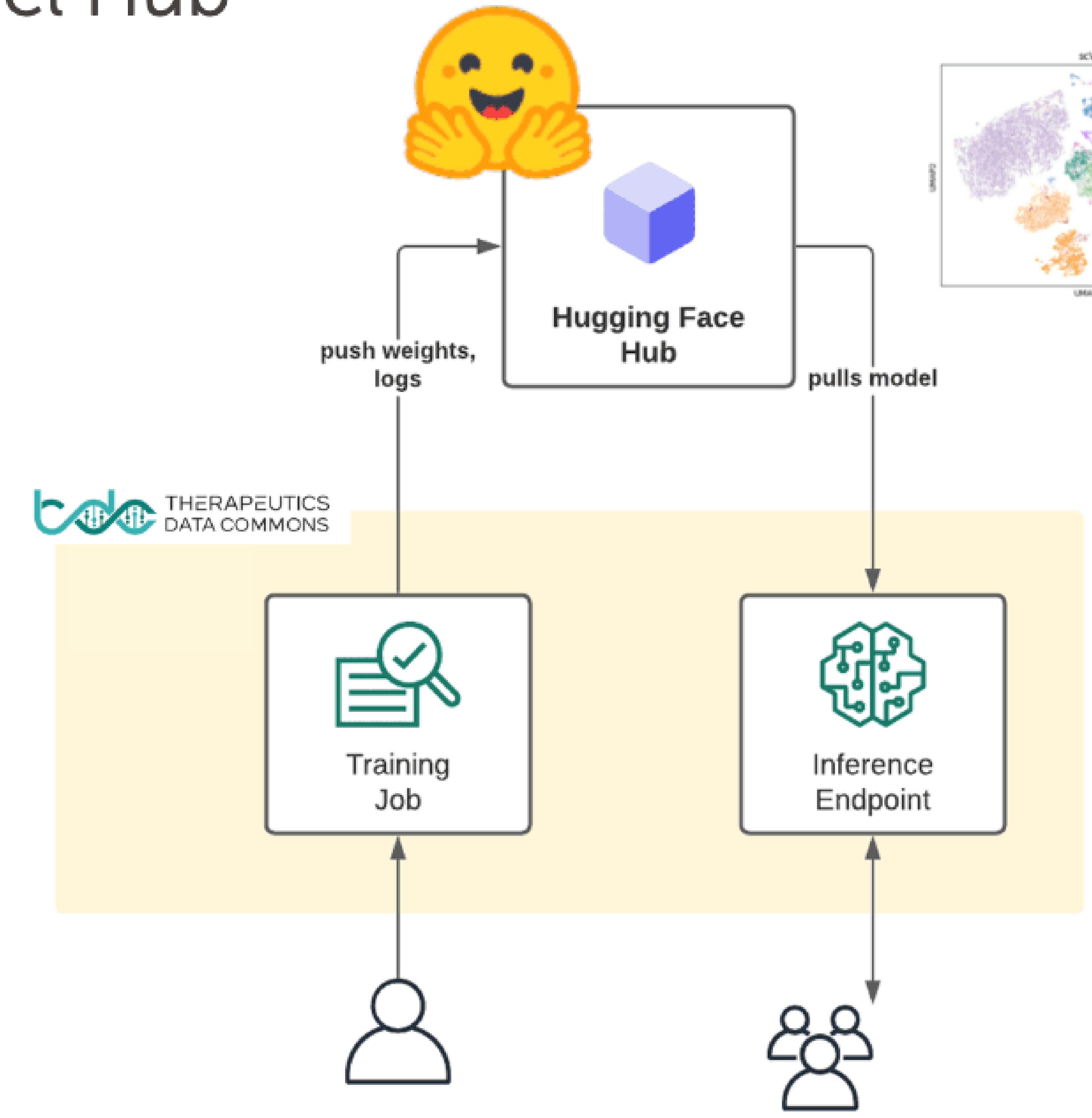
Model description

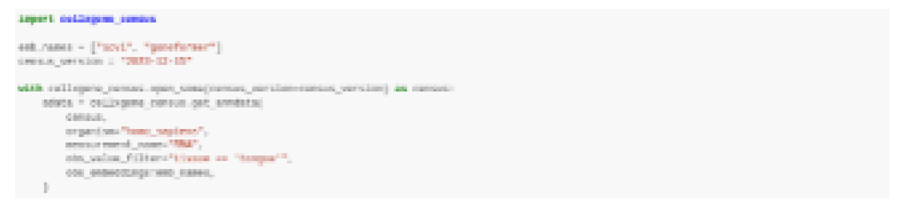
AttentiveFP is a Graph Attention Network-based molecular representation learning method. The model is tuned with 100 runs using the Ax platform. To load the pre-trained model, type

```
from tdc import tdc_hf_interface
tdc_hf = tdc_hf_interface("BBB_Martins-AttentiveFP")
# load deeppurpose model from this repo
dp_model = tdc_hf.load_deeppurpose('./data')
tdc_hf.predict_deeppurpose(dp_model, ['YOUR SMILES STRING'])
```

