Bocconi

Bachelor of Science in Economics, Management and Computer Science

Exploring Transcriptional Interactions and Language Models for Single Cell RNA-seq Masked Value Prediction

Advisor:
Prof. Francesca Buffa

Bachelor of Science thesis by:
GIACOMO CIRÒ
student ID no. 3160499

Academic Year 2023-2024

Bocconi
Università Commerciale
Luigi Bocconi

To Caterina, Tommaso, Matilde, Matteo, Davide, Clizia, Marco, Franco, Tina, Rocco for being part, or having been part, of my everyday normal life.

A special thanks to Professor Francesca Buffa for her support throughout the realization of this Final Paper and to Professor Dirk Hovy for his encouragement to pursue my interests.

Exploring Transcriptional Interactions and Language Models for Single Cell RNA-seq Masked Value Prediction

Giacomo Cirò

Abstract

Large Language Models (LLMs) have exhibited astonishing capabilities in Language Processing, and whether their applications can go beyond this domain is a natural question to ask.

The recently open-sourced Single Cell GPT (scGPT) model showcases the potential of LLMs in Biology. This encoder-only Transformer trained on Single Cell RNA-sequencing (scRNA-seq) data from over 33 millions human cells demonstrated remarkable performance in gene expression value prediction and cell modelling tasks.

However, it does not exploit the huge amount of available information about the complex biological processes which regulate gene expression patterns at cellular level.

In this research, we focus on transcriptional regulation mediated by transcription factors (TF) and investigate how prior biological knowledge can be incorporated into scGPT's framework. We are aware that inferring the activity of a specific TF solely based on scRNA-seq data is not trivial at all. In fact, abundance of a TF-coding gene in scRNA-seq does not always correlate with that TF being active, due to further regulation at later stages.

To account for this major concern, we test the model's performance in a best-case scenario and in a general-case scenario.

For the first one, we consider human breast cancer cells which have been subjected to either hypoxic or normoxic state, where the high number of active regulatory processes might mitigate the aforementioned issue. For the second one, we consider the immune compartment of normal human breast cells.

The results show that the proposed approach improves predictive accuracy in the first case, while no substantial improvement is observed in the second one. Furthermore, performance is enhanced by restricting the number of considered TFs to those whose corresponding gene expression is more likely to correlate with activity.

These findings suggest that the effectiveness of incorporating prior biological knowledge is context-dependent and motivates future research directions to focus both on how to include additional information and especially on determining when it makes sense to do so.

Contents

Introduction

Large Language Models (LLMs) have achieved impressive results in the near past and the fast pace at which they are developing suggests that more is expected to come in the future. The huge amount of text data available in today's society, which is ultimately a virtually infinite source of knowledge, has been successfully exploited to make machines promising to become almost comparable to humans on many language tasks and benchmarks. It is natural to wonder whether these astonishing capabilities can be generalized: are these results restricted to language? Or can these models somehow be exploited to address challenges in different domains as well?

It turns out, the answer is of course they can. Initially developed for machine translation, the Transformer architecture [28] proved to be the state-of-the-art way of processing long sequences of data in general, effectively capturing intricate relationships and temporal nuances. Even more, it showed to be effective for tasks which apparently do not involve the processing of sequences, provided you can somehow cast the analyzed object to a sequence-like, a series of tokens to use the correct terminology.

For example, the Vision Transformer [9] extends the state-of-the-art performance of the original Text Transformer to Image Processing by cleverly splitting a single image into fragments, creating a sequence of patches which are then treated as words in a sentence.

More recently, Transformers have also been trained to efficiently deal with complex planning tasks, solving Sokoban puzzles in fewer steps than an A* algorithm [18], which is considered state-of-the-art when it comes to pathfinding.

Even when dealing with tabular data, which is the furthest you can find from sequences, an adapted Transformer model called TabTransformer [15] proved to be comparable to tree-based ensemble models across a diverse range of datasets.

Going back to sequences in the strict sense, it is natural to also wonder whether these models can be used to answer questions not even human experts can. Once again, the answer is positive and the Transformer architecture has been successfully employed to understand one of the most important sequences we can think of, that is, proteins.

Proteins are the building blocks of life, fundamental molecules which play critical roles in virtually all biological processes within living organisms. Proteins are ultimately just a chain of elementary residues. These are called amino acids, there are 20 of them and when combined they fold into complex 3D structures that determine the final function of the protein they compose, from catalyzing biochemical reactions as enzymes to providing structural support in cells and tissues. However, the link between mere amino acid sequence and 3D structure is not at all trivial, and understanding folding dynamics plays a crucial role in understanding the protein itself. This involves using complex and costly methods which take time, for example X-ray Crystallography. In 2020, Alpha Fold 2, a cutting-edge deep learning architecture developed by Google Deep-mind [17], used Transformer blocks to analyze millions of proteins and accurately predict their structure, achieving unprecedented results. This successful attempt showed the potential of Machine Learning models for drug discovery and disease treatment.

Two other extremely important biological sequences are linked to proteins through the so-called Central Dogma of Molecular Biology, which states that genetic information flows from DNA, to RNA, to protein. DNA and RNA are fundamental molecules that carry genetic information in living organisms and consist of long sequences of elementary residues, called nucleotides.

Thanks to recent advancements in genomics and high-throughput technologies, researchers have now the possibility to access and analyze huge amounts of DNA or RNA sequencing data, i.e., information about the nucleotides composing these molecules and their abundance within cells. Deep Learning have been effectively used to to analyze this type of data.

For example, in 2023, a self-supervised foundation model trained on thousands genomes at nucleotide level resolution, called Nucleotide Transformer, was introduced [7]. The model was pre-trained to predict the next nucleotide in a sequence, and further fine-tuned to outperform a vast majority of baseline models on nucleotide level tasks, such as promoter detection or weak/strong enhancer classification.

When it comes to RNA, we need to distinguish between two types of sequencing. Traditional Bulk RNA sequencing (RNA-seq) measures the average gene expression from an entire population of cells, treated as a whole. In contrast, Single Cell RNA Sequencing (scRNA-seq) allows the examination of gene expression patterns in individual cells, which is not visible through bulk measurements. Machine Learning techniques come handy when dealing with this type of

data. For example, t-SNE [19] and UMAP [20] are used for dimensionality reduction, to successfully isolate semantically relevant clusters of cells, solely based on RNA expression levels. Moreover, Machine Learning models can be trained on scRNA-seq data to perform cell type annotation or gene regulatory network inference.

A few months ago, a first attempt to build a foundation model for single cell multi-omics data was published [6]. The model, dubbed scGPT, is an encoder-only Transformer trained on scRNA-seq data from over 33 millions human cells via Masked Language Modelling (MLM). The authors show that, after appropriate fine-tuning, the model is able to achieve remarkable performance and is comparable to different state-of-the-art methods on a diverse range of tasks. However, they acknowledge this is just an initial attempt towards a foundation model for multi-omics data, and improvements are still to be made.

Literature shows that, in order to achieve significant results in domain-specific tasks, it is of paramount importance to exploit all possible prior knowledge. For example, see how Alpha Fold 2 incorporates Multiple Sequence Alignment (MSA) and existing 3D templates for protein structure prediction. Clearly, the same path has to be followed in order to achieve analogous results for multi-omics data.

In this paper, we explore one direction in which scGPT can be improved by incorporating prior knowledge into the model. In particular, we want to investigate whether injecting information about the complex regulatory processes that govern gene expression within a cell are useful for predicting gene expression values. Specifically, we consider the activity of transcription factors (TF) and try to answer the following research question:

RQ1 Does including prior knowledge about transcriptional interactions improve model performance in gene expression masked value prediction?

We argue the answer is positive and we support our thesis with experimental results. The main rationale behind this hypothesis is pretty intuitive: if we know gene A is expressed and that it also stimulates the expression of gene B, it makes sense to consider this information when inferring the latter expression value.

However, inferring gene regulatory networks and TF activity solely based on RNA abundance

is not a trivial task at all, and this consideration is what guided the choice of the data used. In particular, we conduct our experiment on two distinct datasets. One best case scenario, where a specific perturbation causes the activation of many regulatory processes hence the connection between RNA abundance and TF activity can be assumed to hold. The other, instead, is a general case scenario where such link might not be truthful and our proposed solution perhaps not relevant.

In order to test our hypothesis we build upon the open source scGPT model and incorporate information relative to transcriptional interactions into the model's input. Then, we proceed to pre-train two smaller-scale versions of the model from scratch.

Due to resource constraints, our experiments and results serve only as proof of concept to stimulate future work.

To the best of our knowledge, this idea has not been tested yet and our contributions can be summarized as follows:

- Assess whether information relative to transcriptional interactions can be useful for the specific task of gene expression value prediction;
- Determine if adding more information in the input is beneficial because injecting prior knowledge, or detrimental due to increased noise;
- Provide insights about the generalizability of this approach across different datasets and biological contexts.

In the first part of this dissertation, we present some background knowledge which will be fundamental to understand our experiment. We initially discuss Natural Language Processing as a sub-field of Machine Learning, the tasks it tries to solve and the solutions literature proposes. We then proceed with an overview of some aspects of Life Sciences, with a particular focus on Biology and Transcriptomics, explaining what scRNA-seq data is, how it is obtained, and what are transcriptional interactions. Next, we provide a detailed description of scGPT, its architecture and how it was trained. In the second part, we introduce our modified version of the model, focusing on where we deviate and why. Finally, we discuss the results obtained and suggest future directions we would like to pursue.

Background

This paper places itself at the intersection of two apparently independent fields: Natural Language Processing and Life Sciences. In this section, we introduce them separately and then show how they can actually be extremely intertwined, through the analysis of a recently published work.

2.1 Natural Language Processing

In 1950, Alan Turing proposed the now well-established Turing Test (Imitation Game) as a way of measuring artificial intelligence [27]. In particular, he wanted to measure the ability of an artificial agent to exhibit human-like behaviors. A machine is said to possess artificial intelligence in a specific domain if an external evaluator is not able to discriminate between what is produced by a machine and what by a human. In particular, he analyzed the case of question answering and explained that what matters for a machine to pass this test is not the correctness of what it produces in itself, rather the similarity to what a human would have produced. Turing's need to develop this test is evidence that already in the 50s people were foreseeing the potential of machine intelligence and were anticipating what nowadays we have as state-of-the-art.

As the name suggests, Natural Language Processing (NLP) is a branch of Machine Learning which employs computational techniques for the purpose of learning, understanding, and producing human language content [14].

NLP initial development is dated after World War II, with the goal of automating translation between languages. Interestingly, throughout all its history, the greatest advancements and milestones of NLP were indeed introduced to improve Machine Translation tasks. Nowadays, it has become a very broad field, with different areas focusing on different aspects of natural language.

NLU Natural Language Understanding (NLU) is a sub-field of NLP whose focus is the comprehension and interpretation of human language to grasp the meaning, context, and intent behind natural language. The following are common tasks in NLU:

- *Part-of-Speech (PoS) Tagging*, identify and classify different words into the corresponding grammatical category (verb, noun, adjective etc.);
- *Named Entity Recognition (NER)*, identifying and classifying key elements within text, such as names of people, organizations, locations, etc.;
- *Sentiment Analysis*, determining the emotional tone behind a body of text.

