UNIVERSITY OF CALIFORNIA
Los Angeles

Computational methods for leveraging multiple biodata resources

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Dat Duong

2020

ABSTRACT OF THE DISSERTATION

Computational methods for leveraging multiple biodata resources

by

Dat Duong
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2020
Professor Kai-Wei Chang, Co-chair
Professor Eleazar Eskin, Co-chair

With the advancement of biotechnology, there have been many datasets collected for bioinformatic research. These datasets capture different biological aspects but are closely related. For example, the Genotype-Tissue Expression data and the Roadmap datasets are complementary. The first provides the relationships between genes and genotypes in a tissue, and the latter identifies important genomic regions. Together, both datasets allow us to better understand how genes are regulated in a tissue.

This dissertation presents methods to jointly analyze different data resources. We aim to capture holistic views of the biological problems. First, we study the problem of identifying genes having significant expression levels with respect to the genotypes. We build a statistical model to combine the information from the Genotype-Tissue Expression data and the Roadmap datasets. Second, we study the problem of predicting protein functions. We design a deep learning model that leverages the Gene Ontology, key amino acid motifs, the protein structures, and the protein-protein interaction network.

The dissertation of Dat Duong is approved.

Jingyi Jessica Li

Sriram Sankararaman

Jason Ernst

Kai-Wei Chang, Committee Co-chair

Eleazar Eskin, Committee Co-chair

University of California, Los Angeles

2020

Dedicated to my mother, my wife, and my daughter.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

VITA

2006 – 2010	B.S Molecular Cell Developmental Biology & B.S Applied Mathematics, University of California, Los Angeles, USA
2010 – 2011	Undergraduate Research Assistant, Computer Science, University of California, Los Angeles, USA
2011 – 2013	M.S. Statistics, University of California, Berkeley, USA
2013 – 2014	Statistician, Affinnova Inc., Waltham, Massachusetts, USA
2017	Research Intern, Supply Chain Optimization Amazon Inc., Seattle, Washington, USA
2018	Research Intern, Supply Chain Optimization Amazon Inc., Seattle, Washington, USA
2014 – present	Graduate Student, Computer Science, University of California, Los Angeles, USA

PUBLICATIONS

Dat Duong, Lisa Gai, Ankith Uppunda, Don Le, Eleazar Eskin, Jingyi Jessica Li, Kai-Wei Chang. Annotating Gene Ontology terms for protein sequences with the Transformer model. *bioRxiv*, 2020. pre-print.

Dat Duong, Ankith Uppunda, Lisa Gai, Chelsea Jui-Ting Ju, James Zhang, Muhao Chen, Eleazar Eskin, Jingyi Jessica Li, Kai-Wei Chang. Evaluating Representations for Gene Ontology Terms.

bioRxiv, 2020. pre-print.

Serghei Mangul, Thiago Mosqueiro, Richard J. Abdill, **Dat Duong**, Keith Mitchell, Varuni Sarwal, Brian Hill, Jaqueline Brito, Russell Jared Littman, Benjamin Statz, Angela Ka-Mei Lam, Gargi Dayama, Laura Grieneisen, Lana S. Martin, Jonathan Flint, Eleazar Eskin, Ran Blekhman. Challenges and recommendations to improve the installability and archival stability of omics computational tools. PLoS biology. 2019 Jun 20;17(6):e3000333.

Dat Duong, Wasi Uddin Ahmad, Eleazar Eskin, Kai-Wei Chang, Jingyi Jessica Li. Word and sentence embedding tools to measure semantic similarity of gene ontology terms by their definitions. Journal of Computational Biology. 2019 Jan 1;26(1):38-52.

Dat Duong, Lisa Gai, Sagi Snir, Eun Yong Kang, Buhm Han, Jae Hoon Sul, Eleazar Eskin. Applying meta-analysis to genotype-tissue expression data from multiple tissues to identify eQTLs and increase the number of eGenes. Bioinformatics. 2017 Jul 15;33(14):i67-74.

Dat Duong, Jennifer Zou, Farhad Hormozdiari, Jae Hoon Sul, Jason Ernst, Buhm Han, Eleazar Eskin. Using genomic annotations increases statistical power to detect eGenes. Bioinformatics. 2016 Jun 15;32(12):i156-63.

Buhm Han, **Dat Duong**, Jae Hoon Sul, Paul I. W. de Bakker, Eleazar Eskin, Soumya Raychaudhuri. A general framework for meta-analyzing dependent studies with overlapping subjects in association mapping. Human molecular genetics. 2016 May 1;25(9):1857-66.

Gregory Darnell*, **Dat Duong***, Buhm Han, Eleazar Eskin. Incorporating prior information into association studies. Bioinformatics. 2012 Jun 15;28(12):i147-53.

CHAPTER 1

Research scope and contribution

Since 2010, there has been an ample amount of data collected for bioinformatic research. For example, in 2010, we had the public release of the Encyclopedia of DNA Elements (ENCODE), a database of all functional elements in the human genome [18]. In 2015, we had the first pilot analysis of the Genotype-Tissue Expression (GTEx) data [98]. In 2018, the Uniprot database contained 30% more manually annotated protein functions in Human compared to 2016 [96]. As the rates of data generation and annotation are rapidly growing, there is a dire need for statistical and deep learning models that will aid biologists in mining useful information and in many instances, predicting future outcomes from the collected datasets (e.g. inferring functions of new proteins). Because biological datasets are often related in many ways, we can jointly analyze them to deduce more holistic explanation or to obtain better prediction.

Consider this long-standing problem in biology: the study of how genes affect a phenotype. In the early 2000s, to study this problem, we could only collect genetic data and perform genome wide association study (GWAS) to find the genetic variants (SNPs) correlated with this phenotype. GWAS usually find significant SNPs located in non-coding regions of the genome, and so GWAS could not always infer the candidate genes for the phenotype [33, 39]. Today thanks to the GTEx data, we can retrieve genes whose expressions are correlated with some genetic variants. Suppose these variants are also GWAS significant SNPs, then the retrieved genes are likely to be the determinants of the phenotype of interest.

The problem of inferring protein functions is another key bioinformatic problem that has been greatly benefited from the recent technological advancement. In the early 2000s, many approaches rely on string-matching tools to compare the unknown protein against the database of labeled

sequences [54]. In the recent years, we can leverage other datasets like the 3D protein structures and protein-protein interaction network which have greatly accrued since their introductions in the early 2000s [10, 95]. Previous works have presented ensemble models where each protein function classifier was built using separate data resources [107]. Recently, we can develop large deep learning models to capture all the complex interactions of all the various data resources [29, 59]. Also, there are useful information in the labels used to annotate the protein functions which can be leveraged to improve the classification accuracy. For example, assume we can model how often two similar labels should co-occur. If one of the labels occurs very rarely, then we can still make a good prediction for it by leveraging the prediction accuracy of the other label.

This thesis presents methods for combining multiple data resources to solve sub-problems of the two major bioinformatic problems, learning how genes affect phenotypes and predicting protein functions. The key initial step for studying the effect of genes on phenotypes requires us to discover which genes are expressed in which tissues, because the expression levels of the same gene often vary for different tissues. For example, the gene Apolipoprotein E involved in fat metabolism is expressed only in the adrenal gland, liver, and brain tissues [98]. Scientists can now measure the expression levels of genes in many human body tissues, but a statistical metric is still needed to specify which genes should be prioritized for further causal analysis of the phenotypes. Statistical analyses on gene expression datasets are interested in associating the expression of a gene with Single Nucleotide Polymorphisms (SNPs). SNPs are positions in the DNA that are different for each person in a population. Because not everyone shares the same set of SNPs, genes having expressions associated with the SNPs are possible determinants for the differential phenotypic expression in a population. The literature uses the name eGenes to label the genes whose expressions are correlated with at least one SNP [98].

In this thesis, we consider two key challenges in the problem of discovering eGenes. First, when associating a gene expression with the SNPs, the locations of the SNPs can have strong impact on the outcome. For example, SNPs near transcription start sites are likely to affect the binding affinity of transcription factors which in turn influences the expression level [97, 99]. Yet, existing methods have not yet modeled this effect. Second, despite the declining cost of gene sequencing,

it is still difficult to collect samples to measure gene expression in each tissue. At the time of our work in 2016, the liver tissue in the Gene-Tissue Expression data had 97 individuals (in 2020, this number is 226 which is low compared to other genome wide association studies) [36, 98]. There was a need to develop a method that aggregates data from many tissues to enhance the statistical power for discovering eGenes [94]. In this case, we can not classify the tissues in which the genes are eGenes. However, we still can determine which genes are eGenes for the entire dataset, so that we do not miss possible causal genes when studying the phenotypes. Section 1.1 further describes our search scope and innovation.

To discover protein functions, laboratory experiments are needed to view the protein 3D structures, locate the proteins in the cell, and identify the biological pathways involving the proteins. The discovered protein functions are then written down according to the Gene Ontology (GO) guideline which is a standardized language designed to describe biological functions. For example, the GO label GO:0008218 with the definition *bioluminescence* describes the function for the protein Aequorin-2 in jellyfish. With the decline of sequencing cost, the gap between the numbers of labeled and unlabeled proteins is expected to grow [107]. In the past decade, many automated annotation methods have been introduced to aid the manual validation. In these classifiers, the training data is the Uniprot database which contains protein amino acid sequences and their curated GO labels [96].

Publications from other fields like natural language processing, image recognition, social network analysis, and product recommendation have found that including metadata about the labels into the classifier often improve the prediction accuracy [34, 56, 103]. Very few annotation methods are using metadata about the GO labels. These methods either apply the hierarchical relationships of the labels as a post-processing step [73], or model these relationships together with the protein-protein interaction network [102] (ignoring the sequence information). Using a well established protein-protein interaction network to build an annotation model for new proteins is circular-reasoning, because we will not know the interacting partners of a new protein without the laboratory work. In the current literature, it is unclear how to design a classifier that takes just the protein sequences and the metadata of the GO labels such as their definitions, hierarchy relations,

and co-occurrence frequencies. In this thesis, we introduce and evaluate various ways to create feature vectors for the GO label metadata. Next, we build a deep learning classifier that takes the protein amino acid sequences, the feature vectors of the GO labels, and other information such as motif data, protein 3D structures and protein-protein interaction network. Section 1.2 further explains our research scope and contribution.

1.1 Association study to discover eGenes

The transition from genetic information to a phenotype starts with the DNA, to RNA, and then to proteins. These proteins will then involve in complex biological processes that induce the phenotype of interest. For this reason, the differential phenotypic expression in a population is assumed to start with the variations in the genetic materials. Single Nucleotide Polymorphisms (SNPs) are the locations in the DNA where people in a population have different nucleotides. In an individual, the expression level of a gene is measured by the RNA amount being transcribed. Values of SNPs can affect the production of RNA, by altering the binding affinity of transcription factors and the compactness of the DNA strand [74, 97, 99]. Having very low amount of RNA implies having almost none of the protein produced from this gene. Because proteins interact with each other in complex chemical reactions to produce the observed traits, having too much or not enough of a specific protein can cause the differential expression for the phenotype of interest. In the early days of GWAS, the objective was to identify which SNPs in the genome are strongly associated with the phenotypes (e.g. disease status encoded as 0 and 1). GWAS experiments cannot identify the mechanisms (e.g. possible causal genes) for the phenotypes, because its approach bypasses all the intermediate phases that connect the SNPs to the phenotypes (e.g. transcription and translation). For this reason, heuristically, genomic regions containing these SNPs are assumed to contain key targets for further causal analysis of the phenotypes. Often times, genes located near these significant regions are prioritized to be analyzed first [14].

Within the past 5 years, it has became more affordable and easier to measure the expression level of a gene in specific body tissues. These gene-tissue expression datasets let us identify

genes having expressions associated with the SNPs, and thus provide finer lists of genes to be prioritized than traditional GWAS results [50, 91, 98]. SNPs associated with gene expressions are known as expression quantitative trait loci (eQTLs), and the association experiments to discover these SNPs are called eQTL studies. eQTL studies follow the same principle of GWAS (and conceptually they are the same as GWAS), except that we have gene expression level as the observed dependent variables instead of phenotypes. As byproducts, eQTL studies also discover genes whose expressions in a tissue are associated with at least one SNP. In other words, this type of genes are the genes that have at least one eQTL, and the name eGenes are used to label this specific type of genes [98]. Because everyone has an unique set of SNPs, the eGenes are likely to be the key factors causing the differential phenotypic expression in a population. Importantly, with successful experiments involving CRISPR/Cas9 gene editing in the past decade, today it is essential to discover all the eGenes in each body tissue so that we can target all the possible genes affecting a wide variety of traits [52, 63]. Since 2015, the GTEx consortium has been maintaining an eGene database for every human body tissue [98]. Below we outline our contributions on the discovery of eGenes.

- **Model importance of SNPs to discover eGenes.** During the pilot analysis phase in 2015, the GTEx consortium started a database classifying the eGenes in each of the human body tissues. The method employed are the eQTL studies that follow the same principle as GWAS experiments. Conditioned on a tissue, the expression of a gene is tested against each SNP in the genome. If the gene expression is associated with any SNP, then the gene is considered to be an eGene in this tissue. eGenes are then prioritized for further experiments involving the phenotypes observed in the tissue. Because there are about 4 to 5 million SNPs in the genome, the GTEx consortium tested only cis-SNPs which are SNPs located near a gene [98]. There are other reasons to consider only the cis-SNPs of a given gene. Cis-SNPs are likely to be found near gene transcription start site or other functionally important regions such as DNase hypersensitive sites and histone modification sites[1]. In both instances, cis-SNPs may

[1]We are referring to locations in the gene where certain modifications are likely to happen to the histones found at those locations.

affect the functions of transcription factors or the compactness of the DNA, and therefore will affect the gene expression.

Although the GTEx consortium focused on the cis-SNPs for their eQTL studies, the consortium did not model the specific locations of these SNPs; for example, cis-SNPs in a transcription start site were treated the same as those outside of this site. Our previous GWAS experiments indicated that weighing the SNPs based on their locations increases the statistical power for discovering significant SNPs of a phenotype [20, 33]. For eQTL studies, we reason that if we apply external genomic annotations to assess the importance of cis-SNPs, then we can also increase the statistical power for finding eQTLs of genes.

Because discovering eGenes requires us to find the eQTLs of the genes, increasing statistical power of eQTL studies will improve the power of discovering eGenes. This outcome will retrieve more possible causal genes for a specific phenotype. In our work, we consider the most intuitive region type which is the gene transcription start site, and then two more important region types which are the DNase and histone binding sites. Integrating these region types into the standard eQTL studies discovers at least 16% more eGenes in the GTEx data of the liver tissue. Chapter 2 further explains a few other key challenges that we improve over the traditional eQTL studies.

- **Meta-analysis model to discover eGenes.** A gene does not always have the same expression level in every tissue, and thus the definition of an eGene makes the most sense when we condition the gene expression on a specific tissue. In an effort to identify all the eGenes, in 2015 the GTEx consortium applied eQTL studies on the expression of every gene in each of the 44 human body tissues in their datasets. At the time, many tissues contain only a few samples; for example, the median number of samples per tissue was 126 individuals, with the highest and lowest being Skeletal Muscle (361 people) and Uterus (70 people). When there are not enough samples in a tissue, the eQTL studies may not accurately estimate the association strengths between the cis-SNPs and the gene expression. The reduced performance of the eQTL studies can lead to a loss of statistical power for discovering eGenes.

  Previous GWAS meta-analyses have indicated that combining many data resources can more

accurately measure the association strengths between the SNPs and a phenotype [41, 42, 45]. Intuitively, by joining many datasets and then building a statistical model on this combined dataset, we are analyzing a new data source that has a much larger sample size which will in turn improve the statistical power. Because eQTL studies follow the same principle as GWAS, we hypothesize that a joint analysis of many tissue datasets would provide a higher statistical power to identify eQTLs. This joint analysis can only conclude that a SNP is an eQTL for a gene in at least one tissue of the entire GTEx dataset. In this case, the definition of an eGene makes sense only with respect to the entire datasets, and not for one single tissue. In other words, we can only determine that a gene is an eGene for at least one tissue in the whole GTEx data. Despite not knowing the specific tissues in which a gene is an eGene, at the very least, we can still discover many genes that might be responsible for a phenotype. For example, in 2016, our method detected the CABLES1 gene to be an eGene in the GTEx data. In 2017, CABLES1 was found to be an important gene for Cushing's disease, which is a condition affecting the anterior pituitary in the brain [46]. Suppose we had not used our approach, then we would have missed that CABLES1 is an important gene and would not have suggested CABLES1 for causal analyses of any diseases.

There are a few ways to combine small datasets into a larger dataset. The most computationally efficient approach is meta-analysis which averages the outcome of each smaller dataset. In the context of eQTLs, meta-analysis averages the effect size of the same SNP on the expression of the same gene in each tissue data. Suppose this average is statistically different from zero, then the SNP is an eQTL for the gene in at least one tissue. When we evaluate just one SNP against the gene expression, then the meta-analysis outcome of this SNP allows us to conclude that the gene is an eGene in at least one tissue. The traditional meta-analysis described here does not consider one key biological evidence: the GTEx pilot study reported a high number of shared eQTLs for every pair of tissues, for instance, more than 50% of eQTLs are common for the thyroid, nerve, skin, adipose, heart, artery, lung, blood, and muscles tissue [98].

Based on this observation, we design an new meta-analysis for the GTEx data. We model the fact that the same SNP can be eQTL for the same gene in several tissues; statistically

speaking, we introduce a new parameter into the meta-analysis that captures the correlations of effect sizes for the same SNP on the expression of the same gene in different tissues. We also observe that the GTEx consortium collect many tissue samples from a same donor; thus, many tissue datasets share the same information that came from the same donors. Our previous work in GWAS meta-analysis have indicated that sample sharing among the datasets inflates the false-positive rate of the study [45]. For this reason, in our eQTL meta-analyses, we also design a heuristic correction method to remove the unwanted effect of sample sharing among the tissues in the GTEx data. Next, we explain how to combine the eQTL meta-analyses of the cis-SNP for a gene over all the tissues and determine whether this gene is an eGene for at least one of the tissue in the GTEx data. Chapter 3 further explains other challenges that must be overcome, and our solution.

1.2 Deep learning model to predict protein functions

Laboratory experiments are required to learn the functions of a new protein. The functions of this new protein are written down based on the rules provided by the Gene Ontology (GO) which is a database of standardized language designed to explain the protein's biological processes, molecular functions, and cellular components [38]. Then, the amino acid sequence and the GO labels of this new protein are submitted to the Uniprot database which, as of March 2020, contains 561,911 proteins with their manually reviewed GO terms. The GO database acts like a dictionary, in the sense that it contains many GO terms, and each term has a few sentences describing some biological events. For example, the terms GO:0035556, GO:0004672, and GO:0005634 have the following descriptions *intracellular signal transduction*, *protein kinase activity* and *nucleus*, which are used to describe the biological process, molecular function, and cellular component of the protein Mapkapk5 [12].

With the cost of sequencing dropping, it is expected that the number of unlabeled protein sequences will continue to rise much faster than the number of labeled sequences. In the last decade, there has been great effort in building automated method to aid the discovery of protein

functions. Early annotation methods relied on BLAST, that is, they find the sequences in Uniprot that are most similar to the unknown sequence and then assign the GO labels of these known sequences to the unknown protein [54]. As data accrue over the last decade, along with the protein sequences, many annotation methods also use other metadata about the proteins such as their functional motifs, 3D structures and interaction network [107]. In the last 5 years, with the advancement of computing power, GO annotation methods began to implement deep learning architectures with the hope to capture complex interactions of the amino acids in the sequence [59, 60]. Deep learning has the following intrinsic property; it transforms closely related sequences into similar vector representations, so that these sequences would have equivalent sets of GO terms.

From the current research literature, three main questions arise. First, besides protein metadata, is there useful information from the Gene Ontology that can improve the prediction accuracy? Second, BLAST-based methods have proven to be very competitive against deep learning approaches [29, 59]. We want to know: is there information in a protein sequence that BLAST captures but the neural network cannot? Here, we will not build an ensemble method that averages the outcomes of BLAST approaches and deep learning. Rather, we want to integrate the key information retrieved by BLAST as feature inputs for deep learning. Third, despite the wish for deep learning to capture long-range interactions of amino acids in the sequence, recent works found that recurrent neural network, e.g. Long Short-term Memory (LSTM), fails to produce results better than the convolutional neural network (CNN) which is designed to capture local interactions [59]. There has not been any work exploring neural network with attention mechanism [59]. The third question is: can we build an attention-based neural network classifier that captures the long-range interactions of amino acids in a protein sequence? For example, can such model capture the interactions of far-apart motifs within the same protein, and how does such model compare against BLAST and CNN approaches? Below, we outline our contributions to these three research questions.

- **Learning embeddings of GO labels.** The Uniprot database acts as the training dataset for many GO annotation methods. In Uniprot, each protein sequence is assigned a list of GO labels, and so the problem of predicting protein functions is a multi-label classification problem. The input is an amino acid sequence, and the output is a 1-hot label vector. Each

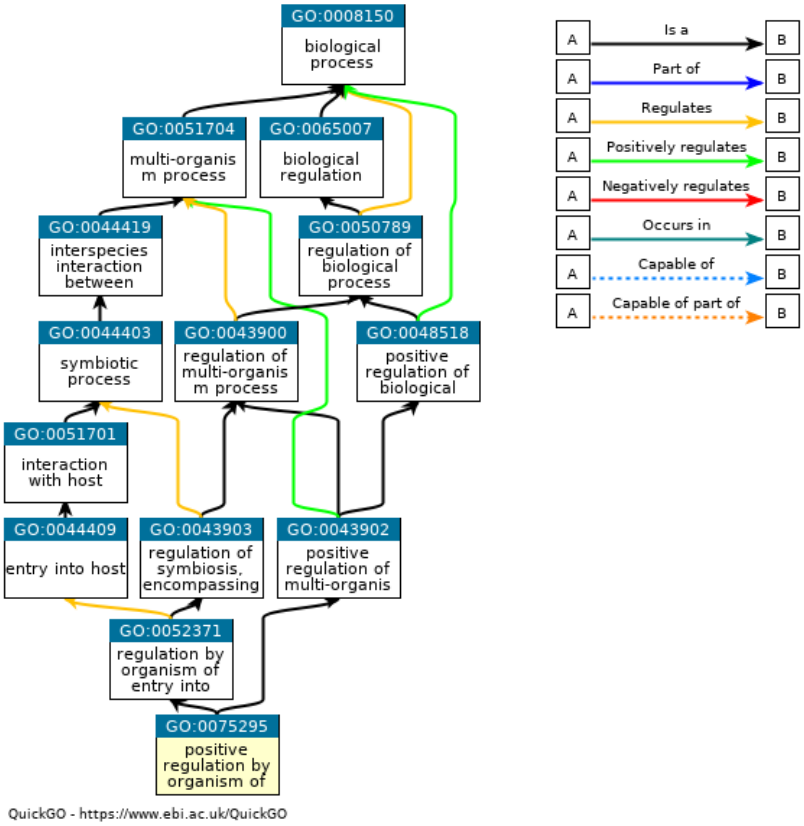


Figure 1.1: GO:0075295 and its ancestors.

i^{th} entry of this vector is a 0 or 1 to indicate absence and presence of the i^{th} GO label for the input sequence. Because the labels are represented by numbers (e.g. 0 or 1), we lose the information that each label entails. For example, in the 1-hot label vector, when we represent the GO term GO:0035556 as a number, we no longer know that this label indicates *intracellular signal transduction* or that the label is a subclass of GO:0050794 *regulation of cellular process*. Other machine learning literature involving multi-label prediction suggested that metadata of the labels, such as the word descriptions and hierarchical relationships, can improve the classification accuracy [7, 72, 81]. It is then important to ask: are there similar types of information from the Gene Ontology that can improve the prediction accuracy for protein functions?

We observe that the GO database contains the word descriptions for each term, and that these terms are arranged in a hierarchy where terms describing very specific functions are child nodes of broader terms (Fig. 1.1). Before we can integrate metadata of GO labels in

a deep learning model, we have to represent this information as numerical feature vectors. For example, in Fig. 1.1, we would want that the vectors representing GO:0075295 (yellow node) and its parent GO:0052371 to be similar, and that these two vectors to be different from the vector for root node GO:0008150.

In the machine learning literature, representing entities as vectors is referred to as *embedding the entities*, and the output vectors are known as *embeddings*. There are many machine learning models to embed entities (e.g. documents in a database, people in social network) into vectors, and these embedding techniques have been applied to a wide variety of resources such as Wikipedia documents, social network, electronic health records, and Pubmed data [15, 31, 51, 62, 75]. However, embedding approaches have not yet been designed for the GO database and thoroughly compared.

We create different types of GO label embeddings from the following neural network architectures: Bidirectional Long-short Term Memory (BiLSTM), Embeddings from Language Models (ELMo), Bidirectional Encoder Representations from Transformers (BERT), and Graph Convolution Network (GCN) [17, 25, 55, 78]. We will describe these methods in detail in Chapter 4. For now, we will just provide high-level descriptions for them. BiLSTM, ELMo, and BERT transform the definitions of GO labels (e.g. *intracellular signal transduction*) into vectors. BiLSTM is a bidirectional encoder that learns the relationship of words in a sequential order, that is, the current word receives the information from the word immediately before it and after it. In the example *intracellular signal transduction*, the word *signal* will receive information from the word *intracellular* first and then the word *transduction* second. ELMo has two layers of BiLSTM where the output of the first BiLSTM is the input of the second BiLSTM, effectively ELMo builds a deeper network of BiLSTM. BERT is an approach to train the Transformer neural network (to be explained later). Unlike BiLSTM and ELMo, BERT models all the pairwise interactions for the words in a definition of a GO term. In our running example, *signal* will receive information from *intracellular* and *transduction* at the same time. GCN does not model definitions of GO terms, rather it transforms the GO labels into vectors based on their topologies in the GO database (e.g.

parent-child nodes will have similar embeddings).

To compare the embedding methods, we design three tasks. Task 1 studies the edge cases where the embeddings may fail to accurately resemble similarity for related GO labels (e.g. parent-child nodes). Task 2 uses the embeddings to compare the functions of orthologous genes and then interacting proteins. Orthologous genes are expected to have closely related functions. Under a good embedding model, sets of GO labels annotating orthologs should have higher similarity scores than the sets annotating unrelated genes (the same argument applies to interacting proteins) [70]. Task 3 evaluates two scenarios in which the GO label embeddings may affect the accuracy of an existing GO annotation method. First, we consider the case where there are ample amounts of data for each GO label in the training and testing datasets. Second, we consider the setting of zeroshot learning, where we predict GO labels that are unobserved in the training data [87].

- **GO annotation by the Transformer neural network.** Many GO annotation methods based on deep learning are recent and share one key idea. They treat each amino acid in the sequence as a vector, and then model the interactions of these vectors by applying CNN or LSTM. CNN captures local interactions, whereas LSTM is able to learn long-range interactions. For protein sequences, long-range interactions among amino acids exist because proteins are folded compactly and can have multiple functioning regions [58]. We would expect that LSTM to outperform CNN. Surprisingly, LSTM is in fact worst than CNN for protein sequences [59], and this same observation have been noticed in similar cases for DNA sequences [5]. Moreover, BLAST-based methods have proven to be very competitive against deep learning approaches [29, 59] In Chapter 5, we address the remaining two research questions: (1) what information in the sequences are extracted by BLAST but the neural network cannot capture, and (2) how to design a deep learning method that sufficiently models the long-range interactions of amino acids in the protein sequences?

  First, we build a new attention-based deep learning GO classifier based on the Transformer neural network [101]. Transformer models all pairwise interactions of amino acids in a sequence and can capture long-range relationships better than LSTM. LSTM may not always

faithfully transmit information from one amino acid to another far way amino acid (a problem known as vanishing gradient [47]). Unlike the other classifiers which treat the GO labels as 1-hot label vector, our classifier uses the GO label embeddings to make predictions, allowing it to learn the label co-occurrences. Second, we observe that the main principle of all BLAST-based models is applying BLAST as a way to identify sequences containing similar amino acid patterns. To combine BLAST with deep learning, we model each type of motifs found by BLAST as a vector feature input for our classifier. Unlike our approach, other works combined BLAST with deep learning via ensemble idea, by averaging outcomes of many independent models. Lastly, we also integrate in our classifier the protein metadata, such as protein-protein interaction network and protein 3D structure information.

