A Novel Approach to Model Transcription in Humans

by

Cameron Witkowski

Supervised by: Professors Stephen Brown and Kevin Truong

April 2023

Abstract

Proteins play crucial and ubiquitous roles in nature. As I will argue in this thesis, controlling the synthesis of proteins is a first step to harnessing their diverse functions, enabling treatment of critical disease and unlocking new possibilities in biotechnology. Realizing such control over protein synthesis requires a robust predictive model of gene expression and especially, transcription. In this thesis, I critically evaluate the current literature on modeling transcription, highlighting the limitations of the leading two schools of thought. Then, I present a novel approach to bridge the gap between these schools, effectively benefiting from the ideas of both. As a departure point, I preprocess and openly provide a 340-million-example supervised dataset for use in machine learning. Then, I design my own model trained on this dataset, compare the results from a selection of various architectural decisions, and offer a long list of ways these results can be improved. Finally, I openly provide all the code for this project on Github and Google Colab, where other researchers can easily review, challenge, and build upon my work.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Professor Brown and Professor Truong, for their unwavering guidance and support throughout this journey. Their insightful comments and constructive criticisms have been an invaluable compass, steering me towards success. I am truly grateful for their mentorship.

I extend my heartfelt thanks to the Applied Protein Engineering Lab research group, including Teddi and Jay, for their keen attention, valuable feedback, and engaging questions during our meetings. I am appreciative of the collaborative atmosphere fostered within the group.

Special thanks goes to Ayra Thomas, whose editorial expertise and insightful feedback have been instrumental in refining this work. Ayra, your astute critiques, well-informed comments, and comprehensive guidance on effective writing have elevated the quality of this document significantly.

I am grateful for my roommates, Saiyam Patel and Mingshi Chi, who have provided countless invaluable conversations, input, and a sounding board for ideas. Their intellectual curiosity and friendship have seriously enriched this experience.

I would like to express my appreciation to my Mom and Dad, for their constant care, encouragement, and support throughout this endeavor.

Lastly, I would like to thank Simone for engaging in stimulating conversations, showing genuine interest in my work, and for managing to put up with me through this whole process.

Contents

List of Figures

List of Tables

1 Introduction

Proteins are the Pandora's box of biological life, capable of yielding remarkable benefits on the one hand, and profound consequences on the other. Although these complex biomolecules are crucial and ubiquitous, the diversity of their functions is often underappreciated. For instance, digestion, DNA repair, cell signaling, muscle contraction, and molecular transport are well-known processes where proteins perform integral functions; yet, these familiar examples are greatly outnumbered by the undiscovered roles proteins play throughout nature. A comprehensive understanding of protein function (including the ability to reverse engineer them) offers us the potential to unlock a vast array of possibilities, endowing humanity with extraordinary power. Protein engineering, as a field, has only begun to tap into this potential, but its accomplishments thus far provide a glimpse into a future brimming with unexplored opportunities.

Consider the discovery by Dr. Judith Melki's team that deletions or mutations of the SMN1 gene—which halts stop the production of the SMN protein—causes Spinal Muscular Atrophy (a rare disorder leading to muscular atrophy, progressive paralysis, and often death) [1]. Melki's breakthrough led to research into gene therapies and the development of Zolgensma, a treatment which cures the disease in infants under the age of two by introducing a replacement SMN1 gene via the AAV9 virus [2]. Zolgensma, as an engineering triumph, demonstrates how understanding and synthesizing a single protein in specific cells can cure paralyzed children.

Turning attention from Spinal Muscular Atrophy to Covid-19, the SARS-CoV-2 spike protein further demonstrates the impact of advancements in protein engineering. By studying messenger RNA (mRNA), a molecule involved in the process of producing proteins, Katalin Kariko and Drew Weissman's pioneering work paved the way for breakthroughs in vaccine development [3]. When the SARS-CoV-2 genetic sequence became available in December 2020, Moderna and BioNTech quickly harnessed this knowledge to create vaccines [3].

In essence, SARS-CoV-2 vaccines introduce a modified mRNA sequence into human cells which instructs the cells to produce the spike protein. The presence of the spike protein activates the body's immune response, teaching cells to recognize and combat the virus. SARS-CoV-2 vaccines have profoundly impacted the health of millions worldwide, further

underscoring the life-altering potential of advancing our understanding of protein synthesis.

A captivating vision unfolds: to date, the groundbreaking developments in protein engineering offer an indication of proteins' potential to benefit mankind. To this end, a mastery of these complex biomolecules, as tools, would enable us to take advantage of their manifold functions. As tools, proteins are far more than simple wrenches or pliers, however. Ultimately, proteins are instrumental in everything that life does and is, from the first stage of cell division, to the construction of the human body, and the ability for you, my reader, to move your eyes across these very words.

2 Background

2.1 Where do proteins come from?

According to the central dogma of molecular biology, an organism's genome ultimately governs the synthesis of every protein. The genome is the entire set of an organism's DNA, which is the same in all of the organism's cells, and is where protein production begins. Protein synthesis can be summarized in two major steps: (1) transcription, in which a multi-protein complex called RNA Polymerase II (POL II) binds to the DNA of coding genes, then travels along the gene to create mRNA, using the DNA as a template, and (2) translation, in which ribosomes convert the mRNA sequence into a chain of amino acids, which subsequently folds in order to form the final protein structure [4].

When a gene is producing proteins, biologists say it is 'expressed.' Genes can exhibit varying levels of expression, creating different amounts of proteins at different times and in different cells. The intricate mechanisms governing gene expression involve numerous factors interacting in complex ways, such as core promoter elements, transcription factors, histone modifications, DNA Methylation, noncoding RNAs, post-transcriptional modifications, and post-translational modifications. These mechanisms, discussed in section 2.3, are often referred to as 'controlling' or 'regulating' gene expression because they ensure the appropriate amounts of proteins are produced in the correct cells and at the right times. Between the two major steps in the process of protein synthesis, gene regulation primarily occurs at the level of transcription.

2.2 Transcription

Transcription is the initial phase in the production of proteins, beginning when POL II binds to the DNA sequence. During this phase, a specific segment of DNA is copied by POL II, one base at a time, into mRNA. The conversion from genetic information stored in DNA to the intermediate mRNA molecule provides a blueprint for the subsequent construction of proteins [5].

The location in the DNA sequence where POL II binds and begins the transcription process is called the 'transcriptional start site,' or TSS for short. The direction POL II

travels is called the 'downstream' direction, and the opposite direction is called the 'upstream' direction.

2.2.1 Structure and Direction of DNA

The direction of transcription, occurring one base at a time, is determined by DNA's molecular structure. DNA is made of two linked strands of repeating units called nucleotides, resembling a rotating ladder known as a double helix, as depicted in Figure 1. Nucleotides consist of a sugar ring (deoxyribose), a phosphate group, and a nitrogenous base. This base is either adenine (A), cytosine (C), guanine (G), or thymine (T), and is attached to the sugar ring.

The ladder's side rails, called backbones, are each composed of the sugar ring and phosphate group alternating in sequence. At one end of a DNA strand, called the '5' end,' a protruding phosphate is found attached to the fifth carbon of the sugar ring. At the other end, called the '3' end,' a protruding hydroxyl group is found attached to the third carbon. The phosphate-hydroxyl orientation is maintained throughout the DNA molecule, giving it a unilateral direction [6].

Transcription occurs in the 5' to 3' direction of the strand that is copied into mRNA; therefore, the 5' to 3' direction is the downstream direction, and the 3' to 5' direction is the upstream direction.

2.2.2 Regions of the DNA

The region that exists downstream of the TSS contains the base pairs transcribed into mRNA, which includes both exons and introns, as depicted in Figure 2. The word 'gene' usually refers exactly to this region.[1] Introns are sections that are removed by post-transcriptional modifications—described in section 2.3.4—leaving only the exons, which are later converted into polypeptides (chains of amino acids, which form proteins when folded), indicated by the four stages shown in Figure 2. The total length of a gene, including its introns and exons, ranges from a few hundred base pairs to millions.

[1] Occasionally, a single transcribed region may produce multiple proteins (or other functional products) and can be considered to contain 'multiple genes,' but this is merely a matter of definition. Going into a deeper discussion of the various definitions is outside the scope of this document.

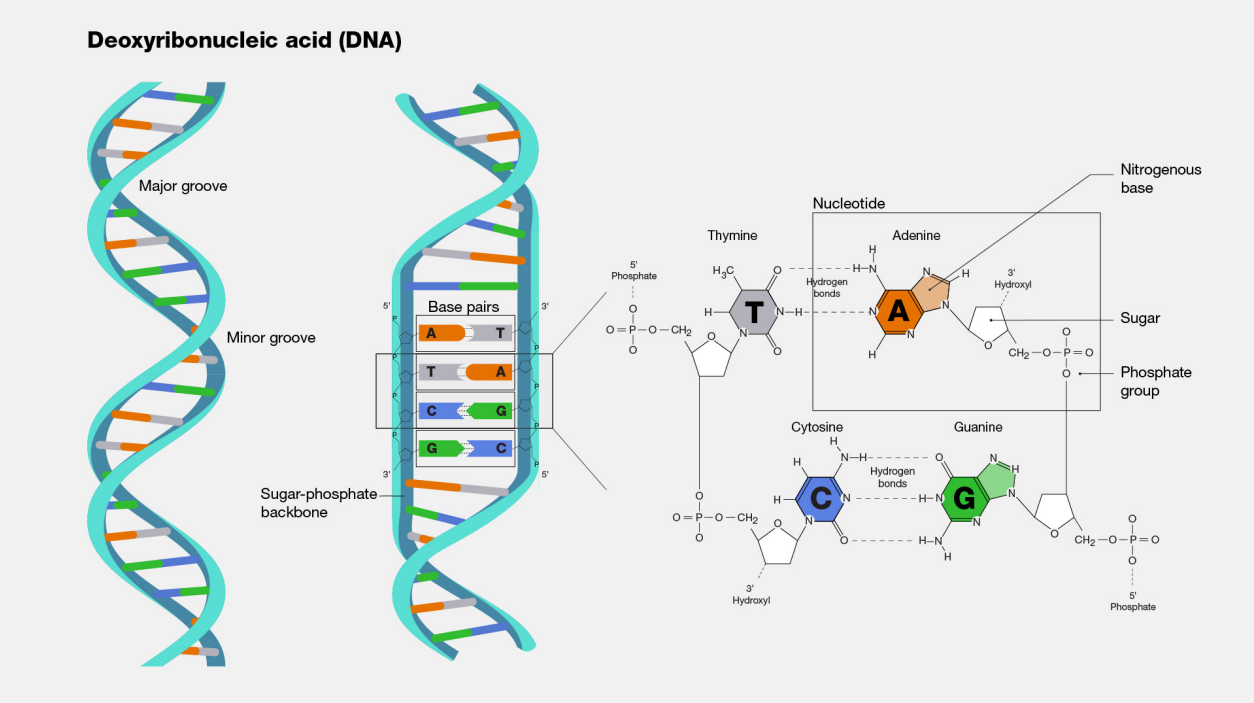

Figure 1: An illustration of the structure and molecular components of DNA. Source: [6]

Conversely, the region that exists upstream of the TSS is crucial for regulating gene expression as it contains binding sites for transcription factors. The upstream region is further divided into the promoter and enhancer regions (although the line between them is blurry). The promoter is shown in Figure 2, but the enhancer is located further upstream. Both regions contain binding sites for transcription factors. Enhancers can be located several thousand bases away from the TSS, while promoters are generally closer, often within 100 bases; however, there is no consensus on a precise border between the promoter and enhancer regions.

2.3 Regulatory Mechanisms

2.3.1 Core Promoter Elements

Core promoter elements are vital components in the DNA sequence that enable POL II to bind and initiate transcription effectively. These elements are specific patterns (also called motifs), consisting of nucleotides A, C, G, and T, that serve as recognizable signals to guide POL II towards the TSS.

These motifs not only indicate the precise location where transcription should begin but

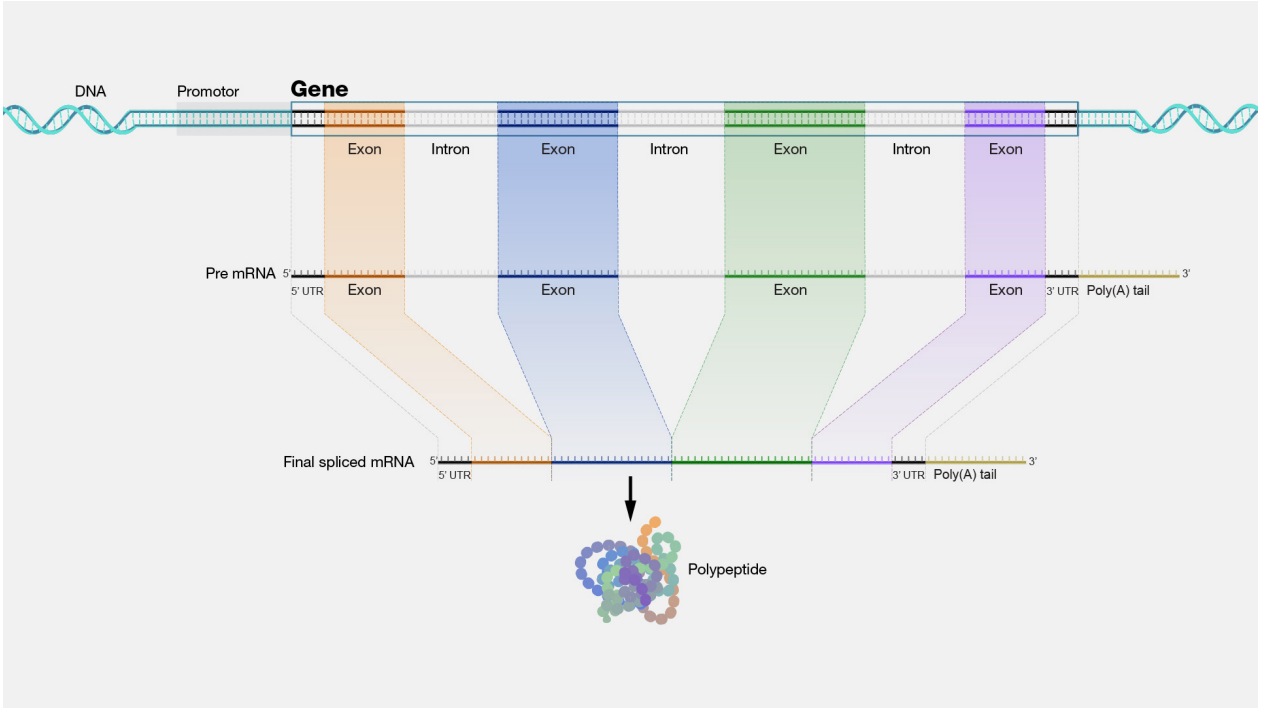

Figure 2: An illustration of the various regions of a gene. Source: [7]

also aid in the proper positioning and orienting of POL II. Core promoter elements vary in length and complexity, and a total of thirteen have been identified for POL II thus far [8]. Not all thirteen are necessary to initiate transcription, however, and different subsets are sufficient for different genes. By recognizing core promoter elements, POL II can accurately and consistently begin the transcription of genetic information from DNA into mRNA.

2.3.2 Transcription Factors

Transcription factors regulate gene expression by binding to the promoter or enhancer and affecting POL II's ability to bind and initiate transcription. Thus, as their name implies, they are *factors* in transcription. Similar to POL II, transcription factors also must recognize specific motifs in the DNA sequence in order to bind. Each transcription factor has its own motif or set of motifs that it looks for.

While some transcription factors help recruit POL II to the TSS (appropriately named 'activators') other transcription factors block such recruitment ('repressors'). The majority of transcription factors, however, interact with coactivators, corepressors, and each other in complicated ways. Coactivators and corepressors are proteins or other molecules that interact

with transcription factors and affect their ability to bind to promoters or enhancers.

In the human genome, around 1,600 transcription factors have been identified thus far, which often work together, in opposition, or in a combination of both in order to fine-tune gene expression [9]. Because of the combinatorial explosion of potential interactions which can occur between transcription factors, deciphering the mechanisms of these factors makes for a complex and challenging task.

In Figure 3, a simplified representation of POL II, motifs, and transcription factors is presented. POL II may bind to four potential genes, A, B, C, and D. The greater number of transcription factors which bind to gene D make it more likely that this gene will be transcribed. With a rudimentary understanding of the mechanism of transcription factors, it is now possible to appreciate an additional layer of complexity.

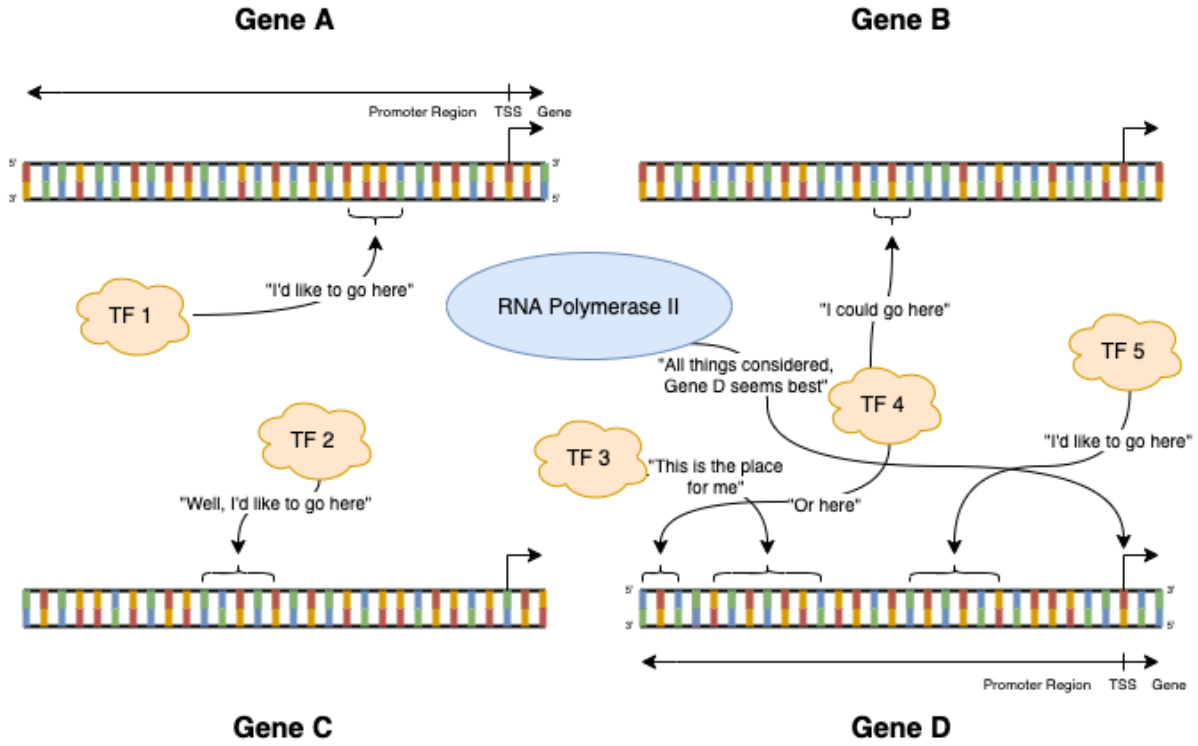

Figure 3: A simplified illustration of the influence of transcription factors.

2.3.3 Histone Modifications

Histone modifications are another crucial gene regulatory mechanism, but to understand them, it is necessary to first revisit the structure of DNA. The depiction of DNA in sec-

tion 2.2.1 and Figure 3 is a simplification, insofar as DNA does not appear in straight linear pieces in eukaryotic organisms (animals, plants, and fungi). In eukaryotes, DNA instead forms a complex, coiled structure known as chromatin, which must be understood in order to appreciate the role that histones and histone modifications play in transcription [10].

Imagine DNA resembling a string that wraps around spools, as depicted in Figure 4. This spool is called the octamer core, oct- meaning eight, since it is made of eight proteins called histones. This wrapped structure is known as a nucleosome, and approximately 147 base pairs of DNA coil 1.75 times around each core. The section of DNA that connects nucleosomes together is called linker DNA, which, importantly, is more exposed than wrapped sections. There are five main types of histones: H1, H2A, H2B, H3, and H4. Four of these histones—H2A, H2B, H3, and H4—form the histone octamer at the core of the nucleosome. Each octamer contains two copies of each of these four histones. The fifth histone, H1, acts as a linker protein, sealing the nucleosome and preventing the DNA strands from unwinding. For a sense of scale, each base pair measures 0.34 nm along the length of the DNA, the DNA double helix itself is about 2 nm wide, and each nucleosome is about 11 nm wide. Collectively, histones form nucleosomes to efficiently organize and compact DNA within the cell nucleus [11].

Nucleosomes form a tightly coiled loop in a structure called the solenoid model. The solenoid is another helix, which contains six nucleosomes in each 360-degree turn with a diameter of 30 nm wide, shown on the left side of Figure 4. As such, the DNA double helix wraps around histones in a 1.75-turn helix to form nucleosomes, subsequently creating the turns in the solenoid helix. Thus, DNA is shaped as a helix within a helix within a helix [10].

To put things in perspective, a human chromosome, made up of millions of nucleosomes, can measure several thousand nanometers. While these numbers vary significantly, the cell nucleus, which contains all 46 chromosomes, has a diameter of about 10,000 nm, or 10 micrometers [12]. The entire cell, on the other hand, measures roughly 50,000 nm, or 0.05 mm across. This means that a typical cell is approximately ten times smaller than a single grain of salt. The entire cell, on the other hand, measures roughly 50,000 nm, or 0.05 mm across; therefore, a typical cell is approximately ten times smaller than a single grain of salt.

Chromatin's tightly coiled structure poses a challenge for transcription factors that require access to bind to target DNA sequences. More condensed chromatin limits access to binding

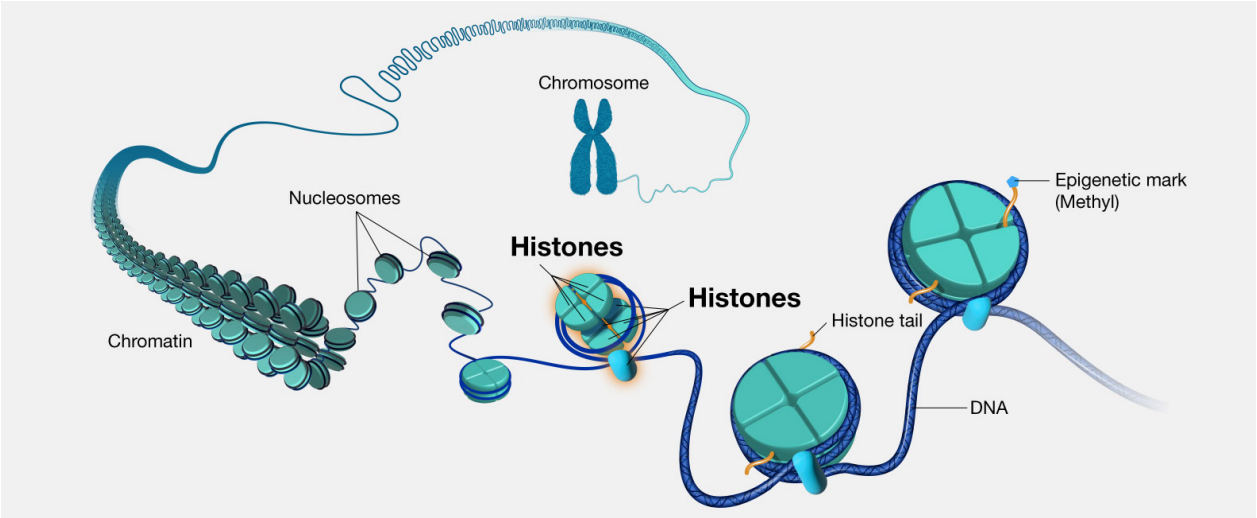

Figure 4: An illustration of the structure of chromatin. Source: [11]

sites, while more open chromatin facilitates it. Histone modifications play a crucial role in addressing transcription factors' challenge binding. By altering the way DNA wraps around histones, histone modifications either create openings or reinforce barriers in the chromatin structure, thus affecting the accessibility of DNA to transcription factors.

More specifically, histone modifications involve molecular changes to histone proteins through the addition or removal of certain functional groups. Such changes include acetylation, which adds an acetyl group to histones; methylation, which adds a methyl group; and phosphorylation, which adds a phosphate group. Histone modifications directly influence the bond between histone proteins and DNA to make the DNA molecule more accessible, and subsequently influences the ease by which transcription factors access and bind to specific sequences. To sum up, the interplay between chromatin structure and histone modifications determines which regions of DNA are accessible to POL II and to transcription factors, ultimately influencing the outcome of transcription and gene expression [13].

2.3.4 Other Mechanisms

In the preceding sections, I outlined transcription factors and histone modifications in detail, as they are fundamental components of the transcription process. Still, this outline does not offer a complete picture of transcription regulation, as numerous other elements also contribute to gene expression.

For the sake of completeness, I will now briefly discuss DNA methylation, noncoding

RNAs, and post-transcriptional and post-translational modifications, but a more detailed analysis of these mechanisms is outside the scope of this document. The limited focus of this document is intended to provide a clear, easily comprehensible foundation for future research. This structure can be built upon by other researchers by incorporating additional factors and complexities for a more comprehensive and accurate understanding of transcription regulation. Thus, the mechanisms that I touch on next do not appear in the model I develop in sections 4 and 6.

DNA Methylation DNA methylation is a crucial gene regulation mechanism that involves the addition of methyl groups to cytosines within specific DNA sequences called CpG dinucleotides. A CpG dinucleotide simply consists of a cytosine (C) base followed by a guanine (G) base in the 5'-3' direction along the DNA strand. CpG sequences often cluster together to form regions known as CpG islands, which are found in around 60% of gene promoters.

Methylation of CpG islands in promoter regions alters the accessibility of DNA to the transcriptional machinery,[2] inhibiting the binding of transcription factors and other regulatory proteins that are necessary for initiating gene transcription. DNA methylation effectively silences or reduces gene expression, working in concert with other gene regulation mechanisms, such as transcription factor binding and histone modifications to control transcription [14].

Noncoding RNAs Another important factor in the regulation of transcription and gene expression is the influence of noncoding RNAs. Unlike mRNA, noncoding RNAs can be produced by POL I, II, or III, and do not code for proteins. Noncoding RNAs can be thought of as 'leaving the protein factory early'—instead of going on to become proteins, they are involved in a number of cellular processes, including the regulation of gene expression in various ways.

The three main types of noncoding RNAs are microRNAs, small interfering RNAs, and long noncoding RNAs. Each of these three types has unique functions that contribute to the overall process of gene regulation. MicroRNAs control gene expression by binding to target mRNAs which can lead to mRNA degradation or inhibition of protein translation. Small

[2]In this document and much of the literature, mechanical jargon is used to portray transcription and gene expression processes as 'machines' that produce proteins (or other functional products). I call this perspective of genes as machines the mechanistic paradigm, and it is closely reexamined in section 5.2.

interfering RNAs also play a role in gene silencing, primarily through a process called RNA interference. Long noncoding RNAs are more diverse in function, participating in processes including transcriptional and post-transcriptional regulation [15].

Post-Transcriptional Modifications After exploring how chromatin structure and histone modifications impact transcription, another key aspect to consider is post-transcriptional modifications. Post-transcriptional modifications occur after the transcription process and include RNA splicing, capping, and polyadenylation, among others.

RNA splicing involves the removal of non-coding sequences (introns) and the assembly of coding sequences (exons) to create mature mRNA. Alternative splicing generates various mRNA transcripts from a single gene, leading to protein isoforms with distinct functions. Isoforms are protein variants resulting from the same gene through alternative splicing [16].

Capping and polyadenylation, on the other hand, protect the mRNA molecule and assist in its transport, stability, and translation. Besides the few examples of post-transcriptional modifications presented here, many more forms exist [17].

Post-Translational Modifications After mentioning post-transcriptional modifications, it's essential to mention another crucial aspect of gene regulation: post-translational modifications. These modifications happen after the translation process, altering the structure and function of proteins. Events such as phosphorylation and ubiquitination are common examples of post-translational modifications.

Phosphorylation, the most common, involves the addition of a phosphate group to a protein, often leading to changes in its activity or interaction with other molecules. As a result, phosphorylation plays an important role in the regulation of many cellular processes. Ubiquitination, on the other hand, involves attaching a small protein called ubiquitin to a target protein, typically marking it for degradation by the proteasome—a sophisticated protein complex with the job of decomposing marked proteins. These are just two examples of post-translational modifications, but there are many more types [18].

3 Aims

The primary goal of this document is to develop a predictive mathematical model for transcription. Generally speaking, a model serves as a simplified representation of a complex phenomenon or system, with the purpose of facilitating understanding, analysis, or prediction of its behavior. Models are often just informal or abstract representations used as thinking aids. Such aids include descriptions, pictures, diagrams, conceptual frameworks, and qualitative narratives. These informal representations provide valuable starting points for comprehending complex systems but lack the rigor, quantitative precision, and predictive power that can be achieved through mathematical models.

In the present context of modeling transcription, the focus is on creating precise, mathematical models rather than informal descriptive or visual representations. By developing a mathematical model, the intricate processes of transcription obtain a structured, numerical representation. In this way, a richer understanding of the underlying mechanisms can be achieved, and accurate predictions about gene expression can be made.

The decision to develop a predictive model of transcription is motivated by several factors. First, predictive models are well-defined, with clearly specified inputs and outputs, allowing for the factors that influence transcription to be accounted for systematically. Second, predictive models can be easily tested and validated by comparing their predictions against experimental data, allowing researchers to assess their models and refine them as needed. This iterative process of testing and refinement ultimately leads to more reliable and informative models. Third, predictions made by these models have practical utility, as they can guide experimental design, inform the development of targeted therapies, or help identify novel gene regulatory mechanisms or transcription factors.

The greatest benefit of a predictive model, however, is that it is relatively straightforward, in principle, to transform it into a generative model. Instead of predicting outcomes, generative models enable the creation of new data. In the case of transcription, a generative model could be used to create synthetic promoter DNA sequences with desired cell-specific expression levels by identifying key transcription factors and manipulating binding sites within the promoter DNA sequence. The predictive model can then verify that these synthetic promoters will, in theory, have the desired expression levels. Although this process is complex and

challenging in practice, the comprehensive understanding provided by a precise mathematical model could reshape what is possible.

If one needs any further convincing of the principle that predictive models can be transformed into generative ones, they need look no further than the recent success of ChatGPT. This groundbreaking technology developed by OpenAI has engaged millions with its unprecedented ability to intelligently generate text in response to user prompts. Despite its extensive capacity for content creation, ChatGPT is powered, at its core, by a model of language which does nothing but predict the next word (or token) to appear in a sequence. [19].

Thus, the design of a precise, predictive model of transcription is a necessary step in deepening our understanding of proteins, and would represent small but tangible progress toward mastery over proteins as tools. The next section will develop the theory necessary to design such a model.

4 Theory

4.1 Structure

At the highest level of abstraction, we wish for our model to take a set of inputs, x, and output a prediction of the corresponding level of transcription, y_{pred}, which should closely align with the true level of transcription, y_{true}, measured through biological experiments. In mathematical notation:

$$y_{pred} = f(x) \qquad x, y_{true} \in \mathbf{X} \tag{1}$$

where f represents a mapping from model inputs to output, x is the input data for one prediction, y_{pred} is a numeric estimate of the level of transcription, y_{true} is the true numeric value of the level of transcription, and $\mathbf{X}$ is the dataset, from which the inputs and the true, experimentally measured output is drawn.

Ideally, the 'level of transcription' should represent the absolute frequency[3] of POL II binding to the TSS and initiating transcription. The frequency of transcription can be thought of as a measure of gene expression, since a greater transcription frequency corresponds to greater mRNA production and protein synthesis. The data and units of measurement for the level of transcription will be described in section 6.1.1.

4.2 Uncertainty

We now wish to extend the model in Equation 1 by incorporating uncertainty, enabling predictions to be made with varying degrees of confidence. Incorporating uncertainty not only indicates when predictions can be relied upon but also allows us to identify instances of significant uncertainty, enabling further refinement and iterative enhancement of the model's performance.

In Equation 1, the model only outputs a single value, such as "the level of transcription

[3]Here, the word absolute is used to indicate that this frequency is not measured relative to other genes or other cell types, but reflects the actual number of transcriptional events per unit of time.

will be 27," representing its best guess for the transcription level; however, this approach doesn't capture the uncertainty or confidence in that prediction, nor the likelihood of any other outcomes.

To address this, we must modify our model's output. Instead of providing a single point estimate, the model will output a probability density function, which maps a different likelihood to every possible level of transcription. To simplify matters, we will assume that the level of transcription is continuous and can take on any value from 0 to infinity. In section 6.3, I will describe the particular type of probability distribution used.

By integrating this probability density function, we can obtain probabilities of the level of transcription being within any range. Such probabilities might look like: "there's a 20% chance of the level of transcription being between 0 and 20, a 30% chance of it being between 20 and 100, and a 50% chance of it being greater than 100." In this way, the probability distribution contains more information than the point estimate. A point estimate can still be recovered from the distribution, however, simply by taking the distribution's mean, median, or mode.

