Translating the Code of Life

Simon Lee[1] Vedavyas Mallela[2] Razvan Marinescu[3] Vanessa Jonsson[1]
siaulee@ucsc.edu *vmallela1@student.gsu.edu* *razvan@csail.mit.edu* *Vjonsson@ucsc.edu*

[1]UCSC Genomics Institute, University of California, Santa Cruz, CA
[2]College of Arts & Sciences, Georgia State University, Atlanta, GA
[3]Computer Science & Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA

Abstract

Single-cell transcriptome sequencing (scRNA-seq) research has recently become one of the newest and widely used methods for biological research due to its ability to gain a more comprehensive understanding of the biology of a cell. These recent advances in single-cell genomics technologies have enabled the investigation of gene regulation programs at an unprecedented scale. Therefore the development of single-cell multi-modal omics computational tools is an active field of research where scientists are working towards understanding the inner workings of biological systems. For this reason, our laboratory has experimented using the *t5-small*, an attention-based Natural Language Processing model developed by *Google*. We have developed a novel method to "translating" the code of life for multi-modal prediction by modifying the t-5 model's parameters and architecture. Being able to get an understanding of this single-cell measurement data can help prompt a leap in scientific discovery in trying to comprehend the causes of complex disease and biological diversity.

1 Introduction

The field of genomics has been growing rapidly in recent years. With novel approaches being developed in gaining extensive insight on genomes through Next Generation Sequencing (NGS) techniques, it has allowed scientists to understand the biology of multi-cellular organisms at a much greater scale. In this research, we worked specifically with single-cell transcriptome sequencing (scRNA-seq) methodology to try and map its chromatin (ATAC-seq) to specific gene expressions (GEX) and vice versa. While these new NGS technologies have been revolutionary in terms of their speed, it is equally important to note that the massive data produced by NGS has also presented a significant challenge for data analyses [1]. The analysis of these multi-dimensional data sets can be described as NP-hard, hence there is a need for a state of the art machine learning model to approach these types of problems. Therefore, our research team worked towards developing a unique methodology for scRNA-seq analyses by taking a pre-existing Natural Language Processing Model developed by *Google* and tailoring its parameters to satisfy our problem. Our goal is to translate this GEX data to ATAC-seq computationally through t5-small and other machine learning techniques. Although this problem is NP-hard, by using these tools, we are able to find corresponding sequences in a much less computational complexity manner. So in the following sections, we will briefly describe some Machine Learning concepts that play integral roles in the *t5-small*, an *attention-based transformer model*.

1.1 Neural Networks

Neural networks are multi-layer algorithms that use machine learning to recognize patterns in data sets. Each layer in the network is made up of many neurons with a specific activation function and these specific functions affect the activations of the following layer. The networks process text, or images as input and give outputs to train the network. As the inputs are passed through the hidden layers, the layers generate output. By applying and finding the best loss functions for the particular neural network [2], backwards propagation nudges the network in the right direction and forms a stochastic gradient descent, thereby optimizing the model. This feature is very analogous to how the human brain works in simulating the process of "learning".

Within the study of neural networks is *Attention*, a technique that mimics cognitive attention. The effect is intended to enhance some parts of the input while diminishing other parts - the thought being that the network should devote more focus to that small but important parts of the data. Learning which part of the data is more

important than others depends on the context and is trained by gradient descent. These attention-based models have been extremely prevalent in the field of Natural Language Processing where attention is used to perform tasks like Q&A, translation, and generate summaries of large bodies of texts. Using this attention-based mechanism has been an inspiration for our design to make multi-modal sequence-to-sequence predictions for scRNA-seq data.

1.2 Natural Language Processing

Natural Language Processing (NLP) is a field of machine learning that allows computers to analyze, understand, and generate human language [3]. NLP and text information retrieval (IR) research has begun to intersect, allowing for features such as sentiment analysis, machine translation, and extracting meaning from user text. In our project, we make use of the *t5-small*, a Machine Translation (MT) or robotized interpretation model by *Google* which performs a procedure that allows computer software to translate text from one language to another without human contribution. At its fundamental level, machine translation performs a straightforward replacement of atomic words in a single characteristic language for words in another. Although the t5-small has other features like understanding syntax, and Q&A features, these will not be discussed within the scope of this paper.

We understand it seems rather peculiar to take a NLP model for scRNA-seq analyses, but our intentions for using it will be cleared up in future sections. To foreshadow a bit, we have selected this model in particular because it takes in input as a sequence-to-sequence. So through this feature, we are able to correspond our GEX to ATAC-seq data (and vice versa) just like we would be able to for English to German translation.

1.3 Transfer Learning

Transfer learning is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task. It is a popular approach in deep learning (a sub-field of machine learning) where pre-trained models are used as the starting point on computer vision and NLP tasks given the vast compute and time resources required to develop neural network models on these problems and from the huge jumps in skill that they provide on related problems. Deep learning models are special because they harness the power of neural networks and use the hidden layers to properly make inferences.

Our *t5-small* utilizes this concept of Transfer learning, where the model is first pre-trained on a data-rich task before being fine-tuned on a downstream task [4]. The effectiveness of transfer learning has given rise to a diverse array of approaches, methodology, and practice which is one of our primary reasons for choosing this specific model from *Google*. The *t5-small* is a model that converts every language problem into a sequence-to-sequence format. As we briefly mentioned in Section (1.2), our reasoning for selecting a sequence-to-sequence format was because we will be working with specific gene expression and chromatin locations which can be passed into these attention-based models as a python dictionary holding two sets of string (text) inputs. Therefore we believe we have carefully chosen and constructed an architecture that will excel in solving this type of research task.