1.3 Overview

The dissertation is organized as follows. Chapter 2 explains how epigenetic data is used to increase the detection power for eGenes in one tissue dataset of the GTEx data. Chapter 3 introduces a new meta-analysis which pools multiple tissue datasets containing few samples to improve the power for discovering eGenes in the GTEx data. Chapter 4 describes various ways to transform GO labels into vectors, and our tasks to critically evaluate different types of embeddings for GO labels. Chapter 5 explains our deep learning model to predict GO labels for proteins from their amino acid sequences and other external datasets like the definitions of the GO labels, the protein motifs, the protein 3D structures, and the protein-protein interaction network. Chapter 6 summarizes our findings, and Chapter 7 discusses future research directions.

CHAPTER 2

Model importance of SNPs to discover eGenes

2.1 Introduction

A genome wide association study (GWAS) identifies single nucleotide polymorphisms (SNPs) which are associated to a given phenotype. In GWAS, the phenotype value is either discrete (e.g. 1 or 0) to label whether a person has the phenotype or not, or continuous for traits like height and weight. These traditional GWAS assume genes affecting the phenotypes to be the genes located near the significant SNPs. However, because genes induce the phenotypes, understanding how SNPs affect gene expressions in different body tissues will better explain the possible causal factors of the phenotypes.

With the advancement of sequencing technology in the past decade, scientists can now readily measure the expression level of a gene. Association studies can then be applied on these gene expression datasets to find SNPs correlated with the expressions of the genes. These SNPs are named expression quantitative trait loci (eQTLs), and these types of association studies are named eQTL studies. Important products of eQTL studies are the eGenes, which are the genes having expression levels associated with at least one eQTL. Because SNPs are unique for each person, eGenes are important as they are likely to be responsible for the phenotypic variations in a population and should be prioritized for causal analysis of the phenotypes.

eQTLs and eGenes are defined with respect to the tissue of interest because the same gene does not have the same expression level in every tissue. Since 2015, thanks to the effort of the Genotype-Tissue Expression (GTEx) consortium, gene expression datasets have become available to determine the eGenes in each of the 44 common human tissues [98]. By the definition of eGenes,

we need to first find the SNPs which are eQTLs for the genes, and then we can determine whether these genes are eGenes. In this work, we focus on SNPs located near a specific gene (cis-SNPs) because these SNPs are more likely to affect the gene expression than the SNPs located farther away (trans-SNPs).

The traditional method for eGene discovery requires the eQTL study to estimate the effect size of each cis-SNP against the gene expression. Theses effect sizes are then transformed into p-values, and the *eGene test statistic* for the gene is defined as the minimum of these p-values [98]. The next step computes the *eGene p-value* which is the p-value of the eGene test statistic (so eGene p-value is the p-value of a p-value). Due to Linkage Disequilibrium (LD) of the cis-SNPs, the eGene p-value is estimated by the permutation test to correct. Recent literature for eGene discovery have focused on reducing the permutation test runtime.

No study has yet improved the statistical power for discovering eGenes by leveraging the physical locations of the cis-SNPs or any other types of epigenetic annotation. Traditional eQTL study assumes uniform weights for the cis-SNPs of a gene. However, cis-SNPs located near transcription start sites (TSS), DNAse and histone binding regions are more likely to affect the gene expression [99]. Modeling the location information of the cis-SNPs should enable us to more accurately identify the SNPs affecting the gene expression. This outcome will result in a higher statistical power for discovering eGenes, because finding the eQTLs is a prerequisite for finding the eGenes.

We introduce a new eGene test statistic integrated with the SNP location information, and estimate its p-value by adapting the method in Sul *et al.* [94]. We consider the following genomic regions: transcription start sites, DNase hypersensitive sites and histone modification sites. We estimate the prior weights for the SNPs located inside and outside these regions, so that SNPs in important locations should contribute more to the eGene test statistic. Modeling each type of SNP information increases the number of discovered eGenes by at least 16% more than the traditional method, and that TSS appears to be the most important factor having up to 57% more discoveries. Our software is at `https://github.com/datduong/Find-eQTL-eGene`.

2.2 Association study for eQTLs

We present the traditional association study for eQTLs. SNPs associated with the expression of a gene are called eQTLs, and an association study to find eQTLs is called an eQTL study. Because genes are expressed nonuniformly across tissues, an eQTL study for a gene g is conducted separately for each tissue t and aims to detect all the SNPs associated with the expression of g in t. Suppose we are given the expression level for g in a tissue t from N individuals, then we can represent this information as a vector $\mathbf{y} \in \mathbb{R}^N$. We now estimate the association strength of each SNP s in the set S for the gene expression $\mathbf{y}$. Let $\mathbf{x} \in \mathbb{R}^N$ denote the genotypes at the SNP s for N individuals in the data. The eQTL study assumes the following relationship for the SNP s and the expression of gene g in tissue t

$$\mathbf{y} = \beta_{sgt}\mathbf{x} + \boldsymbol{\epsilon}_{sgt} \tag{2.1}$$

where $\boldsymbol{\epsilon}_{sgt} \in \mathbb{R}^m$ are sampling errors having the distribution $\boldsymbol{\epsilon}_{sgt} \sim \mathrm{N}(0, \sigma^2_{\epsilon_{sgt}}\mathbf{I})$, and $\beta_{sgt} \in \mathbb{R}$ represents the true effect size of s toward $\mathbf{y}$ conditioned on the tissue t [33]. In Eq. 2.1, we have excluded variables representing possible confounders because the GTEx consortium already removed the confounder effects from their datasets.

The approximation $\hat{\beta}_{sgt}$ of the true value β_{sgt} in Eq. 2.1 is solved via the optimization $\hat{\beta}_{sgt} = \arg\min_{\beta_{sgt}} ||\mathbf{y} - \beta_{sgt}\mathbf{x}||_2^2$, which has the solution $\hat{\beta}_{sgt} = (\mathbf{x}^\intercal\mathbf{x})^{-1}\mathbf{x}^\intercal\mathbf{y}$ where $\hat{\beta}_{sgt} \sim \mathrm{N}(\beta_{sgt}, (\mathbf{x}^\intercal\mathbf{x})^{-1}\sigma^2_{\epsilon_{sgt}})$ [1]. We perform a hypothesis test with $\mathrm{H}_0 : \beta_{sgt} = 0$ and $\mathrm{H}_1 : \beta_{sgt} \neq 0$ to confirm if s is associated with the expression of g. For this hypothesis test, we estimate $\hat{\sigma}^2_{\epsilon_{sgt}}$ for $\sigma^2_{\epsilon_{sgt}}$ as $\hat{\sigma}^2_{\epsilon_{sgt}} = \frac{1}{m-1}||\mathbf{y} - \hat{\beta}_{sgt}\mathbf{x}||_2^2$ and the variance of $\hat{\beta}_{sgt}$ as $\widehat{\mathrm{var}}(\hat{\beta}_{sgt}) = (\mathbf{x}^\intercal\mathbf{x})^{-1}\hat{\sigma}^2_{\epsilon_{sgt}}$. Then, we compute the p-value p_{sgt} of $\hat{\beta}_{sgt}$ based on the t-distribution $\mathrm{N}(0, (\mathbf{x}^\intercal\mathbf{x})^{-1}\sigma^2_{\epsilon_{sgt}})$ [1, 33]. Suppose p_{sgt} is less than the given significance level α, then we reject the null hypothesis $\mathrm{H}_0 : \beta_{sgt} = 0$ and conclude that the SNP s is an eQTL for the expression of gene g conditioned on tissue t.

2.3 Association study for eGenes

The traditional method for finding the eGenes relies on the eQTL results explained above. Conditioned on tissue t, from the eQTL result at each SNP $s \in S$ on the expression of gene g, we have a set of p-values $\{p_{sgt}\}_{s\in S}$. The minimum of the set $p_{gt} = \min \{p_{sgt}\}_{s\in S}$ is the observed eGene test statistic for g in tissue t. There is a p-value $\alpha_{p_{gt}}$ defined for this test statistic p_{gt} [98]. This $\alpha_{p_{gt}}$ is the eGene p-value and depends on two important factors: the number of SNPs $|S|$ and the LD of these SNPs. $\alpha_{p_{gt}}$ is estimated by the permutation test K times to correct for LD [26, 94]. In the k^{th} permutation, we permute the gene expression levels for the N individuals, and compute a new $p_{gt}^{(k)} = \min \{p_{sgt}^{(k)}\}_{s\in S}$. The p-value $\alpha_{p_{gt}}$ is the rank of p_{gt} with respect to the null density created by the permutation values $\{p_{gt}^{(k)}\}_{k\in K}$. If $\alpha_{p_{gt}}$ is less than some desired threshold, then we conclude that g is an eGene in the tissue t. When $|S|$ is large, the permutation test can be time-consuming. Later, we will introduce an efficient approximation for the permutation test.

We present an important idea that will be a key factor in Section 2.5.1. The SNP effect $|\hat{\beta}_{sgt}|$ and its p-value p_{sgt} have a inverse one-to-one relationship, so that we can also use $b_{gt} = \max \{|\hat{\beta}_{sgt}|\}_{s\in S}$ as the eGene test statistic instead of $p_{gt}^{(k)} = \min \{p_{sgt}^{(k)}\}_{s\in S}$. This observation is particularly advantageous, because we will not need to compute the p-value p_{sgt} for each SNP effect. In this case, the eGene p-value $\alpha_{b_{gt}}$ is also computed by doing a permutation test; however, the null density is formed by the permutation values $\{b_{gt}^{(k)}\}_{k\in K}$.

2.4 Multi-threshold association study for GWAS

The traditional method for discovering eGenes does not assume any prior information about the SNPs. We develop a new strategy that emphasizes the SNP locations because SNPs located in regulatory regions are more likely to affect the gene expression. Our strategy is based on the multi-threshold association study developed by us [20, 32]. Our model Darnell *et al.* [20] was designed for finding significant SNPs in a GWAS and not eQTLs for a gene expression. Because of the similarity between GWAS and eQTL study, this model can be implemented to find eQTLs of gene expression. Despite our GWAS model weighted each SNPs based on their own importance,

it assumed that these SNPs are independent. Applying our GWAS model without modifications requires pruning cis-SNPs in high LD or expensive permutation test. It is however important to understand how to adapt our previous model for the problem of finding eQTLs and then eGenes, because this model serves as a foundation for the new method in this chapter.

From our work Darnell *et al.* [20], suppose that we are estimating the association of S SNPs with respect to the expression of gene g in tissue t. From Section 2.3, we can compute the effects $\{\hat{\beta}_{sgt}\}_{s\in S}$ of these SNPs against the gene expression. The statistical power of the association test at one SNP s is the probability $|\hat{\beta}_{sgt}| > F_{H_1}^{-1}(1-\alpha_{sgt}/2)$ assuming that $H_1 : \mu_i \neq 0$ is true. F_{H_1} is the cumulative distribution for $\hat{\beta}_{sgt}$ under H_1. α_{sgt} is the significant threshold for SNP s; for example, Bonferroni correction uses $\alpha_{sgt} = 1/|S|$. When the sample size N is large, $P_s(\alpha_{sgt}, \mu_i)$ can be estimated with the cumulative normal distribution Φ.

$$P_s(\alpha_{sgt}) = \mathbb{P}(|\hat{\beta}_{sgt}| \geq F_{H_1}^{-1}(1-\alpha_{sgt}/2)) \tag{2.2}$$

$$= \Phi(\Phi^{-1}(\alpha_{sgt}/2) - \mu_i) + 1 - \Phi(\Phi^{-1}(1-\alpha_{sgt}/2) - \mu_i) \tag{2.3}$$

When the experiment involves S SNPs, the statistical power is the weighted mean $P(\{\alpha_{sgt}\}_{s\in S}) = \sum_{s=1}^{S} c_i P_s(\alpha_{sgt})$ where $c_s \geq 0$ for all s and $\sum_{s=1}^{S} c_s = 1$ [32]. $\{c_s\}_{s\in S}$ are numeric values representing the relative importance of the SNPs with respect to the gene g. For example, SNPs in regulatory regions are assigned larger c_s than the other SNPs. For now, we assume that $\{c_s\}_{s\in S}$ is known beforehand. Later, we drop this assumption and estimate them from the data.

We now maximize $P(\{\alpha_{sgt}\}_{s\in S})$ with respect to $\{\alpha_{sgt}\}_{s\in S}$. Intuitively, we are finding the best set of nonuniform thresholds for the SNPs based on their relative importance. By maximizing the power for detecting eQTLs of the gene g, we also increase the power for determining if g is an eGene. To account for multiple testing, we constrain that $\alpha_{sgt} \geq 0$ for all s and $\sum_{i=1}^{M} \alpha_{sgt} = \alpha$, where α is the overall significant threshold. We take the gradient of $P(\{\alpha_{sgt}\}_{s\in S})$ and the Lagrangian term $\lambda(1 - \sum_{s=1}^{S} \alpha_{sgt})$ where λ is the Lagrangian. The optimal solution is obtained when the gradients

of any two SNP i and j in S are equal [20]

$$\frac{0.5\ c_i\ [\phi_{\mu_i}(\Phi^{-1}(\alpha_{igt}/2)) + \phi_{\text{-}\mu_i}(\Phi^{-1}(\alpha_{igt}/2))]}{\phi_0(\Phi^{-1}(\alpha_{igt}/2))} = \frac{0.5\ c_j\ [\phi_{\mu_j}(\Phi^{-1}(\alpha_{jgt}/2)) + \phi_{\text{-}\mu_j}(\Phi^{-1}(\alpha_{jgt}/2))]}{\phi_0(\Phi^{-1}(\alpha_{jgt}/2))} \tag{2.4}$$

ϕ_w represents the probability density for a normal distribution with mean w and variance 1. $\Phi^{-1}(w)$ represents the quantile with respect to w under a normal distribution with mean 0 and variance 1. After we find $\{\alpha_{sgt}\}_{s\in S}$ satisfying Eq. 2.4, if the p-value p_{sgt} of $\hat{\beta}_{sgt}$ is less than α_{sgt}, then SNP s is an eQTL for the gene g conditioned on the tissue t. If at least one SNP $s \in S$ is an eQTL then g is an eGene conditioned on tissue t. The method presented in this section accounts for SNP locations; however there are two key problems. First, the formulation holds true only when SNPs are independent. Second, solving for α_{sgt} satisfying Eq. 2.4 can be time-consuming.

2.5 Multi-threshold association study for eQTLs and eGenes

We introduce a new eGene test statistic that accounts for the SNP locations but does not require the solution for the α_{sgt}. We also explain how the permutation test can be applied to correct for LD of the SNPs. We interpret each gradient term in Eq. 2.4 as a likelihood ratio scaled by the SNP relative importance c_s, and define an equivalent form for the observe SNP effect $\hat{\beta}_{sgt}$.

$$g_s(\hat{\beta}_{sgt}) = \frac{0.5\ c_s\ [\phi_{\mu_s}(\hat{\beta}_{sgt}) + \phi_{\text{-}\mu_s}(\hat{\beta}_{sgt})]}{\phi_0(\hat{\beta}_{sgt})} \tag{2.5}$$

Eq. 2.5 is the likelihood ratio evaluated at $\hat{\beta}_{sgt}$ with the null hypothesis being $H_0 : \beta_{sgt} = 0$ and the alternate hypothesis being the average of two hypotheses $H_1 : \beta_{sgt} = \mu_i$ and $H_1 : \beta_{sgt} = \text{-}\mu_i$.

We observe that $g_i(\hat{\beta}_{sgt})$ is monotonic decreasing with respect to $|\hat{\beta}_{sgt}|$; that is, as $|\hat{\beta}_{sgt}|$ gets close to zero then $g_i(\hat{\beta}_{sgt})$ gets larger. Suppose we test only a single SNP s against the gene g, then instead of using $\hat{\beta}_{sgt}$ or its p-value p_{sgt}, we can use the likelihood ratio $g_s(\hat{\beta}_{sgt})$.

For many SNPs S, we define the new eGene test statistic based on the likelihood ratio to be

$$\mathrm{LR}_{gt} = \max\ \{g_s(\hat{\beta}_{sgt})\}_{s\in S} \tag{2.6}$$

We use the name LR_{gt} to indicate that our test statistic is based on the likelihood ratio formulation and to differentiate it from the traditional value p_{gt}. Here, SNPs with large c_s values will contribute more toward LR_{gt}.

To compute the LD-corrected eGene p-value, we perform the traditional permutation test. For each iteration k, we permute the gene expression levels for the N individuals but keep their genotypes unchanged to account for the LD, and then compute the permutation test statistic $\mathrm{LR}_{gt}^{(k)}$. The LD-corrected eGene p-value is $\alpha_{\mathrm{LR}_{gt}}^{\mathrm{Pt}} = \frac{1}{K}\sum_{k=1}^{K} \mathbb{1}(\mathrm{LR}_{gt} \leq \mathrm{LR}_{gt}^{(k)})$.

2.5.1 LD-corrected eGene p-value

Because there are thousands of genes in the GTEx data, the permutation test can take a very long time. Sul *et al.* [94] introduced a sampling procedure, named eGene-Mvn, for computing the eGene p-value that has lower runtime than the permutation test. Their model is designed for the SNP effects $\{\beta_{sgt}\}_{s\in S}$ but not the new test statistics $\{g_s(\hat{\beta}_{sgt})\}_{s\in S}$. We adapt their approach to estimate the p-value of our LR_{gt} in the following way. Assuming the null hypothesis is true, in the k^{th} permutation iteration, we sample the SNP effects $\{\hat{\beta}_{sgt}^{(k)}\}_{s\in S}$ from the multivariate normal distribution $\mathrm{Mvn}(\mathbf{0}, \Sigma)$, and then calculate the eGene test statistic $\mathrm{LR}_{gt}^{(k)} = \max\ \{\hat{\beta}_{sgt}^{(k)}\}_{s\in S}$. The covariance $\Sigma \in \mathbb{R}^{S\times S}$ is the LD for S SNPs and computed by $\frac{1}{N}X^{\intercal}X$ where $X \in \mathbb{R}^{N\times S}$ is the genotype matrix.

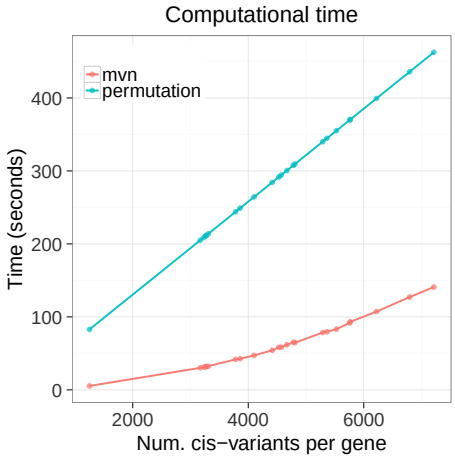
The key assumption in eGene-Mvn is that $\mathrm{cov}(\hat{\beta}_{ugt}, \hat{\beta}_{vgt}) = \Sigma_{uv}$ for any SNP $u, v \in S$ [43]. By default, the sampling process handles the correlations of SNPs. By sampling for $\{\hat{\beta}_{sgt}^{(k)}\}_{s\in S}$, we can also save time by avoiding to compute for these values by fitting Eq. 2.1. We define the eGene p-value of this approximation to be $\alpha_{\mathrm{LR}_{gt}}^{\mathrm{Mvn}} = \frac{1}{K}\sum_{k=1}^{K} \mathbb{1}(\mathrm{LR}_{gt} \leq \mathrm{LR}_{gt}^{(k)})$.

The core of eGene-Mvn is the sampling from Mvn distribution which runs in $\mathcal{O}(S^3)$ with respect to the number of cis-SNPs. To avoid very high runtime, we divide the set S into smaller

subsets, and then take the maximum of the highest $g_s(\hat{\beta}_{sgt}^{(k)})$ in each subset to compute the eGene test statistic $\text{LR}_{gt}^{(k)}$. The permutation test has linear time with respect to the number of cis-SNPs. Interestingly in practice, we observe that on average our approximation costs less time than the permutation test. We compare the runtime of both approaches to compute the eGene p-values for a subset of genes in the GTEx data of the liver tissue. In both cases, we have 97 individuals and $K = 1000$ (e.g. the number of iterations in the permutation test or the number of samples from the Mvn).

We observe that eGene-Mvn runs in polynomial time but grows at a lesser pace than the linear time of permutation test for small S (Fig. 2.1). In the GTEx data of the liver tissues, 95% of the genes have less than 6833 cis-SNPs. In practice, the polynomial runtime of eGene-Mvn does not strictly invalidate its application. Because the permutation test is considered as the gold-standard, we will later compare the permutation test and eGene-Mvn.

Figure 2.1: Permutation test and eGene-Mvn runtime to compute eGene p-values.



2.5.2 Estimate SNP prior information from data

Despite knowing that SNPs should be weighted based on their locations, we will not know the values of the weights $\{c_s\}_{s \in S}$. We introduce a heuristic approach to estimate these values. Suppose there are J types of regulatory regions (e.g. TSS, DNAse and histone binding sites). We assign each type a weight w_j, so that the contribution of a SNP s is $u_s = \sum_{j=1}^{J} w_j \mathbb{1}(s \in j)$ and c_s is computed as

$c_s = u_s / \sum_{k=1}^{S} u_k$. We assume that the weights $\{w_j\}_{j \in J}$ interact linearly, but it is possible to model interaction among these annotations. We observe that c_s scales the likelihood ratio in $g_s(\hat{\beta}_{sgt})$ which are then later used to find the test statistics LR_{gt}. For a grid search on $\{w_j\}_{j \in J}$, we need to compute the likelihood ratio at each SNP only once, which helps reducing runtime.

We apply grid search to find $\{w_j\}_{j \in J}$ for a small subset of genes in the liver tissue, and use the following heuristic strategy to reduce the computation time when there are many types J. Conditioned on a guess $\mathbf{w}^{(k)} = \{w_j^{(k)}\}_{j \in J}$ in the grid search, let $\text{LR}_{gt}(\mathbf{w}^{(k)})$ be the test statistic computed at the weights $\mathbf{w}^{(k)}$. At a gene g there exists a value $x_g(\mathbf{w}^{(k)})$ such that g is a an eGene when $\text{LR}_{gt}(\mathbf{w}^{(k)}) > x_g(\mathbf{w}^{(k)})$. In a gene set G, with the choice $\mathbf{w}^{(k)}$, the value $x_g(\mathbf{w}^{(k)})$ for $g \in G$ will determine the number of discovered eGenes. From the value $x_g(\mathbf{w}^{(k)})$, we aim to infer the quantity computed at another set of weights $x_g(\mathbf{w}^{(\ell)})$.

We pick a starting weight set $\mathbf{w}^{(0)}$ and compute the value $x_g(\mathbf{w}^{(0)})$ for all gene $g \in G$. Next we apply grid search to a small subset G^1 of G, and compute $x_g(\mathbf{w}^{(k)})$ for every choice. Then we estimate a linear relationship for $x_g(\mathbf{w}^{(0)})$ and $x_g(\mathbf{w}^{(k)})$ as

$$\delta(\mathbf{w}^{(k)}, \mathbf{w}^{(0)}) = \frac{1}{|G^1|} \sum_{g \in G^1} \frac{x_g(\mathbf{w}^{(k)})}{x_g(\mathbf{w}^{(0)})} \tag{2.7}$$

Now we can estimate the values $x_g(\mathbf{w}^{(k)}) = \delta(\mathbf{w}^{(k)}, \mathbf{w}^{(0)}) x_g(\mathbf{w}^{(\mathbf{0})})$ for the rest of the genes $g \in G \setminus G^1$, so that we do not perform the grid search on the entire set G. The best choice for $\mathbf{w}^{(k)}$ is the one that returns the most number of discovered eGenes on the set G.

Our strategy can be applied with the permutation test or eGene-Mvn. We emphasize that our approximation is heuristic and finds an option for $\mathbf{w}^{(k)}$ that is good enough. Because we determine $\mathbf{w}^{(k)}$ from the data, to avoid data reusing we divide the data into two subsets. We obtain best $\mathbf{w}^{(k)}$ in each set and then apply this solution to the other set.

2.6 The Genotype-Tissue Expression data

We briefly describe the Genotype-Tissue Expression (GTEx) Pilot Dataset version 6 released on January 12, 2015 [98]. The GTEx consortium collected tissue samples from 544 donors for 44 body tissues, and measured the gene expression level in each sample by evaluating the amount of RNA being expressed by each gene. The genotype of each person was imputed, with about 6.8 million SNPs of minor allele frequency above 5% after quality control. There were 40,490 genes measured with respect to all the body tissues. For some tissues, it was difficult to collect samples, resulting in few data points. For example, the nine brain tissue datasets had a median sample size of 90 people. There were more observations for the tissues which were easy to collect samples from; for instance, whole blood and subcutaneous adipose tissue had 338 and 298 samples, respectively. In 2015, the GTEx pilot analysis suggested a sufficient sample size $\geq$ 80 donors for single-tissue eQTL study.

2.7 False-positive rate simulation

This simulation assumes the true hypothesis to be the null hypothesis which claims that gene g is not an eGene in tissue t. We want to confirm that our eGene test statistic LR_{gt} produces the correct false-positive rate regardless of whether we implement the permutation test or eGene-Mvn. We consider the simplest case and measure the false-positive rate for just one single gene, because this single value will affect the overall false-positive rate when we jointly test many genes in a tissue.

We simulate many datasets under the null hypothesis. Each time, we apply the uniform and nonuniform prior for the SNPs, and compute LR_{gt} and its p-value with the permutation test and eGene-Mvn (in total 4 experiments). Because we are testing one gene, a simulated eGene p-value less than 0.05 is a false-positive. Regardless of the SNP prior information and the choice of permutation test or eGene-Mvn, the distribution of the simulated p-values should align well with the uniform distribution (and the fraction of times a simulated p-value is less than 0.05 should be very close to 0.05). Also, the Mvn sampling method should display the same results as the permutation test.

Figure 2.2a shows that these expectations are true for the simulated distributions of the permutation test and eGene-Mvn p-values $\alpha^{\text{Pt}}_{\text{LR}_{gt}}$ and $\alpha^{\text{Mvn}}_{\text{LR}_{gt}}$. In each experiment, the QQ plot with respect to the uniform density shows that the simulated distribution aligns well with the uniform density. When we apply the uniform prior, the permutation test and eGene-Mvn return false-positive rates for LR_{gt} of 0.046 and 0.044, respectively. For the nonuniform prior, the same two numbers are 0.051 and 0.052. Below, we explain how to generate the data and compute the results in Fig. 2.2a.

For a realistic simulated experiment where LD can affect the outcome, we use the gene TCEA3 in chromosome 1 that has 3872 cis-SNPs. We define the region 150 kilobases up and downstream from the TSS as the TSS_{150} region. 431 of the cis-SNPs are within its this region. We evaluate the permutation test and eGene-Mvn for two cases: the uniform prior where all SNPs are equally weighted (e.g. $c_s = 1/|S|$), and the nonuniform prior where SNPs inside TSS are weighted 100 times more (e.g. $c_i = 100c_j$ for $i \in \text{TSS}_{150}$ and $j \notin \text{TSS}_{150}$). We also try a different relative weight option in Section 2.10. Next, we select the true SNP effect under the alternative $\mu_s = 3.5$ for all $s \in S$; this number is the mean of all the observed SNP effects in the entire gene expression data of the liver tissue. We will propose other ways to select μ_s in the discussion section.

The permutation test has 10^4 iterations (same as the original GTEx pilot study). Each time, we permute TCEA3 expression levels in the liver tissue for the 97 individuals and keep their genotype fixed, and then compute the eGene p-value. Because eGene-Mvn runs faster than the permutation test, we can generate more than 10^4 samples. We draw 10^6 samples from the null density $\text{Mvn}(\mathbf{0}, \Sigma)$, and compute the corresponding eGene p-value for each sample. Σ is computed as the LD of the 3872 cis-SNPs. The false-positive rate is the fraction of times the eGene p-value is less than 0.05. Because we generate more samples for eGene-Mvn, there is a higher chance to observe a simulated p-value below 0.05. Despite being at a disadvantage compared to permutation test, eGene-Mvn can obtain a good false-positive rate in our simulation.