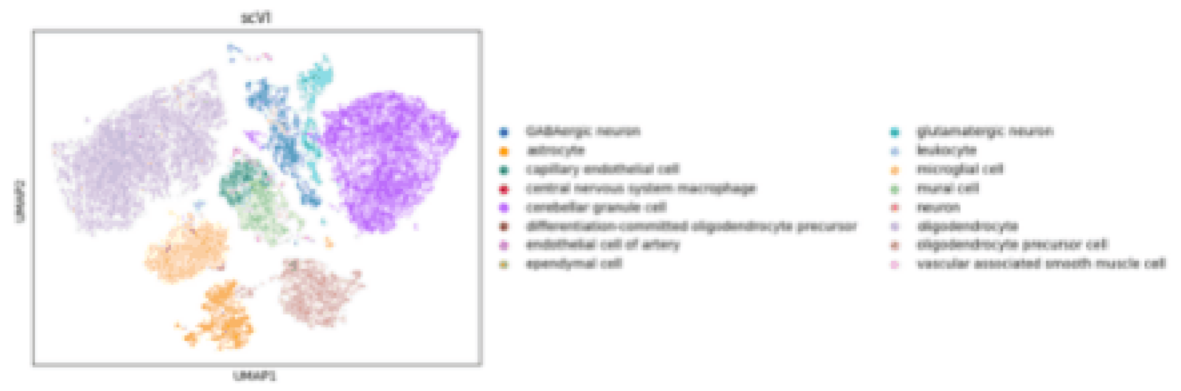
References

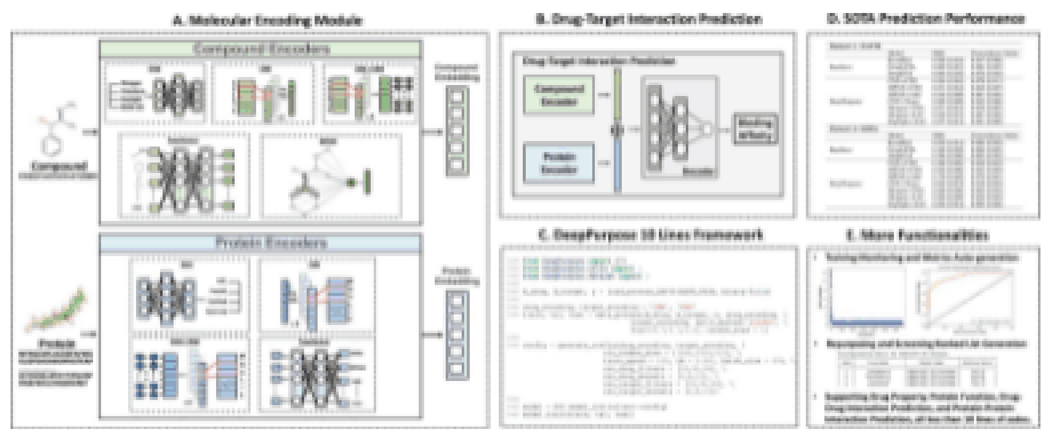
- Dataset entry in Therapeutics Data Commons, https://tdcommons.ai/single_pred_tasks/adme/#bbb-blood-brain-barrier-martins-et-al
- Martins, Ines Filipa, et al. "A Bayesian approach to in silico blood-brain barrier penetration modeling." Journal of chemical information and modeling 52.6 (2012): 1686-1697.



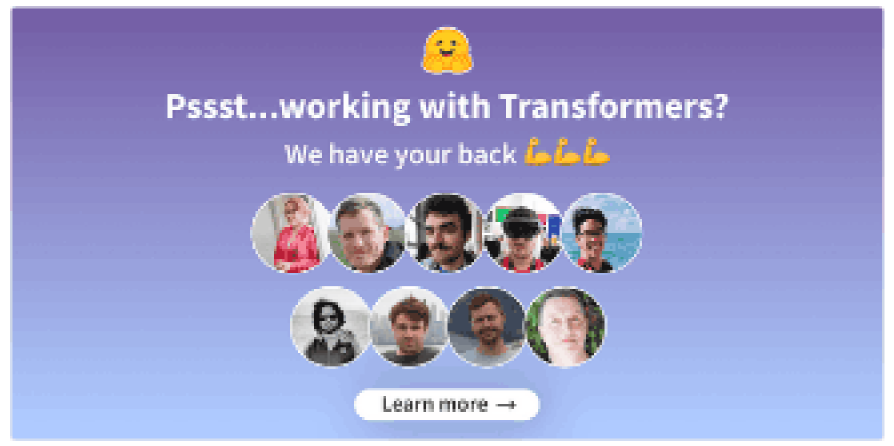


```
import cellxgene_census

emb_names = ["scvi", "geneformer"]
census_version = "2023-12-15"

with cellxgene_census.open_soma(census_version=census_version) as census:
    adata = cellxgene_census.get_anndata(
        census,
        organism="homo_sapiens",
        measurement_name="RNA",
        obs_value_filter="tissue == 'tongue'",
        obs_embeddings=emb_names,
    )
```





A. Molecular Encoding Module
B. Drug-Target Interaction Prediction
D. SOTA Prediction Performance
Compound Encoders
Protein Encoders
C. DeepPurpose 10 Lines Framework
E. More Functionalities



Listing 7: The below illustrates the basic functionality of the model hub to download a model and perform inference on a precdictive task as well as fine-tune the model

```
from tdc import tdc_hf_interface
tdc_hf = tdc_hf_interface("BBB_Martins-AttentiveFP")
# load deeppurpose model from this repo
dp_model = tdc_hf.load_deeppurpose('./data')
tdc_hf.predict_deeppurpose(dp_model, ['YOUR␣SMILES␣STRING'])
# fine-tune
dp_model.train(train, val, test) # for some defined splits
```

Listing 8: The below illustrates using the tdc model hub to download a foundation model [3]

```
from tdc import tdc_hf_interface
from transformers import BertModel
geneformer = tdc_hf_interface("Geneformer")
model = geneformer.load()
assert isinstance(model, BertModel), type(model)
```

