TOSICA, GET, Baseline and Cell-eval

XIN Baiying

August 26, 2025

Overview

1. **Transformer for one stop interpretable cell type annotation (TOSICA)**
2. **A foundation model of transcription across human cell types (GET)**
3. **VCC Baseline and Cell-eval**

Transformer for one stop interpretable cell type annotation (TOSICA)

Chen J, Xu H, Tao W, Chen Z, Zhao Y, Han JJ

Nature Communications, 2023

DOI: 10.1038/s41467-023-35923-4

https://www.nature.com/articles/s41467-023-35923-4.pdf

Introduction

The general goal of TOSICA is **cell type annotation**.

- Its input is a single-cell transcriptome expression profile $\mathbf{e} \in \mathbb{R}^n$, where n is the number of genes (10000+).
- The output is the predicted cell type $\hat{y} \in \{1, ..., C\}$, where C is the number of cell types ($10 \sim 50$).

TOSICA Architecture

TOSICA is based on the Transformer architecture and introduces a mask mechanism with biological prior knowledge. Its design consists of three layers:

- **Cell Embedding:** Maps each cell's expression to a low-dimensional representation space, where each dimension corresponds to a biological pathway/regulatory module.
- **Multi-Head Self-Attention:** Uses attention mechanism to compute attention between the cell embedding and classification CLS token to obtain contextual information for cell types.
- **Cell-Type Classifier:** Maps attention information to cell type space through fully connected layers to achieve final classification prediction.

Cell Embedding I

For each cell $\mathbf{e} \in \mathbb{R}^n$, where n is the number of genes, we aim to map it to a low-dimensional representation through a fully connected network:

$$\mathbf{t} = \mathbf{W}'\mathbf{e} \in \mathbb{R}^k$$

where $\mathbf{W}' \in \mathbb{R}^{k \times n}$ is a learnable weight matrix. In this k-dimensional space, each dimension serves as a token, and each token represents a biological pathway/regulatory module.

Since we want $\mathbf{t}$ to have each dimension represent a specific biologically meaningful pathway in a structured manner, the weight matrix $\mathbf{W}'$ needs to be specially designed with prior knowledge.
To integrate biological prior knowledge into the cell embedding process:

- First introduce a general weight matrix $\mathbf{W} \in \mathbb{R}^{k \times n}$

Cell Embedding II

- Based on prior knowledge, we construct a mask matrix $\mathbf{M} \in \{0,1\}^{k \times n}$
  - Where $M_{ij} = 1$ if and only if the i-th pathway contains the j-th gene. This knowledge is obtained from external databases (specifically, gene set datasets from the Gene Set Enrichment Analysis database)
  - Therefore, only genes within the same pathway are mapped to the same token. This avoids the non-interpretability of mixed components, though it also limits the model's expressive power.
- Then perform Hadamard product (element-wise product) on these two matrices to obtain the final weight matrix:

$$\mathbf{W}' = \mathbf{M} \odot \mathbf{W}$$
$$\mathbf{t} = \mathbf{W}'\mathbf{e} \in \mathbb{R}^k$$

Cell Embedding III

To enhance performance, the above operation is performed in parallel m times ($m = 48$), concatenated to obtain:

$$\mathbf{T} := \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_m \end{bmatrix} \in \mathbb{R}^{k \times m}.$$

Multi-Head Self-Attention I

First, we introduce a learnable dummy token $\mathbf{cls} \in \mathbb{R}^m$ and concatenate it as the first row of $\mathbf{T}$ to obtain the new input:

$$\mathbf{I} := \begin{bmatrix} \mathbf{cls}^\top \\ \mathbf{T} \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & \cdots & c_m \\ \mathbf{t}_1 & \mathbf{t}_2 & \cdots & \mathbf{t}_m \end{bmatrix} \in \mathbb{R}^{(1+k)\times m}.$$

- This is a common practice in Transformer and other NLP models. CLS is an additional, learnable virtual token. Since attention operations output a representation with the same shape as the input, we can use the output of this CLS as the global information extraction result for the current input, similar to a statistic of the current input, serving as the token for subsequent classification.