2 Single Cell Genomics

Just 10 years ago, scientists were only capable of investigating about a thousand cells with the technology accessible then. As of 2020 however, this scope has grown exponentially to where we can examine millions of cells using single-cell, measurement technologies, which are driving a revolution in the life sciences. Recent advances have made it possible to measure multiple high-dimensional modalities (e.g. DNA accessibility, trasncriptome, and proteome) simultaneously in the same cell. Such data provides, for the first time, a direct and comprehensive view of the layers of gene regulation that drive biological diversity and disease. Being able to get an understanding of this single-cell measurement data can help prompt a leap in scientific discovery in understanding the mechanisms behind complex biological systems.

However, these single-cell measurements are not always accessible. Physical access to DNA is a highly dynamic property of chromatin that plays an essential role in establishing and maintaining cellular identity. The organization of accessible chromatin across the genome reflects a network of permissible physical interactions through which enhancers, promoters, insulators, and chromatin-binding factors cooperatively regulate gene expression. This landscape of accessibility changes dynamically in response to both external stimuli and developmental cues, and emerging evidence suggests that homeostatic maintenance of accessibility is itself dynamically regulated through a competitive interplay between chromatin-binding factors and nucleosomes. In simpler terms, the chromatin is deemed accessible if and only if it is uncoiled. This property of being uncoiled is how we

retrieve the genetic information that we desire. However, from the sound of it, the chromatin data is not easily accessible. In human cells, we carry chromosomes that are made up of these chromatin fibers which come coiled up. So if accessible, there is a pooled barcode method called single-cell combinational indexing *(sci-CAR)* that allows a joint profiling of chromatin accessibility (scATAC-seq) and gene expression (GEX) of single cells in which we obtain our two corresponding types of data.

Knowing how NGS has revolutionized the access of multitudes of cell data, it is by no surprise that researchers have built methods to predict the flow of information. In order to study multiple modalities in the same single cell, scientists have studied the changes from DNA to RNA and RNA to Protein through the use and development of computational tools.[5]. Therefore many researchers have been using data science tools sourced largely from Machine Learning to construct methods that will help in this data analysis. Our objective is no different in which we wish to construct a multi-modal predictor using the *t5-small*, attention-based models. We therefore want to highlight the objective/problem that our method will solve in this paper. So given a modality (e.g. GEX or ATAC) the t-5 small will need to predict the other modality (GEX-> ATAC or ATAC-> GEX). We wish to know this information because we can learn a lot from the locations in which certain genes may be expressed. For pedagogical purposes, we wish to expand on the two types of data we have been referring to: GEX and ATAC-seq.

2.1 Gene Expression (GEX)

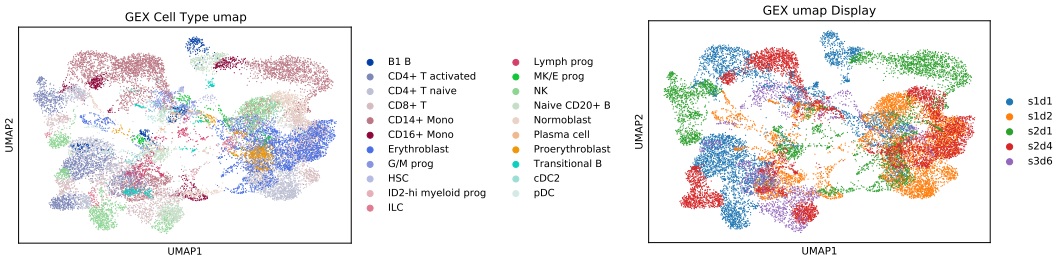


Figure 1: Left UMAP displays our GEX cell types and our right UMAP displays our source and donors

Gene expression analysis has become routine through the development of high-throughput RNA sequencing (RNA-seq) and microarrays. RNA analysis that was previously limited to tracing individual transcripts by Northern blots or quantitative Polymerase chain reaction (PCR) is now used frequently to characterize the expression profiles of populations of thousands of cells. The data produced from the bulk-based assays has led to the identification of genes that are differentially expressed in distinct cell populations and biomarker discovery. This data comes in the form of *annotated data* (anndata) which is a Python package for handling annotated data matrices in memory and on disk, positioned between pandas and xarray. The numbers within the cell can describe the level of gene expression within a specific immune cell as seen in Figure 1.

2.2 Assay for Transposase-Accessible Chromatin (ATAC)

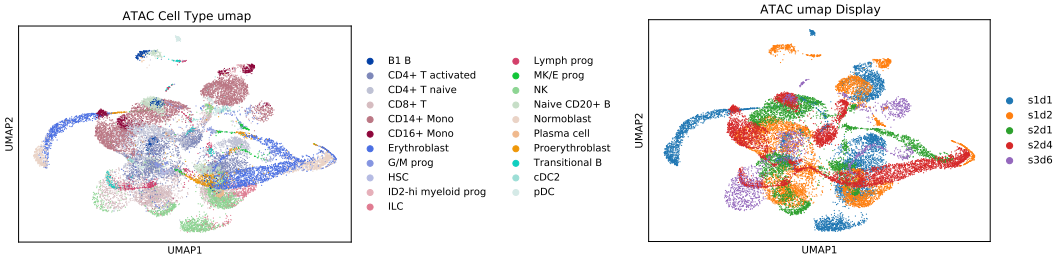


Figure 2: Left UMAP displays our ATAC cell types and our right UMAP displays our source and donors

Assay for Transposase-Accessible Chromatin or ATAC identifies accessible DNA regions by probing open chromatin with hyperactive mutant Tn5 Transposase that inserts sequencing adapters into open regions of the genome [6]. While naturally occurring transposases have a low level of activity, ATAC-seq employs the mutated hyperactive transposase [7]. In a process called “tagmentation", Tn5 transposase cleaves and tags double-stranded DNA with sequencing adaptors [8]. The tagged DNA fragments are then purified, PCR-amplified, and sequenced using next-generation sequencing [8]. Sequencing reads can then be used to infer regions of increased accessibility as well as to map regions of transcription factor binding sites and nucleosome positions [6]. The number of reads for a region correlates with how open that chromatin is, at single-nucleotide resolution. This data is also given to us in the form of anndata but unlike the GEX data, the ATAC-seq data comes in a binary format (e.g. 0,1) to resemble membership of some chromatin within a cell. This property of having binary data will play a huge role in our model’s architecture.