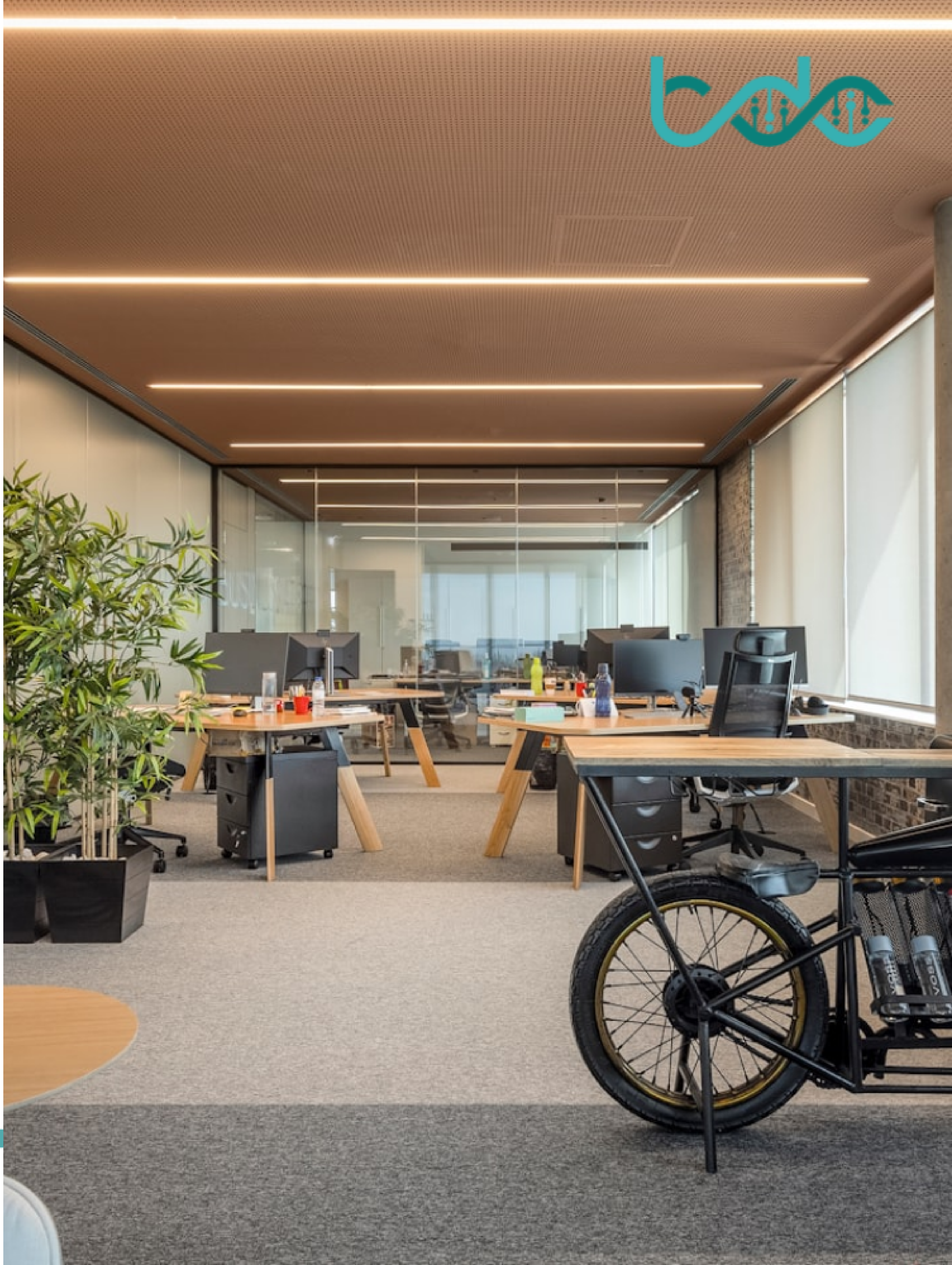
Listing 9: Beyond downloading a foundation model [3], the model server facilitates model inference across a range of datasets. Below an example integrating the TDC-2 CellXGene API with the model server.

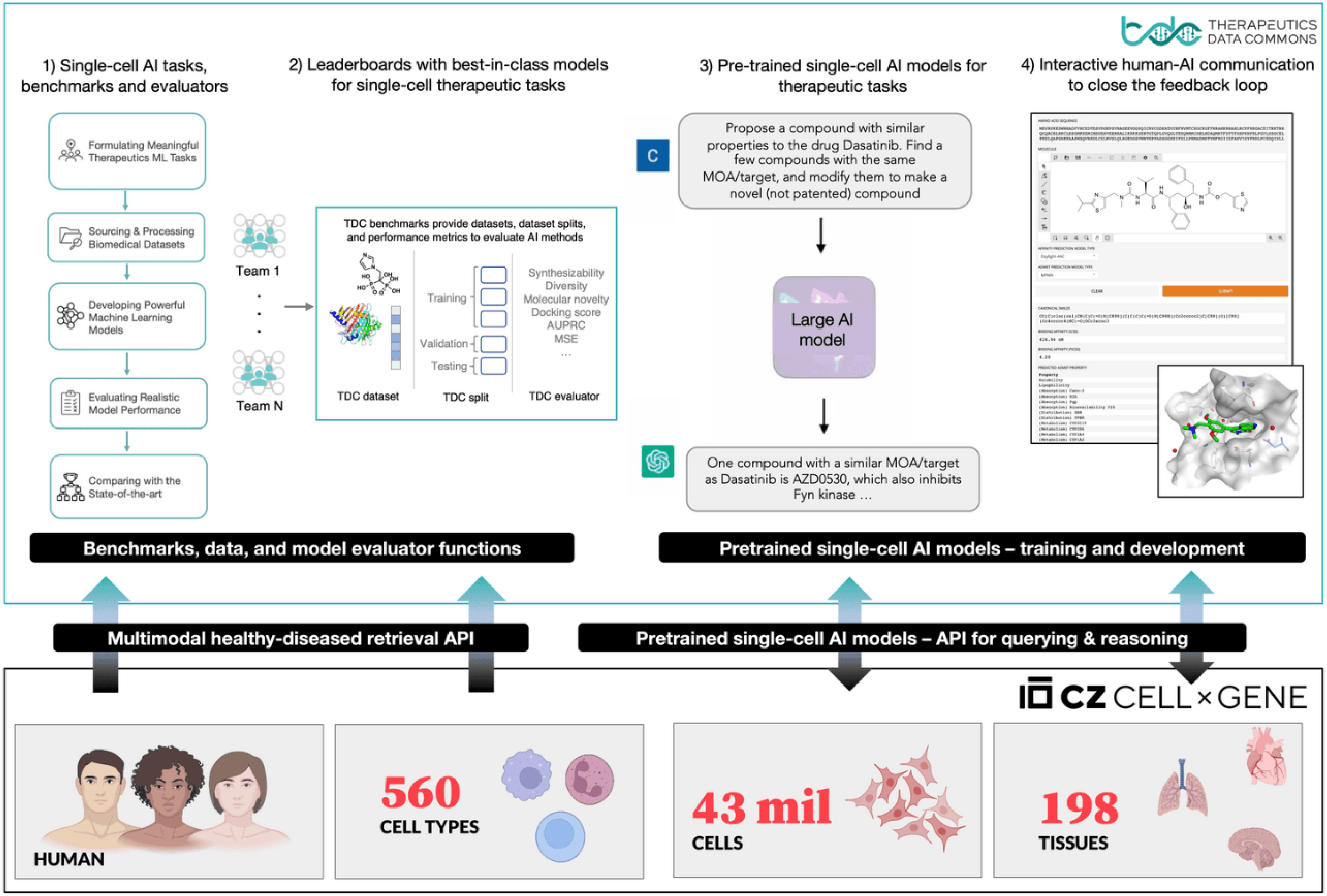
```
from tdc.resource import cellxgene_census
from tdc.model_server.tokenizers.geneformer import GeneformerTokenizer
from tdc import tdc_hf_interface
import torch

# query the CELLXGENE census
adata = self.resource.get_anndata(
var_value_filter=
"feature_id␣in␣['ENSG00000161798',␣'ENSG00000188229']",
obs_value_filter=
"sex␣==␣'female'␣and␣cell_type␣in␣['microglial␣cell',␣'neuron']",
column_names={
    "obs": [
        "assay", "cell_type", "tissue", "tissue_general",
        "suspension_type", "disease"
    ]
},
)

# tokenize gene expression vectors
tokenizer = GeneformerTokenizer()
x = tokenizer.tokenize_cell_vectors(adata,
                                ensembl_id="feature_id",
                                ncounts="n_measured_vars")

cells, _ = x

# load the model
geneformer = tdc_hf_interface("Geneformer")
model = geneformer.load()

"""
Custom pre-processing code can include padding and attention mask
    ↪ definitions.
```

```
"""

input_tensor = torch.tensor(cells)
out = []
for batch in input_tensor:
    # build an attention mask
    attention_mask = torch.tensor(
        [[x[0] != 0, x[1] != 0] for x in batch])
    # run batched inference
    out.append(model(batch, attention_mask=attention_mask))
```