Specifically, the probability distribution will be a conditional probability distribution given the model's inputs x. In mathematical notation:

$$P(y_{pred} \mid x) = f(y_{pred};\ x) \qquad x, y_{true} \in \mathbf{X} \tag{2}$$

where f now represents a mapping from model inputs to probability density functions. Now, in contrast to Equation 1, y_{pred} is essentially another input to our model. You can think about it like this: we may now ask our model the likelihood of any particular level of transcription from 0 to ∞, and we specify this by setting y_{pred} to that level. Thus, $y_{pred} \in [0, \infty)$. Since P is a probability distribution, we must also have $\int_0^\infty P(y_{pred} \,|\, x)\, dy_{pred} = 1$. Thus, Equation 2 forms the most abstract representation of our predictive model of transcription.

4.3 Model Inputs

Having considered the model's outputs, I now turn attention to its inputs. The background provided in the section 2 lays the foundation for this discussion. The selection of inputs not only marks a divide in the literature but also signifies a fundamental difference in perspectives on transcription which will be explored further in section 5.2.

The rationale behind these inputs is the core essence of this work, representing its primary contribution to the field of research on transcription. In an original manner, this work bridges the gap between the two divergent schools of thought discussed in section 5.2.

The inputs given to our model constitute the information that it will base its predictions on. For our model to bear any significant precision, this information must include all the different factors which are known to influence transcription, as elaborated in section 2.3. If any important factor is missing, the model will not have enough information to decide on a level of transcription with confidence.

4.3.1 DNA Sequence

The first critical factor, necessary for transcription to occur, is the binding of POL II to a specific region of the DNA sequence. Recall from section 2.3.1 that POL II binds by the recognition of core promoter elements near the TSS.

A second crucial factor is the binding of transcription factors to motifs along the DNA sequence. Recall that motifs are specific patterns (of A, C, G, and T) in the DNA, similar to core promoter elements but appearing all throughout the promoter and enhancer regions.

The commonality between these two factors is the DNA sequence—specifically near the TSS and throughout the promoter and enhancer regions. Thus, the simplest and most natural way to inform our model about potential binding sites is to give it the DNA sequence. The model will then bear the responsibility of searching for the patterns relevant to POL II and transcription factors and using these in its predictions. The particular representation I use will be explained in section 6.1.2.

4.3.2 Transcription Factor Levels

Understanding the potential binding sites of transcription factors on the DNA sequence is insufficient for predicting transcription; it is also necessary to identify which transcription factors are actually there. For a transcription factor to bind to the promoter or enhancer of some gene and affect transcription, it needs to actually be in the cell nucleus at that moment.

There is more than just one copy of each transcription factor in each cell, however. Multiple copies of these factors exist, and their presence in the cell nucleus can be measured by their concentration. The concentration of transcription factors in the cell nucleus influences their likelihood of binding to promoters & enhancers. Certainly, transcription factors with a concentration of zero will exhibit zero binding. Transcription factors with a high concentration, on the other hand, often bind to promoters & enhancers with high frequency. Between these two extremes, one would reasonably expect that, all else equal, transcription factors with higher concentrations will more frequently bind and exert a more significant effect on transcription.

The upshot is that transcription factor concentrations are critical to include in our model's inputs. Ideally, we would give the model accurate measurements of these concentrations for all 1,600 or so transcription factors simultaneously, and specifically within the cell where the model is predicting transcription levels. This precise synchronization and location specificity are crucial to ensure that the model's predictions accurately reflect the intricate and dynamic interplay of transcription factors that govern gene expression.

In order to satisfy this strict synchronization requirement, it is essential to recognize that transcription factors, as proteins, are created by the very same transcriptional and translational machinery under examination. Consequently, we can focus on quantifying the production of transcription factors by measuring gene expression for genes that synthesize them, in order to estimate their concentration. This approach renders the measurements attainable and practical, but requires the assumption that the concentration of transcription factors is uniform throughout the cell nucleus. This approach to estimating transcription factor concentrations is illustrated in Figures 5 and 6, where gene A produces an activator that increases expression of gene B. In Figure 5, the low expression of gene A leads to low expression of gene B. In Figure 6, on the other hand, the high expression of gene A leads to

high expression of gene B.

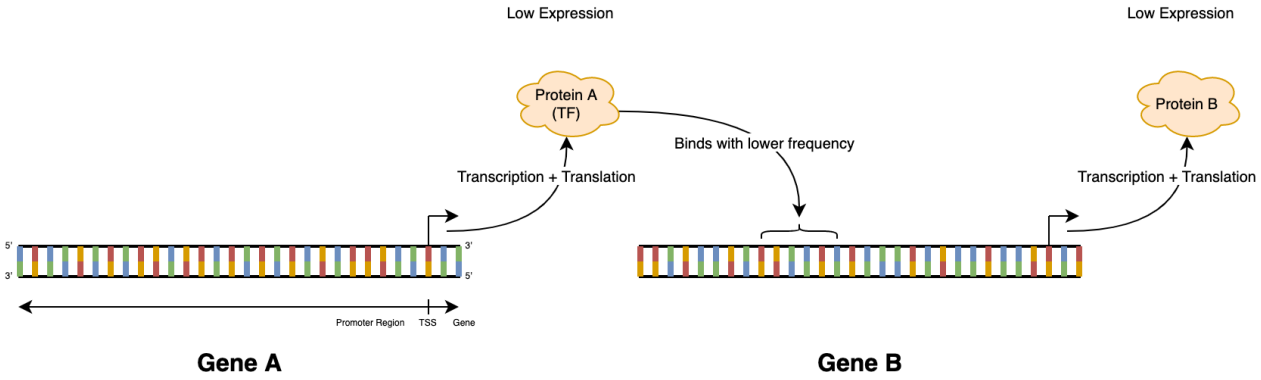

Figure 5: An illustration of the effect of one gene on another, interacting through a single activator (a).

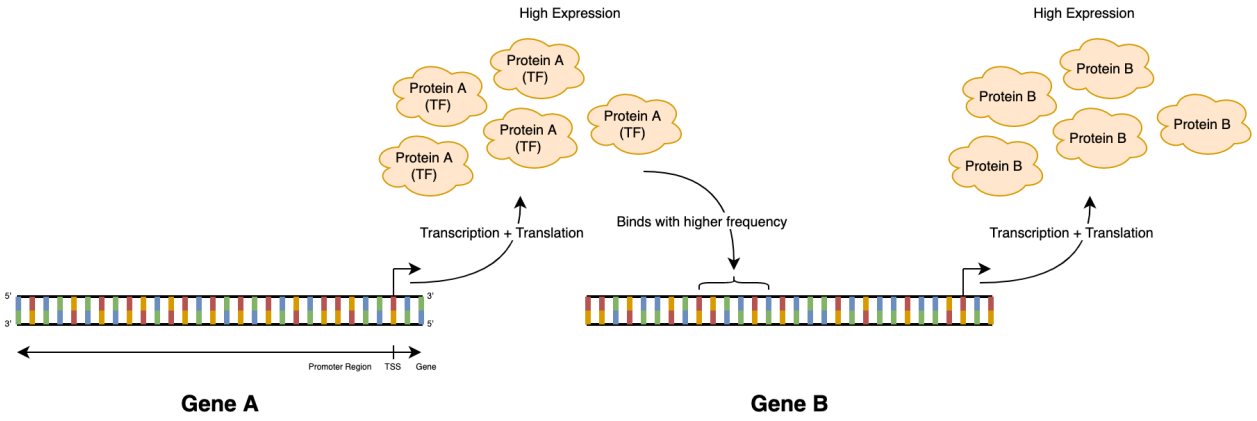

Figure 6: An illustration of the effect of one gene on another, interacting through a single activator (b).

4.3.3 Histone Modifications

Histone modifications significantly impact gene expression, as described in section 2.3.3, making their inclusion in our model essential. Ideally, we would obtain measurements of histone modifications that are synchronized (in the same sense described in section 4.3.2) and within the same cell type as other gene expression measurements. This synchronization and location specificity are essential for accurately capturing the complex interactions that regulate gene expression, just like with transcription factors levels.

Synchronized and cell-specific measurements of histone modifications are unavailable, however, and unlike the case for transcription factors, no feasible workaround exists. As a

result, the data for histone modifications are inherently flawed, which may lead to inaccuracies and errors in the model's predictions. This unfortunate fact is explored in more detail in section 6.1.6.

4.3.4 Conclusion

While the DNA sequence, transcription factor levels, and histone modifications are all crucial inputs, they are insufficient for a comprehensive transcription model, as numerous other factors contribute to the process.[4] This document serves as a first step in introducing the approach, aiming to provide a foundation for future research. In the future, other researchers may be able to build upon this work, incorporating additional factors and complexity to develop more advanced and accurate models of transcription. In this document, the scope is intentionally limited to the initial approach for the sake of clarity and simplicity.

In section 5, which follows, I will focus on relevant research and highlight various approaches to modeling transcription, offering valuable context and insights. Reviewing the literature will also provide an opportunity to compare the general model framework just outlined with other researchers' models and approaches. We will develop our model of transcription further in section 6.

[4] In my previous work, I incorporated ENCODE candidate Cis-Regulatory Elements (cCREs) as an additional input; however, I have chosen not to include them here. ENCODE cCREs are regions in the DNA that display DNase I sensitivity (serving as an indicator of open chromatin structure) and are characterized by histone modifications or the binding of CTCF (a protein involved in transcription regulation) [20]. These cCREs serve as a more consolidated data source, providing a coarse representation of potential regulatory influences. Since the aim of this document is to develop a comprehensive model of transcription, specifically capturing intricate molecular interactions, using cCREs would not align with my purpose.

5 Literature Review

5.1 History

Before jumping straight into the latest predictive models, it is important to appreciate the historical developments which led to the modern understanding of transcription.

Edward Lewis's investigations into the bithorax complex in Drosophila (fruit fly) embryos during the 1940s and 50s laid the foundation for understanding gene regulatory mechanisms [21]. Lewis's groundwork paved the way for Jacob and Monod's discovery of the operon in prokaryotic E. coli [22] [23]. The operon is a cluster of genes under the control of a single promoter that end up being translated into multiple proteins. Jacob and Monod's findings on the lactose and tryptophan operons were crucial for future research on transcriptional regulation.

In 1964, Vincent Allfrey discovered that histone modifications played a role in transcriptional control [24]. However, the most significant catalyst for research into eukaryotic transcriptional mechanisms was the discovery of RNA Polymerase in 1960 by Charles Loe, Audrey Stevens, and Jerard Hurwitz, along with the identification of its three forms (POL I, II, and III) [25] [26].

The discovery of POL I, II, and III led to the identification of core promoter elements and the systematic mapping of these elements. The first gene-specific transcriptional activator in eukaryotes was discovered in 1979 [27]. This activator does not interact with POL II directly, which sparked a paradigm shift as direct interaction was previously thought to be the sole mechanism.

Another critical finding was the identification of the mediator complex in 1991, which 'mediates' the interaction between transcription factors and POL II [28]. The role of DNA bending proteins was also uncovered by the collaborative efforts of many, completing the general picture of eukaryotic transcription.[5] Today, our understanding of transcription has been built upon these critical breakthroughs, each discovery representing a stepping stone towards unraveling the intricate mechanisms governing the expression of genes.

[5]Note, in the model I develop in section 6, neither the mediator complex nor DNA bending proteins are accounted for, but I list these as potential areas for improvement in section 9.

5.2 Breakdown of Approaches

For the most part, predictive models of transcription and gene regulation in the literature fall into two broad categories, which are necessary to explain before presenting them. I refer to the first category as the cybernetic paradigm and the second as the mechanistic paradigm.

5.2.1 The Cybernetic Paradigm

The cybernetic paradigm consists of models which use only expression levels as input. Using inputs in this way is justified by the discussion in section 4.3.2. By using expression levels of one gene to predict the expression levels of other genes, these models essentially seek to unveil the network of interactions between genes. As a result, in the literature these models are not usually referred to as 'predicting transcription' or 'predicting expression,' but instead called 'gene network inferencing,' or 'gene regulatory networks.'

Cybernetic models do not directly use any information specifically related to each gene. As indicated in Figure 7, these models only consider how gene A tends to influence genes B, C, etc., how gene B tends to influence genes A, C, etc., and so on and so forth. In other words, cybernetic models construct a network showing the interactions between genes, but do not explain the factors driving these interactions.

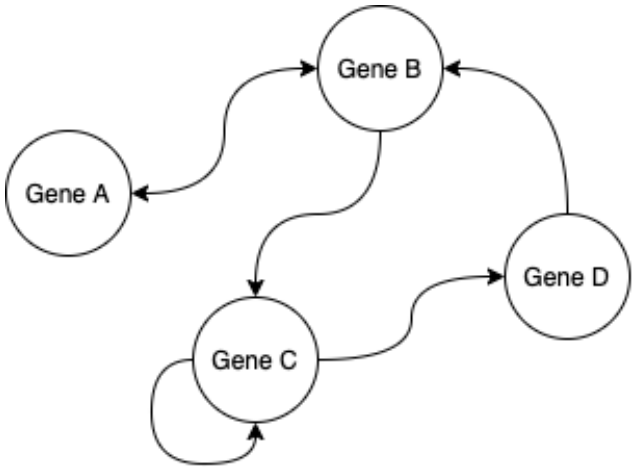

Figure 7: An illustration of the cybernetic approach to modelling gene expression.

While the cybernetic approach seems promising on the surface, there is a serious methodological concern about all models in this category: based on the interaction map alone, it is not possible to predict how a new gene would fit into this network. The key to making

accurate predictions in new scenarios lies in understanding the underlying mechanisms governing phenomena. When a model successfully explains these mechanisms, it demonstrates its generality, which is a vital aspect of all scientific models. Thus, models that are not generalizable do not successfully explain the mechanisms underlying phenomena. In the context of gene regulatory networks, understanding the mechanisms governing gene regulation, such as the interaction of transcription factors, histone modifications, etc., is essential for accurately predicting the behavior of new genes within the network.

Gene regulatory networks, however, fail to provide this level of understanding. As a result, they are not generalizable and only offer a superficial view of the relationships between genes. This lack of generality and understanding of the underlying mechanisms makes gene interaction networks incomplete and unsatisfactory tools for studying the complex processes of transcription and gene regulation.

In the context of complex systems, the term 'black box' refers to a component or process whose internal workings are unknown or ignored, with a focus solely on the relationship between its inputs and outputs. When applied to gene regulatory networks, this means that genes are treated as black boxes, with attention given to their interactions with other genes while disregarding the internal processes and molecular mechanisms that govern their behavior.

Cybernetics is a field of study that examines communication, control, and feedback within complex systems, aiming to understand the system as a whole rather than investigating individual components in isolation. The term 'cybernetic' is appropriate to describe gene regulatory networks because they share this focus on system-level interactions and regulatory effects. By emphasizing the interplay between genes and their influence on each other's expression levels, this approach mirrors the cybernetic perspective of analyzing control and feedback loops within a system.

The cybernetic approach to understanding complex systems is rooted in the idea that these systems can often be better understood by examining the patterns and dynamics of their interactions, rather than dissecting individual components. In the context of gene regulatory networks, this cybernetic approach provides valuable information about the overall organization and dynamics of gene regulation, but it fails to capture the underlying molecular mechanisms driving gene interactions.

5.2.2 The Mechanistic Paradigm

The mechanistic paradigm, on the other hand, looks inside the black box, but ignores the broader, system-level interactions between genes. As indicated in Figure 8, presented in contrast to Figure 7, this paradigm consists of models which ignore the gene network and take only gene-specific information as input, such as the local DNA sequence, histone modifications, etc. Mechanistic models are more commonly referred to as 'predicting transcription' or 'predicting expression' in the literature.

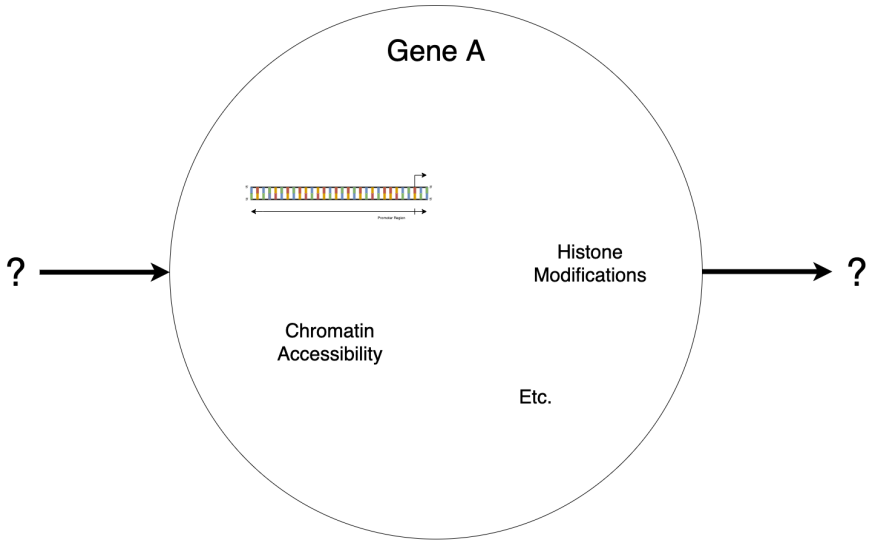

Figure 8: An illustration of the mechanistic approach to modelling gene expression.

Mechanistic models are set up for failure by not being provided the concentration or expression level of transcription factors. Even in principle, mechanistic models cannot explain how varying concentrations of different transcription factors would affect transcription, because they focus on static, gene-specific information.

In accordance with the cybernetic paradigm's shortcomings, the importance of generality for models remains crucial. A mechanistic model that cannot account for varying concentrations of transcription factors lacks the generality needed to make accurate predictions in new cell environments. Consequently, these models fail to uncover the true mechanisms underlying transcription, providing detailed molecular insights but not a comprehensive understanding of gene regulation.

The term 'mechanistic' originates from the concept of a 'mechanism' or 'machine,' which underscores the idea that these models strive to dissect and comprehend the intricate com-

ponents driving transcription and gene expression, similar to the parts of a machine working together. By focusing on the specific molecular factors and their interactions, however, the mechanistic paradigm overlooks the broader context and conditions in which the machine operates. This is all in contrast with the cybernetic paradigm, which adopts a more holistic, system-level perspective, treating this machinery as a black box.

Thus, the literature on modeling transcription can be divided into two broad categories according to the model's inputs, as depicted in Figure 9. This figure also showcases relevant categories of both mechanistic and cybernetic models that will be explored in sections 5.3 and 5.4 which follow. Although this classification is not all-encompassing and has notable exceptions to be discussed, it serves as a helpful framework for comprehending the research on modelling gene regulation and this work's position within it.

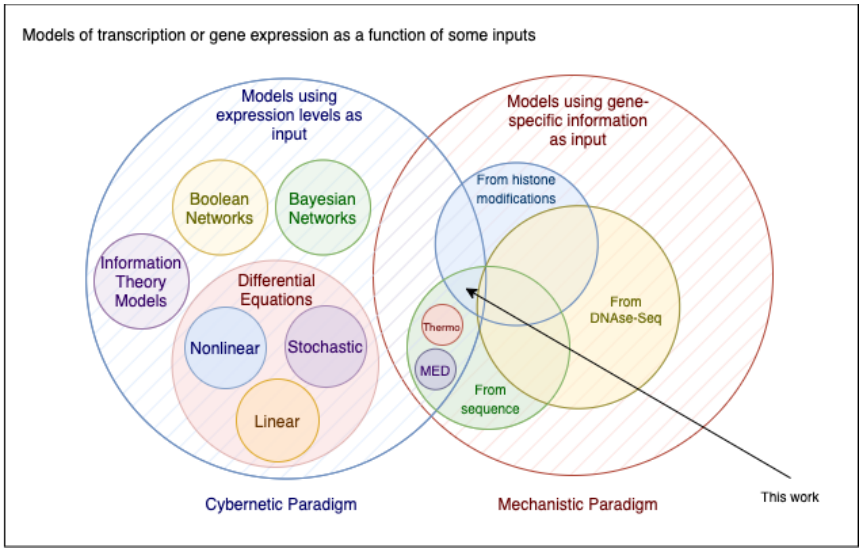

Figure 9: A breakdown of the literature on modelling transcription.

5.3 Cybernetic Models

In this section, we will explore cybernetic models and their unique contributions to the study of transcription. By adopting a more holistic, system-level perspective, these models provide valuable insights into the complex interplay between genes. A brief selection of particular models will now be described, but a more complete review is given by Hecker et al. and more currently by Banf and Rhee [29][30]. We will discuss key examples and methodologies within the cybernetic paradigm, highlighting their advantages and limitations in modelling

gene expression.

5.3.1 Information-Theory Models

The simplest cybernetic models are information-theory models, which utilize concepts from information theory, a field of study focused on quantifying and analyzing the transmission and processing of information. In the context of gene regulation and expression, these models aim to identify and quantify patterns between the expression profiles of different genes. By analyzing correlations and dependencies between gene expression profiles, information-theory models can uncover potential functional similarities and connections between genes.

The characteristic example of an information-theory model is the correlation network [31]. Correlation networks aim to establish links between genes by identifying those with highly correlated expression profiles. To measure these correlations, the Pearson correlation coefficient is commonly used, though other methods such as Euclidean distance or information-theoretic measures have also been employed [32]. The Pearson coefficient evaluates the linear relationship between two variables, while Euclidean distance calculates the straight-line distance between two points in multi-dimensional space. Information-theoretic measures quantify the mutual information shared between variables, which is a measure of their dependency—quantifying the amount of information one variable provides about the other.

Such correlations might be coincidental, however. To determine whether the correlated genes truly share functional similarities, researchers often examine if these correlations persist across different species. While not definitive proof, the presence of conserved correlations across many stages of evolution suggests that the genes could indeed be functionally related.

While correlation networks efficiently map out gene interactions, they cannot establish causative relationships. By utilizing asymmetric information, which considers the directionality of gene relationships, 'influence networks' can be formed to capture a degree of causality [33]. Information-theory models, while simple and easy to conduct, possess inherent limitations due to their static nature. Unlike the dynamic process of gene regulation, which involves constantly changing relationships among genes, information-theory models treat gene interactions as fixed. They capture the relationship between genes using a single number, such as a correlation coefficient, which fails to account for the temporal changes in regulatory networks.

Additionally, information-theory models only consider pairwise interactions, disregarding the potential involvement of multiple genes in the regulation of a single target gene. As a result, these models cannot provide a complete or accurate representation of the true underlying biological processes at play.

5.3.2 Boolean Networks

Boolean networks, first proposed by Kauffman in 1969, are another class of cybernetic models that offer a more dynamic approach to understanding gene regulation [34]. Boolean networks are designed to capture the combinatorial nature of transcription regulation, which in this context refers to the idea that different combinations of transcription factors can lead to distinct changes in the expression of a target gene. Unlike information-theory models, Boolean networks represent each gene in either an 'on' or 'off' state, thus simplifying the complexity of gene regulation.

In Boolean networks, the state of each gene is modeled by a logic function that depends on the states of other genes it interacts with. A logic function is a mathematical operation that takes binary inputs (true or false, or in this context, on or off) and produces a binary output based on specific rules, such as AND, OR, and NOT operations.

Boolean networks, by representing gene expression data in only two states (on or off), sacrifice some of the information present in continuous expression data. This simplification, however, offers an advantage in terms of interpretability, as it allows for a more accessible understanding of the complex gene interactions involved in transcription regulation.

Boolean networks offer a key advantage in representing the dynamic nature of gene regulation. By simplifying gene expression into on and off states and using logical functions to define regulatory relationships, these networks can effectively track and model state transitions over time. This captures the dynamic gene regulation process and various outcomes from different transcription factor combinations. Researchers like Thomas and Martin et al. have shown the success of this approach in depicting temporal changes in regulatory networks [35][36].

Modeling regulatory networks as systems of ordinary differential equations offers a more powerful, quantitative method for capturing the feedback dynamics inherent in gene regulation. Ordinary differential equations (ODEs) are mathematical equations that describe

the relationship between a function and its derivatives, representing how a variable changes with respect to other variables, typically including time. In the context of gene regulation, ODEs can model how the expression level of a gene changes based on the influence of certain regulatory factors and the complex interactions between them.

Both linear and nonlinear ODE models have been proposed to describe the relationships between genes and their regulators mathematically [37][38]. By explicitly incorporating time into the equations, ODEs are able to capture the dynamic nature of gene regulation, with the derivatives representing the rates of change of gene expression levels over time.

Some researchers have also utilized stochastic differential equations to account for the nondeterministic nature of transcription [39]. Stochastic differential equations are similar to ODEs but include a random component, representing the inherent variability in biological processes like transcription.

Advanced models, like nonlinear and stochastic differential equations, boast high representational capacities—meaning they can represent highly complex relationships—but often suffer from being underdetermined. The term 'underdetermined' refers to a situation where there is not enough data to uniquely determine all the parameters within the model, comparable to fitting a line to a single point or a quadratic function to only two points. To overcome this challenge, researchers typically constrain the problem, introduce a priori knowledge, or focus on small subsets of the genome (5-10 genes) at a time. A popular approach in this domain is the S-system model, which shares these limitations but provides innovative methods for parameter optimization. [40].

5.3.3 Bayesian Networks

Bayesian networks are a powerful tool for modeling complex relationships between variables, such as gene expression levels, while taking into account the uncertainty in the relationships. They are built on the principles of probability, allowing the likelihood of certain events or interactions to be incorporated.

In a Bayesian network, each gene is represented as a node within a graph. The connections between these nodes are directed, meaning they have a specific direction indicating the influence one gene has on another. These connections form a directed acyclic graph, which means that the connections never form a loop and always flow in one direction.

The key idea behind Bayesian networks is conditional independence. This means that, given the information about the genes that directly influence a particular gene (parent nodes in the graph), that gene is considered to be independent of all other genes not directly or indirectly influenced by it (non-descendent nodes in the graph). This allows us to simplify the complex relationships between genes and focus on the most important direct interactions. By representing gene interactions in this way, Bayesian networks provide a framework for modeling the probabilistic nature of gene expression and the causal relationships between genes.

One limitation of Bayesian networks is that the requirement for the graph to be acyclic prevents the representation of feedback loops, which are often present in gene regulatory systems. However, researchers have developed dynamic Bayesian networks to address this challenge, allowing for the incorporation of feedback loops and temporal changes. For example, Rangel et al. created a 39-gene state space model, a type of dynamic Bayesian network model, effectively capturing feedback dynamics probabilistically [41].

To reiterate, cybernetic models aim to reveal the overall network of interactions among genes by examining relationships between gene expression levels. Although the cybernetic approach provides valuable insights into the control and feedback loops in complex systems, it falls short in capturing the underlying molecular mechanisms driving these interactions, limiting its generality. As a result, despite its contributions to understanding gene regulation processes, the cybernetic approach remains an incomplete and unsatisfactory tool for comprehensively studying transcription and gene regulation.

5.4 Mechanistic Models

The mechanistic paradigm, on the other hand, examines the 'black box' of transcription while overlooking broader gene interactions. Mechanistic models concentrate on gene-specific information like local DNA sequences and histone modifications, enabling them to capture molecular mechanisms driving transcription. However, by ignoring dynamic cellular changes, such as varying transcription factor levels, there is a tight limit to their generality and capacity to make predictions in new scenarios.

5.4.1 From Sequence Data

The most straightforward mechanistic models are those which use DNA sequence data as input. The relevance of the DNA sequence to transcription is supported by the discussion is section 4.3.1. A primary consideration is the presence of binding motifs for transcription factors in the promoter and enhancer regions.

To predict transcription, then, an immediate idea is to start by simply searching for these motifs, and the classic paper by Beer and Tavazoie does just this while studying yeast. These authors looked for motifs in the 800 base-pair upstream region that were shared between genes with similar expression patterns. The thinking here is that such motifs are a potential causal factor for the common expression pattern. Next, Beer and Tavazoie used a Bayesian Network[6] which takes these motifs as input and outputs the probability of a particular expression pattern [42].

Yuan et. al. improved upon this approach by training a naïve Bayes classifier, which is a greatly simplified form of Bayesian network. Naïve Bayes classifiers make the strong assumption that inputs are independent, earning the 'Naïve' label and significantly reducing the model complexity. Yuan et. al., with their simpler model, achieved superior results to Beer and Tavazoie, further demonstrating the predictive value of transcription factor motifs [43].

Besides searching for motifs, there is another way to consider DNA as input which views the entire activity of an organism, including the synthesis of every protein, as a product of the exact configuration of its DNA. This perspective considers DNA as the blueprint for the whole organism, with all possible measurable cellular phenomena being derived from this genetic information. These phenomena include gene expression, histone modifications, DNA methylation, transcription factor binding, and more. The ambitious goal of some cutting-edge models is to read and interpret this complex blueprint directly from the DNA sequence.

To achieve this, researchers apply machine learning techniques, which involve developing algorithms that allow computers to learn and make predictions from data without explicit programming. Machine learning is particularly helpful in this task, as it can identify patterns and relationships within complex, high-dimensional data sets, such as DNA sequences.

[6] In contrast to the cybernetic Bayes nets which take expression levels as input.

A landmark paper introduced a model called Basenji, which takes as input a 131,000 base-pair DNA sequence centered on some TSS and predicts 4,429 distinct genomic tracks in that DNA region [44]. A 'track' is the name given to a certain measurement repeated at every location along the entire genome. Various tracks include histone modifications, DNA methylation, and expression levels from various sources and cell types. To predict so many of these tracks, Basenji employs dilated convolutional layers, which are specialized neural network layers that enable efficient processing of spatially structured data.

Subsequent improvements to the Basenji approach were achieved by integrating data from the mouse genome [45]. The most notable results were obtained by a team from DeepMind, led by Ziga Avsec, who utilized the transformer architecture [46]. Transformers are a type of neural network architecture that excel in handling sequences of data, making them particularly suitable for analyzing DNA sequences. This approach significantly enhanced the prediction accuracy of gene expression and other cellular phenomena, showcasing the potential of machine learning and further demonstrating the importance of the DNA sequence in predicting expression or transcription.

5.4.2 Other Inputs

Besides sequence data, a variety of other factors relating to the gene/promoter have been used to predict expression. One established approach is to use measured histone modifications as a model input [47]. The importance of histone modifications as an input was established by the discussion in section 4.3.3. One noteworthy model based on histone modifications is called DeepChrome, developed by Singh et. al, and uses machine learning to make its predictions [48]. Another method in predicting transcription involves the use of DNAse sensitivity data as input. To better understand this approach, it is essential to clarify some underlying concepts. Deoxyribonuclease I (DNAse I) is an enzyme that selectively cleaves DNA molecules at regions that are more accessible and less tightly wound around histone proteins. These regions are referred to as having high DNAse sensitivity. So, in other words, DNAse sensitivity may be thought of as a more direct measurement of DNA accessibility than even histone modifications. When compared with using only the DNA sequence near the TSS as input, Natarajan et. al. found that also including DNAse sensitivity as input yielded improvements in predictions [49].

While the methods discussed here, including the use of DNA sequence data, histone modifications, and DNAse sensitivity represent some prominent approaches to predicting transcription, it is important to note that there are even more techniques employed in the field. Two further inputs employed by researchers are investigating DNA methylation patterns and measuring transcription factor binding sites directly [50][51][52][53]. For the sake of brevity, we have only touched on a few examples, but any factor affecting expression, including all those discussed in section 2.3, is theoretically valid to include as input.

To summarize, mechanistic models aim to unravel the inner workings of transcription and gene regulation by focusing on gene-specific information, such as local DNA sequence and histone modifications. While these models provide valuable molecular insights, they fail to consider the broader, system-level interactions among genes and the dynamic changes in cellular context, such as varying levels of transcription factors. This fact places a stark limitation on the generality and overall predictive power of mechanistic models in new cell environments. Therefore, despite their contributions to our understanding of regulatory machinery, mechanistic models are an incomplete and insufficient tool for a comprehensive predictive model of transcription.

5.5 Integrating Both Paradigms

The preceding sections have explored models from both the cybernetic and mechanistic paradigms in the study of gene regulation. However, a small body of literature seeks to combine elements of both paradigms to create a more comprehensive model. The key advantage of this integrated approach lies in the ability to consider both the system-level interactions between genes and the specific molecular factors driving gene regulation, which enables the model to capture the complex interplay of processes governing gene expression.

The majority of integrated models are called 'thermodynamic models,' because they operate under the assumption that the binding reactions between transcription factors and DNA sequence motifs are in thermodynamic equilibrium. Utilizing this assumption, the models first search promoter and enhancer regions for transcription factor motifs. Once identified, they compute all possible combinations of transcription factors that can bind to the promoter, taking into account that two factors cannot bind on top of each other. By integrating

the effects of each different transcription factor, and averaging across all combinations, the models generate a final expression prediction.

He et. al. provide a comprehensive overview of the thermodynamic approach, discussing the advantages, limitations, and implementations by other researchers, then introduce their own model to predict expression levels in Drosophila embryos [54]. The majority of these thermodynamic models focus on Drosophila, primarily due to data availability and ease of analysis. A significant challenge in the research is obtaining high-quality measurements of transcription factor concentrations. However, in fruit fly embryos, these concentrations can be effectively measured across the spatial dimension of the embryo, making it an ideal system for study. Thus, rather than utilizing synchronized measurements in each cell type (as discussed in the section 4.3.2), thermodynamic models use measurements for each region of the embryo. In another notable study, Gertz et al. applied thermodynamic modeling to predict expression in yeast, demonstrating the potential of this integrated approach to study gene regulation in different organisms [55].

