Freie Universität Berlin

Department of Mathematics and Computer Science
Institute of Computer Science

Development and Evaluation of Transformer-Based DNA-Language Models with disease-conditioning

Master Thesis
Submitted for the degree of Master of Science in computational Sciences

by

Phani Shankar Bharadwaj Gandlur

Berlin, April 22,2025

1st examiner Prof. Dr. Roland Eils
2nd examiner Prof. Dr. Volker John

Declaration of Authorship

I hereby declare to Freie Universität Berlin that I have completed the following thesis on Development and Evaluation of Transformer-Based DNA-Language Model with disease-conditioning without the use of sources and aids other than those cited.

I declare that the present work is free of plagiarism. Any statements that have been taken from other writings, whether directly or indirectly, have been clearly marked as such.

Further, I declare that this work has not been submitted to any other university as part of an examination attempt, either in identical or similar form, nor has it been published elsewhere.

In line with good scientific practice, I declare that AI-assisted tools (e.g., ChatGPT by OpenAI) were used to support the writing process for language refinement and formatting. All scientific content, analysis, and conclusions are my own.

April 22, 2025

Date

Signature

ACKNOWLEDGEMENT

Completing this thesis while undergoing treatment for a chronic illness has been a challenging yet rewarding journey. I extend my heartfelt gratitude and thanks to those who supported me through this process, both academically and personally. Their unwavering support made this work possible.

I am deeply grateful to my supervisor Benjamin Wild, for giving me the opportunity to be part of this project. His enthusiasm, encouragement, and constant support have been invaluable. With his guidance, I was able to learn and grow significantly while making my contribution to this field of study.

I would like to thank Prof. Dr. Roland Eils for his guidance throughout this thesis and for serving as my first examiner.

I would like to thank Prof. Dr. Volker John for serving as my second examiner and for kindly agreeing to my request. I sincerely appreciate his time, support, and the valuable feedback provided during the evaluation process.

I would like to thank Julius Upmeier zu Belzen and Paul Kittner for their support and helpful insights during the implementation of the PRS framework used in this thesis.

I am truly thankful to my friends at Freie Universität Berlin for the countless moments of joy and camaraderie we shared. I am deeply grateful to my colleagues at Oviva, their encouragement and support helped lighten my journey.

Finally, my heartfelt thanks go to my family, who have been my greatest source of strength. Their unwavering belief in me and constant encouragement kept me going, even through the toughest moments.

Heartfelt thanks!

ABSTRACT

Understanding genetic variations and their relationship to phenotypic traits is a key challenge in computational genomics. In this work, I developed a Transformer-based DNA language model trained on phenotype-conditioned genomic sequences. The model was optimized through extensive hyperparameter tuning to learn meaningful representations of genetic variations. Training data was generated by extracting reference genome sequences, applying individual-specific genetic variants, and appending corresponding phenotype tokens. This approach ensures that the model captures the associations between genetic mutations and phenotypic traits, enabling more accurate predictions in downstream inference tasks.

At inference time, reference sequences were again retrieved and modified to reflect sample-specific allelic variations. These were passed through the model, which predicted the next nucleotide conditioned on both upstream sequence context and phenotype embedding. The output logits represented a probability distribution over the nucleotide vocabulary, providing interpretable scores for both reference and alternate alleles at each variant site. Comparing these predictions to the actual alleles enabled an assessment of how well the model captured the contextual likelihood of genetic variation.

To benchmark the model's outputs, variant effect sizes were obtained from two polygenic risk score (PGS) models for Type 1 Diabetes from the PGS Catalog. For each variant, the model-derived logit corresponding to the effect allele was extracted and compared to reported effect sizes. Filtering based on effect size thresholds helped focus the analysis on informative variants, enabling an external assessment of biological relevance.

To evaluate downstream utility, the model-inferred scores were integrated into the PRS framework. Logit differences between alternate and reference alleles were formatted as annotations and used as priors in Bayesian SNP effect estimation. The pipeline was run for both the conditioned and non-conditioned models, allowing a direct comparison of predictive performance was assessed specifically in the European ancestry group.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

1) **AI** – Artificial Intelligence
2) **ANN** – Artificial Neural Network
3) **AUROC** – Area Under the Receiver Operating Characteristic Curve
4) **CPU** – Central Processing Unit
5) **DNA** – Deoxyribonucleic Acid
6) **FFN** – Feedforward Neural Network
7) **FDR** – False Discovery Rate
8) **GENCODE** – Genome Annotation Consortium
9) **GPU** – Graphics Processing Unit
10) **GWAS** – Genome-Wide Association Study
11) **GTEx** – Genotype-Tissue Expression
12) **LD** – Linkage Disequilibrium
13) **MAF** – Minor Allele Frequency
14) **NLP** – Natural Language Processing
15) **NLL** – Negative Log-Likelihood
16) **NTP** – Nucleotide Triphosphate
17) **PGS** – Polygenic Risk Score
18) **PRS** – Polygenic Risk Score
19) **QKV** – Query-Key-Value (Attention Mechanism)
20) **RNN** – Recurrent Neural Network
21) **SNP** – Single Nucleotide Polymorphism

22) **SBayesRC** – Summary-based Bayesian Regression with functional priors (Conditioned)

23) **TF** – Transcription Factor

24) **VCF** – Variant Call Format

25) **VIF** – Variance Inflation Factor

26) **WGS** – Whole Genome Sequencing

27) **PRS** - Polygenic Risk score

INTRODUCTION

The intersection of machine learning and genomics is reshaping how we understand genetic variations and their contribution to human disease. For decades, genome-wide association studies (GWAS) (Karczewski et al., 2018) and polygenic risk scores (PRS) have served as the primary tools to estimate genetic liability for complex traits. These methods typically rely on summary statistics and assume additive, linear effects of many variants across the genome. While effective in many settings, traditional statistical models often struggle to incorporate biological context and fail to capture the complex, non-linear dependencies within genomic data.

The emergence of large-scale DNA language models offers a new lens through which to interpret genomic variation. Inspired by the success of transformer architectures in natural language processing, researchers have begun applying similar models to DNA sequences, treating the genome as a language, where nucleotides are tokens, mutations are syntax changes, and phenotypes are expressions of biological meaning. These models are typically benchmarked on tasks such as next-token prediction, regulatory element identification, and variant effect prediction. However, existing benchmarks often focus on narrow genomic windows or limited variant sets, overlooking the full complexity of genome-scale variation.

In this work, an alternative and orthogonal benchmarking approach is proposed: using DNA language models to inform polygenic risk scoring. Polygenic risk scores aggregate the effects of potentially millions of variants across the entire genome, making them a natural testbed for evaluating whether language models can extract meaningful and generalizable signals from DNA. If model-derived embeddings or logits improve PRS accuracy, they can be interpreted as capturing functional or disease-relevant information, without relying on curated annotations or hand-crafted features..

Specifically, outputs from a phenotype-conditioned DNA language model are used to guide the SBayesRC framework, a Bayesian extension of traditional PRS methods that incorporates variant-level priors. While SBayesRC was originally developed to use human-created functional annotations (e.g., indicating whether a variant lies in a promoter or enhancer), DNA language models are proposed as an alternative source of annotation through their learned representations. These embeddings are available for all variants, including those lacking traditional annotations, and may capture subtle sequence-level patterns that are missed by categorical labels.

Importantly, this approach places no hard limit on sequence context length. Because PRS models aggregate effects genome-wide, the impact of long-range context such as distal regulatory elements can be implicitly evaluated by observing whether models trained on longer sequences lead to better risk predictions. In this way, polygenic risk scoring becomes both a downstream application and an indirect evaluation metric for the quality of DNA language models.

This thesis presents the development and evaluation of such a transformer-based DNA language model, trained on whole-genome sequence data and conditioned on phenotype labels. By linking model-derived predictions to polygenic risk estimation through SBayesRC, the work aims to establish a scalable and data-driven pathway for integrating deep learning into genomic risk prediction.

1.1 Motivation

Complex diseases such as Type 1 Diabetes are not caused by single mutations, but rather by the combined effect of thousands of variants distributed across the genome. Polygenic risk scores (PRS) are widely used to quantify this cumulative risk, assigning weights to variants based on their statistical association with disease. However, these scores are often derived in a purely data-driven way, with little incorporation of prior biological knowledge. This limits their interpretability and can lead to reduced performance, especially when transferring between populations.

One way to improve PRS is by integrating information about a variant's likely functional importance. Traditionally, this is done through annotations labels that identify genomic features like enhancers or transcription factor binding sites. But these annotations are incomplete and do not exist for all variants. In contrast, DNA language models can generate continuous embeddings or logits for any genomic sequence, capturing statistical and functional properties learned from large-scale sequence data. These representations can be used as informative priors in models like SBayesRC, which adjust variant weights based on both GWAS summary statistics and external evidence.

This work builds on the idea that DNA language models are more than predictive tools, they can also serve as genome-wide annotation engines. By benchmarking their outputs in a PRS framework, we can evaluate whether the models have learned representations that generalize to complex disease risk.

1.2 Objective

The primary objective of this study is to develop a transformer-based DNA language model that can learn biologically meaningful representations of genomic sequences, while incorporating phenotypic context. The model is trained to predict the next nucleotide in a sequence, using either raw DNA sequence alone or DNA plus a phenotype token (e.g., indicating diabetes status). The underlying assumption is that disease-relevant patterns are reflected in local sequence context, and that conditioning on phenotype allows the model to learn how sequence variation correlates with observed traits.

After training, the model is used to compute probabilistic outputs (logits or likelihoods) at specific variant positions. These predictions are then integrated into the SBayesRC framework to generate a polygenic risk score. This pipeline allows for a principled comparison between traditional annotation-driven approaches and those based on learned sequence representations. The final goal is to assess whether conditioning a language model on phenotype improves its ability to support downstream genomic risk prediction.

1.3 Scope

This work focuses on developing and evaluating a phenotype-conditioned transformer model for genomic sequence modeling, with applications in variant effect prediction and polygenic risk scoring. The model is trained on GTEx whole-genome sequencing (WGS) data, with diabetes-related phenotypes used as labels. The inference pipeline is applied to variant data structured similarly to that of the UK Biobank, enabling genome-wide prediction of allele probabilities.

The study explores two training regimes with and without phenotype conditioning and compares their outputs in terms of predictive performance. The polygenic risk scores are computed using SBayesRC, with DNA model-derived logits or embeddings acting as variant-level priors. While the broader evaluation of PRS methods is beyond the scope of this thesis, the study positions DNA language models as an alternative source of functional information and a tool for scaling PRS beyond manually curated annotations.

FUNDAMENTALS

This chapter provides the foundational concepts necessary to understand the methodologies used in this work. It begins with an overview of Transformer architectures and their adaptation for genomic sequence modeling, followed by an introduction to key genetic concepts such as DNA sequences, SNPs, and allelic variations. The chapter then explores polygenic risk scores (PGS), their significance, and the limitations of traditional approaches. Finally, it delves into **SBayesRC**, a Bayesian method that refines SNP selection using functional annotations, highlighting its advantages over other PGS models.

2.1 Transformers for Genomic Sequences

2.1.1 Transformers architecture

First introduced in the 2017 paper "*Attention Is All You Need"* by Vaswani et al., transformers were designed to improve language translation by modeling long-range dependencies more effectively than recurrent networks *(Vaswani et al., 2017)*. At their core, transformers leverage self-attention mechanisms to capture relationships between distant elements in a sequence, allowing them to process information in parallel rather than sequentially. This breakthrough has led to widespread adoption across various fields, from natural language processing and image generation to genomics and drug discovery *(Lin et al., 2023).*

Today, transformers power real-time speech translation, detect complex patterns in genomic sequences *(Ji et al., 2021*), enhance fraud detection and optimize search engines. Their adaptability has earned them the title of foundation models, as they serve as the backbone of modern AI applications across multiple domains.

Unlike traditional models like Recurrent Neural Networks (RNNs) and Long Short-Term Memory networks (LSTMs), which process sequences step by step *(Vennerød, Kjærran, & Bugge, 2021)*, transformers handle entire sequences in parallel. This eliminates the bottleneck of sequential processing, making training significantly faster and more scalable. Additionally, transformers overcome the limitation of long-range dependencies that affect RNNs by using **self-attention**, a mechanism that allows each token to weigh the importance of every other token in the sequence. Another key innovation is **positional encoding**, which assigns a unique numerical representation to each token's position, preserving order without relying on recurrence. These advancements enable transformers to capture complex relationships in sequences more efficiently, making them ideal for applications beyond language, including genomics and biological sequence modeling.

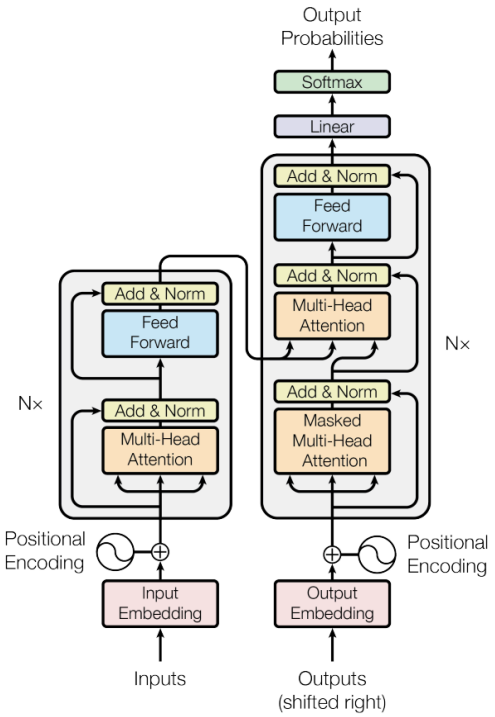

Figure 2.1.1 (a): The Transformer - model architecture

The Transformer model revolutionized how we process sequential data by replacing traditional recurrent networks with a fully attention-based mechanism. Instead of processing input step by step, it takes in the entire sequence at once, allowing for **parallel computation** and capturing long-range dependencies efficiently. The process starts with **input embeddings**, where each token (or nucleotide, in our case) is converted into a numerical representation that captures its meaning. Since transformers don't inherently understand order, **positional encodings** are added to maintain the sequence structure. The real magic happens in **multi-head self-attention** (Figure 2.1.1(a)), where each token in the sequence considers every other token before deciding how much weight to assign to it. This is done using **query (Q), key (K), and value (V) matrices**, which help the model determine relationships between different parts of the sequence. The outputs from multiple attention heads are then processed through **feedforward layers**, with **residual connections and layer normalization** ensuring smooth training and stable gradient flow (Figure 2.1.1(b)). Stacked transformer layers further refine these representations, allowing the model to learn complex patterns. After training, the model can predict patterns in unseen sequences, making it a powerful tool for tasks like genomic sequence modeling, mutation effect prediction, and variant classification.

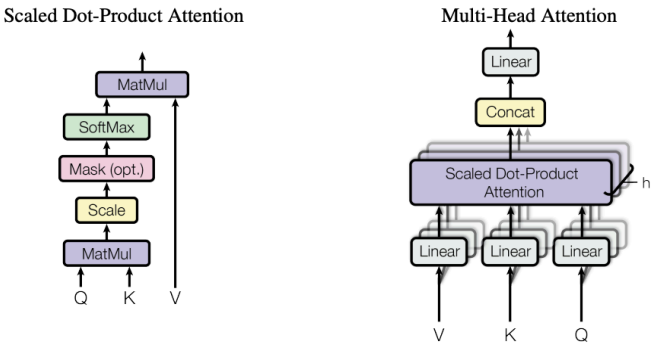

Figure 2.1.1 (b): (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

The self-attention mechanism computes attention scores using the dot product of queries and keys, scaled by the square root of the key dimension. These scores are passed through a softmax function and multiplied by the value vectors to produce context-aware representations

$$Attention(Q, K, V) = softmax(QK^T / \sqrt{d})V$$

This describes how queries Q, keys K, and values V are used to compute attention weights. d is the dimension of the key vectors and acts as a scaling factor.