NLG Natural Language Generation (NLG) is a sub-field of NLP whose focus is the process by which a computer generates human-like text based on some input data, commonly referred to as "prompt". The following are common tasks in NLG:

- *Text Summarization*, generate a brief summary from a longer text;
- *Machine Translation*, translate a sentence into another language;
- *Question Answering*, produce the correct answer to a question.

In general, NLP deals with natural language, whose main abstraction are words, hence inherently different from the binary language of computers. To be able to analyze it, it is fundamental to cast text and words into some sort of representation which involves numbers, and can be processed by a computer. Even before that, it is important to define if words are actually the elementary unit of analysis, perhaps shall we consider an entire sentence? An entire document? Or something else?

In the next sections we introduce some techniques which outline the development of NLP and can help answer these questions. Afterwards, we introduce how neural networks can be employed for Language Modeling and, in particular, the state-of-the-art Attention Mechanism and Transformer architecture.

Before diving into, let's introduce some terminology. We refer to a piece of text with the term "document" (e.g. Obama's 2008 presidential speech). We refer to a collection of documents with the term "corpus" (e.g. all presidential speeches from the last 10 years).

2.1.1 Tokenization

When analyzing a document, it is naive to treat the document as a whole. It makes more sense to break it down into smaller portions (e.g. words, sentences, paragraphs), and analyze how these are related to one another and to the whole.

Tokenization refers to the standardized process of breaking down documents into elementary units called *tokens*, which oftentimes are words, but can also be any other meaningful element, such as letters or sub-words. The collection of all possible tokens is called *vocabulary* and the total number of possible tokens is referred to as *vocabulary size*.

Sub-word Tokenization Nowadays, state-of-the-art LLMs use sub-word tokenization [24]. The specific tokens to use are most of the times defined using an iterative algorithm called Byte-Pair Encoding (BPE), which minimizes the number of tokens required to encode a given corpus. It works as follows:

A desired vocabulary size is set as a hyperparameter. A starting vocabulary is initialized containing all Unicode symbols occurring in the corpus. At each iteration, a temporary list is created with all possible combination of tokens already in the vocabulary. Then, the most frequent among these is added to the vocabulary. The process is repeated until the desired vocabulary size is reached.

E.g. consider the corpus comprising one single document of one string "aaaabaaabb" and target vocabulary size of 3. Initial vocabulary is $\{a, b\}$. At iteration 1, possible new tokens are $\{aa, bb, ab, ba\}$. The most frequent is aa. New vocabulary is $\{a, b, aa\}$. Target size is reached, algorithm stops.

2.1.2 Document-Term Matrix

One of the main goals of NLP is to find an effective way to convert a piece of unstructured text, called corpus, to numbers which could then be analyzed by a computer.

Bag-of-Words In 1954, Harris introduces the concept of Bag-of-Words as an intuitive method to analyze a corpus by simply counting how many times each word appears [13]. These word counts can be compactly visualized using the Document-Term Matrix (DTM), with unique

words on the columns and documents on the rows, as shown in Table 2.1. Even though effective for some elementary tasks, this method has some major pitfalls which cannot be underestimated. For example, it does not account for the order in which words appear or their relative importance for understanding the meaning of a piece of text. For example, consider the word "are" and the word "war". The former is more likely to appear, but clearly less informative than the latter when extrapolating the meaning of a document. This is something which must be taken into consideration.

	term1	**term2**	**term3**
doc1	2	1	0
doc2	0	3	1
doc3	1	0	2
doc4	0	2	3

Table 2.1: Example of a Document-Term Matrix

TF-IDF In 1972, Jones introduces the Term Frequency Inverse Document Frequency (TF-IDF) method as a finer way to construct the DTM by counting occurrences while adjusting for how much a word is informative [16]. The specificity of a term, she argues, can be quantified as an inverse function of the number of documents in which it occurs. Given a set D of documents, each entry in the DTM is computed as follows:

$$TF_{i,j} = \frac{n_{i,j}}{|d_j|}$$
$$IDF_i = log\left(\frac{|D|}{|\{d : i \in d\}|}\right)$$
$$TFIDF_{i,j} = TF_{i,j} \cdot IDF_i$$

Where $n_{i,j}$ is the number of times word i occurs in document j, $|d_j|$ is the number of words in document j, $|D|$ is the total number of documents and $|\{d : i \in d\}|$ is the number of documents containing word i. The $TFIDF_{i,j}$ score is higher if word i occurs many times in document j and not as much in all the others, suggesting this word shapes the content of document j more.

2.1.3 Word Vectors

Even more interesting, it is to find a meaningful way of representing words as sequences of numbers, i.e. vectors, instead of sequences of letters. In doing so, it is useful to require that

semantic relations between words are maintained as mathematical relations between their corresponding vectors. For example, consider the words "dog","cat" and "stadium". It makes sense that the numerical representations for "dog" and "cat" are somehow more similar to each other than to the one for "stadium".

Cosine Similarity The notion of similarity between two vectors v, w is measured using the cosine similarity score. The formula is the following:

$$S_C(v, w) := cos(\theta) = \frac{w \cdot v}{||v|| \cdot ||w||}$$

$\in [-1, 1]$. A score closer to 1 (-1) indicates the vectors are more similar (different).

Word2Vec In 2013, Word2Vec is introduced as an efficient way of estimating vector representations of words [21]. Traditionally, shallow neural networks have been used to encode inputs to a vector space [23]. The idea of the authors is similar and they propose to use neural networks to project words into a multi-dimensional representation. They introduce two variations of the same model, namely *Continuous Bag-of-Words (CBOW)* and *Skip-Gram*. Both involve the use of a neural network made of a single hidden layer (Figure 2.1). The difference lies in what is the input and what is the output. For *CBOW*, a context of surrounding words is used to predict the one whose representation we want to estimate. Instead, *Skip-Gram* employs a single word as input to predict the context of surrounding ones.

The representations obtained from both these models are shown to retain meaningful semantic relations as mathematical relations, and simple arithmetic vector operations make sense from a linguistic perspective. Denote $v(w)$ the vector representation of word w obtained via either *CBOW* or *Skip-Gram*. Then these are only two of the possible examples:

$$v(Queen) \sim v(King) - v(Man) + v(Woman)$$
$$v(Paris) \sim v(Rome) - v(Italy) + v(France)$$

Where the $\sim$ refers to high cosine similarity between word vectors.

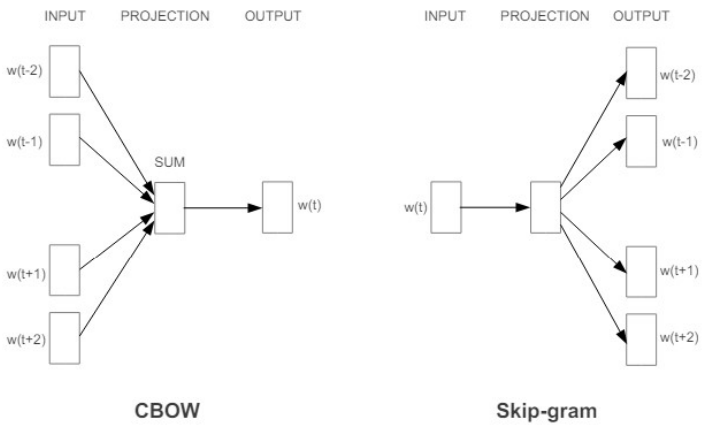

Figure 2.1: The two variants of the Word2Vec model [21]

2.1.4 Language Modelling

The art of speaking, when stripped of its subjective and emotional connotation, can be reduced to a simple sequential generation of words, where each subsequent word is conditionally dependent on all previous ones. A goal of NLP, in particular NLG, is exactly that of learning the joint probability distribution of sequences of words in a language. We distinguish two main tasks:

- *Causal Language Modelling* (*CLM*), predict next word given a sequence of known words;
- *Masked Language Modelling* (*MLM*), predict a set of hidden words within a sequence of known words.

The two tasks differ from a temporal point of view. CLM focus on predicting future words given all preceding ones, whereas MLM operates on a static sequence where some words are hidden, and has to predict them. Typically, CLM and MLM operate at token-level instead of considering words.

N-gram Modelling Let x be any token in a dictionary. A sequence $X = \{x_1, ..., x_n\}$ is referred to as a n-gram. Elementary CLM estimates the conditional probability distribution of any token y_j following n-gram X_i as follows:

$$p(y_j|X_i) = \frac{count(concat(X_i, y_j))}{count(X_i)}$$

Where $count(X)$ is the number of times X appears in a given corpus and $concat(X, y)$ is the sequence obtained by appending y as the last element of X. Limitations of this model

arise when it's tested on sequences not seen during training, for which a relative count cannot be estimated. Additionally, trying to represent and analyze all possible sequences of tokens appearing in a corpus is clearly unfeasible and not optimal.

Neural Modelling In 2003, this problem is tackled by introducing for the first time an effective way of modeling language using neural networks, moving away from the discrete n-grams model and towards a continuous one [3]. The idea is to train a neural network to simultaneously learn a continuous representation of sequences of tokens and a joint probability function expressed in terms of this continuous representation, instead of the tokens themselves. In this way, generalization is obtained because a sequence gets higher probability if it is made of tokens similar to the ones seen during training, in the sense that they have similar continuous representation. The strength of this method is that it can interpolate training sequences to estimate the representation of unseen ones. Consider the following example:

- *Training*: {"my friend Laura is funny", "my friend Marco is funny"}
- *Test*: {"my friend Luca is funny"}

A 4-gram model with tokenization at word-level would assign the same probability to any word coming after "my friend Luca is", as the specific sentence "my friend Luca is funny" has never been seen during training. However, an educated guess would be to predict "funny" as the next word in the sequence, given the similarity with the training samples. A neural network, instead, would have learned a continuous representation and all three sentences would be similar, as they differ only for a proper noun. Thus, the probability distribution for the next word after the test sentence would be conditioned on an input representation similar to the ones seen during training and it would assign higher probability to the word "funny".

2.1.5 Encoder-Decoder Architecture

Many NLP tasks often deal with sequence-to-sequence (seq2seq) problems, that is mapping an input sequence to a correct target sequence. Consider Machine Translation: the goal is to map sentence in one language to the corresponding sentence in another language.

In 2014, a general end-to-end approach to deal with this kind of problems is introduced: the

encoder-decoder architecture [5] [25]. The general idea is to use a two-step approach. First, an input sequence $X = (x_1, \ldots, x_T)$ is mapped by the encoder to a latent representation c, called context vector, using some nonlinear function. Then, the decoder sequentially predicts each symbol[1] in the target sequence $Y = (y_1, ..., y_m)$ given the context vector and all symbols it has already predicted. The decoder is auto-regressive in the sense that it consumes the previously generated symbols as additional input when generating the next one. From a statistical point of view, the decoder models the following conditional probability:

$$p(y_t|y_1, ..., y_{t-1}, c)$$

Where y_t is a symbol in the output sequence.

Usually, both the encoder and the decoder are neural networks trained jointly to maximize the conditional probability of the correct output sequence given an input sequence. The use of either simple Recurrent Neural Networks (RNNs) [5] or Long-Short Term Memory Networks (LSTMs)[2] [25] showed impressive results, and the encoder-decoder approach is now the status-quo for seq2seq learning.