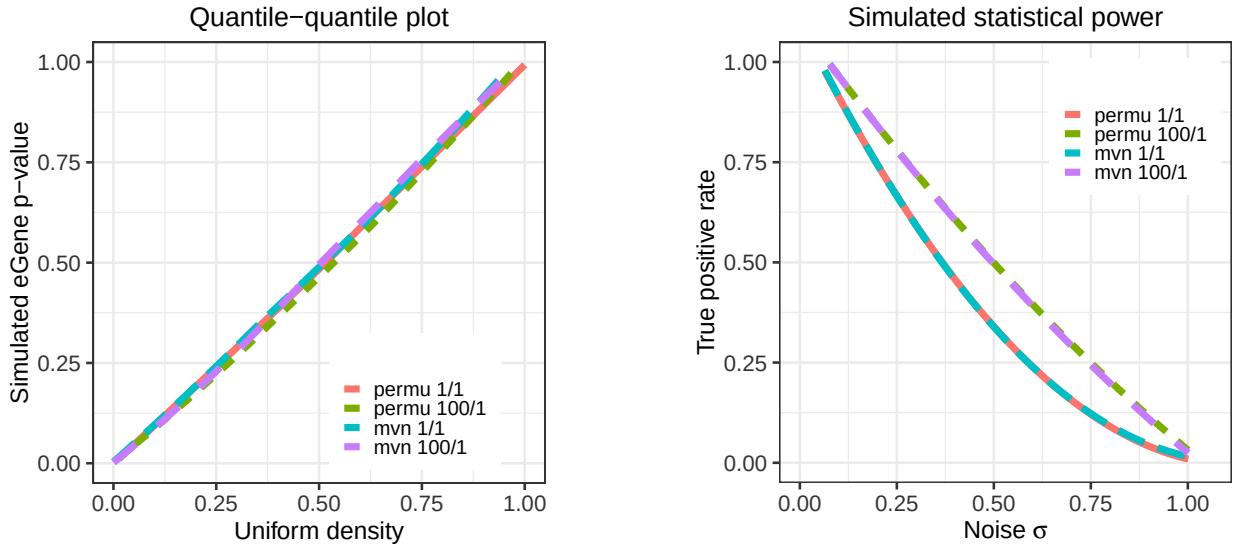
2.8 Statistical power simulation

When provided with meaningful SNP prior information, our test statistic LR_{gt} increases the power to detect eGenes. The statistical power of finding one eQTL will affect the overall power of the whole study which involved all the gene's cis-SNPs. We consider this simple case and perform the simulation for which there exists just one eQTL for the gene expression. We test both the uniform and the nonuniform priors for the SNPs. The weights $\{c_s\}_{s\in S}$ for each prior type are taken from the false-positive rate simulation. Compared to the uniform prior, the nonuniform prior significantly increases the statistical power (Fig. 2.2b). For each prior, the permutation test and eGene-Mvn are the same, indicating that our eGene-Mvn adaptation works well. Below, we explain how the data is generated to compute the results in Fig. 2.2b.

Each simulation study has 97 individuals, same as the true number of samples in the liver tissue. The gene expression levels for these people are generated based on the equation $y = x_{\mathrm{eQTL}}\beta_{\mathrm{eQTL}} + \epsilon$, where $x_{\mathrm{eQTL}} \in \mathbb{R}^{97}$ and β_{eQTL} are the genotypes and true effect size of the eQTL. $\epsilon \in \mathbb{R}^{97}$ is the random noise drawn from $\mathrm{Mvn}(\mathbf{0}, \mathbf{I}\sigma^2)$ where $\mathbf{I} \in \mathbb{R}^{97\times 97}$ is the identity matrix. We vary σ from 0 to 1.5, and at each instance, perform the simulation study 200 times to evaluate the statistical power. When σ is large, the randomness factor dominates the eQTL effect, and the power for finding associations of the SNPs and gene expression should decrease. This specific case serves as a sanity test for our result.

To consider LD, we generate the gene expressions based on the 3872 cis-SNPs of TCEA3. We choose the eQTL as the SNP in the TSS_{150kb} region of TCEA3 that has the highest observed $|\hat{\beta}_{sgt}|$ on the expression of TCEA3 in the liver tissue. Denote this effect size as $\beta_{\max}$ and the genotypes for the corresponding SNP as $x_{s_{\max}}$. We generate the gene expression using $y = x_{s_{\max}}\beta_{\max} + \epsilon$. We set $x_{s_{\max}}$ as the real genotypes of the 97 individuals in the liver tissue. After simulating the expression y, we perform the association test at each of the 3872 cis-SNPs and then compute the eGene p-value using the permutation test and eGene-Mvn. Because we are testing only one single gene, suppose the eGene p-value is less than 0.05 then we consider this gene to be an eGene.

Figure 2.2: Prior on the TSS_{150kb} region is applied to multi-threshold association study for eGenes. 100/1 indicates that relative weights are $c_i = 100c_j$ for $i \in \text{TSS}_{150kb}$ and $j \notin \text{TSS}_{150kb}$. Number 1/1 indicates a uniform prior for cis-SNPs of a gene. (a) Quantiles of uniform density versus quantiles of many eGene p-values simulated under null hypothesis. (b) Simulated statistical power under alternative hypothesis.



(a) **(b)**

2.9 Model transcription start sites

For proof-of-concept, we apply our method to the pilot GTEx data of the liver, which has 97 samples and expression measurements for 21,868 genes in 22 autosomal chromosomes [98]. Following the GTEx consortium, we consider only the cis-SNPs of each gene which are SNPs located within 1 Megabase upstream and downstream of the TSS. The average number of cis-SNPs per gene is 4681. We define SNPs to be in the TSS_{150kb} region if they are within 150 Kilobase upstream and downstream of the TSS for a gene. The average fraction of SNPs inside this region is 14.74%.

For the simplicity sake, we use one single type of SNP location information and weigh the cis-SNPs within the TSS_{150kb} 100 times more than the other cis-SNPs (e.g. $c_i = 100c_j$ for $i \in \text{TSS}_{150kb}$ and $j \notin \text{TSS}_{150kb}$). Because there are two ways to compute the eGene p-values (permutation test versus eGene-Mvn), we compute the test statistics LR_{gt} and its eGene p-value with both approaches (so that there are four experiments). An eGene p-value is computed with 10^6 iterations for eGene-Mvn, and with 10^4 iterations for the permutation test (same iteration number in GTEx pilot study)

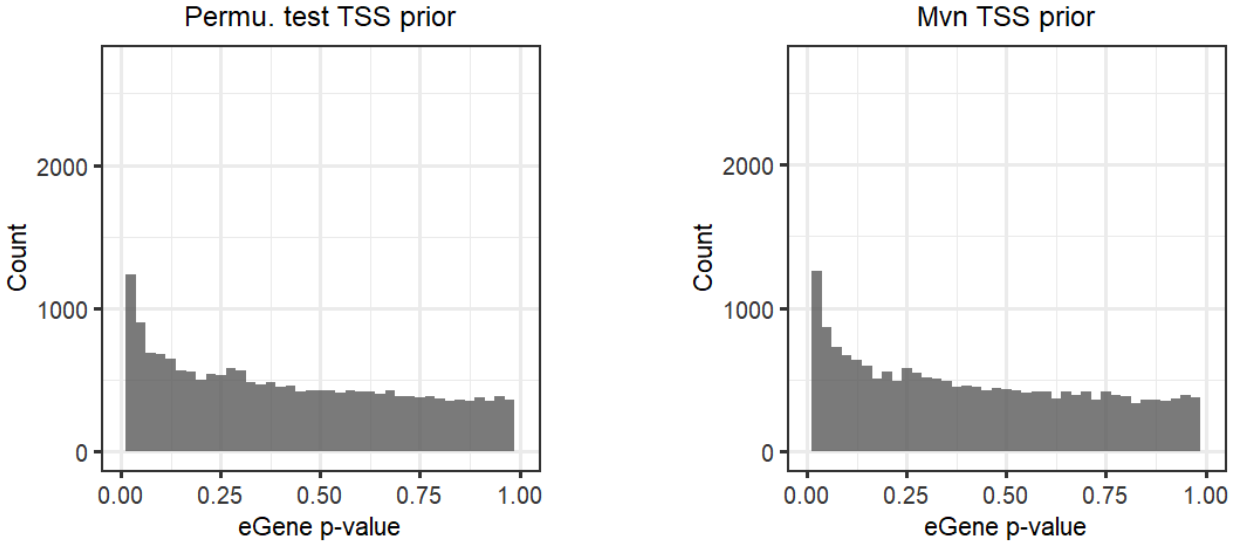
[98]. For eGene-Mvn, we divide the cis-SNPs into independent segments of length 500 Kilobase, and perform the Mvn sampling to each block. The final eGene p-value for the gene is the minimum p-value over taken all the segments. This strategy keeps the runtime of Mvn sampling $\mathcal{O}(S^3)$ from getting very high.

Following the GTEx pilot analysis, we control for the total false discovery rate on the entire gene set G by transforming the set of eGene p-values $\{\alpha_{\mathrm{LR}_{gt}}\}_{g\in G}$ into q-values $\{q_{\mathrm{LR}_{gt}}\}_{g\in G}$. Genes with q-values less than 0.05 are classified as eGenes [90, 98]. One key assumption for producing valid q-values is that the distribution of the eGene p-values must have a relatively flat right tail [90], and our p-value set $\{\alpha_{\mathrm{LR}_{gt}}\}_{g\in G}$ meet this condition (Fig. 2.3).

Table 2.1 shows the number of discovered eGenes increases by 57% when using just the TSS as the SNP location formation. There are small differences between the permutation test and eGene-Mvn for computing the eGene p-values (and thus the two methods also produce different q-values). To quantify the differences, we compare $\{\alpha^{\mathrm{Mvn}}_{\mathrm{LR}_{gt}}\}_{g\in G}$ and $\{\alpha^{\mathrm{pt}}_{\mathrm{LR}_{gt}}\}_{g\in G}$. Visually, the two sets are very similar (Fig. 2.4a), and the absolute difference $|\alpha^{\mathrm{Mvn}}_{\mathrm{LR}_{gt}} - \alpha^{\mathrm{pt}}_{\mathrm{LR}_{gt}}|$ is less than 0.10 for almost all $g \in G$ (Fig. 2.4b). We also compare the q-values $\{q^{\mathrm{Mvn}}_{\mathrm{LR}_{gt}}\}_{g\in G}$ and $\{q^{\mathrm{pt}}_{\mathrm{LR}_{gt}}\}_{g\in G}$ and do not observe significant differences (Fig. 2.4c and 2.4d).

The permutation test and eGene-Mvn agree on 2379 eGenes in the GTEx liver tissue dataset. Because the permutation test is considered as the gold-standard, we have that $\{q^{\mathrm{Mvn}}_{\mathrm{LR}_{gt}}\}_{g\in G}$ approximate $\{q^{\mathrm{pt}}_{\mathrm{LR}_{gt}}\}_{g\in G}$. We analyze the false negative and false positive cases for the Mvn sampling method. For the 66 false negative cases, their maximum and median q-value are 0.079 and 0.053. For the 70 false positive cases, their minimum and median q-value are 0.028 and 0.045. We observe that these incorrect cases are genes having borderline q-values and are very hard to classify. Because in practice eGene-Mvn method has lower runtime than permutation test, we will apply only eGene-Mvn for estimating eGene p-values for the rest of this paper and accept that there are small errors.

Figure 2.3: Histograms for eGene p-values of all the genes in Liver data computed by (a) permutation test and (b) eGene-Mvn. Assumption for valid q-values is that eGene p-value distribution should have flat right end. Both ways to compute eGene p-values satisfy this condition.

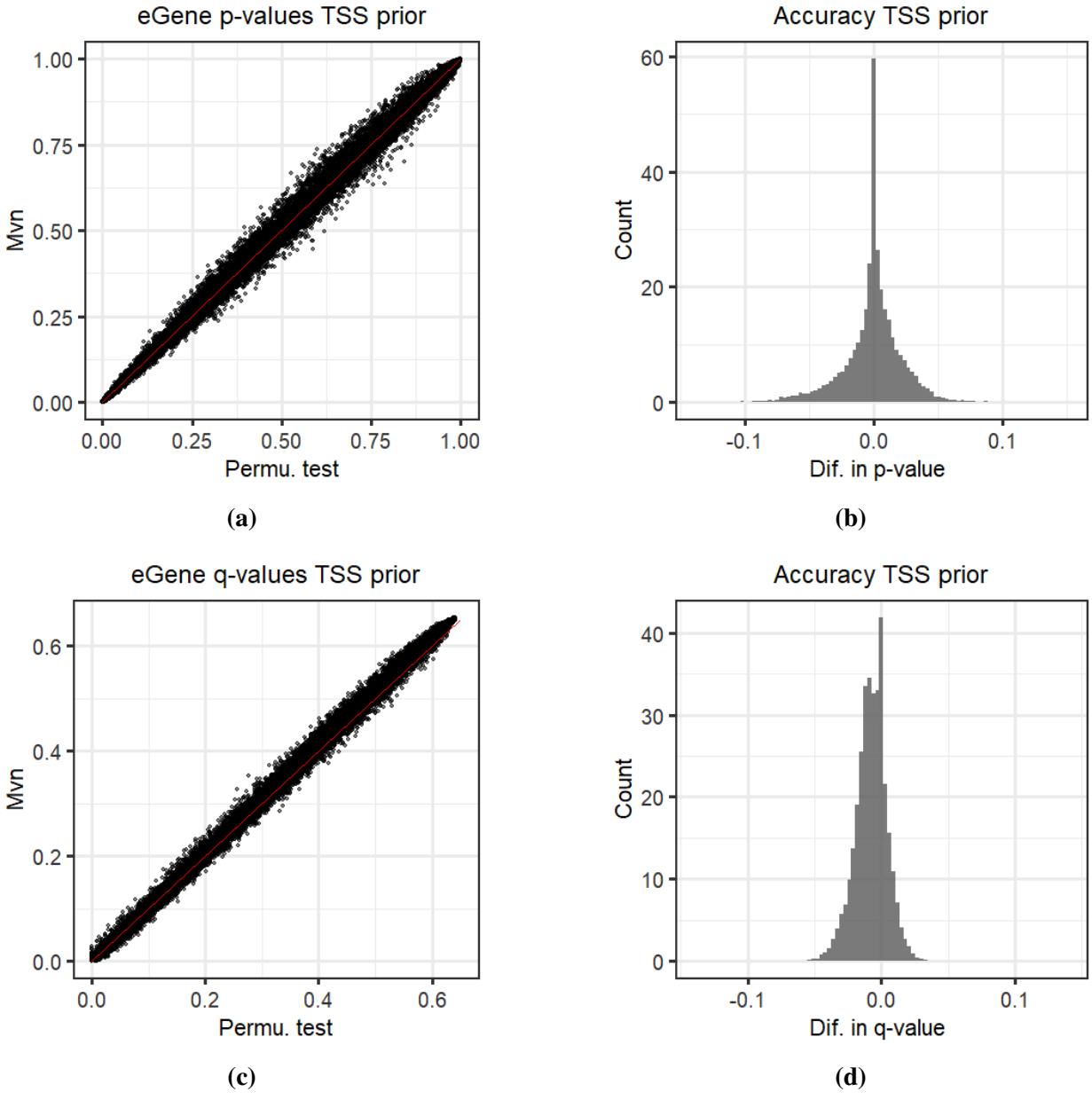


(a) **(b)**

Table 2.1: Number of eGenes (out of 21,868 genes in the Liver) for each option of uniform versus TSS prior and permutation test versus eGene-Mvn. is_eGene is an indicator for eGene. Conditioned on an option to compute eGene p-values, overlap[1] indicates the number of eGenes (or not) found by uniform and TSS prior data. Conditioned on a type of SNP location information, overlap[2] indicates the number of eGenes (or not) found by permutation test and eGene-Mvn. Permutation test is the gold-standard, and we do not observe significant differences for our eGene-Mvn implementation.

	permu.		**mvn**		overlap[2]	
is_eGene	yes	no	yes	no	yes	no
uniform	1626	20,242	1582	20,286	1549	20,209
TSS_{150kb}	2445	19,423	2449	19,419	2379	19,353
overlap[1]	1484	19,281	1457	19,294		

Figure 2.4: Visual comparison for (a) eGene p-values and (c) their q-values computed via permutation test and eGene-Mvn. Numerical comparison for the differences of (b) eGene p-values and (d) their q-values computed via permutation test and eGene-Mvn. eGene-Mvn overestimates large q-values in the permutation test, but this problem does not affect our result because large q-values will not be significant.



2.10 Model DNase hypersensitive sites

For the liver tissue data, we integrate into the eGene test statistic the TSS_{150kb} region and DNase hypersensitive sites as the two types of prior information, and determine their weights $\{w_j\}_{j\in J}$ from the data, where $J = \{\text{TSS}_{150kb}, \text{DNase site}, \text{neither}\}$. We define SNPs to be in the DNase sites if they are within the DNase hypersensitivity narrow gapped peaks of the Roadmap data [99]. DNase binding sites are usually indicators of accessible DNA regions. In these regions, SNPs which are eQLTs were found to be more likely to affect gene expression [23]. For the 21,868 genes in the GTEx data of the liver tissue, the average fraction of cis-SNPs in the TSS_{150kb} regions, DNase binding sites, and their intersection are 14.74%, 4.66%, and 0.88%, respectively.

We use $\mathbf{w} = [w_1, w_2, w_3]$ to represent the weights for the annotation types: TSS_{150kb}, DNase site, and neither. Only the relative weights matter; for example, the values $[1,1,1]$ and $[10,10,10]$ will result in the same set of $\{c_s\}_{s\in S}$ for the SNPs. We constrain each of w_1, w_2, and w_3 to be between 100 times more and less than the other two, and then apply Section 2.5.2 to find the supposedly best $\mathbf{w}$.

We begin with the first guess $\mathbf{w}^{(0)} = [1,1,1]$ and compute the eGene p-values for all the genes in the data. $\mathbf{w}^{(0)}$ is the uniform prior and we already have its result in Section 2.9. For each gene, we save the quantile $x_g(\mathbf{w}^{(0)})$ corresponding to the p-value threshold $\alpha_{sgt} = 0.01$ which is about the largest eGene p-value that has q-value less than 0.05 for our data.

At another choice $\mathbf{w}^{(k)}$, we compute the observed test statistic $\text{LR}_{gt}(\mathbf{w}^{(k)})$ for each gene as $x_g(\mathbf{w}^{(k)}) = \delta(\mathbf{w}^{(k)}, \mathbf{w}^{(0)}) x_g(\mathbf{w}^{(0)})$ as explained in Section 2.5.2. We compute the number of eGenes by counting the number of times $\text{LR}_{gt}(\mathbf{w}^{(k)}) > x_g(\mathbf{w}^{(k)})$ for all $g \in G$. For a few choices of $\mathbf{w}^{(k)}$, we apply eGene-Mvn without our approximation, and find the two results to be comparable (Table 2.2). Modeling the SNP prior based on just the TSS_{150kb} region yields the best results.

To validate the above conclusion, we estimate the effect of the TSS_{150kb} or DNase binding sites alone. We perform a complete grid search on $\mathbf{w}$. Grid search is feasible because this step requires solving for w_1 in $\mathbf{w}$ while fixing $w_2 = w_3 = 1$. The same logic applies when we model just the DNase sites, where we vary w_2 and keep $w_1 = w_3 = 1$. We find that TSS_{150kb} indeed has more

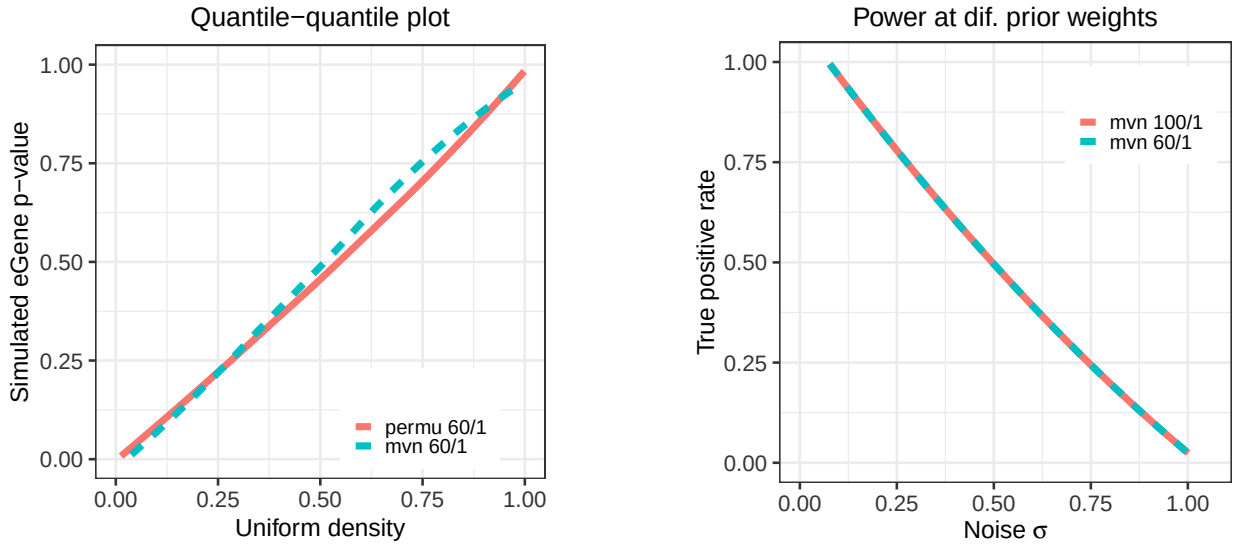
Table 2.2: Number of eGenes for the best 19 choices of $\mathbf{w}$ found via our estimation in Section 2.5.2. Subscripts are numbers computed via full grid-search to judge the approximation accuracy. Typically, downweighing SNPs located in TSS_{150kb} reduces the number of discovered eGenes compared to the traditional eQTL study without any SNP prior information (italicized number).

TSS_{150kb}	DNase	Other	eGenes
100	1	1	**2493** $_{2449}$
100	10	1	2489 $_{2449}$
100	1	10	2473 $_{2413}$
10	1	1	2450
10	10	1	2331
100	100	1	2329
10	1	10	2060
1	10	1	2032
100	1	100	2014
10	100	1	1991
1	100	10	1904
1	100	1	1890 $_{1834}$
1	10	10	1747
1	100	100	1673
1	1	1	*1582*
1	1	10	1579
10	1	100	1548
1	10	100	1391
1	1	100	1280

impact than DNase sites. For the liver tissue, it is best to upweigh the cis-SNPs inside TSS_{150kb} by 60 times (Table 2.3).

For simplicity, in the following, we write one number for the weights w_2 and w_3; for example, we write $\mathbf{w} = [100, 1]$ instead of $\mathbf{w} = [100, 1, 1]$. As a sanity test, we simulate the false-positive and statistical power using $\mathbf{w} = [60, 1]$ to specify the SNP prior in the TSS_{150kb} region of the gene TCEA3 in Section 2.7 and 2.8. TCEA3 may has its own optimal $\mathbf{w}$ which may be different from $\mathbf{w} = [60, 1]$. $\mathbf{w} = [60, 1]$ is best when evaluated on the entire liver data and not specifically for gene TCEA3. For TCEA3, the false-positive rate at $\mathbf{w} = [60, 1]$ is not inflated, and the statistical power is not worse than the guess $\mathbf{w} = [100, 1]$ in Section 2.8 (Fig. 2.5).

Figure 2.5: TSS prior is applied to multi-threshold association study for eGenes. 60/1 indicates that relative weights are $c_i = 60c_j$ for $i \in \text{TSS}_{150kb}$ and $j \notin \text{TSS}_{150kb}$. Same logic applies for 100/1. (a) Quantiles of uniform density versus quantiles of many eGene p-values simulated under null hypothesis. (b) Simulated statistical power where the eGene p-values are computed by Mvn-sampling at two different prior options.



(a) **(b)**

2.11 Model histone modification sites

Histones are proteins which the DNA wraps around into compact form [53]. We consider the same histone and six of its modifications: H3K27ac, H3K4me1, H3K4me3, H3K9ac, H3K27me3, and H3K9me3. These modifications are of two major types, acetylation and methylation, and are associated with different levels of gene expression. For example, trimethylation of the histone H3 lysine 4 (H3K4me3) has been found to be correlated with promoter regions of active genes [11, 40, 89]. Different genomic regions have their own affinity for each modification type [99]. For eQTL studies, we want to nonuniformly weigh the effects of SNPs located inside and outside these regions. We do not have the computational resources to evaluate all the combinations of these annotation types. In this experiment, we separately measure the impact of each histone annotation so that we can at least provide some overview of which type is the most useful for discovering eGenes. We model whether SNPs are in a histone annotation or not, and then apply grid search to get the best weight $\mathbf{w} = [w_{\text{in site}}, w_{\text{not in}}]$ for this annotation (and not use any approximation strategy).

Modeling the importance of SNPs based on their presence in a histone annotation increases the number of discovered eGenes (Table 2.3). However, the effect is less prominent compared to modeling just the TSS_{150kb} region.

Table 2.3: Number of eGenes in the liver tissues (21,868 genes in total) for each SNP information type, in decreasing order. Weight $\mathbf{w} = [w_{\text{in site}}, w_{\text{not in}}]$ is found via grid search in the range $[1, 100]$ with increment of 10. Superscript ^{a} or ^{s} indicates that a histone modification is associated with gene activation or suppression. Upweighing SNPs in sites of histone modifications associated with gene activation increases the number of candidate eGenes. Ave. freq. is the fraction of cis-SNPs located in the annotation averaged over all the genes.

Type	Ave. freq. (%)	$w_{\text{in site}}$	$w_{\text{not in}}$	eGenes
TSS_{150kb}	14.74	60	1	**2479**
H3K27ac^a	12.25	40	1	1944
H3K4me3^a	7.73	50	1	1917
H3K27me3^s	7.26	1	70	1880
H3K9me3^s	11.92	1	50	1879
H3K9ac^a	9.74	100	1	1861
H3K4me1^a	16.38	80	1	1858
DNase	4.66	100	1	1834
None	100	1	1	1582

2.12 Summary and discussion

The flow of information from genetic diversity to phenotypic variations begins with the SNPs, to gene expression, to protein functions, and then to the phenotype of interest. Traditional GWAS focuses on the two endpoints, correlating the SNPs with the phenotype, and so cannot provide very detail explanation about the causal mechanism (e.g. which genes are most likely to be responsible for the trait). More importantly, significant SNPs from GWAS were found primarily in noncoding regions of the genome, suggesting that these SNPs are involved in regulations of gene expressions. In this chapter, we improve the traditional eQTL study that detects association for the cis-SNPs of a gene and the expressions of that gene. Our objective is to discover a high number of eGenes. eGenes are the genes that are not just statistically highly expressed but also possible candidates for future causal analysis of phenotypic variations (because expression of eGenes are linked to the

SNPs which differ for each person).

To discover more eGenes, we leverage extra information about the cis-SNPs. We upweigh SNPs in important regions so that these SNPs have higher chances of being found associated to the gene expression. We evaluate three types of region: TSS_{150kb}, DNase hypersensitive sites, and histone modification sites. DNase sites and histone modification sites are parts of the epigenomic information [99]. TSS_{150kb} region is not a part of epigenomic data, but it is inherently important because this is likely where the transcription factors begin transcribing the genes. For the liver tissue, TSS_{150kb} has the most impact versus DNase hypersensitive sites and histone modification sites.

From the technical view, our new model is built on our previous works designed for GWAS [20, 32]. We make two key changes in our new model. First, to reduce p-value computation time, we replace the permutation test with eGene-Mvn. Second, our new model takes more than one type of prior information on the SNPs. Currently, the SNP prior is modeled as a step-function, where the effect is $w_j \mathbb{1}_{\text{SNP in type } j}$. There can be more complex priors; for example, we can scale the SNP importance according to its distance to the TSS (the same logic applies for DNase hypersensitivity and histone modification sites).

We also want to discuss that, in all the experiments, we select the alternative SNP effect to be $\mu_s = 3.5$ in Eq. 2.5 for all the cis-SNPs. This number is the average of all the observed SNP effects in the liver data. In our previous studies, we empirically showed that different choices of the SNP effect size in the alternative hypothesis do not greatly affect the outcome [20, 32]. There is a way to select the SNP effect μ_s in the alternative hypothesis: we can assume some continuous prior density on μ_s and then integrate over its valid domain [8, 48, 49]. This idea is one interesting future research plan. In any case, our choice of $\mu_s = 3.5$ for the alternative hypothesis is not optimal, yet we have already discovered many more eGenes compared to the traditional approach. We expect that more sophisticated technique for solving Eq. 2.5 to perform even better.