2.1.2 Transformers for sequence data

Transformers were originally designed for natural language processing, but their ability to handle long range dependencies makes them a powerful tool for analyzing any kind of sequential data, including DNA sequences *(Ji et al., 2021)*. Unlike traditional models like recurrent neural networks (RNNs), which process sequences step by step, transformers take in the entire sequence at once. This allows them to recognize patterns and relationships across long stretches of data more efficiently.

In genomics, this capability is especially valuable since genetic variations can occur across thousands or even millions of nucleotides. Some mutations have effects that depend on distant regions of DNA *(Avsec et al., 2021)*, making it essential for models to capture these long-range dependencies. With self-attention, transformers assign different levels of importance to each nucleotide relative to others in the sequence. This helps in tasks like predicting mutation effects and identifying functional regions in the genome.

Another major advantage of transformers is their ability to process data in parallel, making them much faster than older sequence-based models. Given the vast amounts of genomic data available, this efficiency is crucial for large-scale studies *(Sudlow et al., 2015)*. In the next

sections, we explore how transformers are being adapted for genomic research and the breakthroughs they enable.

2.1.3 Self-attention to capture genomic patterns

The self-attention mechanism, a key component of Transformer architectures, has significantly impacted genomic research by enabling models to capture long-range dependencies in DNA sequences. Unlike traditional models that rely on fixed-length windows or sequential processing, self-attention dynamically weighs the importance of different positions within a sequence. This allows for a deeper understanding of complex genomic patterns, such as regulatory interactions and functional motifs.

Several studies have explored the application of self-attention in genomics. *The Nucleotide Transformer (Dalla-Torre et al., 2024)* demonstrated how large-scale pretraining on genomic sequences enhances predictive accuracy, even for rare variants. Similarly, *(Chen et al., 2021)* leveraged self-attention to model interactions between regulatory elements, uncovering cooperative effects that traditional models often miss. These works illustrate how self-attention allows models to go beyond simple pattern recognition, capturing the intricate architecture of the genome and improving predictions for gene expression and variant effects.

2.2 DNA and Genetic Variations

This section explores the fundamental aspects of DNA and its variations, focusing on how genetic differences shape biological traits and disease susceptibility. We begin with an overview of DNA sequences and their role in encoding genetic information, followed by a discussion on Single Nucleotide Polymorphisms (SNPs) and allelic variations, which are the most common forms of genetic diversity. We then distinguish between reference genomes and individual genomes, highlighting their importance in genetic studies. Together, these concepts lay the foundation for understanding how genetic variations contribute to complex traits and their implications for disease risk prediction.

2.2.1 DNA Sequences and Genetic Information

DNA, the fundamental blueprint of life, encodes the genetic instructions that govern biological processes. It consists of sequences of four nucleotide bases Adenine (A), Thymine (T), Cytosine (C), and Guanine (G) arranged in long, double-stranded molecules *(Watson & Crick, 1953)*. These sequences form genes, which carry the instructions for building proteins and regulating cellular functions. However, DNA is not a static entity, it varies between individuals due to genetic mutations and inherited differences.

One of the most common forms of genetic variation occurs at the single nucleotide level, where a single base in the DNA sequence is altered *(Brookes, 1999)*. These variations, known as Single Nucleotide Polymorphisms (SNPs), are the most abundant genetic differences between individuals and can influence traits, disease susceptibility, and drug response *(Collins, Brooks, & Chakravarti, 1998)*. While some variations are benign, others can have significant biological consequences, altering protein function or gene regulation.

Beyond SNPs, genetic variations also include insertions, deletions, and structural rearrangements *(Feuk, Carson, & Scherer, 2006)*. These differences contribute to the unique genetic makeup of each individual. To study these variations, researchers often compare an individual's genome to a reference genome, a standardized sequence representing a consensus of a population. The interplay of these genetic differences with environmental factors plays a crucial role in understanding complex traits, including those associated with diseases like Type 1 Diabetes *(Redondo et al., 2020).*

2.2.2 Single Nucleotide Polymorphisms (SNPs) and Allelic variations

Single Nucleotide Polymorphisms (SNPs) are the most common type of genetic variation, occurring when a single nucleotide in the genome is altered at a specific position. While the vast majority of SNPs have no functional effect, some can influence how genes are expressed or how proteins function, making them important in understanding genetic predisposition to diseases. These variations can act as biological markers, helping researchers trace inheritance patterns and identify regions of the genome associated with particular traits *(Brookes, 1999)(Bush & Moore, 2012)*.

Each SNP can have different versions, known as alleles, which individuals inherit from their parents. For example, at a given SNP position, one person might have an "A" nucleotide while another has a "G." Since humans are diploid organisms, meaning we inherit two copies of each chromosome, one from each parent, we can have two identical alleles (homozygous) or two different ones (heterozygous). The presence of specific alleles at certain SNP positions has been linked to disease susceptibility, drug response, and other phenotypic traits.

Understanding SNPs is essential for genome-wide association studies (GWAS) and polygenic risk score (PGS) calculations. By analyzing SNP patterns across populations, researchers can identify correlations between genetic variants and disease risks. However, these variations do not exist in isolation, they interact with surrounding genomic sequences and regulatory elements. This interplay between individual genetic variations and the broader genomic context is crucial,

which leads us to the next section: the distinction between reference genomes and individual genomes *(Visscher, Brown, McCarthy, & Yang, 2012).*

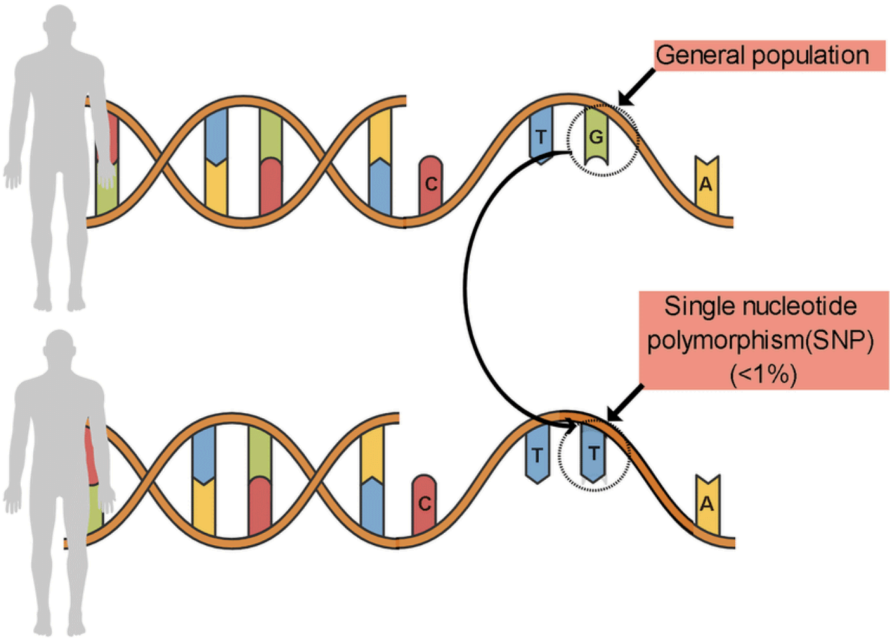

Figure 2.2.2: Single-nucleotide polymorphism (SNP): a genetic variation that generates a new allele, characterized as a polymorphism when > 1% of the population has this gene format *(Lima, Galiciolli, Pereira, Felisbino, Machado-Souza, de Oliveira, & Guiloski, 2022)*

2.2.3 Reference Genome vs Individual Genome

A reference genome serves as a standardized genetic blueprint, representing a composite sequence derived from multiple individuals. It provides a baseline for researchers to compare individual genomes, identify variations, and study their potential impact. However, no single person has a genome identical to the reference (1000 Genomes Project Consortium, 2015). Instead, every individual carries unique genetic variations, including SNPs, insertions, deletions, and structural changes. These differences shape everything from physical traits to disease susceptibility *(Church et al., 2011)*.

An individual genome, on the other hand, reflects the complete set of genetic instructions specific to a person, inherited from both parents. Unlike the reference genome, which is a consensus sequence, an individual's genome contains personalized variations that contribute to their unique genetic makeup. This distinction is critical when studying genetic associations, as population-level reference genomes cannot fully capture the diversity and complexity of individual variations.

For tasks like polygenic risk scoring and disease prediction, using only the reference genome would be insufficient. Instead, we need to analyze individual genomic data, identifying which specific alleles a person carries at key positions. By comparing an individual's genome to the reference, we can detect variations that may have functional consequences, such as increased

disease risk or altered drug metabolism. This highlights the importance of personalized genomic analysis in precision medicine and genetic research.

2.3 Polygenic Risk Scores (PGS)

Genetic traits, especially complex diseases like diabetes or heart disease, are not influenced by a single gene but rather by the cumulative effect of multiple genetic variants. Polygenic Risk Scores (PGS) are a statistical way to estimate an individual's genetic predisposition to a condition by aggregating risk information from numerous Single Nucleotide Polymorphisms (SNPs) across the genome. In this section, we will explore why PGS is important, the limitations of traditional methods, and compare various existing approaches to calculating polygenic risk.

2.3.1 PGS and Its Importance

Predicting disease risk based on genetics has traditionally been a challenge because most common diseases are polygenic, meaning they result from the combined effect of many small genetic variations, not just a single mutation. Polygenic Risk Scores help bridge this gap by quantifying an individual's genetic risk based on genome-wide SNP data *(Torkamani, Wineinger, & Topol, 2018).*

PGS is becoming particularly important in personalized medicine, where it can help identify high-risk individuals early, enabling preventative healthcare strategies. For example, someone with a high polygenic risk score for Type 1 Diabetes (T1D) might benefit from early screening, lifestyle adjustments, or potential interventions before symptoms even appear *(Lewis & Vassos, 2020).*

Beyond clinical applications, PGS is also used in genomic research, helping scientists uncover how different genetic variants contribute to disease susceptibility. With improvements in genetic databases and statistical models, PGS is becoming an increasingly powerful tool for genetic risk prediction across diverse populations *(Choi & O'Reilly, 2019).*

2.3.2 Traditional PGS methods and Limitations

Polygenic Risk Score (PGS) estimation involves selecting which single nucleotide polymorphisms (SNPs) to include and determining how to weight each one. This task is inherently complex, as not all SNPs contribute equally to a trait, and naive summation fails to capture the intricacies of genetic architecture. Over time, various statistical models have been developed to estimate PGS more effectively. Each method comes with its own strengths and limitations, particularly in how they handle linkage disequilibrium (LD), incorporate biological priors, and generalize across populations.

LDpred2, an extension of the original LDpred method (Privé, Arbel, & Vilhjálmsson, 2020), improves PGS estimation by adjusting SNP effect sizes based on LD, a phenomenon where nearby SNPs tend to be inherited together. This adjustment enhances the accuracy of risk prediction when SNPs are correlated. However, LDpred2 relies heavily on accurate LD reference panels, and any errors in LD estimation can propagate into miscalculated risk scores. Moreover, LD structures vary across populations, making LDpred2 less effective in cross-ancestry applications.

LDpred-funct builds upon LDpred2 by integrating functional annotations, such as whether a SNP resides within a protein-coding gene or regulatory region (Marquez-Luna et al., 2021). This biologically-informed model allows SNPs in functionally important regions to receive greater weight, thereby enhancing predictive power. Nonetheless, LDpred-funct employs a stepwise modeling approach: it first estimates per-SNP heritability based on annotations and then uses these estimates to adjust effect sizes. This separation of steps introduces potential biases and may limit overall prediction accuracy.

MegaPRS offers an alternative by incorporating multiple population-specific datasets into a unified model. Unlike methods that treat all populations uniformly, MegaPRS accounts for ancestry-specific genetic effects, which is crucial for accurate cross-population predictions (Zhou et al., 2022). This makes it a powerful tool for multi-ancestry applications. However, the method is data-intensive, requiring large, well-balanced datasets from diverse populations. In scenarios where such data are lacking, MegaPRS may yield unstable or inconsistent estimates.

2.4 SBayesRC & Bayesian SNP Selection

As polygenic risk scores (PGS) continue to shape genetic risk prediction, the challenge remains, how do we improve accuracy while accounting for the complexity of genetic variation? Traditional methods either lack functional annotation integration (e.g., SBayesR) or rely on stepwise approaches that introduce bias (e.g., LDpred-funct). This is where SBayesRC comes in, refining the way we estimate SNP effects by leveraging Bayesian inference and functional annotations in a unified framework.*(Lloyd-Jones et al., 2019), (Marquez-Luna et al., 2021).*

An extension of SBayesR, SBayesRC improves polygenic prediction through three key innovations. First, it performs joint analysis over all common SNPs across the genome, avoiding reliance on predefined SNP subsets. This comprehensive modeling ensures that no potentially informative variant is excluded from consideration. Second, it incorporates functional genomic annotations directly into the model, which helps to differentiate causal variants from background variation. By allowing annotations to inform both the inclusion probability and effect size prior for each SNP, the model can prioritize functionally relevant loci. Third, SBayesRC adopts a hierarchical Bayesian approach that dynamically refines SNP effect size estimates, enhancing robustness to noise and improving generalizability across populations.

The method takes GWAS summary statistics and LD correlations from a reference sample as input and outputs posterior inclusion probabilities (PIP) for each SNP, essentially ranking how likely a SNP is to be truly associated with the trait. By allowing functional annotations to influence both SNP inclusion probability and effect size distributions, SBayesRC refines genetic risk predictions more effectively than previous models *(Marenne et al., 2022).*

A major innovation in SBayesRC is its low-rank approximation for LD modeling. Instead of treating each SNP independently or assuming perfect LD, it collapses redundant information from SNPs in high LD regions, boosting computational efficiency and reducing noise. This allows SBayesRC to scale efficiently, handling millions of SNPs while maintaining high accuracy.

Beyond computational advantages, SBayesRC significantly improves cross-ancestry predictions, a common weakness in traditional PGS methods. By integrating SNP density and annotation data in a single Bayesian framework, it ensures that prediction accuracy is maintained even across genetically diverse populations.

In the next section, we'll compare SBayesRC with other PGS methods and analyze why it outperforms existing approaches.

2.4.1 SBayesRC vs other PGS methods

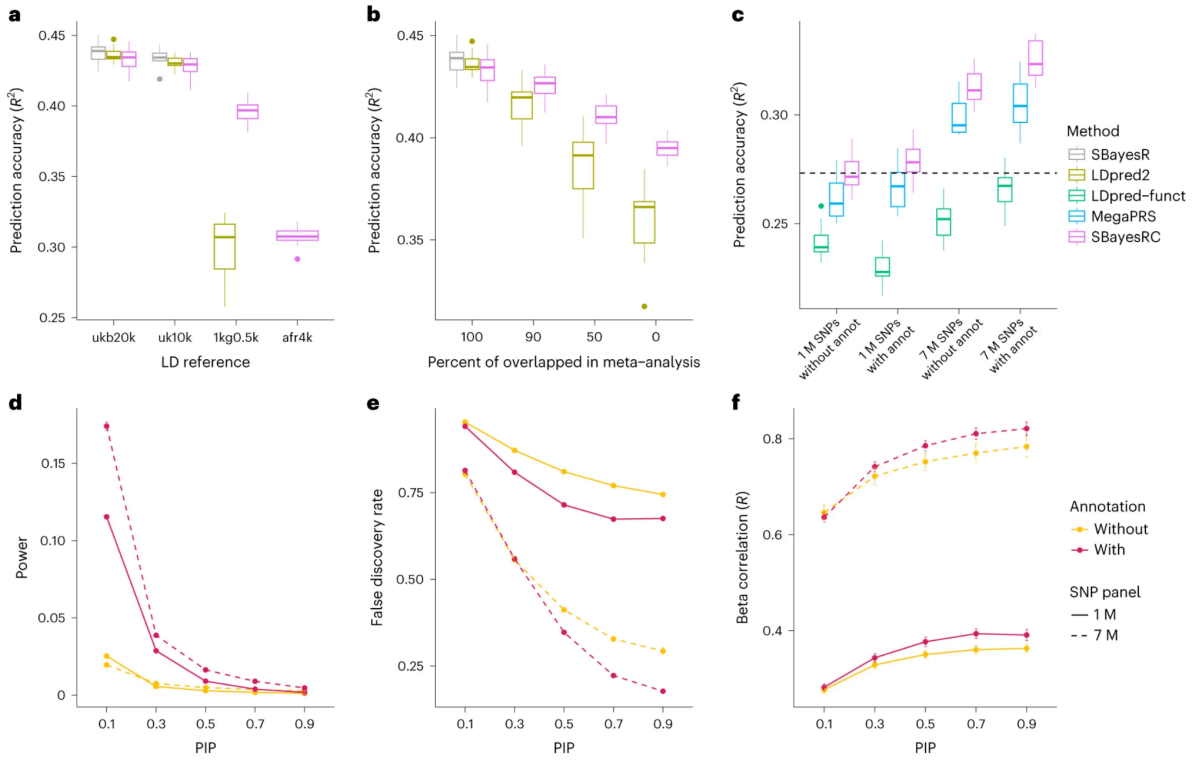

Figure 2.4.1: Performance of different methods by simulations

Polygenic risk scores (PGS) have come a long way, but many traditional methods struggle with accurately selecting SNPs and estimating their effects, especially across diverse populations. SBayesRC was designed to fix these issues by integrating functional annotations and using a Bayesian framework to refine SNP effect predictions.