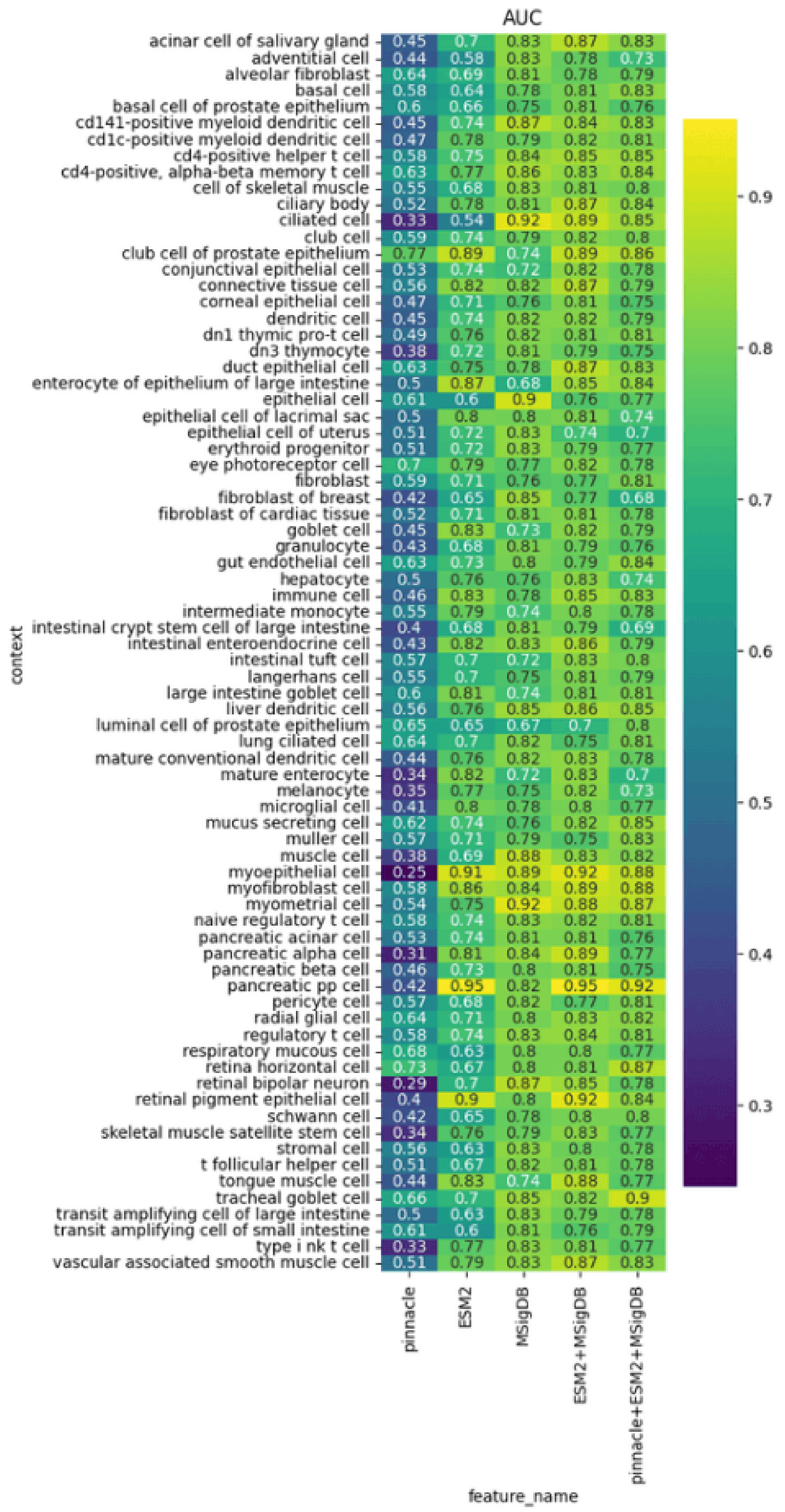
Future Directions

Nov 14th, 2024 | alejandro_velez-arce@hms.harvard.edu




THERAPEUTICS DATA COMMONS
1) Single-cell AI tasks, benchmarks and evaluators
2) Leaderboards with best-in-class models for single-cell therapeutic tasks
3) Pre-trained single-cell AI models for therapeutic tasks
4) Interactive human-AI communication to close the feedback loop
Formulating Meaningful Therapeutics ML Tasks
Sourcing & Processing Biomedical Datasets
Developing Powerful Machine Learning Models
Evaluating Realistic Model Performance
Comparing with the State-of-the-art
Team 1
Team N
TDC benchmarks provide datasets, dataset splits, and performance metrics to evaluate AI methods
Training
Validation
Testing
Synthesizability
Diversity
Molecular novelty
Docking score
AUPRC
MSE
TDC dataset
TDC split
TDC evaluator
Propose a compound with similar properties to the drug Dasatinib. Find a few compounds with the same MOA/target, and modify them to make a novel (not patented) compound
Large AI model
One compound with a similar MOA/target as Dasatinib is AZD0530, which also inhibits Fyn kinase …
Benchmarks, data, and model evaluator functions
Pretrained single-cell AI models – training and development
Multimodal healthy-diseased retrieval API
Pretrained single-cell AI models – API for querying & reasoning
CZ CELL×GENE
HUMAN
560 CELL TYPES
43 mil CELLS
198 TISSUES