Another technique that combines elements of both the cybernetic and mechanistic paradigms is motif expression decomposition, introduced by Nguyen and D'haeseleer [56]. This method seeks to analyze the influence of each transcription factor on the expression of a specific gene under a particular cellular context, or condition. It does so by breaking down this influence into two key components.

First, motif strength represents the affinity between a transcription factor and its corresponding DNA sequence motif(s) in the gene's promoter or enhancer regions. This value indicates how strongly a transcription factor is attracted to or repelled by a specific DNA sequence.

Second, transcription factor activity reflects the capacity of the transcription factor to either enhance or suppress gene expression under the given condition. This 'capacity' can be thought of as incorporating both transcription factor concentration and degree of activating or inhibiting effect. By considering both motif strength and transcription factor activity, the motif expression decomposition approach provides a more comprehensive understanding of how transcription factors impact gene expression.

In Nguyen and D'haeleseer's work, this computation is represented in matrix form as follows:

$$\boldsymbol{E} = \boldsymbol{M}\boldsymbol{A},\ \ \boldsymbol{E} \in \mathbb{R}^{m \times n},\ \ \boldsymbol{M} \in \mathbb{R}^{m \times k},\ \ \boldsymbol{A} \in \mathbb{R}^{k \times n} \tag{3}$$

where $\boldsymbol{E}$ is an expression matrix containing the expression level of m genes under n conditions, $\boldsymbol{M}$ is a condition-independent motif strength matrix of m genes for k motifs, and $\boldsymbol{A}$ is a matrix of TF activity for k transcription factors in n conditions. This matrix form enables modelling the expression of all genes under a set of different conditions in one equation. By representing the relation between transcription factors and sequence motifs as multiplicative, motif expression decomposition effectively bridges the gap between both paradigms.

5.6 Conclusion

The models discussed above effectively integrate both the cybernetic and mechanistic paradigms to provide a much broader understanding of gene regulation than either paradigm can on its own. The cybernetic paradigm, which focuses on gene network inferencing, considers the interactions between genes and their influence on one another's expression levels. By treating genes as black boxes, however, cybernetic models disregard the internal processes and molecular mechanisms that govern the behavior of genes, as discussed in section 5.2.1. Consequently, these models lack the ability to generalize to new scenarios and predict the behavior of new genes within the network.

On the other hand, the mechanistic paradigm opens up the black box, considering gene-specific information such as local DNA sequences and histone modifications to predict transcription or expression. In doing so, however, mechanistic models overlook the broader system-level interactions between genes and the impact of varying concentrations of transcription factors. As a result, mechanistic models also lack the ability to generalize and predict expression in new cell environments, as discussed in section 5.2.2.

By incorporating transcription factor activities or concentrations with specific motifs along the promoter sequence, models that combine both paradigms offer an explanatory capacity unmatched by others. These integrated models effectively balance the system-level

interaction between genes with the actual molecular mechanisms by which transcription occurs. In doing so, these models are capable of capturing the relative importance of each transcription factor and sequence element for expression in every different cell context.

The deeper level of understanding provided by integrated models is not just a nice plus. It is truly the only path forward to achieving the aims of section 3. To reiterate, a major step toward harnessing proteins as tools is controlling their synthesis. Our bodies control the production of proteins largely using promoters & enhancers, so, in theory, it should be possible for us to do the same by designing synthetic promoters & enhancers. Attaining any understanding of how these designed sequences will actually operate in cells requires a predictive model that can handle new genes in new cell contexts. This degree of generality is missing from both cybernetic and mechanistic models, and can only be achieved by an integrated approach.

Despite the necessity of combining these paradigms, the literature is absent of any integrated models of transcription in humans, to the best of my knowledge and research. This absence highlights an unexplored territory within the field of gene regulation research, presenting fertile ground for new investigations. This document attempts to take a first step into this new territory, carving a path for others to follow.

6 Methods

The development of the model in section 4 was entirely abstract. In this section, we will continue with this development, focusing now on concrete details and the model's implementation.

6.1 Data

6.1.1 Level of Transcription

Picking up where section 4 left off, we currently have a model that takes as inputs the DNA sequence, transcription factor levels, and histone modifications, and outputs a probability distribution over different levels of transcription. This abstract model is represented in mathematical form in Equation 2.

Throughout all previous sections and until now, no distinction was made between 'expression level,' and 'level of transcription' for the following reason. Ideally, we wish for our model to predict the "level of transcription," which should reflect the absolute frequency of POL II binding to the TSS and initiating transcription. However, direct measurements of this frequency are unavailable, so measurements of mRNA abundance are used instead. Such measurements can be obtained at enormous scale from RNA sequencing experiments (RNA-seq), making them highly suitable for analysis.

Since mRNA is produced by transcription, the abundance of mRNA is a reasonable measure of the frequency of transcription. 'Gene expression' is a somewhat vague term encompassing the degree of a gene's activity, production of RNA, or production of proteins. As such, measurements of mRNA abundance are usually called measurements of gene expression in the literature. In other words, transcription frequencies (which we desire ideally) are approximated by widely available measurements of gene expression. Thus, we will have our model predict gene expression measurements instead of the 'level of transcription.'

Gene expression measurements, normalized in units of transcripts per million (TPM), allow for comparison across genes, cells, and experiments. It should be noted that there are other units for measuring gene expression besides TPM. However, an in-depth analysis of the exact methods of measurement and differences between these units is beyond the scope

of this document. A comprehensive comparison can be found in other sources [57].

Using TPM unavoidably conflates transcription with post-transcriptional modifications and mRNA degradation, as certain mRNA transcripts have varying half-lives.[7] A higher concentration of a transcript may result from a longer half-life rather than greater transcription levels. Despite these limitations, RNA sequencing experiments remain among the best and most abundant sources of data for modeling transcription. If better data becomes available, better models will be possible with greater precision and fidelity in predictions.

A respected, commonly used source of mRNA TPM data that we can use is the GTEx dataset, obtained from the GTEx portal [58]. The subset of this dataset that we will use consists of expression measurements of 19,786 coding genes, each measured 17,382 times. Each measurement is taken from one of 54 different cell types in one of 948 different individuals. The word 'sample' will be used to refer to a measurement of all genes in a single cell type in a single individual of this dataset. Thus, one sample consists of exactly 19,786 measurements of gene expression (one for each coding gene). The GTEx data exists in a 19,786 x 17,382 table, which resembles Figure 10.[8]

	Gene A (TF)	Gene B	Gene C	...
John, skin cell	0.34	1293.20	5.40	...
John, neuron	487.92	0.00	49.83	...
Mary, neuron	527.93	0.29	86.92	...
...	...	...	...	...

Figure 10: A representation of the structure of the raw data from GTEx experiments.

6.1.2 DNA Sequence

As discussed in section 4.3.1, the DNA sequence is a critical input for our model. For the sake of simplicity, I will make the assumption that all 948 individuals in GTEx experiments have the same DNA, as the genetic makeup of any two human beings is 99.9% similar. Of course, the differences between individuals are not insignificant, and some error will result

[7] Half-life is a measure of how quickly a substance degrades, decomposes, or decays.

[8] Note, the actual text document one might download from the GTEx portal appears as the transpose of the table in Figure 10.

from making this assumption. While the unique DNA sequence data for each individual in GTEx experiments exists, making use of this would provide only a small benefit compared to the large effort required to process and make use of all this data. Accounting for the influence of genetic differences on expression is one area for improvement listed in section 9.

To represent the DNA of all 948 individuals, we must select a single DNA sequence, and for this purpose, I will use the Genome Reference Consortium Human Build 38, commonly referred to as hg38 [59]. A reference genome serves as a standardized representation of the genetic makeup of a species, acting as a basis for comparison and analysis in various genetic studies. In the case of the hg38 reference genome, it is a widely accepted, comprehensive representation of the human genome that encompasses the genetic information and structure of a typical human being.

Naturally, it is not necessary to provide our model with the entire human reference genome each time it needs to make a prediction. Instead, the model should only require the DNA sequence near the gene's TSS, particularly the regions where transcription factors may bind and influence transcription. Although defining regional boundaries is somewhat discretionary, I refer to unpublished results from my research group to select 2,773 base pairs upstream of the TSS as the cutoff for promoter and enhancer regions. To ensure the inclusion of all core promoter elements necessary for POL II binding, I also consider downstream elements. One core promoter element, known as the downstream promoter element, is located 28 to 32 base pairs downstream of the TSS [60]. To err on the side of caution, I will include 50 base pairs downstream of the TSS in our input. Consequently, the input DNA sequence for our model will span 2,823 base pairs, extending from 2,773 base pairs upstream to 50 base pairs downstream of the TSS.

In order to utilize the DNA sequence as input to our model, we must choose a suitable representation. To this end, we can employ a method called one-hot encoding. In the context of data processing, DNA sequences are considered 'categorical' data. Categorical data consists of discrete categories or classes, rather than numerical values. For DNA, the categories are the four nucleotide bases: adenine (A), cytosine (C), guanine (G), and thymine (T).

One-hot encoding is a technique used to represent categorical data as vectors. In the context of DNA sequences, the one-hot encoding process involves creating a separate column

vector for each of the nucleotides in the input sequence. Each vector has a length of 4 and contains a '1' at the position corresponding to the presence of that specific nucleotide, while the rest of the vector is filled with '0's. For example, consider a short DNA sequence: ACGTGTAC. The one-hot encoding for this sequence would be represented as follows:

$$\begin{matrix} A: \\ C: \\ G: \\ T: \end{matrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{4}$$

where each column represents a vector for the corresponding position in the DNA sequence, and the '1' in each vector indicates the nucleotide at that position. In this manner, one-hot encoding efficiently translates DNA sequences into a format that can be easily processed by our model, enabling it to make predictions based on the provided input.

6.1.3 Transcription Factor Levels

As discussed in section 4.3.2, we can utilize gene expression measurements to estimate the concentration of transcription factors within a cell, forming another important input. In order to ensure that these measurements are synchronized in time and location with the gene expression levels we want to predict, we must be cautious about their integration, as illustrated in Figure 11 (presented in contrast to Figures 5 and 6). In the top half of Figure 11, the black **X** over the arrow from the first individual to the second indicates that transcription factors will not physically hop between people. In the bottom half of Figure 11, the black **X** over the arrow from the skin cell to the neuron indicates that transcription factors will not jump between cell types.[9] Therefore, we can only use the expression level of transcription factors in the same sample as the gene whose expression we wish to predict.

[9]While this is conceivable to a limited extent, I would suggest that transcription factors primarily influence expression in the cell in which they are produced.

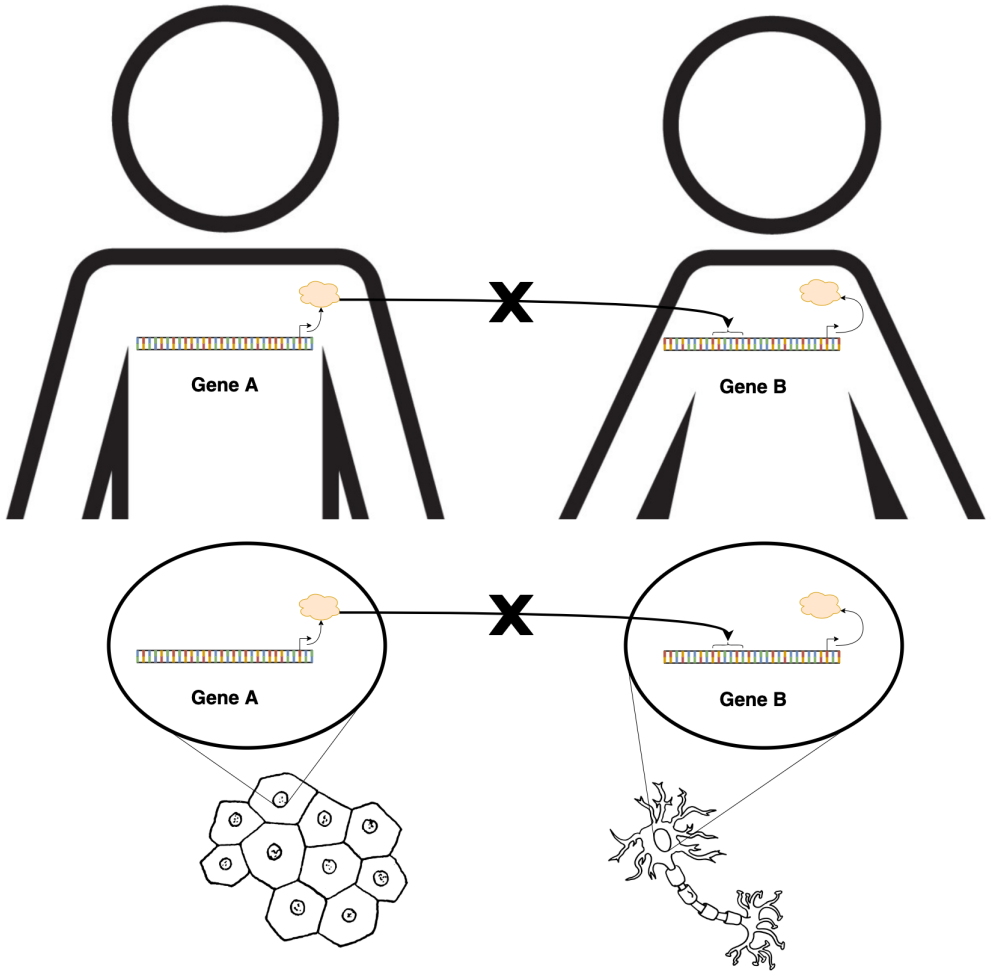

Figure 11: An illustration of two cases where gene interactions through a transcription factor generally do not occur.

To translate this caution into a rule for selecting inputs, consider again Figure 10, and imagine that we wish to predict the expression level of gene B in John's neurons. According to Figure 11, it would be inappropriate to use the expression of gene A in John's skin, or the expression of gene A in Mary's neurons. Such inappropriate inputs are illustrated in Figure 12. In this figure, a red arrow pointing from table cell x to table cell y indicates that it is inappropriate to use x as an input when predicting y.

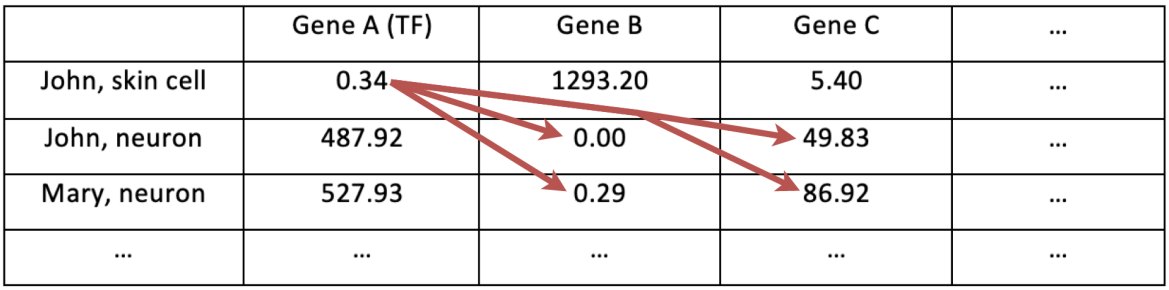

Figure 12: An illustration of inappropriate transcription factor inputs.

On the other hand, when predicting the expression level of gene B in John's neurons, it is perfectly appropriate to use the expression of gene A in John's neurons. Such appropriate inputs are illustrated in Figure 13. In this figure, a green arrow pointing from table cell x to table cell y indicates that it is appropriate to use x as an input when predicting y.

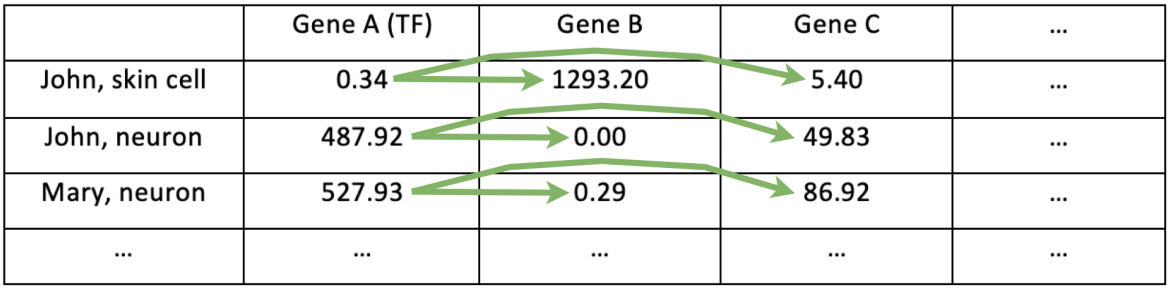

Figure 13: An illustration of appropriate transcription factor inputs.

To summarize, when predicting the expression level in some cell in the GTEx table, the expression level of every transcription factor in the same row (the same sample) is appropriate to include as input.

Until now we have assumed that a knowledge of which genes produce transcription factors is given. Such knowledge, however, is imperfect, and represents another source of error in this model. Nevertheless, a list of 2,753 potential regulatory proteins was collected by Lambert et. al., of which around 1,600 have been labelled 'likely transcription factor' [9]. Among these likely transcription factors, there are 1,072 for which binding motif(s) have been identified. The data source for binding motifs will be described in section 6.1.7. Since the interaction between transcription factors and the DNA sequence is critical for our model, I will use the expression measurements of exactly 1,072 genes in the appropriate row of the GTEx table as another input to the model, alongside the DNA sequence.

6.1.4 GENCODE Annotations

Lambert et. al. provides us with the names of the transcription factors, but we also need to map these factors to their corresponding genes and align these genes with the corresponding expression measurements found in the GTEx table. To accomplish this mapping, I use the GENCODE release v26 annotations, which are also utilized by the GTEx dataset. GENCODE is a comprehensive, high-quality gene annotation project that is part of the Encyclopedia of DNA Elements (ENCODE) consortium. The ENCODE consortium is a large-scale collaborative research project initiated by the National Human Genome Research Institute aimed at identifying all functional elements in the human genome sequence, including genes, regulatory elements, and other functional DNA sequences [61].

Annotations are essential for providing a standardized and accurate description of genomic features, such as the locations, structures, and functions of genes and other functional elements. By using GENCODE release v26 annotations, we ensure that our model is based on a reliable and consistent labeling framework for all genes involved, aligning with the GTEx data source we're working with. This facilitates accurate mapping of transcription factors to their respective genes and expression measurements, which is indispensable for a model that utilizes the expression levels of transcription factors as input.

6.1.5 Same Inputs as Outputs?

It is essential to address whether there is a methodological issue in predicting the expression of transcription factor-producing genes while also using them as model input. One might wonder if the model could 'cheat' since it already has the information it is supposed to predict. Despite this apparent paradox, using transcription factor inputs in this manner is still justified, as it is entirely plausible for a transcription factor to regulate the gene that produces it, such as in a negative feedback loop. As our model predicts the expression of all 19,786 genes, there should be no strong inclination to cheat on a mere 1,072. Moreover, we can design the model to avoid cheating, as discussed in section 6.4.6.

Still, it is principled to test whether the model is cheating, and to do this we will create a special dataset, called the validation set, with entirely new genes the model has not seen before. This dataset split strategy is discussed in more detail in section 6.4.2. If the model's

success is solely due to cheating on transcription factor-producing genes, it will struggle to make accurate predictions on the validation set. However, if it can make such predictions, we can be confident in the model's level of generalization.

6.1.6 Histone Modifications

Next I turn attention to histone modifications, another important input for our model. As explained in section 4.3.3, measurements of histone modifications that are synchronized in time and location (like the transcription factor levels) are unavailable. As a result, the data for histone modifications are inherently insufficient, which contributes to inaccuracies in the model.

Despite this limitation, we can consider data for three types of histone modifications, each measured across seven different cell types, resulting in a total of 21 tracks. These tracks are produced by the Bernstein Lab at the Broad Institute [62]. Tracks represent measurements of a specific quantity (in this case histone modifications) at every position throughout the entire human reference genome.

I will align these histone modification tracks with the DNA sequence used as input in our model. This alignment ensures that the histone modification input incorporates measurements spanning from 2,773 base pairs upstream to 50 base pairs downstream of the transcription start site. Aligning these inputs is not only simplest but also allows for a cohesive integration of histone modification data with the DNA sequence.

One key limitation of the histone data is that it is availabile for only seven cell types, while our model aims to predict expression in 54 different cell types. Consequently, each time the model makes a prediction, it must essentially determine which of the seven cell types is 'most similar' to the cell type for which it is predicting expression. This calculation is approximated by taking a unique superposition (linear combination) of all seven cell types, for each of the 54 cell types the model makes predictions on. The weights of this superposition are learned, meaning the model can optimize this mapping to deliver the best predictions. The concept of learned parameters is discussed in more detail in section 6.2.

The exact calculation is represented visually in Figure 14. In this diagram, the various inputs and intermediary tensors are represented by rectangular prisms. Here, the one-hot encoded cell type of the sample for which the model is predicting expression forms one input,

and the 21 histone modification tracks form another input. This calculation produces four unique superpositions of the seven cell types, for each of the three histone modification marks.

It cannot be overemphasized how hand-wavy and overly simplified this approach is as a solution, but something of this kind must be used to map between cell types, due to the inherent limitation that the source data only has seven cell types. Thus, obtaining better histone modification data in the future will lead to improved models.

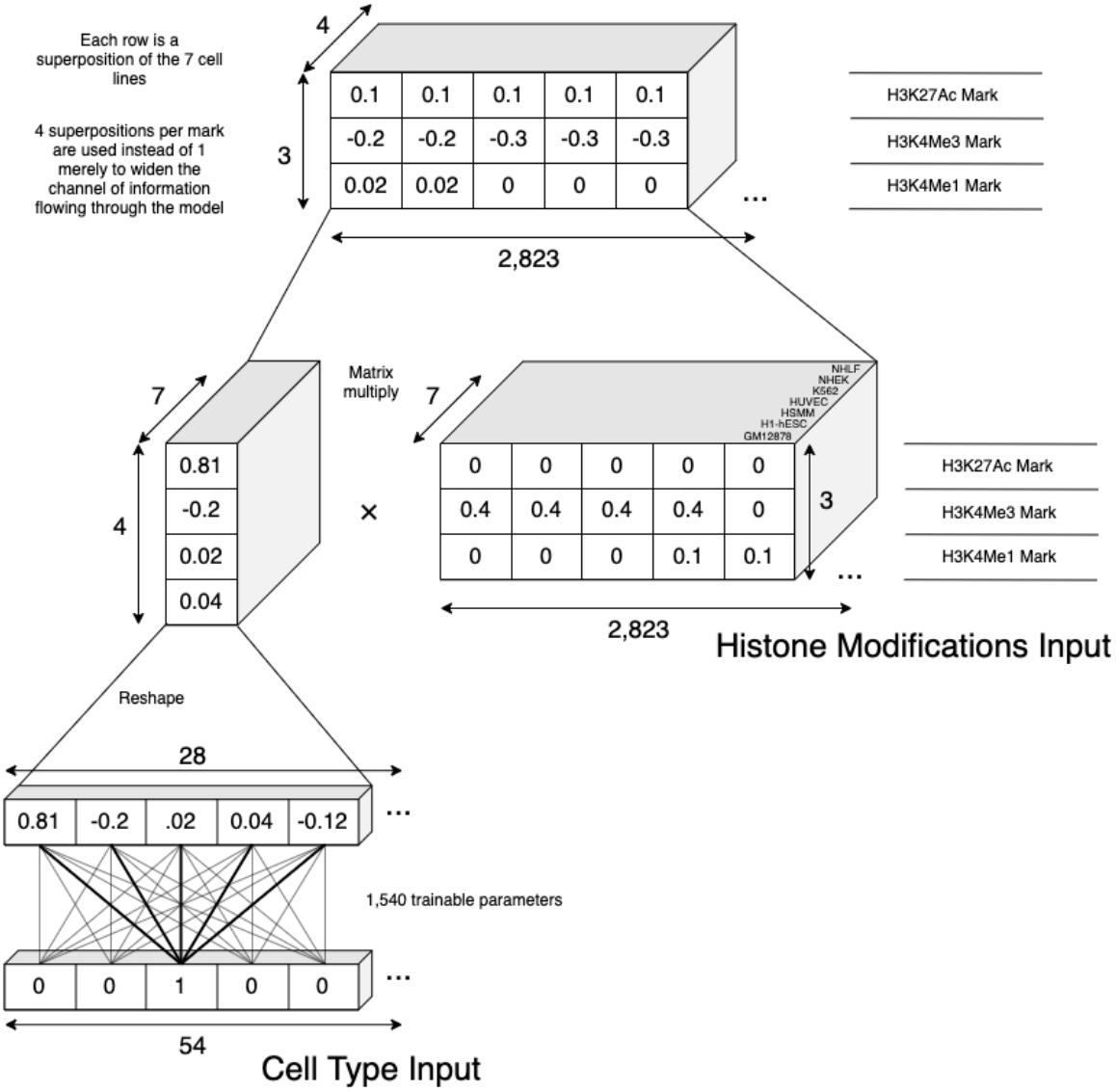

Figure 14: An illustration of the parametrized mapping from the seven cell types in histone measurements to the 54 cell types in GTEx measurements.

6.1.7 Binding Motifs

To effectively predict transcription factor binding sites—critical for ultimately predicting transcription—our model requires more than just the DNA sequence, transcription factor levels, and histone modification data. It also needs to know the specific motif(s) to which each transcription factor is likely to bind. Armed with this knowledge, the model can scan the DNA sequence for these motifs, identifying possible binding sites.

To ensure the model has access to accurate binding motif data, we will use the JASPAR dataset, widely recognized as one of the best resources for transcription factor binding motifs [63]. However, it is important to note that the JASPAR data is not perfect and represents another source of error in the model. As mentioned in section 6.1.3, the intersection between transcription factors identified by Lambert et. al. and motifs present in the JASPAR dataset consists of 1,072 proteins. Consequently, our model will be designed to search for 1,072 motifs within each promoter and enhancer sequence we provide. In addition to the 1,072 transcription factor motifs, we can also incorporate thirteen core promoter elements from the JASPAR dataset, which serve as recognition sites for the transcription machinery, as discussed in section 2.3.1 [8].

It's worth noting that I don't consider motif data as an input, even though we are indeed 'inputting' it into our model. This may seem paradoxical, but I treat it this way because the data remains the same for every single prediction the model makes. Furthermore, the motifs serve as patterns the model searches for, making them more akin to the model's design rather than its inputs and outputs. Think of the motifs as filters the model employs during its search process. Instead of being treated as inputs, motifs will be represented as parameters of our model, which will be discussed in the next section.

6.2 Parametrization

Parametrization lies at the heart of our model, as it involves assigning parameters that act as defining characteristics or descriptors, shaping the model's behavior and its transformation of inputs into outputs. The model's parameters will come in two forms: trainable and untrainable. Trainable parameters are adjustable and updated continuously through the training process, while untrainable parameters are set once to begin with and remain fixed.

The training process will be described in section 6.4.

The parametrization of the output distribution, however, is a slightly different concept from trainable and untrainable parameters. While the model's parameters describe its behavior, the parameters of the output distribution serve to define the shape of the output distribution. The model learns to generate appropriate parameters for the output distribution based on its inputs, rather than adjusting them directly as it does with the model's trainable parameters. Despite this distinction, a connection between the two ideas remains: both are descriptors, one of the model's behavior and the other of the output distribution's shape.

I will denote the set of all trainable and untrainable model parameters by $\mathbf{\Theta}$, and then, we may rewrite Equation 2 as follows:

$$P(y_{pred} \mid x) = f(y_{pred};\ x, \mathbf{\Theta}) \qquad x, y_{true} \in \mathbf{X} \tag{5}$$

where f is the mapping from model inputs to probability density functions, x is the input data for one prediction, $\mathbf{\Theta}$ is the model parameters, $y_{pred} \in [0, \infty)$, $\int_0^\infty P(y_{pred} \mid x)\, dy_{pred} = 1$, y_{true} is the true gene expression level, and $\mathbf{X}$ is the dataset.

Having explored the model's inputs, outputs, and data, we can now appreciate the importance of parametrization. By adjusting these parameters, we refine the model behavior and its overall ability to predict gene expression based on the given inputs, leading to better predictions. In essence, parametrization serves as the bridge between the model's design and its ability to effectively utilize the provided inputs to generate accurate outputs.

6.3 Output Distribution

The first aspect of our model to be parametrized is the output distribution, which involves defining a parametrized probability distribution to represent the range of possible gene expression values. A parametrized probability distribution is a mathematical function that uses parameters to describe the shape and characteristics of the distribution. A common example of such a distribution is the normal or Gaussian distribution, which is defined by

two parameters: the mean (μ) and the standard deviation (σ). These parameters determine the center and spread of the distribution, respectively.

It is important to note that the parameters of the probability distribution (such as μ and σ in the Gaussian distribution) are not the same as trainable or untrainable parameters. Instead, our model will generate these parameters based on its inputs to represent its output predictions.

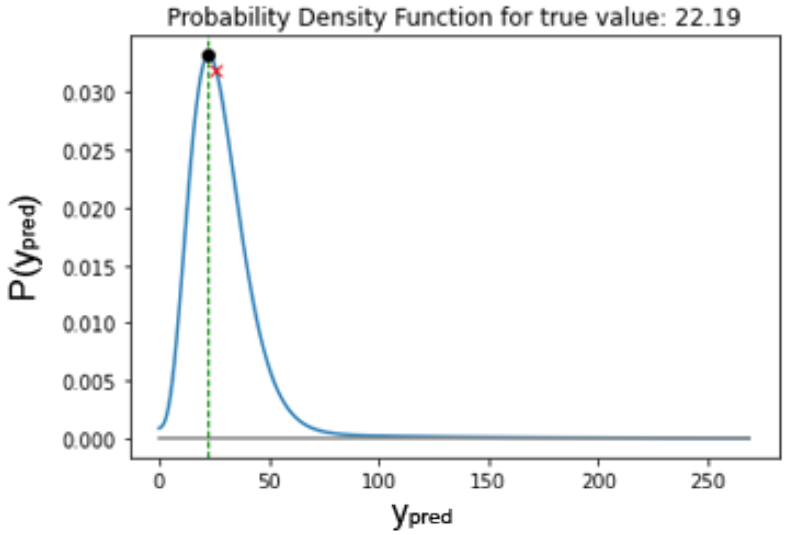

Figure 15: An example probability density function output by the model.

In our model, I will use a combination of a negative binomial distribution and the exponential distribution to represent gene expression data. Since the negative binomial distribution is actually a discrete probability distribution, we will instead use a continuous generalization of the negative binomial distribution. This combined distribution has a total of four parameters: two parameters for the continuous negative binomial distribution, one parameter for the exponential distribution, and one parameter for the relative quantities of each. In Figure 15, an example of a probability density function output by my model is presented. In this figure, the x-axis represents y_{pred}, the predicted expression level in units of TPM, the y-axis represents the value of the probability density function, the red x represents the mean of the distribution, and the black dot represents y_{true}, the value of which is stated in the plot title. The parametrized probability density function I use is represented mathematically as follows:

$$P(y_{pred}) = (s) \cdot NB(y_{pred}; r, p) + (1 - s) \cdot EXP(y_{pred}; \lambda) \tag{6}$$

where NB is the probability density function of the continuous negative binomial distribution, EXP is the probability density function of the exponential distribution, r and p are the two parameters of the negative binomial distribution, λ is the one parameter of the exponential distribution, and $s \in [0, 1]$ is a parameter defining the relative weight of the exponential and negative binomial distributions.

Recall that the four parameters of the output distribution are generated by our model. The model outputs may thus be represented as follows:

$$\begin{bmatrix} r \\ p \\ s \\ \lambda \end{bmatrix} = f(x; \mathbf{\Theta}) \qquad x, y_{true} \in \mathbf{X} \tag{7}$$

where f now represents the model's mapping from inputs to output distribution parameters. Finally, if we denote elements of the output vector in Equation 7 by f_1, f_2, f_3, and f_4 respectively, we may represent our model's output calculation as:

$$P(y_{pred} \mid x) = (f_3) \cdot NB(y_{pred}; f_1, f_2) + (1 - f_3) \cdot EXP(y_{pred}; f_4) \tag{8}$$

The negative binomial distribution is a suitable choice for representing gene expression data, as it can effectively capture overdispersion, a phenomenon often observed in such data [64][65]. Overdispersion occurs when the observed variance in the data is higher than what would be expected under a simpler distribution, such as the Poisson distribution.

I empirically found that the combination of these two distributions allows for easier training of the model and offers greater flexibility in capturing the complex patterns and variability

present in gene expression data, including overdispersion. The peak of the exponential distribution is always at zero, while the continuous negative binomial distribution can shift its peak to any location greater than zero. This allows the model to make highly uncertain, large guesses without facing a severe penalty if y_{pred} turns out to be zero. I found that the model learns faster when the it is 'unafraid' to make guesses. The process of learning, or model training, which adjusts the trainable parameters of the model to generate the output distribution based on the input data, will be discussed in greater detail in the next section.