From the application view, we are analyzing just one tissue dataset at a time, and still face the problem of low sample sizes for many tissues. In Chapter 3, we explain ways to analyze datasets with low sample sizes.

CHAPTER 3

Meta-analysis model to discover eGenes

3.1 Introduction

Chapter 2 explains how to discover eGenes in one tissue. In 2015, the GTEx data contains few samples in many tissues, where the median is 126 samples/tissue. Low sample sizes make it very hard to detect eQTLs, particularly eQTLs with low association strengths [33]. For this reason, it is not always possible to find many eGenes in each tissue. By combining multiple tissue datasets, we can increase the statistical power to identify eGenes in at least one tissues, which will provide additional useful knowledge.

To recap, a gene is defined as an eGene in a tissue when its expression in this tissue is associated with at least one of its cis-SNPs. To discover eGenes in each of the 44 tissues in the GTEx data, the GTEx consortium applied the tissue-by-tissue (TBT) procedure which comprises of independent eQTL studies performed for each gene in every tissue [98]. The type of eQTL study implemented is the traditional association study in Section 2.3. The number of discovered eGenes by the TBT procedure relies on how well the eQTL studies estimate the association strengths for the SNPs on the gene expressions.

When the GTEx consortium first started collecting data, many tissues did not have enough samples to reliably compute the SNP effects on the gene expressions. TBT method may fail to find true eQTLs, and the statistical power for eGene discovery would also decrease. Since then, there have been efforts in developing methods for finding the eQTLs by combining the expression datasets of this gene from many tissues. Intuitively, when combing many datasets, we increase the total sample size which should raise the statistical power for finding the eQTLs. This outcome in

turn will also improve the number of discovered eGenes for the whole GTEx data.

We first discuss the two main strategies for combining datasets from many tissues to find eQTLs. The first one combines the individual-level data of each tissue into one single larger dataset, and then measures the SNP effects for this larger dataset. The second idea applies meta-analysis to combine SNP effects which were already independently estimated for each tissue. Below we discuss a few methods for each approach, and explain the method that the GTEx consortium selected for their analysis.

Meta-Tissue and eQtlBma are two methods adapting the first strategy [35, 93]. Meta-Tissue combines individual-level data from many tissues into one large dataset, and then estimates the SNP effects on the expression level of a gene, by applying linear mixed model (LMM) with dependent variables specifying the tissue in which an observation comes from. Because Meta-Tissue solves for the LMM parameters for every SNP-gene pair, its runtime is not yet practical when there are thousands of genes in many tissues as in the GTEx data [93]. eQtlBma models a binary vector specifying the tissues in which a SNP is considered as an eQTL for a specific gene [35]. eQtlBma requires computing the probabilities for the 2^T possible configurations for all the cis-SNPs of a gene, where T is the number of tissues, and is also not yet practical for the GTEx data which has 44 tissues.

Metasoft is a software that follows the second strategy, and implements both fixed and random effects meta-analysis to combine the effects of the same SNP on the expression of the same gene in many tissues [41, 42]. Meta-analysis does not require individual-level data and can be applied to the SNP effects which the GTEx consortium has already computed for every SNP-gene pair in each tissue. Metasoft requires much less computation power than Meta-Tissue and eQtlBma, and can be applied to the 44 tissues in the GTEx data.

GTEx consortium has been using the random effects (RE) meta-analysis of Metasoft for discovering whether a gene is an eGene with respect to a set of body tissues [98]. Metasoft cannot determine the specific tissues in which a gene is classified as eGene. However, it is still meaningful to identify all the eGenes in the GTEx data, so that we have a list of all possible candidate genes affecting a specific disease. We will provide an example of this case in the result section.

Meta-Tissue, eQtlBma, and Metasoft assume that the same SNP has independent effect on the expression of the same gene in each tissue. Yet, the GTEx consortium has found that, in across tissues, the same SNP usually exhibits correlated effect sizes on the expression of the same gene [98]. In this chapter, we introduce a novel meta-analysis method (RECOV) that models this empirical evidence. RECOV is based on the RE meta-analysis, but the innovation is that RECOV has a covariance matrix to capture the correlations among the effect sizes of the same SNP on the expression of the same gene in many tissues.

Because the initial step for discovering eGenes requires us to find the eQTLs, we will first explain how the traditional TBT method, the RE meta-analysis, and our model RECOV find the eQTLs for a specific gene from datasets of many tissues. We then describe the subsequent steps needed for these methods to classify eGenes. We apply the three methods on all 44 tissues of the GTEx data, and analyze the cases for which one of these methods is more applicable than the other two. Our software is at `https://github.com/datduong/RECOV`.

3.2 Tissue-by-tissue analysis

From the notations in Chapter 2, suppose we have G number of genes and T number of tissues. The tissue-by-tissue (TBT) method applies the traditional eQTL study in Chapter 2.3 to the expression data of every gene $g \in G$ for each tissue $t \in T$ [98]. In tissue t, if at least one cis-SNPs of g are associated to its expression, then g is classified as an eGene in t. When gene g is expressed in T tissues, then TBT would perform T number of eQTL studies, one study per tissue.

Three levels of multiple testing correction are required because TBT applies one eQTL study per gene in each tissue in the whole GTEx data. The first layer is done within a tissue to correct for the LD of the cis-SNPs of g. Here, the GTEx consortium used the permutation test to estimate the eGene p-value of g in each tissue t. The second layer considers the fact that we test many genes in a tissue, possibly thousand of genes in each tissue. The GTEx consortium transformed the eGene p-values into eGene q-values to account for multiple testing across T total tissues [19, 98]. The third layer handles the fact that the same gene is tested T number of times, once per tissue

[93]. We apply Bonferroni correction where the false-positive rate threshold for the q-values in a tissue becomes α/T. There are other multiple testing correction strategies. However, evaluating the effects of different multiple testing correction methods is beyond our scope, as we will focus on developing meta-analysis method for eGene discovery.

3.3 Random effects meta-analysis for multiple eQTL studies

The statistical power for discovering eGenes depends on the statistical power for finding eQTLs of the genes. When a tissue has large enough sample size so that the SNP effects can be well estimated, then the TBT method works fine. The pilot GTEx data in 2015 however contains few samples for many tissues. One idea for increasing the statistical power of finding eQTLs is to increase the sample size, but collecting more samples is not always feasible. Instead, we can jointly analyze results of the datasets from many tissues, so that we would increase the total sample size when estimating the SNP effects on the gene expression. Meta-analysis provides the base solution for us to perform this exact task. Below, we explain the meta-analysis baseline to combine the individual eQTL results from many tissues, and then describe our new meta-analysis RECOV that models the correlation of the effects of the same SNP on the same gene in all the tissues.

We define the notations used in this chapter, some of which follow directly from Chapter 2. Suppose there are T number of eQTL studies (one per tissue) that measure the association of SNP s on the gene g. Like before, denote the observed effect of SNP s on gene g in tissue t as $\hat{\beta}_{sgt}$ which is computed by Eq. 2.1. Let $\hat{\boldsymbol{\beta}}_{sg} \in \mathbb{R}^T$ be the vector that contains the observed effects of the same SNP s on the same gene g in the T tissues, where $\hat{\boldsymbol{\beta}}_{sg} = [\hat{\beta}_{sg1} \ \ldots \ \hat{\beta}_{sgT}]^\intercal$. Let $\hat{\mathbf{V}}_{sg} = \mathbf{diag}(\widehat{\text{var}}(\hat{\beta}_{sg1}) \ \ldots \ \widehat{\text{var}}(\hat{\beta}_{sgT}))$ be the diagonal matrix containing the estimated variances of $\{\hat{\beta}_{sgt}\}_{t \in T}$.

The random effects (RE) meta-analysis assumes that the vector $\hat{\boldsymbol{\beta}}_{sg}$ is an instance of the random variable $\boldsymbol{\beta}_{sg}$ which has the following underlying generation [41, 100]

$$\boldsymbol{\beta}_{sg} = \boldsymbol{\lambda}_{sg} + \boldsymbol{\epsilon}_{sg} \tag{3.1}$$

The random variable $\boldsymbol{\epsilon}_{sg} \in \mathbb{R}^T$ represents the random sampling error assumed to be $\boldsymbol{\epsilon}_{sg} \sim \mathrm{N}(0, \boldsymbol{V}_{sg})$. The random variable $\boldsymbol{\lambda}_{sg} \in \mathbb{R}^T$ is known in the meta-analysis literature as the random effect and has the following distribution $\boldsymbol{\lambda}_{sg} \sim \mathrm{N}(\mu_{sg}\mathbf{1}, \tau^2_{sg}\mathbf{I})$ with $\mu_{sg} \in \mathbb{R}$, $\tau^2_{sg} \geq 0$, and $\mathbf{I} \in \mathbb{R}^{T\times T}$ is the identity matrix. μ_{sg} is the true shared value that the each effect of s on g in each tissue t inherits. τ^2_{sg} is the heterogeneity for the effects of the same SNP s on g in all T tissues. When τ^2_{sg} is near zero, the vector $\hat{\boldsymbol{\beta}}_{sg}$ will contain values closely centered at μ_{sg}. In this case, all the values β_{sgt} will be similar, indicating that there is low heterogeneity (assuming that $\boldsymbol{V}_{sg}$ is also close to zero). The interpretation is reversed when τ^2_{sg} is very large. Here, there is high heterogeneity because all the β_{sgt} will be distributed far from μ_{sg}. Based on this generation, $\hat{\boldsymbol{\beta}}_{sg}$ is an instance of the random variable $\boldsymbol{\beta}_{sg}$ that has the following the distribution

$$\boldsymbol{\beta}_{sg} \sim \mathrm{N}(\mu_{sg}\mathbf{1}, \tau^2_{sg}\mathbf{I} + \mathbf{V}_{sg}) \tag{3.2}$$

Now $\mathbf{V}_{sg}$ can be estimated as $\hat{\mathbf{V}}_{sg}$. There are two unknown parameters μ_{sg} and τ^2_{sg}. The RE model assumes that, under the null hypothesis the SNP s does not affect gene g in all T tissues, so that we have $\mu_{sg} = 0$. Han and Eskin [41] however showed that this hypothesis does not have optimal power for finding eQTLs, and introduced a modification to the RE meta-analysis, named RE2 in their software Metasoft. The RE2 null hypothesis states that suppose s does not affect g in T tissues, then $\mu_{sg} = 0$ and $\tau^2_{sg} = 0$. Under RE2 null hypothesis, the effect $\boldsymbol{\lambda}_{sg}$ is not a random variable but is a fixed at zero, and the observed $\hat{\boldsymbol{\beta}}_{sg}$ is attributed by just the random sampling error $\boldsymbol{\epsilon}_{sg}$ [41, 42].

The null hypothesis in RE2 model is $\mathrm{H}_0 : \mu_{sg} = 0, \tau^2_{sg} = 0$, and the log likelihood ratio to test this hypothesis is

$$\ell_{sg} = 2\log \frac{\sup_{\mu_{sg},\tau^2_{sg}} L(\hat{\boldsymbol{\beta}}_{sg}|\mu_{sg}, \tau^2_{sg})}{L(\hat{\boldsymbol{\beta}}_{sg}|\mu_{sg} = 0, \tau^2_{sg} = 0)} \tag{3.3}$$

The function L denotes the likelihood of $\hat{\boldsymbol{\beta}}_{sg}$ based on the multivariate normal distribution having the parameter μ_{sg} and τ^2_{sg} (Eq. 3.2). The numerator $\sup_{\mu_{sg},\tau^2_{sg}} L(\hat{\boldsymbol{\beta}}_{sg}|\mu_{sg}, \tau^2_{sg})$ can be estimated by derivative-based methods or other heuristic approaches. We apply the Nelder-Mead method which is a derivative-free heuristic approach. While finding the supremum, we need $\tau^2_{sg} \geq 0$, and this

restricted parameter space makes the asymptotic density of the likelihood ratio ℓ_{sg} to be an average of two chi-square distributions χ^2_1 and χ^2_2 (we will denote this mixed distribution as χ^2_{1+2}) [41, 84].

When the number of tissues T is large, this asymptotic density estimates the p-value of the likelihood ratio. Because GTEx pilot data in 2015 contains 44 tissues, we will use this asymptotic distribution of the likelihood ratio. Otherwise, we need to compute the p-value by doing a permutation test. For the k^{th} permutation, we permute the gene expression levels of the individuals in each tissue dataset, and then compute the test statistic $\ell^{(k)}_{sg}$. We repeat the permutation K total times to get the set $\{\ell^{(k)}_{sg}\}_{k\in K}$. The p-value of the observed ℓ_{sg} is its rank with respect to $\{\ell^{(k)}_{sg}\}_{k\in K}$. When this p-value is less than some significant threshold, then SNP s is an eQTL for the gene g in at least one of the T tissue.

3.4 RECOV: Random effects meta-analysis with covariance

The RE2 alternative hypothesis assumes that the random effects $\boldsymbol{\lambda}_{sg} \sim \mathrm{N}(\mu_{sg}\mathbf{1}, \tau^2_{sg}\mathbf{I})$ has a diagonal covariance matrix, specifying that the effects of SNP s on gene g in T tissues are independent. The GTEx Consortium [98] however observed that the same SNPs are consistently found as eQTLs for the same genes in many tissues. For instance, in their analysis, more than 50% of all detected eQTLs are common in the adipose, tibial artery, left ventricle of heart, lung, muscle, tibial nerve, skin, thyroid, and whole blood tissue. To capture this observation, our new model RECOV meta-analysis specifies a different distribution for the random effects where $\boldsymbol{\lambda}_{sg} \sim \mathrm{N}(\mu_{sg}\mathbf{1}, \boldsymbol{\Sigma}_{sg})$ and $\boldsymbol{\Sigma}_{sg}$ is non-diagonal. $\boldsymbol{\Sigma}_{sg} \in \mathbb{R}^{T\times T}$ represents the covariance of the effect sizes of SNP s on gene g in all T tissues. Due to symmetry, $\boldsymbol{\Sigma}_{sg}$ contains $T^2/2 - T$ unknown parameters. We assume a simpler form for $\boldsymbol{\Sigma}_{sg}$ and set $\boldsymbol{\Sigma}_{sg} = c_{sg}\mathbf{U}_{sg}$. $\mathbf{U}_{sg} \in \mathbb{R}^{T\times T}$ is unknown but can be estimated from the data without doing optimization. μ_{sg} and $c_{sg} \geq 0$ are unknown parameters to be optimized.

From the data, we estimate $\mathbf{U}_{sg}$ for each SNP-gene pair s, g as follows. Denote the matrix $\mathbf{B}_g \in \mathbb{R}^{T\times S}$ where $\mathbf{B}_g = [\hat{\boldsymbol{\beta}}_{1g} \ \ldots \ \hat{\boldsymbol{\beta}}_{Sg}]$. A column vector $\hat{\boldsymbol{\beta}}_{sg}$ in matrix $\mathbf{B}_g$ contains the observed effects of SNP s on gene g in all T tissues. We aim to use $\mathbf{B}_g$ to estimate $\mathbf{U}_{sg}$. When analyzing a SNP s, we need to remove its information from $\mathbf{B}_g$ to avoid circular reasoning. We partition the cis-

SNPs into ten separate continuous segments based on their physical locations on the chromosome, and then compute $\mathbf{U}_{sg}$ from the nine segments that do not contain s. Denote $\mathbf{B}_{-sg}$ as the matrix $\mathbf{B}_g$ without the columns corresponding to all the SNPs found in the same segment as the SNP s. $\mathbf{U}_{sg}$ is then computed by $\mathbf{U}_{sg} = \mathbf{B}_{-sg}\mathbf{B}_{-sg}^{\mathsf{T}}$ after applying proper scaling to $\mathbf{B}_{-sg}$. This computation follows the same principle for how a kinship matrix is estimated from the genotype data [33].

In RECOV, we still assume the SNP effects of s on g in T tissues to be $\boldsymbol{\beta}_{sg} = \boldsymbol{\lambda}_{sg} + \boldsymbol{\epsilon}_{sg}$. Unlike before, we now have $\boldsymbol{\lambda}_{sg} \sim \mathrm{N}(\mu_{sg}\mathbf{1}, c_{sg}\mathbf{U}_{sg})$ and $\boldsymbol{\epsilon}_{sg} \sim \mathrm{N}(0, \mathbf{V}_{sg})$. Based on this new generative process, the observed $\hat{\boldsymbol{\beta}}_{sg}$ is an instance of the random variable $\boldsymbol{\beta}_{sg}$ that has the following distribution

$$\boldsymbol{\beta}_{sg} \sim \mathrm{N}(\mu_{sg}\mathbf{1}, c_{sg}\mathbf{U}_{sg} + \mathbf{V}_{sg}) \tag{3.4}$$

Again, $\hat{\boldsymbol{V}}_{sg}$ estimates $\boldsymbol{V}_{sg}$, and μ_{sg} and c_{sg} are unknown parameters. RECOV null hypothesis is the same like in RE2 meta-analysis that is, $\mathrm{H}_0 : \mu_{sg} = 0, c_{sg} = 0$ to imply that s does not affect g in all T tissues (e.g. s is not an eQTL of g in all T tissues). The log likelihood ratio to test this hypothesis becomes

$$\ell_{sg} = 2\log \frac{\sup_{\mu_{sg},\, c_{sg}} L(\hat{\boldsymbol{\beta}}_{sg}|\mu_{sg},\, c_{sg}, \mathbf{U}_{sg})}{L(\hat{\boldsymbol{\beta}}_{sg}|\mu_{sg} = 0, c_{sg} = 0)} \tag{3.5}$$

Similar to RE2 meta-analysis, to find the supremum in the numerator, we need $c_{sg} \geq 0$. This restricted parameter space makes the asymptotic density of ℓ_{sg} to be an average of the two chi-square distributions χ^2_1 and χ^2_2. Otherwise, we can compute the empirical p-value of ℓ_{sg} with the permutation test where in every permutation we would estimate a different $\mathbf{U}_{sg}$. Intuitively, the numerator in Eq. 3.5 has three unknown parameters, and the permutation test accounts for all their possible values. When the p-value of ℓ_{sg} is less than a specified significance threshold, then SNP s is an eQTL of the gene g in at least one tissue.

3.5 Discover eGenes from meta-analyses of eQTL studies

So far, we have presented meta-analyses for finding eQTLs in multiple tissues. Now, for discovering eGenes in multiple tissues, we explain the steps required after the meta-analyses. These steps are

applicable with both RE2 and RECOV meta-analysis. In Chapter 2.3, conditioned on the expression of gene g in a tissue t, the eGene test statistic is $p_{gt} = \min\{p_{sgt}\}_{s \in S}$ where p_{sgt} is the p-value of the effect of SNP s on the expression of g.

Now, in meta-analysis we will define the eGene test statistic for g to be $p_g = \min\{p_{sg}\}_{s \in S}$ where p_{sg} is the p-value of ℓ_{sg}. p_g does not need the subscript t as our interpretation is with respect to the entire set of T tissues. We interpret g to be an eGene in at least one of the T tissues when its eGene test statistic p_g is significant. Before testing whether p_g is significant, we emphasize that meta-analysis models need just two layers of multiple testing correction, whereas TBT method requires three layers. In meta-analysis, the first layer is the same as TBT approach and corrects for the LD of the cis-SNPs. The second correction is applied to the gene set G because there are many genes in the GTEx datasets being tested at once.

To test whether p_g is significant, we compute its eGene p-value α_{p_g} via a permutation test which controls for first multiple testing correction, that is, the effect caused by LD of the cis-SNPs of g [26, 94, 98]. In the permutation iteration $k \in K$, we permute the expression values of g for the individuals in each of the T tissues so that there are T randomized datasets. This approach reflects the hypothesis that g is not an eGene in any of the tissues. Next, we perform the meta-analysis at each cis-SNP of g so that we have a new $p_g^{(k)} = \min\{p_{sg}^{(k)}\}_{s \in S}$. The eGene p-value α_{p_g} is the fraction of times the observed p_g is less than the permutation values $\{p_g^{(k)}\}_{k \in K}$. The gene g is an eGene in at least one tissue if its eGene p-value α_{p_g} is below some threshold.

Because there are G number of genes for each tissue, to control for the overall false-positive rate α on the entire dataset, we apply the second layer of multiple testing correction on the set of eGene p-values $\{\alpha_{p_g}\}_{g \in G}$. Here, we apply Bonferroni correction where a gene $g \in G$ having $\alpha_{p_g} < \alpha / G$ is classified as an eGene in at least one of the T tissues. Bonferroni correction is easy to implement, but the permutation test to handle LD of SNPs can have very high runtime. In the next section, we discuss an efficient way to approximate this permutation test.

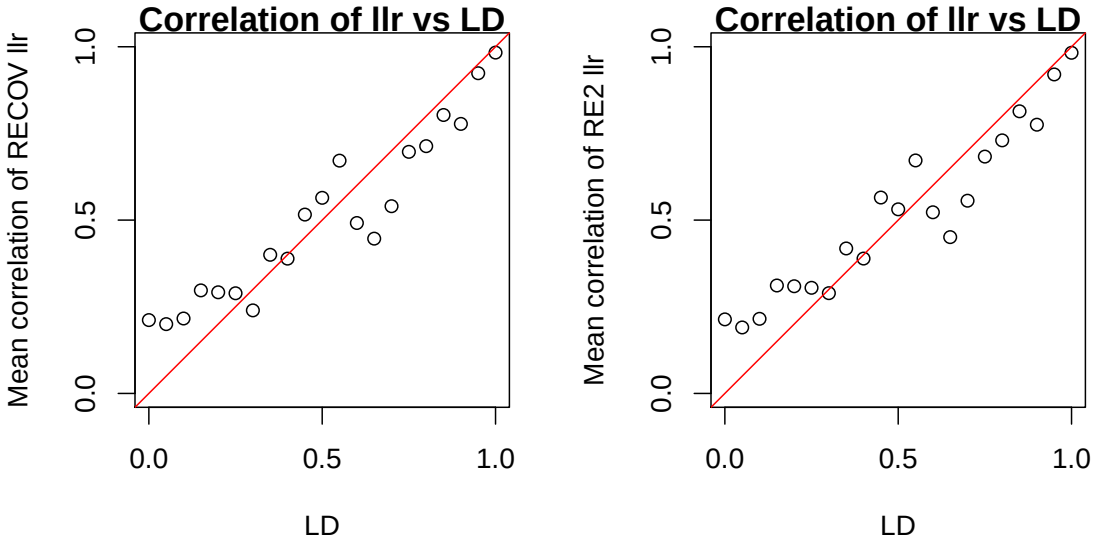
3.5.1 LD-corrected eGene p-value in meta-analysis

The permutation test must be performed K times for each cis-SNP of each gene $g \in G$ in every tissue $t \in T$ and would require about $KSGT$ permuted datasets which is very time consuming, where S is the number of cis-SNPs per gene (for the sake of simplicity, in this section, we assume each gene has the same number of cis-SNPs). Instead of the permutation test, we introduce an another way to estimate the meta-analysis eGene p-value. Within the same set of cis-SNPs, the permutation test is meant to remove the effects of LD. To approximate the permutation test, we relate the LD of the cis-SNPs of g with their likelihood ratio p-values $\{p_{sg}\}_{s\in S}$. At each SNP-gene pair, we can work with either the likelihood ratio ℓ_{sg} or its corresponding p-value p_{sg}, because these two entities have one-to-one and inverse relation. An eGene test statistic p_g can then be defined as $\max\{\ell_{sg}\}_{s\in S}$ or $\min\{p_{sg}\}_{s\in S}$. Importantly, having a null distribution of $\max\{\ell_{sg}\}_{s\in S}$ is the same as having a null distribution of $\min\{p_{sg}\}_{s\in S}$. The eGene p-value will be the same in both cases.

Suppose for now, we are using ℓ_{sg} instead of p_{sg}, so that $p_g = \max\{\ell_{sg}\}_{s\in S}$ for the rest of this section. We draw inspiration from Han *et al.* [44] who showed that the correlation of the SNP effects at two SNPs on the same gene is about equal to their LD. Empirically, we also observe that the correlation of the likelihood ratios at any two SNPs is about equal to their LD; that is, on average $\text{cor}(\ell_{ug}, \ell_{vg}) \approxeq \text{LD}(u, v)$ for any SNPs $u, v \in S$ (Fig. 3.1).

The permutation test can be viewed as function that maps a test statistic to its p-value while accounting for LD of SNPs. From the observations that p_g is defined as $\max\{\ell_{sg}\}_{s\in S}$ or $\min\{p_{sg}\}_{s\in S}$, and that $\text{cor}(\ell_{ug}, \ell_{sg}) \approxeq \text{LD}(u, v)$ for any $u, v \in S$, suppose we can find a function f that estimates the p-value of any entity $a^m = \max\{a_s\}_{s\in S}$ and accounts for the correlation $\text{cor}(a_u, a_s)$ for any $u, s \in S$. Then, we can apply f to $p_g = \max\{\ell_{sg}\}_{s\in S}$ to compute α_{p_g} instead of the permutation test. We emphasize that $\{a_s\}_{s\in S}$ represents a set of any items; we do not care what are these items, and are concerned with only the p-value of the maximum of this set. Each gene g has its own LD structure based on its own cis-SNPs and will require its own function f. To estimate f at a gene g, we apply eGene-Mvn in Section 2.5.1. eGene-Mvn determines whether a gene is an eGene in just one tissue and does not have an extension for analyzing many tissues [94].

Figure 3.1: $\text{cor}(\ell_{ug}, \ell_{vg})$ for two SNPs u, v versus their LD. We select many SNP pairs that co-occur as cis-SNPs in at least two genes. Different pairs are grouped into bins by their LD (bin-width 0.05). We compute ℓ_{ug}, ℓ_{vg} for each SNP pair in every gene g for which they are cis-SNPs. Next, we estimate $\text{cor}(\ell_{ug}, \ell_{vg})$ for each pair u, v, and then average $\text{cor}(\ell_{ug}, \ell_{vg})$ for all pairs u, v in each LD bin. We plot the absolute value of this average against the LD bin. In RECOV and RE2, $\text{cor}(\ell_{ug}, \ell_{vg})$ for two SNPs aligns well with their LD.



Below, we explain how to estimate the function f with eGene-Mvn.

In eGene-Mvn, conditioned on tissue t, the test statistic of gene g is the most significant effect size taken from all its S cis-SNPs. From Chapter 2, this test statistic is $b_{gt} = \max\{|\hat{\beta}_{sgt}|\}_{s \in S}$. The p-value of b_{gt} depends on the LD of the cis-SNPs of g. Instead of doing a permutation test to compute this p-value, in the iteration k, eGene-Mvn draws the vector of effect sizes $[\hat{\beta}^{(k)}_{1gt} \cdots \hat{\beta}^{(k)}_{Sgt}]$ for the cis-SNPs from $\text{Mvn}(0, G_g)$ where $G_g \in \mathbb{R}^{S \times S}$ is the LD of the cis-SNPs. After K iterations, $\{b^{(k)}_{gt}\}_{k \in K}$ will form a null distribution of b_{gt} which is then used to compute the p-value of b_{gt}.