TDC-2+: Multimodal Retrieval, Model Hub, and LLm Agents



context	pinnacle	ESM2	MSigDB	ESM2+MSigDB	pinnacle+ESM2+MSigDB
acinar cell of salivary gland	0.45	0.7	0.83	0.87	0.83
adventitial cell	0.44	0.58	0.83	0.78	0.73
alveolar fibroblast	0.64	0.69	0.81	0.78	0.79
basal cell	0.58	0.64	0.78	0.81	0.83
basal cell of prostate epithelium	0.6	0.66	0.75	0.81	0.76
cd141-positive myeloid dendritic cell	0.45	0.74	0.87	0.84	0.83
cd1c-positive myeloid dendritic cell	0.47	0.78	0.79	0.82	0.81
cd4-positive helper t cell	0.58	0.75	0.84	0.85	0.85
cd4-positive, alpha-beta memory t cell	0.63	0.77	0.86	0.83	0.84
cell of skeletal muscle	0.55	0.68	0.83	0.81	0.8
ciliary body	0.52	0.78	0.81	0.87	0.84
ciliated cell	0.33	0.54	0.92	0.89	0.85
club cell	0.59	0.74	0.79	0.82	0.8
club cell of prostate epithelium	0.77	0.89	0.74	0.89	0.86
conjunctival epithelial cell	0.53	0.74	0.72	0.82	0.78
connective tissue cell	0.56	0.82	0.82	0.87	0.79
corneal epithelial cell	0.47	0.71	0.76	0.81	0.75
dendritic cell	0.45	0.74	0.82	0.82	0.79
dn1 thymic pro-t cell	0.49	0.76	0.82	0.81	0.81
dn3 thymocyte	0.38	0.72	0.81	0.79	0.75
duct epithelial cell	0.63	0.75	0.78	0.87	0.83
enterocyte of epithelium of large intestine	0.5	0.87	0.68	0.85	0.84
epithelial cell	0.61	0.6	0.9	0.76	0.77
epithelial cell of lacrimal sac	0.5	0.8	0.8	0.81	0.74
epithelial cell of uterus	0.51	0.72	0.83	0.74	0.7
erythroid progenitor	0.51	0.72	0.83	0.79	0.77
eye photoreceptor cell	0.7	0.79	0.77	0.82	0.78
fibroblast	0.59	0.71	0.76	0.77	0.81
fibroblast of breast	0.42	0.65	0.85	0.77	0.68
fibroblast of cardiac tissue	0.52	0.71	0.81	0.81	0.78
goblet cell	0.45	0.83	0.73	0.82	0.79
granulocyte	0.43	0.68	0.81	0.79	0.76
gut endothelial cell	0.63	0.73	0.8	0.79	0.84
hepatocyte	0.5	0.76	0.76	0.83	0.74
immune cell	0.46	0.83	0.78	0.85	0.83
intermediate monocyte	0.55	0.79	0.74	0.8	0.78
intestinal crypt stem cell of large intestine	0.4	0.68	0.81	0.79	0.69
intestinal enteroendocrine cell	0.43	0.82	0.83	0.86	0.79
intestinal tuft cell	0.57	0.7	0.72	0.83	0.8
langerhans cell	0.55	0.7	0.75	0.81	0.79
large intestine goblet cell	0.6	0.81	0.74	0.81	0.81
liver dendritic cell	0.56	0.76	0.85	0.86	0.85
luminal cell of prostate epithelium	0.65	0.65	0.67	0.7	0.8
lung ciliated cell	0.64	0.7	0.82	0.75	0.81
mature conventional dendritic cell	0.44	0.76	0.82	0.83	0.78
mature enterocyte	0.34	0.82	0.72	0.83	0.7
melanocyte	0.35	0.77	0.75	0.82	0.73
microglial cell	0.41	0.8	0.78	0.8	0.77
mucus secreting cell	0.62	0.74	0.76	0.82	0.85
muller cell	0.57	0.71	0.79	0.75	0.83
muscle cell	0.38	0.69	0.88	0.83	0.82
myoepithelial cell	0.25	0.91	0.89	0.92	0.88
myofibroblast cell	0.58	0.86	0.84	0.89	0.88
myometrial cell	0.54	0.75	0.92	0.88	0.87
naive regulatory t cell	0.58	0.74	0.83	0.82	0.81
pancreatic acinar cell	0.53	0.74	0.81	0.81	0.76
pancreatic alpha cell	0.31	0.81	0.84	0.89	0.77
pancreatic beta cell	0.46	0.73	0.8	0.81	0.75
pancreatic pp cell	0.42	0.95	0.82	0.95	0.92
pericyte cell	0.57	0.68	0.82	0.77	0.81
radial glial cell	0.64	0.71	0.8	0.83	0.82
regulatory t cell	0.58	0.74	0.83	0.84	0.81
respiratory mucous cell	0.68	0.63	0.8	0.8	0.77
retina horizontal cell	0.73	0.67	0.8	0.81	0.87
retinal bipolar neuron	0.29	0.7	0.87	0.85	0.78
retinal pigment epithelial cell	0.4	0.9	0.8	0.92	0.84
schwann cell	0.42	0.65	0.78	0.8	0.8
skeletal muscle satellite stem cell	0.34	0.76	0.79	0.83	0.77
stromal cell	0.56	0.63	0.83	0.8	0.78
t follicular helper cell	0.51	0.67	0.82	0.81	0.78
tongue muscle cell	0.44	0.83	0.74	0.88	0.77
tracheal goblet cell	0.66	0.7	0.85	0.82	0.9
transit amplifying cell of large intestine	0.5	0.63	0.83	0.79	0.78
transit amplifying cell of small intestine	0.61	0.6	0.81	0.76	0.79
type i nk t cell	0.33	0.77	0.83	0.81	0.77
vascular associated smooth muscle cell	0.51	0.79	0.83	0.87	0.83

The Commons 2.0 (TDC-2)

Multimodal AI Foundations for Therapeutic Science

Alejandro Velez-Arce, Kexin Huang, Michelle M. Li, Xiang Lin, Wenhao Gao, Tianfan Fu, Manolis Kellis, Bradley L. Pentelute, Marinka Zitnik