2.1.6 Attention Mechanism

In 2014, the first Attention Mechanism is introduced as a finer way of modeling temporal dependencies in Machine Translation using encoder-decoder RNNs [2].

Instead of having a fixed context vector c for each symbol y_t to predict, this method allows to define a new context vector c_t conditional on the decoder's current state s_{t-1}, which is itself a function of the symbols $\{y_1, ..., y_{t-1}\}$ predicted so far. Therefore, this approach allows the input to be encoded differently according to what's relevant for the current prediction. The decoder

[1] We now use the term symbol because we are considering a general notion of sequences, which may contain words, tokens, vectors or any other element. For example, each symbol can be a multi-dimensional embedding of an input token, thus the input sequence becomes a sequence of vectors.

[2] RNNs are Deep Learning models developed specifically for inputs which come as sequences. The idea is to have a modular model which can be repeated as many times as needed, to process each symbol sequentially without having a fixed input length. However, one of the biggest pitfalls of RNNs is that they are not able to account for long term dependencies, with the output conditioned mainly on immediately preceding symbols. LSTMs were introduced to tackle this problem by introducing an additional path to allow for more information to flow and ideally separate long and short term dependencies.

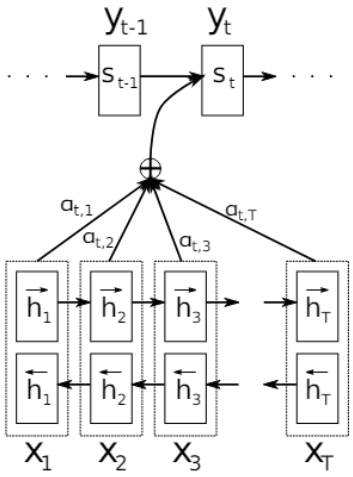

Figure 2.2: The original Attention Mechanism [2]

now models the following probability density function:

$$p(y_t|y_1, ..., y_{t-1}, c_t)$$

Given an input sequence $X = (x_1, \ldots, x_T)$, the encoder creates for each input symbol x_i an annotation h_i by concatenating the hidden states from two RNNs, running in opposite directions, as shown in Figure 2.2. The first one processes the input from left to right, the second one from right to left. Such approach allows the annotation h_i to contain information relative to both the preceding and following symbols of each input x_i. When predicting a new symbol y_t, a new context vector c_t is defined as a weighted sum of the h_i, where the weights are themselves a function of both h_i and the hidden state s_{t-1} of the decoder. The latter characterization is what allows the model to adapt the encoding based on what it's currently predicting, without restricting to a static representation of the input.

$$c_t = \sum_{i=1}^{T} \alpha_{ti} h_i \qquad \alpha_{ti} = \frac{exp(e_{ti})}{\sum_{i=1}^{T} exp(e_{ti})} \qquad e_{ti} = a(s_{t-1}, h_t)$$

In particular, $a(\cdot, \cdot)$ is parameterized as a Feed Forward Neural Network (FFNN) jointly trained with the other components of the model.

2.1.7 Scaled Dot-Product Attention

A few years later, a new Attention Mechanism is presented, dubbed Scaled Dot-Product Attention (SDPA) [28]. The idea is still that of allowing a dynamic input embedding, aware of

the context and the current prediction at hand, but through a process which involves matrix projections.

Given a sequence $X = (x_1, \ldots, x_T)$ of vectors, the first step is to map each x_i into three distinct vector spaces using three distinct projection matrices, obtaining the so-called queries (Q), keys (K) and values (V):

$$Q = XW^Q = (q_1, \ldots, q_T)$$
$$K = XW^K = (k_1, .., k_T)$$
$$V = XW^V = (v_1, .., v_T)$$

Where $Q \in \mathbb{R}^{d_{model} \times d_q}, K \in \mathbb{R}^{d_{model} \times d_k}, V \in \mathbb{R}^{d_{model} \times d_v}$ and each $q_i \in \mathbb{R}^{d_q}, k_i \in \mathbb{R}^{d_k}, v_i \in \mathbb{R}^{d_v}$ is a vector corresponding to one symbol in the input sequence. Then, the following is obtained:

$$softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)$$

$\in \mathbb{R}^{T \times T}$. Which is commonly referred to as the Attention Matrix. Each entry i, j is obtained by normalizing the dot product $\frac{q_i \cdot k_j^T}{\sqrt{d_k}}$, via softmax. Loosely speaking, each entry measures how close the query q_j is to the key k_i. The final attention score is computed as follows:

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

$\in \mathbb{R}^{T \times d_k}$. The attention score for an input symbol x_i, i.e. row i of the above matrix, is simply a weighted average of the values v_j of all the other symbols x_j, where the weights for each v_j is the entry i, j of the attention matrix above.

This process can be repeated multiple times in parallel via multiple so-called attention heads. To achieve this, the original Q, K, V are further projected using new projection matrices W_h^Q, W_h^K, W_h^V, and for each head h attention is computed on these newly obtained queries, keys and values. Finally, all the attention scores from different heads are concatenated and projected back to a

lower dimensional space, obtaining the multi-head attention:

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, ..., head_H)W^O \\ head_h &= Attention(Q_h, K_h, V_h) \\ &= Attention(QW_h^Q, KW_h^K, VW_h^V) \end{aligned}$$

Where H is the total number of heads and $W^O \in \mathbb{R}^{(H \cdot d_k) \times d_k}$. Intuitively, this approach of using multiple heads and stacking the results together allows each head to specialize in analyzing different aspects of the input sequence, giving the model increased performance and flexibility.

According to what is used to compute the Q, K, V, we distinguish:

- Self-Attention, Q, K, V are all computed from the same sequence;
- Cross-Attention, Q comes from a sequence and K, V from another one.

The former allows to capture relations among symbols within a given sequence, while the latter allows symbols from multiple sequences to interact with each other.

Moreover, the attention matrix can be masked in order to prevent specific pairs of symbols to communicate.

2.1.8 Transformer Architecture

In 2017, a groundbreaking deep learning architecture, dubbed the Transformer, was introduced [28]. In a few years, it was widely adapted as the state-of-the-art for seq2seq modelling, crushing previous solutions based on recurrent networks.

The Transformer is an encoder-decoder model (Figure 2.3) which solely relies on Feed Forward Neural Networks and Scaled Dot-Product Attention to analyze sequences by processing each element in parallel. It does not require any sort of recurrence anymore, allowing for extreme parallelization of computations and increased performance.

The main idea behind both the encoder and the decoder is to use a first layer of communication, which employs SDPA, in conjunction with a second layer of computation, a shallow FFNN,

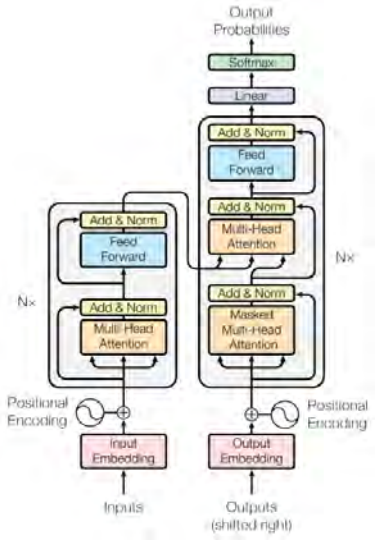

Figure 2.3: The Transformer architecture [28]

to project an input sequence of vectors into another, context-aware, sequence of vectors. Intuitively, the input sequence flows through the attention layer enabling each symbol to influence the representation of the others. Then, it is analyzed by a FFNN which yields a final representation. This dual procedure is repeated N times in the N stacked encoder layers. Afterwards, the obtained representation is passed to the decoder, where the same process is repeated. Finally, a linear layer is used to map the processed sequence to the output required for the task at hand. In the following paragraphs we analyze some important aspects in more detail.

Input Embedding The input of the model, which originally was text, is tokenized using $vocab\_size$ possible tokens. Then, to obtain a vector representation of each token, the idea is the same as in the original Word2Vec model (section 2.1.3), but with a minor adjustment. Now, the embedding layer is trained jointly with the rest of the model, not just to predict the surrounding context of words, but to optimize the final task.

Positional Encoding Given the non-recurrent structure of this model, yet the sequential nature of the input, it is important to inject a notion of the position of each symbol within the input sequence. This is done in the Positional Encoding layer. In the original paper, the authors use sine and cosine functions of different frequencies to determine unique position vectors which are then included to the input embedding via simple vector addition. More recent methods, instead, rotate each token embedding based on their position, for example via multiplication with complex numbers [4].

Self-Attention Self-Attention is employed when the goal is to obtain a meaningful representation of the input sequence. Hence, it's used in the attention layers of each encoder layer and in the first attention layer of each decoder layer, where the input is the sequence produced so far.

Cross-Attention Cross-Attention, instead, is used when the goal is to extrapolate meaningful relations between two sequences. Hence, it's used in the second attention layer of each decoder layer, with Q coming from the auto-regressive input and K, V from the output of the encoder.

For many natural language tasks you don't actually need both the encoder and the decoder: a simplified transformer model with just one of the two turns out to be the best choice. The Bidirectional Encoder Representations from Transformers (BERT) and the Generative Pre-trained Transformer (GPT) are an example of encoder-only and a decoder-only model which achieve state-of-the-art performance on MLM and CLM, respectively [8][4].

In the original paper, the authors set:

$$\begin{aligned} N &= 12 \\ H &= 8 \\ d_{model} &= 512 \\ d_q = d_k = d_v &= \frac{d_{model}}{H} = 64 \\ d_{ff} &= 4 \cdot d_{model} = 2048 \\ vocab\_size &= 37,000 \end{aligned}$$

The FFNN is made of a single layer of dimension d_{ff}.

2.1.9 Transfer Learning

Pre-training The term pre-training refers to the training of a large-scale machine learning model on a broad and diverse dataset. The resulting model is called a foundation model, and provide a general-purpose foundation upon which more specific applications can be built. For example, GPT-3-175B is a 175 billions parameters decoder-only transformer trained on over 450 billions sub-word tokens via CLM [4]. Similarly, BERT-LARGE is a 340 millions param-

eters encoder-only transformer trained on over 3 billions words via MLM [8].

Fine-tuning The term fine-tuning refers to the act of further training a foundation model on a dataset specific for a downstream task. For example, GPT-3 can be further trained on a specific dataset of questions and answers to make the model able to perform the question-answering task correctly. Similarly, BERT-LARGE can be further trained on a specific dataset of text with labels for sentiment analysis.

This framework of transferring ”knowledge” from a foundation model to a fine-tuned version is referred to as Transfer Learning.

2.2 Life Sciences

Life Sciences is an umbrella term which encompasses all the fields of Science focusing on the study of life, such as microorganism, plants and animals, including human beings. Some examples are:

- *Biology*, the study of living organisms, their structure, function, growth, evolution, and interactions;
- *Biochemistry*, the study of chemical processes related to living organisms;
- *Molecular Biology*, the study of biological processes at the molecular level;
- *Neuroscience*, the study of the nervous system;
- *Bio-informatics*, the development of methods and software tools for storing, retrieving, organizing and analyzing biological data.
- *Genetics*, the study of heredity in living organisms. It explores how traits and characteristics are passed from one generation to the next through genes, their molecular structure and function.