6.4 Machine Learning

Machine learning is centrally important to our gene expression prediction model, as it is the process by which the model learns to adjust its parameters to best represent the underlying patterns in the data. Essentially, machine learning involves iteratively fine-tuning these parameters, usually called 'weights,' to minimize the difference between the model's predictions and the actual gene expression values. The measure of this difference is called the 'loss function,' 'loss metric,' or simply 'loss.' One of the most common methods for achieving this is the stochastic gradient descent algorithm (SGD), which adjusts the model's weights to minimize the loss.

The 'gradient' is a multi-dimensional derivative that represents the relationship between the loss and the weights. In simpler terms, the gradient informs us how the prediction error would be affected if we were to make small adjustments to each weight. Using the gradient, SGD takes small, random steps in the direction of the steepest descent, which is the direction that will decrease the loss fastest.

Each step is a weight update, or an adjustment of the model's parameters fine-tuning the model's behavior to improve its performance in predicting gene expression. As the model undergoes training, it gradually approaches the optimal configuration of weights. This process is often referred to as 'learning,' hence the phrase 'machine learning.'

The process of machine learning, specifically using SGD, is intricately connected to parametrization. By iteratively adjusting the weights of the model, SGD enables it to learn from the data and generate appropriate output distribution parameters based on its inputs.[10]

[10]Specifically, I use the Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.999, \epsilon = 1 \times 10^{-7}$), which is a variant of SGD with certain additional techniques that ease training. However, going into detail on the differences between

Updating parameters, in turn, refines the model's ability to predict gene expression values, leading to improved performance. Thus, machine learning and parametrization together form the foundation for effectively transforming the abstract concept of our model into a real, functioning system that can make accurate gene expression predictions.

6.4.1 Example Generation

To effectively harness machine learning, the model needs to be given training examples. Using these examples, the model will learn an optimal mapping of inputs to outputs. Each example comprises a unique combination of input features and a corresponding gene expression value. The model will be given inputs and will make a prediction using these, then calculate a loss value based on the difference between the prediction and the correct output. This style of machine learning is called 'supervised learning.'

Due to the unique nature of our input and output data, examples must be generated for the model in a very specific way. The details of this implementation are described in Appendix A.

6.4.2 Dataset Splits

A fundamental aspect of machine learning is dividing the dataset into training, validation, and test splits, which ensures that the model learns from distinct sets of examples. This separation fosters generalization, allowing the model to perform well on unseen data rather than simply memorizing the training set patterns.

Our gene expression prediction model allocates 65,536 examples each for validation and testing. While this deviates from the typical 80/10/10 split (80% training, 10% validation, and 10% test), the massive dataset of 340 million examples[11] renders allocating 34 million examples for validation and testing extremely prohibitive. Despite this, the vast training data enables effective learning and pattern capture, while validation and test sets allow reliable model fine-tuning and evaluation.

Crucially, no gene or sample present in validation or test splits exists in the training set,

optimization algorithms is outside the scope of this document.

[11] From the GTEx table, 19,786 genes times 17,382 samples yields a total of 343,920,252 examples across all three dataset splits.

ensuring the model's exposure to entirely unseen data during validation and testing. For example, if the model must predict gene B in Mary's neurons in the validation set, it will not have seen gene B in John's skin in the training set, nor gene A in Mary's neurons. This data split strategy ensures we can measure the model's true generalization and ability to capture the underlying mechanisms driving gene expression.

6.4.3 Loss Metric

The selection of an appropriate loss metric is a critical aspect of our model, as it forms the precise thing that will be optimized using gradient descent. In the previous sections, we discussed how our gene expression prediction model generates a probability distribution as its output. With this in mind, using the negative log likelihood (NLL) as the loss metric is a natural choice. Before diving into the rationale behind this choice, it's important to understand what the likelihood represents.

The likelihood is a measure of how probable it is to observe the true gene expression levels, given the model's predictions. Specifically, it is the value of the probability density function when evaluated at the true gene expression levels. A higher likelihood indicates that the model's predictions align well with the observed gene expression data, making it a desirable outcome. Using Equation 8, the precise formula for the negative log likelihood of our model is as follows:

$$NLL = -log\Big((f_3) \cdot NB(y_{true}; f_1, f_2) + (1 - f_3) \cdot EXP(y_{true}; f_4)\Big) \tag{9}$$

where, as in Equation 7, $f = f(x; \mathbf{\Theta})$, and $x, y_{true} \in \mathbf{X}$. There are several reasons for using the negative log of the likelihood instead of the likelihood itself. Logarithms offer several advantages in this context. First, they transform products into sums, which helps numerical stability when dealing with multiple data points, as the product of probabilities can result in very small values. Second, logarithms simplify derivatives, making the optimization process, such as stochastic gradient descent, more computationally efficient. Finally, logarithms are monotonic transformations, meaning that maximizing the log likelihood is equivalent to

maximizing the likelihood itself. As a result, we can use the log likelihood without altering the optimal parameter values. Finally, since it is convention to minimize the loss, we take the negative of the log likelihood.

The negative log likelihood is widely used in machine learning because minimizing it yields the maximum likelihood estimate (MLE) for the model's parameters. The MLE is known to be the best estimate for the parameters regarding its rate of convergence as the number of data points approaches infinity [66]. Furthermore, under specific conditions, such as the true data-generating distribution belonging to the model family and having a unique set of parameters, the MLE converges to the true parameter values [66]. With these justifications, the negative log likelihood serves as an intuitive and effective loss metric for our gene expression prediction model.

6.4.4 Neural Networks

Having explored parametrization, machine learning, and the loss metric, the use of neural network layers can now be explained. As discussed in section 2.3, transcription and gene regulation are complex biological processes, which involve the interplay between thousands of various molecular components. This complexity poses a challenge when developing a predictive model for gene expression.

Neural networks, a type of artificial intelligence inspired by the structure and function of the brain, offer a promising solution to tackle this challenge. They consist of interconnected layers of artificial neurons or nodes that can process and learn patterns in the input data. The power of neural networks lies in their capacity as universal function approximators, enabling them to approximate virtually any continuous function to an arbitrary degree of accuracy [67]. This capacity makes neural networks valuable for modeling complex, non-linear relationships, such as those found in transcription and gene regulation.

Incorporating neural networks into our gene expression prediction model allows us to capture the intricacies of these processes and generate accurate predictions. The model can learn patterns and relationships among input data, including DNA sequences, transcription factor levels, histone modifications, and other relevant factors, by leveraging the ability of neural networks to model complex relationships.

The parametrized components of neural networks are the neural network layers, each con-

sisting of numerous interconnected nodes or 'neurons.' Neural network layers are organized sequentially, with the output of one layer feeding into the input of the next. Typically, each neuron receives input from multiple neurons in the previous layer, computes a weighted sum of these inputs, adds a bias term, and applies a non-linear activation function. The trainable parameters of the neural network are the weights and biases associated with each connection between neurons, which are fine-tuned during the training process using techniques such as stochastic gradient descent.

There are different types of neural network layers, each designed to serve specific purposes in the model. In the following sections, we will discuss the various layers used in our gene expression prediction model and their respective roles.

6.4.5 Convolutional Layer

The first thing we would like our model to do is search for motifs and core promoter elements in the input DNA sequence, which can be accomplished with convolutional layers directly connected to the input DNA sequence. Convolutional layers are a key type of neural network layer designed specifically for processing and analyzing grid-like data, such as images or DNA sequences. Convolutional layers perform a mathematical operation called convolution, which enables the network to recognize and extract local features or patterns from the input data.

In a convolutional layer, multiple small filters or kernels slide across the input data, computing dot products between the kernel's weights and the corresponding input values. Each kernel is designed to detect a specific feature or pattern, such as an edge in an image or a motif in a DNA sequence. By applying multiple kernels, the convolutional layer can recognize a variety of different features in the input data.

The result of the convolution operation is a set of 'feature maps,' which are essentially filtered versions of the input data, highlighting the presence of the detected features. These feature maps are then passed on to subsequent layers in the neural network for further processing and analysis. By employing convolutional layers, our model can effectively scan the DNA input sequence for both transcription factor motifs and core promoter elements.

The inspiration to use convolutions comes from the bioinformatics technique of using position specific scoring matrices (PSSMs) to detect sequence motifs [68][69]. Multiplying PSSMs by a DNA sequence at a particular position, using some basic probability theory, and

applying Bayes rule yields the log probability of a binding event between some protein and that position on the sequence. To demonstrate the power of this approach, and show that it is identical to taking convolutions in a machine learning sense, the steps to construct a PSSM, and the theory behind them, will be walked through briefly.

First, for some particular transcription factor, a position frequency matrix is constructed. Given a set of observed binding events, the PFM contains a count of the number of each base found at each position relative to the binding site. As an example, consider the hypothetical position frequency matrix for protein Z, representing observations of 100 binding events:

$$PFM = \begin{matrix} A: \\ C: \\ G: \\ T: \end{matrix} \begin{bmatrix} 23 & 92 & 1 & 1 & 1 & 10 \\ 27 & 2 & 94 & 1 & 91 & 67 \\ 44 & 1 & 3 & 1 & 5 & 20 \\ 6 & 5 & 2 & 97 & 3 & 3 \end{bmatrix} \tag{10}$$

Here, we can readily see that an A in the 2nd position, a C in the 3rd position, etc. is almost always observed when Z binds, while the bases at the 1st position, 6th positions are more random. By normalizing across columns, one attains the position probability matrix in which each element contains the conditional probability of finding that base at that position, given a binding event:

$$PPM = \begin{matrix} A: \\ C: \\ G: \\ T: \end{matrix} \begin{bmatrix} 0.23 & 0.92 & 0.01 & 0.01 & 0.01 & 0.10 \\ 0.27 & 0.02 & 0.94 & 0.01 & 0.91 & 0.67 \\ 0.44 & 0.01 & 0.03 & 0.01 & 0.05 & 0.20 \\ 0.06 & 0.05 & 0.02 & 0.97 & 0.03 & 0.03 \end{bmatrix} \tag{11}$$

Assuming statistical independence between positions in the pattern, by multiplying the probabilities at each position, one attains conditional probability of any sequence, given a binding event of protein Z. For example:

$$P(CACTCG \mid Z_{bind}) = 0.44 \times 0.92 \times 0.94 \times 0.97 \times 0.91 \times 0.20 \tag{12}$$

$$P(CACTCG \mid Z_{bind}) = 0.067 \tag{13}$$

Assuming each base appears in DNA with equal probability, the probability of a base at any particular position is 0.25. Because we assumed statistical independence between sequence positions, we may write the probability of any given sequence as 0.25^l, where l is the length of the sequence. Thus, using Bayes rule one can attain the conditional probability of this transcription factor binding to any given sequence:

$$P(Z_{bind} \mid CACTCG) = \frac{P(CACTCG \mid Z_{bind})P(Z_{bind})}{P(CACTCG)} \tag{14}$$

$$P(Z_{bind} \mid CACTCG) = \frac{0.067 \cdot P(Z_{bind})}{0.25^6} \tag{15}$$

where $P(Z_{bind})$ is the global probability of Z binding to any DNA sequence, which is a function of its binding affinity and its concentration. The concentration is another input to the model, which as indicated in this equation, should be multiplied by the conditional probability of binding. This multiplication will be accomplished by another layer discussed in section 6.4.6. The binding affinity can be treated as a constant and learned by the model.

Instead of multiplying all the conditional probabilities together, they can be converted to log probabilities and added for easier computation. Typically log base 2 is used, and the base probability of 0.25 at each position is integrated right into the matrix. This is done by dividing every element of the matrix by 0.25 before taking the log. These operations yield the final PSSM:

$$PSSM = log_2\left(\frac{1}{0.25} \cdot \begin{bmatrix} 0.23 & 0.92 & 0.01 & 0.01 & 0.01 & 0.10 \\ 0.27 & 0.02 & 0.94 & 0.01 & 0.91 & 0.67 \\ 0.44 & 0.01 & 0.03 & 0.01 & 0.05 & 0.20 \\ 0.06 & 0.05 & 0.02 & 0.97 & 0.03 & 0.03 \end{bmatrix}\right) \tag{16}$$

$$PSSM = log_2\left(\begin{bmatrix} 0.92 & 3.68 & 0.04 & 0.04 & 0.04 & 0.40 \\ 1.08 & 0.08 & 3.76 & 0.04 & 3.64 & 2.68 \\ 1.76 & 0.04 & 0.12 & 0.04 & 0.20 & 0.80 \\ 0.24 & 0.20 & 0.08 & 3.88 & 0.12 & 0.12 \end{bmatrix}\right) \tag{17}$$

$$PSSM = \begin{bmatrix} -0.12 & 1.88 & -4.64 & -4.64 & -4.64 & -1.32 \\ 0.11 & -3.64 & 1.91 & -4.64 & 1.86 & 1.42 \\ 0.82 & -4.64 & -3.06 & -4.64 & -2.32 & -0.32 \\ -2.06 & -2.32 & -3.64 & 1.96 & -3.06 & -3.06 \end{bmatrix} \tag{18}$$

With DNA sequence data represented in a one-hot encoding, the dot product[12] of any sequence at some position with the PSSM will then yield the log conditional probability of binding given that sequence. The only differences between this operation and convolution is that in convolutions, 1) the PSSM is called a 'convolutional kernel,' or 'filter,' and 2) the kernel is flipped along both axes. Fortunately, in most modern frameworks, such as Tensorflow (discussed in section 6.7), the cross-correlation is used instead of convolution, which performs the same operation without flipping the kernel.

The upshot of all this is that PSSMs, which we get directly from the JASPAR database, can be used as untrainable convolutional kernels in our model. Thus, the first convolutional layer of our model will effectively detect all 1,072 motifs and 13 core promoter elements, outputting a feature map for each. In Figure 16, an illustration of the convolution calculation is presented for the SPI1 transcription factor, and in Figure 17 an illustration of the convolution calculation for the TATA-Box core promoter element. In both of these figures, the calculation shown is essentnially a dot product between the one-hot encoded DNA and the PSSM.[13]

[12] By "dot product" all is meant is an element-wise multiplication followed by summation over all elements.

[13] In Figures 16 and 17, the PSSMs are shown as length 6 for clarity. In the JASPAR database, however, motifs range up to length 35 for transcription factor motifs, and length 19 for core promoter elements. In code, motifs shorter than these max lengths are padded with zeros so that all motifs are the same length. To understand the exact operation in code, please consult the data_jaspar_get.py file, provided through section 11.

While the theoretical underpinnings of convolutions suggest their potential for motif detection, there might be lingering doubts about their performance in practice. To address these concerns and demonstrate their practical effectiveness, I conducted a test on a conjured dataset, with the results presented in Appendix B.

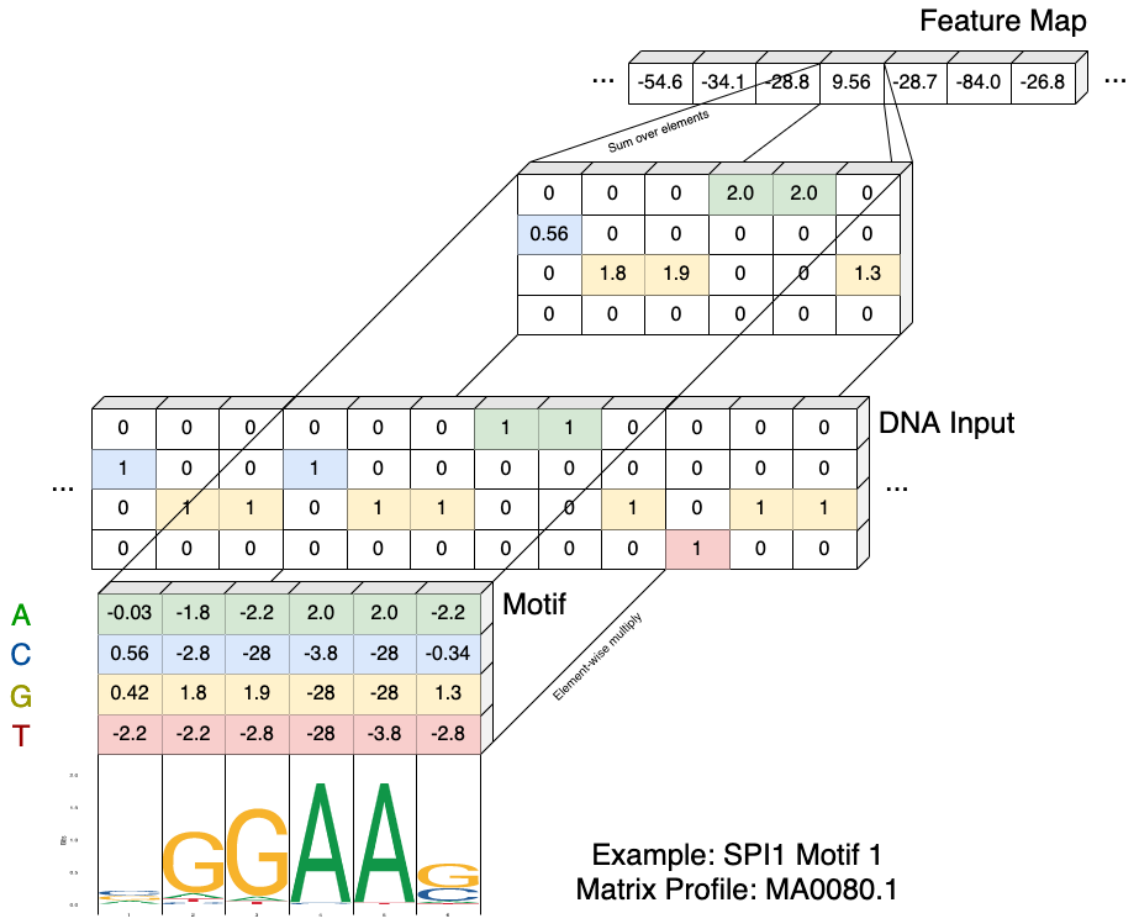

Figure 16: An illustration of the calculation used to generate a feature map for a transcription factor motif.

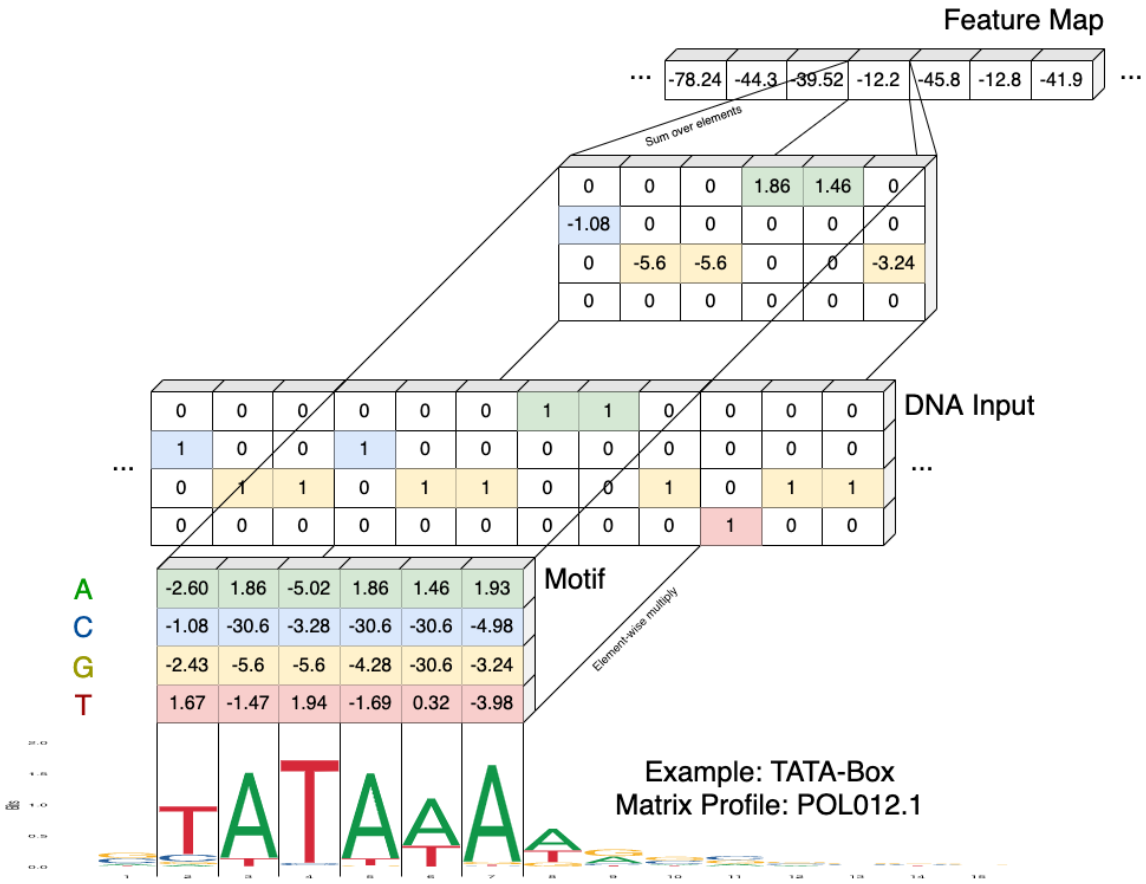

Figure 17: An illustration of the calculation used to generate a feature map for a core promoter.

6.4.6 Multiplication Layer

Now that we have explained how convolutional layers can process the input DNA sequence data to detect transcription factor motifs, we must integrate the concentrations of these transcription factors into the model. As described in section 6.1.3, these concentrations, measured by expression levels, are another crucial input. Our goal is to develop a method that effectively combines this information with the motif detection provided by the convolutional layers.

We can achieve this by multiplying the concentration of each transcription factor with its corresponding motif's feature map, as output by the convolutional layer. Multiplying the concentrations with the corresponding feature maps is a simple yet effective way to combine the information, as it inherently couples the presence of transcription factors in the cell with their matching motifs.

This coupling accomplishes two crucial objectives. Firstly, as discussed earlier, there is a concern that the model may cheat by predicting transcription factor expressions using the input expressions themselves. By coupling the two inputs together, we prevent the model from considering either the DNA sequence or the transcription factor concentrations in isolation, effectively inhibiting its ability to cheat.

Secondly, there is another concern regarding the DNA sequence: the model may not learn the actual mechanism of transcription that we desire, and instead, merely memorize which DNA sequences tend to yield higher expressions. In early work designing this model, this issue persistently came up. By coupling the two inputs using multiplication, however, we prevent the model from only looking at the DNA sequence and discourage it from memorizing which promoter DNA sequences have higher expression.

It is crucial to ensure that the model represents the interaction between these factors, as every input is essential to accurately model transcription, as described in section 4.3. If the model only considers one input at a time, it cannot accurately model the mechanisms we desire. To enforce this interaction, we directly multiply the concentrations with the untrainable convolutional layer. At this stage of the model, there are still no trainable parameters, meaning the model cannot modify this calculation to attempt cheating. In Figure 18, an illustration of the transcription factor multiplication is presented. In the figure, each transcription factor level is multiplied element-wise by the corresponding motif feature map. A nonlinear activation is also used to selectively ignore sequence locations where no motif is detected.

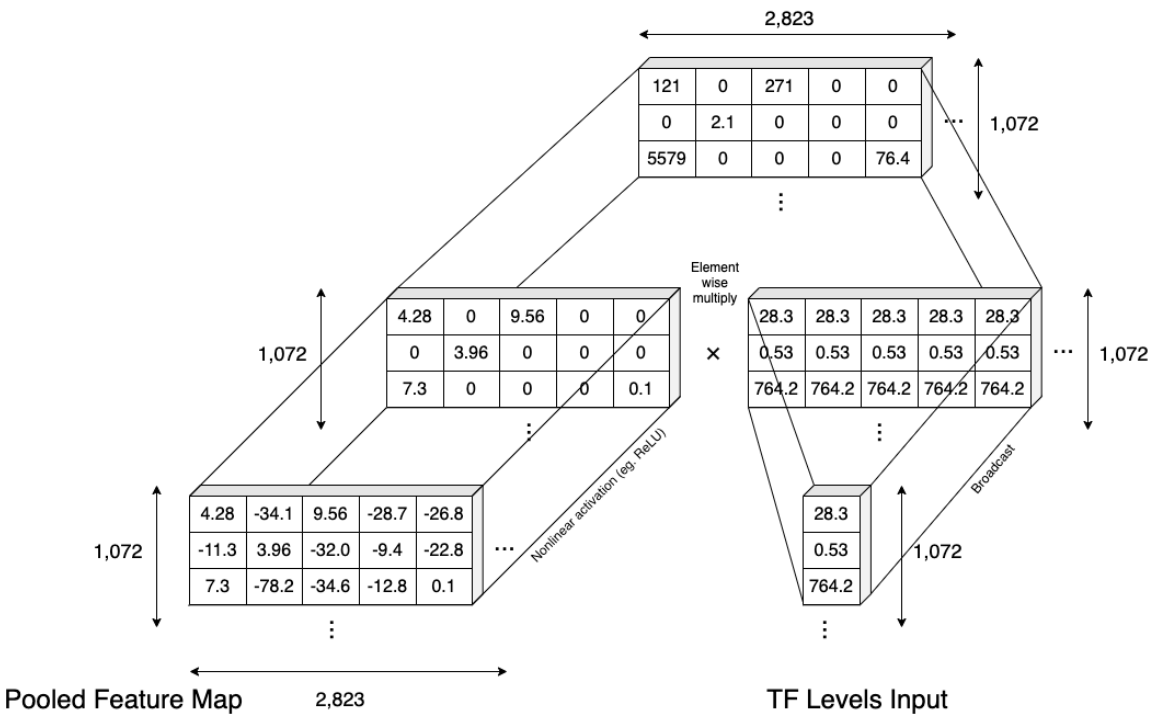

Figure 18: An illustration of the operation used to multiply transcription factors by motif feature maps.

6.4.7 Distance Factor

In section 4.3.2, we assumed that the concentration of transcription factors is constant throughout the cell nucleus. In reality, the distribution of transcription factors within the cell nucleus is far from uniform.

Recall that the process of protein synthesis involves multiple steps, with the information flow going from DNA to mRNA and finally to protein. As such, transcription factors originate from a point source (the gene that produces them) and from there move outward. Thus, one might imagine the concentration of transcription factors is likely to be higher in the vicinity of their source genes.

In reality, the actual distribution of transcription factors depends not just on their production rate, but also on their physical movement—beginning from their source gene and travelling throughout the nucleus. Unfortunately, this movement is not a simple diffusion process. The motion of transcription factors is in fact quite intricate, as they can slide along the DNA sequence, hop from one sequence to another, or undergo open diffusion [70].

Moreover, the concentration of transcription factors is also tightly controlled by the ubiq-

uitin system. Ubiquitination, a post-translational modification described in section 2.3.4, marks proteins for degradation. Transcription factors are marked for degradation quickly compared to other proteins, which further impairs the uniform distribution assumption [71][72].

Yet another layer of complexity results from considering the movement of mRNA, since genes do not directly create proteins, but instead create mRNA which is later translated into proteins. The time and distance that an mRNA molecule travels—starting from its source gene until it is translated into protein—further affects the spatial distribution of transcription factors. In eukaryotic organisms, mRNA must leave the cell nucleus to be translated in either the cytoplasm or the endoplasmic reticulum, meaning the distance that mRNA travels is indeed quite significant.

All in all, the intricate motion of transcription factors and mRNA leads to complex concentration gradients that are not fully captured in our model. Future research in this area could explore ways to model the sophisticated movement of both transcription factors and mRNA within cells, providing a more accurate representation of the processes involved.

In order to revisit the assumption in section 4.3.2, we will make a significant simplification: the concentration of a transcription factor will decrease as the distance from its source gene increases. This is not likely to hold in general, but might stir inspiration for more innovative approaches to modelling the movement of molecules involved in transcription. With this assumption, we can say that transcription factors have a smaller influence on target genes located farther away in the genome.

The concept of distence-dependent influence is illustrated in Figure 19, which is presented in contrast to Figures 5 and 6. In Figure 19, the larger physical distance between genes A and B leads to a smaller number of transcription factor proteins reaching the promoter of gene B, in turn decreasing gene B's expression.

In Figure 20, the calculation of the distance factor is presented. This calculation utilizes the transcriptional start site location of the gene being predicted and 1,072 transcription factors. The calculation also utilizes the one-hot encoded chromosome location of the gene being predicted and all 1,072 transcription factors. With these four inputs, a unique distance factor is calculated for each of the 1,072 transcription factors, which represents their proximity to the gene for which expression is being predicted.

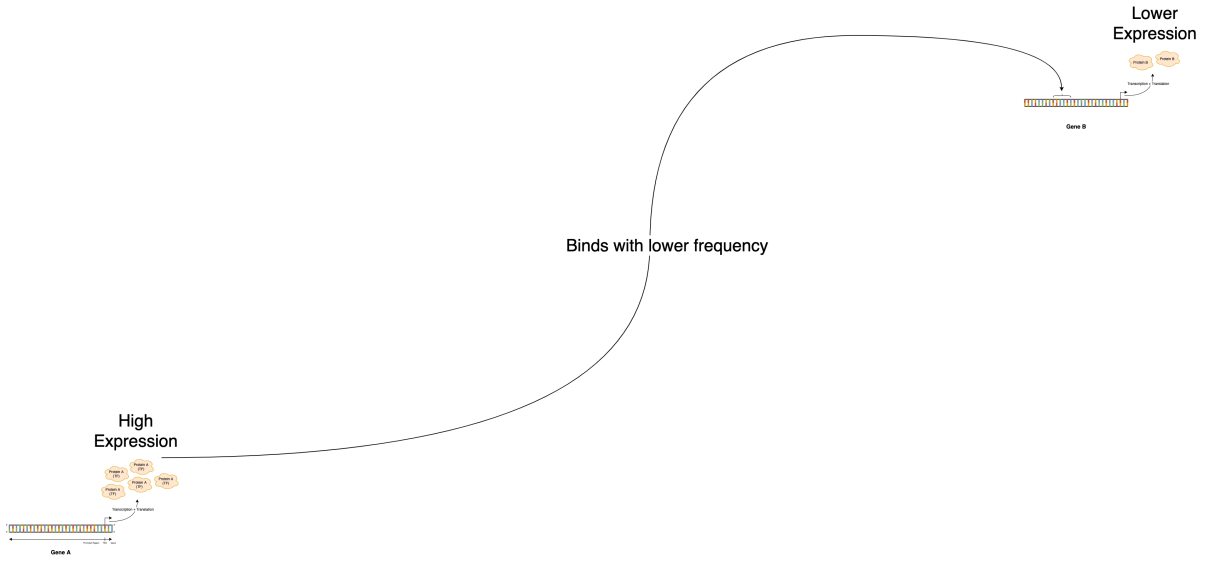

Figure 19: An illustration of the possible influence of physical distance on gene regulation.

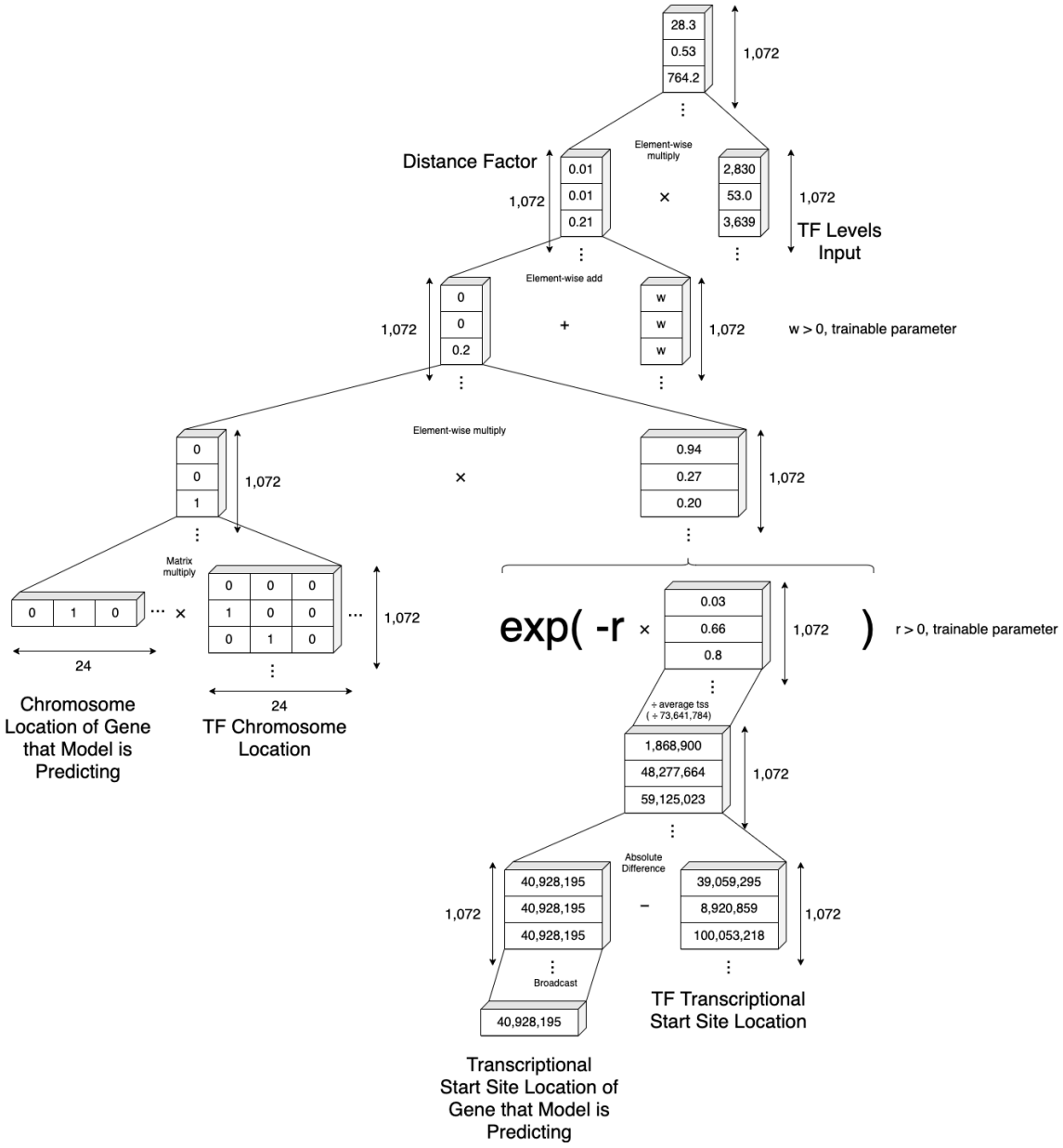

Figure 20: An illustration of the calculation of a distance factor.