3 Transformer Attention Model

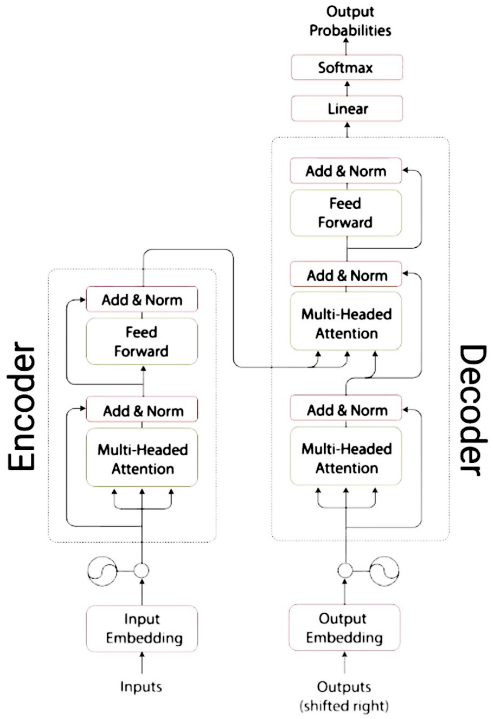


Figure 3: Full architecture on display

Transformers are taking the NLP world by storm. These incredible models are making multiple NLP breakthroughs and pushing the state of the art. They are used in many applications like machine language translation, conversational chatbots, and even to power better search engines. To understand transformers we first must revisit the attention mechanism. One of the main advantages of the attention mechanism is its ability to have long-term memory. A transformer model can “attend” or “focus” on all previous tokens (transfer learning) that have been generated. In the following sections, we will describe our model’s architecture to justify our reasoning for the selection due to it efficiency alone. To demonstrate the power of the architecture we will be working with a standard conversational chatbot example to understand the model at a basic level. For this reason, consider the following chatbot conversation:

Question: Hi How are you
Answer: I am fine

3.1 Encoder

On a high level, the encoders role is standard to those of other neural networks. It maps our input sequence into an abstract continuous representation that holds all the learned information of that input.

Preface. Positional Encoding

$$\mathrm{PE}_{(\mathrm{pos},2i)} = \sin\left(\frac{\mathrm{pos}}{10000^{\frac{2i}{d_{\mathrm{model}}}}}\right) \tag{1}$$

$$\mathrm{PE}_{(\mathrm{pos},2i+1)} = \cos\left(\frac{\mathrm{pos}}{10000^{\frac{2i}{d_{\mathrm{model}}}}}\right) \tag{2}$$

Before getting into the encoders and decoders we first must feed our input into a word embedding layer. A word embedding layer is similar to a lookup table that assigns a unique learned feature vector $\Phi(X)$ representation for each word. Neural networks learn through tensors so each word maps to a vector with continuous values to represent that word.

Luckily for us, mathematics can be used to make a unique positional encoding for each entry of our input. The developers working on Transformers simply took the properties of the trigonometric functions sine and cosine and made them produce unique values for each positional encoding. As seen in Eq. (1), for every odd index on the input vector, the model creates a vector using the cosine function. Similarly in Eq. (2), for every even index, we create a vector using the sine function. Then add those vectors to their corresponding input embedding layers. This successfully gives the network information on the position of each vector.

Now that each word has been given a unique identity, we can proceed to the encoder layer. The Encoder's job is to map all input sequences into an abstract continuous array representation that holds the learned information for that entire sequence. The encoder contains 2 sub-modules, *multi-headed attention*, followed by a fully connected network. There are also residual connections around each of the two sub-layers followed by a layer normalization.

I. Multi-Headed Layer

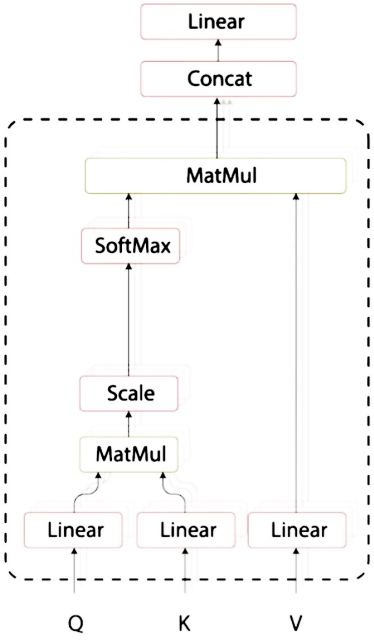


Figure 4: Multi-Headed Layer architecture on display

Multi-headed attention in the encoder applies a specific attention mechanism called *self-attention.* [9] Self-attention is a powerful property that allows the model to associate each word in the input, to other words. So in our conversational chatbot example, our model can learn to associate the word “you”, with “how” and “are”. It is also possible that the model learns to distinguish that words structured in this form are questions that need immediate responses. To achieve self-attention, we feed the input into 3 distinct fully connected layers to create the *query, key*, and *value* vectors.