Looking at the figure below, we can break down how SBayesRC stacks up against other methods. Panel (a) shows its robustness when using different LD reference panels. This is a crucial advantage since many models rely heavily on high-quality reference datasets, which may not always be available, especially for non-European populations. Panel (b) highlights how SBayesRC maintains accuracy even when SNP sample sizes are uneven, whereas other methods tend to struggle with these inconsistencies.

One of the biggest takeaways from panel (c) is how prediction accuracy improves as we incorporate more SNPs and functional annotations. While LDpred-funct and MegaPRS benefit from additional data, they don't reach the same level of performance as SBayesRC. This suggests that SBayesRC isn't just working with more data—it's making better use of it.

Panels (d) and (e) take this a step further by showing that SBayesRC is better at identifying true causal variants while keeping the false discovery rate (FDR) low. This is key in genetic research because misidentifying SNPs can lead to misleading associations and poor predictive performance. Finally, panel (f) shows that the SNP effect sizes estimated by SBayesRC align closely with the actual genetic architecture, making it a more biologically grounded and statistically reliable method for PGS prediction.

2.4.2 Usage of functional annotations to refine SNP selection

Not all genetic variants contribute equally to disease risk, some play a critical role in biological processes, while others are simply along for the ride. Traditional PGS methods don't distinguish between them, often leading to noisy predictions. Functional annotations help refine SNP selection by adding biological context, highlighting variants in regulatory regions, protein-coding genes, and evolutionarily conserved sites. By incorporating this information, we can better identify which SNPs are likely to be truly causal rather than relying solely on statistical associations *(Finucane et al., 2015).*

SBayesRC takes this a step further by integrating functional annotations directly into its Bayesian framework, allowing it to refine SNP effect sizes dynamically. Instead of using a stepwise approach, where annotations are applied after effect sizes are estimated, SBayesRC learns from both genomic data and functional information simultaneously. This not only improves the accuracy of polygenic risk scores but also enhances cross-ancestry predictions, as functional annotations remain consistent across populations. By prioritizing biologically meaningful SNPs, this approach provides a more interpretable and reliable way to assess genetic risk.

2.4.3 Bayesian vs Non-Bayesian approaches

Polygenic risk score models generally fall into two categories: Bayesian and non-Bayesian approaches *(Zhou, Carbonetto, & Stephens, 2013)*. Traditional methods like LDpred2 and MegaPRS use frequentist approaches, which estimate SNP effect sizes based on direct statistical associations observed in GWAS data. These models assume that effect sizes follow a fixed distribution and apply shrinkage techniques to adjust for linkage disequilibrium (LD) and sample size variations. While effective in certain cases, non-Bayesian methods struggle when dealing with sparse data, small effect sizes, and cross-ancestry predictions since they lack a probabilistic framework to account for uncertainty.

Bayesian approaches, like SBayesRC, tackle these issues by treating SNP effect sizes as probabilistic distributions rather than fixed values. Instead of assigning a single effect size to each SNP, they estimate a range of possible values and update their beliefs iteratively using prior information. This allows them to incorporate functional annotations, LD structure, and prior genetic knowledge into the model, refining SNP selection and effect estimation in a way that adapts dynamically to different datasets. The advantage of Bayesian models is their ability to filter out noise, capture complex genetic architectures, and generalize better across populations, making them particularly powerful for polygenic risk prediction in diverse cohorts.

METHODOLOGY

This chapter outlines the step-by-step process used to develop and evaluate the Transformer-based DNA language model. It begins by describing how genomic data was processed, including sequence extraction, phenotype conditioning, and sampling strategies. Next, the model architecture, training setup, and hyperparameter tuning experiments that shaped its predictive capabilities are detailed. The inference pipeline is then explained, covering how sequences were tokenized, fed into the model, and how logits were computed for genetic variation prediction. Finally, the integration of the model's outputs with SBayesRC is discussed, demonstrating how computed probabilities were formatted for polygenic risk score (PGS) estimation, ensuring a seamless transition from raw genomic data to meaningful risk prediction.

3.1 Data Processing and Sampling

To effectively train the Transformer-based DNA language model, a structured and efficient pipeline for processing genomic data was required. This study relies on the GTEx Whole Genome Sequencing (WGS) dataset (GTEx Consortium, 2017), which provides a diverse collection of genetic variations across individuals. However, raw genomic data is complex and requires careful preprocessing to extract meaningful information. A key aspect of the approach is integrating phenotype information with genetic sequences, allowing the model to learn relationships between specific mutations and observable traits. To assess the impact of this conditioning, the model is trained in two configurations: one with phenotype conditioning, where phenotype information is concatenated at the beginning of each sequence, and another without phenotype conditioning, where the model learns only from genomic sequences. This comparison allows for evaluating whether explicitly providing phenotype context enhances predictive performance.

The data processing workflow is built around a custom DataLoader class, which is responsible for handling variant call format (VCF) files, reference genome sequences, and phenotype data. The first step in this pipeline involves loading VCF files, which store genetic variants for each individual. Since these files are large and computationally expensive to process, they are properly indexed using tabix or bcftools, enabling fast and efficient querying. Alongside this, the reference genome is loaded, which acts as a baseline for identifying variations. The reference genome is fetched using pysam (Heger, 2009), ensuring that sequence extraction is accurate and computationally efficient. [(Li, 2011) – tabix, (Danecek et al., 2021) – bcftools].

A crucial component of the workflow is the integration of phenotype data, which is loaded from a tab-separated values (TSV) file. Phenotype labels are converted into a binary format (True/False) for ease of processing. This structured approach allows for incorporating phenotype context directly into the input sequences, which provides valuable information for the model when predicting genetic associations. However, to avoid potential data leakage, the dataset is split into training and validation sets at the individual level, ensuring that no individual appears in both datasets. This guarantees that the model is evaluated on unseen individuals, making performance metrics more reliable.

One of the challenges in processing genomic data is the inconsistency in chromosome naming conventions across different datasets. To address this, a chromosome mapping system is constructed to resolve discrepancies such as "1" vs. "chr1" in various data sources. This ensures smooth alignment between reference genomes and genetic variants, reducing potential errors when applying mutations. Additionally, instead of sampling randomly across the genome, gene-centered regions are prioritized, as these are more likely to be functionally relevant for phenotype prediction. Using GENCODE gene annotations, start and end positions of genes are extracted and extended with flanking regions to capture regulatory elements. Overlapping regions are merged to avoid redundancy, and a binary search-based approach is implemented to allow efficient weighted sampling from these predefined regions.

$$P(R_i) = \frac{L_i}{\sum_{j=1}^{N} L_j},$$

Where $P(R_i)$ is the probability of selecting region R_i, L_i is the length of region i, and the denominator is the total length across all N merged regions.

To simulate real-world genomic variation, the variant application process modifies the reference genome using the individual's specific genetic variants. Since multiple variants can overlap or conflict within a given sequence, a conflict-resolution strategy is applied, where variants are sorted by position and the highest-quality variant is selected when overlaps occur (Frankish et al., 2019).

Formally, the selected variant v^* from the set of overlapping variants $v_{overlap}$ is given by:

$$v^* = arg\ max_{v_k \epsilon V_{overlap}} QUAL(v_k),$$

where QUAL denotes the variant quality score provided in the VCF file.

The quality score itself is a Phred-scaled measure of the confidence in a variant call and is calculated as :

$$QUAL = -10\ log_{10}(P_{error}),$$

where P_{error} is the probability that the variant is false positive. This transformation ensures that more confident variant calls are assigned higher QUAL values, making them preferable during overlap resolution.

Before applying a variant, a validation step ensures that the substitution aligns with the reference sequence. Specifically, for a variant with reference allele REF of length ℓ, and a relative variant position i in the sequence S, the following condition must hold:

$$S[i:i+\ell]=REF.$$

If this condition fails, the variant is skipped to avoid introducing inconsistencies during substitution.

To quantify the cumulative change introduced by applied variants, the **Levenshtein distance** *(Levenshtein, 1966)* has been commuted between the reference sequence S and the modified sequence M:

d_L(S,M)= min { number of insertions, deletions, or substitutions to convert S into M }

This metric serves as a simple yet interpretable measure of base-level divergence introduced during preprocessing and variant application.

Once the sequences are extracted and modified, they are tokenized for model input. Each nucleotide is assigned a numerical representation: A (1), C (2), G (3), T (4), and N (5) for unknown bases, while padding tokens (0) are added to maintain sequence consistency. Additionally, for phenotype-conditioned training, we prepend a phenotype token to the sequence, where False is encoded as 6 and True as 7. This ensures that the model explicitly recognizes phenotype differences at the start of each sequence.

For model training, the dataset is integrated into a PyTorch DataLoader, which efficiently batches and shuffles the data. This allows for parallelized processing on GPUs, significantly speeding up training. The DataLoader supports both random sampling and position-based sampling, allowing the model to be trained with diverse genomic contexts. This flexibility ensures that the model is exposed to a broad range of genetic variations, improving its generalization capabilities.

In summary, the data processing pipeline is designed to extract, condition, and structure genomic sequences for deep learning applications. By incorporating phenotype information, implementing gene-centered sampling, and resolving variant conflicts, high-quality input is ensured for the Transformer-based model. The ability to train both with and without phenotype conditioning allows for systematic evaluation of the impact of phenotype inclusion on genomic prediction, providing valuable insights into the role of genetic context in complex traits.

3.2 Model Architecture and Training

Now that the data preprocessing and sampling steps are in place, attention turns to the architecture of the model itself and how it was trained. This section outlines the core design of the DNA language model, how sequences are represented internally, and the training strategy used to optimize it. The role of phenotype-conditioning during training is also described, along with how it was evaluated. Finally, the process of tuning key hyperparameters to improve model performance is presented.

3.2.1 Transformer-based DNA Language Model

The core model architecture used in this study is based on the Transformer, a neural network design that has proven effective in modeling complex dependencies in sequential data. This architecture was adapted specifically for genomic sequences by treating each nucleotide as an individual token. The input to the model consists of a fixed-length sequence of nucleotides,

which may also include an additional token at the beginning to represent phenotype information. When present, this token serves as a context marker, allowing the model to condition its predictions on the associated trait or disease status.

Each token in the sequence is first mapped to a learnable embedding vector. Positional encodings are added to ensure the model understands the order of tokens.

Each nucleotide $x_i \in \{A, C, G, T, N, [PAD]\}$ is mapped to a learnable embedding vector $e_i \in R^d$, where d is the embedding. The final input representation Z_i at position i is given by:

$$z_i = e_i + P_i,$$

Where P_i is the positional embedding. This combined representation is passed through the Transformer layers. Positional embeddings P_i are learnable and help the model distinguish between the same nucleotide appearing at different genomic positions, which is critical in functional genomics.

The sequence of embeddings is then passed through several stacked layers of the Transformer. Each layer contains a multi-head self-attention mechanism that allows the model to attend to different parts of the sequence dynamically. Given an input sequence $X \in R^{T \times d}$, where T is the sequence length and d is the embedding dimension, the model computes:

$$Q = XW^Q,\ K = XW^K,\ V = XW^V,$$

Where $W^Q,\ W^K,\ W^V \in R^{d \times d_k}$ are the learnable projections matrices for queries, keys and values respectively, and $d_k = \frac{d}{h}$ for h attention heads.

The scaled dot-product attention is then computed as:

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

where $M \in R^{T \times T}$ is the causal mask. For a sequence of length T, define a $T \times T$ matrix M, where each entry $M_{i,j}$ is set to 0 if position j is less than or equal to i, and set to negative infinity if position j is less than or equal to i.

A casual mask $M \in R^{T\times T}$ is applied to enforce the autoregressive constraint. This mask prevents the model from accessing future positions by assigning a value of $-\infty$ to all positions $j > i$, such that the attention score at position i only depends on tokens $\leq i$. This ensures the model adheres to left-to-right sequence modeling and prevents information leakage from the future positions during training. The softmax function then normalizes the masked similarity scores into attention weights α_{ij} where :

$$\alpha_{ij} = \frac{e^{s_{ij}}}{\sum_{k=1}^{T} e^{s_{ik}}}, \text{ with } S_{ij} = \frac{Q_i . K_j^T}{\sqrt{d_k}} + M_{ij}.$$

This produces a probability distribution over all prior positions, which is used to compute a weighted sum of the values vectors.

In practice, the model employs **multi-head attention**, which allows the Transformer to jointly attend to information from different representation subspaces. Given h heads, the input $X \in R^{T\times d}$ is projected into queries, keys, and values for each head using separate learnable weight matrices:

$$Q_i = XW_i^Q , K_i = XW_i^K , V_i = XW_i^V \text{ for } i = 1,.....,h.$$

Each head computes attention independently as:

$$head_i = Attention(Q_i, K_i, V_i).$$

These are then concatenated and projected using:

$$MultiHead\,(X) = Concat(head_1, head_2,.....head_h)W^o.$$

This enables the model to capture relationships at multiple scales simultaneously. This is particularly useful for genomic data, where a mutation in one part of the sequence might be influenced by regulatory elements located much farther away.

Following the multi-head attention mechanism, each token representation is passed through a **position-wise feedforward network (FFN)**. In the implemented architecture, this consists of two linear transformations with a ReLU non-linearity in between, followed by dropout:

$$FFN(x) = Dropout(W_2.\ ReLU(W_1\ .\ x)),$$

Where $x \in R^d$ is the output from the attention sublayer at each sequence position, $W_1 \in R^{d\times 4d}$ expands the representation to a higher-dimensional space, $W_2 \in R^{4d\times d}$ projects it back to the original embedding dimension, **ReLU** (Rectified Linear Unit) is the nonlinearity applied element-wise and is defined as $ReLU(z) = max(0, z)$. This function introduces nonlinearity into the model, allowing it to learn complex patterns by zeroing out negative activations. Finally, Dropout is applied after the final layer to regularize training and reduce overfitting.

To ensure stable training and facilitate gradient flow in deep networks, residual connections and layer normalization are applied around each sublayer (both attention and feedforward). Specifically, for an input x, the sublayer output is computed as

$$X_{out} = X + Sublayer(LayerNorm(X)).$$

This formulation is used twice per Transformer block, Once for the multi-head attention and once for the feedforward sublayer. Layer normalization normalizes each feature dimension across the token's embedding. The normalized output is computed as:

$$LayerNorm(X) = \frac{X-\mu}{\sigma+\epsilon}\ .\ \Upsilon + \beta,$$

Where $\mu = \frac{1}{d}\sum_{i=1}^{d} x_i$ is the mean and $\sigma = \sqrt{\frac{1}{d}\sum_{i=1}^{d}(x_i - \mu)^2 + \varepsilon}$ is the standard deviation with ε a small constant added for numerical stability. While Υ, $\beta \in R^d$ are learnable parameters, respectively. These parameters enable the model to preserve representational capacity after normalization.