6.4.8 Pooling Schedule

The concept of employing a pooling schedule is inspired by the work of Polina Govorkova et al., who demonstrated that effective synthetic promoters could be generated by concatenating the promoter's highly palindromic subsequences with a minimal core promoter. This concatenation process effectively eliminates sections of the promoter that lack high palindrome content. Palindromic subsequences serve as indicators of potential transcription factor binding sites, as most known motifs exhibit imperfect palindromes.

A key insight from Govorkova's research is that, beyond a certain distance from the TSS—in this case, after the minimal core promoter—the precise location of transcription factor binding sites becomes less critical than their mere presence. This observation aligns with the approaches documented in section 5.5.

Consequently, it is advantageous to downsample positions far from the TSS, with increased downsampling applied to more distant positions in order to further reduce dimensionality. This progressive downsampling is referred to as a pooling schedule, as it utilizes the max pooling operation, as shown in Figure 22. The dimensionality reduction achieved through this pooling schedule, shown in Figure 21, simplifies the model's architecture, enhancing the model's efficiency and performance.

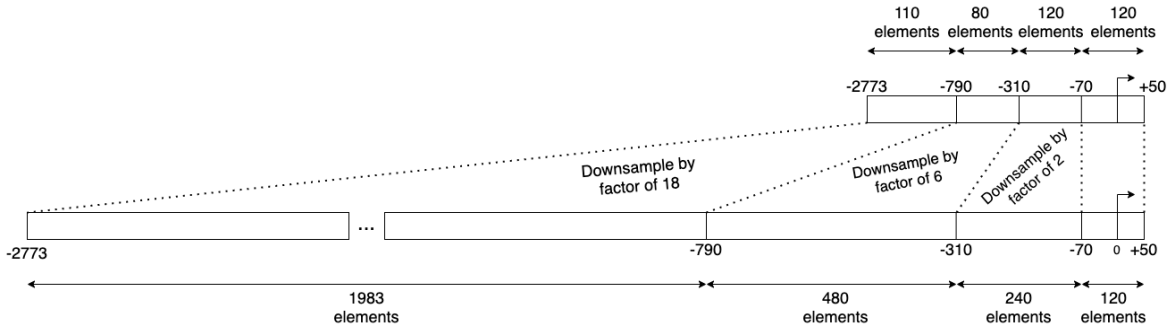

Figure 21: An illustration of the downsampling achieved by a pooling schedule.

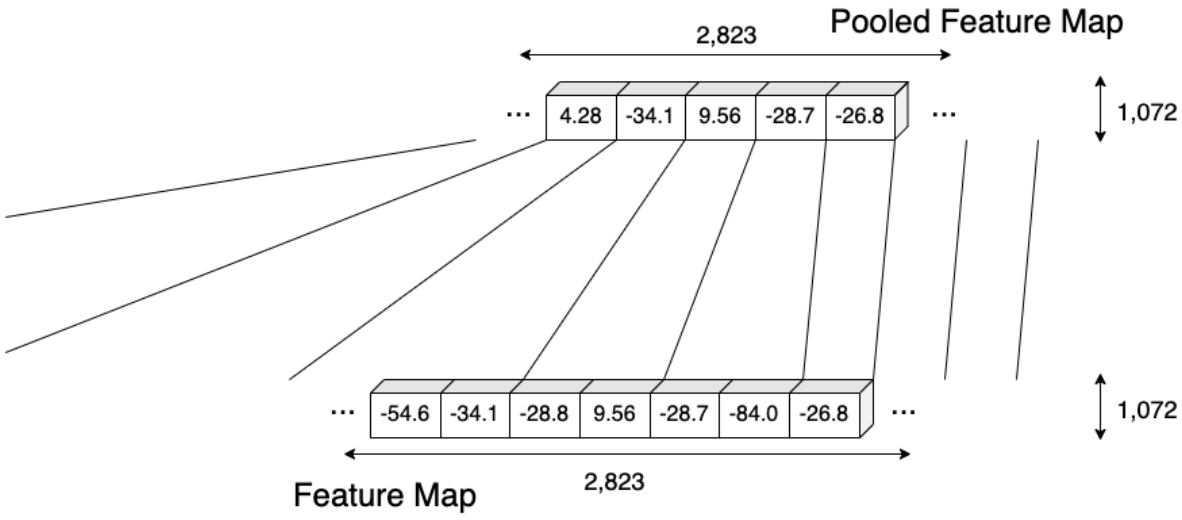

Figure 22: A schematic illustration of the pooling schedule operation.

6.5 Computational Block

At the core of our model lies the computational block, which serves as the primary component responsible for transforming processed inputs into output distributions. Encompassing a significant portion of the model's trainable parameters, these computational blocks are essential to the model's overall functionality.

In Figure 23, an illustration of the computational block is presented. In the complete model, there are twelve computational blocks repeated sequentially. These blocks consist of trainable convolutional layers, designed to capture relative spatial structure, and locally connected layers that account for absolute positional information. Additionally, layer normalizations and residual connections are incorporated to improve the stability and efficiency of the training process.

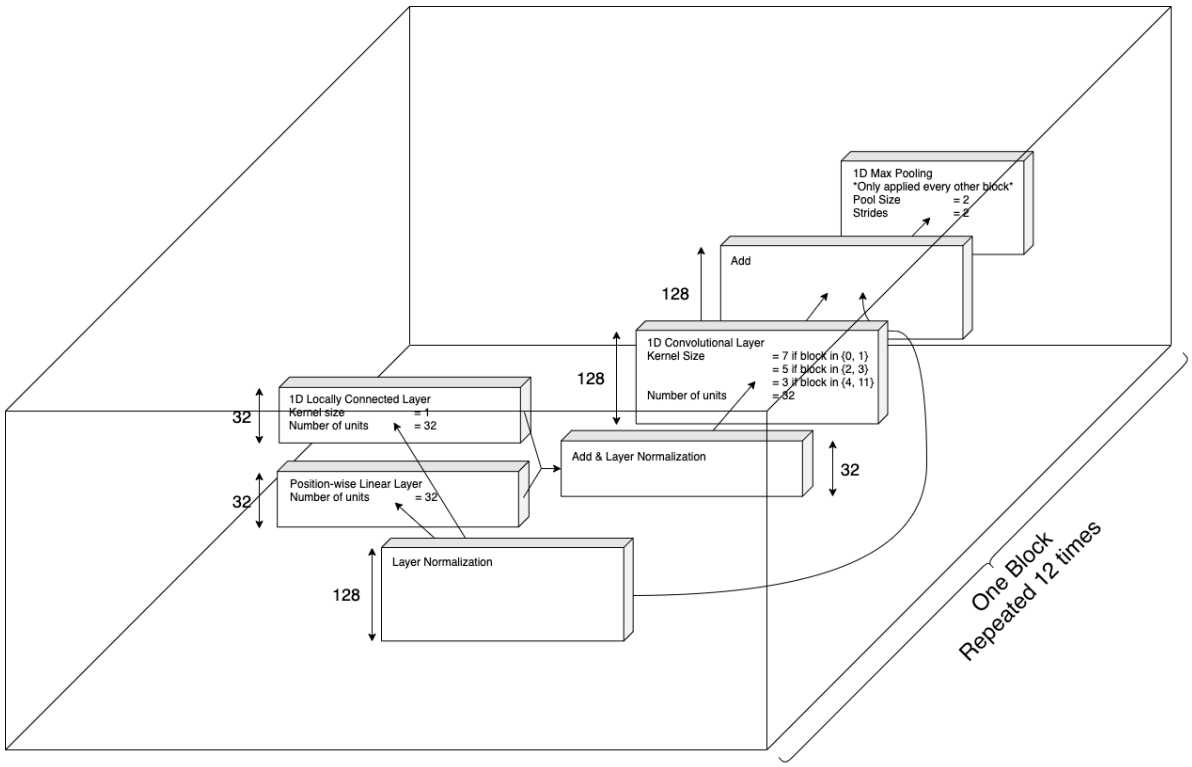

Figure 23: An illustration of the computational block.

6.6 Overall Architecture

Having discussed the key components of the model, it is now time to illustrate how they integrate to form the complete architecture. Figure 24 presents the model's structure, show-

casing the interplay between the various elements previously described. The only takeaway from Figure 24 should be how Figures 14, 15, 16, 17, 18, 20, 22, and 23 all fit together, as each of these figures is contained within Figure 24, thus rendered unreadable. A zoomable, pdf version of Figure 24 can be downloaded from this link.

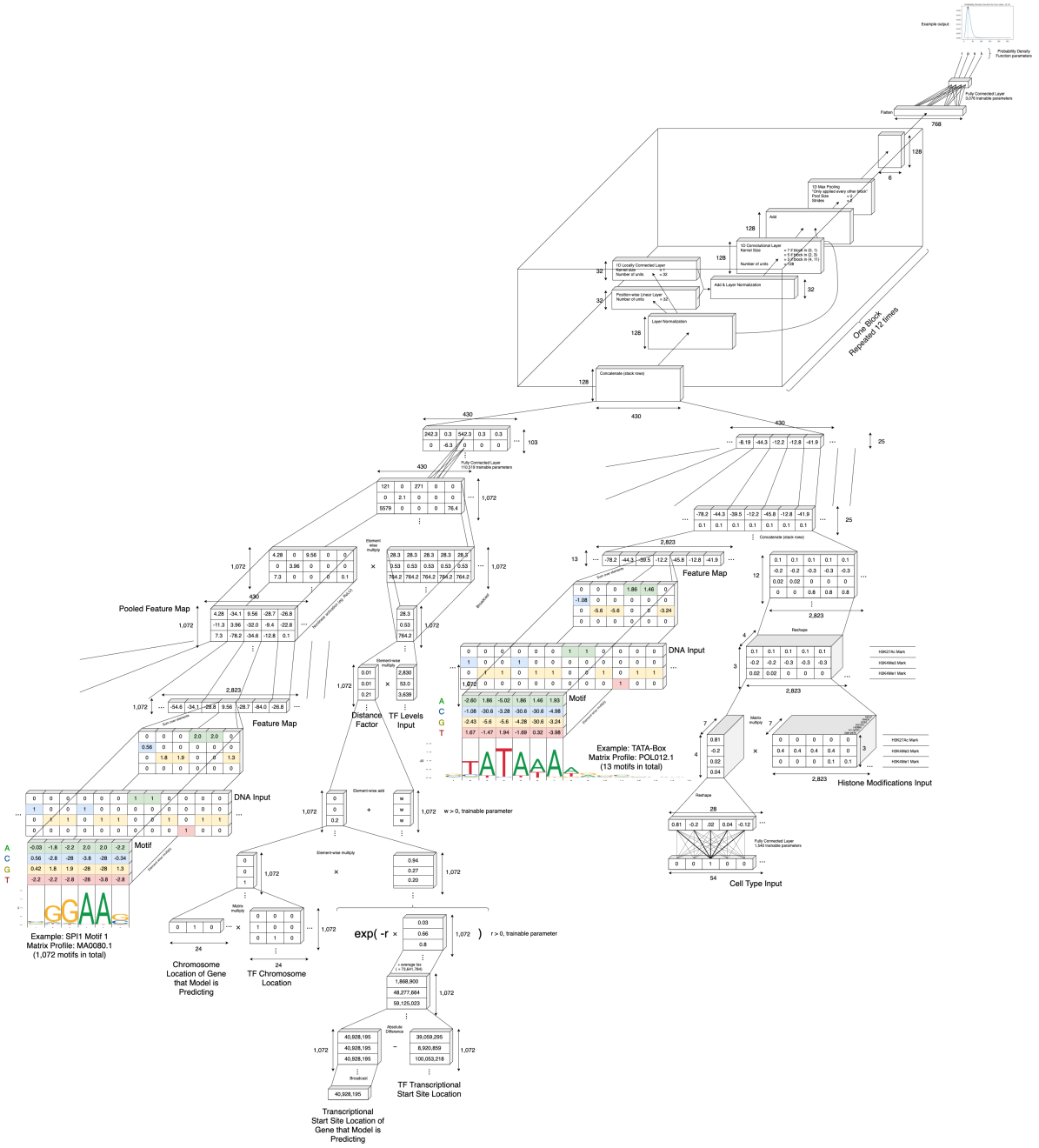

Figure 24: An illustration of overall model architecture.

6.7 Software Tools

To implement a machine learning model for gene expression prediction, a variety of software tools and frameworks are employed which provide an efficient and robust environment for model development, training, and evaluation. This section discusses the key software tools used in this project, their purpose, and how they contribute to the overall machine learning pipeline.

6.7.1 Python

Python is a versatile, high-level programming language that is widely used in various fields, including data science, machine learning, and artificial intelligence. Its simplicity, readability, and extensive library support make it an ideal choice for implementing machine learning models. For our gene expression prediction model, Python serves as the primary programming language, providing the foundation upon which all other tools and libraries are built.

6.7.2 Tensorflow

TensorFlow is an open-source machine learning library developed by Google, which enables the development and training of complex neural networks. It provides an efficient and flexible platform for implementing, training, and deploying machine learning models. In our project, TensorFlow is used to create the neural network layers, define the loss function, and optimize the model using a gradient descent algorithm.

6.7.3 NumPy, SciPy, and Matplotlib

These are popular Python libraries that provide essential functionality for numerical computing, scientific computing, and data visualization, respectively. NumPy offers powerful array and matrix operations, while SciPy provides additional functionality for optimization, linear algebra, and signal processing. Matplotlib enables the visualization of results and model performance during training and evaluation. Together, these libraries support data preprocessing, analysis, and visualization in our gene expression prediction model.

6.7.4 Google Colab

Google Colab is a cloud-based platform that offers an interactive environment for running and sharing Jupyter notebooks. In this project, Google Colab is utilized to develop, train, and evaluate the gene expression prediction model, as well as to document and share the results. A link to the Colab for this project can be found in section 11.

6.7.5 Google Cloud Compute

Google Cloud Compute is an infrastructure-as-a-service (IaaS) offering that provides scalable and customizable computing resources. Google Cloud Compute is integrated with Google Colab to provide additional resources, such as the NVIDIA A100 used in this project. The NVIDIA A100 is a high-performance GPU specifically designed for deep learning and AI workloads. With this GPU, the training process is significantly accelerated, enabling experimentation with larger and more complex neural network architectures.

6.8 Metrics of Performance

At present, there is no universally accepted metric for evaluating the performance of models that predict transcription. This lack of consensus poses a significant obstacle, as it makes it challenging to determine which models are truly better and impedes the development of more precise and efficient methods.

As far as I know, there has been no systematic examination of various metrics for the task of predicting transcription. I believe that a thorough investigation of different metrics in the context of transcription prediction is essential for advancing the field, and it is crucial for the research community to collaborate in order to establish a widely agreed-upon metric.

Metrics are vital for assessing a model's success or failure, as they provide a benchmark for measuring performance. Without a common metric, it is difficult or impossible to identify the models that outperform others and to conduct systematic comparisons between them in the research community.

In many deep learning tasks, such as object recognition and machine translation, considerable advancements have resulted from the creation of standard benchmarks that researchers use to optimize their models. For instance, the ImageNet Large Scale Visual Recognition

Challenge (ILSVRC) has been a driving force in the field of computer vision, leading to the development of advanced algorithms for object recognition and classification [73].

Similarly, the annual Workshop on Machine Translation (WMT) has spurred advancements in machine translation systems by providing a shared platform for researchers to evaluate and compare their models. A prime example of success enabled by WMT is the rapid improvement of neural machine translation models, such as the Google Translate system, which now relies on the Transformer architecture to deliver high-quality translations in real-time, outperforming earlier methods and offering a better user experience [74].

These benchmarks provide a common framework, uniting researchers' efforts and enabling them to work together toward a shared objective. Adopting a similar approach for transcription modelling may catalyze the field.

While no metric can ever be perfect and there is currently no consensus, for the purposes of this document it remains necessary to choose metrics for evaluation purposes. In the subsequent section, I present a range of metrics, each with its benefits and drawbacks, and offer a high-level comparison. Throughout this document, these metrics will be used to measure the model's performance, providing options for evaluation frameworks that can inform future research and potentially contribute to the development of a more widely accepted benchmark in the field.

6.8.1 Negative Log Likelihood

The negative log likelihood is the objective function optimized during model training, making it a natural choice for evaluation. NLL directly reflects the model's primary goal: minimizing the difference between predicted probabilities and true labels. NLL suffers from potential issues when applied to validation and test sets, however, such as poor, unstable scores due to overconfident incorrect predictions. Another issue with NLL is that models and datasets may have different baselines for NLL, making comparisons difficult. Despite these limitations, NLL remains worthy of consideration.

6.8.2 Pearson Correlation

The Pearson Correlation Coefficient is widely used for assessing the linear relationship between gene expression values [75]. The primary advantage of Pearson correlation is in directly capturing the correlation between predicted and true values. However, Pearson correlation may not accurately reflect the relationship between predicted and true expression levels in the presence of non-linear relationships or when expression values span multiple orders of magnitude, as large errors could disproportionately influence the correlation coefficient.

6.8.3 Log Pearson Correlation

The Log-transformed Pearson Correlation Coefficient applies the transformation y = log(1 + x) to predicted and true expression values before calculating the correlation. This transformation compresses the range of expression levels, mitigating the impact of large values and outliers in the metric calculation. Although the rationale for the log Pearson correlation may seem heuristic, it has been useful in measuring other models in the literature [46][44][45].

6.8.4 Spearman Rank Correlation

The Spearman Rank Correlation Coefficient is a widely used metric in biological literature for evaluating the monotonic relationship between variables, such as predicted and true gene expression values. It is particularly beneficial when relationships are non-linear or have unknown distributions. Unlike Pearson correlation, Spearman correlation is based on relative rankings, making it more robust to outliers, non-linear relationships, and non-normal distributions [76]. Importantly, log transforming the data does not affect the Spearman correlation because the log function is monotonic, meaning that it preserves the rank order of the data.

6.8.5 Log R^2

The Coefficient of Determination (R^2) is an appealing metric that measures the proportion of variation in the true data explained by a model. However, R^2 is only valid for linear models and can lead to extremely large negative values when the model performs poorly, making it less informative. The most extreme values can be mitigated by log transforming the predicted and true expression values before calculating R^2, although the metric is still

statistically invalid. While log-transformed R^2 can provide some limited insights, it should not be considered with much weight.

6.8.6 Accuracy

Accuracy is defined as the proportion of correct predictions in which the model correctly classifies a gene's expression level as above or below the dataset's median value. Thus, by random guessing the model should achieve around 50% accuracy. This metric offers a quick, easy-to-interpret measure of a model's performance. However, accuracy may oversimplify the evaluation by disregarding the nuances captured by other metrics. While it serves as a straightforward measure, it is a gross oversimplification and should not be relied upon too heavily.

7 Results

I conducted a series of tests to evaluate the model's performance and to determine how well the proposed architecture and methodology stand up to reality. The initial tests involved training the model with various combinations of inputs to assess their impact on performance. Since the selection of inputs is a crucial aspect of this model's design, it is essential to validate whether the inclusion of different inputs truly makes a significant difference. The results, as presented in Table 1, indicate that these inputs do indeed have a substantial impact on performance.

It must be noted that each of the tests presented could only be performed once, due to the prohibitive cost of training models. Therefore, it is quite possible that the numbers presented may vary slightly if the model were retrained with different initialization. This fact is unavoidable due to limited resources.

Seven models were constructed with different inputs. Each model was trained for 100,000 steps with a batch size of 128, using a learning rate of 0.0001. The results from evaluating each model on the validation set are presented in Table 1. One should refer to Appendix C (or click the symbolic links under 'Model Architecture') to understand what each row of Table 1 tangibly represents in the context of the model.

Metrics were calculated on a validation set of 65,536 examples. For Pearson correlation, log Pearson correlation, and Spearman correlation, which range from -1 to +1, a difference of at least 0.45% indicates a significantly better performance ($P < 0.05$) according to the Student's t-test, and a difference of at least 0.64% corresponds to $P < 0.01$. For accuracy, which ranges from 0 to 1, the corresponding values are 0.23% ($P < 0.05$) and 0.33% ($P < 0.01$). As log R^2 values for nonlinear models vary between negative infinity and 1, a test of statistical significance for this metric is not valid and not worth considering.

To evaluate these metrics, a point estimate of the output distribution is needed. For this purpose, I found the median to be the best performing, and all results are calculated using this estimate.

Model Architecture	Pearson Correlation	Log Pearson Correlation	Spearman Correlation	Log R^2	Accuracy
Seq	3.2%	31.2%	33.0%	1.3%	62.8%
Seq + TF	6.9%	32.4%	34.9%	-18.8%	62.4%
Seq + TF + DF	5.7%	31.7%	32.9%	3.8%	63.3%
Seq + TF + Core	3.7%	29.2%	31.1%	-4.5%	62.4%
Seq + TF + Core + HM	6.4%	55.0%	58.3%	16.0%	73.9%
Seq + Core + HM	4.6%	52.5%	56.7%	10.6%	72.9%
Seq + TF + HM	5.6%	53.6%	56.0%	14.0%	73.4%

Table 1: A table showcasing the final validation results from testing seven model architectures.

The optimal model, selected from all tested configurations, is Seq + TF + Core + HM. This model architecture was subjected to an extended training period (around 500,000 training steps) in an effort to achieve the highest performance attainable. This model was then evaluated on the test set to gain an idea of its ultimate generalization performance. The test set results of this final model are presented in the table below.

Model Architecture	Pearson Correlation	Log Pearson Correlation	Spearman Correlation	Log R^2	Accuracy
Final Model	6.9%	53.1%	56.6%	15.9%	73.9%

Table 2: A table showing the test set performance of the final model under an extended training period.

For each of the seven models tested, I generated a number of plots including 1) a plot of predicted vs true expression values on the validation set, 2) a plot of the training and validation loss over the training process, 3) a plot of the training and validation, Pearson and log Pearson correlations over the training process, and 4) a plot of the training and validation accuracy over the training process. These plots are presented in Appendix D.

8 Discussion

To interpret these results, it is useful to first compare rows of the table above against each other, to assess whether inputs which were identified as important do indeed improve performance. Keep in mind that T > 1.65 signifies significantly better performance with P < 0.05, and T > 2.33 indicates significantly better performance with P < 0.01. Conversely, I use negative T values to denote significantly worse performance, with T < -1.65 corresponding to P < 0.05, and T < -2.33 corresponding to P < 0.01. All results are reported on the validation set.

8.1 Seq vs. Seq + TF

The first comparison that can be made, shown in Table 3, assesses the benefit of including transcription factor expressions as an input (as discussed in section 6.1.3) against a baseline of just using the promoter & enhancer sequence.

Model Architecture	Pearson Correlation	Log Pearson Correlation	Spearman Correlation	Log R^2	Accuracy
Seq	3.2%	31.2%	33.0%	1.3%	62.8%
Seq + TF	6.9%	32.4%	34.9%	-18.8%	62.4%
T	13.4	4.3	6.9	N/A	-2.9

Table 3: A table comparing the Seq architecture against the Seq + TF architecture.

According to all three correlation metrics, there is a significant benefit to including the transcription factors as input, compared to only using sequence data. This is exactly in line with expectations, according to the discussion in section 4.3.2. It's worth noting that the accuracy and log R^2 metrics do not follow the same trend as the correlation metrics. This is an interesting curiosity, and the reason is not exactly clear. However, it may simply be the case these metrics do not accurately reflect the model performance. Log R^2 is technically invalid for nonlinear models, and for accuracy, it could be the case that too much information is lost by treating each prediction as either right or wrong, as described in section 6.8.6. This phenomenon is observed in other comparisons and I would speculate the reasons are the same.

8.2 Seq + TF vs. Seq + TF + DF

The next comparison, shown in Table 4, assesses the benefit of including a distance factor in the model (as discussed in section 6.4.7).

Model Architecture	Pearson Correlation	Log Pearson Correlation	Spearman Correlation	Log R^2	Accuracy
Seq + TF	6.9%	32.4%	34.9%	-18.8%	62.4%
Seq + TF + DF	5.7%	31.7%	32.9%	3.8%	63.3%
T	-4.3	-2.5	-7.2	N/A	6.3

Table 4: A table comparing the Seq + TF architecture against the Seq + TF + DF architecture.

It is interesting to note that the distance factor makes the model perform significantly worse on all metrics except for Log R^2 and accuracy. I would speculate that using a single distance factor is far too simplified a representation of the complex dynamics of transcription factor motion throughout the genome. This distance factor also ignores mRNA's movement outside the nucleus before translation, which is likely a critical deficit. Lastly, one other possible source of error could be that a distance factor might weaken the coupling between the DNA sequence and transcription factors, as described in section 6.4.6. For these reasons, the distance factor does not appear in any other model architecture.

8.3 Seq + TF vs. Seq + TF + Core

The following comparison, shown in Table 5 assesses the benefit of having the model search for core promoter elements, compared to just transcription factor motifs (as discussed in section 6.4.5).

Model Architecture	Pearson Correlation	Log Pearson Correlation	Spearman Correlation	Log R^2	Accuracy
Seq + TF	6.9%	32.4%	34.9%	-18.8%	62.4%
Seq + TF + Core	3.7%	29.2%	31.1%	-4.5%	62.4%
T	-11.6	-11.6	-13.8	N/A	0

Table 5: A table comparing the Seq + TF architecture against the Seq + TF + Core architecture.

Interestingly, the model shows significantly worse performance on all correlation metrics, while maintaining accuracy and improving log R^2. This outcome may be attributed to core promoter elements not being linked to transcription factor concentrations, which was identified as important in section 6.4.6. This decoupling allows the model to potentially focus solely on the DNA sequence input without considering its interaction with transcription factors, leading the model to potentially cheat. As the results are based on the validation set, taking such a shortcut could result in the poor generalization performance reflected here.

8.4 Seq + TF + Core vs. Seq + TF + Core + HM

The next comparison, shown in Table 6, assesses the benefit of including histone modification data as input (as discussed in section 6.1.6).

Model Architecture	Pearson Correlation	Log Pearson Correlation	Spearman Correlation	Log R^2	Accuracy
Seq + TF + Core	3.7%	29.2%	31.1%	-4.5%	62.4%
Seq + TF + Core + HM	6.4%	55.0%	58.3%	16.0%	73.9%
T	-4.3	-2.5	-7.2	N/A	6.3

Table 6: A table comparing the Seq + TF + Core architecture against the Seq + TF + Core + HM architecture.

Interestingly, incorporating histone modifications leads to a significant performance improvement across all metrics. I suggest this large improvement is due to histone modifications being a reliable indicator of gene expression, with their mere presence providing substantial information about the level of transcription. Consequently, it is relatively easy for the model to learn the relationship between histone modifications and expression. In contrast, the interaction of transcription factors is a more complex and challenging process to learn, making exceptional performance harder to achieve.

Additionally, it is possible that the histone modification input enhances the value of the transcription factor input. Since the binding of transcription factors is modulated by histone modifications, the relationship between transcription factors and expression levels largely depends on histone modification information. Therefore, histone modification information is likely quite important for accurately modeling the interaction between transcription factors

and DNA, and the next comparison expands on this idea.

8.5 Seq + Core + HM vs. Seq + Core + TF + HM

The next comparison, shown in Table 7, assesses the benefit of transcription factors against the baseline of using sequence data, core promoter elements, and histone modifications.

Model Architecture	Pearson Correlation	Log Pearson Correlation	Spearman Correlation	Log R^2	Accuracy
Seq + Core + HM	4.6%	52.5%	56.7%	10.6%	72.9%
Seq + TF + Core + HM	6.4%	55.0%	58.3%	16.0%	73.9%
T	6.5	9.1	5.8	N/A	7.2

Table 7: A table comparing the Seq + Core + HM architecture against the Seq + TF + Core + HM architecture.

As anticipated, incorporating transcription factor expression enhances all metrics, with a more significant improvement than observed in the comparison of section 8.1. This suggests that histone modifications likely boost the value of transcription factor input and reinforces the notion that including all crucial factors in transcription is essential for accurately modeling the underlying mechanisms, as noted in section 4.3. It is plausible that incorporating additional inputs, such as DNA methylation, noncoding RNAs, and others, would further improve performance and similarly increase the value of all other inputs.

9 Next Steps

Many simplifications, approximations, shortcuts, and assumptions were made during the development of our model, and the data utilized were imperfect. As a result, there are a wide variety of opportunities to enhance and refine the model in the future. The primary areas for improvement are listed below, which, if addressed, would significantly improve the model's performance.

- Better data. In machine learning, it is commonly noted that the quality of a model largely depends on the quality of its training data. Therefore, improved data from every source will likely make the largest improvements in the quality of the model. Listed below are the most significant ways of accomplishing this.
  - Accounting for expression TPM being used instead of transcription frequencies, as noted in section 6.1.1. This could be accomplished in the following ways.
    - Account for post-transcriptional modifications.
    - Account for mRNA degradation.
    - Collecting data specifically on transcription frequency, rather than mRNA abundance.
  - Histone modifications measured in the 54 GTEx cell types, ideally in the same tissues as the expression measurements.
  - A more accurate complete set of binding motifs for transcription factors.
  - A more complete list of transcription factors.
- Including more inputs to the model.
  - Coactivators and corepressors.
  - DNA Methylation.
  - Noncoding RNAs.
  - Other histone modifications besides the three marks used.
  - Accounting for chromatin geometry, linker DNA locations, etc.

  - Accounting for DNA bending.
  - Accounting for mediator complex interactions.
  - Inputting a wider DNA sequence—2,823 base pairs may not be enough.
    * Including more downstream regions that transcription factors might bind to.
  - Accounting for the different DNA of different patients in GTEx experiments.

- Optimizing model architecture. There were countless structural decisions that could be reevaluated and yield improvement. Any part of Figure 24 could change and possibly improve results.

- More accurately modelling the movement of transcription factors throughout the cell nucleus, in order to better capture the concentration gradient and binding frequencies of transcription factors.
  - Accounting for the movement of mRNA.
  - Accounting for the degradation of mRNA.
  - Accounting for transcription factors sliding along DNA sequences.
  - Accounting for transcription factors hopping across DNA sequences.
  - Accounting for transcription factors diffusing openly throughout the cell.
  - Accounting for ubiquitination and degradation of transcription factors.

- Accounting for changes in state over the cell cycle. Many genes express differently in different cell phases, yet this is conveniently ignored in our model.

Importantly, the scale and variety of factors identified above exceed the capabilities of a single individual, and beneficial progress is to be achieved only through the collaborative efforts of a dedicated community. To be sure, the advancement of vision models to human-like capabilities, the development of machine translation, and the creation of sophisticated language models were all made possible by the collective commitment of researchers who recognized the importance of their work and devoted their efforts to optimizing models. Fully exploring transcription is beyond the scope of one person, indeed requiring the concerted efforts of a large group to address and ultimately solve its challenges.

Yet, there is an undeniable sense of promise on the horizon. The path forward is clear, and the rewards of tackling transcription are, as I have argued, immense. Achieving a complete, accurate predictive model of transcription will enable us to confront formidable challenges and unlock unprecedented possibilities, paving the way for groundbreaking advancements in biotechnology.

The next step, after accurately predicting gene expression in the human genome, is to extend our model to DNA sequences transfected into human cells. As sequences up to 4,700 base pairs in length can be effectively transfected using the AAV9 virus, the next task is determining these sequences' expression patterns. Though many similarities with the human genome will exist, transfected sequences may interact differently with chromatin, histone modifications, and other factors. Designing and refining a predictive model that can handle transfected sequences represents the next frontier in mastering protein expression in human cells.

With a robust predictive model capable of handling transfected sequences, it is possible to confidently determine the protein output from any DNA sequence in any specific cell type or state. Such a model can be used to design appropriate promoters for any desired expression levels across various cells. The potential of such technology is enormous—from expressing toxic proteins solely in cancer cells to applications currently unimagined.

As I look ahead, I envision a world where our collective efforts have manifest the full potential of transcription and protein expression. In such a world, an extensive knowledge of these complex biomolecules would transform the fields of biotechnology and medicine, enabling us to conquer formidable challenges and achieve monumental progress. Ultimately, gaining proficiency in proteins holds the power to open a Pandora's box of possibilities, presenting both unprecedented opportunities and potential risks. As we continue on this journey, it is imperative that we exercise wisdom and restraint, ensuring that our pursuit leads to the betterment of humanity rather than unintended consequences.