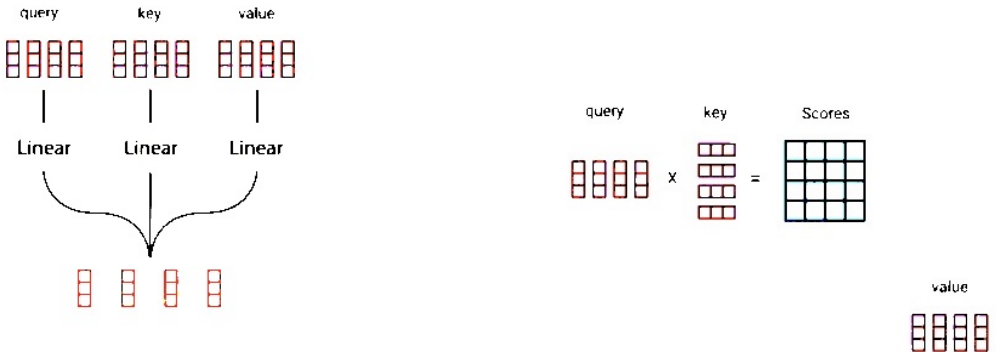


Figure 5: Left: Our input being set up into query, key and values
Right: The dot product between queries and keys

The query key and value concept come from retrieval systems [12].To understand these three important parts, let’s take an everyday example of how a *Youtube* search works using this retrieval system. When a user types a query to search for some video on *Youtube*, the search engine will compare the query against a set of keys (video title, description, tags, etc.) associated with candidate videos within the database, then presents the user with the best videos (values) that matches their search.

In the previous section, we defined that we have created a query, key, and value to achieve self-attention. After creation, we feed the query, key, and value vector through a linear layer, where the queries and keys undergo a dot product matrix multiplication to produce a score matrix as seen in Figure 5. This score matrix determines how much focus a word should have relative to other words. So each word will have a score that corresponds to other words in the time-step. The higher the score the more focus. This is how the queries are mapped to the keys. Figure 6 demonstrates our example showing us our score matrix of the conversational chatbot.

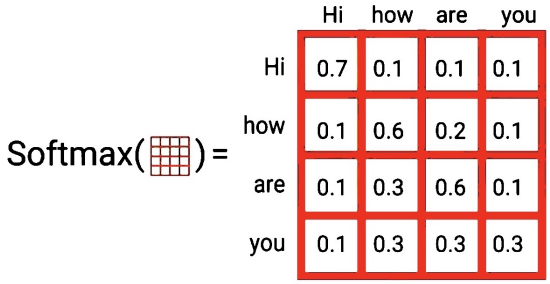
	Hi	how	are	you
Hi	98	27	10	12
how	27	89	31	67
are	10	31	91	54
you	12	67	54	92

Figure 6: Example Problem - Producing a score matrix

However, an issue arises from this as currently constructed. An attention score is not always guaranteed to be a small number, and to manage these attention scores within a finite limit we must come up with an inventive solution.

II. Softmax

In machine learning, specifically in neural networks, we have a *sigmoid function* denoted by the Greek letter σ which is a mathematical function having a characteristic "S"-shaped curve or sigmoid curve. A sigmoid function is simply a logistic function whose y-axis boundaries are bounded within the range $0 \leq x \leq 1$ [10]. Having this unique property of always being within this range has helped machines process much smaller numbers through adding non-linearity to the model, making it much easier to "learn" through the means of backwards propagation.



$$\text{softmax}(x) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{3}$$

Figure 7: Scaling down our matrix using the softmax

In the algorithm we present, we perform a very similar task to the softmax function. Our scores get scaled down in a similar process by dividing the number by the square root of the dimension of query and key. This is to allow for more stable gradients, as multiplying large values can have an exponentially exploding effect. This process is called applying the *softmax*. Similar to the sigmoid function, once we take the softmax of a matrix, we obtain a scaled-down score with a probability value between 0 and 1 to get the attention weights. By doing a softmax the higher scores get heightened, and lower scores are depressed. This allows the model to be more confident about which words to attend to.

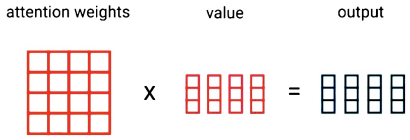


Figure 8: Another dot product between softmax matrix and value vector to obtain output vector

Our final process involving the query, keys, and values is to take our attention weights and perform another dot product with the value vector to retrieve an output vector. The higher softmax scores will keep the value of words the model deems as more important. Likewise, the lower scores will drown out the irrelevant words that should not have much focus. Then we feed the output of that into a linear layer to process. So as shown in this example, the algorithm is especially clever in determining how the model attends to specific words.

III. MULTI-headed Layer

To make this a multi-headed attention computation, the query, key, and value need to be split into N amount of vectors before applying self-attention. The split vectors then go through the self-attention process (the process we just covered) individually. Each self-attention process is called a *head*. Each head produces an output vector that gets concatenated into a single vector before going through the final linear layer. In theory, each head would learn something different therefore giving the encoder model more representation power.

One important question we asked during our development was finding the balance between the number of heads we wish to compute versus the accuracy of our attention-based neural network. In theory, the increased

numbers in heads would only benefit the model's ability to learn, however at what point was this model too taxing and computationally slow to be considered a "good" model. Because we intend to work with a data set that needs to process hundreds of millions of cell data, at what point were we to decide how many heads we should compute for our particular model. We feared that if we had "too little" heads, that our model would not be a very good predictor. So as we move forward we wanted to acknowledge this particular question we had during our development.