We feed this input $\mathbf{I}$ into the attention mechanism. We first discuss the case of a single head. Overall, we have $\mathbf{O} = \text{Attention}(\mathbf{I}) \in \mathbb{R}^{(1+k)\times m}$. The specific computational details are as follows:

Multi-Head Self-Attention II

- Through three linear transformations $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{(1+k)\times(1+k)}$, we map the input to Query, Key, and Value spaces:

$$\mathbf{Q} = \mathbf{W}_q\mathbf{I} \in \mathbb{R}^{(1+k)\times m},$$
$$\mathbf{K} = \mathbf{W}_k\mathbf{I} \in \mathbb{R}^{(1+k)\times m},$$
$$\mathbf{V} = \mathbf{W}_v\mathbf{I} \in \mathbb{R}^{(1+k)\times m}.$$

- We compute attention scores to obtain contextual information:

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) \in \mathbb{R}^{(1+k)\times(1+k)}.$$

where $d_k = m$ is the dimension of the Key vectors. This attention score matrix $\mathbf{A}$ represents the similarity between different tokens.

Multi-Head Self-Attention III

- We obtain contextual representations for each token through weighted summation:

$$\mathbf{O} = \mathbf{AV} \in \mathbb{R}^{(1+k)\times m}.$$

The multi-head case is similar to the above, equivalent to performing H computations in parallel.

- For the h-th head, we independently introduce learnable weight matrices $\mathbf{W}_q^h, \mathbf{W}_k^h, \mathbf{W}_v^h \in \mathbb{R}^{(1+k)\times(1+k)}$. We finally obtain the output $\mathbf{O}^h = \text{Attention}^h(\mathbf{I}) \in \mathbb{R}^{(1+k)\times m}$.
- We concatenate the H independent outputs and apply an additional linear mapping to reshape back to the original input shape: $\widetilde{\mathbf{O}} = \mathbf{W}_o \begin{bmatrix} \mathbf{O}^1 & \mathbf{O}^2 & \cdots & \mathbf{O}^H \end{bmatrix} \in \mathbb{R}^{(1+k)\times m}$.

Cell-Type Classifier

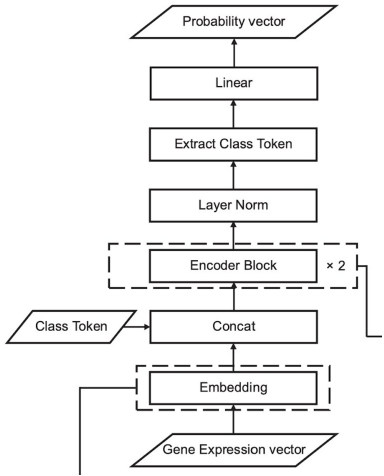
Finally, since the attention mechanism's input and output have the same shape, we extract the first row of $\widetilde{\mathbf{O}}$ (denoted as $\widetilde{\mathbf{cls}}$), which is the output of the input CLS token after the attention mechanism, as the global information extraction result for the current input.
We pass this result through a fully connected neural network to complete classification:

$$\mathbf{p} = \text{softmax}(\mathbf{W}_c\widetilde{\mathbf{cls}}) \in \mathbb{R}^C$$

where $\mathbf{W}_c \in \mathbb{R}^{C\times m}$ is the learnable classification weight matrix, and C is the number of cell types.

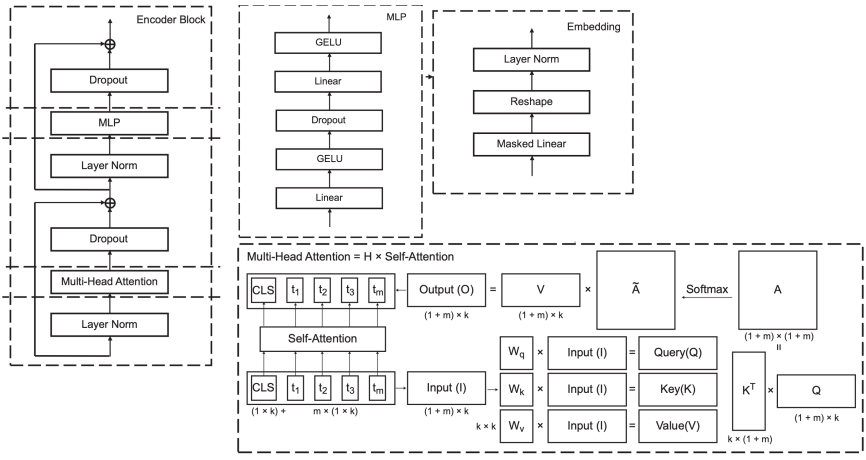
Model Architecture I

The main structure is as follows:



Model Architecture II

The detailed implementation is as follows:



Training Details

- **Data splitting:**
  - Sample splitting is performed using different studies and biological states (cross-dataset/batch splitting, rather than simple splitting from the same source).
  - 30% of the training set is further divided into a validation set.
- **Loss function and optimizer:**
  - Cross Entropy loss function is used.
  - SGD optimizer is used.
  - Cosine learning rate decay is introduced for the learning rate.
- **Training period:** Convergence within 20 epochs.