2.2.1 Omics

The suffix "-omics" is used to form nouns meaning "a study of the totality of something". The subject matter of study is defined with the suffix "-ome", which indicates the complete "whole of class".

The branches of biology commonly known as omics fields study various aspects of organisms on a large scale, using high-throughput technologies to generate comprehensive datasets, aiming at the collective characterization of pools of biological molecules. Each omics field focuses on a different component or level of biological information. Major examples are:

- *Genomics*, the study of *genomes*, that is the complete set of genes, genetic material, and non-coding regions of an organism's DNA;
- *Transcriptomics*, the study of *transcriptomes*, that is all RNA molecules produced by the genome which determine the gene expression patterns and regulatory mechanisms at the RNA level;
- *Proteomics*, the study of *proteomes*, that is the entire set of proteins produced by a cell or organism.
- *Metabolomics*, the study of the *metabolomes*, that is all small molecules (metabolites) present in a biological sample, which are intermediates and end products of metabolic pathways;
- *Epigenomics*, the study of *epigenomes*, that is heritable changes in gene expression that do not involve alterations to the underlying DNA sequence.

2.2.2 DNA

The Deoxyribonucleic Acid (DNA) is the molecule that carries the genetic instructions for the development, functioning, growth, and reproduction of living organisms. It is the fundamental building block of life, encoding the information necessary for the synthesis of proteins and the regulation of cellular activities.

It consists of two long strands, which are twisted around each other forming a double helix

(Figure 2.4). Each strand is made up of smaller units called nucleotides. Each nucleotide consists of three components:

- *Phosphate Group*, made of one phosphorus atom bonded to four oxygen atoms;
- *Deoxyribose Sugar*, a five-carbon sugar molecule derived from the ribose sugar by loss of a hydroxy group;
- *Nitrogenous Base*, different molecules containing nitrogen. They can be of four types, namely *adenine* (A), *thymine* (T), *cytosine* (C), *guanine* (G).

The two strands are held together by hydrogen bonds between pairs of compatible nitrogenous bases. In particular, adenine pairs with thymine and cytosine pairs with guanine.

In order to carry out its function, DNA must express its information and guide the synthesis of other molecules in the cell. This process is the same in all living organisms and it is described by the so-called Central Dogma of Molecular Biology: genetic information flows in one direction only, from DNA, to RNA, and finally to proteins.

We refer to segments of DNA that contain the coded instructions for synthesizing a specific protein or a set of proteins as genes. Within genes, we can further distinguish two regions: exons and introns. Exons are portions of genes that are transcribed into RNA and contain the information that is translated into protein, i.e. they are coding sequences. Introns, instead, are portion of genes that are transcribed into RNA, but are not translated into protein, i.e. they are non-coding sequences and play other crucial roles such as regulation of gene expression.

Figure 2.4: The DNA structure [1]

2.2.3 RNA

During transcription, segments of the DNA sequence are used as templates for the synthesis of a shorter molecule, called Ribonucleic Acid (RNA). Its structure (Figure 2.5) is the same as that of DNA, but for the following:

- *Ribose Sugar*, instead of Deoxyribose Sugar;
- *Uracil* (U), instead of Thymine;
- *Single Stranded*, instead of Double Stranded

To be precise, this kind of RNA just described is specifically called Messenger RNA (mRNA), which serves as the essential intermediary in the flow of genetic information from DNA to protein. In addition to mRNA, several other types of RNA molecules play important roles in various cellular processes, such as gene regulation, protein synthesis, RNA processing and other functions. However, their understanding is not required for the purpose of this dissertation.

The complementary DNA (cDNA) is a synthesized DNA molecule that is complementary to a specific RNA molecule, typically mRNA. cDNA is generated through a process called reverse transcription, which converts RNA back into DNA. Unlike genomic DNA, cDNA contains only exons.

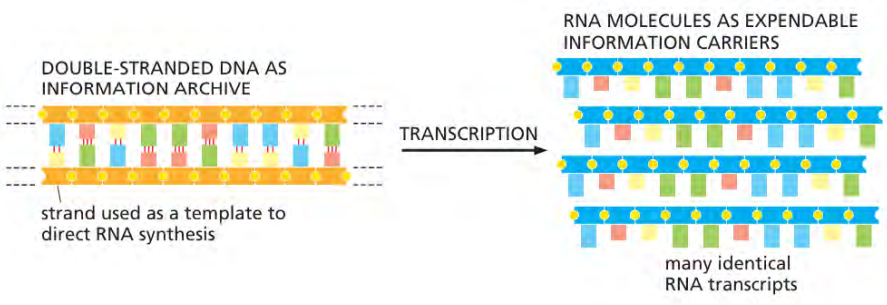

Figure 2.5: DNA Transcription and the RNA structure [1]

2.2.4 Single Cell RNA-sequencing

Sequencing data refers to the output obtained from various sequencing technologies that determine the precise order of residues in biological molecules which come as a sequence, such as

DNA or RNA. This data is fundamental in genomics, transcriptomics, and other omics fields, enabling researchers to decode genetic information and study its implications in health, disease, and evolution.

RNA-sequencing (RNA-seq) is a genomic approach for the detection and quantitative analysis of mRNA molecules in a biological sample. Since these molecules exist in a very small quantity in an individual cell, ensembles of thousands are used in a process called bulk RNA-sequencing (bulk RNA-seq). However, the averaging that occurs with this approach does not allow detailed assessment of individual expression patterns, which is fundamental for heterogeneity analysis and studying gene regulatory networks [12].

In 2009, the first Single Cell RNA-sequencing (scRNA-seq) study was published [26], enabling researchers to describe RNA molecules in individual cells with high resolution and on a genomic scale. In the following years, many different scRNA-seq protocols have been developed. They all share the main steps which can be summarized as follows [12]:

1. *Isolation*, of single cells from a tissue;
2. *Cell Lysis*, to capture as many mRNA molecules as possible;
3. *Reverse Transcription*, to obtain cDNA, which is more stable and easier to work with;
4. *Amplification*, of cDNA through either PCR or in vitro transcription and reverse-transcription;
5. *Barcode-tagging*, to preserve information on cellular origin;
6. *Sequencing*, using next generation sequencing techniques (NGS), similar to those traditionally used for bulk samples or other sequencing experiments.

The last step is crucial for actually measuring how many times a mRNA molecule coding for a specific gene appears in a cell, which is referred to as the number of reads and measures the expression of a specific gene. The data obtained can be re-conducted to the so-called cell-by-gene Matrix $X \in \mathbb{R}^{N \times G}$. Each entry x_{ij} refers to the number of reads for gene $j = 1, ..., G$ in cell $i = 1, ..., N$. We refer to one row of such matrix as the RNA-seq of cell i (table 2.2).

This data has several features that require specific bio-informatics approaches. It's usually very sparse and expression levels are subject to temporal fluctuations. Moreover, intrinsic differences

such as sequence length and amplification depth can influence the raw count. In the following paragraphs we explain some of the main techniques use to process scRNA-seq raw data.

Normalization Each cell's gene expression counts are divided by the total counts for that cell and then multiplied by a scaling factor, such that the row-wise sums of the cell-by-gene matrix are constant and equal to k.

$$x_{ij}^{norm} = \frac{x_{ij}}{\sum_{j=1}^{G} x_{ij}} \cdot k$$

Log-transformation This transformation is applied to make the data more normally distributed. It reduces the impact of highly expressed genes and brings the gene expression levels of lowly expressed genes closer to each other.

$$x_{ij}^{log} = log(x_{ij}^{norm} + 1)$$

Notice this transformation is usually applied to the normalized data.

Highly Variable Genes Selection Not all genes are equally informative for characterizing different cell types. Highly variable genes (HVGs) are those that exhibit greater variability across cells than would be expected by chance. To reduce dimensionality, a maximum number of HVGs to consider is chosen and all the others are discarded.

Value Binning Different absolute values for raw counts can convey different meanings, due to different sequencing protocols and methodology used. In order to tackle this issue, the raw counts are transformed to a relative measure using the value binning technique. Given the distribution of raw counts for a single cell RNA-seq, a total of B quantiles are computed. Then, each raw count in the cell-by-gene matrix is replaced with the quantile it falls into. In this way, the meaning of x_{ij} is consistent among all cells. For example, $x_{ij} = B$ indicates that gene j'expression is the highest among all possible expressions for the genes in cell i.

UMAP Uniform Manifold Approximation and Projection (UMAP) is a dimensionality reduction technique used to visualize high-dimensional data in a low-dimensional space[20]. It preserves both the global structure of the data and the local relationships between data points. In the context of scRNA-seq, it is particularly useful to reduce the multi-dimensional coordi-

nates of one cell expressed in terms of thousands of gene, to just a 2D representation which effectively captures clusters and relationships between different cell types or states.

Cell	**Gene 1**	**Gene 2**	**...**	**Gene** G
Cell 1	112	14	$\dots$	0
Cell 2	88	36	$\dots$	55
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Cell N	0	21	$\dots$	4

Table 2.2: Example of a raw cell-by-gene Matrix, entries are reads.

2.2.5 Transcriptional Interactions

Even though they all contain the same genetic information, different cell types in a multicellular organism differ dramatically in structure and function. This is possible because even if present, not all genes are expressed in the same way in all cells. The complex mechanisms which control gene expression are what enables cells to differentiate themselves and serve different purposes within a living being, ultimately determining which proteins are produced and to serve what functions. Moreover, each cell is capable of altering its pattern of gene expression in response to extracellular cues and stimuli. Such regulatory processes can act at any stage in the pathway leading from DNA to protein. For example [1]:

1. *Transcriptional* Control, when and how often a given gene is transcribed;
2. *RNA Processing* Control, how RNA sequences are processed and matured;
3. *RNA Transport* Control, which RNA sequences are transported from the nucleus to other regions of the cells;
4. *Translational* Control, which RNA sequences are translated into proteins;
5. *RNA Degradation* Control, selectively deteriorate specific RNA molecules;
6. *Protein Activity* Control, selectively inactivate or activate proteins after they have been created.

A transcription factor (TF) is a protein that controls the rate of transcription of genetic information from DNA to mRNA, by binding to a specific DNA sequence. The function of TFs is

to regulate gene expression in order to make sure that genes are expressed in each cell at the right time and in the right amount. TFs can act as activators or repressors, hence stimulating or inhibiting the transcription of specific DNA regions. A gene whose expression is regulated by a specific TF is called a regulon of that TF.

Transcriptional interactions refers to the set of processes which involve the activity of transcription factors, hence regulating gene expression at the very first stage of the pathway from DNA to protein, when the former is transcribed to RNA.

2.3 scGPT

In 2024, a 50 millions parameters foundation model trained on scRNA-seq data from over 33 million human cells, dubbed Single Cell GPT (scGPT), was published [6].

The model is an encoder-only transformer optimized via MLM. Loosely speaking, random entries of the RNA-seq of one cell are masked and the model is asked to predict them based on the other expression values. Its architecture can be divided into three main parts:

1. *Input Embedding*, combines different information from the scRNA-seq data into a single element;
2. *Encoder*, encodes the input embedding using the traditional Transformer encoder;
3. *Expression Decoder*, maps the encoded input at the masked position to the predicted expression value.