Because the normal CDF maps $\hat{\beta}^{(k)}_{sgt}$ into its p-value $p^{(k)}_{sgt}$, the null distribution of b_{gt} can be transformed into the null distribution of $p_{gt} = \min\{p_{sgt}\}_{s \in S}$. p_{gt} null distribution handles the LD of the SNPs and also specifies a null distribution for the minimum in a set of p-values. In this case, the subscript t may bare little significance for the following reason. Many tissues in the GTEx data contain samples from the same individuals, and many donors are of European ancestry. We assume that the LD of cis-SNPs of a gene does not change very much from tissue to tissue. It is

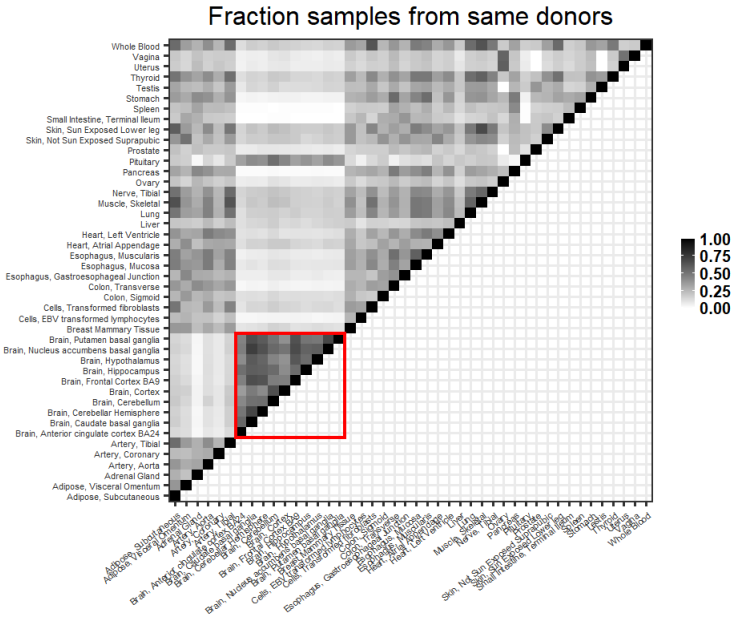
not too important which tissue t is chosen; we only require that there are enough samples in that tissue. We will use the subcutaneous adipose tissue that has 298 individuals.

For the meta-analysis eGene test statistic p_g, we reason that the null distribution of p_{gt} can compute the p-value α_{p_g} (where t is the adipose tissue). Section 3.6 shows that our approximation controls for the false-positive rate of the meta-analysis.

3.5.2 Remove effect of overlapping samples among tissues

Three main factors affecting the meta-analysis false positive rate are: batch effects in the datasets, LD of SNPs, and shared individuals in multiple tissue datasets. Batch effects were already removed from the GTEx data by applying PEER factors to each tissue dataset [98]. Section 3.5.1 explains how we handle LD of SNPs in a meta-analysis. The GTEx consortium collected samples of different body tissues from the same person, so that distinct tissue datasets are statistically related because they share data from the same donors (Fig. 3.2). RE2 and RECOV meta-analysis have high false positive rate when applied to the GTEx data (similar phenomena was seen for GWAS meta-analysis [45]) (Fig. 3.3).

Figure 3.2: Fraction of samples coming from the same donors for two tissue datasets. Brain tissues contain many samples from the same donors (red box).



We correct for this problem by computing a genomic control (GC) factor to scale the eGene test statistics so that their null hypothesis p-values are not inflated [24]. We generate and compare the behaviors of two kinds of dataset, named A and B. Type A is free of all confounders affecting the meta-analysis false positive rate. Type B has just the effect due different tissue datasets sharing samples from the same donors, where the number of samples shared by two datasets is the same as the GTEx data. We simulate the genotypes and the gene expressions in both type A and B so that these numbers are independent of the values in the GTEx data which helps reducing the problem of reusing the data.

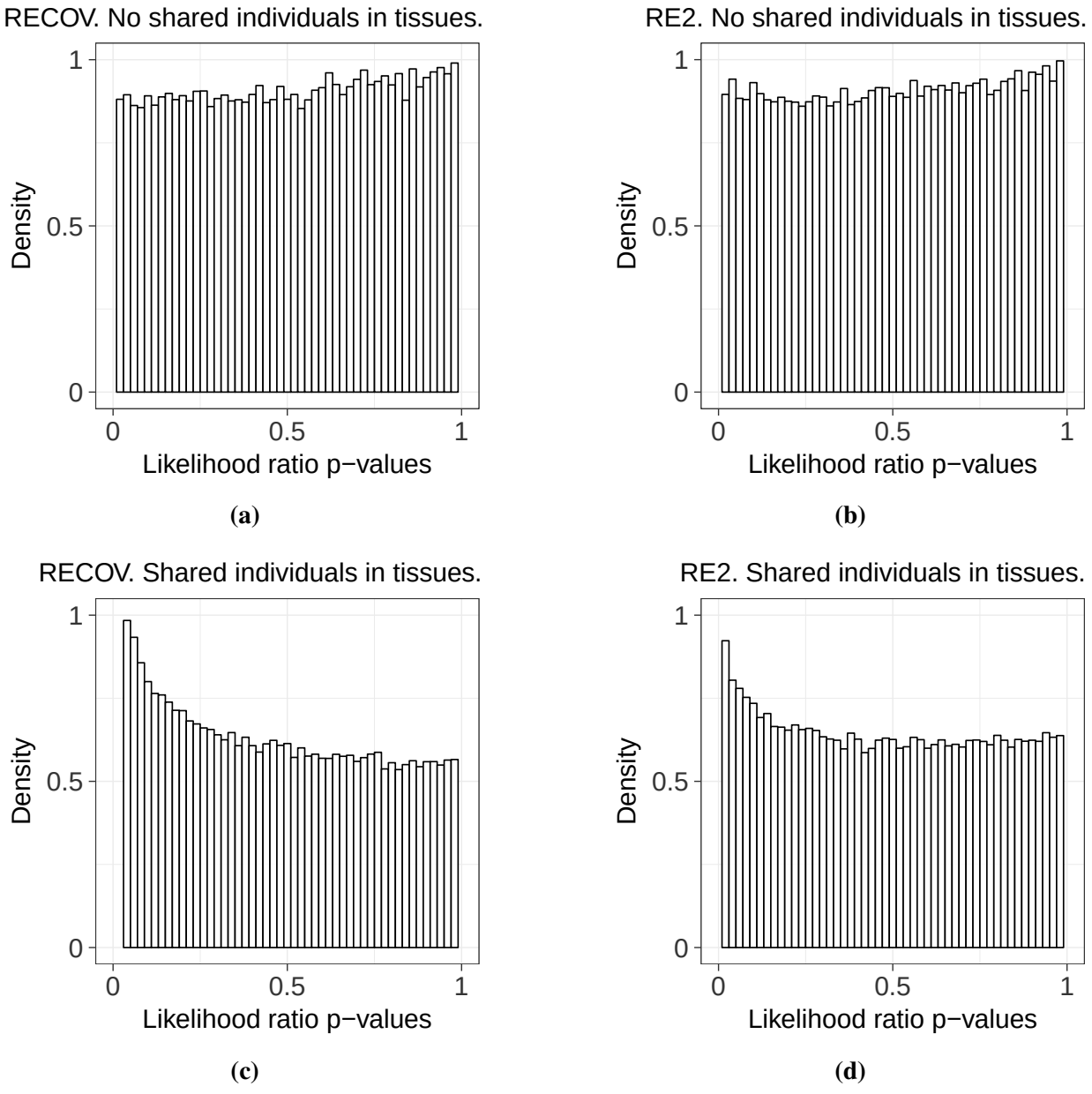
We simulate 1000 datasets of type A. In each simulation, the number of people per tissue is taken from the GTEx data, but the tissue data do not have samples coming from the same individuals. Each time, we simulate 1000 independent SNPs with different minor allele frequencies and the expression values of a gene g for the people in each tissue. Next, we compute the meta-analysis p-value of ℓ_{sg} at each SNP s. In this simulated data, SNPs and gene expressions are independent so that the meta-analysis p-values of the 1000 SNPs should form a uniform distribution. Over the 1000 datasets, the histogram of the 1 million simulated p-values (1000 datasets, each having 1000 SNPs) confirms this expectation (Fig. 3.3a, 3.3b).

Next, we simulate 1000 datasets of type B, each having 1000 SNPs, to observe the effect caused by distinct tissue datasets having samples from the same individuals. The genotypes and gene expressions are simulated as in type A. Again, we compute the p-values of ℓ_{sg} for the SNPs to obtain a total of 1 million simulated p-values. We observe that the meta-analysis false-positive rate is much higher than expected, where histograms of the simulated p-values shift toward zero (Fig. 3.3c, 3.3d).

We now compute the GC factor to remove the effect caused by multiple tissues sharing samples from the same donors. Following Devlin and Roeder [24], the median of the p-values simulated from type B datasets are transformed into a random variable z having a chi-square distribution. From Section 3.3, this chi-square distribution is the distribution χ^2_{1+2} which is a weighted average of two chi-square distributions χ^2_1 and χ^2_2. Next, we calculate a scaling constant for z, converting it into another random variable $\tilde{z}$ corresponding to the p-value at 0.50 under the distribution χ^2_{1+2}.

The GC factors for the RE2 and RECOV meta-analysis are 1.1045 and 1.2947, respectively.

Figure 3.3: Under the null hypothesis, (a) RECOV and (b) RE2 meta-analysis are applied to multiple tissue datasets where no two datasets contain samples from same individuals. Histograms of p-values of ℓ_{sg} are uniformly distributed, indicating equal chance of observing any p-value in the range [0,1]. (c) RECOV and (d) RE2 meta-analysis applied to multiple tissue datasets where samples in two tissue datasets may come from the same individuals (fraction shared are taken from GTEx data). Histograms of p-values of ℓ_{sg} under the null hypothesis shift toward the left side, showing that we are more likely to see significant p-values.



3.6 False-positive rate simulation

We evaluate the meta-analysis false positive rate for just one SNP-gene pair s, g across T tissues. The eGene test statistic is ℓ_{sg}. We test this simple case because the overall false positive rate for the entire GTEx data depends on this simple case.

For one SNP s, we simulate 1000 datasets under the null hypothesis that s does not affect the expression of g in all 44 tissues of the GTEx data. We first select a SNP s (any SNP will suffice) from the GTEx data, and then use its minor allele frequency to generate the genotypes in each of the 1000 datasets. Next, we assign for each tissue the same number of individuals as in the GTEx data. Every two tissues will also contain samples coming from the same donors, where the numbers are the same as in the GTEx data. Hence, most tissues will share the same genotypes for some samples. To simulate the gene expression, we follow Sul *et al.* [93] and generate the expression of the same gene in a person to be correlated in any two tissues, where the average correlation is 0.50. Expression of the same gene in an individual can be correlated across tissues, because these tissues are exposed to the same environmental factors.

In each of the 1000 datasets, we conduct eQTL study to measure the SNP effect $\hat{\beta}_{sgt}$ and its variance $\widehat{\text{var}}(\hat{\beta}_{sgt})$ in each tissue t. Then, we apply the meta-analysis on the eQTL results $[\hat{\beta}_{sg1} \cdots \hat{\beta}_{sgT}]$ and compute the meta-analysis p-value α_{p_g}. Here, α_{p_g} is computed by our approximation method and not the permutation test, because we want to see that our estimation producing valid false positive rate. Also, we apply the GC factor to transform α_{p_g} in each simulation, so that we remove side effects caused by tissue data sharing samples from the same donors. Over the 1000 datasets, the meta-analysis false positive rate is the fraction of times the simulated p-values are less than the threshold $\alpha = 0.05$.

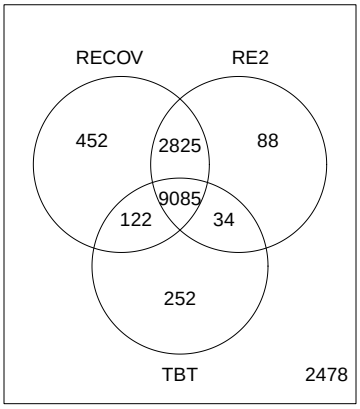
We repeat the same analysis above but for 1000 different SNPs each having its own minor allele frequency, so that we will have 1000 numbers of meta-analysis false positive rate (one number per SNP). For each SNP, we generate the gene expression as described above. We run RE2 and RECOV meta-analysis on all the simulated datasets. In both meta-analyses, the median false positive rate is 0.05. In RE2, the 75^{th} and 95^{th} quantiles for the 1000 numbers of meta-analysis false positive rate

are 0.07 and 0.10. In RECOV, the same numbers are 0.06 and 0.09. We do not expect our p-value estimation to always return the false positive rate at 0.05. However, the simulation study shows that most of the time our approach obtains correct false positive rates in realistic settings.

3.7 Application to the GTEx data

We apply the traditional TBT method, the RE2 meta-analysis of Metasoft, and our model RECOV to the GTEx Dataset in Chapter 2. We analyze the 15,336 genes expressed in all the 44 tissues. Batch effects on the gene expressions were removed by the the GTEx consortium [98]. We define cis-SNPs of a gene as the SNPs located within 1 Megabases from its transcription start site [98]. We consider only cis-SNPs of genes where the genotypes are not missing in any of the tissues. The median for number of genotyped cis-SNP per gene is 3744. The GTEx consortium already computed the SNP effect $\hat{\beta}_{sgt}$ and its variance $\widehat{\text{var}}(\hat{\beta}_{sgt})$ for each cis-SNP of a gene in each tissue t.

Figure 3.4: Venn diagram of the numbers of eGenes found by TBT method, RE2 and RECOV meta-analysis.



In TBT method, we apply eGene-Mvn to compute the p-value of the test statistic $b_{gt} = \max\{|\hat{\beta}_{sgt}|\}_{s\in S}$ at each gene g. eGene-Mvn just removes the effect of LD of SNPs, so we implement the other two layers of multiple testing correction: transforming the eGene p-values into q-values, and using the Bonferroni significance threshold 0.05/15336 for each q-value.

Meta-analysis takes two inputs, the vector of SNP effect sizes $\hat{\boldsymbol{\beta}}_{sg} = [\hat{\beta}_{sg1} \ \ldots \ \hat{\beta}_{sgT}]^\intercal$ and the matrix $\boldsymbol{V}_{sg} = \textbf{diag}(\widehat{\text{var}}(\hat{\beta}_{sg1}) \ \ldots \ \widehat{\text{var}}(\hat{\beta}_{sgT}))$ where $T = 44$. At a cis-SNP s, we compute the

likelihood ratio ℓ_{sg} and its p-value p_{sg} by using the distribution χ^2_{1+2}. Then, we transform this p_{sg} with the GC factor to remove effects caused by tissues having samples from the same people. The eGene test statistic for g is the minimum of the transformed p_{sg} taken across all its cis-SNPs. The eGene p-value is approximated as described in Section 3.5.1. We use the Bonferroni significance threshold 0.05/15336 for each eGene p-value. We do not transform the eGene p-values into q-values because we jointly analyze the same gene in all the tissues rather than separately analyzing each gene in each tissue.

In the entire GTEx data, many genes are predicted to be eGenes (at least 61.90%). This result is somewhat expected, because we analyze many tissue datasets and, even with multiple testing correction, it is possible that the same gene will have at least one eQTL in one of the 44 tissues. RECOV and RE2 meta-analysis discover about 20% more than TBT baseline, indicating that there is an advantage for meta-analysis over the simpler per-tissue approach (Fig. 3.4). RECOV meta-analysis discovers the most number of eGenes, and we observe a modest improvement over the RE2 method. From the 15336 genes expressed in the 44 tissues of the GTEx data, the fraction discovered as eGenes by TBT, RE2 and RECOV are 61.90%, 78.45%, and 81.40% respectively.

3.8 Case studies

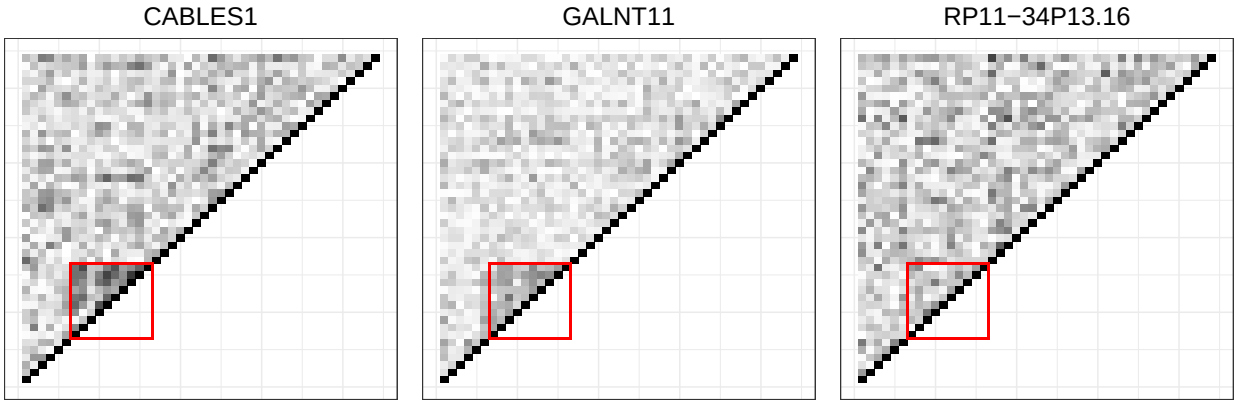
There are scenarios for which one of the TBT baseline and RE2 and RECOV meta-analysis is more advantageous than the other two. The TBT baseline independently analyzes each tissue data, and we can know the tissues in which a gene is found as eGene. From the 252 eGenes discovered by only the TBT approach, the numbers of genes which are eGenes in exactly 1, 2 and 3 tissues are 225, 25 and 2, respectively. We suspect that when a gene happens to be eGene in only very few tissues, then the aggregated information from many other tissues may still indicate that the meta-analysis effect size μ_{sg} should be about zero. In this case, the meta-analysis would not be able to discover this type of eGenes.

Next, we analyze an example where RECOV meta-analysis is more applicable than RE2 meta-analysis and the TBT baseline. We consider the gene CABLES1 that is expressed mostly in nine

of the brain tissues, where the median sample size is only 90 individuals. The GTEx pilot analysis which implemented the TBT approach, did not detect eQTLs for this gene in any of the tissues. It makes sense that meta-analysis would find CABLES1 to be eGene in at least one of the 44 tissues, because meta-analysis aggregates samples from tissue datasets to enhance association strengths of SNPs and gene expression, which in turn increases the power for discovering eGenes.

But why does RECOV meta-analysis predict CABLES1 to be eGenes, and not RE2 model? We observe that there are significant correlations of SNP effects for CABLES1 expression in all the tissue datasets, especially in the brain tissues (Fig. 3.5a). RECOV meta-analysis models this exact evidence and so is more applicable than RE2 meta-analysis. Or statistically speaking, RECOV meta-analysis more accurately models the alternative hypothesis and then produces a lower p-value for CABLES1 than the RE2 approach (the two eGene p-values are $4.94E^{-13}$ and $5E^{-5}$).

Figure 3.5: Correlations of SNP effects for the expressions of (a) CABLES1 (b) GALNT11 and (c) RP11-34P13.16 in 44 tissues (tissue names omitted). Correlation is computed by $\mathbf{B}_g\mathbf{B}_g^\intercal$, similar to how genetic kinship is computed. We indicate correlation values for the brain tissues in red.



We emphasize that we finished this work in Dec 2016. In Aug 2017, Hernández-Ramírez *et al.* [46] discovered that loss-of-function mutations in the CABLES1 gene are a novel cause of Cushing's disease, which is a condition affecting the brain tissue anterior pituitary. Suppose we had used TBT approach or RE2 meta-analysis instead of RECOV meta-analysis, then we would have missed that CABLES1 is an important gene, which means that we would not have recommended CABLES1 for any further causal analysis of any diseases.

There are cases for which RECOV meta-analysis is less applicable than the RE2 model. For instance, we consider the gene GALNT11 whose correlation patterns for the effects of cis-SNPs in the GTEx data are very similar to CABLES1, where the strongest correlation block happens for the brain tissues. However, on average, all the correlation values are much lower than those for CABLES1 (Fig. 3.5b). GALNT11 is expressed mostly in the frontal cortex and modestly in the other brain tissues.

The TBT approach in the GTEx pilot analysis detected eQTLs for the GALNT11 expression in only the frontal cortex, but these eQTL effects produced an eGene q-value of 0.0189 which is much higher than the TBT overall significant threshold. We expect that a meta-analysis, which combines information from other similar brain tissues as the frontal cortex, should increase the association strengths for the SNPs and gene expression, and then find that GALNT11 is an eGene. This is indeed the result for RE2 meta-analysis but not RECOV. Because the correlations of the SNP effects on GALNT11 are low, the likelihood ratio ℓ_{vg} of GALNT11 possibly does not strictly require $c_{sg} > 0$ in the term $c_{sg}\mathbf{U}_{sg}$ (Fig. 3.5b). RE2 alternative hypothesis would be a better fit than that of RECOV. The eGene p-values of GALNT11 by RECOV and RE2 meta-analysis are 3.50E^{-4} and 7.08E^{-8} respectively.

We now analyze the pseudogene RP11-34P13.16 which is an interesting case. RP11-34P13.16 is expressed highly in just a few tissues (highest in Whole Blood) and very weakly in many other tissues in the GTEx data. In the tissues where RP11-34P13.16 is weakly expressed, the effects of cis-SNPs on RP11-34P13.16 expression are near zero, and the TBT approach in the GTEx pilot analysis did not find eQTLs for RP11-34P13.16. In the tissues where RP11-34P13.16 is highly expressed, the TBT approach also did not find any eQTLs. RP11-34P13.16 is classified as an eGene by just RE2 meta-analysis and not RECOV. The RECOV and RE2 meta-analysis eGene p-values are 1.50E^{-4} and 1.37E^{-8} respectively. SNP effects on RP11-34P13.16 and CABLES1 have comparable correlation structures, except for the strong association pattern within the brain tissues (Fig. 3.5c). It is likely that the term $c_{sg}\mathbf{U}_{sg}$ in RECOV, which tries to model trends among the correlations, fails to fit well for the SNP effects on RP11-34P13.16. Here, RE2 appears more robust despite its simple assumption, which says that the same SNP does not have correlated effects

on the same gene expression in multiple tissues.

3.9 Summary and discussion

Meta-analysis increases the statistical power of a hypothesis test, by combining the results of multiple independent studies [79]. Usually, each study has low sample sizes so that on its own, the individual result may not be representative enough for the whole population. In the last decade, meta-analysis has become popular in the GWAS and eQTL literature, because various research groups may analyze the same phenotype with their own datasets, where each dataset may not have a lot of samples [6, 22, 61, 64]. Meta-analysis is appealing because of its simplicity, as it requires just the outcomes of different experiments and not the entire datasets at the individual-level.

In the context of GTEx data, meta-analysis combines the effects sizes of the same SNP on the expression of the same gene in all the tissues. The joint outcome has higher statistical power than each individual eQTL study for that SNP in each tissue, and more confidently determines whether the SNP is an eQTL for that gene. Our application of meta-analysis can only classify that a gene is an eGene in at least one tissue in the GTEx data. Although we may not know the particular tissues in which the gene is an eGene, at least we will not miss a lot of candidate genes for a phenotype of interest.

We introduce a new random effects meta-analysis (RECOV) that has a covariance parameter to capture the relatedness of the SNP effects on the gene expressions in different tissues. This covariance is the term $\mathbf{U}_{sg}$ in the alternative likelihood of Eq. 3.5. Our formulation of $\mathbf{U}_{sg}$ may be suboptimal with respect to this likelihood function, yet we are able to discover the most number of eGenes in the entire GTEx data compared to the RE2 meta-analysis and the TBT approach. We expect more complex formulations to have even better outcomes. For future work on meta-analysis of eQTL studies, the problem of selecting the most suitable $\mathbf{U}_{sg}$ is one interesting topic.

At the time of our publication, the GTEx data contains very few samples for many tissues. Over the last 5 years, more data has been collected, yet the number of samples remains low compared to other association studies. For example, in late 2015 the liver tissue had 97 individuals, and this

number in April 2020 is just 226 (with 208 people genotyped so far). In contrast, the UK Biobank data has 500,000 individuals genotyped [13]. We expect that meta-analysis will be applicable for the GTEx data for some foreseeable future.

CHAPTER 4

Learning embeddings of Gene Ontology terms

4.1 Introduction

One way to study functions of a protein is through laboratory work to observe the protein 3D structure, its location in the cell, the chemical reactions it participates in, and the other proteins that it interacts with. The discovered functions are written down following the rules provided by the Gene Ontology (GO) consortium which provides a database of standardized vocabularies (like a dictionary), where each vocabulary has a definition explaining the biological events that it represents [38]. These vocabularies formally known as *GO terms* have hierarchical relationships. Terms describing more specific biological functions are child nodes of more generic terms (Fig. 1.1).

A typical protein is assigned with three types of GO terms: Biological Processes (BP) which identify events that involve this protein (e.g. cell division), Molecular Functions (MF) which specify the types of reactions the protein induces (e.g. kinase binding), and Cellular Components (CC) which determine the protein's location (e.g. mitochondria) [96]. To aid the manual annotation, automatic GO classifiers have been introduced. Many of them have the primary input as the protein amino acid sequence, and secondary inputs as the protein metadata, such as 3D structure information and protein-protein interaction data (ideally inferred without laboratory experiment). These annotation methods return the output as a set of GO labels (the canonical terminology is GO terms, but we will use *GO labels* in the context of classification). Because each protein is assigned many GO labels, predicting protein functions is a multi-label classification problem.

This multi-label classification problem is supervised learning and thus faces two issues. First,

supervised model works well when each GO label annotates a lot of samples. The model cannot accurately predict labels occurring just a few times or only in the testing samples; for example, the model may not accurately predict the label *bioluminescence* annotates just 99 proteins out of the 561,911 manually reviewed proteins in the Uniprot database [96]. Second, as new protein functions are being discovered, the GO database is constantly being updated with terms being added and removed. A supervised classifier will have to be retrained for each update in the ontology, which can be time-consuming for many deep learning models. These two problems are not unique to GO database; for example, for electronic health records, some diseases are rarely seen, and disease catalogs are always being updated [72, 81].

To mitigate these two problems, literature on multi-label classification have suggested integrating metadata about the labels into the classifier [72, 81]. To use the GO label metadata, we need to transform this data into vector representations based on the definitions or hierarchy relation of the labels. For example, we would transform *bioluminescence* and its parent *cellular metabolic process* into similar vectors. Next, we fit the model using these vectors as feature inputs. For a new protein, we can estimate the likelihood of a rare or unseen label based on the predictions of the other labels which have similar vector representations to the label of interest (assuming these other labels are well predicted) [87].

In the context of GO database, there are two types of metadata, the definitions of the GO terms and their hierarchical relationships in the database. The classification accuracy depends on how well these types of metadata are transformed into vectors. Ideally, related terms (e.g. having comparable definitions or parent-child nodes) should be encoded into similar vectors, and vice versa. Below we present existing ways to map GO metadata into vectors, and introduce new techniques based on recent advances in deep learning. Before going forward, we will define *GO embeddings* as the vectors representing the GO terms, and *GO encoder* as the method that produces the GO embeddings. Typically, GO encoders either embed the definition of a GO term, or the term itself as one single entity (e.g. GO name) based on its topology in the GO hierarchy. We will refer to these two types as *GO definition encoders* and *GO entity encoders*, respectively.

Embedding GO terms via neural networks is a recent research topic. Some of the earliest GO

encoders were introduced in mid 2018. In 2018, we designed a Bidirectional Long-short Term Memory (BiLSTM) encoder to embed the definitions of GO terms [27]. In the same year, another method Onto2vec was also introduced which applied Word2vec skip-gram model on axioms such as "GO:0060611 is_subclass GO:0060612" to capture relatedness of the GO names [71, 86]. Our BiLSTM encoder and the Onto2vec were not directly compared. More importantly, neither method was integrated into an existing GO classifier that predicts labels for an input amino acid sequence. Until now, it is unknown for which scenarios will a GO classifier gain accuracy from having the GO metadata as extra feature.

This chapter addresses the first question proposed in Section 1.2, that is, are there useful metadata about the GO labels that can improve the accuracy for protein function classification? In particular, we are interested in how to best embed and compare the two types of GO metadata—the definitions of the GO terms and their hierarchical relationships in the ontology. We will not just evaluate our BiLSTM encoder and the Onto2vec, but will also introduce new GO encoders built from popular neural network approaches of Graph Convolution Network (GCN) [55], Embeddings from Language Models (ELMo) [78] and Bidirectional Encoder Representations from Transformers (BERT) [25]. GCN, ELMo and BERT are key components in many machine learning methods, but have not yet been implemented to produce GO embeddings. GCN, ELMo and BERT have their own characteristics. GCN is an entity encoder related to Onto2vec, and applies convolutional neural network to the terms on the GO hierarchy. ELMo and BERT are definition encoders just like BiLSTM. ELMo uses two BiLSTM layers, where the first layer is the input of the second layer. BERT however does not apply BiLSTM or convolutional neural network, but rather relies on the Transformer architecture to model all pairwise interactions of the words in the definition of a GO term [101].