10 Data Availability

I openly provide the complete, preprocessed dataset used for supervised training and evaluation. I name this dataset 'IBBME_TR300M'—the Institute of Biomaterials and Biomedical Engineering Transcription Regression dataset, with over 300 million supervised examples. The IBBME_TR300M dataset has a download size of 1.17 GB and can be downloaded from this link.

The dataset is stored efficiently using compressed numpy arrays, which simplifies the process for other researchers who wish to work with this dataset. The preprocessing pipeline, which was employed to create these numpy arrays from multiple sources, is detailed in Appendix E. Links to the multiple sources of this data are provided below.

Data sources:

1. The GTEx Portal
2. GENCODE v26 Release
3. JASPAR
4. U of T Applied Protein Engineering Lab SQL Server
5. The Human Transcription Factors
6. Histone Modification Tracks
   (a) H3K4Me1 Mark
   (b) H3K4Me3 Mark
   (c) H3K27Ac Mark

11 Code Availability

I openly provide all of the code required to generate the dataset from scratch, define the model, modify architecture, train the model, and evaluate the model on Github. In addition, I provide a self-contained Google Colab environment which contains all of the packages, functionality, and code required to modify the model architecture and retrain it on the dataset—all in just a few clicks.

References

[1] S. Lefebvre, L. Bürglen, S. Reboullet, *et al.*, "Identification and characterization of a spinal muscular atrophy-determining gene," *Cell*, vol. 80, no. 1, pp. 155–165, 1995.

[2] *See how zolgensma® (onasemnogene abeparvovec-xioi) works*. [Online]. Available: https://www.zolgensma.com/how-zolgensma-works.

[3] L. Shapiro and R. Losick, "Delivering the message: How a novel technology enabled the rapid development of effective vaccines," *Cell*, vol. 184, no. 21, pp. 5271–5274, 2021.

[4] E. A. Ostrander, *Central dogma*. [Online]. Available: https://www.genome.gov/genetics-glossary/Central-Dogma.

[5] P. Ganguly, *Transcription*. [Online]. Available: https://www.genome.gov/genetics-glossary/Transcription.

[6] S. A. Bates, *Deoxyribonucleic acid (dna)*. [Online]. Available: https://www.genome.gov/genetics-glossary/Deoxyribonucleic-Acid.

[7] E. Green, *Gene*. [Online]. Available: https://www.genome.gov/genetics-glossary/Gene.

[8] O. Fornes, J. A. Castro-Mondragon, A. Khan, *et al.*, "Jaspar 2020: Update of the open-access database of transcription factor binding profiles," *Nucleic acids research*, vol. 48, no. D1, pp. D87–D92, 2020.

[9] S. A. Lambert, A. Jolma, L. F. Campitelli, *et al.*, "The human transcription factors," *Cell*, vol. 172, no. 4, pp. 650–665, 2018.

[10] P. P. Liu, *Chromatin*. [Online]. Available: https://www.genome.gov/genetics-glossary/Chromatin.

[11] D. A. Gilchrist, *Histone*. [Online]. Available: https://www.genome.gov/genetics-glossary/histone.

[12] H. B. Sun, J. Shen, and H. Yokota, "Size-dependent positioning of human chromosomes in interphase nuclei," *Biophysical journal*, vol. 79, no. 1, pp. 184–190, 2000.

[13] A. J. Bannister and T. Kouzarides, "Regulation of chromatin by histone modifications," *Cell research*, vol. 21, no. 3, pp. 381–395, 2011.

[14] L. D. Moore, T. Le, and G. Fan, "Dna methylation and its basic function," *Neuropsychopharmacology*, vol. 38, no. 1, pp. 23–38, 2013.

[15] T. R. Cech and J. A. Steitz, "The noncoding rna revolution—trashing old rules to forge new ones," *Cell*, vol. 157, no. 1, pp. 77–94, 2014.

[16] S. Clancy *et al.*, "Rna splicing: Introns, exons and spliceosome," *Nature Education*, vol. 1, no. 1, p. 31, 2008.

[17] D. Day and M. F. Tuite, "Post-transcriptional gene regulatory mechanisms in eukaryotes: An overview," *Journal of endocrinology*, vol. 157, no. 3, pp. 361–371, 1998.

[18] R. G. Krishna and F. Wold, "Post-translational modifications of proteins," *Methods in protein sequence analysis*, pp. 167–172, 1993.

[19] OpenAI, *Gpt-4 technical report*, 2023. arXiv: 2303.08774 [cs.CL].

[20] J. E. Moore, M. J. Purcaro, H. E. Pratt, *et al.*, "Expanded encyclopaedias of dna elements in the human and mouse genomes," *Nature*, vol. 583, no. 7818, pp. 699–710, 2020.

[21] E. B. Lewis, "A gene complex controlling segmentation in drosophila," *Nature*, vol. 276, no. 5688, pp. 565–570, 1978.

[22] F. Jacob, D. Perrin, C. Sánchez, J. Monod, *et al.*, "The operon: A group of genes whose expression is co-ordinated by an operator.," *Compte Rendu de l'Academie des Sciences*, vol. 250, pp. 1727–1729, 1960.

[23] F. Jacob and J. Monod, "Genetic regulatory mechanisms in the synthesis of proteins," *Journal of molecular biology*, vol. 3, no. 3, pp. 318–356, 1961.

[24] V. G. Allfrey, R. Faulkner, and A. Mirsky, "Acetylation and methylation of histones and their possible role in the regulation of rna synthesis," *Proceedings of the National Academy of Sciences*, vol. 51, no. 5, pp. 786–794, 1964.

[25] J. Hurwitz, "The discovery of rna polymerase," *Journal of Biological Chemistry*, vol. 280, no. 52, pp. 42 477–42 485, 2005.

[26] R. G. Roeder and W. J. Rutter, "Multiple forms of dna-dependent rna polymerase in eukaryotic organisms," *Nature*, vol. 224, pp. 234–237, 1969.

[27] D. R. Engelke, S.-Y. Ng, B. Shastry, and R. G. Roeder, "Specific interaction of a purified transcription factor with an internal control region of 5s rna genes," *Cell*, vol. 19, no. 3, pp. 717–728, 1980.

[28] P. M. Flanagan, R. J. Kelleher III, M. H. Sayre, H. Tschochner, and R. D. Kornberg, "A mediator required for activation of rna polymerase ii transcription in vitro," *Nature*, vol. 350, no. 6317, pp. 436–438, 1991.

[29] M. Hecker, S. Lambeck, S. Toepfer, E. Van Someren, and R. Guthke, "Gene regulatory network inference: Data integration in dynamic models—a review," *Biosystems*, vol. 96, no. 1, pp. 86–103, 2009.

[30] M. Banf and S. Y. Rhee, "Computational inference of gene regulatory networks: Approaches, limitations and opportunities," *Biochimica et Biophysica Acta (BBA)-Gene Regulatory Mechanisms*, vol. 1860, no. 1, pp. 41–52, 2017.

[31] J. M. Stuart, E. Segal, D. Koller, and S. K. Kim, "A gene-coexpression network for global discovery of conserved genetic modules," *science*, vol. 302, no. 5643, pp. 249–255, 2003.

[32] R. Steuer, J. Kurths, C. O. Daub, J. Weise, and J. Selbig, "The mutual information: Detecting and evaluating dependencies between variables," *Bioinformatics*, vol. 18, no. suppl_2, S231–S240, 2002.

[33] A. Rao, A. O. Hero III, D. J. States, and J. D. Engel, "Using directed information to build biologically relevant influence networks," in *Computational Systems Bioinformatics: (Volume 6)*, World Scientific, 2007, pp. 145–156.

[34] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of theoretical biology*, vol. 22, no. 3, pp. 437–467, 1969.

[35] R. Thomas, "Boolean formalization of genetic control circuits," *Journal of theoretical biology*, vol. 42, no. 3, pp. 563–585, 1973.

[36] S. Martin, Z. Zhang, A. Martino, and J.-L. Faulon, "Boolean dynamics of genetic regulatory networks inferred from microarray time series data," *Bioinformatics*, vol. 23, no. 7, pp. 866–874, 2007.

[37] N. S. Holter, A. Maritan, M. Cieplak, N. V. Fedoroff, and J. R. Banavar, "Dynamic modeling of gene expression data," *Proceedings of the National Academy of Sciences*, vol. 98, no. 4, pp. 1693–1698, 2001.

[38] E. Sakamoto and H. Iba, "Inferring a system of differential equations for a gene regulatory network by using genetic programming," in *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546)*, IEEE, vol. 1, 2001, pp. 720–726.

[39] A. Climescu-Haulica and M. D. Quirk, "A stochastic differential equation model for transcriptional regulatory networks," *BMC bioinformatics*, vol. 8, pp. 1–9, 2007.

[40] S. Kimura, K. Ide, A. Kashihara, *et al.*, "Inference of s-system models of genetic networks using a cooperative coevolutionary algorithm," *Bioinformatics*, vol. 21, no. 7, pp. 1154–1163, 2005.

[41] C. Rangel, J. Angus, Z. Ghahramani, *et al.*, "Modeling t-cell activation using gene expression profiling and state-space models," *Bioinformatics*, vol. 20, no. 9, pp. 1361–1372, 2004.

[42] M. A. Beer and S. Tavazoie, "Predicting gene expression from sequence," *Cell*, vol. 117, no. 2, pp. 185–198, 2004.

[43] Y. Yuan, L. Guo, L. Shen, and J. S. Liu, "Predicting gene expression from sequence: A reexamination," *PLoS computational biology*, vol. 3, no. 11, e243, 2007.

[44] D. R. Kelley, Y. A. Reshef, M. Bileschi, D. Belanger, C. Y. McLean, and J. Snoek, "Sequential regulatory activity prediction across chromosomes with convolutional neural networks," *Genome research*, vol. 28, no. 5, pp. 739–750, 2018.

[45] D. R. Kelley, "Cross-species regulatory sequence activity prediction," *PLoS computational biology*, vol. 16, no. 7, e1008050, 2020.

[46] Ž. Avsec, V. Agarwal, D. Visentin, *et al.*, "Effective gene expression prediction from sequence by integrating long-range interactions," *Nature methods*, vol. 18, no. 10, pp. 1196–1203, 2021.

[47] R. Karlić, H.-R. Chung, J. Lasserre, K. Vlahoviček, and M. Vingron, "Histone modification levels are predictive for gene expression," *Proceedings of the National Academy of Sciences*, vol. 107, no. 7, pp. 2926–2931, 2010.

[48] R. Singh, J. Lanchantin, G. Robins, and Y. Qi, "Deepchrome: Deep-learning for predicting gene expression from histone modifications," *Bioinformatics*, vol. 32, no. 17, pp. i639–i648, 2016.

[49] A. Natarajan, G. G. Yardımcı, N. C. Sheffield, G. E. Crawford, and U. Ohler, "Predicting cell-type–specific gene expression from regions of open chromatin," *Genome research*, vol. 22, no. 9, pp. 1711–1722, 2012.

[50] H. Zhong, S. Kim, D. Zhi, and X. Cui, "Predicting gene expression using dna methylation in three human populations," *PeerJ*, vol. 7, e6757, 2019.

[51] D. B. Seal, V. Das, S. Goswami, and R. K. De, "Estimating gene expression from dna methylation and copy number variation: A deep learning regression model for multiomics integration," *Genomics*, vol. 112, no. 4, pp. 2833–2841, 2020.

[52] Z. Ouyang, Q. Zhou, and W. H. Wong, "Chip-seq of transcription factors predicts absolute and differential gene expression in embryonic stem cells," *Proceedings of the National Academy of Sciences*, vol. 106, no. 51, pp. 21 521–21 526, 2009.

[53] C. Cheng, R. Alexander, R. Min, *et al.*, "Understanding transcriptional regulation by integrative analysis of transcription factor binding data," *Genome research*, vol. 22, no. 9, pp. 1658–1667, 2012.

[54] X. He, M. A. H. Samee, C. Blatti, and S. Sinha, "Thermodynamics-based models of transcriptional regulation by enhancers: The roles of synergistic activation, cooperative binding and short-range repression," *PLoS computational biology*, vol. 6, no. 9, e1000935, 2010.

[55] J. Gertz, E. D. Siggia, and B. A. Cohen, "Analysis of combinatorial cis-regulation in synthetic and genomic promoters," *Nature*, vol. 457, no. 7226, pp. 215–218, 2009.

[56] D. H. Nguyen and P. D'haeseleer, "Deciphering principles of transcription regulation in eukaryotic genomes," *Molecular systems biology*, vol. 2, no. 1, pp. 2006–0012, 2006.

[57] Y. Zhao, M.-C. Li, M. M. Konaté, *et al.*, "Tpm, fpkm, or normalized counts? a comparative study of quantification measures for the analysis of rna-seq data from the nci patient-derived models repository," *Journal of translational medicine*, vol. 19, no. 1, pp. 1–15, 2021.

[58] G. Consortium, "The gtex consortium atlas of genetic regulatory effects across human tissues," *Science*, vol. 369, no. 6509, pp. 1318–1330, 2020.

[59] V. A. Schneider, T. Graves-Lindsay, K. Howe, *et al.*, "Evaluation of grch38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly," *Genome research*, vol. 27, no. 5, pp. 849–864, 2017.

[60] J. T. Kadonaga, "The dpe, a core promoter element for transcription by rna polymerase ii," *Experimental & molecular medicine*, vol. 34, no. 4, pp. 259–264, 2002.

[61] A. Frankish, S. Carbonell-Sala, M. Diekhans, *et al.*, "Gencode: Reference annotation for the human and mouse genomes in 2023," *Nucleic acids research*, vol. 51, no. D1, pp. D942–D949, 2023.

[62] V. W. Zhou, A. Goren, and B. E. Bernstein, "Charting histone modifications and the functional organization of mammalian genomes," *Nature Reviews Genetics*, vol. 12, no. 1, pp. 7–18, 2011.

[63] J. A. Castro-Mondragon, R. Riudavets-Puig, I. Rauluseviciute, *et al.*, "Jaspar 2022: The 9th release of the open-access database of transcription factor binding profiles," *Nucleic acids research*, vol. 50, no. D1, pp. D165–D173, 2022.

[64] Y. Di, D. W. Schafer, J. S. Cumbie, and J. H. Chang, "The nbp negative binomial model for assessing differential gene expression from rna-seq," *Statistical applications in genetics and molecular biology*, vol. 10, no. 1, 2011.

[65] X. Ren and P.-F. Kuan, "Negative binomial additive model for rna-seq data analysis," *BMC bioinformatics*, vol. 21, pp. 1–15, 2020.

[66] I. J. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016, http://www.deeplearningbook.org.

[67] Y. Lu and J. Lu, *A universal approximation theorem of deep neural networks for expressing probability distributions*, 2020. arXiv: 2004.08867 [cs.LG].

[68] M. Beckstette, R. Homann, R. Giegerich, and S. Kurtz, "Fast index based algorithms and software for matching position specific scoring matrices," *BMC bioinformatics*, vol. 7, no. 1, pp. 1–25, 2006.

[69] M. C. Frith, M. C. Li, and Z. Weng, "Cluster-buster: Finding dense clusters of motifs in dna sequences," *Nucleic acids research*, vol. 31, no. 13, pp. 3666–3668, 2003.

[70] H. G. Schmidt, S. Sewitz, S. S. Andrews, and K. Lipkow, "An integrated model of transcription factor diffusion shows the importance of intersegmental transfer and quaternary protein structure for target site finding," *PLOS one*, vol. 9, no. 10, e108575, 2014.

[71] M. Hochstrasser and D. Kornitzer, "Ubiquitin-dependent degradation of transcription regulators," *Ubiquitin and the Biology of the Cell*, pp. 279–302, 1998.

[72] J. Desterro, M. Rodriguez, and R. Hay*, "Regulation of transcription factors by protein degradation," *Cellular and Molecular Life Sciences CMLS*, vol. 57, pp. 1207–1219, 2000.

[73] O. Russakovsky, J. Deng, H. Su, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[74] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[75] S. K. Schulze, R. Kanwar, M. Gölzenleuchter, T. M. Therneau, and A. S. Beutler, "Sere: Single-parameter quality control and sample comparison for rna-seq," *BMC genomics*, vol. 13, no. 1, pp. 1–9, 2012.

[76] S. de Siqueira Santos, D. Y. Takahashi, A. Nakata, and A. Fujita, "A comparative study of statistical methods used to identify dependencies between gene expression signals," *Briefings in bioinformatics*, vol. 15, no. 6, pp. 906–918, 2014.

[77] G. Marsaglia, "Xorshift rngs," *Journal of Statistical software*, vol. 8, pp. 1–6, 2003.

A Example Generation

The entire dataset is stored in compressed numpy files, so there are several options for converting these to a format that Tensorflow can use for training. I found the best approach, all things considered, to be loading all of these arrays into python numpy arrays and generating examples on the fly during training, despite having the drawback of requiring around 20 GB of application memory. Another option would be to construct a custom Tensorflow dataset structure from which examples could be loaded serially, which is the fastest method and requires little RAM. The problem with this method is that to load examples serially they must be represented serially, and usually uncompressed for maximal speed. Unfortunately, there are 340 million examples in this dataset, and each example contains several inputs which together take up more than 1 MB. Thus, storing the full dataset serially would require storage space on the order of 340 TB.

Clearly, such a solution is intractable, and therefore the best option is to generate examples on the fly during training. The generation of examples on the fly is represented in Figure 25. One immediate problem with this solution is generating examples in a shuffled way. Shuffling the dataset prior to training is a crucial step in the machine learning pipeline, as it ensures that the data is randomly distributed throughout the training set. Randomization helps prevent the model from learning spurious correlations or biases present in subsections of the data, which could lead to overfitting and reduced generalization performance. Generating examples on the fly poses a significant challenge as to how and when the dataset will be shuffled.

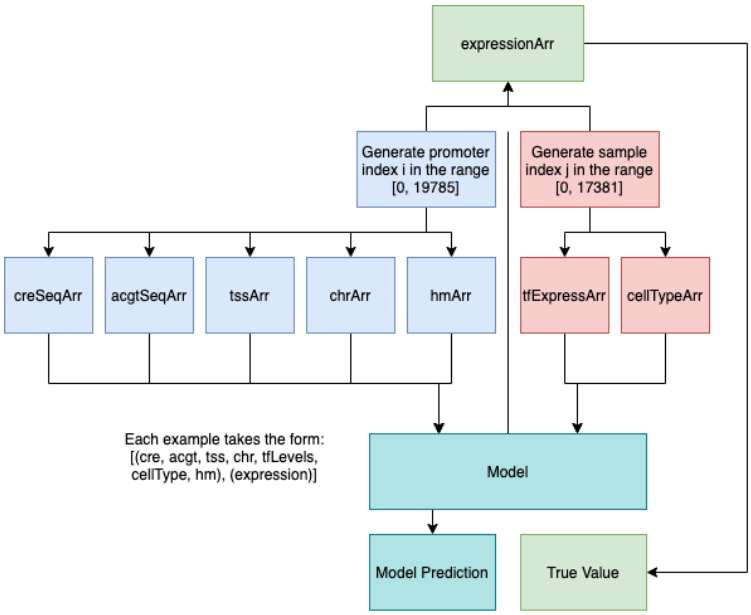

Figure 25: A flow diagram illustrating the example generation pipeline.

Standard shuffling algorithms have O(n) space complexity, so shuffling 340M examples is as intractable as storing this many examples serially. Luckily, there is a way to generate promoter and sample indices in O(1) time and space complexity in random order, without repetition. These indices, taken together, correspond to a unique training example and can be used to directly index into the numpy arrays in which the dataset is stored.

Xorshift RNGs, introduced by George Marsaglia, are a class of pseudo random number generators with periods of 2^{32}–1 (for 32-bit integers), 2^{64}–1 (for 64-bit), etc. in a random order [77]. Now $2^{32} - 1$ is just over 4 billion, and we have 340 million examples, so by using a 32-bit Xorshift RNG it is possible to generate a random integer which corresponds to a unique example about 8% of the time, while simply generating another integer if the one we generated is greater than 340 million. In other words, it takes only around 12 calls to the Xorshift RNG on average to generate indices which correspond to a unique training example randomly, without repetition. The precise algorithm I use to generate example indices is presented below.

On line 1, the xorshift RNG is defined. This function will take in two indices (the same indices presented in Figure 25) and generate two new indices randomly, without repetition, which is guaranteed by the results presented by Marsaglia [77].

While the promoter and sample indices are updated after every example is generated,

the arrays of indices for the validation and test sets do not change after being defined on lines 61, 62, 64, and 65. These four arrays specify the indices of the examples that will be used for validation and testing. On line 25, a check is made to ensure that no gene or sample in any of these arrays is used in a trainable example.

On line 18, the example generation function is defined, which is called by Tensorflow for every example generated during training or inference. The example generation function first checks the set argument, which is used to specify whether the function should return examples from the training, validation, or test set. On lines 28, 35, and 42, examples are returned using the yield keyword.

Finally, on lines 44-49, the sizes of the various dataset splits and batch size are set. Only the NUM_TRAIN and BATCH_SIZE parameters should be changed in order to keep the validation and test sets consistent.

```
1  def xorshiftNextIndex(promoterDataIdx, sampleIdx):
2
3    i = np.array(promoterDataIdx*17382 + sampleIdx, dtype=np.uint32)
4    a = np.array(13, dtype=np.uint32)
5    b = np.array(17, dtype=np.uint32)
6    c = np.array(5, dtype=np.uint32)
7
8    while True:
9      i ^= i << a
10     i ^= i >> b
11     i ^= i << c
12     if i // 17382 < 19786:
13       break
14   nextPromoterDataIdx = i // 17382
15   nextSampleIdx = i % 17382
16   return nextPromoterDataIdx, nextSampleIdx
17
18 def generateExamples(set):
19   global promoterDataIdx, sampleIdx, val_promIdxs, val_smplIdxs, test_promIdxs
       , test_smplIdxs
20   if set == 1: # Train
21     for i in range(NUM_TRAIN):
```

```
22          promoterDataIdx, sampleIdx = xorshiftNextIndex(promoterDataIdx,
        sampleIdx)
23
24          # If this promoter index or sample index is in our validation set,
        generate another.
25          while promoterDataIdx in val_promIdxs or promoterDataIdx in
        test_promIdxs or sampleIdx in val_smplIdxs or sampleIdx in test_smplIdxs:

26              promoterDataIdx, sampleIdx = xorshiftNextIndex(promoterDataIdx,
        sampleIdx)
27
28          yield ((creSeqArr[promoterDataIdx], acgtSeqArr[promoterDataIdx], chrArr[
        promoterDataIdx], tssArr[promoterDataIdx], tfExpressArr[sampleIdx],
        cellTypeArr[sampleIdx], hmArr[promoterDataIdx]), (expressionArr[
        promoterDataIdx, sampleIdx]))
29
30      elif set == 2: # Validation
31          for i in range(NUM_VAL_TRANSCRIPTS * NUM_VAL_SAMPLES):
32              promIdx = val_promIdxs[i%NUM_VAL_TRANSCRIPTS]
33              smplIdx = val_smplIdxs[i//NUM_VAL_TRANSCRIPTS]
34
35              yield ((creSeqArr[promIdx], acgtSeqArr[promIdx], chrArr[promIdx], tssArr
        [promIdx], tfExpressArr[smplIdx], cellTypeArr[smplIdx], hmArr[promIdx]), (
        expressionArr[promIdx, smplIdx]))
36
37      elif set == 3: # Test
38          for i in range(NUM_TEST_TRANSCRIPTS * NUM_TEST_SAMPLES):
39              promIdx = test_promIdxs[i%NUM_TEST_TRANSCRIPTS]
40              smplIdx = test_smplIdxs[i//NUM_TEST_TRANSCRIPTS]
41
42              yield ((creSeqArr[promIdx], acgtSeqArr[promIdx], chrArr[promIdx], tssArr
        [promIdx], tfExpressArr[smplIdx], cellTypeArr[smplIdx], hmArr[promIdx]), (
        expressionArr[promIdx, smplIdx]))
43
44  NUM_TRAIN = 1280000
45  NUM_VAL_TRANSCRIPTS = 256
46  NUM_VAL_SAMPLES = 256
```

```
47 NUM_TEST_TRANSCRIPTS = 256
48 NUM_TEST_SAMPLES = 256
49 BATCH_SIZE = 128
50
51 # Define global variables which hold dataset indices, from which we will
       generate examples. This seed can be modified to get different train data
       from the set.
52 np.random.seed(3)
53 seed = np.random.randint(0, 2**32, dtype=np.uint32)
54 promoterDataIdx, sampleIdx = xorshiftNextIndex(seed // 17382, seed % 17382)
55
56 # Generate the indices for our validation set. This seed cannot be modified
       else validation & test indices will be shuffled into the training data.
57 np.random.seed(0)
58 valAndTestProm = np.random.choice(19746, NUM_VAL_TRANSCRIPTS+
       NUM_TEST_TRANSCRIPTS, replace=False)
59 valAndTestSmpl = np.random.choice(17382, NUM_VAL_SAMPLES+NUM_TEST_SAMPLES,
       replace=False)
60
61 val_promIdxs = valAndTestProm[0:NUM_VAL_TRANSCRIPTS]
62 test_promIdxs = valAndTestProm[NUM_VAL_TRANSCRIPTS:NUM_VAL_TRANSCRIPTS+
       NUM_TEST_TRANSCRIPTS]
63
64 val_smplIdxs = valAndTestSmpl[0:NUM_VAL_SAMPLES]
65 test_smplIdxs = valAndTestSmpl[NUM_VAL_SAMPLES:NUM_VAL_SAMPLES+
       NUM_TEST_SAMPLES]
```

B Convolutional Motif Detection Test

To validate the principle that convolutional layers can effectively detect motifs as expected, I created a conjured dataset of 101,000 (100k training set, 1k validation set) completely random promoters. In 5% of these, I inserted the sequence motif "ACGGCATAGAATA" at a random location and set y_{true} to 100. In the other 95%, no sequence was inserted, and y_{true} was set to 0. With this test, I seek to answer whether one trainable convolutional layer (with exponential activation), one max pooling layer, two trainable dense layers, and an output negative binomial distribution—all optimized using a NLL loss—can learn to detect the inserted motif and output the desired expression (y_{true}). The results are presented in Figures 26 and 27.

In the top left diagram of Figure 26, I present the training loss and validation loss over 15 training epochs. In the top right diagram of Figure 26, I present the Pearson and log Pearson correlations between predicted and true expression values over 15 epochs for training and validation data. In the bottom left plot of Figure 26, I present a characteristic model prediction (the output probability density function) for an example where no motif was inserted. The red x represents the distribution mean; here, one can see that nearly the entire probability mass has converged to 0. In the bottom left of Figure 26, I present a characteristic model prediction for an example where a motif was inserted. Here one can see the probability distribution is correctly shifted out to around 100.

In Figure 27, I present a plot of predictions vs true expression values on the conjured validation set. Since the conjured data was generated with no noise, the model was able to learn the data-generating function completely and make near perfect predictions for all 1,000 validation examples. All 1,000 points on this scatter plot are located at either (0, 0) or (100, 100); thus, the plot looks quite empty. I used the median to attain a point estimate of the negative binomial distribution.

These results demonstrate that convolutions can detect motifs to any desired degree of accuracy. The only caveat with this conclusion is that convolutions will also detect short motifs that may appear by pure chance alongside those that were inserted. In the human genome, however, motifs occurring by chance may still constitute binding sites, and it could even be argued all motifs in the genome occurred 'by chance.'

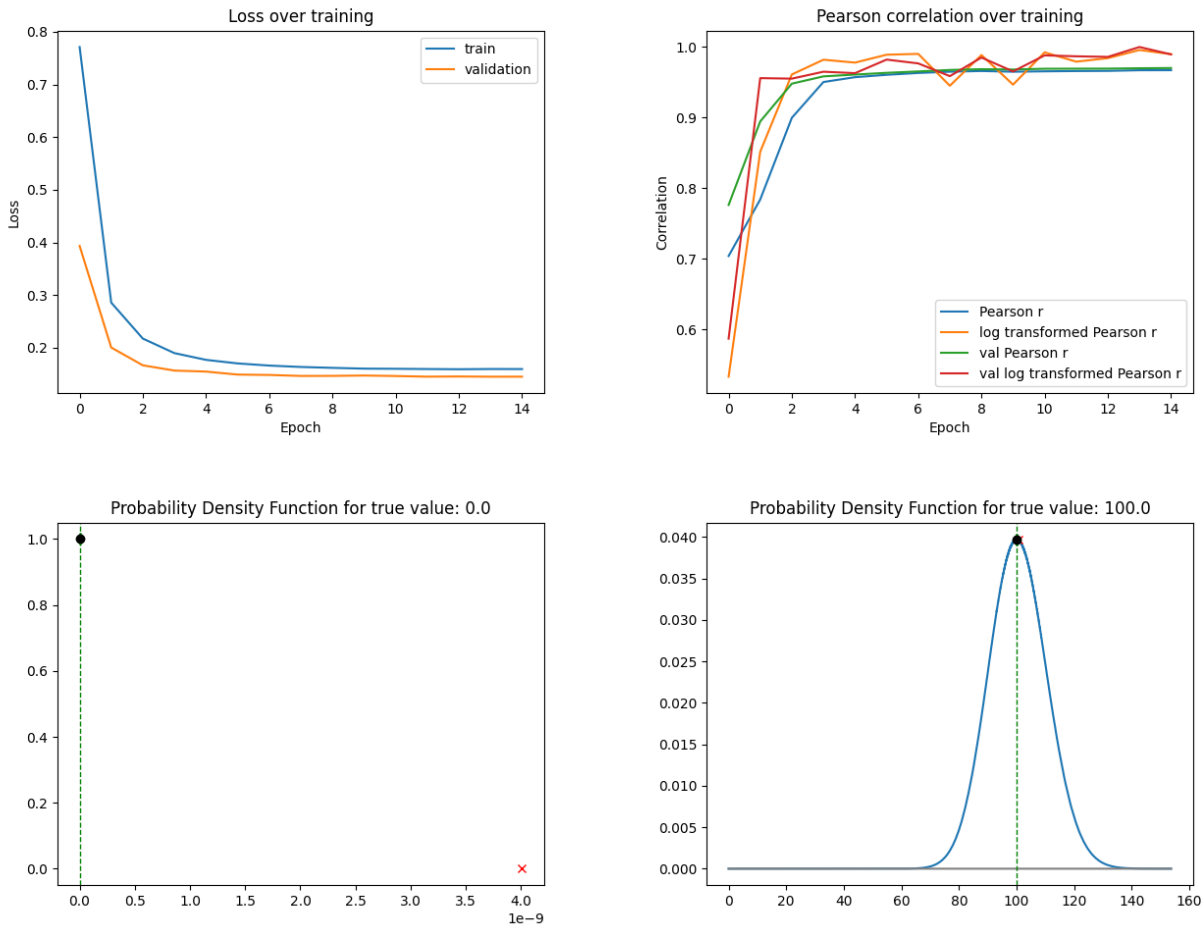

Figure 26: Four plots which showcase key results from training a toy model on a conjured dataset.

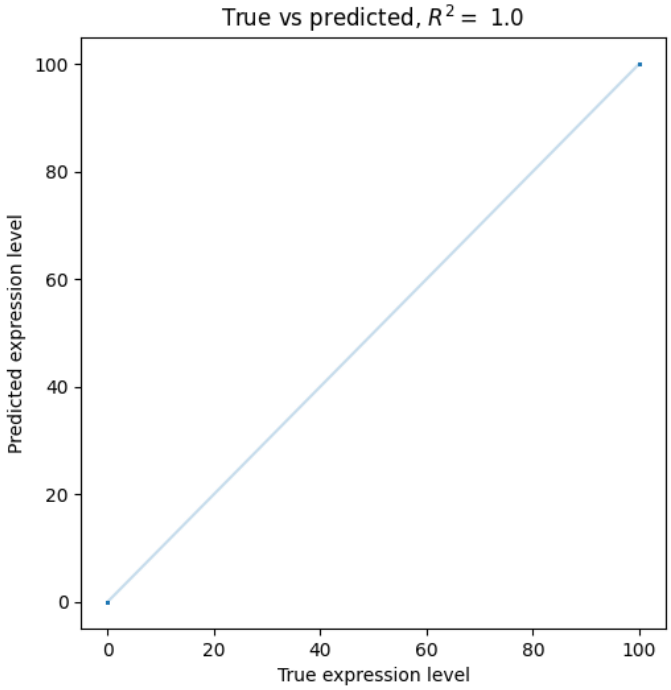

Figure 27: A plot of predictions vs true expression values on the conjured validation set.

C Architectures Tested

In this Appendix, diagrams are presented for each of the seven model architectures tested, which should be taken in contrast to Figure 24. As with Figure 24, the purpose of these diagrams is to showcase which high level components are involved, and the intricate details are not readable or important. Zoomable, pdf versions of all seven architectures can be downloaded from this link.

C.1 Seq

The Seq architecture, shown in Figure 28, takes only the sequence as input, and only scans for transcription factor motifs in this sequence.