2.3.1 Input Embeddings

Given some scRNA-seq data, denote $X \in \mathbb{R}^{N \times G}$ the corresponding cell-by-gene matrix and $i = 1, ..., N$ one of its rows, i.e., one cell. The model's input is made of three components:

1. *Gene Tokens*, the gene names treated as elementary tokens. The authors use $vocab\_size =$

$n_g = 60,697$;

$$g^i = [g^i_1, ..., g^i_M]$$

2. *Expression Values*, the RNA expression values corresponding to the genes in the gene tokens. The raw counts from the cell-by-gene matrix are processed using normalization, log-transformation, highly variable gene selection and value binning;

$$x^i = [x^i_1, ..., x^i_M]$$

3. *Condition Tokens*, the authors suggest to use this vector to include additional information about the input genes, choosing among n_g different labels. However, they never actually use it.

$$c^i = [c^i_1, ..., c^i_M]$$

Usually, scRNA-seq data is very sparse, with thousands of genes measured, but only a small fraction actually expressed in each cell. Hence why the need of reducing the input to only $M \leq G$ genes, which the authors set equal to the number of highly variable genes selected.

In order to get the embedding, each symbol in each of the three components of the input is projected to a d_{model}-dimensional space. For g^i, c^i, whose symbols are tokens, this is done as in the traditional Transformer using projection matrices $W^g \in \mathbb{R}^{n_g \times d_{model}}$ and $W^c \in \mathbb{R}^{n_c \times d_{model}}$. Instead, even though the entries of x^i are the bins the corresponding raw counts fall into, hence theoretically labels as well, the authors decide to encode this component of the input using neural network projections to enhance expressivity. In particular, each symbol of x^i is sent through a two-layers FFNN with d_{model} hidden units and d_{model} output dimension, hence yielding a d_{model} vector representation of the original bin value.

The final input embedding h^i for cell i is obtained by summing the embedding of each component:

$$h^i = emb_g(g^i) + emb_x(x^i) + emb_c(c^i)$$

Lastly, a special $< cls >$ token is appended at the beginning of the input sequence. This is done because, ideally, the model will use this new token to learn and summarize information about the entire cell.

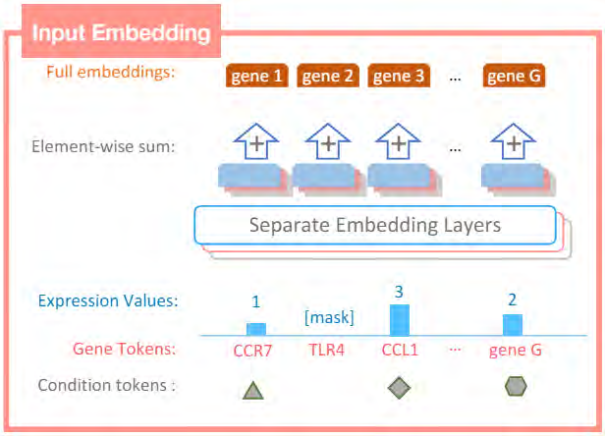

Figure 2.6: scGPT Input Embedding [6]

2.3.2 Encoder

The encoder is made of $N = 12$ layers. Each encoder layer is made of a multi-head attention layer with $H = 8$ and FFNN of one single hidden layer of dimension $d_{ff} = 512$.

Notice the parameters are the same as those used in the original Transformer architecture, but for the hidden size of the FFNN which is four times smaller (section 2.1.8).

Moreover, they use a custom attention mask which allows keys and values to come from known genes and the $< cls >$ token only, while queries can originate from the unknown genes as well. We refer to the learned embedding of the $< cls >$ token simply as the "cell embedding".

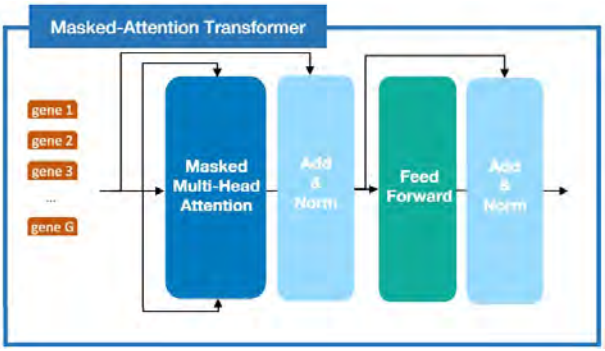

Figure 2.7: scGPT Encoder [6]

2.3.3 Expression Decoder

The final component of the model is the Expression Decoder, which maps the embedding at the masked position to the predicted expression value. This is simply a FFNN made of two hidden layers of dimension d_{model} and output a single value.

In parallel to this main task, additional output heads made of FFNNs are used to obtain parallel predictions which are used to train the model on many objectives at the same time, all benefiting from each other. The objectives the overall model optimizes can be summarized as follows.

Gene Expression Prediction (GEP) It is the main task of the model. At each iteration, a random subset of expression values in the x^i component of the input is masked and the model is asked to predict the missing values, using all gene tokens and the other non-masked expression values.

Gene Expression Prediction for Cell Modelling (GEPC) It involves predicting the entire set of expression values (masked or not) from the $< cls >$ token embedding only, which forces the model to learn meaningful whole-cell representations.

Non-zero Probability (NZP) This objective aims at modelling gene expressions as Bernoulli variables. That is, predict if the gene is either expressed or not, without specifying the exact value.

Non-zero Probability for Cell Modelling (NZPC) Same as the above objective, but using the $< cls >$ token embedding only. Again, forces the model to learn the overall expression patterns within a cell.

Elastic Cell Embedding (ECS) The idea is to use a similarity penalty to force the model to learn cell embeddings which are as different as possible.

2.3.4 Pre-training

The authors conduct via MLM and employ a composite loss function, which encompasses all aforementioned objectives. Specifically, it is made as follows:

$$\mathcal{L} = \mathcal{L}_{GEP} + \mathcal{L}_{GEPC} + \mathcal{L}_{NZP} + \mathcal{L}_{NZPC} + \mathcal{L}_{ECS}$$

In particular, the Mean Squared Error (MSE) is used for both $\mathcal{L}_{GEP}$ and $\mathcal{L}_{GEPC}$. The negative Binary Cross Entropy (BCE) is used for $\mathcal{L}_{NZP}$ and $\mathcal{L}_{NZPC}$. Lastly, $\mathcal{L}_{ECS}$ employs the negative squared cosine similarity between two embeddings.

The model is trained for 6 epochs on the 33 millions cells dataset.

Moreover, the authors use Mean Relative Error (MRE) together with MSE to evaluate the goodness of the model, yet MRE is never used as a training objective itself.

Experiment

The idea of our experiment is to exploit the condition tokens in the scGPT default input to inject some prior knowledge about transcriptional interactions into the model.

Intuitively, when predicting the masked value for gene j^*, it might be useful to have some information about whether a TF, which either inhibits or stimulates j^*, is present and active in the cell.

However, the only information available about a given cell is its RNA-seq, and inferring TFs' activity solely based on transcriptional activity is not trivial at all. Consider the case where a DNA region coding for a specific TF is transcriptional active, i.e., it's being transcribed into mRNA and hence the TF-coding gene is abundant in the scRNA-seq of a specific cell. It might be that as soon as the TF is translated from mRNA to protein, it is immediately deteriorated. In this scenario, even though the corresponding gene is present, the TF itself is not influencing the transcription of its target genes at all. Only when some external stimuli are received or some conditions are met, the TF stops being deteriorated and it can proceed with its regulatory activity. We tackle this issue by accurately selecting the training data.

To test our hypothesis, we re-train from scratch a simplified version of the scGPT model following the same procedure of the original paper, while including custom condition tokens.

3.1 Data

In order to tackle the TF transcription issue, we use two distinct scRNA-seq datasets. The first one, *BREAST-25K* [22], represents a general case scenario, where it might be that our proposed improvement to the model turns out to be useless, due to the impossibility of linking transcriptional activity of the corresponding gene with TF activity. The other one, *HYPOXIA_9K*, is a best case scenario, where a specific experimental-induced perturbation triggers many TFs and we can assume the link between TF activity and corresponding gene expression to hold, at least for a fraction of the considered TFs.

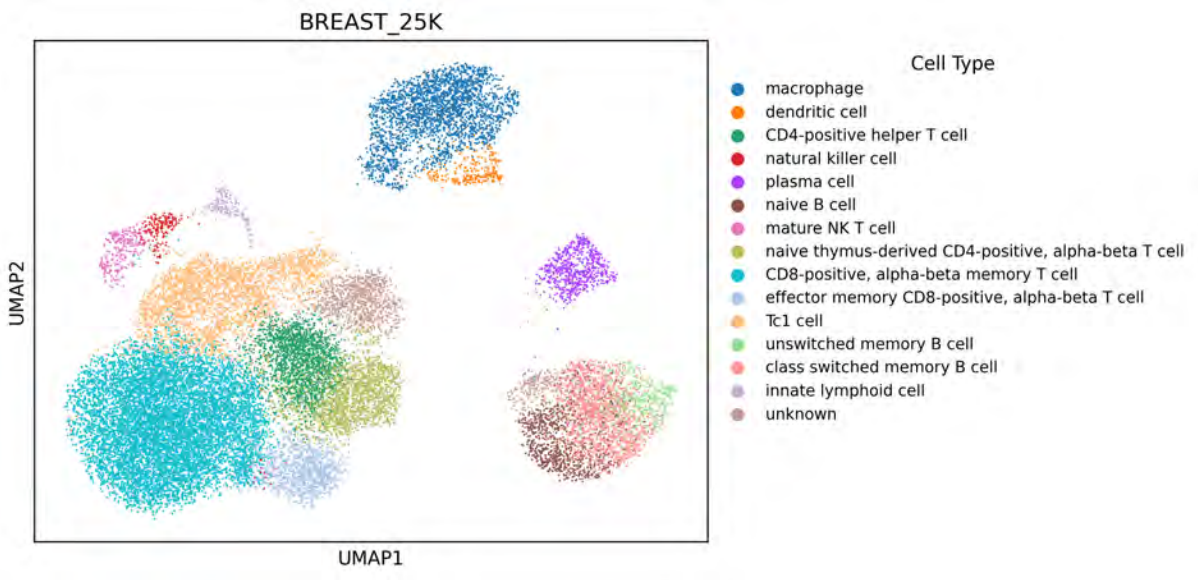

Figure 3.1: UMAP plot of the *BREAST-25K* dataset

BREAST-25K The first dataset comprises the RNA expression values of 34,455 genes measured for 25,382 human breast cells from the immune compartment. As shown in Figure 3.1, the dataset encompasses 14 cell types and 1,162 cells have unknown type.

HYPOXIA-9K The second dataset comprises the RNA expression values of 19,046 genes measured for 9,234 cells from the MDA-MB-231 breast cancer cell line. As shown in Figure 3.2, the cells were subjected to either normoxic or hypoxic state characterized by a normal 20-21% oxygen concentration for the former, and a reduced 1% for the latter.

DoRoThEA The information relative to TFs, regulons and interaction type are obtained from the Discriminant Regulon Expression Analysis (DoRoThEA) database [10].

3.2 Methodology

In order to conduct our experiment, we slightly deviate from the original model implementation of the authors. The modifications introduced are summarized in the following paragraphs.