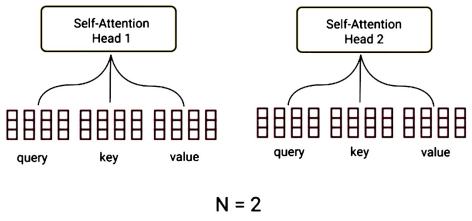


Figure 9: Multi-Headed Layer split into N = 2 "heads"

To sum it up, multi-headed attention is a module in the transformer network that computes the attention weights for the input and produces an output vector with encoded information on how each word should attend to all other words in that particular sequence.

IV. The Residual Connections, Layer Normalization, and Feed Forward Network

Next, the multi-headed attention output vector that we just computed is added to the original positional input embedding. This process is called a *residual connection*. The output of the residual connection goes through a layer normalization.

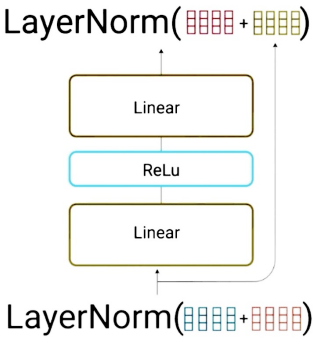


Figure 10: Layer Normalization

Inside the layer normalization, the residual output gets projected through a point-wise feed-forward network for further processing. The point-wise feed-forward network is a couple of linear layers with a Rectified Linear Unit (ReLU) activation in between. In the context of neural networks, ReLU activation function[11] is an activation function defined as the positive part of its argument: where x is the input to a neuron.

$$f(x) = x^+ = \max(0, x) \tag{4}$$

In mathematical terms a ReLU activation function can be described as a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. The rectified linear activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better.

Now going back to the point-wise feed-forward network, the output of that is then again added to the input of the point-wise feed-forward network and further normalized. The residual connections help the network train, by allowing gradients to flow through the networks directly. The layer normalizations are used to stabilize the network which results in substantially reducing the training time necessary. The point-wise feed-forward layer is used to project the attention outputs potentially giving it a richer abstract representation.

V. Encoder Final Remarks

With that, that wraps up the encoder layer. All of these operations are to encode the input to a continuous representation with attention information. This will help the decoder focus on the appropriate words in the input during the decoding process. We can stack the encoder N times to further encode the information, where each layer has the opportunity to learn different attention representations therefore potentially boosting the predictive power of the transformer network.

3.2 Decoder

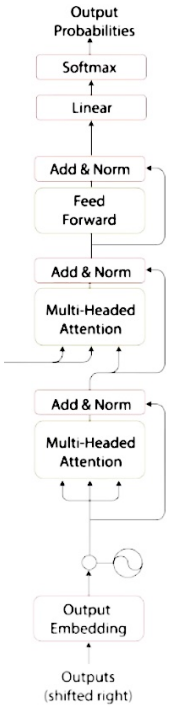


Figure 11: Decoder Architecture

The decoder's job is to generate text sequences from the abstract representation containing attention information. To our surprise, the decoder has a similar architecture in terms of its sub-layer as the encoder. It has two multi-headed attention layers, a point-wise feed-forward layer, and residual connections, and layer normalization after each sub-layer. These sub-layers behave similarly to the layers in the encoder but each multi-headed attention layer has a different job. The decoder is capped off with a linear layer that acts as a classifier, and a softmax to get the word probabilities. So on a high level, the decoder's role is to take our continuous representation and step by step generates a single output while also being fed the previous output.

The decoder is *auto-regressive* meaning it begins with a start token, and it takes in a list of previous outputs as inputs, as well as the encoder outputs that contain the attention information from the input. The decoder stops

decoding when it generates a token as an output.

I. Multi-Headed Layer

As mentioned in the introduction the primary difference occurs within the multi-headed layer. Since the decoder is auto-regressive and generates the sequence word by word, we need to prevent it from conditioning the future tokens. For example, when computing attention scores on the word "am", you should not have access to the word "fine", because that word is a future word that was generated after. The word "am" should only have access to itself and the words before it. This is true for all other words, where they can only attend to previous words.

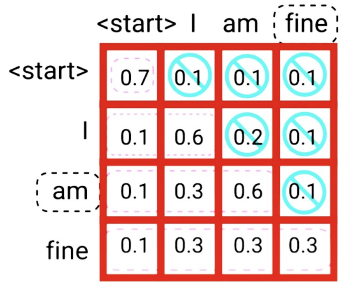


Figure 12: Multi-Headed Layer shouldn't access future tokens

As a result, we need a method to prevent the issue of computing attention scores for future words. Therefore we use a method called *masking*. To prevent the decoder from looking at future tokens, we apply a look-ahead mask. The mask is added before calculating the softmax, and after scaling the scores.

II. Look Ahead Mask

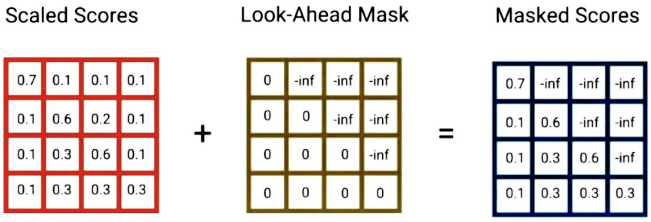


Figure 13: Scaled scores being added to the look ahead mask to retrieve a matrix of masked scores

The look-ahead mask is a matrix that's the same size as the attention scores filled with values of 0's and/or negative infinities. When we add the mask to the scaled attention scores, you get a matrix of the scores, with the top right triangle filled with negativity infinities as shown in Figure 13. Adding these two sets of matrices together helps us obtain our masked scores.