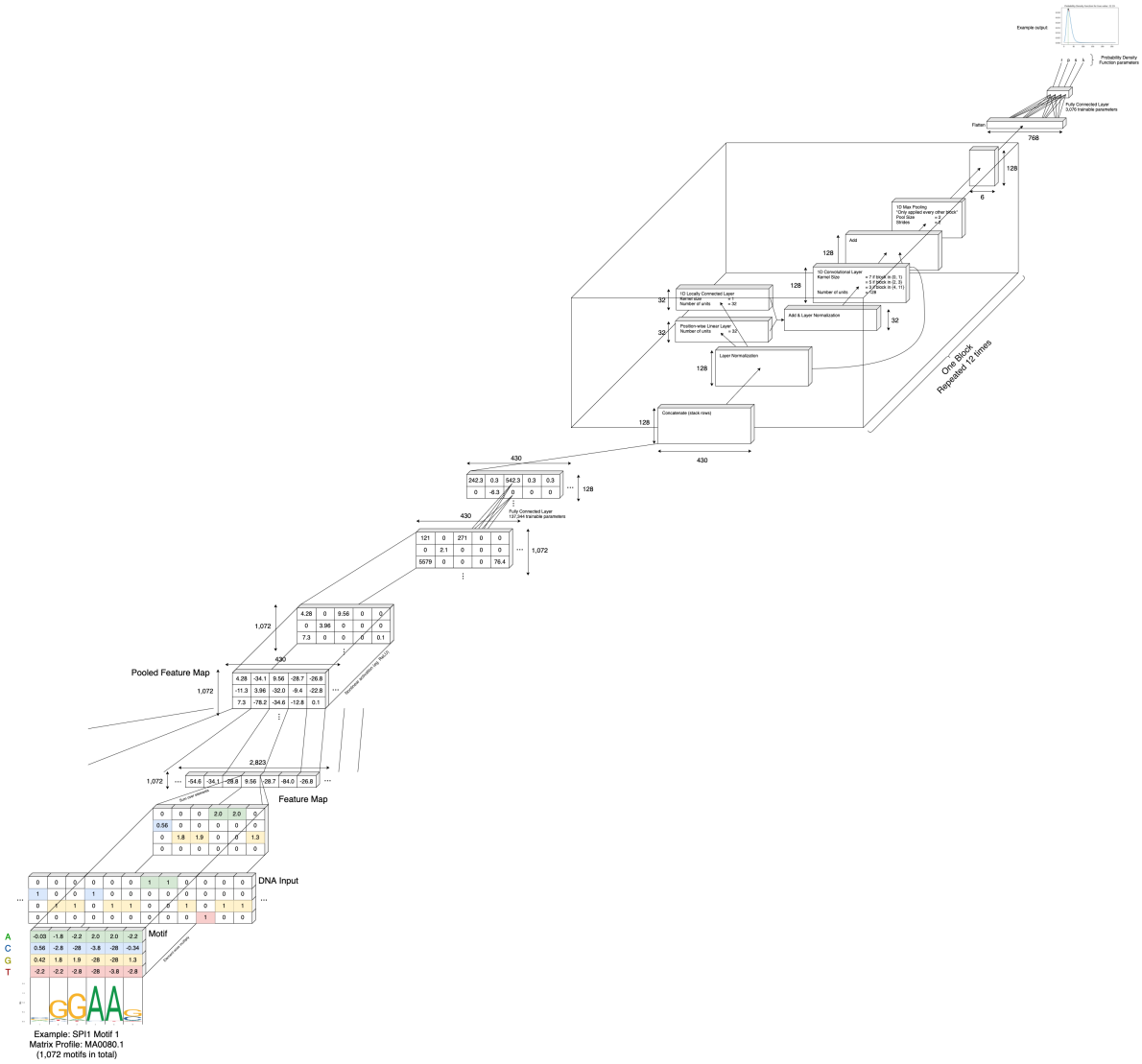

Figure 28: An illustration of the Seq architecture.

C.2 Seq + TF

The Seq + TF architecture, shown in Figure 29, takes the sequence and transcription factor levels as input, and only scans for transcription factor motifs in the sequence.

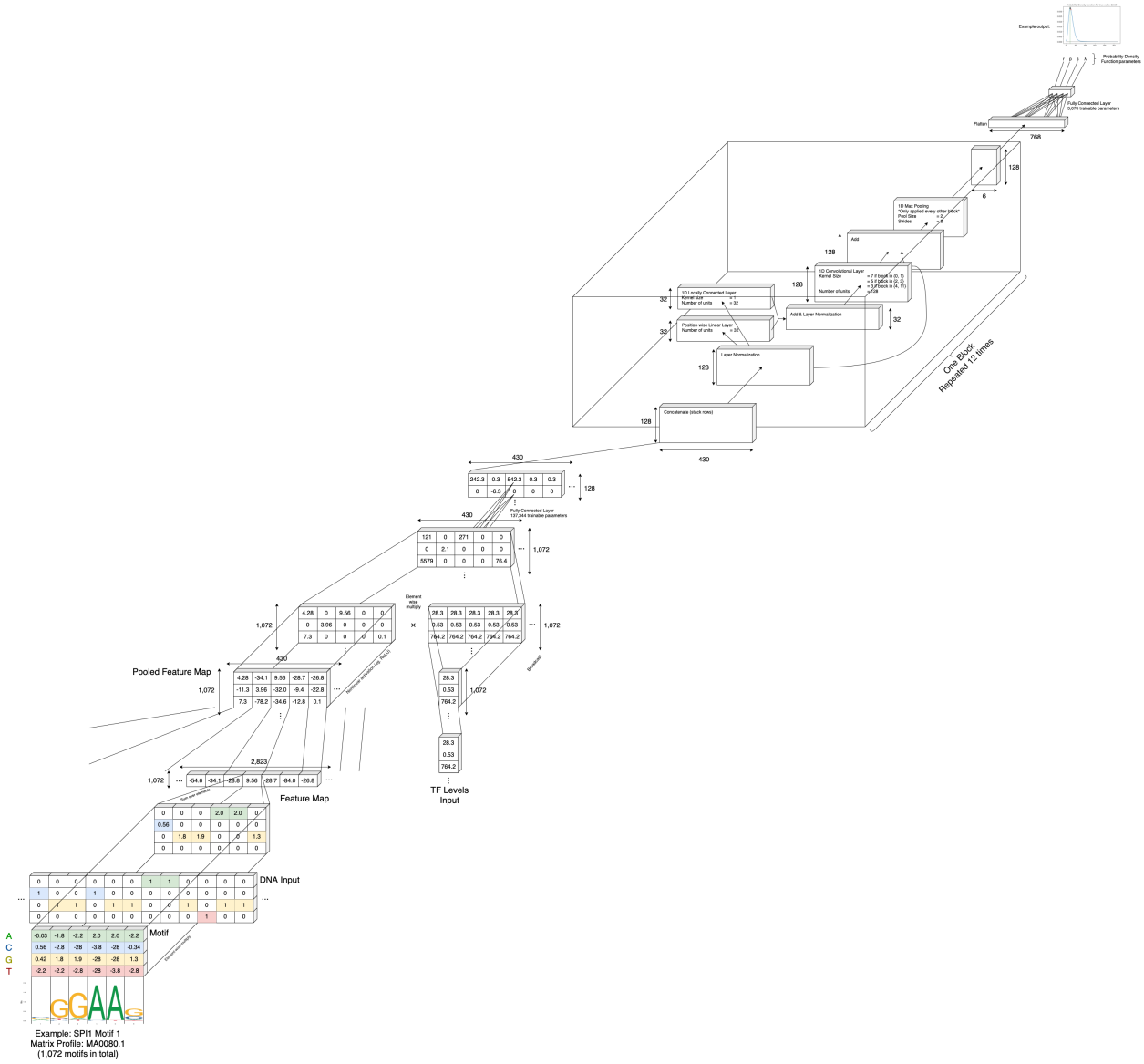

Figure 29: An illustration of the Seq + TF architecture.

C.3 Seq + TF + DF

The Seq + TF + DF architecture, shown in Figure 30, takes the sequence and transcription factor levels as input, and only scans for transcription factor motifs in the sequence. This architecture also applies a distance factor to the transcription factor levels.

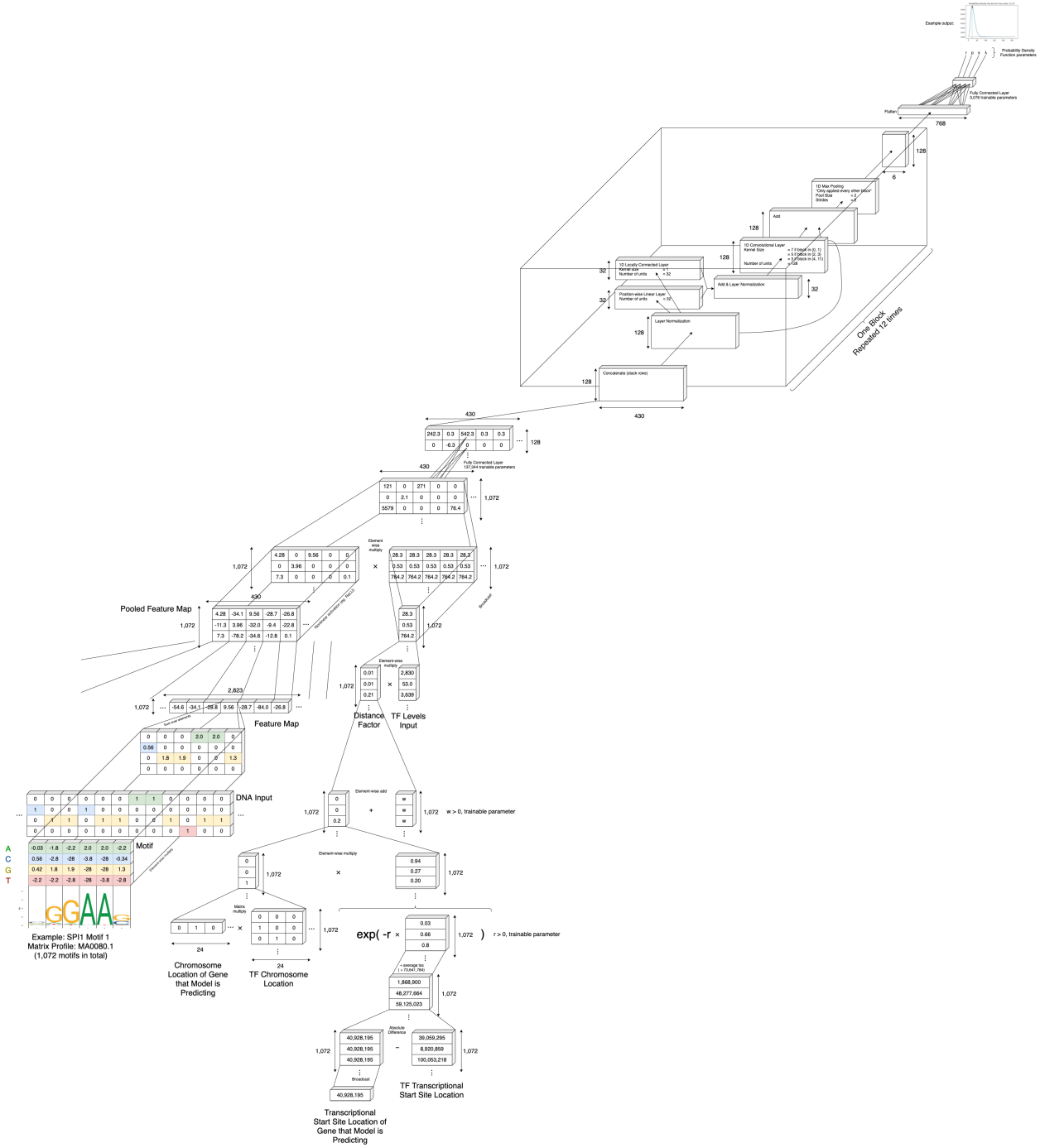

Figure 30: An illustration of the Seq + TF + DF architecture.

C.4 Seq + TF + Core

The Seq + TF + Core architecture, shown in Figure 31, takes the sequence and transcription factor levels as input, and scans for both core promoter elements and transcription factor motifs in the sequence.

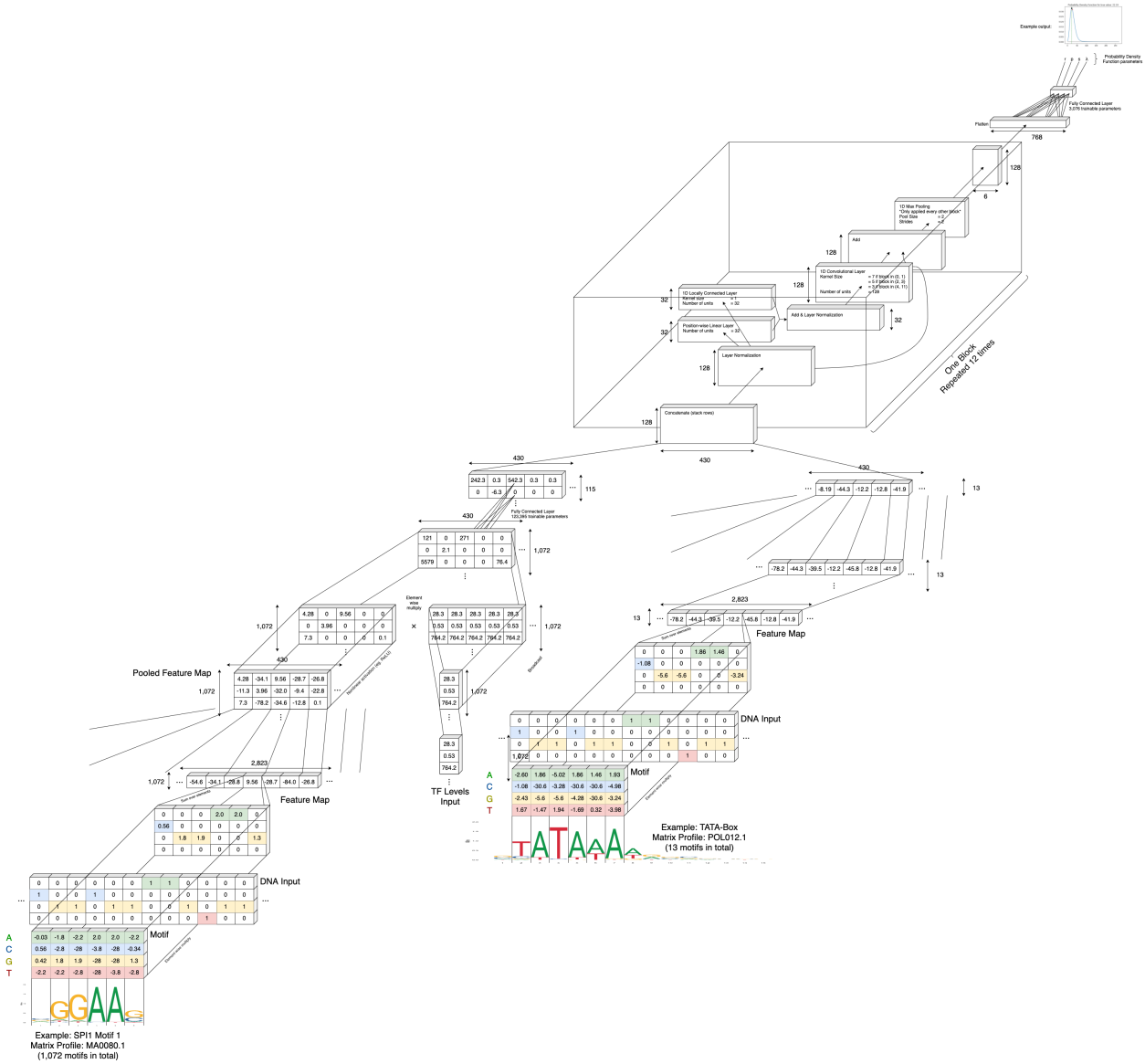

Figure 31: An illustration of the Seq + TF + Core architecture.

C.5 Seq + TF + Core + HM

The Seq + TF + Core + HM architecture, shown in Figure 32, takes the sequence, transcription factor levels, and histone modifications as input, and scans for both core promoter elements and transcription factor motifs in the sequence.

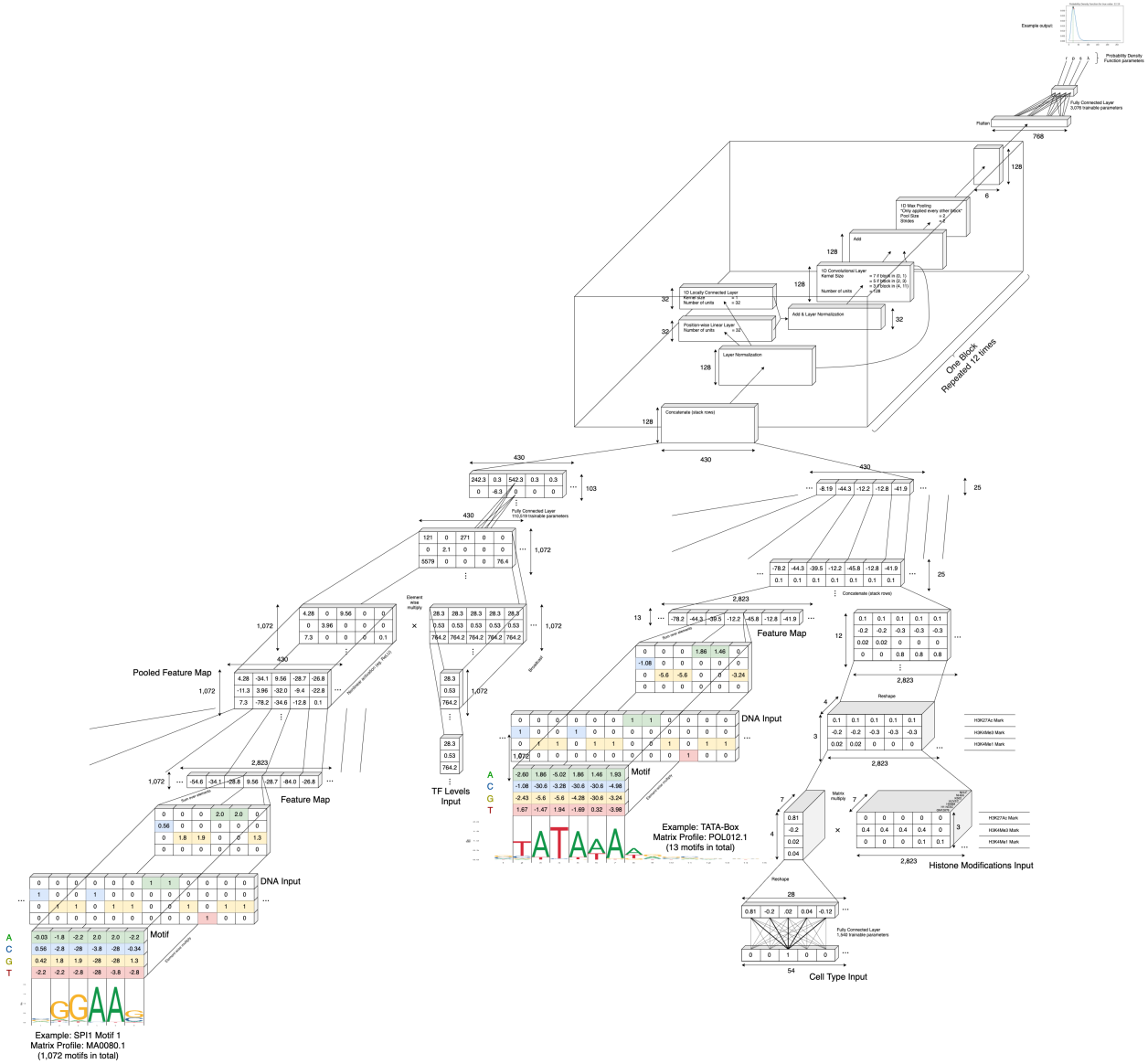

Figure 32: An illustration of the Seq + TF + Core + HM architecture.

C.6 Seq + Core + HM

The Seq + Core + HM architecture, shown in Figure 33, takes the sequence and histone modifications as input, and scans for both core promoter elements and transcription factor motifs in the sequence.

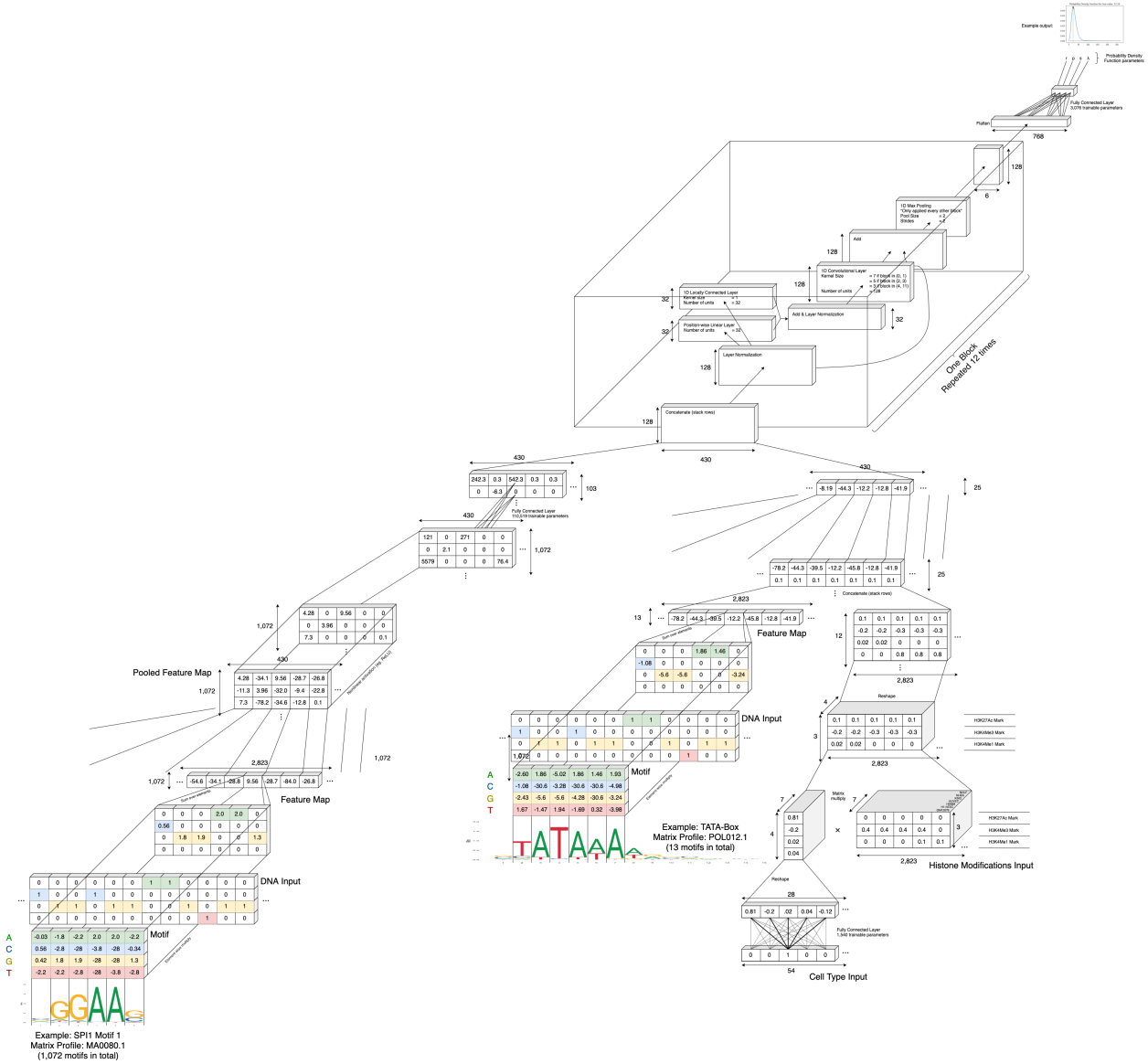

Figure 33: An illustration of the Seq + Core + HM architecture.

C.7 Seq + TF + HM

The Seq + TF + HM architecture, shown in Figure 34, takes the sequence, transcription factor levels, and histone modifications as input, and only scans for transcription factor motifs in the sequence.

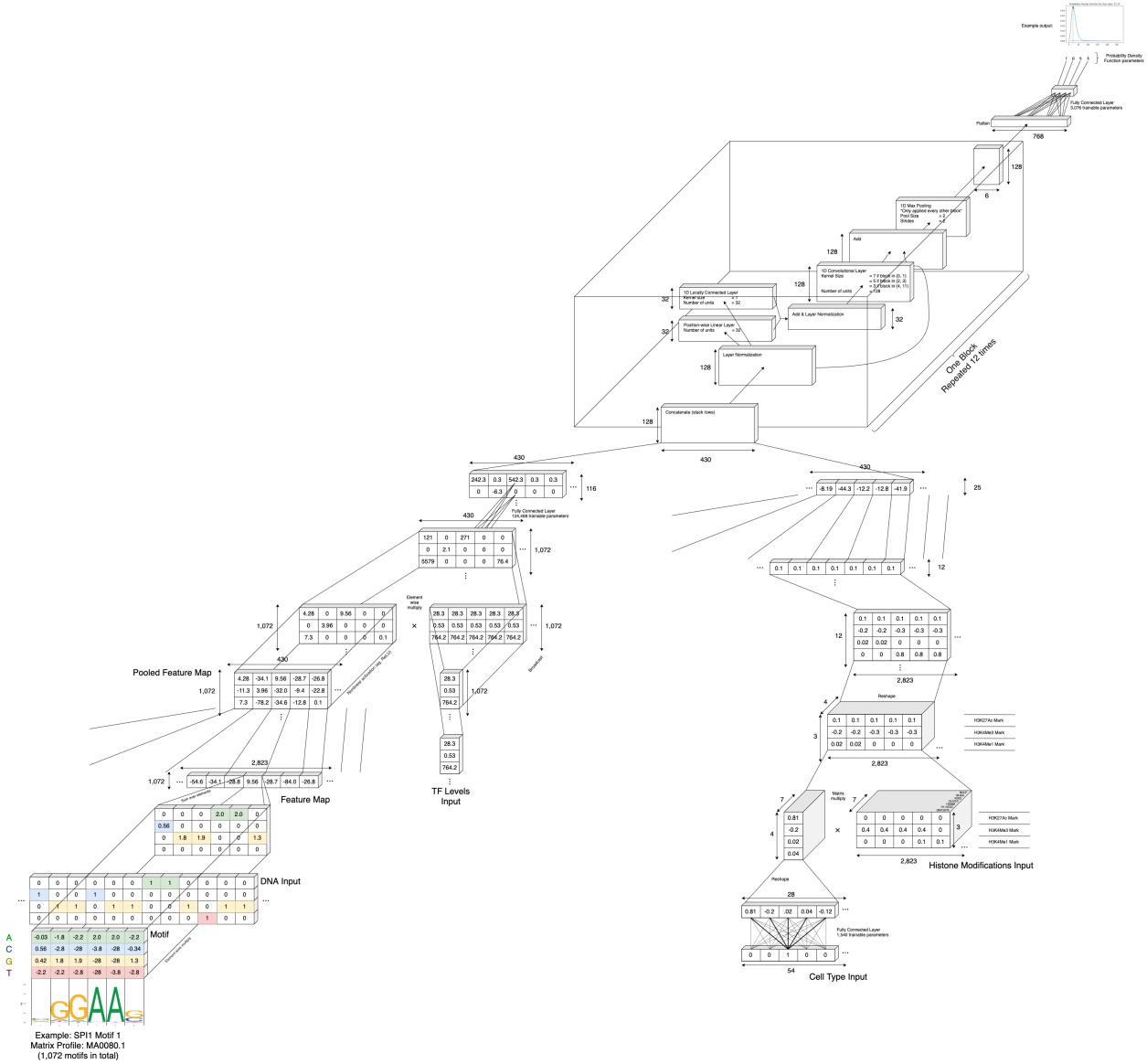

Figure 34: An illustration of the Seq + TF + HM architecture.

D Training and Plots

In this Appendix, plots are presented from training and inference for the seven model architectures defined in Appendix C. In the top left of Figures 35-41, a plot is presented of the true vs predicted expression values on the validation set. The y coordinate of each point on these scatter plots represents the median of the output distribution, and the x coordinate represents the corresponding true expression level. The values are log transformed for easier viewing. In the top right of Figures 35-41, the training and validation loss is plotted over 100,000 training steps. In the bottom left of Figures 35-41, the training and validation, pearson and log pearson correlations are plotted over 100,000 training steps. Finally, in Figures 35-41, the training and validation accuracy is plotted over 100,000 training steps.

D.1 Seq

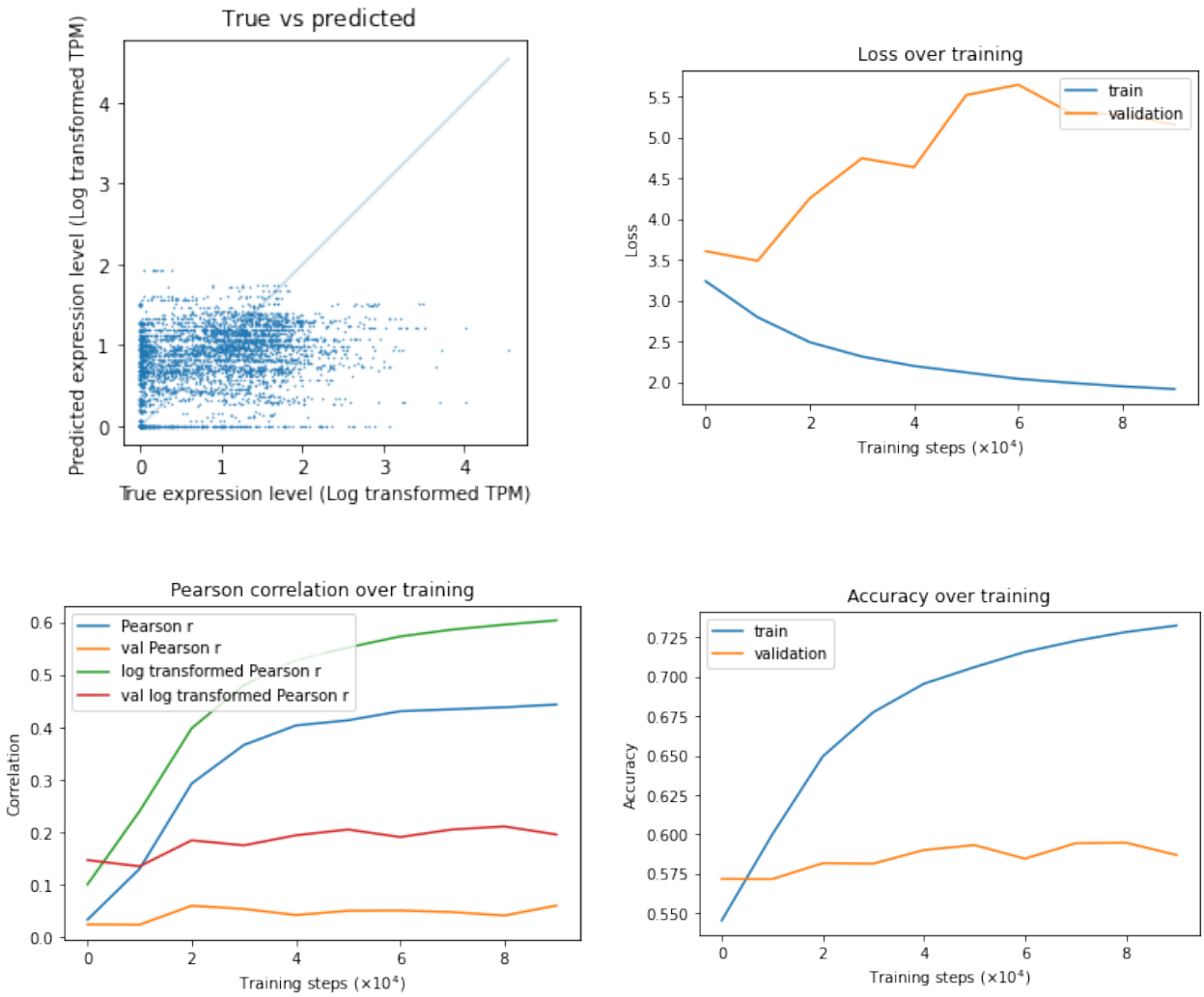

Figure 35: Training and inference plots for the Seq architecture.