Model Size Due to computational constraints, we scale down model size from 50 millions to 3 millions and 1 million parameters. The input sequence length, which corresponds to the number of highly variable genes selected, is reduced to 500 and 100 maximum number of genes

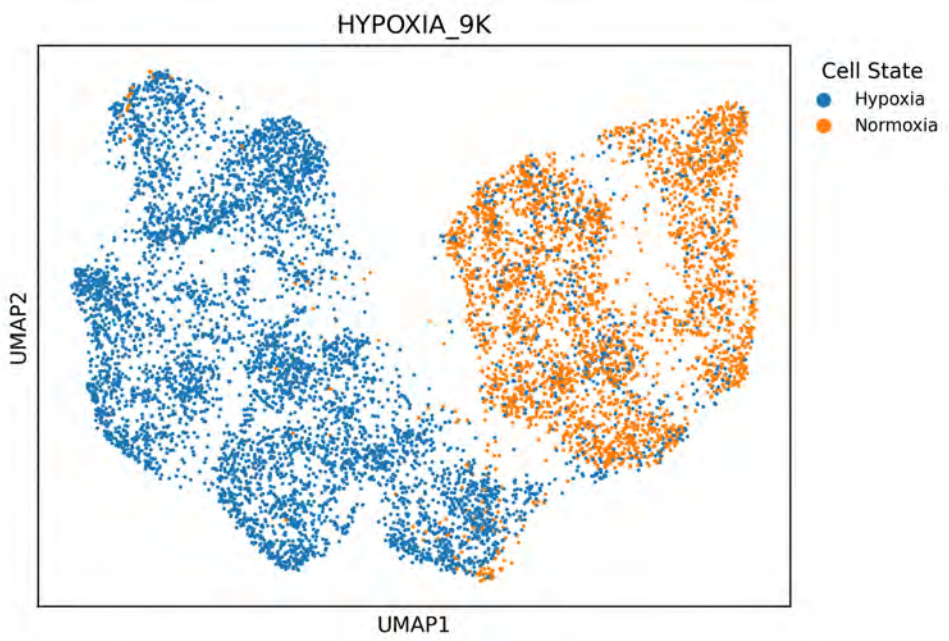

Figure 3.2: UMAP plot of the *HYPOXIA-9K* dataset

for the two model sizes respectively (see Table 3.1).

Masking We simplify the masking procedure by allowing only one expression value to be masked per input sequence, instead of a random subset.

Condition Tokens We introduce a novel type of condition tokens, which summarize transcriptional interactions between any gene in the input sequence and the masked gene whose value the model is trying to predict. In particular, given a masked gene j^*, each token c^i_j in the input component $c^i = [c^i_1, ..., c^i_M]$ for cell i is now defined as follows:

$$c^i_j = \begin{cases} 1 & \text{if the TF coded by } j \text{ stimulates } j^* \\ -1 & \text{if the TF coded by } j \text{ inhibits } j^* \\ 0 & \text{otherwise} \end{cases}$$

Similarly to what the authors suggest for the expression values, we encode this condition tokens using neural networks projections instead of simple embedding tables. In fact, the token values are by construction related to their semantic meaning: positive values if a gene stimulates the target, negative if inhibits, zero otherwise. If encoded using simple embedding tables, this information would be lost. Notice that we include in the "otherwise" the cases where j does not code for a TF or where j codes for a TF which can both inhibit and stimulate j^* according to

Model Name	n_{params}	d_{model}	N	H	d_{ff}	M
scGPT-50M	50.0M	512	12	8	512	1200*
scGPT-3M	3.0M	256	6	4	256	500
scGPT-1M	1.0M	128	4	4	128	100

Table 3.1: Specifics of the different model sizes. scGPT-50M refers to the original model. * this refers to the original input length, for our tests we reduced this to 500 as well.

external factors, which are interactions not useful for the scope of our analysis.

To test the validity of the proposed idea, two instances of each model size are trained on the two datasets separately, either with or without including the condition tokens component in the input, resulting in eight total models to be compared.

To further test efficacy, we also train the models using a random vector c^i_{random}, whose entries are sampled from a discrete uniform distribution on $\{-1, 0, 1\}$ and resemble true condition tokens yet without conveying any meaningful information.

Lastly, we conduct the same experiments in light of a recently proposed method [11] which aims at inferring the activity of a TF solely based on scRNA-seq. In a nutshell, it computes the correlation between the scRNA-seq of one TF-coding gene and an enrichment score, which is proportional to the scRNA-seq of its regulons. High correlation means that when a TF-coding gene is highly expressed, so are the TF's regulons, which might indicate activity of the TF. We exploited this method by reducing the list of TFs in two ways. The first, more conservative, considered those TFs whose correlation were in the top or bottom 25%. The second one, more aggressive, restricted to top or bottom 10% only.

For validation purposes, we compute out-of-sample metrics on 10% of the original dataset.

As a baseline, we consider the original scGPT-50M model fine-tuned on each dataset for 3 epochs.

3.3 Results

In this section, we report the MRE as main discriminant metric for the analysis and comparisons. The MSE either follows the same trend as the MRE or is not informative at all about the validity

Dataset	*Model*	*Condition Tokens*	*MRE* (10^4)	*MSE*
BREAST-25K	scGPT-3M	No	**54.05**	**32.05**
		Yes	69.19	35.03
	scGPT-1M	No	**109.65**	**52.96**
		Yes	120.35	56.23
	scGPT-50M	No	107.94	37.83
HYPOXIA-9K	scGPT-3M	No	210.57	96.82
		Yes	**190.59**	**94.97**
	scGPT-1M	No	224.54	**104.17**
		Yes	**210.63**	105.33
	scGPT-50M	No	83.98	44.82

Table 3.2: Models' performance on the two datasets with or without condition tokens. The baseline model is reported, but not used for comparison.

of the proposed approach. Nonetheless, all validation MSE plots can be found in Appendix 5.2. All results are reported in Table 3.2.

As one could expect, the main trend we observe with Transformer and NLP emerges once again: increasing model size, input length and training data consistently improves performance, as shown in Figure 3.3.

Notice that, however, increasing model size alone might not be enough to improve performance, and the results suggest that it's better if input length is augmented as well. In fact, after as few as 5 epochs, scGPT-3M trained from scratch on *BREAST-25K* becomes comparable to scGPT-50M fine-tuned on the same dataset for only 3 epochs, both using reduced input length of 500 highly variable genes.

At the same time, a rich and diverse training dataset, such as *BREAST-25K* with its 25 thousands cells of at least 15 different types, is shown to be superior to a more restricted one, such as *HYPOXIA-9K* which contains 9 thousands cells from one cell line only. In fact, even though on *BREAST-25K* the fine-tuned model and the trained-from-scratch one seem to be comparable, the same is not true at all for *HYPOXIA-9K*, where the advantages of transfer learning with a model trained on over 33 millions cells might explain the huge gap in predictive accuracy. Additionally, the same scGPT-50M achieves higher fine-tuning performance on the latter dataset, suggesting it's able to capture more entirely the nuances of this particular group of cells, compared to the more complex and variegated breast cells.

The inclusion of condition tokens does not benefit the model on *BREAST-25K* (figure 3.4),

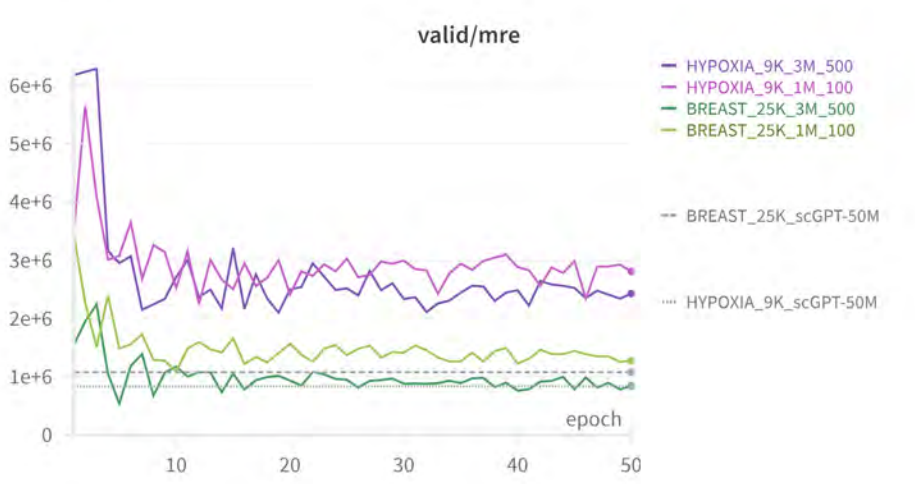

Figure 3.3: MRE of both model sizes on both datasets. The horizontal lines show the performance of scGPT-50M fine-tuned for 3 epochs on the two datasets, with input length $M = 500$ (refer to Appendix 5.1 for naming convention).

Figure 3.4: MRE with and without using condition tokens on *BREAST-25K*.

which confirms our initial concerns about weak linkage between scRNA-seq abundance and TF activity within the considered sample.

Instead, the inclusion of condition tokens does benefit the model on *HYPOXIA-9K*, where performance constantly improves throughout all training epochs compared to the model trained without this component in the input (figure 3.5). Efficacy of this feature is further confirmed by the drop in performance when condition tokens are just randomly sampled, which proves that the previously observed gain is not due to some random pattern in the data (figure 3.6), but to the model actually extrapolating useful insights from the transcriptional interactions reported.

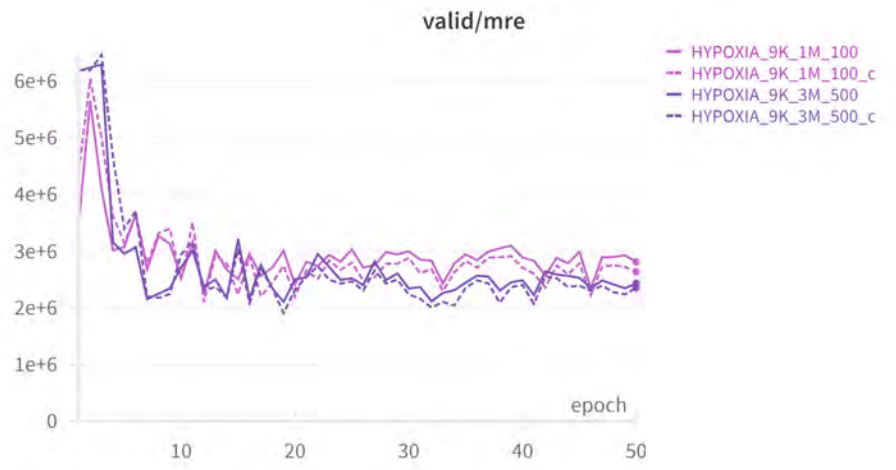

Figure 3.5: MRE with and without using condition tokens on *HYPOXIA-9K*.