We design three tasks to compare the types of GO embeddings from the encoders presented above. Task 1 compares the similarity score distributions for the embeddings of child-parent GO terms versus two unrelated terms. This task analyzes the edge cases where the embeddings may fail. For example, definition encoders may not yield similar embeddings for child-parent terms located near the root node, because these terms can have very broad definitions which make them appear

unrelated. Also, a high-level term can have many child nodes and so appear in many different contexts. As a result, entity encoder like Onto2vec may produce a nonspecific embedding for this term which will not be close the embeddings of its child terms. For all the types of GO embeddings, we find that the similarity scores for child-parent terms are strongly affected by these edge cases. When one of the child-parent terms has low Information Content (IC) value, then the similarity score is not much higher than the score for two randomly chosen terms.

Task 2 follows the experiment in our previous work [27]. The objective is to evaluate how well the GO embeddings measure the similarity scores for sets of GO terms. Because genes and proteins are annotated by their own sets of GO terms, we reason that good embeddings should provide high similarity scores for two orthologous genes and for two interacting proteins [70]. Task 2 is a more realistic evaluation for different embedding types, because in practice, manually reviewed genes and proteins are rarely annotated with uninformative terms. Hence, an encoder can perform well in Task 2 even when it may fail in the edge cases of Task 1. In Task 2, definition encoders are better than entity encoders. Interestingly, the performances for various choices of definition encoders have mostly the same results.

In Task 3, we redesign the deep learning classifier DeepGO to take different kinds of GO embeddings as its inputs, and evaluate which embedding type would have the most impact [60]. DeepGO is a supervised learning model. Importantly, its datasets contains only frequent BP, MF and CC terms annotating at least 250, 50, and 50 proteins. Here, none of the GO embeddings significantly improves the classification accuracy. We suspect that the labels in DeepGO datasets are not sparse and can be well predicted without the extra information from the GO embeddings.

Next, to better quantify the impact for each embedding type, we consider the zeroshot learning case which is more difficult than the DeepGO setting [87]. We make our own semi-supervised classifier in which the GO embeddings are critical factors to predict labels unseen in the train data. Here, definition encoders built from the BERT approach obtain the best classification accuracy. Our software is available at `https://github.com/datduong/EncodeGeneOntology`.

4.2 Information Content

We take small detour to describe the Aggregate Information Content (AIC) Method, as it will be used as one of the baselines in our experiments [88]. The Information Content of a GO term t is defined to be $\text{IC}(t) = -\log(p(t))$ where $p(t)$ is the probability of observing a term t in the ontology. $p(t)$ is computed as $p(t) = \frac{\text{freq}(t)}{\text{freq}(root)}$. The term $\text{freq}(t)$ computes the frequency of a term t, where $\text{freq}(t) = \text{count}(t) + \sum_{c \in \text{child}(t)} \text{freq}(c)$. $\text{count}(t)$ is the number of genes annotated with the term t and $\text{child}(t)$ are the children of t. Based on this definition, $\text{IC}(root) = 0$, and a node near the leaves has higher IC than nodes at upper levels.

AIC defines a knowledge function for the GO term t as $k(t) = 1/\text{IC}(t)$ which is used to measure its semantic value $sw(t) = 1/(1 + \exp(-k(t)))$. Here $sw(root) = 1$. The semantic value $sv(t)$ of t is then defined as $sv(t) = \sum_{p \in \text{path}(t)} sw(p)$. Function $\text{path}(t)$ contains all the ancestors of t and the term t itself. Usually, $sv(a) < sv(b)$ when term a is nearer to the root than b. Song *et al.* [88] define their similarity score of two GO terms a, b as

$$\text{AIC}(a,b) = \frac{2\sum_{p \in \{\text{path}(a) \cap \text{path}(b)\}} sw(p)}{sv(a) + sv(b)}\,. \tag{4.1}$$

$\text{AIC}(a,b)$ ranges from 0 to 1. In this model, $\text{AIC}(a,a) = 1$. When a and b only have the root node as the common ancestor, then $\text{AIC}(a,b) = 2/(sv(a) + sv(b))$ which depends on where a, b are on the GO hierarchy.

4.3 Training datasets and objective function

We use cosine similarity score to compare the embeddings of two GO terms. Embeddings for a GO term and its parent terms are expected to be comparable, and should have higher similarity scores than the embeddings for unrelated terms. Suppose a GO encoder produces the vectors v_u and v_t for the GO term u and t. We train the parameters of this encoder to $\max \cos(v_u, v_t)$ when u and t are child-parent terms, and to $\min \cos(v_u, v_t)$ when u and t are randomly chosen, where the cos is the cosine similarity score.

To create the training dataset, we treat the BP, MF and CC terms as one giant connected network by treating the following one-directional relationships "is a", "part of", "regulates", "negatively regulates", and "positively regulates" as the same edge type. We define a positive sample as two child-parent terms, and a negative sample as two randomly chosen terms. We select a positive sample by randomly choosing a GO term and one of its parents. We compute the AIC scores for these positive samples and keep only samples having scores above the median [88]. This step ensures each remaining sample contains truly related terms. We explain AIC method in the Appendix.

We create two types of negative samples. First, we randomly select half the terms seen in the positive samples, and couple each term c in this set with an unrelated term d also seen in the positive samples. Second, we put the same term d with a random term e not found in the positive samples. This strategy helps the training process by allowing a GO encoder to observe the same terms under different scenarios [27]. We then compute the AIC scores for the negative samples, and keep samples having scores below the median. All encoders in our paper are trained on this dataset unless specified otherwise. Our dataset is available at `https://github.com/datduong/EncodeGeneOntology`.

4.4 Definition encoders

Every GO term has a definition defining the biological event that it represents; for example, the term GO:0008218 has the name *bioluminescence*, and the definition "production of light by certain enzyme-catalyzed reactions in cells". We concatenate the name and the definition of the term into one single description. To simplify, we will refer to this concatenation as the definition of a GO term. Naturally, to represent a GO term as a vector, we can transform its definition into a vector. Below, we explain ways to do so based on the BiLSTM method in Duong *et al.* [27], and the two recent popular approaches in Natural Language Processing, Embeddings from Language Models (ELMo) and Bidirectional Encoder Representations from Transformers (BERT) [25, 78].

4.4.1 Bidirectional Long-short Term Memory

We describe our previous BiLSTM encoder [27]. The BiLSTM network provides contextualized vectors for words in a sentence, so that the same word has different vectors depending on its positions in the sentence. Our input to BiLSTM is the word vectors produced by Word2vec. Word2vec assigns similar vectors to words with related meanings or that are likely to co-occur [71]. We use the Word2vec output in Duong *et al.* [27] which was trained on open access Pubmed papers and has word dimension in $\mathbb{R}^{300}$.

Using Word2vec, we transform the definition of GO term into a matrix M where the column M_j is the vector for the j^{th} word. The same word is always assigned to the same vector. To capture the fact that the same word often has different meanings depending on its position in the sentence, we apply $\tilde{M} = \text{BiLSTM}(M)$. Consider the word vector M_j at position j in a sentence of length L. BiLSTM computes the forward and backward LSTM model to produce the output vectors $\overrightarrow{h_j} = \text{LSTM}(\overrightarrow{h}_{j-1}, M_j)$ and $\overleftarrow{h_j} = \text{LSTM}(\overleftarrow{h}_{j+1}, M_j)$ and then returns $\tilde{M}_j = [\overrightarrow{h_j}; \overleftarrow{h_j}]$ where $[\overrightarrow{h_j}; \overleftarrow{h_j}]$ indicates the concatenation of the two vectors into one single vector.

To produce a vector representing the definition of one GO term, we take the max-pooling across the columns of $\tilde{M}$, $\text{maxpool}(\tilde{M})$ [17]. We apply a final linear transformation to this *aggregated* vector, making it into $\mathbb{R}^{768}$ to match BERT output dimension. We train this BiLSTM model on our own dataset in Section 4.3. During training, we freeze the input M and update only the BiLSTM parameters. We emphasize that 768 is chosen because of BERT (to be explained later); in practice, our BiLSTM can be trained to have output of any dimension.

4.4.2 Embeddings from Language Models

Embeddings from Language Models (ELMo) improves the BiLSTM encoder in two key ways [78]. First, instead of representing a whole word as a vector, ELMo represents each character in the alphabet as a vector and uses convolution filters of varying sizes to transform the alphabet vectors into a word vector. Second, ELMo trains a 2-layer BiLSTM. The first BiLSTM input are the word vectors from the character layer, and the second BiLSTM input are the output of the first BiLSTM.

The final vector for one word is a weighted average of the word vector from the character layer, and the output of the first and second BiLSTM, where the weights are computed for a given specific task and thus are jointly trained with the other parameters. We download the ELMo pretrained on Pubmed[1], freeze the character convolution layer, and train only the two BiLSTM layers. Borrowing notation from the previous section, let $\tilde{M}_j^l$ be the BiLSTM output of layer l. Our final vector for a word in a GO definition is $a_j\tilde{M}_j^1 + (1-a_j)\tilde{M}_j^2$ where $a_j \in [0,1]$ is specific to position j and is jointly trained with the two BiLSTMs. To encode a sentence (e.g. GO definition), we take the mean-pooling of the ELMo output matrix. ELMo is trained on the dataset in Section 4.3.

4.4.3 Bidirectional Encoder Representations from Transformers

BERT is a training strategy that provides contextualized vectors for words in a sentence [25]. BERT uses the Transformer architecture [101] which models all pairwise interactions between words in a sentence. We briefly describe the key idea in the Transformer neural network model. Transformer has 12 layers, where each layer takes as input the output of the previous layer. Each layer has 12 independent units (referred to as *heads* in the original paper). We describe one Head h in one Layer i. Let j denote the j^{th} word in the input sentence. Consider the input "*perforation plate* is-a *cellular anatomical entity*". We use the names for these two GO terms in this example, but in the experiment we will use the complete definitions. This input is partitioned into the following tokens *[CLS] per ##fo ##ration plate [SEP] cellular an ##ato ##mic ##al entity [SEP]*, where the special token [CLS] denotes the start of the whole input and [SEP] denotes end of each sentence [25].

Each token has three types of embeddings: the token W, position P and type T. Token embeddings assign a token to a vector (analogous to Word2vec embeddings). Position embeddings assign a location index j to a vector; for example, we assign the position vector P_0 to the first token [CLS]. Type embeddings assign the vector T_1 and T_2 to tokens in the first and second definitions respectively; for example, we assign T_1 to all the tokens *[CLS] per ##fo ##ration plate [SEP]*. The input of the first layer denoted as w_{0j} is a sum of the token, position and type embeddings

[1] https://allennlp.org/elmo

corresponding to the j^{th} token. The output vector of the Head h in Layer i denoted as $o_{i,j}^h$ is computed as the following weighted average

$$o_{i,j}^h = \sum_{k \in \{1:L\}} a_{ik}^h V^h(w_{i-1,k}) \tag{4.2}$$

$$a_{ik}^h = \text{softmax}(e_{ik}^h) \tag{4.3}$$

$$e_{ik}^h = Q^h(w_{i-1,j})^\intercal K^h(w_{i-1,k}) \tag{4.4}$$

where L is the input length, and V^h, Q^h, K^h are transformation functions. Loosely speaking, Eq. 4.3 models the pairwise interaction between the j^{th} and k^{th} tokens in the sentence. To merge all the heads at Layer i, Transformer concatenates the output $o_{i,j}^h$ at the position j of all the heads, and then applies a linear transformation on this concatenated vector. Linear transformation output o_{ij} is then passed to the next Layer $i+1$.

We now explain the two phases in BERT training strategy to train the Transformer model. In Phase 1, BERT employs a self-supervised strategy with two key objectives. First, from the training corpus, BERT removes words from the sentences, and then predicts these removed words using the remaining words. To do this, BERT applies a classifier to the output of Layer 12 in Transformer, i.e. the vector $o_{12,j}$. Second, BERT predicts if two sentences in a document are sequential or unrelated. Here, BERT applies a classifier to only the Layer 12 output vector $o_{12,0}$ corresponding to the [CLS] token. Because Transformer models all pairwise interactions of the words, the [CLS] token can be used to represent the entire input string (e.g. definitions of two GO terms in an input string).

To train Phase 1, we create our own training corpus with respect to the context of the Gene Ontology. To create one *document*, we concatenate the definitions of all the GO terms in one single branch of the GO hierarchy, starting from the leaf node to the root. We consider only the is-a relation, and randomly select only one parent if the node has many parents. Phase 1 will train the Transformer parameters to learn the interactions of words within the same definition, and the relationships among the definitions of terms on the same path to the root node. We use the same hyperparameters as the original BERT, where there are 12 heads and 12 layers, and the output vector in each layer is $o_{ij} \in \mathbb{R}^{768}$.

Phase 1 is often enough to produce an embedding for a sentence; for example, bert-as-service [104] takes the mean of Layer 11 output to represent the input sentence. Xiao [104] recommends the Layer 11 because they believe that Layer 12 values may be strongly affected by the two self-supervised tasks. We apply the same strategy to get the embedding for the definition of a GO term, and refer this strategy as $\text{BERT}_{\text{SERVICE}}$. Specifically, to represent *perforation plate* as one single vector, we first retrieve Transformer Layer 11 output for the input *[CLS] per ##fo ##ration plate [SEP]*, and then compute the mean over the six tokens $\frac{1}{6}\sum_{j=1}^{6} o_{11,j}$. This example uses the short name of a term, but in the final model, we use the long definition. We emphasize that $\text{BERT}_{\text{SERVICE}}$ output is fixed at 768 because of the pretrained setting in the original paper of [25].

In Phase 2 of BERT, Transformer is trained with a task-specific objective; for example, in Name Entity Recognition, this objective is to predict if a word in a sentence refers to a person, a location, or an object. Our objective is to produce a vector representing a GO term definition. We explore two different options for Phase 2. In the first choice, we average the Layer 12 output from the Transformer in Phase 1 to represent the definition of the GO term c, and then do the same for another term p. Denote v_c and v_p as the vector representing the label definitions of c and p respectively. We now train the Transformer on the dataset in Section 4.3. The objective is to maximize the cosine similarity score, $\max \cos(Av_c, Av_p)$ if c and p are child-parent terms, and to minimize $\min \cos(Av_c, Av_p)$ if c and p are randomly chosen, where A is a linear transformation. The same Transformer in Phase 1 is used in Phase 2, and we do not train a brand new Transformer. We just replace the objective function, and continue training the parameters with respect to this new objective function. We refer to this first choice as $\text{BERT}_{\text{LAYER12}}$.

Our second choice is exactly the same as the first one, except for a single key step. For a term c, we use the Layer 12 output vector $o_{12,0}$ of the [CLS] token to represent its definition, and then do the same for another term p. Because the vector $o_{12,0}$ is a function of all the words in the sentence, the [CLS] token can represent the entire input string [25]. Abusing the notation, let us define $o_{12,0}^{c}$ and $o_{12,0}^{p}$ as the vectors representing $o_{12,0}$ for the term c and p, respectively. We train the Transformer model to maximize $\max \cos(Ao_{12,0}^{c}, Ao_{12,0}^{p})$ if c and p are child-parent terms, and to minimize $\min \cos(Av_c, Av_p)$ if c and p are randomly chosen, where A is a linear transformation. We refer

to this second strategy as BERT_{CLS}. We set the final vector output for $\text{BERT}_{\text{LAYER12}}$ and BERT_{CLS} to be in $\mathbb{R}^{768}$, same as $\text{BERT}_{\text{SERVICE}}$.

4.5 Entity encoders

Because GO terms are arranged as a hierarchical structure, we can treat a term as a single entity and encode it into a vector without using its definition. We adapt the Graph Convolution Network (GCN) for the GO terms and evaluate Onto2vec. There are other node embedding methods, but GCN has shown to work well in various applications [81, 105].

4.5.1 Graph Convolution Network

Graph Convolution Network encodes each GO term in the tree into a vector [55]. Let A be the adjacency matrix, where $A_{ij} = 1$ if term i is the parent of term j. Compute $\tilde{A} = A + I$, where I is identity matrix. Compute the degree matrix $\tilde{D}$, where $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. Next scale A into $\hat{A} = \tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}$. Let W_1 and W_2 be two transformation matrices. Define X to be the initial vector embedding for the GO terms, where a column in X corresponds to a GO vector. Before training, X is initialized with random numbers.

During training X is transformed into a new matrix $E = W_2\hat{A}\,\text{relu}(W_1\hat{A}X)$. Loosely speaking, one column i in $\hat{A}X$ is a summation of all its child nodes and itself, so that the information flows from child to parent under this framework [55]. $W_1\hat{A}X$ then transforms this summation into a new vector. We repeat this transformation twice as in Kipf and Welling [55]. At the end, the column i in E is the vector for the i^{th} term. We set GCN to produce the final vector representation of size 768, same as $\text{BERT}_{\text{SERVICE}}$. We train GCN using the objective function and the data in Section 4.3.

4.5.2 Onto2vec

Onto2vec encodes GO terms into vectors by transforming their relationships on the GO hierarchy into sentences, which are referred to as axioms in the original paper [86]. For example, the

child-parent GO terms GO:0060611 and GO:0060612 are rewritten into the following sentence "GO:0060611 is_subclass GO:0060612". Onto2vec then applies Word2vec [71] on these sentences, so that GO names occurring in the same sentence are encoded into similar vectors. Because the training *sentences* are constructed from the GO trees without GO definitions, Onto2vec can conceptually be viewed as method that encodes nodes on graph into vectors like GCN. Because the Word2vec objective function is based on cosine similarity, for Onto2vec, GO terms in close proximity will have high cosine similarity score. We set Onto2vec to produce the final vector in $\mathbb{R}^{768}$, same as $\text{BERT}_{\text{SERVICE}}$.

4.5.3 BERT as entity encoder

Following Onto2vec, we apply BERT as an entity encoder (denoted as $\text{BERT}_{\text{NAME}}$) where the key objective is to encode the GO names into vectors. We create training data as follows. For each GO term, we select one path from that term to the root node via only is_a relation. For each path, we split the set of GO terms into half so that they represent the *first* and *second* sentence. $\text{BERT}_{\text{SERVICE}}$ and $\text{BERT}_{\text{NAME}}$ use a similar idea. In $\text{BERT}_{\text{SERVICE}}$, the training step requires GO definitions, whereas this phase in $\text{BERT}_{\text{NAME}}$ uses only the GO names. For example, consider the path GO:0000023, GO:0005984 GO:0044262, GO:0044237, and GO:0008152. In $\text{BERT}_{\text{NAME}}$, we format it into the input *[CLS] GO:0000023 GO:0005984 [SEP] GO:0044262 GO:0044237 GO:0008152 [SEP].*

Next, we set the vocabularies to be learned as the GO names; that is, the token embeddings (i.e. the values to be passed into Layer 1 of Transformer) return a vector for each GO name. Then, we train the two self-supervised objectives in Phase 1 of BERT on this data, so that we can capture the relatedness among the GO names. We use the same hyperparameters as the original BERT, where the final token embedding size is 768. After the model is trained, we treat the token embeddings as the GO embeddings. We do not take the last layer output because we do not want the contextualized vectors of the GO names which will vary depending on their locations in the input sequence and the surrounding words.

4.6 Task 1: Similarity score for two GO terms

Our GO encoders are trained so that the output vectors representing related terms will have high cosine similarity scores (e.g. high cosine similarity scores for child-parent pairs). We now describe a few edge cases where it may be challenging for GO embeddings to satisfy this property.

Two child-parent terms may describe very broad biological concepts, and their definitions can appear to be very unrelated. For example, consider the term *bioluminescence* and its parent *cellular metabolic process* whose definitions are "production of light by certain enzyme-catalyzed reactions in cells" and "chemical reactions and pathways by which individual cells transform chemical substances", respectively. One definition mentions the production of light; whereas the other does not. It is possible that the vectors representing these two terms may not have a high similarity score.

Next, consider a GO term that has many child nodes, which is often the case for terms located near root node. As an example, consider the term *cellular process* which has 59 child terms. In Onto2vec, *cellular process* will appear in many axioms and act like a hub, forcing all its child nodes to have very similar embeddings even when these nodes have different meanings. Subsequently, it is possible that the descendants of these dissimilar terms can also have similar embeddings. Onto2vec embeddings are then likely to return high similarity scores for unrelated terms. In GCN, our implementation builds the vector for a GO term by collecting the information from its descendant nodes. When two siblings have very different descendant nodes of their own, then these two siblings can have dissimilar embeddings. This strategy implies that the vector representing the parent of these two siblings cannot be very close to either of them. GCN then has the same problem observed in definition encoders.

To validate our intuition, we compute the cosine similarity scores between pairs of GO terms using different types of GO embeddings. We observe how the IC values of GO terms affect these similarity scores. The definition for IC value of a term g is $IC(g) = -\log p(g)$ where $p(g)$ is the probability that term g is used annotate a protein [77]. Terms having generic definitions or many child nodes tend to have low IC values because a curator is unlikely to use them to annotate proteins.

From the Human GO database, we sample 3000 child-parent pairs and 3000 pairs chosen at random. We stratify samples by the minimum IC value of the terms in a pair, and then compute the cosine similarity scores for the pairs using the embeddings of the terms. In this experiment, we compare the IC-based model AIC with the neural network encoders [88].

Figure 4.1: A neural network encoder's ability to accurately classify child-parent terms is correlated to the IC values of these terms.

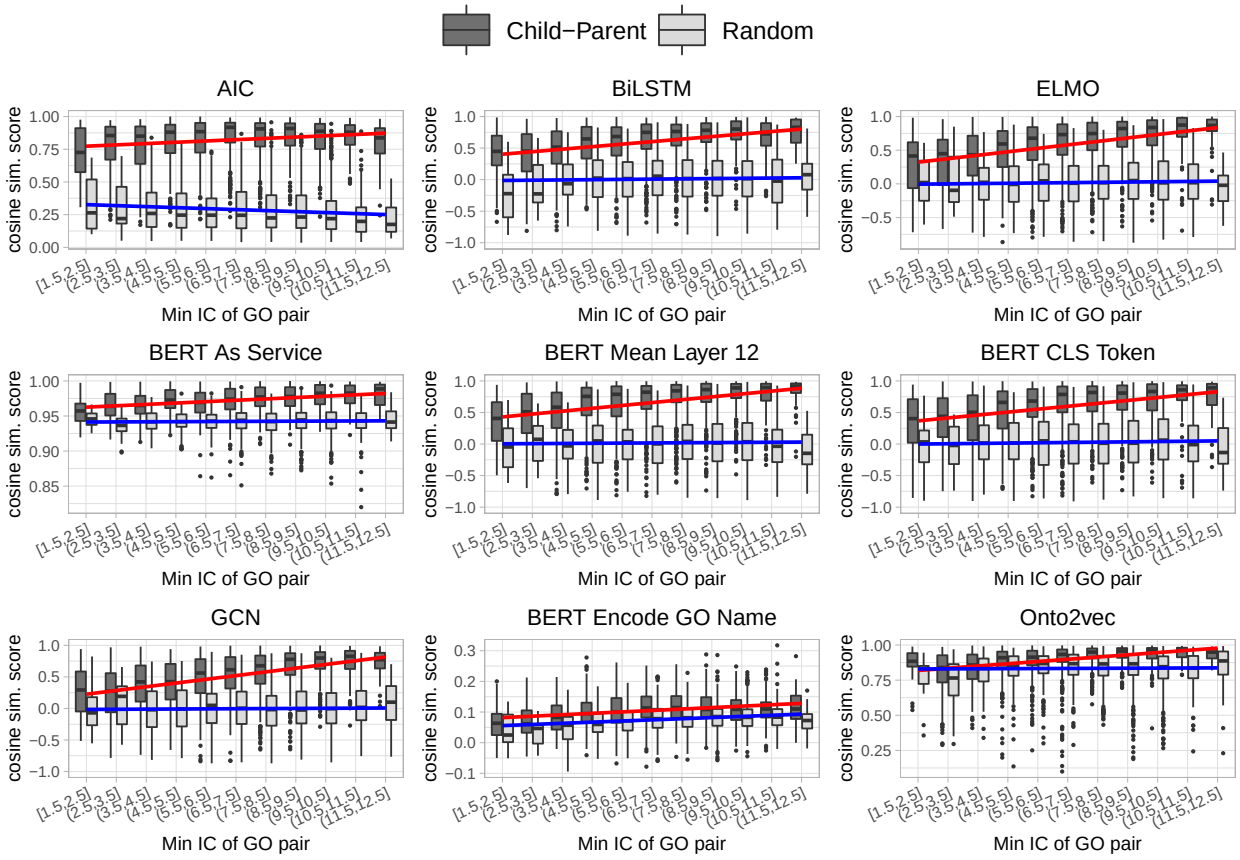


Figure 4.1 shows that AIC is better than the neural network models. At each IC value interval, AIC almost always returns higher similarity scores for child-parent terms than for unrelated terms. Definition encoders and GCN achieve this result only when both child-parent terms have high IC values (Fig. 4.1). This outcome agrees with our intuition about when the definition encoders and GCN may not properly measure the similarity scores for parent-child terms. $\text{BERT}_{\text{NAME}}$ and Onto2vec are noticeably worse than the other methods, where the score distributions for related and unrelated pairs intersect at every IC value interval. Task 1 indicates that to generate better GO

embeddings, we must integrate components of IC models into the neural network frameworks.

4.7 Task 2: Compare gene and protein functions

In Task 2, we compare two genes or two proteins by computing the similarity scores for the sets of GO terms annotating these genes and proteins. This is a canonical task applied in several earlier works [68, 70]. We briefly describe the procedure to compare protein functions which is explained in more detail in Duong *et al.* [27]. We evaluate different embedding types on the orthologous gene dataset for Human–Mouse, Human–Fly, and Mouse–Fly [27], and the Human [69] and Yeast Protein-Protein Interaction (PPI) network data [76]. Because the same procedure applies to orthologous gene and PPI network datasets, we will discuss the orthologous gene dataset in more detail. The orthologous gene datasets for Human–Mouse, Human–Fly, and Mouse–Fly have 10235, 4880, and 5091 samples respectively. Each dataset contains a set of positive samples which are pairs of orthologs, and an equally-sized set of negative samples which are pairs of randomly chosen genes. Because two orthologous genes have conserved functions, their sets of GO terms are often comparable and should have a higher similarity score than two GO sets annotating two randomly chosen genes.

We apply the Best-Match Average (BMA) metric [77] to compare two sets of GO terms. This metric requires a pairwise comparison for each term in the two annotation sets. Let A and B be two sets of GO terms, and let t_1 be a term in set A, and t_2 be a term in set B, then the score is

$$\text{BMA}(A,B) = \frac{1}{2}(\text{AVG}_{t_1}(\max_{t_2} \text{sim}(t_1,t_2)) + \text{AVG}_{t_2}(\max_{t_1} \text{sim}(t_1,t_2))) \tag{4.5}$$

We use cosine similarity for the function $\text{sim}(t_1,t_2)$. Embeddings that best estimate the similarity score for two terms will yield the most accurate BMA score. We compute the BMA score for pairs in the positive and negative samples, and then compute the Area Under the Curve (AUC) using the BMA scores as the predicted values and the true relationship between genes as the gold labels. Proper GO encoders will produce high AUC scores for each of the datasets. We repeat the same process for Human and Yeast PPI datasets, which contain 6031 and 3938 pairs respectively.

Table 4.1 shows the AUC for each neural network encoder. We observe that the AUCs of all the methods decrease in Human–Fly and Mouse–Fly ortholog datasets as compared to Human–Mouse dataset. We suspect that the accuracy drops because Human–Fly and Mouse–Fly datasets contain less well-annotated genes. The accuracy drops more for neural network encoders than for AIC method; this result agrees with Task 1 where AIC is better at comparing GO terms with low ICs. There is no single encoder that works best for all the datasets, but the definition encoders are consistently better than the entity encoders. Among the definition encoders, BERT variants obtain the best outcome. In term of model complexity and implementation, BiLSTM is the simplest definition encoder and can perform close to the BERT variants. We recommend BiLSTM for comparing sets of GO annotations.

Table 4.1: AUC for classifying true orthologous genes in Human, Mouse and Fly, and interacting proteins in Human and Yeast.