D.2 Seq + TF

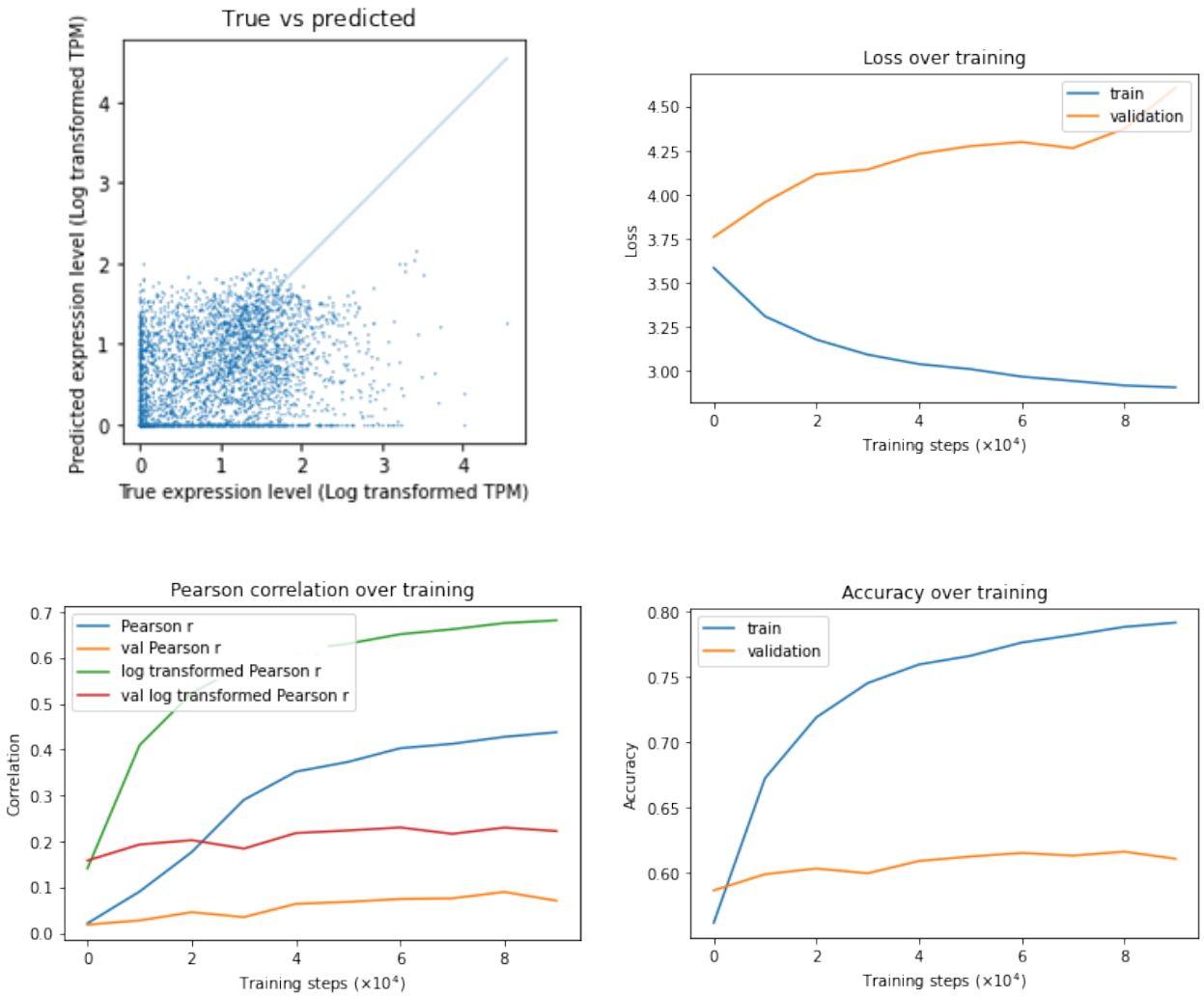

Figure 36: Training and inference plots for the Seq + TF architecture.

D.3 Seq + TF + DF

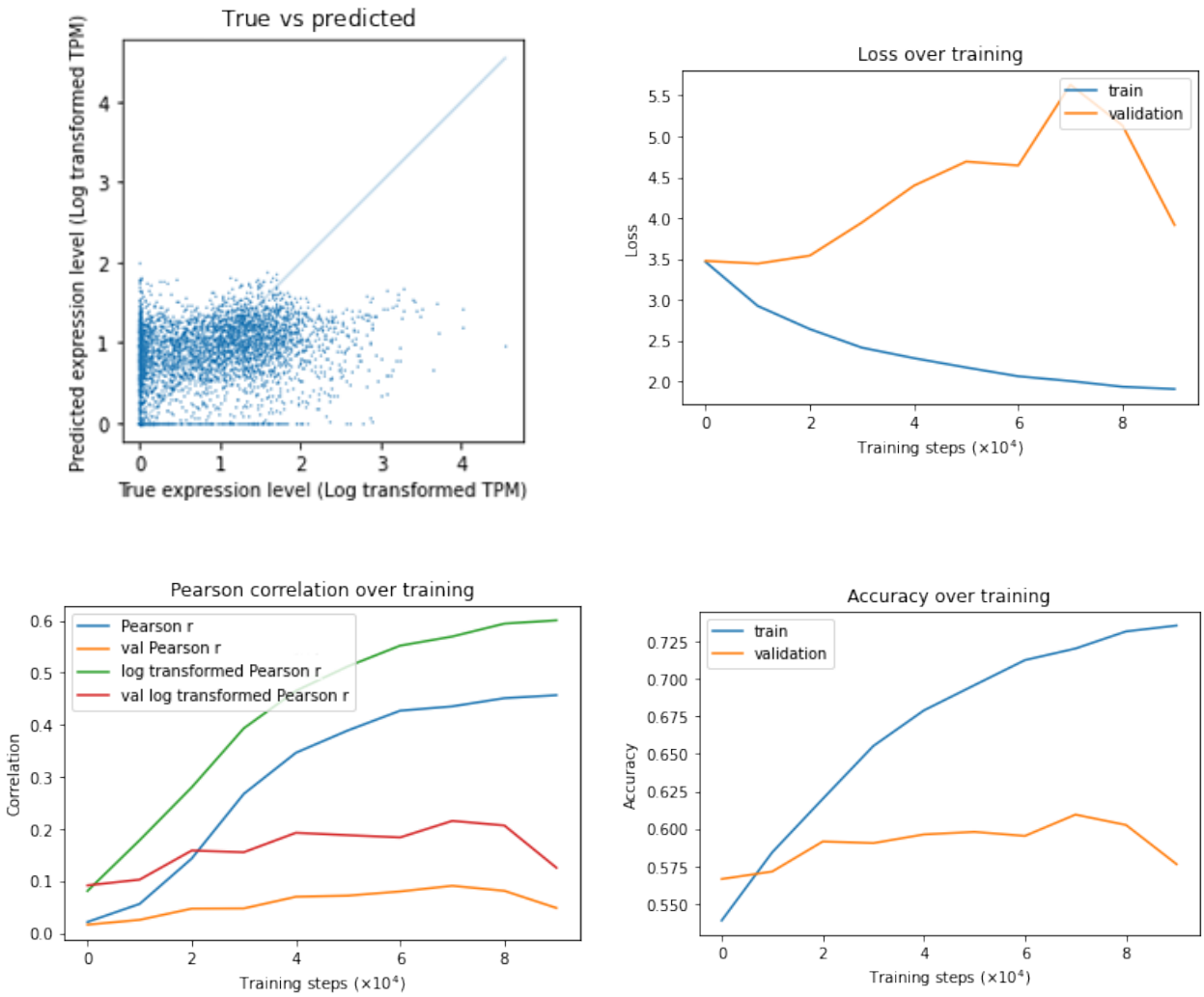

Figure 37: Training and inference plots for the Seq + TF + DF architecture.

D.4 Seq + TF + Core

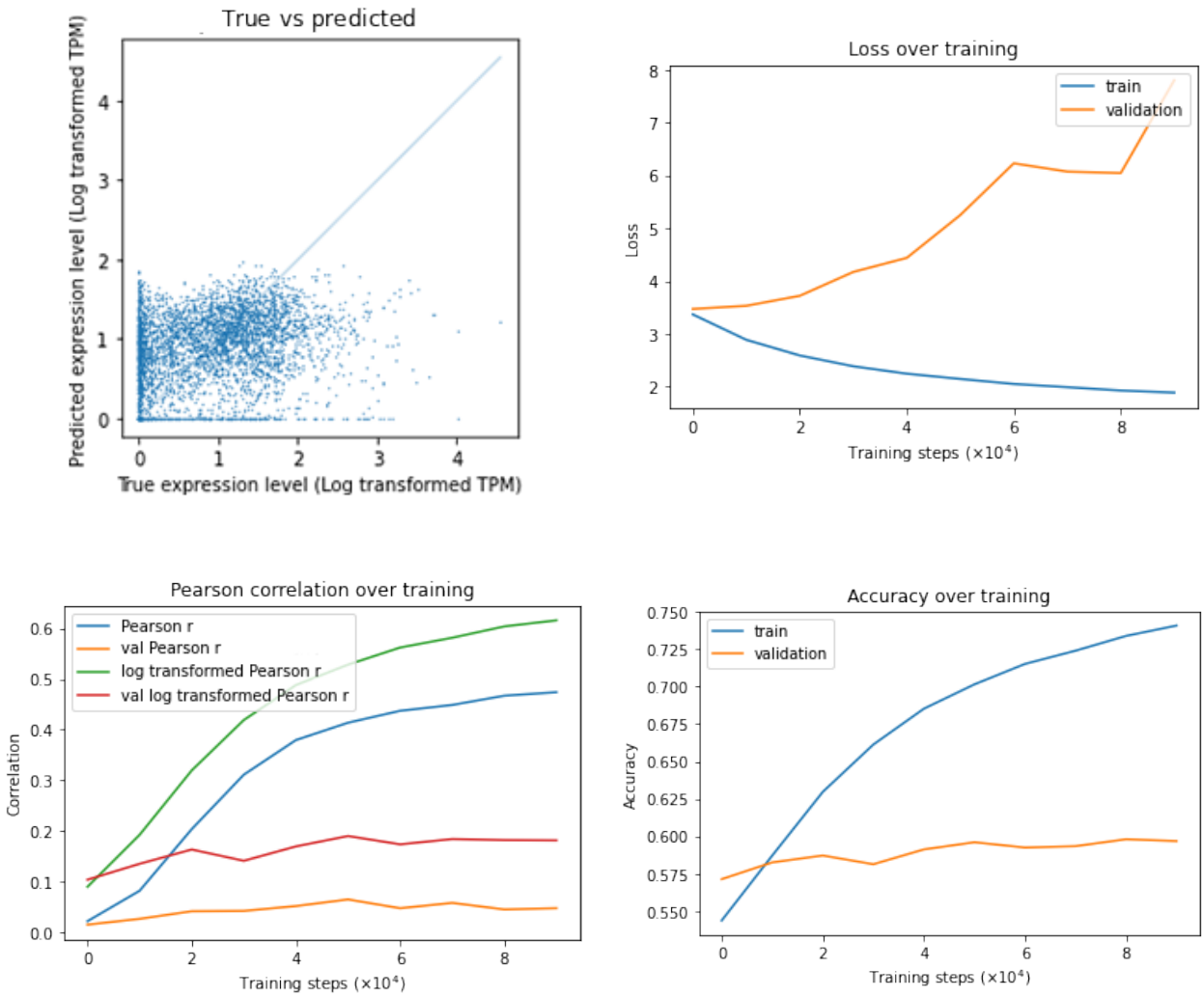

Figure 38: Training and inference plots for the Seq + TF + Core architecture.

D.5 Seq + TF + Core + HM

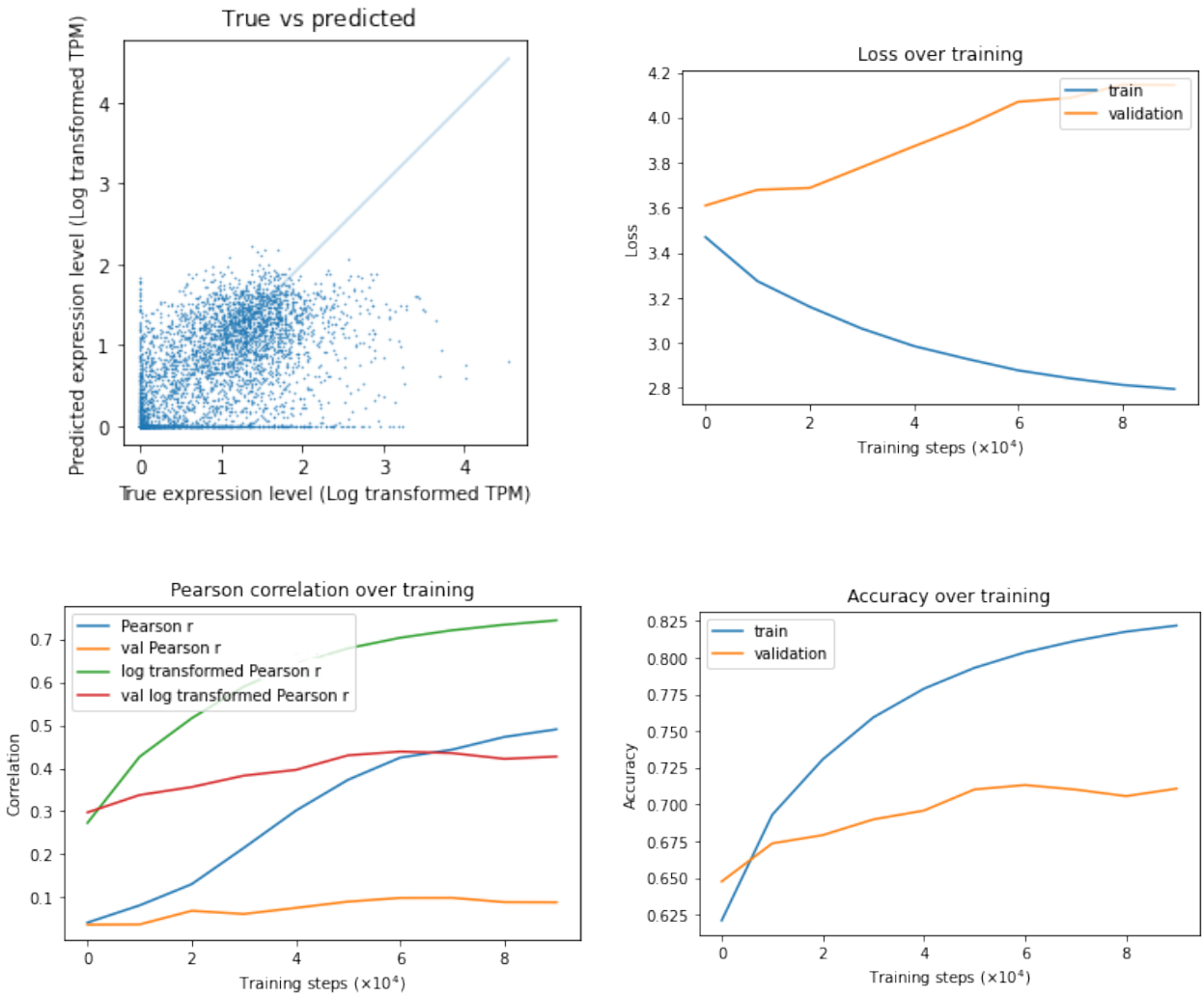

Figure 39: Training and inference plots for the Seq + TF + Core + HM architecture.

D.6 Seq + Core + HM

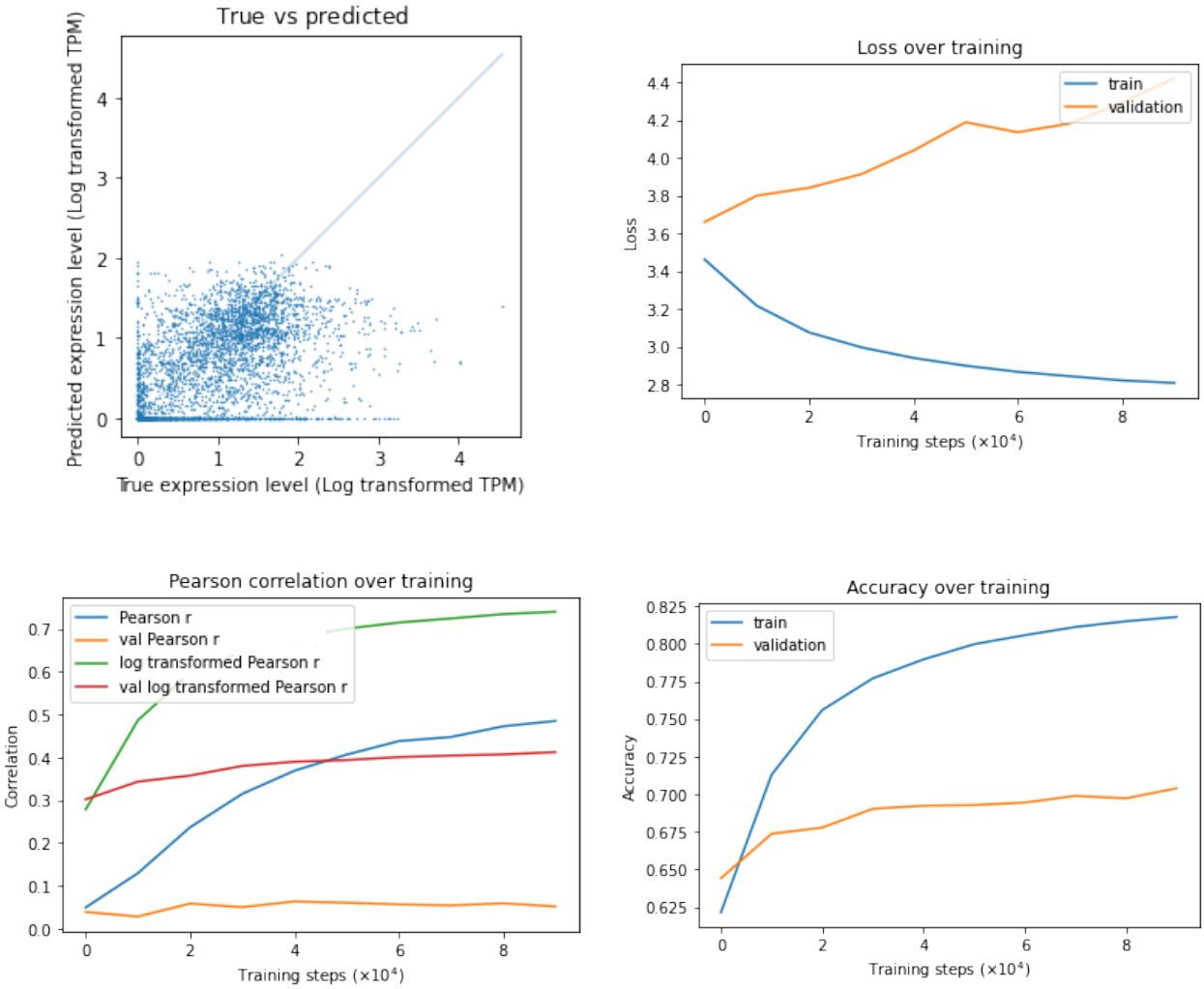

Figure 40: Training and inference plots for the Seq + Core + HM architecture.

D.7 Seq + TF + HM

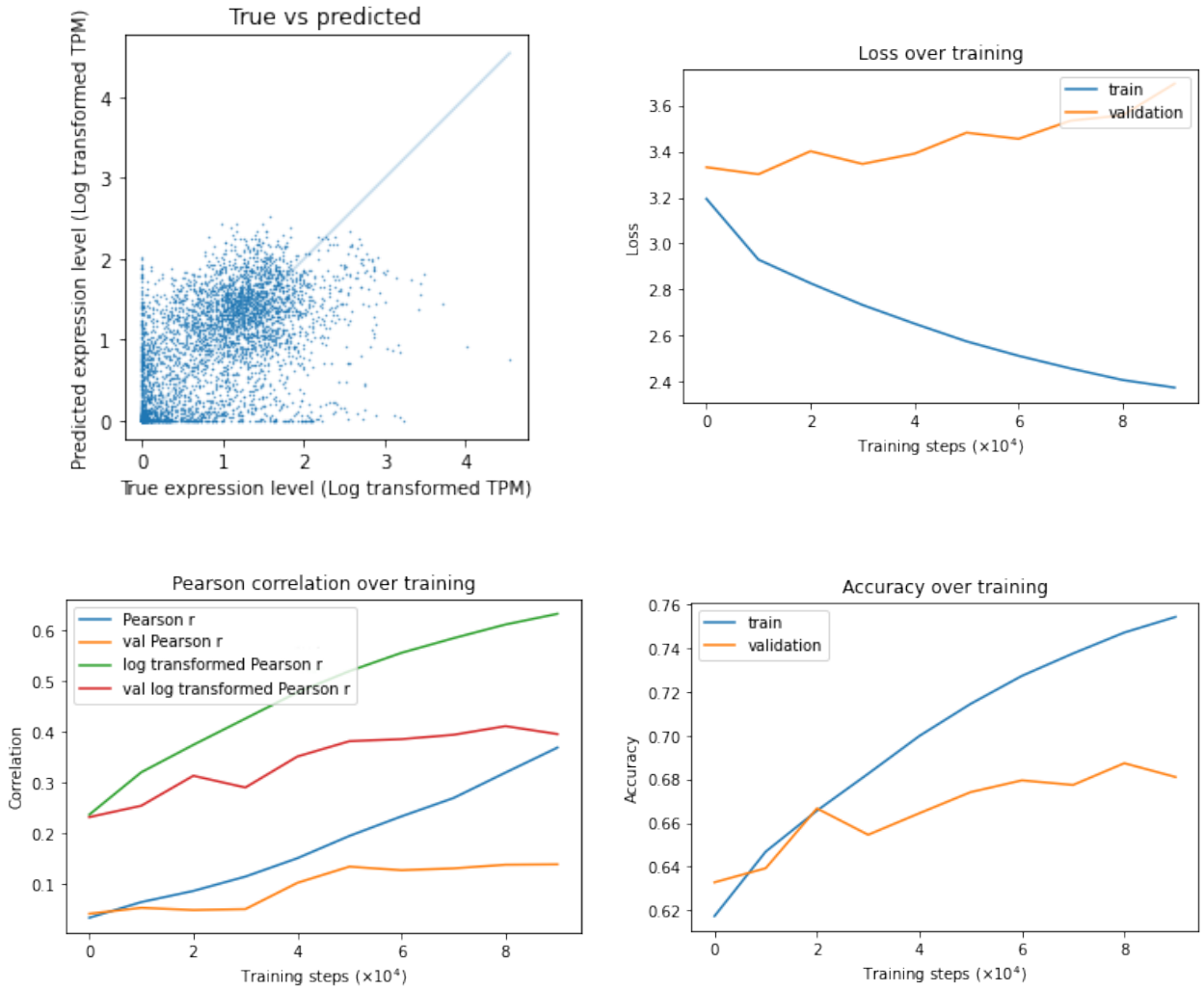

Figure 41: Training and inference plots for the Seq + TF + HM architecture.

E Data Preprocessing Pipeline

This section outlines the data processing pipeline used to generate the numpy arrays from which examples are generated in Appendix A. The following figures— 42, 43, and 44—should be examined alongside the python files made available in section 11, if one wishes to understand how to generate the dataset from scratch.

E.1 Gene-Related Data

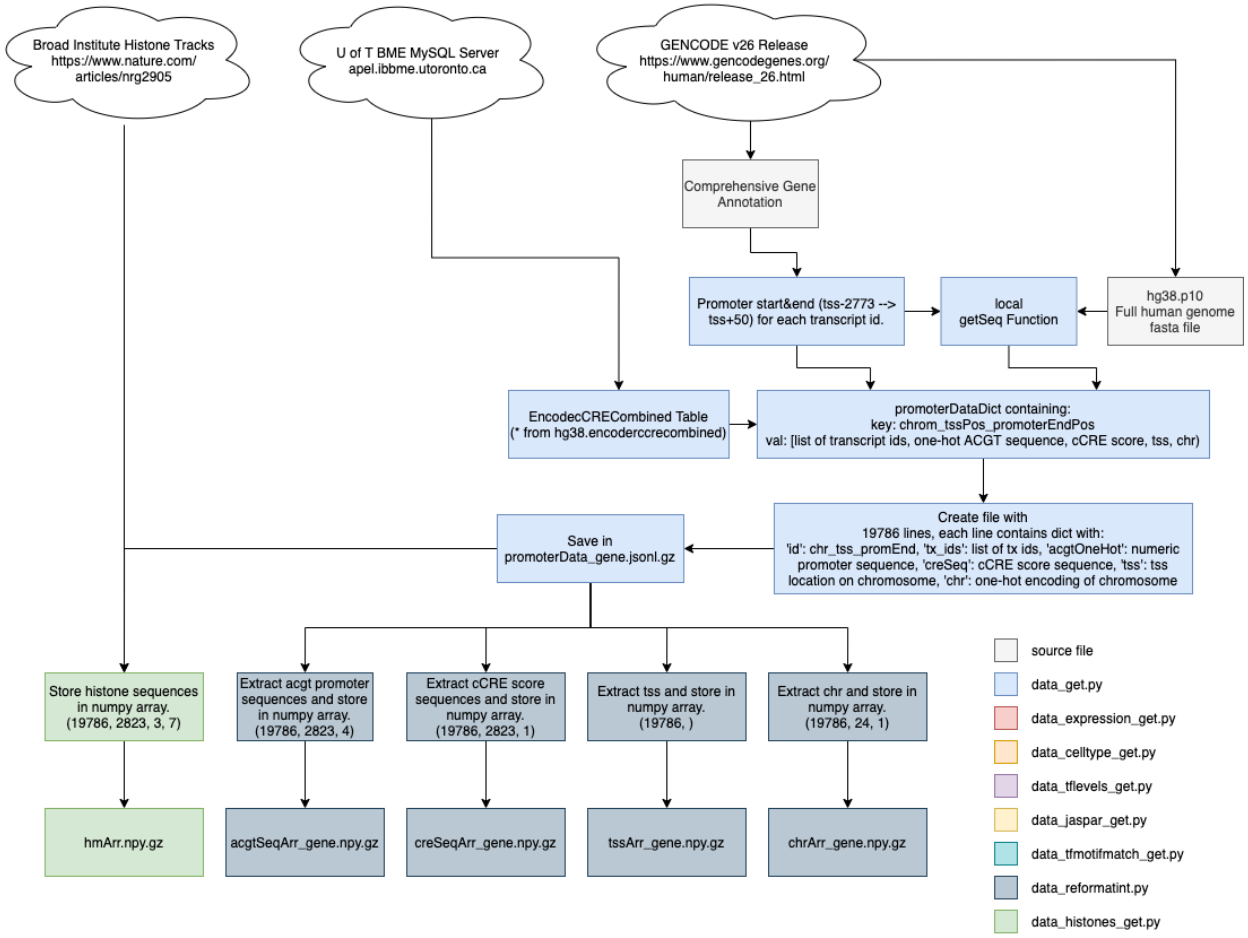

Figure 42: The pipeline to gather and process data from source to final array for gene-related data.

E.2 Sample-Related Data

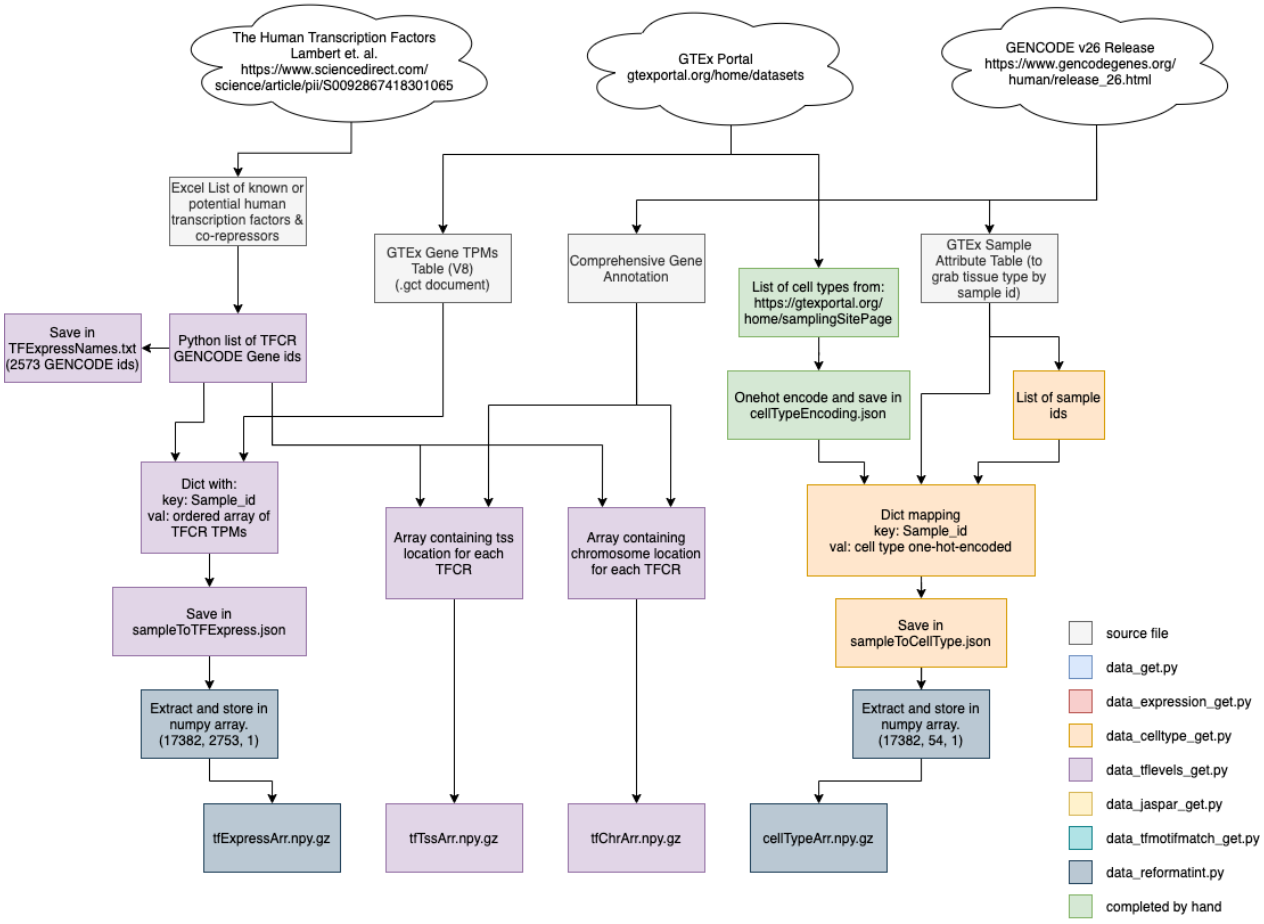

Figure 43: The pipeline to gather and process data from source to final array for all sample-related data.

E.3 JASPAR and Expression data

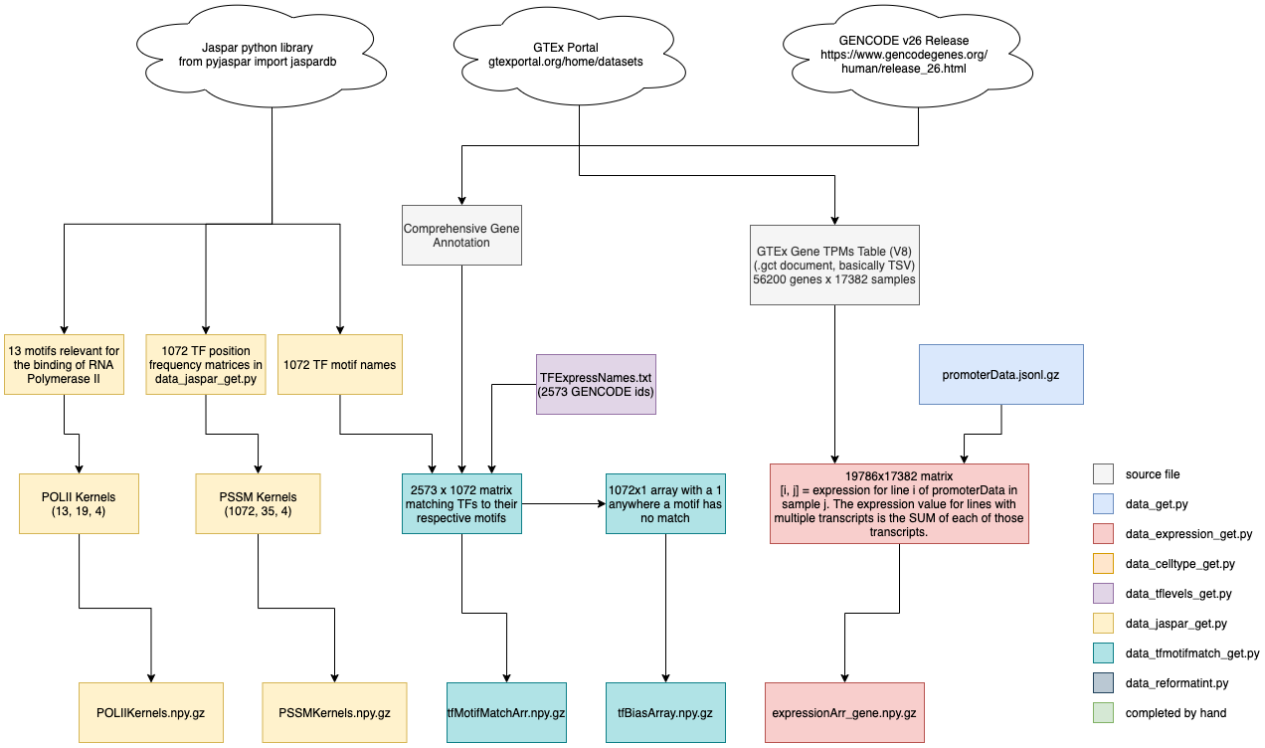

Figure 44: The pipeline to gather and process data from source to final array for expression data and JASPAR binding motifs.

This page is intentionally left blank.