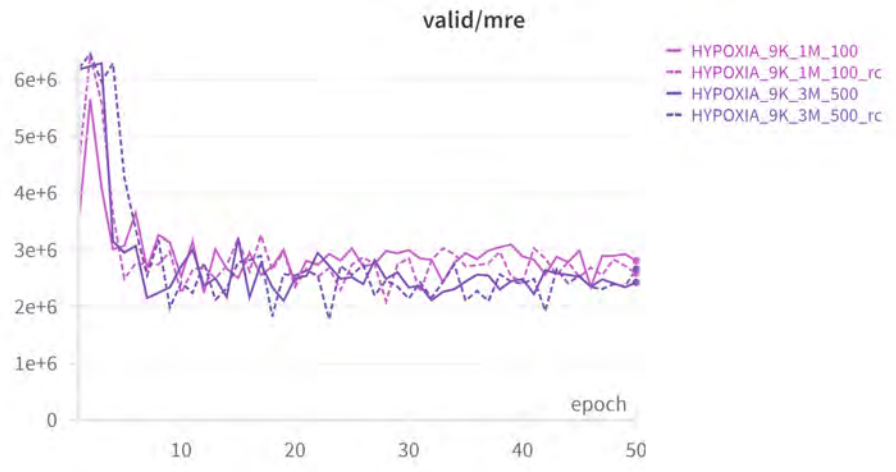

Figure 3.6: MRE with and without using random condition tokens on *HYPOXIA-9K*.

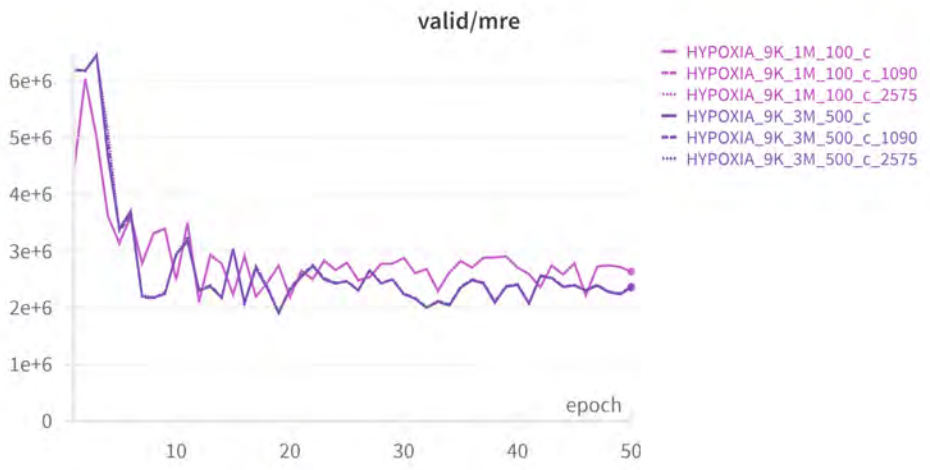

Figure 3.7: MRE with condition tokens using all TFs or a filtered list on *HYPOXIA-9K*.

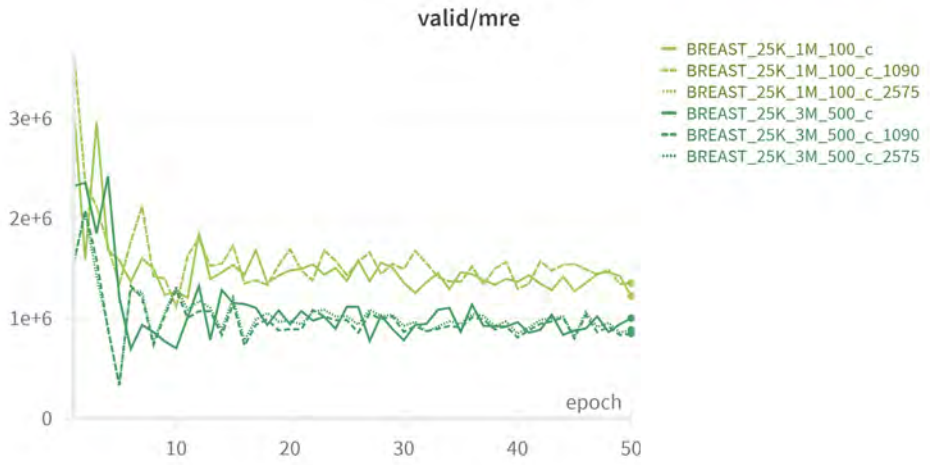

Figure 3.8: MRE with condition tokens using all TFs or a filtered list on *BREAST-9K*.

When restricting the list of TFs, we do not see any change in performance on *HYPOXIA-9K*. For *BREAST-25K*, instead, we do see an effect: even though there is still no advantage compared to the base model without condition tokens, we see an improvement compared to including all TFs without discrimination, especially on the largest model with the more aggressive filtering approach (figure 3.8).

The difference in performance observed across the two datasets can be explained also by the effective number of condition tokens used (figure 3.9). On *HYPOXIA-9K*, we observe a significantly higher number of non-zero condition tokens, which suggests the model is exploiting the additional information more, hence why we have better performance.

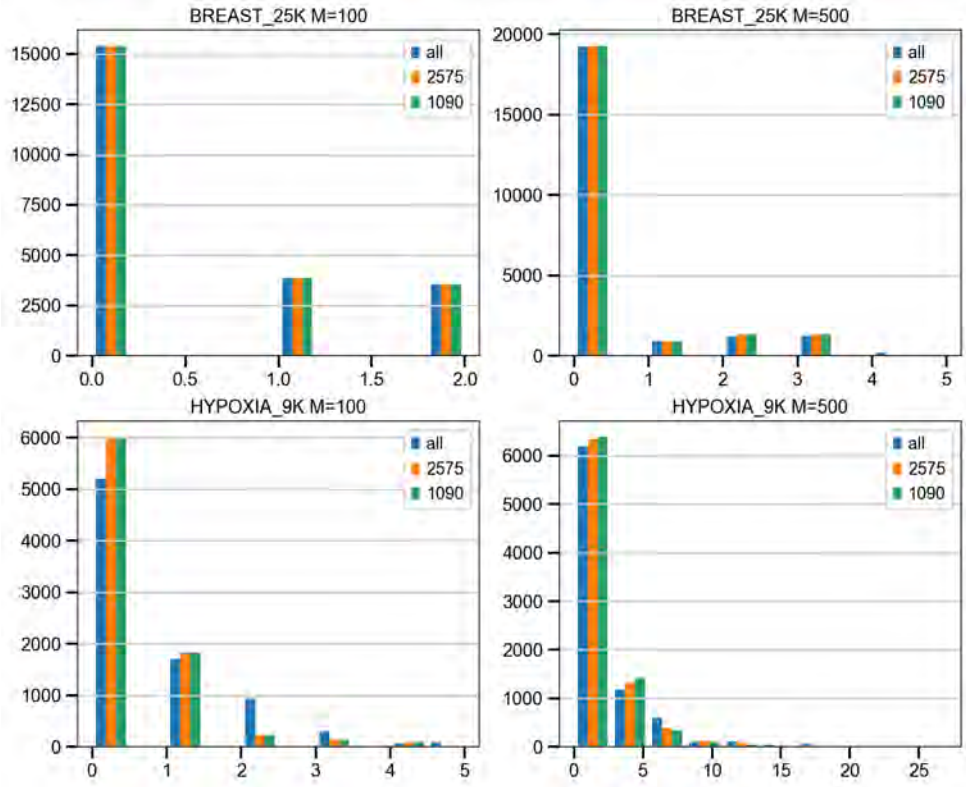

Figure 3.9: Number of non-zero condition tokens used in one epoch of training, grouped by dataset and input length.

Conclusion

In the first part of this dissertation, we introduced Natural Language Processing by discussing its origins as a sub-field of Machine Learning, leading up to the groundbreaking introduction of the Transformer architecture. Then, we explained some key concepts in Life Sciences, with a focus on Genomics and Single Cell RNA-sequencing (scRNA-seq). Lastly, we introduced a recent endeavor which links these two disciplines together by adapting an encoder-only Transformer model, dubbed scGPT, to perform gene expression value prediction.

In the second part, we presented our experiment which highlights the potential of enhancing model performance by incorporating prior knowledge. In particular, we considered transcriptional interactions and tested our idea through the inclusion of condition tokens into scGPT's input, which is how the authors propose to integrate additional information into the model. We showed a significant positive impact on the model's predictive accuracy on certain datasets, where specific validity assumptions hold.

When using condition tokens on the *HYPOXIA-9K* dataset, the model's performance improved consistently throughout the training process compared to not using them. On the other hand, the *BREAST-25K* dataset did not show similar improvements. This lack of benefit can be explained by the fact that scRNA-seq abundance of a gene coding for a specific transcription factor (TF) does not always correlate with that specific TF being actually active. Hence, in this scenario, including without any sort of discrimination information about all TFs coded by genes present in the scRNA-seq does not help prediction at all.

Indeed, when restricting the list of TFs considered such that the validity assumptions are more likely to hold, we see an improved performance on *BREAST-25K*, suggesting this might be the key driver of the behaviors observed.

Still, the mixed results across different datasets highlight an essential consideration: the effectiveness of incorporating prior knowledge is context-dependent and, in order to optimize model performance, identifying where such knowledge can be beneficial is as crucial as successfully incorporating it into the model.

Nonetheless, leveraging domain-specific information is important and future research direc-

tions might focus on scaling and refining this proposed approach. In particular, testing the same method on a broader range of datasets and with greater scale models to understand its generalizability is something we are interested in. At the same time, developing and constructing more sophisticated methods for incorporating prior knowledge, which go beyond the simple use of condition tokens shall be taken into consideration. Lastly, since one of the main contributions of this work is to show the importance of understanding the exact contexts and scenarios where including prior knowledge is actually useful, new methods and techniques developed for this purpose should also be the focus of future endeavors.

In general, this Final Paper aims at showcasing the potential of Machine Learning applied to Biology, and we can't wait to see what the future holds for this rapidly evolving and fascinating field where machines and living beings are extremely intertwined.

Bibliography

[1] Bruce Alberts, Alexander Johnson, Julian Lewis, David Morgan, Martin Raff, Keith Roberts, and Peter Walter. Molecular biology of the cell. *Molecular Biology of the Cell*, 2017.

[2] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, 2015.

[3] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3, 2003.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 2020-December, 2020.

[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2014.

[6] Haotian Cui, Chloe Wang, Hassaan Maan, Kuan Pang, Fengning Luo, Nan Duan, and Bo Wang. scgpt: toward building a foundation model for single-cell multi-omics using generative ai. *Nature Methods*, 2024.

[7] Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Hassan Sirelkhatim, Guillaume Richard, Marcin Skwark, Karim Beguir, Marie Lopez, and Thomas Pierrot. The nucleotide transformer: Building and evaluating robust foundation models for human genomics. *bioRxiv*, 2023.

[8] Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training

of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1, 2019.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR 2021 - 9th International Conference on Learning Representations*, 2021.

[10] Luz Garcia-Alonso, Christian H. Holland, Mahmoud M. Ibrahim, Denes Turei, and Julio Saez-Rodriguez. Benchmark and integration of resources for the estimation of human transcription factor activities. *Genome Research*, 29, 2019.

[11] Tommaso Giacomello. Enriched gene regulatory network algorithm applied to cancer cell dataset. *Final Paper for Bocconi University BSc*, 2024.

[12] Ashraful Haque, Jessica Engel, Sarah A. Teichmann, and Tapio Lönnberg. A practical guide to single-cell rna-sequencing for biomedical research and clinical applications. *Genome Medicine*, 9, 2017.

[13] Zellig S. Harris. Distributional structure. *WORD*, 10, 1954.