THERAPEUTICS DATA COMMONS

5 Foundation model embeddings *via TDC HF Model Hub*

1,000+ Multi-modal single-cell datasets *via CZ CellXGene*

4M+ Precision medicine knowledge base *via TDC-PrimeKG*

55 Therapeutics heuristics algorithms *via TDC data functions*

15,933 clinical trials

47,182 TCR-epitope pairs

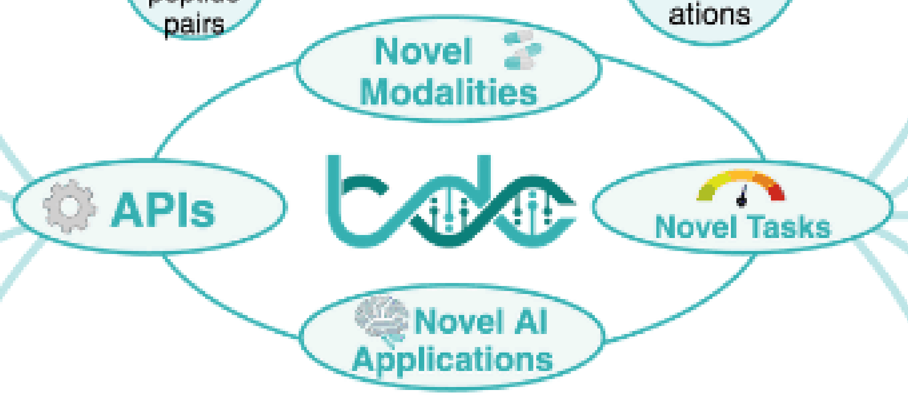
85M single cell expressions

308 cell contexts

4,480 protein-peptide pairs

35,479 protein-ligand pairs

5,225 CRISPR perturbations



Contextualized drug-target identification

Single-cell Perturbation response prediction

Protein-peptide binding affinity prediction

Clinical trial outcome prediction

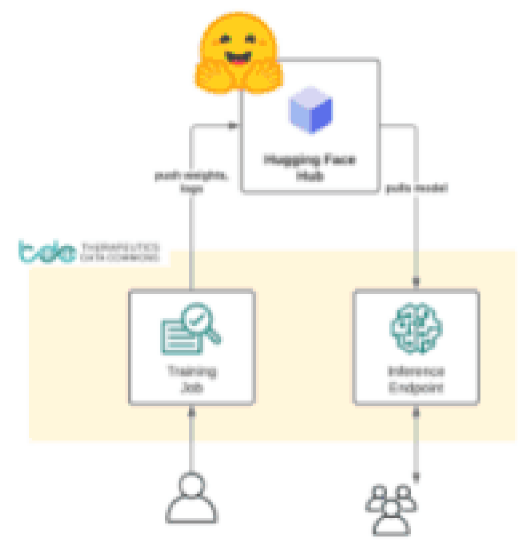
Structure-based drug design