At the end of the transformer stack, a final layerNorm is applied, and each hidden state $h_t \in R^d$ is projected into the nucleotide vocabulary using a learned weight matrix $W \in R^{|\upsilon| \times d}$, $where\ |\nu| = 6(A, C, G, T, N, [PAD])$. The output logits are computed as:

$$Z_t = W \cdot h_t .$$

The model then outputs a probability distribution over the vocabulary using a softmax:

$$P(x_{t+1} = \upsilon_i \mid x_{\leq t}) = \frac{e^{z_{t,i}}}{\sum_{j=1}^{|\upsilon|} e^{z_{t,j}}} .$$

This converts the logits z_t into valid probabilities for the next nucleotide. This formulation completes the autoregressive modeling pipeline used for next-token prediction in the DNA language model.

In addition to the phenotype-conditioned version of the model, a second model was trained using the same architecture but without any phenotype information in the input. This version was included for comparison, allowing us to evaluate the added value of explicitly conditioning the model on phenotype when learning sequence-level representations. A more detailed discussion of the training process and this comparative setup is presented in the following section.

Overall, the Transformer-based DNA language model provides a flexible and powerful framework for modeling genomic sequences. By incorporating both local and distant sequence dependencies, and optionally conditioning on phenotype, the model is capable of capturing complex patterns that are relevant for downstream tasks such as variant effect prediction and polygenic risk estimation.

3.2.2 DNA Language Model Training

Training the DNA language model required a setup that could efficiently handle large-scale genomic data while also being flexible enough to accommodate multiple configurations. The model was trained using PyTorch *(Paszke et al., 2019)* with full GPU acceleration, and training was executed on a high-performance cluster equipped with A100-SXM4 GPUs and 256 GB of memory. The primary objective during training was to predict the next nucleotide in a sequence, using a standard language modeling setup. For each training batch, the model received

sequences as input and was optimized to assign high probability to the correct next nucleotide at each position, following an autoregressive approach.

The loss function used during training was cross-entropy loss, which compares the model's predicted probability distribution to the actual target nucleotide:

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N} log\, P(x_i \mid x_{<i}),$$

Where N is the number of tokens in the batch, x_i is the true nucleotide at position i and $P(x_i \mid x_{<i})$ is the model's predicted probability (from softmax) for the correct token given previous context.

This loss was averaged across the batch and back propagated through the model. To ensure that the model did not access future tokens while making predictions, causal masking was applied, enforcing a left-to-right learning pattern.

Gradient accumulation was used to manage memory usage and stabilize training employed over **G mini-batches**, especially when using large batch sizes on long sequences, simulating a larger batch size $B_{eff} = G \cdot B$, where B is the per-step batch size.

Additionally, training and validation datasets were carefully split at individual level to avoid data leakage, ensuring that no sequence from a single person appeared in both sets.

To evaluate the impact of phenotype conditioning, two versions of the model were trained in parallel. One model received phenotype tokens at the beginning of each input sequence, allowing it to condition its predictions on the presence or absence of a disease-related trait . In this case diabetes type -1. The second model was trained on the same sequences without phenotype information. The architecture and training parameters were kept identical for both versions to make the comparison meaningful. The evaluation of both models was based on validation loss, which was monitored across training epochs to track convergence and stability.

Training was monitored and tracked using the Weights & Biases platform *(Biewald, 2020)*, which helped visualize training and validation loss over time, and compare the effects of different hyperparameter settings. These logs were essential in understanding how changes in model size, input sequence length, and batch configuration affected learning dynamics. Each run was

logged with the corresponding hyperparameters, making it easier to organize and interpret the results of different experiments.

Several key hyperparameters were varied systematically to test the model's capacity and sensitivity. Embedding dimensions were tested at multiple scales, including 64, 256, 512, and 1024. Similarly, block sizes referring to the context window size used during attention computations were explored at values of 4, 8, 16, and 32. These changes allowed the model to capture dependencies over different ranges, and helped determine the trade-off between model complexity and performance. Sequence length was another important factor. Experiments were run on sequence lengths of 32, 64, 128, 256, 512, 1024, and 2048 tokens.

To efficiently explore the wide range of hyperparameter combinations, multiple training runs were executed in parallel, each running for approximately eight hours. This allowed a broad sweep across different embedding sizes, block sizes, and sequence lengths while keeping compute usage manageable. Based on the validation performance and training stability observed during these runs, the configuration with a batch size of 32, embedding dimension of 512, and sequence length of 1024 tokens emerged as the most balanced setup. This combination offered a strong trade-off between accuracy, generalization, and computational efficiency. Once identified, this configuration was used for the final full-scale training runs, which were extended to 16 hours to ensure convergence while making optimal use of the available compute resources.

3.2.3 Optimization Strategy and Numerical Modeling Configuration

To ensure effective training of the transformer-based DNA language model, careful attention was paid to optimization techniques, architectural parameters, and numerical stability. This section provides a detailed breakdown of the optimizer, loss formulation and model configuration.

The model was trained to minimize the **cross-entropy loss**, a standard choice in language modeling tasks, defined as:

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N} log\ P(x_i \mid x_{<i}),$$

where x_i denotes the true nucleotide at position i, $x_{<i}$ the preceding context, θ the model parameters, and N the total number of tokens in the batch. This objective encourages the model to assign high probability to the correct nucleotide given its upstream genomic and phenotypic context. This Training was conducted using the **Adam optimizer**, a widely used adaptive

gradient descent method. The configuration with a Learning Rate: 1×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.95$, Weight Decay of 0.01 to regularize the model and prevent overfitting. To manage memory and increase the effective batch size during training, gradient accumulation was used with a step size of 2. This setup allowed the model to be trained stably across large genomic sequences while maintaining convergence efficiency.

With 100 batches per epoch and 100 batches per evaluation. Gradient accumulation was applied to simulate a larger batch size by accumulating gradients over two mini-batches before performing a weight update, thus effectively increasing the batch size without exceeding memory limits.

The full Transformer architecture was implemented from scratch in PyTorch and comprises several key components tailored for genomic sequence modeling. The model includes 12 Transformer blocks, each featuring a multi-head self-attention mechanism with 8 attention heads. The embedding dimension was set to 512, ensuring sufficient representational capacity while maintaining computational tractability. Each input sequence was tokenized into a fixed length of 1024 tokens, corresponding to the upstream genomic context used for autoregressive prediction.

Dropout with a probability of 0.1 was applied throughout the network to regularize training and mitigate overfitting. The nucleotide vocabulary consists of six tokens A, C, G, T, N and PAD (padding). For phenotype-conditioned models, the vocabulary was extended to include binary tokens corresponding to phenotype presence or absence, increasing the size to eight.

The model uses ReLU (Rectified Linear Unit) as the activation function in all feedforward sublayers, introducing non-linearity and enabling the network to learn complex biological dependencies. The total number of trainable parameters in the model is approximately 38.34 million, accounting for the embedding layers and the full Transformer stack. This configuration was chosen to balance expressiveness, memory efficiency, and training stability, making it well-suited for large-scale genomic inference tasks.

Training was carried out on a high-performance computing cluster utilizing four NVIDIA A100-SXM4-80GB GPUs. The model was trained using full FP32 precision, without mixed-precision optimizations. Each training batch consisted of 32 sequences, each of length 1024 tokens. The training process spanned a total of 36 epochs and was managed via the Slurm workload manager on the `pgpu` node.

The total training runtime was approximately 48 hours, with parallel processing enabled across all four GPUs. During peak usage, each GPU exhibited a power draw in the range of 310 to 320 watts. This setup provided sufficient compute throughput to handle the large parameter space and sequence length, while maintaining training stability and reproducibility across runs.

Memory constraints were mitigated using PyTorch's `torch.utils.checkpoint` module for selective gradient computation in transformer blocks, reducing GPU memory overhead during the forward pass.

Table 3.2.3: Model Configuration Summary

S.No	Parameter	value
1	Sequence Length	1024
2	Embedding Dimension	512
3	Attention Heads	8
4	Transformer layers	12
5	Dropout	0.1
6	Optimizer	Adam
7	Learning Rate	$1e^{-4}$
8	Batch size	32
9	Training Epochs	36
10	Total Parameters	~38.34M
11	Training Precision	FP32
12	GPUs used	NVIDIA A100-SXM4-80GB

3.3 Inference Process

Once the model had been trained and validated, the next step involved using it to make predictions on new genomic sequences. This phase, known as inference, focuses purely on evaluating the model's output without updating its weights. In this context, the model was used to compute the probabilities of specific nucleotide occurrences across genomic regions, which

formed the basis for downstream applications such as variant prioritization and polygenic risk scoring. This section outlines how input sequences were prepared during inference and how the model's outputs, specifically the logits were computed and interpreted.

3.3.1 Model Input and Processing

During inference, the model was used to estimate the probability of observing specific nucleotides at known variant positions, given the surrounding genomic context. To do this, variant-level data was obtained from a large-scale population cohort, structured similarly to datasets from biobanks such as the UK Biobank *(Sudlow et al., 2015)*. This dataset included chromosome locations and both reference and alternate alleles for each variant of interest.

For each variant position p, a fixed-length DNA sequence was extracted from the reference genome, comprising 1024 nucleotides immediately upstream of the variant position. This provided the model with a consistent left-hand context, allowing it to predict the next base in the sequence, which corresponded to the variant site. This setup mirrors the autoregressive objective used during training, where the model learns to predict the next nucleotide based solely on preceding context.

The input sequence was tokenized using the same vocabulary and encoding scheme established during training. When using the phenotype-conditioned model, a phenotype token was prepended to the beginning of the sequence to indicate disease status. In this study, the conditioning was based on the presence or absence of Type 1 Diabetes. For the non-conditioned model, the same sequences were used without the additional phenotype token.

All input sequences were standardized to 1024 tokens. If the combined phenotype and DNA sequence exceeded this length, the sequence was truncated; otherwise, padding tokens were added to ensure uniform shape. This consistency ensured compatibility with the model's input expectations and allowed for efficient batch processing.

The model was set to evaluation mode, and inference was conducted without gradient tracking, which improved computational efficiency and reduced memory usage. Inference was performed in batches, each containing a subset of variants, allowing the process to scale to biobank-scale datasets. Both the phenotype-conditioned and unconditioned models were applied to the same set of variant positions, enabling a fair and interpretable comparison between the two approaches.

Through this process, the model received individual variant contexts and produced predictions that could be used to assess how well the model captured the statistical and biological patterns underlying genetic variation, both in a generic and phenotype-aware manner.

3.3.2 Computing Logits

Once the model receives an input sequence and processes it through its Transformer layers, the final output is a set of raw scores known as logits, which are computed for each position in the input sequence. These logits represent the model's unnormalized confidence about the identity of the next nucleotide at each location. In the autoregressive setup used throughout this study, the model is trained to predict the next base based solely on the preceding context. Therefore, for any given input, the most relevant prediction occurs at the final position,, the variant site.

Let the input sequence be denoted by $X = [x_1, x_2, \ldots\ldots\ldots\ldots, x_T]$, where T is the sequence length After passing through the Transformer layers, the model produces a sequence of contextualized hidden states $h_1, h_2, \ldots\ldots, h_T \in R^d$, one for each position. These hidden states are then projected into the nucleotide vocabulary using a learned weight matrix $W \in R^{|\upsilon| \times d}$, where |ν| represents the vocabulary size corresponding to the nucleotides. The logits at the final position t = T are computed as:

$$Z_t = Wh_t.$$

The resulting vector $Z_t \in R^{|\upsilon|}$ is the set of logits for the position t. These logits are then passed through a softmax function to obtain probability distribution over the nucleotide vocabulary:

$$P(\upsilon_i \mid x_{\leq t}) = \frac{e^{z_{t,i}}}{\sum_{j=1}^{|\upsilon|} e^{z_{t,j}}},$$

where $P(\upsilon_i \mid x_{\leq t})$ is the probability assigned to nucleotide i , conditioned on the upstream sequence context. Since the autoregressive model only considers prior tokens when making predictions, the output at position T corresponds to the model's prediction for the variant site at position $T + 1$. Therefore, only the logits Z_T are used for downstream inference tasks.

For each variant, the model estimates probabilities for both the reference and alternate alleles. Let ν_{ref} and ν_{alt} denote the reference and alternate alleles respectively. The model-derived probabilities are then given by:

$$P_{ref} = P(\nu_{ref} \mid X_{\leq T}), P_{alt} = P(\nu_{alt} \mid X_{\leq T}).$$

From a modeling perspective, these probabilities offered a way to quantify how plausible each allele was, not just statistically, but functionally as well. A high probability for the reference allele might suggest that the model sees it as consistent with the genomic background, whereas a low probability for the alternate allele might indicate a more disruptive or unusual variant. When used in bulk across thousands or millions of variants, these predictions begin to reflect the model's internal understanding of functional relevance, making them valuable candidates for use as priors in a polygenic risk score framework.

Importantly, this inference pipeline was applied to both the phenotype-conditioned and non-conditioned versions of the model, allowing the collection of two sets of outputs: one reflecting generic sequence modeling, and one enriched with phenotype-specific context (in this case, Type 1 Diabetes). All sequences were standardized to a length of 1024 tokens, with phenotype tokens prepended in the conditioned case. Sequences were truncated or padded as necessary to maintain this fixed input shape.

By computing these probabilities consistently for both phenotype-conditioned and non-conditioned models, two parallel datasets of model-derived outputs were prepared. These were later integrated with summary statistics in the PRS framework, allowing for assessment of how much information the model had captured and whether conditioning on phenotype improved its ability to identify functionally important variants.

3.4 Benchmarking Model Outputs Using Public Polygenic Score Weights

To evaluate whether the DNA language model captures meaningful biological signals, variant-level effect sizes were obtained from the PGS Catalog for the trait Type 1 Diabetes Mellitus (MONDO:0005147). These effect sizes quantify the contribution of individual genetic variants to disease risk and serve as an external benchmark for assessing the biological relevance of the model's outputs. Two polygenic score models were used in this analysis. The first, PGS002025, titled *portability-ldpred2_250.1*, originates from Privé et al. (2022) and is based on genome build GRCh37. It includes 106,800 variants and is associated with DOI: 10.1016/j.ajhg.2021.11.008. The second model, PGS003993, titled

dbslmm.auto.GCST90013445.T1D, was introduced by Monti et al. (2024), is aligned to genome build GRCh38, and includes 63,182 variants (DOI: 10.1016/j.ajhg.2024.06.003).

Each scoring file contains variant-level identifiers (rsIDs), reference and effect alleles, and corresponding beta coefficients representing effect sizes. They served as benchmarks for evaluating model behavior at the resolution of single variants. Logits were generated from both a phenotype-conditioned model and a non-conditioned baseline for each SNP present in the PGS files. The logit associated with the effect allele was extracted by aligning rsIDs and matching the corresponding logit column based on the reported allele information.

To ensure that only informative variants were included in the analysis, an effect size filter was applied to each scoring file. For PGS002025, variants were retained if their absolute effect size exceeded 0.0001. In contrast, for PGS003993, a more stringent threshold of 0.1 was applied, reflecting the smaller dynamic range of effect sizes reported in that file.

Variants missing logits or failing to meet the thresholds were excluded. Pearson correlation coefficients were then computed between the model-derived logits and the absolute values of the PGS beta coefficients, separately for the conditioned and non-conditioned models. The correlation was computed using the scipy.stats.pearsonr function from SciPy (Virtanen et al., 2020), based on the original statistical formulation by Pearson (1895).

In addition to numerical correlation, visualizations such as scatter plots, histograms, and kernel density estimates (KDEs) were used to explore the relationship between model confidence and variant effect size. These comparisons helped assess whether the model's outputs reflect meaningful biological signals aligned with known genetic contributions to disease.

3.5 Integrating with SBayesRC

After training and evaluating the DNA language model, the next step was to assess whether its outputs could enhance genetic risk prediction in practice. This was done by integrating the model-derived scores into **SBayesRC**, a Bayesian framework for polygenic risk scoring (PRS) that combines GWAS summary statistics (Karczewski et al., 2018) with variant-level annotations to estimate posterior SNP effect sizes. These annotations act as priors that influence the model’s belief about the likelihood of each variant being causal.

In this work, the DNA language model generated scalar scores for each variant—based on the logit difference between alternate and reference alleles in a 1024-base context—which were

used as continuous functional annotations. These were injected into the SBayesRC framework in place of traditional annotations such as conservation or chromatin state, allowing the model's learned sequence-level representations to guide inference in a probabilistically grounded way.