A foundation model of transcription across human cell types

Fu X, Mo S, Buendia A, Laurent AP, Shao A, Alvarez-Torres MDM, Yu T, Tan J, Su J, Sagatelian R, Ferrando AA, Ciccia A, Lan Y, Owens DM, Palomero T, Xing EP, Rabadan R

Nature, 2025

DOI: 10.1038/s41586-024-08391-z

https://www.nature.com/articles/s41586-024-08391-z.pdf

Introduction

In genearl, GET is a probalistic model aiming to predict $p(E|X, C)$ where:

- E: Gene expression levels (response variable)
- X: Regulatory feature matrix (chromatin accessibility + transcription factor binding sites)
- C: Cell type conditions (even many other conditions, e.g. different experiment methods, sequencing platforms, physiological states, etc.)

It adopts a two-stage framework: pre-training and fine-tuning.

- **Pre-training:** In pretraining, it learns the distribution pattern of regulatory features given the cell types $p(X|C)$ with unsupervised learning.
- **Fine-tuning:** Adapting the pre-trained model to specific cell types to predict $f \circ p : X \to E$.

The ultimate goal is to generalize the model's predictions across different cell types and conditions:

$$\min_{\theta} \mathbb{E}_{C^{\text{test}}} \left[\mathcal{L}(f_\theta(X^{\text{test}}), E^{\text{test}}) \right]$$

Notation I

- For a gene g in cell type c, its regulatory region window (2 Mbp long) can detect N regions/peaks, denoted as $\{r_i\}_{i=1}^{N}$ (here $N = 200$).
- For each region r_i:
  - We perform motif analysis to obtain motif scores $\boldsymbol{m}_i \in \mathbb{R}^{d_m}$, where $d_m = 282$ is the motif dimension.
  - We have chromatin accessibility information $a_i \in \mathbb{R}$, measured through scATAC-seq experiments using logCPM counts.
- We concatenate motif scores and accessibility to obtain the feature representation:

$$\boldsymbol{x}_i = \begin{bmatrix} \boldsymbol{m}_i \\ a_i \end{bmatrix} \in \mathbb{R}^d$$

where $d := d_m + 1 = 283$.

Notation II

- All N regions are encoded to obtain the input feature matrix:

$$X = \begin{bmatrix} \boldsymbol{x}_1^\top \\ \boldsymbol{x}_2^\top \\ \vdots \\ \boldsymbol{x}_N^\top \end{bmatrix} \in \mathbb{R}^{N \times d}$$

Key Architecture: Region Embedding

For each region r_i, we first transform its feature vector $\boldsymbol{x}_i$ into a higher dimensional space through linear transformation (without activation):

- Original feature matrix: $X \in \mathbb{R}^{N \times d}$
- Linear transformation: $W_{\text{Emb}} \in \mathbb{R}^{d \times D}$, where $D = 768$ in the paper
- Transformed feature matrix:

$$X' := \text{RegionEmb}(X) = XW_{\text{Emb}} \in \mathbb{R}^{N \times D}$$

Note: Since motifs in the original vectors have high correlation, nonlinear transformations are avoided early to prevent compression/interference.

Key Architecture: Token-wise Self-Attention I

To model regulatory relationships between peaks (including cis-interactions and trans-interactions), we employ a 12-layer Transformer Encoder with standard self-attention mechanism:

- Input: Region embeddings $X' \in \mathbb{R}^{N \times D}$ from previous layer
- For each attention head:
  - Linear projections to Query, Key, Value spaces:

$$Q = X'W_q \in \mathbb{R}^{N \times d_k}$$
$$K = X'W_k \in \mathbb{R}^{N \times d_k}$$
$$V = X'W_v \in \mathbb{R}^{N \times d_v}$$

where $W_q, W_k \in \mathbb{R}^{D \times d_k}$, $W_v \in \mathbb{R}^{D \times d_v}$ are learnable

Key Architecture: Token-wise Self-Attention II

- Compute scaled dot-product attention:

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

- Multi-head processing:
  - Compute multiple attention heads in parallel
  - Concatenate outputs and project through fully connected layer
  - Add residual connections and layer normalization
  - Include feed-forward layers between attention blocks

The 12-layer architecture enables modeling complex interactions between regulatory regions.