	Orthologous gene datasets			PPI data	
	Human-Mouse	Human-Fly	Mouse-Fly	Human	Yeast
Info Content					
AIC	95.79	**93.91**	**89.84**	87.89	87.77
Definition encoders					
BiLSTM	95.19	91.49	80.28	86.61	88.24
ELMo	95.47	91.10	79.78	87.73	89.03
$\text{BERT}_{\text{SERVICE}}$	**96.72**	92.94	79.62	88.15	89.00
$\text{BERT}_{\text{LAYER12}}$	95.94	92.50	81.65	**88.33**	**89.96**
BERT_{CLS}	96.05	90.80	78.99	86.94	89.27
Entity encoders					
GCN	94.99	85.54	72.99	85.45	86.75
Onto2vec	91.80	82.83	74.41	79.72	83.98
$\text{BERT}_{\text{NAME}}$	96.27	85.40	70.49	83.93	82.67

4.8 Task 3: Predict GO annotations for protein sequences

Predicting GO terms for novel protein sequences is an important research problem. As of January 2020, there are 44,700 terms in the GO database; many terms are sparse, annotating only a few proteins, and cannot be readily predicted by machine learning models. For this reason, several existing statistical and deep learning annotation models are built and tested on a smaller set of GO

terms that are not sparse; for example, Liu *et al.* [65] trained hierarchical classifiers on 281 and 790 labels for their yeast and human datasets, and Kulmanov *et al.* [60] trained a deep learning model on 1957 labels.

Work in other research fields have showed that the accuracy of an annotation method can improve when metadata about the labels are used as extra inputs [81, 105]. Because our GO embeddings capture the metadata about the terms such as their definitions and positions in the GO hierarchy, we want to observe how these embeddings affect the accuracy of an existing GO annotation model. We design two sub-tasks for Task 3. First, we modify the deep learning model DeepGO [60] to take as extra inputs the GO embeddings, and evaluate our model on the same datasets in the original paper. Second, we consider a more difficult task and build a model that uses the GO embeddings to predict labels unseen in the train datasets.

4.8.1 GO embeddings in supervised learning

In this sub-task, we integrate the GO embeddings into the supervised classifier DeepGO [60]. We use the DeepGO version which analyzes only the amino acid sequences, because we aim to observe the effect of the GO embeddings in the absence of any other factors such as PPI network data. This DeepGO version (denoted as DeepGoSeq in our paper) converts an amino acid sequence, e.g. $p =$ MARS ..., into a list of overlapping 3-mers MAR ARS .... Each 3-mer is assigned a vector in $\mathbb{R}^{128}$, so that if p has length 1002 amino acids, then the matrix representing p is $E_p \in \mathbb{R}^{128\times 1000}$. A 1D-convolution layer, 1D-maxpooling, and Flatten layer are then applied to get a vector v_p representing p, where $v_p = \text{flatten}(\text{maxpool}\ (\text{conv1d}(E_p))$. To predict if the term i is assigned to p, DeepGO fits a logistic regression layer $\text{sigmoid}(B_i^\intercal v_p + b_i)$, where B_i and b_i are parameters unique to label i. The objective loss function is binary cross entropy.

To add a type of GO embeddings into DeepGO, we make one minor change to the original model. Let g_i be the vector representing the term i, which can be retrieved by using any of our GO encoders. We concatenate $c_{pi} = [v_p\ g_i]$, and apply a linear transformation $\tilde{c}_{pi} = \text{relu}(\ Wc_{pi})$ so that $\tilde{c}_{pi}$ is the interaction of the sequence and the GO vector. To predict if term i is assigned to p, we fit $\text{sigmoid}(B_i^\intercal[v_p\ g_i\ \tilde{c}_{pi}] + b_i)$ where $[v_p\ g_i\ \tilde{c}_{pi}]$ is the concatenation of the three vectors. In

words, $[v_p\ g_i\ \tilde{c}_{pi}]$ represents the information from the amino acid sequence alone, the GO metadata alone, and their interaction effects. To train the model, we keep the GO vector g_i constant and update only the other parameters.

We create two more baselines to critically evaluate the effect of the GO embeddings on the classification accuracy. The first baseline (denoted as +ExtraLayer) is the same as DeepGO but has one extra linear layer where the logistic regression layer now becomes $\text{sigmoid}(B_i^\intercal\ \text{relu}(Wv_p) + b_i)$. This baseline allows us to estimate the effect of adding more layers to the original model without using any GO embeddings. The second baseline (denoted as RandomEmb) takes random embeddings where each entry is sampled from a uniform distribution $U(-1, 1)$. RandomEmb acts as the control case for the actual GO embeddings, and are integrated into DeepGO in the same way that real GO embeddings are.

We train and evaluate all methods on the same DeepGO datasets. The original DeepGO paper has three datasets, one for each of the BP, MF, and CC ontology. The BP, MF, and CC training data sample sizes are 29100, 20159, and 28437 proteins, and their test data sizes are 9095, 6294, and 8886, respectively. Ancestors of the GO terms annotating a protein are added into the ground-truth label set, and then BP, MF and CC terms occurring below 250, 50, and 50 times are removed from the datasets. In total, the number of BP, MF and CC terms to be predicted are 932, 589, and 436, respectively. These BP, MF and CC labels are not sparse; the median occurrence frequencies are 365, 88, and 111 times, respectively. In this experiment, we do not want very common labels to affect the accuracy metric; for example, the term *cell part* occurs 25850 times and has a high AUC value 83.57 in the original DeepGO paper. We compute Micro AUC and Macro AUC, but we will focus our discussion on the Macro AUC which is the unweighted average of the per-label AUC, so that the AUC values of infrequent labels have more contributions.

In a supervised learning, we can usually obtain accurate predictions for labels occurring frequently enough in the datasets. We suspect that DeepGO may not benefit from having GO embeddings as additional inputs. Table 4.2 shows that this is indeed the case. Our baseline +ExtraLayer performs very well compared to DeepGO integrated with GO embeddings. Micro and Macro AUC are similar for all the embedding types, where $\text{BERT}_{\text{NAME}}$ and Onto2vec produce

competitive accuracy even when they are subpar to the other encoders in Task 1 and 2. We suspect that the parameters in DeepGO can be trained to compensate for imperfect GO embeddings. This idea is supported by the result of RandomEmb which is about the same as all the other methods. We observe that by adding only one more layer to DeepGoSeq, we can significantly increase the Macro AUC in the MF and CC data (Table 4.2 row 2). A more complex neural network model can probably extract more useful information form the amino acid sequences. In Chapter 5, we will build a more complex deep learning model.

Table 4.2: Models are trained and tested on the same original DeepGO datasets. DeepGoSeq indicates most basic DeepGO version that analyzes only amino acid sequences of proteins. GO embeddings produced by different types of GO encoders are then integrated into the DeepGoSeq. We do not observe significant differences among the encoders, except for GCN in the BP ontology. Italicized numbers are the best baseline values, and bold numbers are the best values for the GO encoders.

	BP AUC		MF AUC		CC AUC	
	Macro	Micro	Macro	Micro	Macro	Micro
Baselines						
DeepGoSeq	62.60	82.17	73.90	87.49	67.12	93.07
+ExtraLayer	62.82	82.51	77.39	88.89	71.03	*93.32*
RandomEmb	*65.49*	*82.76*	*78.17*	*88.94*	*71.55*	93.10
Defintion encoders						
BiLSTM	64.37	83.16	76.53	88.83	70.95	**93.56**
ELMo	64.00	82.97	76.53	88.25	71.26	93.40
$BERT_{SERVICE}$	**64.95**	**83.51**	**77.74**	**89.21**	**71.85**	93.54
$BERT_{LAYER12}$	64.50	83.30	76.59	88.63	67.36	93.01
$BERT_{CLS}$	64.49	83.38	77.08	88.75	70.45	93.35
Entity encoders						
GCN	60.26	78.20	75.09	86.19	69.53	92.06
Onto2vec	64.85	83.41	76.52	88.90	69.82	93.31
$BERT_{NAME}$	63.50	82.94	76.03	88.65	69.36	93.26

4.8.2 GO embeddings in zeroshot learning

In the supervised learning setting where the labels occur many times, we observe that adding GO embeddings does not improve prediction accuracy. In this section, we consider a more difficult problem where the GO embeddings will hold critical roles. We build a model that uses the

embeddings to predict the unseen labels, a scenario which has not been tried in previous deep learning approaches [60, 65]. It is worthwhile to predict these rare labels because they are the most related to the true protein functions. For example, *perforation plate* is precise to a protein location but not its parent *cellular anatomical entity* or ancestor *cellular component*. Moreover, the GO database is frequently updated with terms being added and deleted. If we can readily predict new labels, then we will not need to train the existing classifier for each update.

Our model in this experiment relies on the zeroshot learning idea [87]. Suppose we can train a classifier $C(v_p, v_s)$ that takes two vectors v_p and v_s representing the input p and the label s seen in the training data. Then for an unseen label u which can be represented as a vector v_u, we can apply $C(v_p, v_u)$ to classify if p has the unseen label u. Building a good zeroshot learning model is nontrivial. In this paper, our goal is not building a state-of-the-art zeroshot model; rather, we want to know which types of GO embeddings will be most suitable for future work toward this direction.

We introduce our model DeepGOZero which is based on the original DeepGO. We use the same $v_p = \text{flatten}(\text{maxpool}\ (\text{conv1d}(E_p))$ to encode the amino acids. Next, we convert v_p into the same dimension as the GO vector g_i by using $\tilde{v}_p = W_2 \text{relu} W_1 v_p$. Then we create the vector c_{pi} for the final classification by concatenating four vectors $c_{pi} = [\tilde{v}_p, g_i, \tilde{v}_p \otimes g_i, |\tilde{v}_p - g_i|]$. The predicted probability for label i is computed from three layers of fully-connected feed forward network with Relu activation. We will refer to these three layers as one layer L, and write the prediction as $\text{sigmoid}(L(c_{pi})) = \text{sigmoid}(L(v_p, g_i))$.

The parameters in L are shared for all the labels, so that terms with similar embeddings (e.g. child-parent terms) will be forced to have comparable predictions. We fix the GO embeddings as constants and train only the other model parameters. Importantly, after the model is trained, we can still predict if a protein p is assigned a GO term u which is not seen in the training label set. First, we get the vector g_u for this term by using any of our GO encoders. Then, we obtain v_p for the protein p, and apply the classifier layer L to v_p and g_u. Unlike L, the parameters B_i and b_i in the logistic layer of DeepGO are unique for each label and cannot be applied to labels not included in the train datasets.

We describe our new datasets used to evaluate DeepGOZero. We extend the label sets in the

original DeepGO data by including BP, MF, and CC terms annotating at least 50, 10, and 10 proteins, respectively (the same criteria in the original paper are 250, 50 and 50 respectively). Our BP, MF and CC datasets now have 2980, 1697 and 989 labels instead of 932, 589 and 439 labels, respectively. We keep the same number of proteins as the original datasets, so that the added labels are sparse. 95% of the added labels occur below 252, 34, and 68 times in the BP, MF and CC training data, respectively. We do not lower the selection criteria further because we want to reliably train our baseline +ExtraLayer on this entire larger datasets. Having too many sparse labels can decrease the performance of +ExtraLayer. The baseline +ExtraLayer represents the upper bound for our zeroshot learning experiment, because models trained on the complete data should be better than models trained only on part of the label sets.

Table 4.3: We add 2048 BP, 1108 MF and 550 CC labels to the original DeepGO datasets, and keep the same number of proteins. Our improved DeepGO baseline +ExtraLayer is trained and tested on the entire larger dataset, and acts as an upper bound for the other models. Different types of GO embeddings are then integrated into DeepGOZero. Models are trained on the original DeepGO datasets, but tested on the added 2048 BP, 1108 MF and 550 CC labels in our own larger datasets. These added labels are unseen by models during training. Italicized numbers are the upper bound, and bold numbers are the best values for the GO encoders.

	BP AUC	MF AUC	CC AUC
	Macro	Macro	Macro
Baselines			
+ExtraLayer	*64.64*	*72.45*	*69.31*
RandomEmb	54.39	49.72	52.05
Defintion encoders			
BiLSTM	54.91	61.35	61.45
ELMo	56.00	57.60	58.74
$\text{BERT}_{\text{SERVICE}}$	**62.48**	**69.96**	62.55
$\text{BERT}_{\text{LAYER12}}$	59.91	66.39	**65.53**
BERT_{CLS}	55.97	57.78	58.13
Entity encoders			
GCN	54.05	50.25	53.81
Onto2vec	56.30	56.82	52.38
$\text{BERT}_{\text{NAME}}$	56.06	55.46	58.00

DeepGOZero is trained on the original DeepGO data but tested on the added 2048 BP, 1108 MF and 550 CC labels in our larger datasets (Table 4.3). As a control case, DeepGOZero is also implemented with RandomEmb to estimate the accuracy due to chance. We pay attention

specifically to Macro AUC because it gives equal weights to rare and common labels; whereas, Micro AUC focuses more on the common labels which are often easier to classify.

In Table 4.3, $\text{BERT}_{\text{SERVICE}}$ and $\text{BERT}_{\text{LAYER12}}$ have Macro AUC closest to +ExtraLayer and furthest from RandomEmb. $\text{BERT}_{\text{SERVICE}}$ is better than $\text{BERT}_{\text{LAYER12}}$ for BP and MF labels; whereas, $\text{BERT}_{\text{LAYER12}}$ is better than $\text{BERT}_{\text{SERVICE}}$ for CC labels. DeepGOZero may not be the most effective zeroshot model to annotate protein functions; yet, $\text{BERT}_{\text{SERVICE}}$ and $\text{BERT}_{\text{LAYER12}}$ still come close to the upper bound +ExtraLayer. These results demonstrate that GO embeddings are meaningful for zeroshot learning models. We recommend the $\text{BERT}_{\text{SERVICE}}$ or $\text{BERT}_{\text{LAYER12}}$ embeddings for future zeroshot learning methods.

4.9 Summary and discussion

This chapter presents a comprehensive experiment comparing different ways to encode the metadata of the terms in the GO database. There are two types of encoders (1) definition encoders which encode the definitions of the term and (2) entity encoders which encode the hierarchal relationships of the terms in the database. In this experiment, the definition encoders are BiLSTM and ELMo, whereas the entity encoders are GCN and Onto2vec. BERT is a special case because we can design it to be either in defintion ($\text{BERT}_{\text{SERVICE}}$, $\text{BERT}_{\text{LAYER12}}$ and BERT_{CLS}) or entity ($\text{BERT}_{\text{NAME}}$) encoder.

We evaluate all the encoders in the following tasks. Task 1 studies edge cases where the GO encoders may not produce accurate GO embeddings. We find that all the neural network models often fail to properly produce the embeddings for child-parent terms when these terms have low IC values (Fig. 4.1). Usually, manually annotated proteins are unlikely to contain terms with low IC values. To observe a realistic application of the GO encoders, in Task 2, we measure the relationships of proteins by comparing the set of GO terms annotating them. In this case, definition encoders work better than entity encoders, and more accurately classify whether two proteins are interacting or two genes are orthologous. Surprisingly, our BiLSTM encoder in Duong *et al.* [27] obtains very close results compared to the more complex ELMo and BERT-based encoders.

Because BiLSTM is easier to implement than the other definition encoders, we recommend the BiLSTM encoder for comparing functions of genes or proteins.

Task 3 evaluates the scenarios where GO metadata may help the accuracy of annotating protein functions. We add GO embeddings as additional features for DeepGO [60]. DeepGO datasets have enough observations for each GO label, and we do not observe that the GO embeddings significantly help the classification accuracy. This result is rather expected because one key factor leading to high accuracy for a classification problem is to have a lot of labeled data.

Predicting sparse labels is important for a protein, because these labels are more precise to the protein functions. We design another classifier DeepGOZero where the GO embeddings are key factors for predicting rare labels which were excluded in the original DeepGO datasets. We train DeepGOZero on the 932 BP, 589 MF and 439 CC labels in the original DeepGO datasets, and test on the 2048 BP, 1108 MF and 550 CC sparse labels which were removed due to the exclusion criteria in the original DeepGO paper. $\text{BERT}_{\text{SERVICE}}$ and $\text{BERT}_{\text{LAYER12}}$ obtain the highest Macro AUCs and come close to the supervised DeepGO trained on the full label sets (original and sparse labels). This experiment shows that GO embeddings are meaningful for zeroshot learning models. Zeroshot learning has two key advantages: it can predict any labels in the GO database, and is not required to be trained each time the database is updated.

Our encoders in this chapter produce GO embeddings at dimension 768 to match the $\text{BERT}_{\text{SERVICE}}$ output. $\text{BERT}_{\text{LAYER12}}$ however can be pretrained to produce embedding of any dimension for other downstream tasks (depending on GPU constraint). In contrast, by construction the $\text{BERT}_{\text{SERVICE}}$ embedding is always at dimension 768, because each layer including the output must have the same dimension as the pretrained setting in Devlin *et al.* [25]. To reduce $\text{BERT}_{\text{SERVICE}}$ embedding size, we may add a transformation layer in the downstream task. This extra step may complicate the training procedure (e.g. needing more memory and longer runtime).

Due to $\text{BERT}_{\text{LAYER12}}$ flexibility and good performance in all the evaluation tasks, we recommend it for future works involving the application of GO metadata. In Chapter 5, we will use $\text{BERT}_{\text{LAYER12}}$ as inputs to our new GO label classifier.

Our neural network encoders are not built independently from the GO hierarchy and the IC-

methods. In $\text{BERT}_{\text{SERVICE}}$, $\text{BERT}_{\text{NAME}}$, GCN, and Onto2vec, the training corpus are generated from the terms that are on the same branch in the GO hierarchy. In BiLSTM, ELMo, $\text{BERT}_{\text{LAYER12}}$ and BERT_{CLS}, we apply the AIC method to keep training samples that are very similar or are very different. For our future work, we will study other techniques to combine neural network encoders with the GO hierarchy topology and the components of IC-models.

CHAPTER 5

Deep learning model to predict protein functions

5.1 Introduction

Predicting protein functions is an important task in computational biology. With the decline of sequencing cost, the gap between the numbers of labeled and unlabeled sequences continues to grow [107]. Protein functions are described by Gene Ontology (GO) terms [96]. Predicting protein functions is a multi-label classification problem where the input is an amino acid sequence and the output is a set of GO terms. GO terms are organized into a hierarchical structure, where generic terms (e.g. cellular anatomical entity) are parents of specific terms (e.g. perforation plate). Due to this tree structure, in the training and testing datasets, if a GO term is assigned to a protein, then all its ancestors are also assigned to this same protein.

When analyzing only the amino acid sequence data to predict protein functions, there are two major trends. The first trend relies on string-matching tools like Basic Local Alignment Search Tool (BLAST) to match the unknown sequence with labeled proteins in the database [3, 80, 107]. Zhang *et al.* [107] combined BLAST with Position-Specific Iterative Basic Local Alignment Search Tool (PSI-BLAST) to retrieve even more labeled proteins related to the unknown sequence [4]. The key idea behind BLAST-based methods is to retrieve proteins that resemble the unknown sequence. Most likely, the retrieved proteins will contain similar evolutionarily conserved regions, motifs (e.g. kinase domain), and general secondary structures (e.g. helix-turn-helix) that align well with the unknown input sequence. Then, all GO labels assigned to these retrieved proteins are assigned to the unknown sequence [107].

The second trend transforms the amino acid sequences into feature vectors, then applies clas-

sification methods to these vectors. For example, DeepGO converts an amino acid sequence into a string of k-mers, where each k-mer is represented by a vector so that the amino acid sequence is represented as a matrix m [60]. The next objective is to find a function $f(m, g)$ that returns the correct assignment for the label g. Deep learning models like DeepGO and related work on DNA sequences use the convolutional neural network (CNN) as the key component for this function f [59, 60, 110]. There have been attempts in applying long-short term memory (LSTM) but the results did not outperform the CNN architecture [59].

We address the remaining two questions in Section 1.2. The first question is: how does an attention-based neural network compared against CNN models? Here, we introduce a novel classifier GOAT, **GO a**nnotation based on the **T**ransformer framework [101]. In Natural Language Processing, the Transformer was designed to capture how much attention each word in an input sequence (e.g. sentences) gives to and receives from the other words in the same sequence. In the context of protein sequences, unlike CNN and LSTM, Transformer models all the pairwise interactions of the amino acids in the protein and may have a better chance of capturing meaningful long-range relationships among these amino acids. Besides the amino acid sequences, GOAT also takes GO label embeddings (Chapter 4) so that it leverages label metadata to make prediction.

The second question is: is there key information in a protein sequence that is best retrieved by BLAST, and how to integrate this knowledge into a deep learning method? We want to reconcile the gap between BLAST and deep learning methods. BLAST models retrieve sequences with key amino acid patterns similar to the unlabeled protein. BLAST may even recognize secondary structural motifs (e.g. helix-turn-helix) [92]. However, BLAST models do not apply machine learning techniques on these patterns to predict the GO labels. Deep learning models want to correlate these patterns to the labels via the function f, but are not always guaranteed to discover such amino acid patterns. For this reason, we will identify motifs in the protein sequences via tools like PROSITE [85], and then use this information as extra features for our classifier.

We will also evaluate how protein metadata such as high-level 3D structures and protein-protein interaction (PPI) network affect the prediction. We focus on neural network models, whereas previous work of Zhang *et al.* [107] focused on BLAST-based methods. Zhang *et al.* [107] built

ensemble model from individual classifier created from each type of metadata. We will use all the metadata as inputs into one single classifier.

In the following sections, we will compare GOAT against a recent BLAST-based approach and two CNN methods on the Uniprot data [59, 60, 107]. We will study key properties of GOAT (e.g. do the motifs pay attention to each other?) and the effect of three types of protein metadata: amino acid motifs, high-level 3D structures, and PPI network. Without metadata, GOAT is better than CNN methods for rare MF and CC labels, but not for BP labels. With metadata, GOAT has higher accuracy for rare labels than the CNN methods and the BLAST baseline. Ablation study shows that motifs and 3D folding information are meaningful factors, but PPI network is the most important. We hope that GOAT will serve as a meaningful neural network baseline for future research work. GOAT is available at `https://github.com/datduong/GOAnnotationTransformer`.

5.2 BLAST and PSI-BLAST

We describe our first baseline which is a very competitive BLAST-based method by Zhang *et al.* [107]. Zhang *et al.* [107] consider several best matched proteins from BLAST and additional sequences attained by PSI-BLAST when predicting annotations. The authors [107] refer to their BLAST and PSI-BLAST method as the sequence-based module of their software MetaGO. Here, we refer to it as $\text{MetaGO}_{\text{BLAST}}$, and briefly describe this method. Let N^{blast} and N^{psiblast} be the number of sequences in the train data retrieved by BLAST and PSI-BLAST that best match the unknown protein, and q be a GO label. Next define $N^{\text{blast}}(q)$ and $N^{\text{psiblast}}(q)$ as the number of sequences having the label q in N^{blast} and N^{psiblast}. Let $S_n^{\text{blast}}(q)$ and $S_n^{\text{psiblast}}(q)$ be the sequence-similarity scores of the n^{th} retrieved sequence in $N^{\text{blast}}(q)$ and $N^{\text{psiblast}}(q)$ which contains the GO label q. The assigned prediction probability for label q with respect to the unknown query sequence is

$$score(q) = w\frac{\sum_n^{N^{\text{blast}}(q)} S_n^{\text{blast}}(q)}{\sum_n^{N^{\text{blast}}} S_n^{\text{blast}}} + (1-w)\frac{\sum_n^{N^{\text{psiblast}}(q)} S_n^{\text{psiblast}}(q)}{\sum_n^{N^{\text{psiblast}}} S_n^{\text{psiblast}}} \tag{5.1}$$

where $w = \max_n S_n^{\text{psiblast}}$ so that BLAST has a stronger weight when very close homologs are found [107]. In other words, the prediction scores for the GO labels are scaled by how similar the

query protein is to the known sequences.

5.3 Convolutional neural network

We describe DeepGO by Kulmanov *et al.* [60] which is built from the convolution neural network (CNN) architecture. This baseline was described in Section 4.8.1. We will briefly discuss it here again. We consider the DeepGO version which does not correct for consistent predicted probabilities. Consistency is defined as the fact that if a GO label is assigned to a protein then all its ancestors must also be assigned to the same protein. This consistency is corrected for each GO label, where the final prediction is the maximum of its own predicted probability and the probabilities of its descendants. In other words, in DeepGO when a descendant of a specific label has high predicted probability then this label will also have high predicted probability. There are other types of correction in Obozinski *et al.* [73] which DeepGO did not compare. We believe this research direction on consistency correction requires its own analysis. Moreover, consistency correction relies on the per-term prediction accuracy. For this reason, we focus comparing GOAT to only the basic DeepGO.

DeepGO converts an amino acid sequence, for example $p = \text{MARS}\ldots$, into a list of overlapping 3-mers, e.g. MAR, ARS .... Each 3-mer is represented by a vector of in $\mathbb{R}^{128}$, so that p is represented by a matrix $E_p \in \mathbb{R}^{128\times(L-2)}$ where L is the sequence length. A 1D-convolution layer, 1D-maxpooling and Flatten are then applied to E_p, so that we have $v_p = \text{flatten}(\text{maxpool}\ (\text{conv1d}(E_p))$ as the vector representing this k-mer sequence. To predict a GO label i, DeepGO fits a logistic regression layer $\text{sigmoid}(B_i^\intercal v_p + b_i)$ with the binary cross entropy as the objective loss function.

To integrate metadata about the protein (e.g. PPI network), DeepGO concatenates this information into v_p before sending it through the logistic layer. For example, in the original paper, Kulmanov *et al.* [60] convert proteins from a PPI network into vectors. Let $c_p \in \mathbb{R}^d$ be the vector representing protein p in this PPI network; then the classification layer becomes $\text{sigmoid}(B_i^\intercal [v_p, c_p] + b_i)$ where $[v_p, c_p]$ is the concatenation of the two vectors. We emphasize that the vector c_p does not have to be from the PPI network; in the results, we evaluate the DeepGO model with vectors representing

3D structures of proteins (the outcome of this experiment is not shown in table but explained in result section). To train DeepGO, we use the same hyperparameters as the original paper [60].

DeepGOPlus developed by the same authors of DeepGO is an ensemble method of BLAST and DeepGOCNN [59]. We will focus on the neural network component DeepGOCNN of DeepGOPlus. DeepGOCNN follows the same idea of DeepGO. However, DeepGOCNN does not use 3-mers but encodes each unique amino acid as 1-hot vector. For example, the first amino acid (out of 21 amino acids) will have the vector $[1, 0, 0 \dots 0]$ of length 21. DeepGOCNN uses 512 CNN components of sizes $8, 16, 32 \dots 128$ (512×16 total CNN components). These CNN layers are not stacked to create a deep network, instead max-pooling is applied for each CNN module. The final feature vector v_p of the protein sequence (in $\mathbb{R}^{8192}$) is the concatenation of these max-pooled outputs. The authors of DeepGOCNN trained one single model for all three ontologies. We will train DeepGOCNN separately, one model for each of the three ontologies using the same hyperparameters as the original paper.

5.4 GOAT: GO annotation method with Transformer

We introduce our novel **GO a**nnotation method with **T**ransformer (GOAT). GOAT takes as an input a sequence of amino acids and GO labels. For illustration purposes, consider the protein MAP Kinase-activated Protein Kinase 5 (UniProtKB O54992), which we will refer to by its id O54992 for brevity. Let L denote the sequence length, and let $g_1 \dots g_G$ denote the names of the labels to be predicted, where G is the number of labels to be predicted. Our input will be the string $MSEDS \dots LPHEPQ\ g_1 \dots g_G$ of total length $L + G$. Next, define E as the embeddings for the amino acids, for example E_M and E_S are the vectors representing the amino acids M and S respectively.

Let E^G be the embeddings for the GO labels, so that $E^G_{g_1}$ and $E^G_{g_2}$ are the vectors representing the first and second GO label g_1 and g_2 respectively. E^G is analogous to Word2vec embedding; except in this case, instead of having a vector for each word in a corpus, we will have a vector for each GO label in the train and test datasets. In this chapter, we set the vectors represent the amino

acids and GO labels to be in the same dimension; that is, E_M and $E^G_{g_1}$ are vectors of the same size. To reduce the number of parameters, for the GO labels, we fix E^G as the $\text{BERT}_{\text{LAYER12}}$ GO embeddings in Chapter 4 instead of setting it as a trainable parameter.