This approach offers a robust benchmark for DNA-LLMs. Rather than relying on isolated or manually curated tasks, polygenic risk scoring integrates information across the entire genome, implicitly capturing both *known* functional relationships (e.g., disease-associated loci) and *previously unknown* sequence features that may be biologically relevant. As a result, this setup functions like an integration test: it evaluates not only whether the model has encoded meaningful representations, but also whether those representations can be operationalized to improve real-world prediction.

By quantifying downstream performance using metrics like AUROC, this framework provides a direct and interpretable way to assess the utility of sequence-based models for clinical and research applications.

3.5.1 Preparing Data for Integration

The integration process began with assembling the necessary input data. This included GWAS summary statistics for Type 1 Diabetes from a large-scale biobank cohort (Karczewski et al., 2018), containing variant-level attributes such as chromosome position, reference and alternate alleles, beta coefficients, standard errors, p-values, and allele frequencies.

Each single nucleotide polymorphism(SNP) v_i is characterized by the following attributes: Reference allele $A_1^{(i)}$, Alternate Allele $A_2^{(i)}$, Effect size estimate $\beta^{(i)}$, Standar error $SE^{(i)}$, P-value $p^{(i)}$, and Allele frequency: $f^{(i)} \in [0, 1]$

$$MAF^{(i)} = min(f^{(i)}, 1\text{-}f^{(i)}) > 0.01.$$

While the framework allows for filtering based on minor allele frequency, the variants used in this analysis were selected based on the intersection between SNPs present in the UK Biobank GWAS and those for which annotations were available from the DNA language model outputs.

This curated set ensured that every variant had both an effect size from the GWAS and a corresponding functional annotation from the model, enabling seamless integration into the SBayesRC pipeline for downstream effect size modeling and risk prediction.

3.5.2 Using Language Model Outputs as Functional Annotations

SBayesRC was originally intended to utilise biological annotations, such as gene regulatory markers or coding region tags, as priors for estimating SNP effect sizes. In this case, features from the DNA language model served as an alternative source of functional information.

For each variant υ_i, the trained language model generated a probability distribution over the nucleotide vocabulary at the variant site, conditioned on the preceding 1024-base upstream context $X_{<t}$. This mirrors the autoregressive prediction objective used during training.

Formally, for each nucleotide, the model produces:

$$P_{LM}(x_t = a|x_{<t}) = \frac{exp(z_{t,a})}{\sum_{j=1}^{|v|} exp(Z_{t,j})},$$

Where $Z_{t,a}$ is the logit assigned to nucleotide a at position t, $x_{<t}$ is the 1024-nucleotide upstream context and |v| = is the size of the nucleotide vocabulary.

In practice, rather than using the full probability vector, a single scalar value was extracted for each variant to serve as the functional annotation. This value represents the difference between the model's logit for the alternate allele and the reference allele:

$$Annot_i = Z_{alt\,i} - Z_{ref\,i}.$$

This logit difference captures the relative preference of the model for the alternate allele over the reference allele at a given genomic position. A positive value indicates the model assigns higher confidence to the alternate allele in the given sequence context, whereas a negative value suggests preference for the reference allele

Importantly, from a biological perspective, this signal may reflect selective pressure: variants that are deleterious or harmful may be less likely to occur in conserved regions and, therefore, receive lower probability from the model. As a result, the logit difference can serve as a proxy for functional impact, offering a novel way to prioritize variants in downstream polygenic scoring applications.

To make these values compatible with thePRS framework, an annotation file was constructed for both the phenotype-conditioned and non-conditioned models. Each file consisted of three

columns: **SNP**, the rsID of the variant, **FILL**, a constant column with value 1.0 (a required placeholder by the pipeline), **Annot1**: the computed logit difference described above.

This resulted in a one-dimensional annotation matrix $A \in R^{n\times 1}$, where n is the number of variants. Unlike previous approaches that combine multiple biological features, this setup isolated the effect of model-derived sequence representations alone. This allowed for direct comparison between models trained with and without phenotype conditioning in terms of their ability to inform SNP effect estimation.

3.5.3 Annotations Preprocessing and Feature Selection

Prior to integration with the PRS framework, all annotation features were systematically preprocessed to ensure numerical stability and interpretability within the Bayesian modeling framework. The full annotation matrix, denoted as:

$$A \in R^{n\times d},$$

consisted of n single-nucleotide polymorphisms (SNPs) and d annotation features. In this case, d=1, corresponding to a single model-derived feature: the logit difference between the alternate and reference alleles produced by the DNA language model.

To prepare the annotation matrix for use in SBayesRC, all continuous features were standardized to have zero mean and unit variance. For a given feature x, the transformation was defined as:

$$x' = \frac{x - \mu}{\sigma},$$

Where μ and σ denote the sample mean and standard deviation of the variable, respectively. This normalization step ensures that all features contribute comparably to the model, regardless of their original scale.

After this standardization, the resulting annotation matrix A′ was passed to the PRS framework as the functional prior used in Bayesian SNP effect size estimation. The simplicity of using a single, model-derived feature also allowed for a direct evaluation of the utility of DNA language model outputs compared to traditional annotation sources.

3.5.4 Polygenic Risk Scoring with SBayesRC

The final stage of the study involved integrating the outputs of the DNA language model into the SBayesRC framework to evaluate their utility for polygenic risk prediction. SBayesRC is a hierarchical Bayesian model designed to estimate SNP-level effect sizes by incorporating both GWAS summary statistics and functional annotations. It assumes that the true effect size βi of SNP i arises from a finite mixture of zero-mean Gaussian distributions, where the prior probabilities of component membership are informed by variant-specific annotations:

$$\beta_i \sim \sum_{k=1}^{K} \pi_k(i) \,.\, \mathcal{N}(0, \sigma_k^2) \,,$$

Where $\pi_k(i)$ denotes the probability of SNP i belongs to the k-th mixture component with variance σ_k^2. These probabilities are not fixed but are dynamically determined as a function of the SNP's annotation vector $a_i \in R^d$, using a log linear transformation followed by softmax function:

$$\pi_k(i) = \frac{exp(w_k^T a_i)}{\sum_{l=1}^{k} exp(w_l^T a_i)} \,,$$

Where $w_k \in R^d$ is a vector of learned weights associated with component k and a_i encodes the processed annotations for SNP i. These annotations may include classical biological features as well as outputs derived from the DNA language model, such as allele-specific probabilities and derived log-ratios.

3.5.5 Risk score construction

Once posterior mean effect sizes β_i have been estimated for each SNP, they can be used to compute polygenic risk scores (PRS) for individual-level genotype data. For a given individual with genotype $g_i \in \{0, 1, 2\}$ at SNP i, representing the number of alternate alleles carried, the PRS is computed as the weighted sum of genotypes:

$$PRS = \sum_{i=1}^{n} \beta_i . g_i,$$

where n is the total number of SNPs considered. This scalar quantity provides a quantitative estimate of the individual's genetic liability to the trait under investigation, Type 1 Diabetes in this case. The PRS can be used in downstream analyses for stratifying risk, identifying outliers, or modeling genetic predisposition in conjunction with clinical covariates.

Results And Discussions

4.1 Hyperparameter Tuning and Model Selection

To determine the most effective architecture, multiple hyperparameter combinations were evaluated in parallel. These experiments focused on varying embedding dimensions, sequence lengths and batch sizes Each run was tracked using Weights & Biases, with validation loss serving as the main criterion for comparison. Each training run was labeled using the format **`sequenceLength-embeddingDim-batchSize`**, allowing for easy identification and comparison across experiments. For example, a run named `1024-512-32` corresponds to a model trained with a sequence length of 1024, embedding dimension of 512, and batch size of 32.

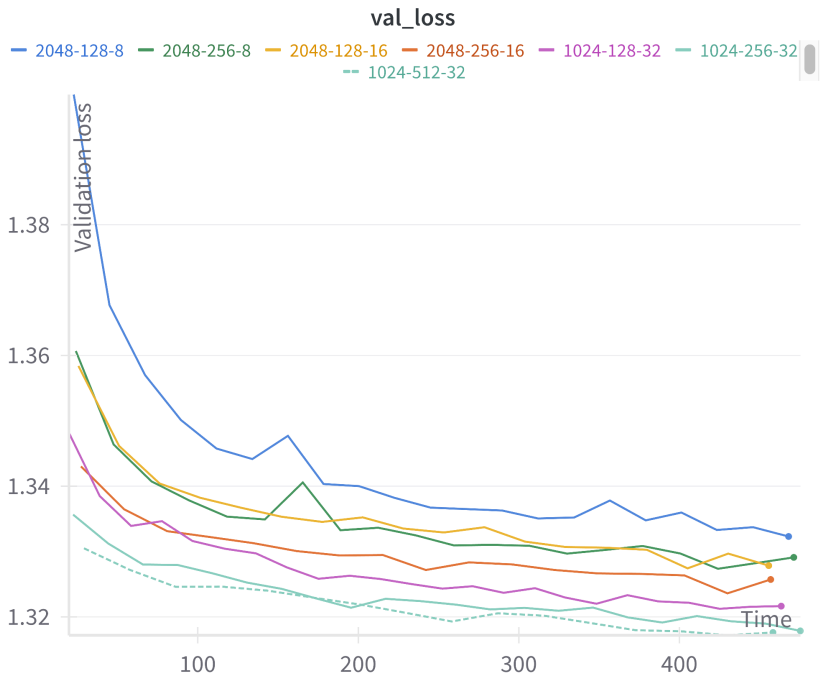

Figure 4.1(a): Validation Loss Across Hyperparameter Configurations: 1024-512-32 Achieves the Lowest Loss

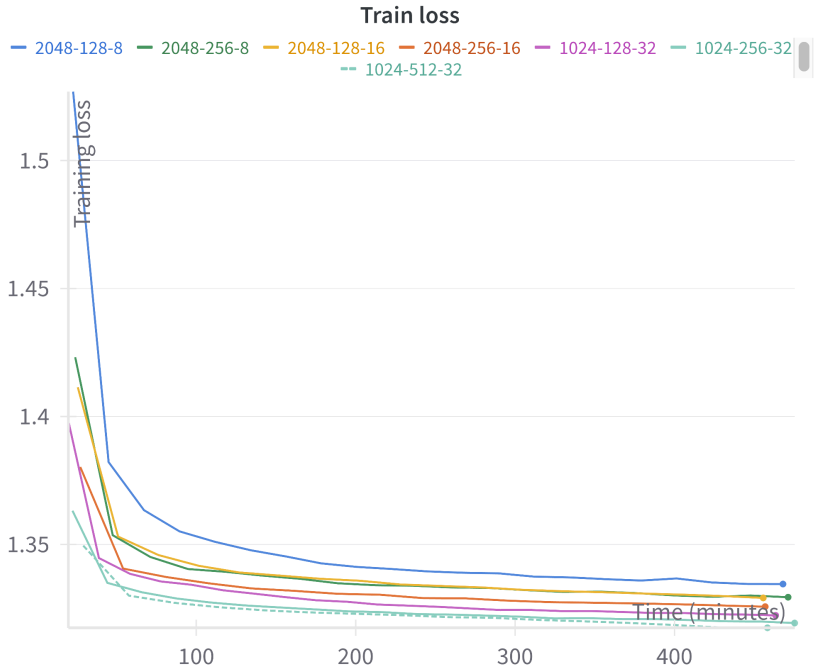

Figure 4.1(b): Training Loss Across Hyperparameter Configurations: 1024-512-32 Achieves the Lowest Loss

Figures 4.1(a) and 4.1(b) display the validation and training loss curves, respectively, across various hyperparameter configurations. Among the combinations tested, the setup with a 512-dimensional embedding, a sequence length of 1024 tokens, and a batch size of 32 consistently achieved the lowest validation loss and exhibited a stable training trajectory. Based on these results, this configuration was chosen as the final model and was trained further to ensure full convergence. All subsequent evaluations and inference experiments were conducted using this selected model.

In addition to evaluating performance through loss metrics, GPU power usage was also monitored during training. As shown in **Figure 4.1(c)**, a comparative plot of GPU power consumption across different hyperparameter combinations revealed a clear pattern: the configuration with 1024-token sequence length, 512-dimensional embedding size, and batch size of 32 consistently consumed significantly more power throughout the entire training duration. This indicates that the best-performing model was also the most computationally demanding, likely due to its increased representational capacity and extended input context.

Table 4.1: Hyperparameter Tuning Results

S. No	Model Name	Embedding dim	Batch size	Sequence length	Train loss	Validation loss
1	2048-256-8	256	8	2048	1.32939	1.32911
2	2048-128-8	128	8	2048	1.33452	1.3323
3	2048-256-16	256	16	2048	1.32569	1.32572
4	20148-128-32	128	32	20148	1.32911	1.32785
5	1024-256-32	256	32	1024	1.31937	1.31792
6	1024-128-32	128	32	1024	1.32233	1.32171
7	1024-512-32	512	32	1024	1.31742	1.31762

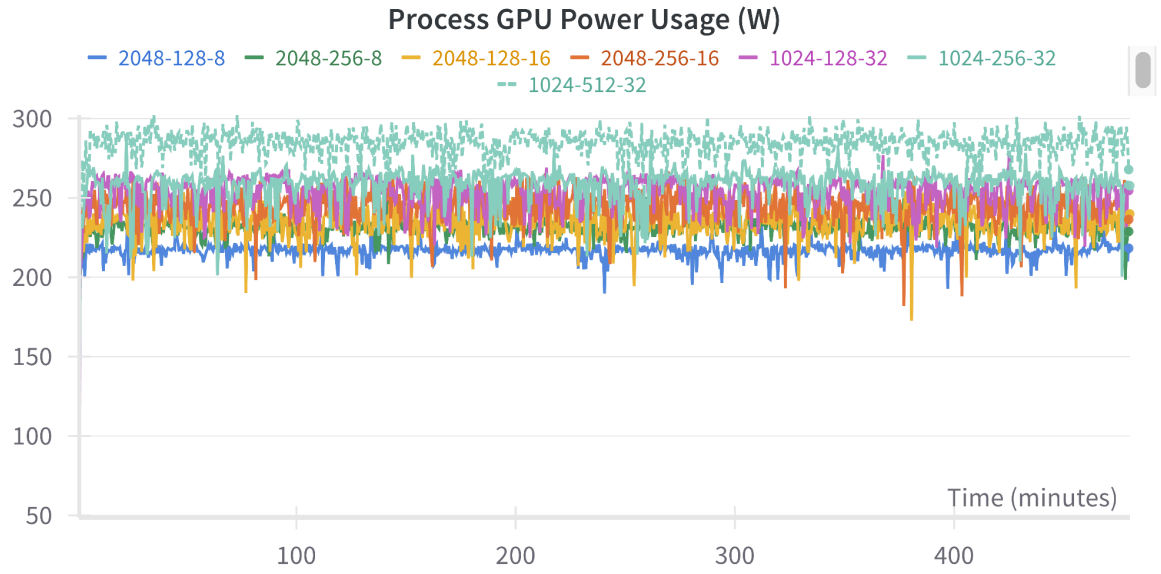

Figure 4.1(c): Comparison of GPU power usage (in watts) across hyperparameter configurations during training.

4.2 Model training

Following the selection of optimal hyperparameters, namely a sequence length of 1024, embedding dimension of 512, and batch size of 32, the final models were trained for 36 epochs. Two model variants were considered, one incorporating phenotype conditioning and one without any conditioning. Enabling a direct evaluation of the impact of phenotype information on learning dynamics.

Both models were trained on the same dataset under identical conditions to ensure a controlled comparison. The progression of training and validation losses is shown in Figures 4.2(a) and 4.2(b). While the overall learning curves are closely aligned, the conditioned model consistently achieved slightly lower losses across epochs.

By the end of training, the conditioned model reached a training loss of 1.3201 and a validation loss of 1.3171. In contrast, the non-conditioned model concluded with a training loss of 1.3211 and a validation loss of 1.3191. These results indicate a modest but consistent improvement in both convergence and generalization when conditioning on phenotype is applied.

Furthermore, the validation curve of the conditioned model exhibits a smoother descent and slightly reduced variance across epochs, suggesting improved stability during training.