Training Procedure: Pre-training I

The pre-training phase employs a **masked autoencoding** strategy, inspired by models like BERT, to learn meaningful representations of genomic regions.

- **Masking:** For each input matrix $X \in \mathbb{R}^{N \times d}$, half of the regions are randomly selected for masking. Let $\mathcal{M}$ be the set of indices for these masked regions, where $|\mathcal{M}| = N/2$.
  - The feature vectors $\boldsymbol{x}_i$ for all masked regions ($i \in \mathcal{M}$) are replaced by a single, learnable mask token vector $\boldsymbol{m} \in \mathbb{R}^d$. This results in a masked input X^{masked}.
- **Reconstruction Task:** The model is trained to reconstruct the original feature vectors of the masked regions from the corrupted input.

The training process is as follows:

Training Procedure: Pre-training II

1. The masked input X^{masked} is passed through the Region Embedding layer to get token embeddings:

$$H^{\text{masked}} = \text{RegionEmb}(X^{\text{masked}}) \in \mathbb{R}^{N \times D}$$

2. The embeddings are processed by the Transformer Encoder to produce contextualized representations:

$$Z^{\text{masked}} = \text{Transformer}(H^{\text{masked}}) \in \mathbb{R}^{N \times D}$$

3. A linear decoder head uses the output representations $\boldsymbol{z}_i^{\text{masked}}$ to predict the original features for the masked regions:

$$\hat{\boldsymbol{x}}_i = \boldsymbol{z}_i^{\text{masked}} W_{\text{dec}} \in \mathbb{R}^d, \quad \forall i \in \mathcal{M}$$

Training Procedure: Pre-training III

4. The loss function is the reconstruction error (e.g., Mean Squared Error) between the predicted and original features for the masked regions:

$$\mathcal{L} = \sum_{i \in \mathcal{M}} \|\hat{x}_i - x_i\|_2^2$$

Abstractly, the goal is to train the encoder p and a prediction head g to minimize the expected reconstruction error over all possible data samples and masks:

$$\min_{p,g} \mathbb{E}_{X,\mathcal{M}} \left[\sum_{i \in \mathcal{M}} \left\| g(p(X^{\text{masked}}))_i - x_i \right\|^2 \right]$$

Training Procedure: Pre-training IV

The training details in this paper are as follows:

- **Optimizer:** AdamW with a weight decay of 0.05
- **Batch Size:** 256
- **Epochs:** 800, with a 40-epoch linear warmup period
- **Max Learning Rate:** 1.5×10^{-4}
- **Compute:** Trained for 7 days on 16 NVIDIA V100 GPUs

Training Procedure: Fine-tuning I

After extensive masked pre-training, we fine-tune the model for gene expression prediction tasks.

- **Task:** Predict the RNA expression level $\hat{y}$ for a gene g in cell type c, given its regulatory region features X.
- **Process:**
  1. Use pre-trained encoder to obtain latent representations:

$$H = p(X) = \begin{bmatrix} h_1^\top \\ \vdots \\ h_N^\top \end{bmatrix} \in \mathbb{R}^{N \times D}$$

  2. Apply attention pooling to aggregate region representations:

$$z = \sum_{i=1}^{N} \alpha_i h_i \in \mathbb{R}^D, \quad \text{where } \alpha_i = \frac{\exp(w^\top h_i)}{\sum_{j=1}^{N} \exp(w^\top h_j)}$$

Training Procedure: Fine-tuning II

3. Pass pooled representation through an MLP:

$$\hat{y} = f_\phi(z)$$

- **Optimization objective:**

$$\min_{\phi,\theta} \sum_{(g,c)} \mathcal{L}_{\text{expr}} \left(f_\phi(p_\theta(X_{g,c})), y_{g,c} \right)$$

where $\mathcal{L}_{\text{expr}}$ is Poisson Negative Log Likelihood loss for count data.

Note: In implementation, this paper adopts LoRA (Hu et al., 2021) for efficient parameter fine-tuning.

VCC Baseline and Cell-eval

https://github.com/ArcInstitute/cell-eval