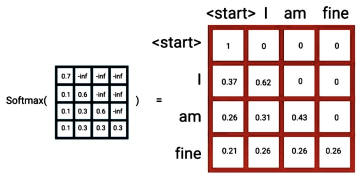


Figure 14: Softmaxing the masked scores

The reason for the mask is because once we take the softmax of the masked scores, the negative infinities get zeroed out, leaving zero attention scores for future tokens. As seen in the Figure 14, the attention scores for “am”, has values for itself and all words before it but is zero for the word “fine”. This essentially tells the model to put no focus on those words.

This masking is the only difference in how the attention scores are calculated in the first multi-headed attention layer. This layer still has multiple heads, that the mask is being applied to, before getting concatenated and fed through a linear layer for further processing. The output of the first multi-headed attention is a masked output vector with information on how the model should attend to the decoder’s input.

III. Second Multi-Headed Layer

For this layer, the encoder’s outputs are the queries and the keys, and the first multi-headed attention layer outputs are the values. This process matches the encoder’s input to the decoder’s input, allowing the decoder to decide which encoder input is relevant to put a focus on. The output of the second multi-headed attention goes through a point-wise feed-forward layer for further processing.

IV. Linear Classifier and Final Softmax for Output Probabilities

The output of the final point-wise feed-forward layer goes through a final linear layer, that acts as a classifier. The classifier is as big as the number of classes we have. For example, if we have 10,000 classes for 10,000 words, the output of that classifier will be of size 10,000. The output of the classifier then gets fed into a softmax layer, which will produce probability scores between 0 and 1. We take the index of the highest probability score, and that equals our predicted word.

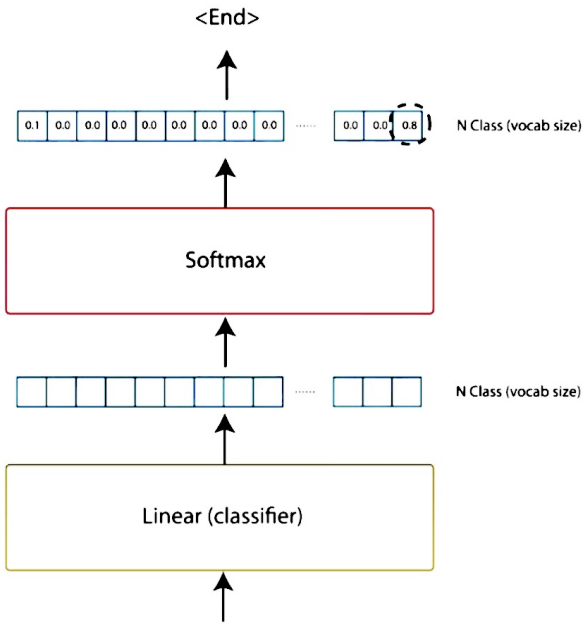


Figure 15: Last steps in decoder to properly determine the predicted word

The decoder then takes the output, add’s it to the list of decoder inputs, and continues decoding again until a token is predicted. For our case, the highest probability prediction is the final class which is assigned to the end token.

The decoder can also be stacked N layers high, each layer taking in inputs from the encoder and the layers before it. By stacking the layers, the model can learn to extract and focus on different combinations of attention from its attention heads, potentially boosting its predictive power.

4 Preprocessing & Training

In this section, we will discuss the parameters we decided to change for our biology-based problem. As it currently stands this model will not necessarily work because our data is not properly prepared to be fed into the *t5-small*. With our main objective to be able to map GEX to ATAC-seq data and vice versa, there are potential architectural issues we had to address in order to make this work. Further, our design will be discussed in this section.

4.1 Data Preprocessing: Binarizing our Data

	AL627309.5	LINC01409	LINC01128	NOC2L	ISG15	C1orf159	SDF4	B3GALT6	UBE2J2	ACAP3	...	MT-ATP8
TAGTTGTCACCCTCAC-1-s1d1	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	...	0.000000
CTATGGCCATAACGGG-1-s1d1	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	2.194758	0.0	...	0.000000
CCGCACACAGGTTAAA-1-s1d1	0.410619	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	...	0.410619
TCATTTGGTAATGGAA-1-s1d1	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	...	0.000000
ACCACATAGGTGTCCA-1-s1d1	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	...	0.000000
...	...	...	...	...	...	...	...	...	...	...	...	...
TAGTAAGCAACTAGGG-8-s3d6	0.000000	0.0	0.0	0.0	1.915288	0.0	0.0	0.0	0.000000	0.0	...	0.000000
TGGTCCTTCGGCTAGC-8-s3d6	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	...	0.000000
CGCTTGCGTTGTTGGA-8-s3d6	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	3.427911	0.0	...	0.000000
ACCCTCCCAGCCAGTT-8-s3d6	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.000000	0.0	...	0.000000
AGTGAACCATCCCGCT-8-s3d6	0.000000	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	1.338799	0.0	...	0.000000

22463 rows × 12160 columns

Figure 16: Our GEX data

One of the major issues with our data is that it is not "legible." If we were to feed the data into our model without any edits, it would most definitely fail because our transformer model does not know how to extract data from the anndata matrix. Anndata is a rather hard data structure to work with overall because it is difficult to manipulate the entries of the matrix. Its primary purpose is to just create a scalable way of keeping track of data and its learned annotations. Therefore one of the first changes we made was converting our anndata into a NumPy array. Our decision to change our matrix data structure was influenced by the fact that it is much easier to manipulate any size matrix with the built-in tools of NumPy.