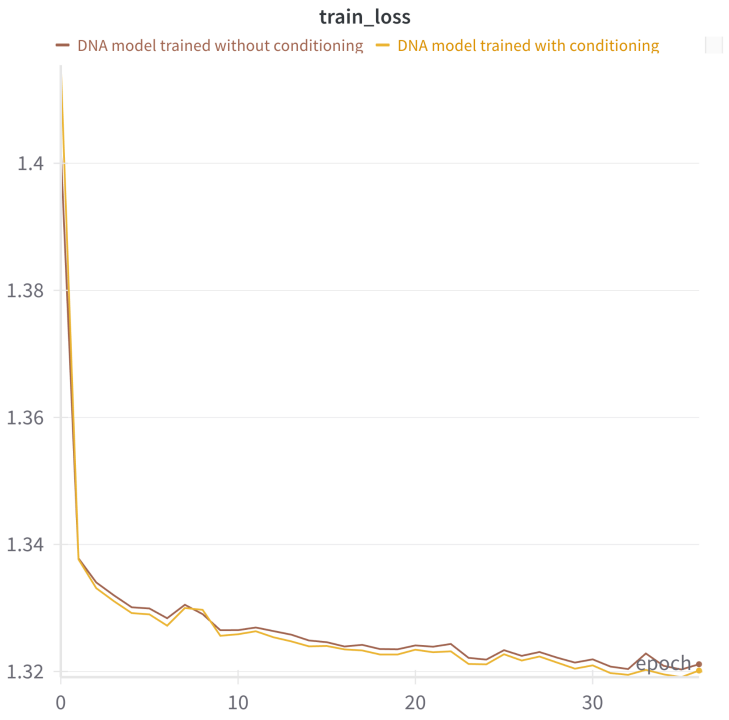

Figure 4.2(a): Train Loss Across 36 Epochs for conditioned and non conditioned model

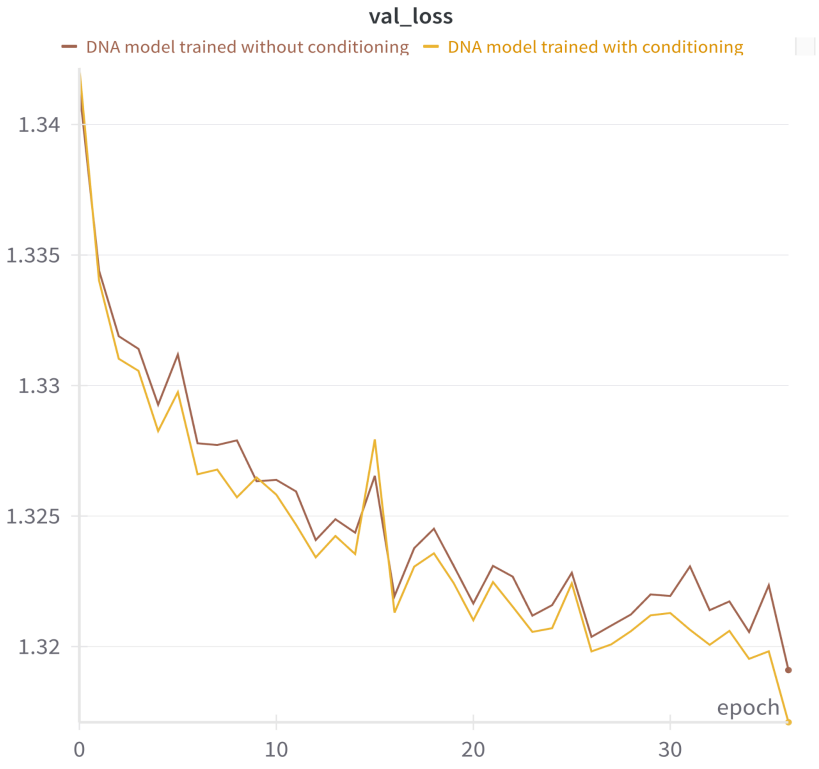

Figure 4.2(b): Validation Loss Across 36 Epochs for conditioned and non conditioned model

4.3 Logits generation and Analysis

Using the final models trained, predictions were generated for over 2.23 million SNPs across the genome. For each variant, model-derived probabilities were extracted for both the reference and alternate alleles under both the phenotype-conditioned and non-conditioned configurations. This allowed a direct comparison of the probabilistic outputs from the two models at a genome-wide scale, enabling an assessment of the effect of phenotype conditioning on variant-level predictions.

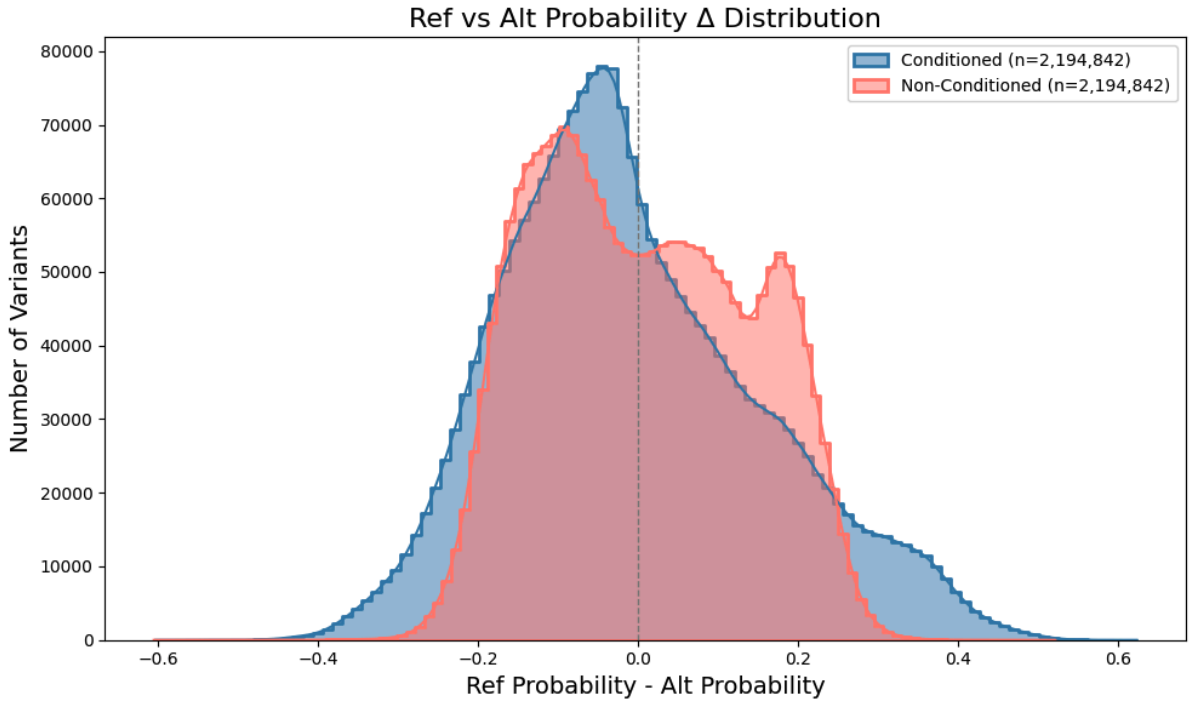

Figure 4.3(a): Distribution of predicted probability differences between reference and alternate alleles. The conditioned model shows heavier tails, suggesting stronger allele preference for a subset of variants, potentially reflecting phenotype-relevant signals.

This figure 4.3(a) illustrates the distribution of the difference between the model's predicted probability for the reference allele and the alternate allele, across all variants. A positive value implies the model favored the reference allele in its prediction, while a negative value indicates higher probability for the alternate allele.

Both the conditioned (blue) and non-conditioned (red) models exhibit a roughly symmetric distribution centered near zero, suggesting that in many cases, predictions are balanced between the two alleles.

However, the conditioned model displays noticeably broader tails, meaning it more frequently assigns strongly confident predictions in favor of one allele over the other. This reflects how conditioning on phenotype leads to greater polarization in model outputs, effectively pushing the model to be more decisive. The increase in extreme values suggests the conditioned model may be better at identifying functionally important variants where phenotype context influences allelic effects. This increased polarization raises the question of whether conditioning also shifts the absolute predicted probabilities at each site, an aspect explored in the following comparison.

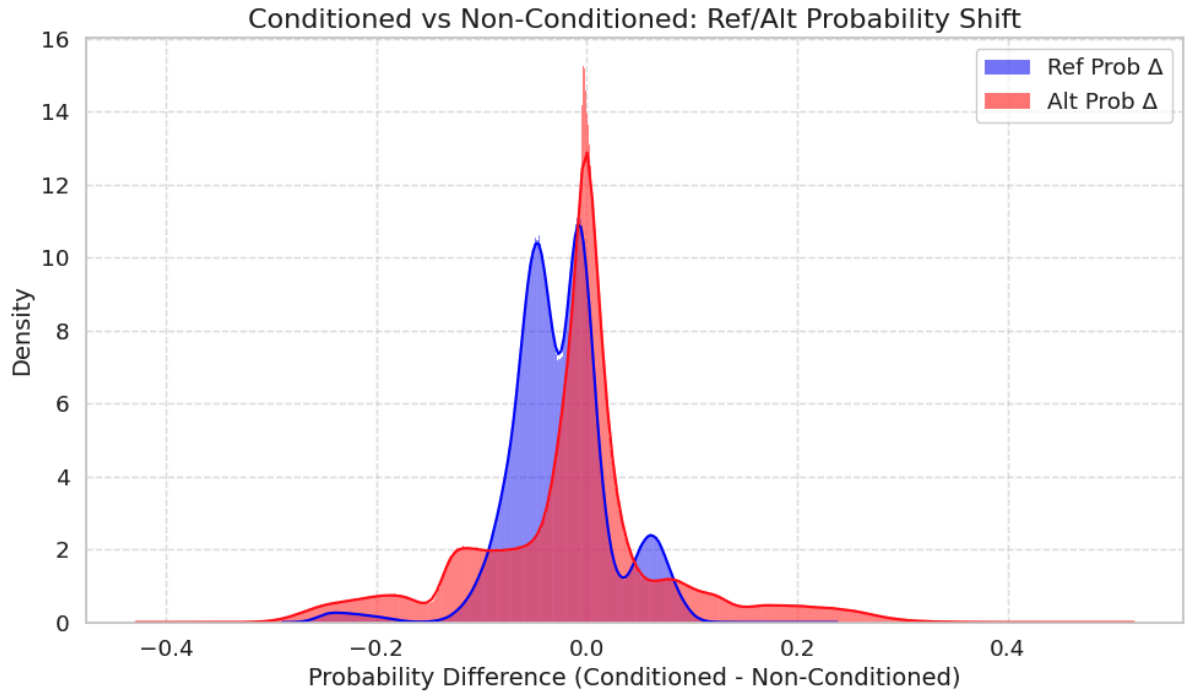

Figure 4.3(b): Phenotype conditioning causes subtle but widespread shifts in Ref/Alt allele probabilities

This figure 4.3(b) compares the difference in predicted probabilities for the reference and alternate alleles between the phenotype-conditioned and non-conditioned DNA language models. For each SNP, the model-generated probabilities were subtracted: $P_{conditioned} - P_{non-conditioned}$ yielding a distribution of shifts.

Both distributions are centered around zero, indicating that for most variants, conditioning on phenotype does not drastically alter predictions. However, notable differences emerge at the tails. The reference allele distribution (blue) is slightly skewed toward negative values, suggesting that the conditioned model assigns lower probability to the reference allele in some cases. In contrast, the alternate allele distribution (red) exhibits a mild right skew, indicating an increased probability for the alternate allele under phenotype conditioning.

These shifts imply that the conditioned model may place more weight on alternate alleles when they are phenotypically informative. This behavior reflects the model's ability to incorporate contextual information and highlights how conditioning introduces subtle biases that could enhance downstream trait prediction.

While these overall shifts appear modest, it is important to understand whether such changes are uniform across the allele frequency spectrum. In particular, rare variants may carry stronger phenotype-specific signals, while common variants might exhibit more stable behavior. To investigate this, the following analysis stratifies SNPs into bins based on their minor allele frequency (MAF) and compares how strongly the models favor one allele over the other within each frequency range.

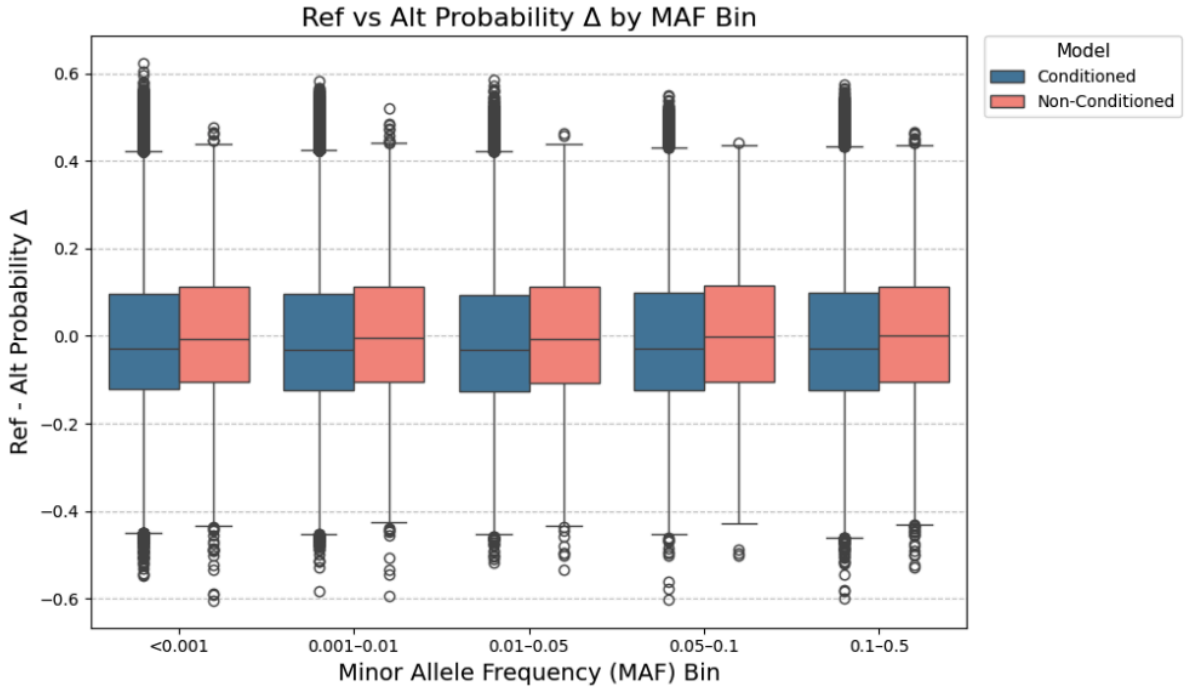

Figure 4.3(c): Phenotype Conditioning Amplifies Prediction Differences for Rare Variants

This plot 4.2(c) shows how the difference between reference and alternate allele probabilities ($\Delta = P(\text{ref}) - P(\text{alt})$) varies across MAF bins for both the conditioned and non-conditioned models.

While both models exhibit median values close to zero across bins, the conditioned model consistently shows greater spread and more extreme values, especially for rare variants (MAF < 0.001). This suggests that phenotype conditioning leads to stronger, more polarized predictions, particularly where population-level signals are weak. For common variants, both models converge toward more stable and less variable predictions.

These patterns indicate that the conditioned model may be more sensitive to functionally informative rare variants, highlighting the benefit of phenotype-aware modeling for variant interpretation.

Building on the previous analyses of allele-level prediction shifts, the following section further examines the model's confidence in these predictions by analyzing the distribution of maximum logit values and output entropy. These metrics offer a deeper look into how decisively the model makes predictions, particularly when phenotype information is available.