[14] Julia Hirschberg and Christopher D. Manning. Advances in natural language processing. *Science*, 349, 2015.

[15] Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. 2020.

[16] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28, 1972.

[17] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A.A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596, 2021.

[18] Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul Mcvay, Michael

Rabbat, and Yuandong Tian. Beyond a*: Better planning with transformers via search dynamics bootstrapping. 2024.

[19] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9, 2008.

[20] Leland McInnes, John Healy, Nathaniel Saul, and Lukas Großberger. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3, 2018.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*, 2013.

[22] Aviv Regev, Sarah A. Teichmann, Eric S. Lander, Ido Amit, Christophe Benoist, Ewan Birney, Bernd Bodenmiller, Peter Campbell, Piero Carninci, Menna Clatworthy, Hans Clevers, Bart Deplancke, Ian Dunham, James Eberwine, Roland Eils, Wolfgang Enard, Andrew Farmer, Lars Fugger, Berthold Göttgens, Nir Hacohen, Muzlifah Haniffa, Martin Hemberg, Seung Kim, Paul Klenerman, Arnold Kriegstein, Ed Lein, Sten Linnarsson, Emma Lundberg, Joakim Lundeberg, Partha Majumder, John C. Marioni, Miriam Merad, Musa Mhlanga, Martijn Nawijn, Mihai Netea, Garry Nolan, Dana Pe'er, Anthony Phillipakis, Chris P. Ponting, Stephen Quake, Wolf Reik, Orit Rozenblatt-Rosen, Joshua Sanes, Rahul Satija, Ton N. Schumacher, Alex Shalek, Ehud Shapiro, Padmanee Sharma, Jay W. Shin, Oliver Stegle, Michael Stratton, Michael J.T. Stubbington, Fabian J. Theis, Matthias Uhlen, Alexander Van Oudenaarden, Allon Wagner, Fiona Watt, Jonathan Weissman, Barbara Wold, Ramnik Xavier, and Nir Yosef. The human cell atlas. *eLife*, 6, 2017.

[23] D.E Rumelhart, G.E Hinton, and R.J Williams. Learning internal representations by error propagation (original). *Explorations in the Micro-Structure of Cognition Vol. 1 : Foundations*, 1986.

[24] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *54th Annual Meeting of the Association for Computational Linguistics, ACL 2016 - Long Papers*, 3, 2016.

[25] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *Advances in Neural Information Processing Systems*, 4, 2014.

[26] Fuchou Tang, Catalin Barbacioru, Yangzhou Wang, Ellen Nordman, Clarence Lee, Nanlan Xu, Xiaohui Wang, John Bodeau, Brian B. Tuch, Asim Siddiqui, Kaiqin Lao, and M. Azim Surani. mrna-seq whole-transcriptome analysis of a single cell. *Nature Methods*, 6, 2009.

[27] A. M. Turing. Computing machinery and intelligence. *Machine Intelligence: Perspectives on the Computational Model*, 1950.

[28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017-December, 2017.

Appendix

5.1 Naming Convention

We stick to the following naming convention when referring to a training run:

DatasetName_DatasetSize_nParams_InputLength_conditionTokens_quantiles

Where ”conditionTokens” can be either ” ” if nothing is used, ”c” if true condition tokens are used, ”rc” if random condition tokens are used. ”quantiles” can be either ” ” if no filtering is applied, ”1090” if the top and bottom 10% is selected, ”2575” if the top and bottom 25% is selected.

5.2 Validation MSE

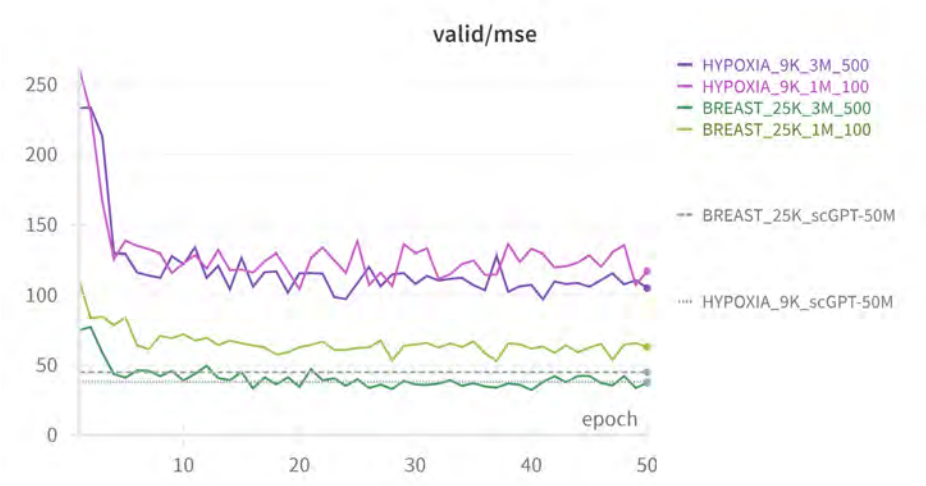

Figure 5.1: MSE of both model sizes on both datasets. The horizontal lines show the performance of scGPT-50M fine-tuned for 3 epochs on the two datasets, with input length $M = 500$.

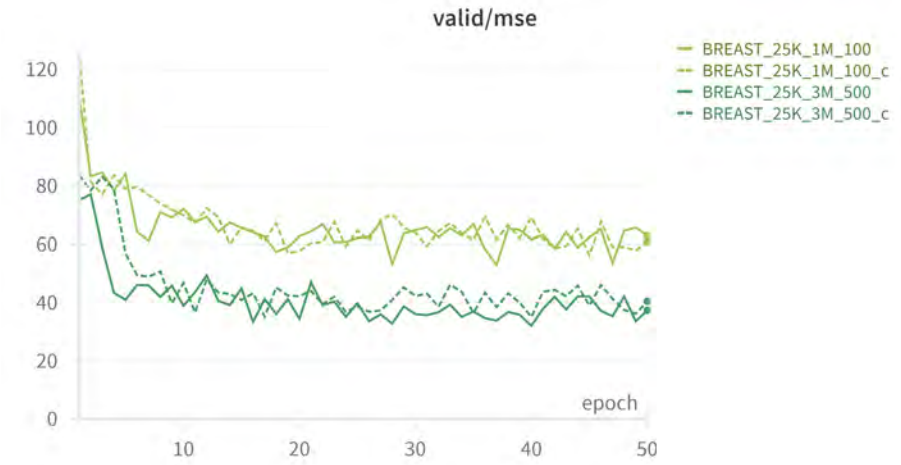

Figure 5.2: MSE of both model sizes with and without condition tokens on *BREAST-25K*.

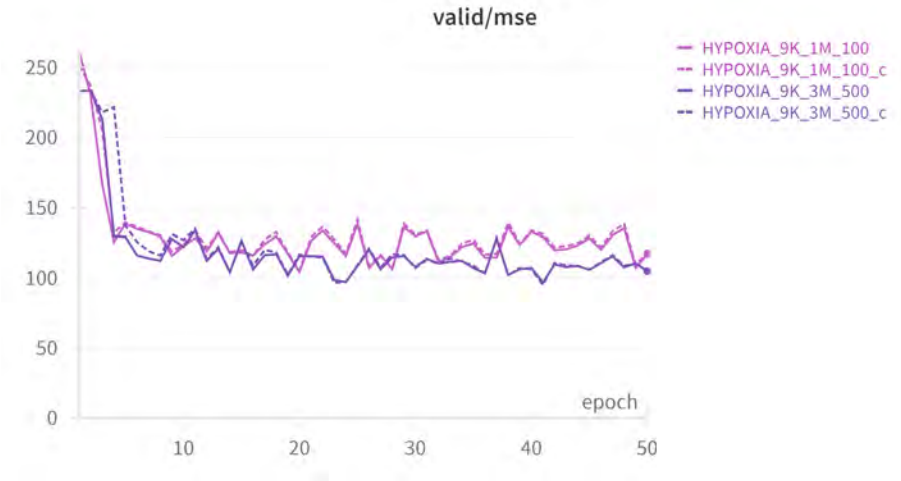

Figure 5.3: MSE of both model sizes with or without condition tokens on *HYPOXIA-9K*.

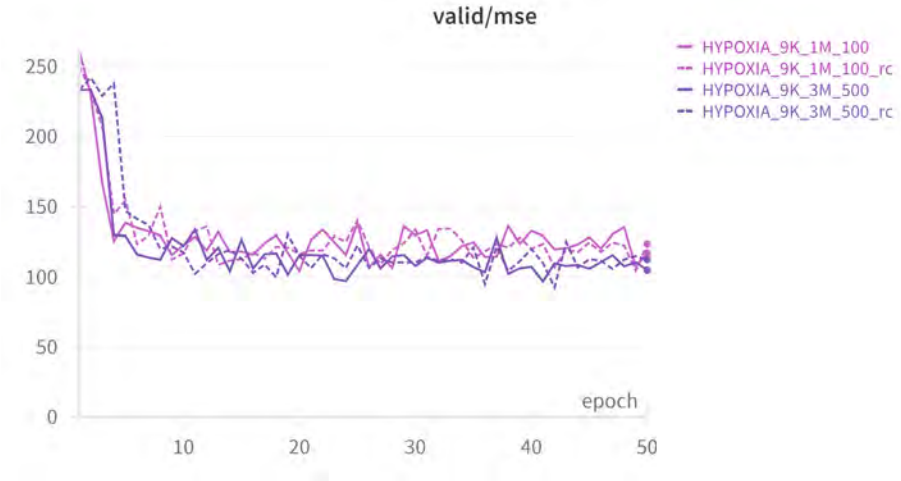

Figure 5.4: MSE of both model sizes with or without random condition tokens on *BREAST-25K*.

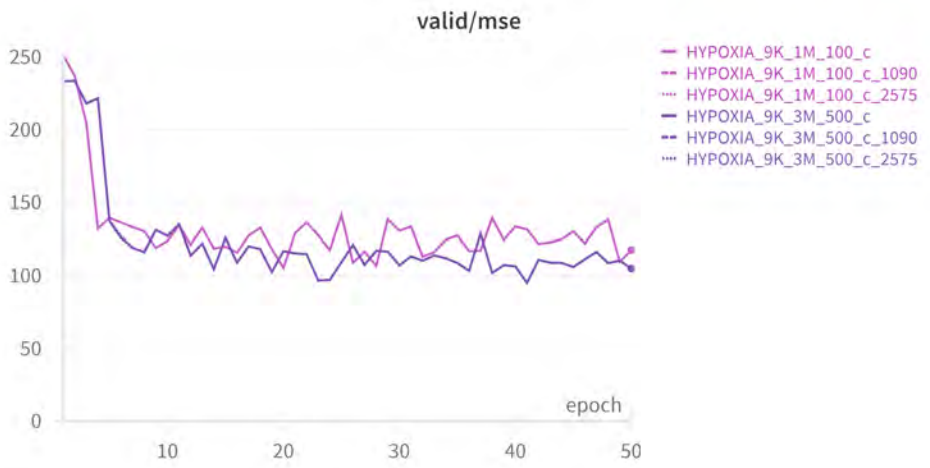

Figure 5.5: MSE with condition tokens using all TFs or a filtered list on *HYPOXIA-9K*.

Figure 5.6: MSE with condition tokens using all TFs or a filtered list on *BREAST-25K*.