Although we are now able to manipulate the entries of the matrix, we still have not fixed the legibility issue. It is indeed true that neural networks learn from numbers but with our data as it is, we cannot reliably feed our float numbers and expect a result in the training process. For example in our GEX data at row 2, we can see that CTATGGCCATACGGG-1-s1d1 has expressed the gene UBE2J2 at 2.194758. However, what valuable information does our model grasp in this process? While we do not have an exact answer, we can assume it will learn little because there are no patterns found within the float numbers alone. We overcame this issue with a clever solution.

Our ATAC-seq anndata was presented to us in a binary form. Traditionally binary has been used to express whether something has membership within a specific group. Therefore, to pass in strings instead of integers and floats, we decided to binarize our GEX data. However, it is important to acknowledge that by doing this, we lost some pieces of the cell's information. However, by binarizing our data we can extract the specific gene expression and/or chromatin in its string form to pass it in as a python dictionary for preprocessing. We set the experimental threshold (ϵ) to be over/under 2.0 meaning that if the gene expressed was under 2.0 it would be converted to 0 and if it was 2.0 or above it would be converted into 1. In our example, the CTATGGCCATACGGG-1-s1d1 which expressed the gene UBE2J2 at 2.194758 would be converted to a 1 because 2.194758 is greater than 2.0. So while

we think this is an inventive solution, we also know that gene expression is not a one-dimensional measurement and we lose some important information in the process.

4.2 Data Preprocessing: Setting up our input

Now that our data is binarized, the second part of the preprocessing is to get our data ready to be fed into our Transformer attention model. As mentioned in the previous section, the primary motivation for converting our data into binary was to express membership within a group. In our case, it is to express membership within a certain gene being expressed or a chromatin location at which it occurs. With this information in mind, we came up with a solution to pass in strings as input versus the integers and floats. This strategy was discussed in the previous section. Because the transformer is a NLP model it was important that we made our input into strings so that our model could properly learn in the process.

$$\textbf{\{'En': 'That is good.', 'De': 'Das ist gut.'\}} \tag{5}$$

$$\textbf{\{ 'TAGGTA': 'chr1-633515 chr1-633525 ', 'TAGGTA': 'MT-ND3 MT-ND4 MT-CYB '\}} \tag{6}$$

The transformer takes in input through the use of Python dictionaries. Python Dictionaries are an intuitive data structure because they come with a key and a value. The key is typically some label that we can use to identify a certain entry and the value is some quantity or text that is associated with the key. So in Eq. (5) we can see a NLP example. In this example, the keys are the languages in which we want to perform a machine translation and the values are the English and German phrases that correspond with one another. Similarly, in Eq. (6) the key resembles the cell name and the value is the gene being expressed and the chromatin location. To get all cells with their associated GEX and ATAC-seq dictionaries we made a python function that traversed the whole matrix and extracted the text if and only if their binary value was switched to 1.

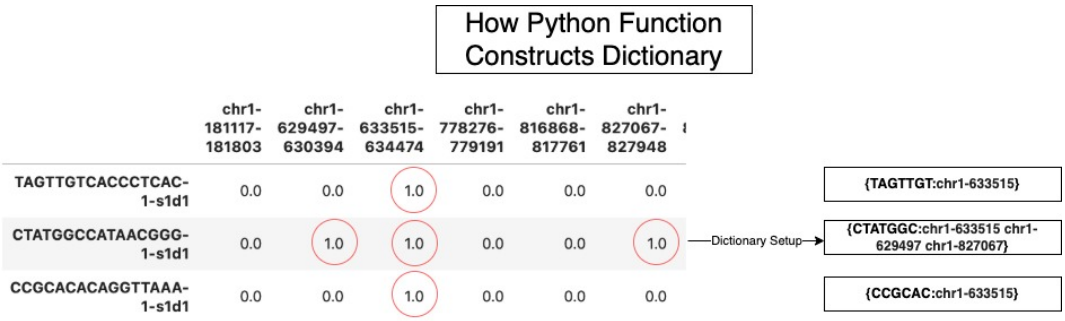


Figure 17: Dictionary Construction

After creating all the dictionaries for each individual cell we stored them into a .txt file as a place to hold all information. This was the final step for the preprocessing portion of our multi-modal predictor program.

4.3 Training

After this initial work, we were able to begin training our model. Because the *t5-small* is a well-established model from *Google*, we simply had to pip install all the model's packages as well as import them into our code and then begin the training process. The t5-small is a pre-trained model, so we had to retrain this model from scratch. What that means for us is that our model relies solely on the 22,463 cells to build its intuition even with the sparse nature of our data. When a data set is sparse it means that the majority of the matrix data are filled with zeros. In our scATAC-seq and GEX matrices, only about 3% of our entries were non-zero. So while we have a decent-sized matrix of 22,463 different types of cells, this may be slightly deficient for fully accurate predictions.

5 Discussion & Future Works

With the model completed we have completed our goal of building a multi-modal prediction method through the use of an attention based transformer model. One of the many accomplishments of this model is its ability to

solve an NP-hard problem. NP-hard problems are defined as having complexities that are computationally taxing. Therefore, it is an amazing feat when these problems get solved. In fact, in the current age of genomics research, state of the art machine learning models have become a requirement to construct any type of analysis type model. Therefore, our t-5 small's ability to perform a search of corresponding sequences through modality prediction will be huge in this field.