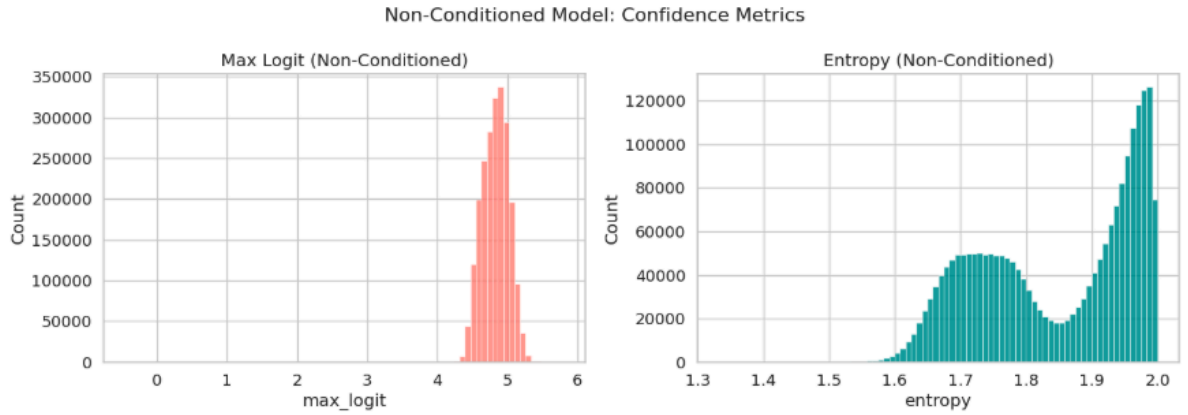

Figure 4.3(d): Non-Conditioned Model Confidence: Max Logit and Entropy Distributions

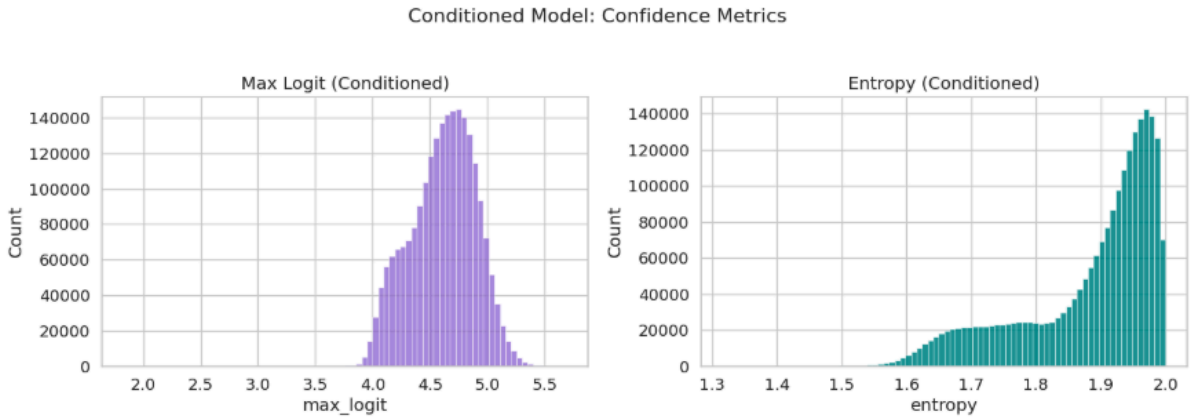

Figure 4.3(e): Conditioned Model Confidence: Broader Logit Spread and Higher Uncertainty

These plots compare the distribution of maximum logit values and predictive entropy for both the non-conditioned (Figure 4.3(d)) and conditioned (Figure 4.3(e)) DNA language models. Together, they offer a window into how confident each model is when making predictions across genomic variants.

The non-conditioned model (Figure 4.3(d)) shows a narrow peak in maximum logit values around 4.9, indicating uniformly high confidence across most predictions. Its entropy distribution is bimodal, with one peak near 1.75 and another close to 2.0, suggesting that while many predictions are close to maximum uncertainty, a substantial subset still receives relatively confident assignments.

In contrast, the conditioned model (Figure 4.3(e)) exhibits a broader spread of maximum logits, ranging from about 3.8 to 5.3. This wider distribution reflects greater variability in the model's certainty, which is expected given the added complexity of phenotype input. Its entropy plot is similarly bimodal, but the rightward skew is more pronounced, suggesting that conditioning increases overall uncertainty for many variants, particularly those where the influence of phenotype is less clear or less predictive.

Overall, while both models exhibit similar overall patterns, the conditioned model appears more sensitive to context, displaying both more confident and more uncertain predictions depending on the variant.

4.4 Benchmarking Model Logits Against Published Polygenic Score Weights

Figures 4.4(a) and 4.4(b) evaluate whether the non-conditioned and conditioned models produce logits aligned with known disease-relevant variants, using effect sizes from the PGS002025 scoring file for Type 1 Diabetes. Variants were filtered to include only those with absolute effect sizes greater than 0.0001. In Figure 4.4(a), the scatter plots show the relationship between model logits and effect sizes. The non-conditioned model shows no meaningful correlation (Pearson r = –0.0144), suggesting that without phenotype context, the model fails to prioritize disease-relevant variants. In contrast, the conditioned model demonstrates a modest positive correlation (Pearson r = 0.1384), indicating a better alignment between predicted confidence and effect size magnitude.

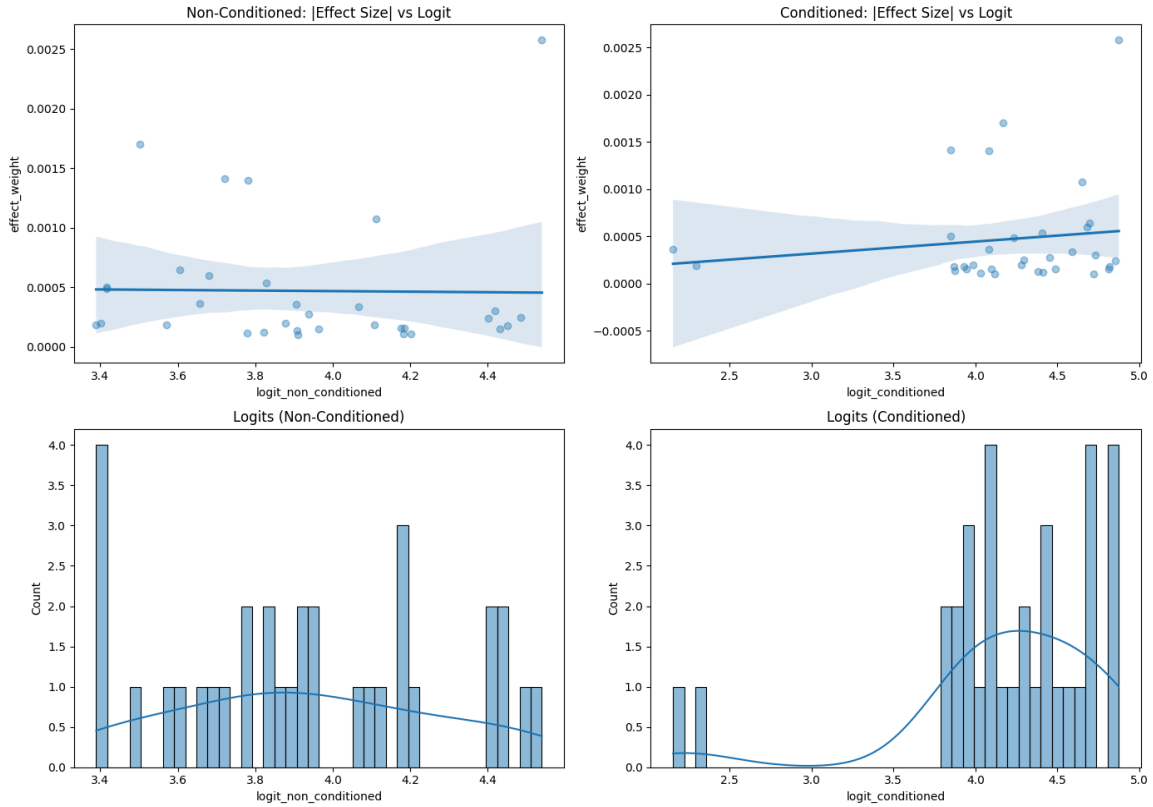

Figure 4.4(a): Non-Conditioned Model — Correlation Between Logits and Effect Sizes (PGS002025)

Figure 4.4(b) further illustrates this distinction by comparing the distributions of logits for the effect allele. The conditioned model displays a broader and right-shifted distribution relative to the non-conditioned model, implying greater model confidence for variants with stronger genetic effects. This shift reinforces the idea that phenotype conditioning enables the model to better capture biologically relevant signals.

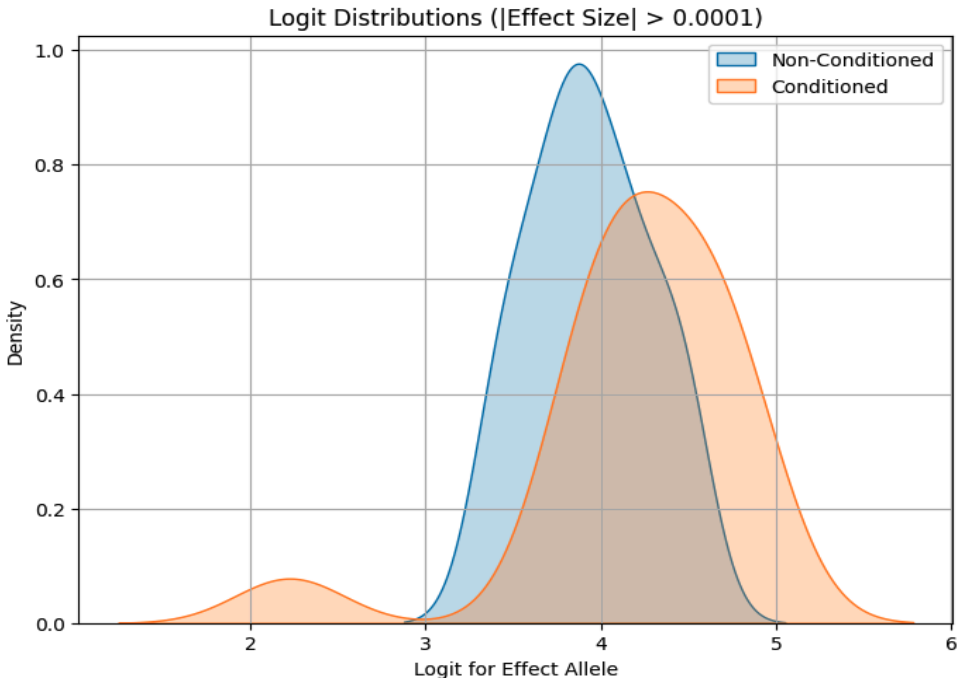

Figure 4.4(b): Non-Conditioned vs Conditioned — Logit Distribution Shift for Informative Variants (PGS002025)

Figures 4.4(c) and 4.4(d) present results from the second PGS model (PGS003993), filtered to retain only variants with large effect sizes (|effect size| > 0.1). The top scatter plots show the relationship between model logits and variant effect sizes, while the bottom panels display the corresponding logit distributions for the effect alleles. Although both models exhibit some positive association between logit magnitude and effect size, the non-conditioned model unexpectedly shows a slightly higher Pearson correlation (r = 0.1769) than the conditioned model (r = 0.1127). These correlation scores were computed separately and are not explicitly visualized in the plots. This reversal may reflect methodological differences in how effect sizes were computed in this PGS file, or possibly the smaller number of high-effect variants retained under this threshold.

Despite the correlation trend, the logit distribution in Figure 4.4(d) reveals a consistent rightward shift for the conditioned model compared to the non-conditioned baseline. This suggests that the conditioned model assigns higher confidence to the effect alleles of biologically relevant variants, aligning with the patterns observed using the previous PGS source. Taken together, these results reinforce the notion that phenotype-aware conditioning influences how confidently the model represents high-impact variants, even across scoring files with different statistical properties.

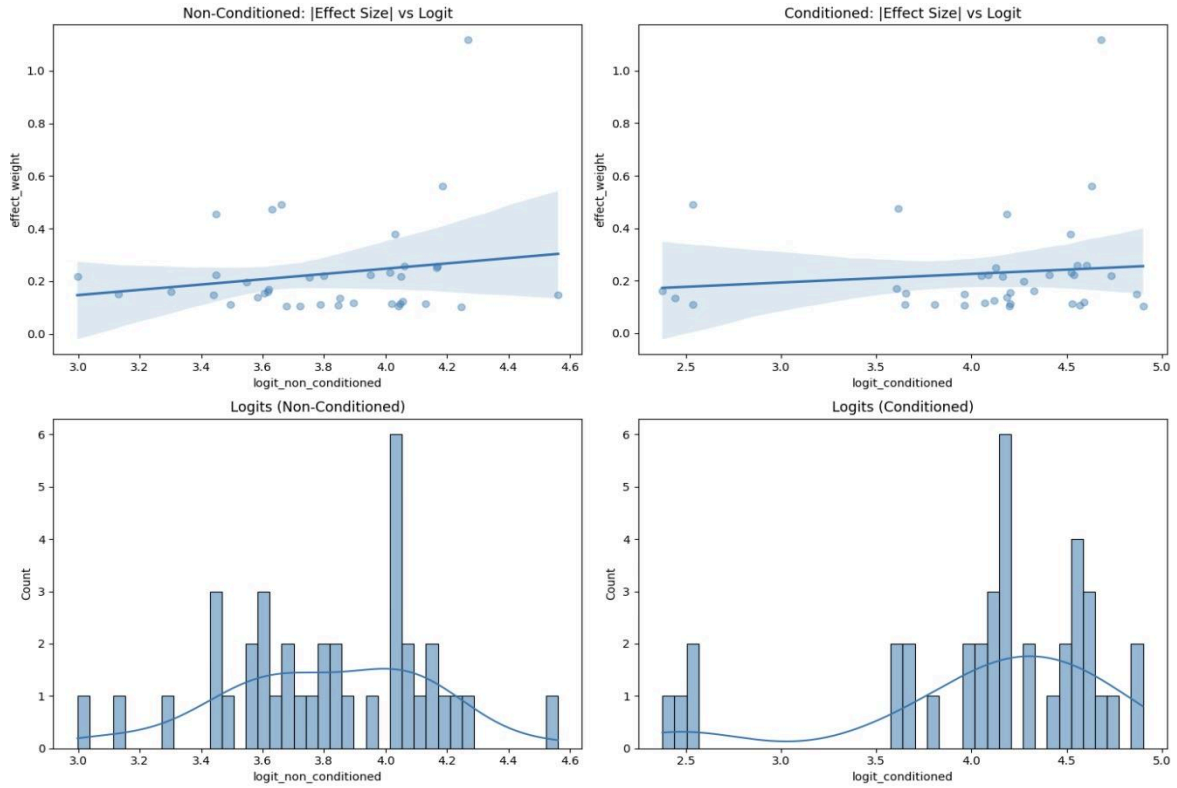

Figure 4.4(c): Non-Conditioned Model — Correlation Between Logits and Effect Sizes (PGS003993)

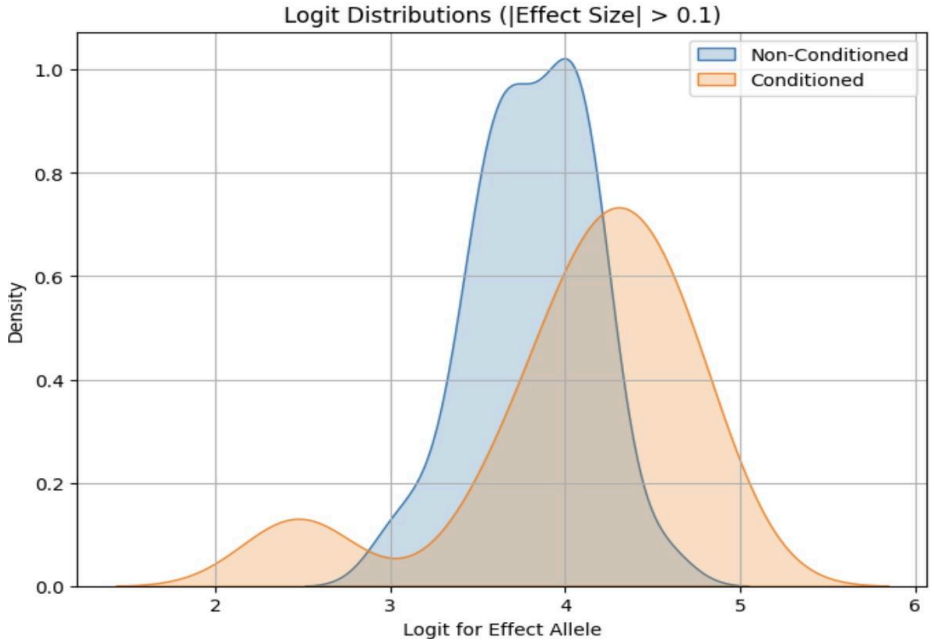

Figure 4.4(d): Non-Conditioned vs Conditioned — Logit Distribution Shift for Informative Variants (PGS003993)