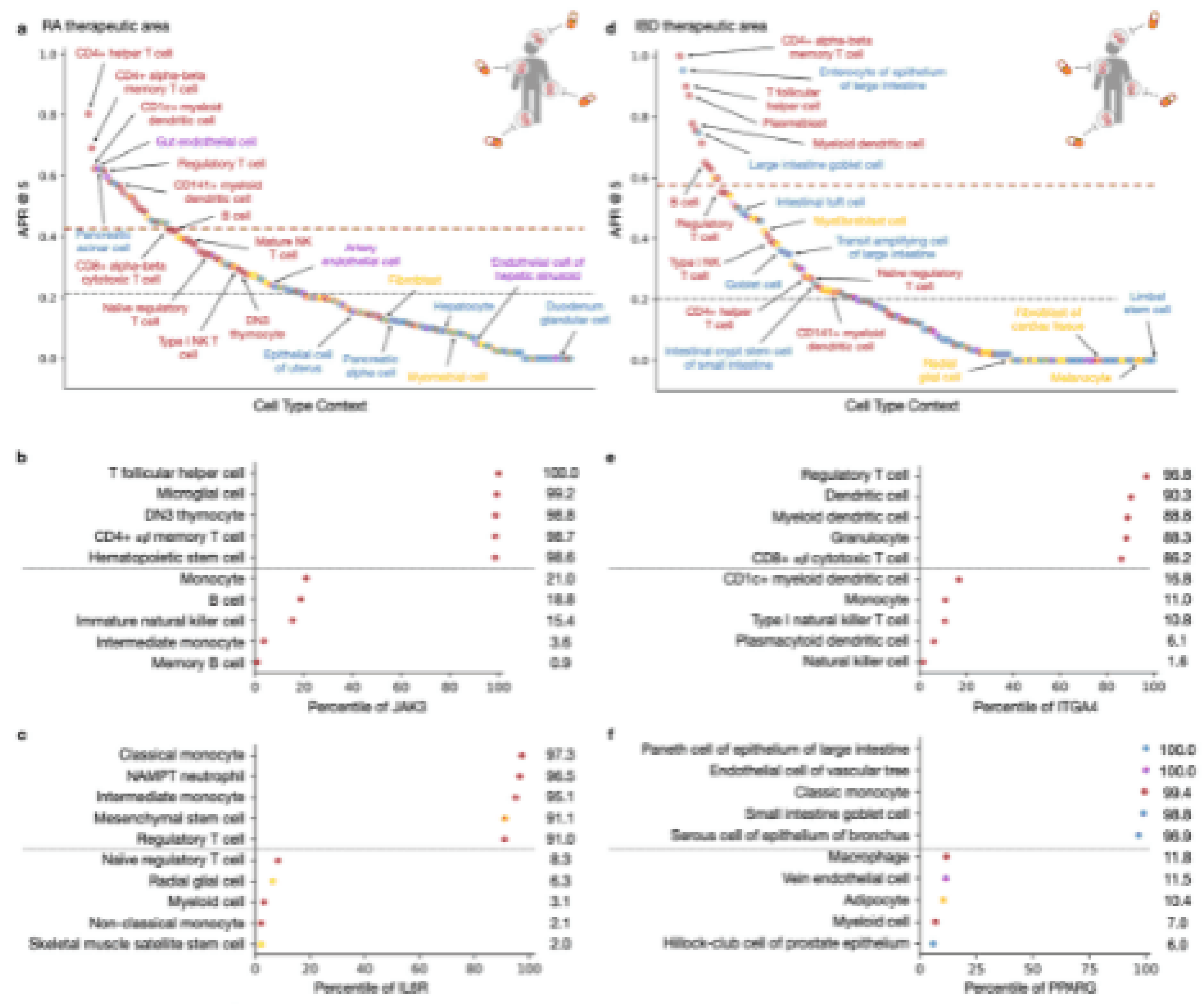
Biological foundation model

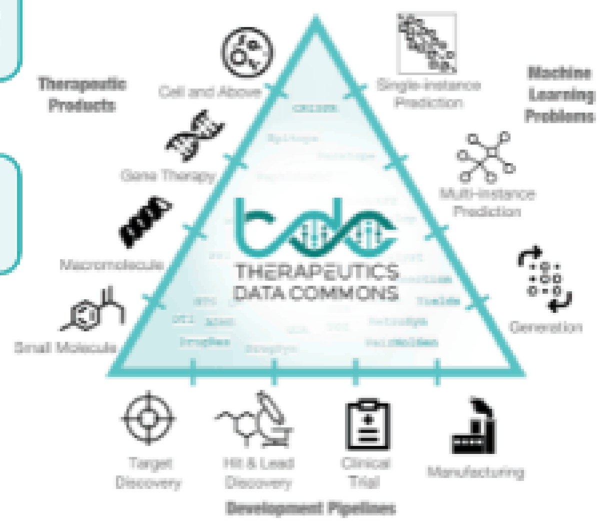
OOD generalization

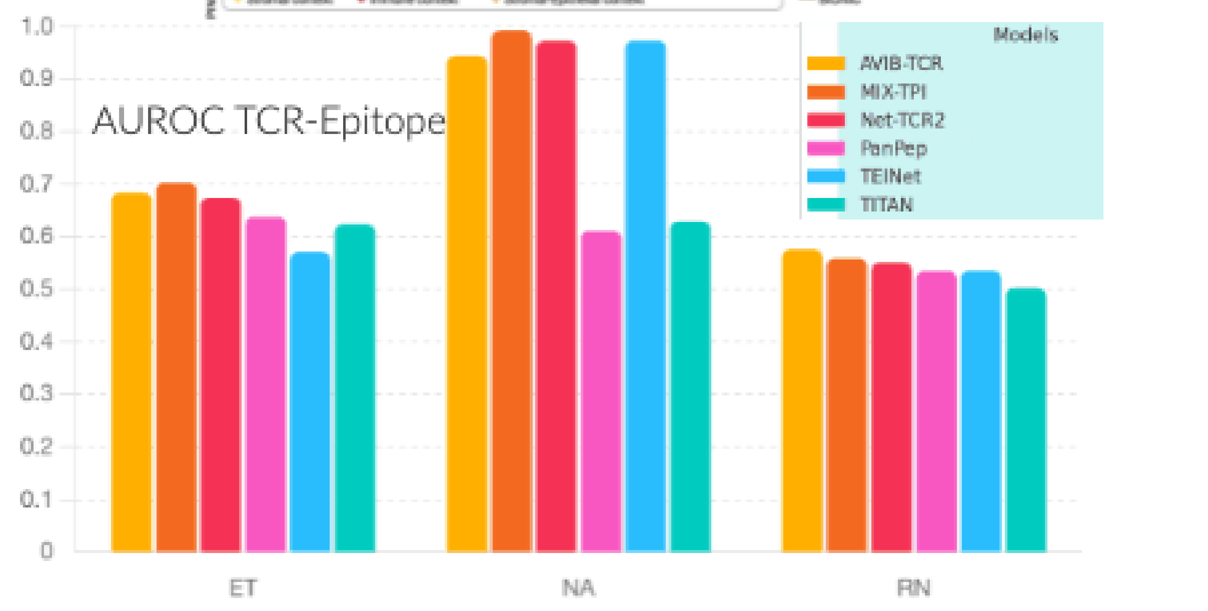
Cell-type specific contextual AI model

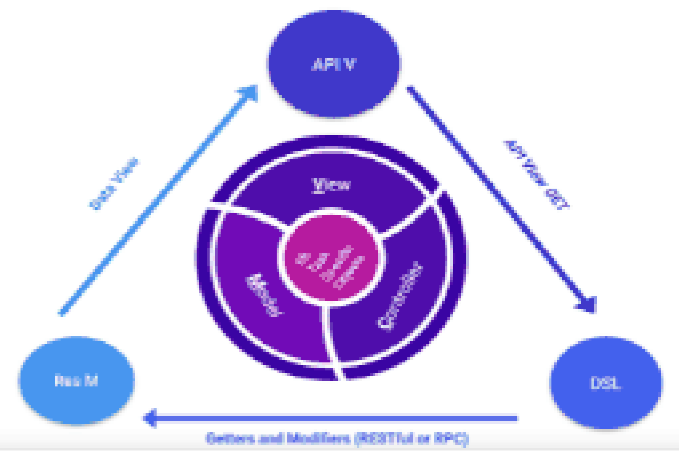
LLM agent

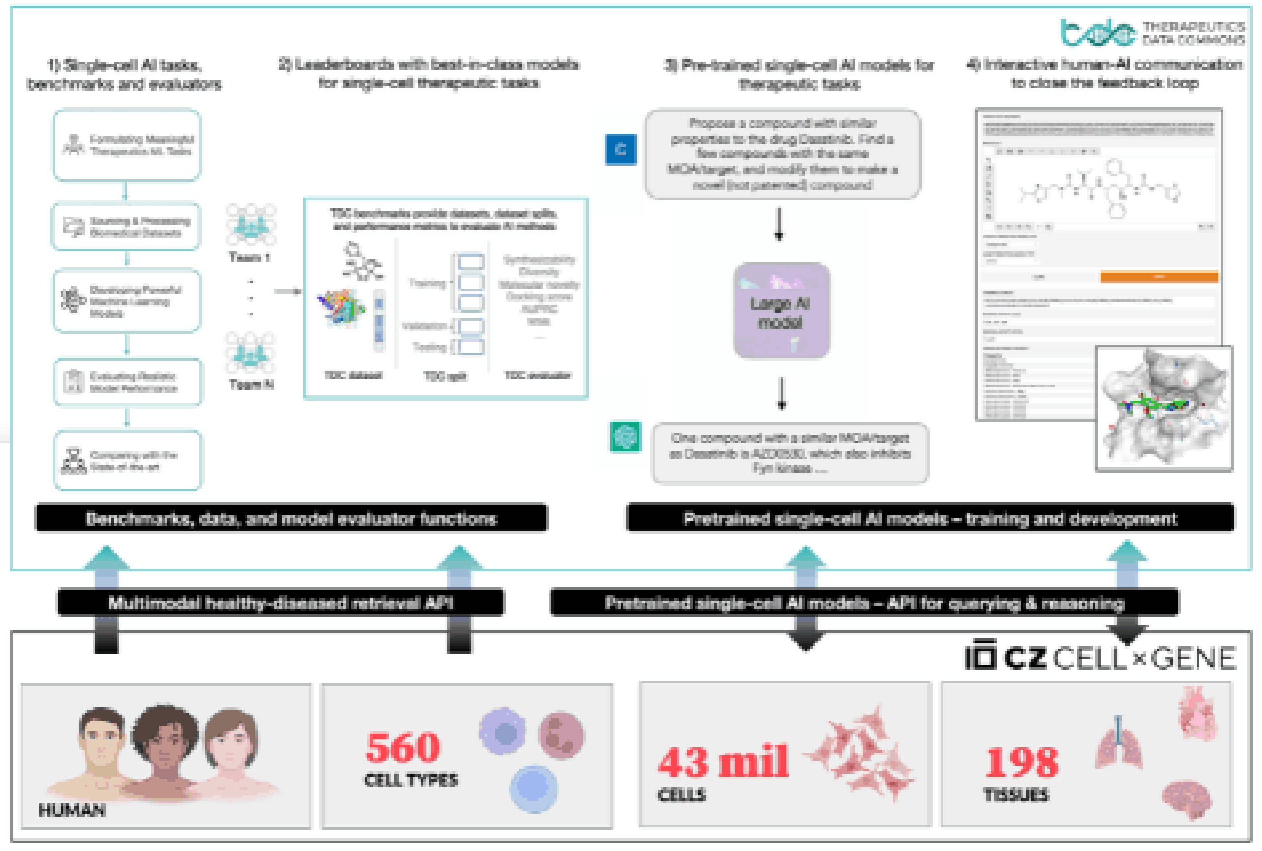


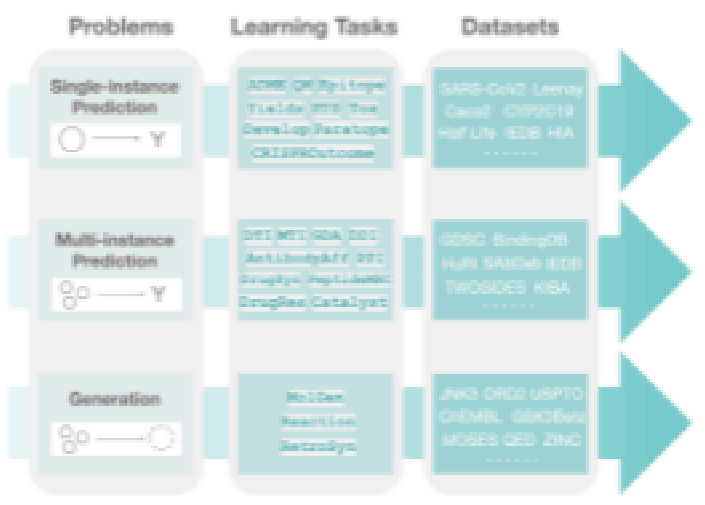


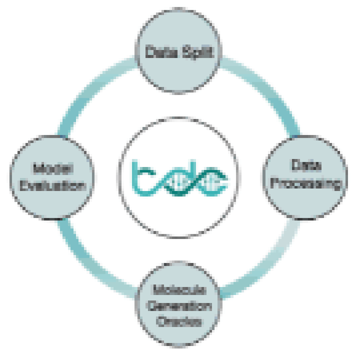


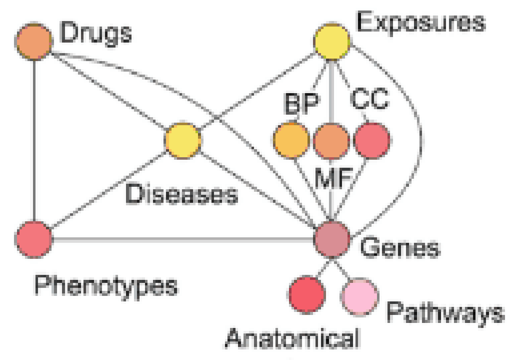


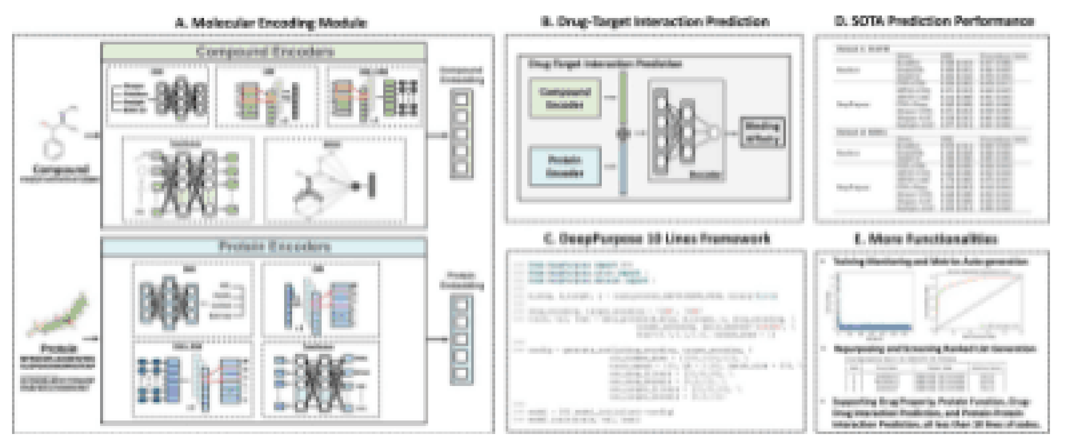



1) Single-cell AI tasks, benchmarks and evaluators
2) Leaderboards with best-in-class models for single-cell therapeutic tasks
3) Pre-trained single-cell AI models for therapeutic tasks
4) Interactive human-AI communication to close the feedback loop
Large AI model
Benchmarks, data, and model evaluator functions
Pretrained single-cell AI models – training and development
Multimodal healthy-diseased retrieval API
Pretrained single-cell AI models – API for querying & reasoning
CZ CELL×GENE
HUMAN
560 CELL TYPES
43 mil CELLS
198 TISSUES








A. Molecular Encoding Module
B. Drug-Target Interaction Prediction
D. SOTA Prediction Performance
C. DeepPurpose 10 Lines Framework
E. More Functionalities

Why would we need single cell therapeutics

slide 1: problem

slide 2 : what has been done

slide 3: gap (we know drugs vary by cell...)

slide 4: goal is to show my research bridging this gap

In the above slide we should emphasize the connections are not there (ie. "crosses on the links")