However there are some limitations to our research, models, and solutions. We have already discussed the issues caused by binarizing our gene expression data. As explained by binarizing based on a threshold of 2.0, we lose out on all the cells that have some gene expression of less than 2.0 and it is never considered within our model. This filtering out process takes some information away from the cell's true gene expression. Another caveat within this field is that there is no other models to truly compare our model to. Therefore we cannot truly measure the impact of this model due to the unique approach we took in NLP. There are other models that have taken transformers, but they were used to study different objectives (correlation and unsupervised cluster analysis; analysis and integration of data from different experiments and a single measurement type across different samples, e.g., sc-Spatial Transcriptomics; integration of data from SC population, with more than one measurement type, different samples, and a single experiment, etc.). Though this is rather unfortunate, we can also take pride in developing a method that is truly a novel approach to these genomics problems.

And with that we want to acknowledge some future work we have planned for this particular research project. As it currently stands, the execution of our script is not perfect and there are many optimizations we can try to make which may provide better performance. Training the model took roughly a day to compute, and we believe there are also better methods for feeding our transformer model than our current dictionary setup. We also did not take advantage of optimizing our matrix into a sparse matrix to negate nearly 97% of our matrix cells that contain a 0. By looking into some of these more advanced data structures and optimizations, we believe there will be better versions of this model in the near future.

6 Conclusion

Despite the various setbacks of our project, we were able to achieve a new approach for analyzing the multimodal data sets. There is still much to be done however, and we have just scratched the surface of what an approach like ours can achieve. We therefore wish to state some concluding remarks and the future of our research. The emerging field of single cell genomics has a massive need for data analysis and we are very much prepared to continue to integrate data science tools into these enormous datasets. In this paper we were able to take one of these data science tools in the field of natural language processing and transform this machine translation tool into a modality predictor. The mechanics for this model are largely the same, but we did contribute our own niche method to prepare our data for the transformer, as well as train the model from scratch. Our hope for this modality predictor is that we can map corresponding sequences to one another to detect changing levels of gene expression in order to see what effects this has on biological systems. Some applications of varying levels of gene expression is understanding the development of complex disorders and we are widely interested to see how single-cell analysis evolves within the next few years. Building the framework was a good step forward in science but we also understand that there is a lot more we must accomplish in this field to reach these ambitious goals. Fortunately, this field remains exciting as ever and we are continuing to unpack the mysteries of our world.

7 Acknowledgement

I want to thank the many people who contributed to this project and guided me through this work. First, I want to acknowledge Vedavyas, my fellow undergraduate student who has helped bring his expertise in machine learning. I am proud to say we accomplished something novel in the field of single-cell genomics. I also wanted to thank Razvan who helped me tremendously in understanding the foundations of deep learning in just under a month. His advice and vision has widely guided Vedavyas and I for this entire research project. Lastly, I want to thank Professor Jonsson who brought me onto this team of brilliant academics and advised me through the whole journey. This experience was truly one to remember and it has really pushed me to use my Applied Mathematics degree and continue on to the field of computational biology. I am very grateful for it all and believe I would be nowhere without this group of brilliant scholars.

References

[1] Zhang, J., Chiodini, R., Badr, A., & Zhang, G. (2011). “The impact of next-generation sequencing on genomics". Journal of genetics and genomics = Yi chuan xue bao, 38(3), 95–109. https://doi.org/10.1016/j.jgg.2011.02.003

[2] Janocha, K., Czarnecki, W. M. (2017). “On loss functions for deep neural networks in classification". Retrieved August 2, 2020, from the arXiv database.

[3] Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). “Natural language processing: an introduction". Journal of the American Medical Informatics Association: JAMIA, 18(5), 544–551. https://doi.org/10.1136/amiajnl-2011-000464

[4] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. and Liu, P.J., (2019). “Exploring the limits of transfer learning with a unified text-to-text transformer". arXiv preprint arXiv:1910.10683.

[5] Luecken, M.D., Burkhardt, D.B., Cannoodt, R., Lance, C., Agrawal, A., Aliee, H., Chen, A.T., Deconinck, L., Detweiler, A.M., Granados, A.A. and Huynh, S., (2021). “A sandbox for prediction and integration of dna, rna, and proteins in single cells." In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2).

[6] Buenrostro JD, Wu B, Chang HY, Greenleaf WJ (2015). “ATAC-seq: A Method for Assaying Chromatin Accessibility Genome-Wide". Current Protocols in Molecular Biology. 109: 21.29.1–21.29.9. doi:10.1002/0471142727.mb2129s109. PMC 4374986. PMID 25559105.

[7] Reznikoff WS (2008). “Transposon Tn5". Annual Review of Genetics. 42 (1): 269–86. doi:10.1146/annurev.genet.42.110807.091656. PMID 18680433.

[8] Picelli S, Björklund AK, Reinius B, Sagasser S, Winberg G, Sandberg R (2014). “Tn5 transposase and tagmentation procedures for massively scaled sequencing projects". Genome Research. 24 (12): 2033–40. doi:10.1101/gr.177881.114. PMC 4248319. PMID 25079858.

[9] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I., (2017). “Attention is all you need. In Advances in neural information processing systems” (pp. 5998-6008).

[10] Han, J., Morag, C (1995). “The influence of the sigmoid function parameters on the speed of backpropagation learning". In Mira, José; Sandoval, Francisco (eds.). From Natural to Artificial Neural Computation. Lecture Notes in Computer Science. 930. pp. 195–201. doi:10.1007/3-540-59497-3_175. ISBN 978-3-540-59497-0.

[11] Brownlee, J (2019). “A Gentle Introduction to the Rectified Linear Unit (ReLU)". Machine Learning Mastery. Retrieved 8 April 2021.

[12] Jansen, B. J. and Rieh, S. (2010) “The Seventeen Theoretical Constructs of Information Searching and Information Retrieval Archived 2016-03-04 at the Wayback Machine." Journal of the American Society for Information Sciences and Technology. 61(8), 1517-1534.