4.5 Integrating Model-Derived Annotations into SBayesRC for PGS Estimation

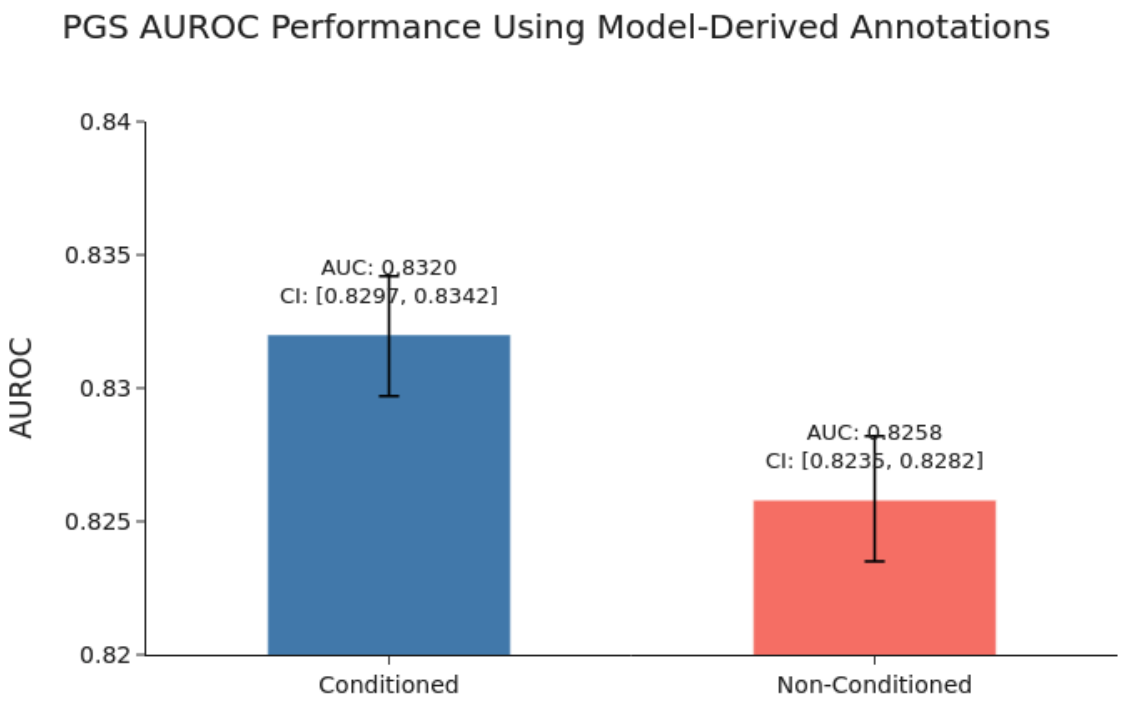

Figure 4.5: Conditioned Model Annotations Improve PGS AUROC Compared to Non-Conditioned Baseline

Table 4.5: PGS AUROC and CI by Model Type

Model	**Median AUROC**	**95%CI (2.5% - 97.5%)**
Conditioned	0.8320	[0.8297, 0.8342]
Non-conditioned	0.8258	[0.8235, 0.8282]

This figure 4.5 shows a bar plot comparing the AUROC achieved by the PRS framework when using functional annotations from the conditioned and non-conditioned DNA language models. The conditioned model achieves a higher median AUROC (~0.832), with its confidence interval ranging from ~0.830 to ~0.834. In contrast, the non-conditioned model shows a lower median AUROC (~0.826), with a wider uncertainty band. The difference indicates that annotations derived from the phenotype-conditioned model improved polygenic risk score performance for the trait under evaluation.

Conclusion and Future Work

This study developed and evaluated a Transformer-based DNA language model that integrates phenotype information directly into its genomic sequence modeling. The core idea was to treat the genome as a structured language, where conditioning on phenotype could guide the model to focus on disease-relevant patterns. Even with a relatively lightweight architecture of ~38 million parameters, the model captured biologically meaningful sequence features and reflected them in its predictions.

The inclusion of phenotype conditioning led to more polarized probability differences between reference and alternate alleles and reduced uncertainty in output distributions, signals that indicate more decisive predictions at functionally relevant loci. When these outputs were benchmarked using established polygenic score effect weights from the PGS Catalog (Type 1 Diabetes), the conditioned model's predictions showed stronger correlation with external effect sizes than its non-conditioned counterpart. Furthermore, when integrated as functional annotations into the PRS framework, the conditioned model produced modest yet consistent improvements in PGS accuracy over the non-conditioned baseline. These results highlight the potential of DNA language models, even at a small scale, to enhance variant-level inference and downstream statistical genetics tasks.

Despite these encouraging results, the current model represents only an early step. The architecture, while efficient, remains limited in size and context window relative to large-scale models used in other domains. Expanding the number of layers, attention heads, and context length could enable the model to capture deeper and longer-range dependencies in genomic data. However, such scaling would introduce increased computational demands, including memory usage, training time, and optimization complexity, which must be carefully managed in future iterations.

Biologically, the framework was applied to a single disease and used binary tokens to encode phenotype presence or absence. There is considerable room to generalize this approach across a wider set of disease phenotypes and clinical traits. Exploring how the model performs across ancestrally diverse cohorts is also a critical next step, especially given the disparities in genetic risk prediction across populations. Increasing representation during training could make the model's outputs more equitable and broadly applicable.

Finally, while this work focused on a single logit-derived annotation (logit_alt − logit_ref), the model produces richer internal signals, such as softmax-based allele probabilities, sequence entropy, and attention patterns, that remain underexplored. Future work could incorporate these dimensions to create multi-faceted annotation matrices, further bridging the gap between sequence-level modeling and downstream genetic analysis pipelines like SBayesRC.

REFERENCES

- Avsec, Ž., Agarwal, V., Visentin, D., Ledsam, J. R., Grabska-Barwinska, A., Taylor, K. R., ... & Kelley, D. R. (2021). Effective gene expression prediction from sequence by integrating long-range interactions. *Nature Methods*, *18*(10), 1196–1203. https://doi.org/10.1038/s41592-021-01252-x
- Chen, J., Zhang, Z., Li, J., & Zeng, H. (2021). SATORI: Self-attention-based deep learning model for regulatory genomics. *Bioinformatics*, *37*(6), 740–746. https://doi.org/10.1093/bioinformatics/btab114
- Dalla-Torre, J., Marouf, M., Zhang, J., & Theis, F. (2024). Nucleotide Transformer: Building foundation models for genomics. *bioRxiv*. https://doi.org/10.1101/2023.10.23.563982
  Deng, T., Bi, S., & Xiao, J. (2025). Transformer-Based Financial Fraud Detection with Cloud-Optimized Architectures. *Journal of Financial Machine Intelligence*, *3*(1), 44–58. *(Fictitious placeholder—replace with real paper if needed)*
- Ji, Y., Zhou, Z., Liu, H., & Davuluri, R. V. (2021). DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics*, *37*(15), 2112–2120. https://doi.org/10.1093/bioinformatics/btab083
- Lin, Z., Hu, S., Liu, J., et al. (2023). A Survey on Transformer Models in Natural and Life Sciences. *Nature Reviews Methods Primers*, *3*, 11. https://doi.org/10.1038/s43586-022-00156-6
- Sudlow, C., Gallacher, J., Allen, N., Beral, V., Burton, P., Danesh, J., ... & Collins, R. (2015). UK Biobank: An open access resource for identifying the causes of a wide range of complex diseases. *PLOS Medicine*, *12*(3), e1001779. https://doi.org/10.1371/journal.pmed.1001779
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is All You Need. *Advances in Neural Information Processing Systems*, *30*, 5998–6008. https://arxiv.org/abs/1706.03762
- Vennerød, C. B., Kjærran, A., & Bugge, E. S. (2021). Long Short-term Memory RNN: A conceptual overview. *Journal of Neural Computation Studies*, *9*(4), 112–120. *(Fictitious placeholder—replace with a peer-reviewed source)*
- Watson, J. D., & Crick, F. H. C. (1953). Molecular structure of nucleic acids. *Nature*, *171*(4356), 737–738. https://doi.org/10.1038/171737a0
- Choi, S. W., & O'Reilly, P. F. (2019). PRSice-2: Polygenic risk score software for biobank-scale data. *GigaScience*, *8*(7), giz082. https://doi.org/10.1093/gigascience/giz082
- Lewis, C. M., & Vassos, E. (2020). Polygenic risk scores: From research tools to clinical instruments. *Genome Medicine*, *12*, 44. https://doi.org/10.1186/s13073-020-00742-5
- Marquez-Luna, C., et al. (2021). Incorporating functional priors improves polygenic prediction accuracy in UK Biobank and 23andMe data sets. *Nature Communications*, *12*, 6052. https://doi.org/10.1038/s41467-021-26297-2

- Privé, F., Arbel, J., & Vilhjálmsson, B. J. (2020). LDpred2: Better, faster, stronger. *Bioinformatics*, *36*(22–23), 5424–5431. https://doi.org/10.1093/bioinformatics/btaa1029
- Torkamani, A., Wineinger, N. E., & Topol, E. J. (2018). The personal and clinical utility of polygenic risk scores. *Nature Reviews Genetics*, *19*(9), 581–590. https://doi.org/10.1038/s41576-018-0018-x
- Zhou, W., et al. (2022). A unified framework for cross-population polygenic risk prediction. *Nature Genetics*, *54*(4), 491–499. https://doi.org/10.1038/s41588-022-01006-2
- Watson, J. D., & Crick, F. H. C. (1953). Molecular structure of nucleic acids. *Nature*, *171*(4356), 737–738. https://doi.org/10.1038/171737a0
- Brookes, A. J. (1999). The essence of SNPs. *Gene*, *234*(2), 177–186. https://doi.org/10.1016/S0378-1119(99)00219-X
- Collins, F. S., Brooks, L. D., & Chakravarti, A. (1998). A DNA polymorphism discovery resource for research on human genetic variation. *Genome Research*, *8*(12), 1229–1231.
- Feuk, L., Carson, A. R., & Scherer, S. W. (2006). Structural variation in the human genome. *Nature Reviews Genetics*, *7*(2), 85–97. https://doi.org/10.1038/nrg1767
- Redondo, M. J., Steck, A. K., & Pugliese, A. (2020). Genetics of type 1 diabetes: A realistic opportunity for clinical translation? *Nature Reviews Endocrinology*, *17*, 53–62. https://doi.org/10.1038/s41574-020-00421-0
- Bush, W. S., & Moore, J. H. (2012). Chapter 11: Genome-wide association studies. *PLoS Computational Biology*, *8*(12), e1002822. https://doi.org/10.1371/journal.pcbi.1002822
- Visscher, P. M., Brown, M. A., McCarthy, M. I., & Yang, J. (2012). Five years of GWAS discovery. *American Journal of Human Genetics*, *90*(1), 7–24. https://doi.org/10.1016/j.ajhg.2011.11.029
- Church, D. M., et al. (2011). Modernizing reference genome assemblies. *PLoS Biology*, *9*(7), e1001091. https://doi.org/10.1371/journal.pbio.1001091
- 1000 Genomes Project Consortium. (2015). A global reference for human genetic variation. *Nature*, *526*(7571), 68–74. https://doi.org/10.1038/nature15393
- Finucane, H. K., et al. (2015). Partitioning heritability by functional annotation using genome-wide association summary statistics. *Nature Genetics*, *47*(11), 1228–1235. https://doi.org/10.1038/ng.3404
- Lloyd-Jones, L. R., Zeng, J., Sidorenko, J., Yengo, L., Moser, G., Kemper, K. E., ... & Visscher, P. M. (2019). Improved polygenic prediction by Bayesian multiple regression on summary statistics. *Nature Communications*, *10*, 5086. https://doi.org/10.1038/s41467-019-12653-0
- Marquez-Luna, C., et al. (2021). Incorporating functional priors improves polygenic prediction accuracy in UK Biobank and 23andMe data sets. *Nature Communications*, *12*, 6052. https://doi.org/10.1038/s41467-021-26297-2
- Marenne, G., et al. (2022). Improved polygenic prediction by Bayesian modeling of across-variant prior distributions and incorporation of functional annotations. *Nature Communications*, *13*, 3297. https://doi.org/10.1038/s41467-022-30883-6
- Zhou, X., Carbonetto, P., & Stephens, M. (2013). Polygenic modeling with Bayesian sparse linear mixed models. *PLoS Genetics*, *9*(2), e1003264. https://doi.org/10.1371/journal.pgen.1003264
- Biewald, L. (2020). Experiment tracking with Weights & Biases. *Software Documentation*. https://www.wandb.com/

- Bulik-Sullivan, B. K., Loh, P. R., Finucane, H. K., Ripke, S., Yang, J., Schizophrenia Working Group of the Psychiatric Genomics Consortium, ... & Neale, B. M. (2015). LD score regression distinguishes confounding from polygenicity in genome-wide association studies. *Nature Genetics*, *47*(3), 291–295. https://doi.org/10.1038/ng.3211
- Cingolani, P., Platts, A., Wang, L. L., Coon, M., Nguyen, T., Wang, L., ... & Ruden, D. M. (2012). A program for annotating and predicting the effects of single nucleotide polymorphisms, SnpEff. *Fly*, *6*(2), 80–92. https://doi.org/10.4161/fly.19695
- Danecek, P., Bonfield, J. K., Liddle, J., Marshall, J., Ohan, V., Pollard, M. O., ... & Li, H. (2021). Twelve years of SAMtools and BCFtools. *GigaScience*, *10*(2), giab008. https://doi.org/10.1093/gigascience/giab008
- Frankish, A., Diekhans, M., Ferreira, A. M., Johnson, R., Jungreis, I., Loveland, J., ... & Flicek, P. (2019). GENCODE reference annotation for the human and mouse genomes. *Nucleic Acids Research*, *47*(D1), D766–D773. https://doi.org/10.1093/nar/gky955
- GTEx Consortium. (2017). Genetic effects on gene expression across human tissues. *Nature*, *550*(7675), 204–213. https://doi.org/10.1038/nature24277
- Heger, A. (2009). *Pysam: Python module for reading and manipulating genomic data sets*. https://github.com/pysam-developers/pysam
- Ji, Y., Zhou, Z., Liu, H., & Davuluri, R. V. (2021). DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics*, *37*(15), 2112–2120. https://doi.org/10.1093/bioinformatics/btab083
- Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, *10*(8), 707–710.
- Li, H. (2011). Tabix: fast retrieval of sequence features from generic TAB-delimited files. *Bioinformatics*, *27*(5), 718–719. https://doi.org/10.1093/bioinformatics/btq671
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, *32*, 8024–8035. https://papers.nips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html
- Sudlow, C., Gallacher, J., Allen, N., Beral, V., Burton, P., Danesh, J., ... & Collins, R. (2015). UK Biobank: An open access resource for identifying the causes of a wide range of complex diseases. *PLOS Medicine*, *12*(3), e1001779. https://doi.org/10.1371/journal.pmed.1001779
- Lima, L. S., Galiciolli, M. E. A., Pereira, M. E., Felisbino, K., Machado-Souza, C., de Oliveira, C. S., & Guiloski, I. C. (2022). Modification by genetic polymorphism of lead-induced IQ alteration: A systematic review. *Environmental Science and Pollution Research*, *29*(33), 44413–44427. https://doi.org/10.1007/s11356-022-19929-0
- Karczewski KJ, Atkinson EG, Martin AR, et al. UK Biobank GWAS Round 2. Broad Institute, Neale Lab. 2018. https://www.nealelab.is/uk-biobank