We add a position vector P_j to the j^{th} amino acid in the sequence; for example, the first and second amino acid M and S will have the following two vectors, $E_M + P_1$ and $E_S + P_2$. We observe that the position embedding makes sense for amino acids so that the same amino acid appearing at different locations will be treated differently. However, position embedding does not apply to GO labels; that is, the ordering of the labels should not affect the prediction outcome. For this reason, we do not add position embedding to the GO labels.

Next, we introduce the region-type embeddings R to highlight the fact that some amino acids belong to known motifs. Again consider the protein O54992, which is 473 residues long and contains two key regions: a kinase motif at position 22-304 and a coiled coil domain at position 409-440. In this case the 25^{th} amino acid is T and is inside the kinase motif; for this reason, it will have the vector $E_T + P_{25} + R_{\text{kinase}}$. Likewise, the 410^{th} amino acid is N and will have the vector $E_N + P_{410} + R_{\text{coiled coil}}$. Amino acids outside any key regions will not have region embedding added to them. Motifs can be found by using PROSITE; fortunately, many labeled sequences in Uniprot already have this information [21, 85].

We now describe how the Transformer architecture in GOAT analyzes the input sequence. The original Transformer has 12 layers of encoders and each layer has 12 independent identical units (referred to as heads in the original paper) [101]. To keep our software manageable for all users, we use only one head and so we will exclude description of head. We will use 12 layers. The first layer takes as arguments the vectors representing the input string. Here we simplify the notation, let w_j be the vector representing the j^{th} element in the input string. For protein O54992 of length $L = 473$ and G number of GO labels, from the input string $MSEDS \ldots LPHEPQ\, g_1 \ldots g_G$, we

will have for example $w_{25} = E_T + P_{25} + R_{\text{kinase}}$, and $w_{474} = E_{g_1}$. At the first layer, we have

$$o_{1j} = \sum_{k \in \{1:(L+G)\}} a_{jk} V^1(w_k) \tag{5.2}$$

$$a_{jk} = \text{softmax}(e_{jk}) \tag{5.3}$$

$$e_{jk} = Q^1(w_j)^\intercal K^1(w_k) \tag{5.4}$$

V^1, Q^1, K^1 are transformation functions for layer 1. o_{1j} is a weighted sum of the transformed vectors representing itself and the other entities in the input string. o_{1j} is then transformed as $p_{1j} = L_2^1(\text{gelu}(L_1^1 o_{1j}))$ where L_1^1 and L_2^1 are two linear transformations with the gelu activation function in between. The final output of Layer 1 is $h_{1j} = \text{LayerNorm}(p_{1j} + o_{1j})$. Loosely speaking, the first layer computes all pairwise interactions of w_j and w_k for all k, where the attention a_{jk} in Eq. 5.3 indicates how much w_k contributes toward w_j.

The second layer takes the output of the first layer as its input, so that we have for any layer i

$$o_{ij} = \sum_{k \in \{1:L+G\}} a_{jk} V^i(h_{i-1,k}) \tag{5.5}$$

$$a_{jk} = \text{softmax}(e_{jk}) \tag{5.6}$$

$$e_{jk} = Q^i(h_{i-1,j})^\intercal K^i(h_{i-1,k}) \tag{5.7}$$

where V^i, Q^i, K^i are transformation functions for layer i. This layer i will have its own linear transformations L_1^i and L_2^i to transform o_{ij}. Again, loosely speaking, layer i computes all pairwise interaction for the output from the previous layer $i-1$.

At layer 12, we focus only on the output $h_{12,k}$ corresponding to the GO labels. Let us denote h_{12,g_i} as the final output for the term g_i. We use a single linear classifier $\text{softmax}(Ch_{12,g_i})$ to return the presence and absence probability of g_i for the input protein. The same transformation C is applied to all labels, so that a set S of GO terms having similar $h_{12,g_{i \in S}}$ will have similar predictions.

At each label g_i, the output h_{12,g_i} encapsulates all the information from the amino acids and from all the other labels, so that values which affect h_{12,g_i} will also affect the output h_{12,g_j} at

another label g_j. Intuitively, with this fact and the fact that all labels share the same classifier C, the prediction at g_i and g_j are to some degree correlated. We suspect that Transformer can model co-occurrences of labels. To validate this, from the T-SNE plot of the vectors h_{12,g_i} in the result section, we observe that a label and its ancestors will be nearby, even when their initial definition embeddings E_{g_i} in Duong *et al.* [28] are dissimilar, as is the case when a term and its distant ancestors can have dissimilar definitions.

When there are metadata for the proteins, such as their embedding c_p from a PPI network, we can concatenate such embeddings into h_{12,g_i} as in DeepGO. Next, we use a two-layer fully connected classifier, $\text{softmax}(\ C_1(\text{relu}(C_2[c_p, h_{12,g_i}])))$ to return the presence and absence probability of g_i, where C_1 and C_2 are shared for all the labels.

5.5 Uniprot data and evaluation metrics

GOAT takes several data sources as inputs, some of which (e.g. PPI network or new motifs) may not be known for brand new proteins. To fully understand its behaviors, we evaluate GOAT on a subset of well-annotated proteins in the Uniprot data. We use same the Uniprot data as in DeepGO and follow the same preprocessing steps [60]. DeepGO removed proteins with ambiguous amino acid codes (B, O, J, U, X, Z) and sequences longer than 1000 amino acids. The remaining dataset covers more than 90% of the Uniprot data. For each GO term annotating a protein, all the ancestors of that term are also added to the ground truth. DeepGO removed BP, MF and CC labels annotating less than 250, 50, and 50 proteins, respectively. The final BP, MF and CC datasets have 29100, 20159, and 28437 training proteins, and 9095, 6294, and 8886 testing proteins, respectively. In total, the numbers of BP, MF and CC terms in the label sets are 932, 589, and 439, respectively. For baselines, we will not include DeepGOCNN until Section 5.9 where we have larger datasets with more labels.

We present three main results in the following sections. First, we evaluate the base implementation of GOAT that analyzes just the raw amino acid sequence data alone. We use the name $\text{GOAT}_{\text{BASE}}$ to indicate this base implementation, and ignore the subscripts when the context is clear.

Second, we show that GOAT obtains higher classification accuracy when it takes as extra inputs the motif information from the protein sequences. We use the name $\text{GOAT}_{\text{MOTIF}}$ for this version of GOAT. We also observe how the Transformer architecture in GOAT analyzes the motif information when it predicts GO labels for an amino acid sequence. Third, because motif information is a key input of GOAT, we integrate 3D structure and PPI network data on top of $\text{GOAT}_{\text{MOTIF}}$ to obtain even better prediction outcome. We use the name $\text{GOAT}_{\text{MOTIF,3D}}$, $\text{GOAT}_{\text{MOTIF,PPI}}$, $\text{GOAT}_{\text{MOTIF,3D,PPI}}$ to indicate joint inputs in GOAT.

We measure the per-label accuracy using Macro and Micro AUC which are the unweighted and weighted averages of the AUC at each label, respectively. We are interested in the accuracy for rare labels because these labels are closer to the true functions of the proteins. Rare labels affect Macro AUC more than Micro AUC, so we will report Macro AUC more often. We also compute recall-at-k (R@k) which measures the per-protein accuracy. In practice, a classifier would return k of the most probable labels for an unknown sequence, which a curator can then review. These k labels are referred to as *top-k labels* because they are the k labels having the highest predicted probabilities for an input sequence. For one protein, R@k measures the fraction of correct labels retrieved among the top-k labels. We report the final R@k which is the average R@k over all test samples.

5.6 GOAT base implementation

For the Transformer parameters in GOAT, we set input embedding at size 256 (that is $E^G \in \mathbb{R}^{932\times256}$ for MF), and intermediate vectors at size 512. To keep our software GOAT manageable for all users, we implement Transformer with only one head and 12 layers. We initialize the GO embedding E^G as the pretrained embedding $\text{BERT}_{\text{LAYER12}}$ in Duong *et al.* [28] which transforms the GO definitions into vectors, where GOs having related definitions will have comparable vectors. Using pretrained GO embeddings reduces the number of parameters in the Transformer, which can also reduce overfitting, use less GPU memory, and decrease run time. We train all GOAT variations on a single GTX 1080 Ti with 11GB memory for all three datasets.

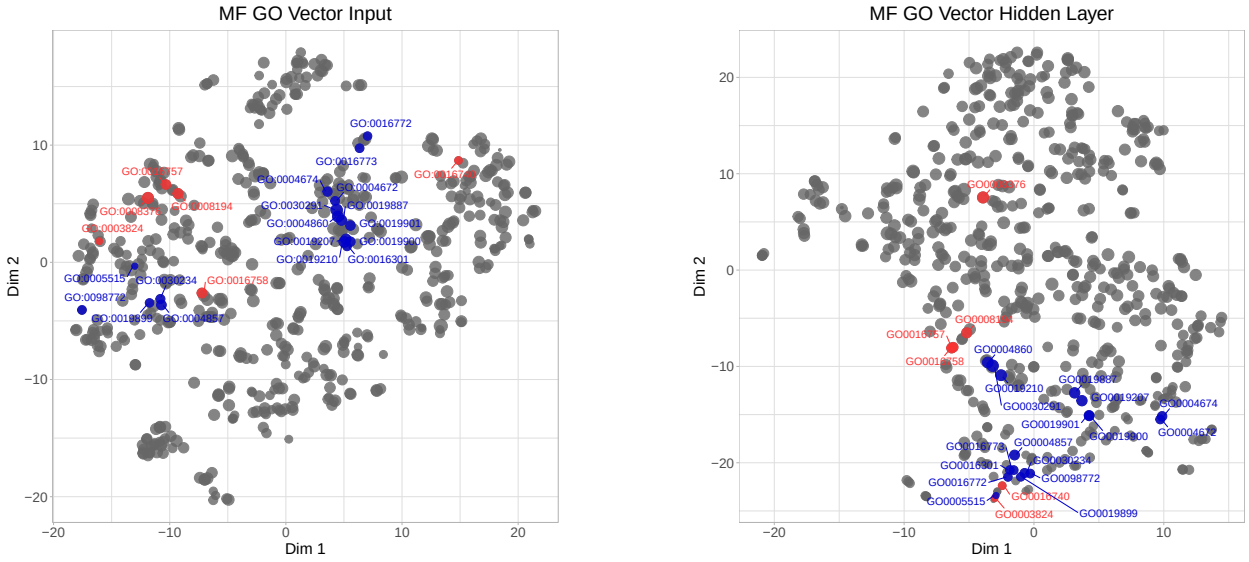
We first compare the base implementations of GOAT and DeepGO, and in the next section we will include $\text{MetaGO}_{\text{BLAST}}$. In BP, MF and CC data, $\text{GOAT}_{\text{BASE}}$ exceeds $\text{DeepGO}_{\text{BASE}}$ in Macro and Micro AUC, indicating that the base implementation of GOAT attains better per-label accuracy (Table 5.1.A row 3 and 5). On per-protein accuracy, we select R@50, R@30, and R@30 for the BP, MF and CC data, respectively (approximately 5% of total label size). GOAT increases recall rates by a small amount.

Recall for the entire label set can be affected by common terms which are often easier to classify compared to rare labels. Accurately predicting rare terms is more important for the proteins, because rarer labels describe more detailed biological events which reflect the true properties of the unknown proteins. Table 5.1.B shows the R@k for 232 BP, 143 MF, and 110 CC rare labels which appear below the 25^{th} quantile occurrence frequency in the label sets. For recall rates to make sense in this case, we compute recall rates for the proteins annotated by at least one of these rare labels. GOAT has a noticeable improvement over DeepGO, especially for larger top-k label sets (Table 5.1.B row 3 and 5).

We next evaluate whether our Transformer adaptation can learn the co-occurrences of labels. Duong *et al.* [28] noticed in their GO embeddings that when one of the child-parent GO labels describes very broad biological events (e.g. low IC), then their vector representations may be far apart. This fact implies that for Transformer to work well, to some degree it must learn the co-occurrences of labels and adjust E^G so that any two related GO labels (regardless of their frequencies in the train data, IC values and distance to roots) will have comparable vectors. To observe that Transformer can implicitly learn label co-occurrences, we compare the T-SNE plots of the input GO embedding E^G and its output h_{12,g_i} from the Transformer layer 12 which is directly passed into the classification layer.

For every input protein, we have a different value of h_{12,g_i} for the same label g_i because h_{12,g_i} is function of the vector representing the amino acids. We apply our trained Transformer on the test set, and take the average h_{12,g_i} over each input proteins in test data (denoted as $\bar{h}_{12,g_i}$). We compute $\bar{h}_{12,g_i}$ from the test data because these proteins are not seen in training and provide a more realistic evidence. We use $\bar{h}^G_{12}$ to denote the set of $\bar{h}_{12,g_i}$ for all g_i.

Figure 5.1: T-SNE plot of (a) input GO embeddings and (b) their transformed values created by Transformer layer 12. Red and blue nodes are the ancestors of the term GO:0008376 and GO:0030291 respectively.



(a) **(b)**

Figure 5.1 shows the T-SNE plot of E^G for the MF labels in DeepGO dataset; we highlight two terms GO:0008376 (red) and GO:0030291 (blue) and their corresponding ancestors in the same colors. The dot sizes are scaled by IC values, where a smaller size implies lower IC (so the label is more common). Smaller dots tend to cluster well together but large dots do not. For example, consider the term GO:0016740 and its parent GO:0003824 (far right and far left red nodes in Fig. 5.1(a)) which should often co-occur because in our dataset ancestors of an assigned label are included as the ground truth labels. For Transformer to work well, it should reposition the h_{12,g_i} vectors representing GO:0016740 and GO:0003824 closer together.

The T-SNE plot of $\bar{h}^G_{12}$ from Transformer shows that the red and blue nodes have clustered more closely compared to E^G. Now, the blue dots are gathering at the bottom of Fig. 5.1(b), as compared to being two separated groups in Fig. 5.1(a). Our two running examples GO:0016740 and GO:0003824 have moved closer to each other, the two red nodes at bottom of Figure 5.1(b). However, the lowest child node GO:0008376 remains far from its ancestors (top most red node). The red and blue dots are not yet tightly compacted into two dense clusters, and so there is room for further development. In conclusion, Transformer weakly models the co-occurrences of labels

even when such a constraint is not explicitly enforced.

5.7 Motifs in amino acid sequences as features

Before neural network models, earlier methods integrated BLAST as a key component. BLAST retrieves annotated proteins in the database that share similar amino acid patterns with the unknown sequence, and then assigns the GO labels of these retrieved proteins to the unknown query. Loosely speaking, the amino acid patterns shared by the training sequences can be considered as the key factors in BLAST-based methods. We evaluate whether neural network models can automatically learn these key patterns from the training sequences, and explain how to introduce these patterns as input features to GOAT.

For fair comparison, we select a strong BLAST baseline $\text{MetaGO}_{\text{BLAST}}$ and apply it to the same DeepGO datasets [107]. We emphasize that $\text{MetaGO}_{\text{BLAST}}$, which matches multiple related sequences to the query, has much stronger performance than the BLAST baseline used in DeepGO where only a single best matching sequence is selected [60, 107]. For BLAST and PSI-BLAST, we perform the experiments using both e-value at 10 and 100. Lower e-values leave too many unmatched testing sequences; for example, in the CC dataset at e-value 1, only 7236 out of 8886 test samples match to some sequences in the train data. Moreover, in the context of finding possible protein functions, a higher e-value can allow for higher recall rates.

On the entire label set, when considering all the metrics, the base DeepGO and GOAT outperform $\text{MetaGO}_{\text{BLAST}}$ only for CC data, but not for MF and BP where $\text{MetaGO}_{\text{BLAST}}$ has higher recall rates (Table 5.1.A row 1–3 and 5). On rare labels, $\text{MetaGO}_{\text{BLAST}}$ yields better recall rates especially for small top-k label set. For larger top-k label sets, $\text{GOAT}_{\text{BASE}}$ comes close to $\text{MetaGO}_{\text{BLAST}}$, whereas DeepGO cannot (Table 5.1.B row 1–3 and 5). The CNN in DeepGO and Transformer in GOAT have not fully captured the amino acid patterns shared among the sequences with related functions, which BLAST or PSI-BLAST can detect.

We had hoped that the complex Transformer architecture in GOAT would learn key amino acid patterns, but this is not the case. There are information about the amino acid sequences that we

must explicitly specify for GOAT to function better. For this reason, we use the motifs extracted by string-matching methods as inputs to our method GOAT. For example, we can apply PROSITE to scan the protein database for known motifs in a given input sequence [85]. Loosely speaking, we are combining motif-based methods and the Transformer architecture into one pipeline by taking the output of motif-based models and passing them as inputs to our Transformer. Our strategy is different from an ensemble approach that averages the predictions of independent classifiers.

In the Uniprot database, each protein already has a Family & Domains section describing its key regions [96]. We add these region types as input features into our method GOAT. We emphasize that not all region types are meaningful for predicting labels in a certain ontology. For example, Serine/threonine-protein kinase TBK1 (UniProtKB number Q9UHD2) has a kinase domain at position 9-310 (PROSITE annotation rule PRU00159). The kinase domain is involved in a wide range of biological processing at various locations in the cell like metabolism, transcription, cytoskeletal rearrangement and movement, and cell apoptosis and differentiation [30, 83]. Thus, this kinase domain in Q9UHD2 tells us which Molecular Functions are more likely to be assigned, but is less informative about which Biological Processes or Cellular Components Q9UHD2 will have.

For each protein in the original DeepGO datasets, we download its sequence annotation rule (e.g. PROSITE rule) from the 2019 data at https://www.uniprot.org/. We do not consider region types for sequences that have changed in length; otherwise, we consider region types only for portions that have not changed in amino acid composition. Uniprot data divides region types into subgroups. Some subgroups require curated comments and are not truly applicable for analyzing new proteins; for example, the subgroup Domain Non-positional Annotation is not determined by sequence analysis.

We use the following six subgroups which can be found by sequence analysis: Zinc finger (e.g. C2H2-type), Repeat (e.g. AA tandem repeat), Motif (e.g. LXXLL motifs), Compositional bias (e.g. Asp/Glu-rich), Coiled coil (e.g. Leucine-zippers), Domain (e.g. Ser/Thr kinase domain). In the original DeepGO datasets, we found 1629, 1450 and 1655 amino acid region types for the BP, MF and CC train data, respectively. Region types found in test sequences but not seen in the train

data are set as zero; effectively we treat these cases as if the region types do not exist. To model the region types in the amino acid sequence, we apply the region-type embedding R explained in Section 5.4; for example, in BP ontology we will have a embedding $R \in \mathbb{R}^{1629\times 256}$. We will use the name MOTIF to denote all types of information in these six subgroups reported by The UniProt Consortium [96].

On the entire data, $\text{GOAT}_{\text{MOTIF}}$ obtains better AUCs and comparable recall rates to $\text{MetaGO}_{\text{BLAST}}$ (Table 5.1.A row 6). On rare labels, $\text{GOAT}_{\text{MOTIF}}$ has subpar recalls when the top-k label set is small. However, at larger top-k label set, especially in MF and CC data, GOAT retrieves more correct labels (Table 5.1.B row 6). Our result also highlights an important point. Ideally, a neural network should teach itself the patterns associated with certain key protein functions. However, a neural network may fail to learn such key information, and needs these inputs to be explicitly provided.

To observe how the MOTIF are analyzed by the Transformer architecture in GOAT, we select the human protein Serine/threonine-protein kinase TBK1 (UniProtKB Q9UHD2) in DeepGO test data. As described in Section 5.4, we concatenate the amino acid sequence and GO labels into one single input into GOAT. Q9UHD2 is 729 amino acids long, and when predicting MF labels, the input into GOAT is a string of amino acids and GO labels of length 1318 (from 729 + 589 MF labels). We integrate into the Transformer framework the three key domains in Q9UHD2 derived from sequence analysis methods; these are Protein Kinase at 9-310, Ubiquitin-like at 309-385, and Coiled coil at 407-713.

We plot the attention heatmap of α_{jk} in each layer. α_{jk} measures how much position k contributes toward position j (Eq. 5.3). Each row in the heatmap adds to 1. The heatmap is divided into four quadrants. The first quadrant shows the interactions of GO labels among themselves (e.g. α_{jk} for $j,k \in [730,1318]$), the second shows contribution of amino acids toward the GO labels, the third shows interactions of amino acids among themselves (e.g. α_{jk} for $j,k \in [1,729]$), and the fourth shows contribution of GO labels toward the amino acids.

The Transformer without region-type embeddings has a noisy attention heatmap (Fig. 5.3). Transformer with region-type embedding displays meaningful patterns (Fig. 5.2). For example,

layer 1 illustrates the cross interactions between Protein Kinase and Coiled coil domain (black boxes quadrant 3); whereas layers 4 and 8 show interactions within the Protein Kinase and Coiled coil themselves. In layer 12, the final vectors representing GO labels receive more attention from the Protein Kinase than the other regions (left most box in quadrant 2). Because these layer 12 vector output are sent to the classification layer, we can assume that the Protein Kinase region contributes more to the label classification compared to the other domains. This observation is consistent with the true molecular functions of Q9UHD2 which are: phosphoprotein binding, protein kinase activity, protein phosphatase binding, and protein serine/threonine kinase activity.

Figure 5.2: Heatmap of the attention values α_{jk} in each layer when analyzing the protein kinase TBK1 (UniProtKB Q9UHD2). The three key regions of this sequence (separated by red lines) are explicitly given as inputs to the Transformer model. The first quadrant shows the interactions among the GO labels, the second shows contribution of amino acids toward the GO labels, the third shows interactions of amino acids among themselves, and the fourth shows contribution of GO labels toward the amino acids.

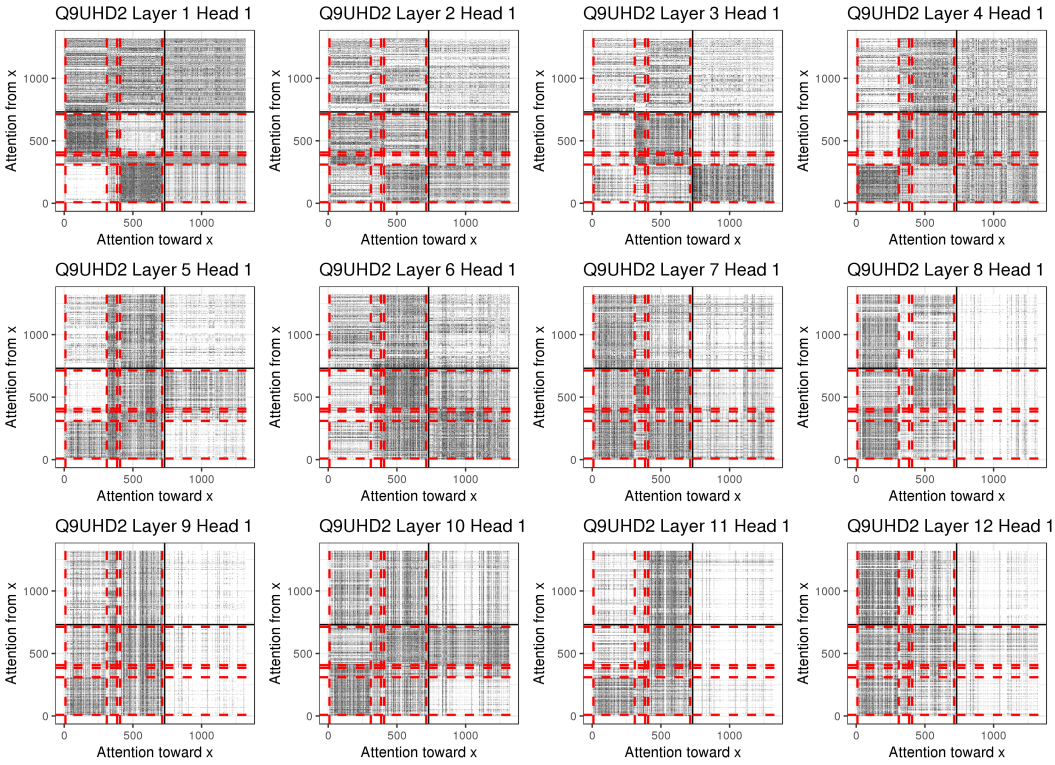
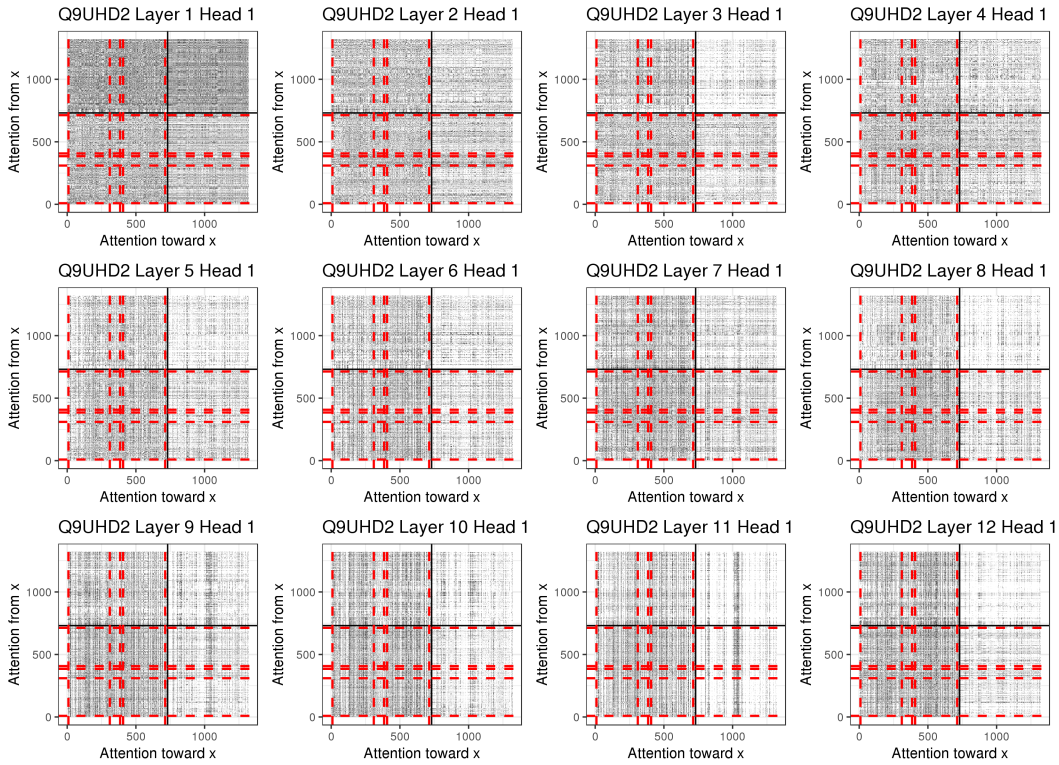


Figure 5.3: Heatmap of the attention values α_{jk} in each layer. Motifs of the sequences are not explicitly given as inputs to this Transformer model.



5.8 Other protein metadata as features

Zhang *et al.* [107] evaluate how PPI network and 3D structure data affect the annotation accuracy for methods built from BLAST. We assess the contributions of these components in the context of a neural network classifier. Unlike Zhang *et al.* [107] which use ensemble of the models built from each component, we will model all these components as inputs to the neural network. We consider these other protein metadata as any information about the proteins coming from some external resources besides the Uniprot data. For example, one can train a neural network on the PPI network (e.g. STRING database) to transform interacting proteins into similar vectors and then use these vectors as extra features for the classifier [2, 16, 95]. Introducing these protein vectors into annotation methods is motivated by the fact that interacting proteins (ideally encoded into similar vectors) should have related functions (e.g. found in the same biological processes and cellular locations). It is only recently that proteins in knowledge graph have been transformed into vectors

via neural network models [16]. We will not build new models to encode proteins from knowledge graph, and reserve this topic for future work. We will focus on evaluating how much can protein vectors built from external data sources increase the prediction accuracy.

Table 5.1: Evaluation on the preprocessed Uniprot data, containing 932 BP, 589 MF, and 439 CC labels for 9095, 6294, and 8886 testing proteins in BP, MF and CC data.

Table 5.1.A Macro and Micro AUC, and R@k on the entire label sets.

		BP			MF			CC		
		Macro	Micro	R@50	Macro	Micro	R@30	Macro	Micro	R@30
BLAST Psi-BLAST										
1	Evalue10	65.99	81.34	48.02	76.51	87.12	69.75	64.17	89.36	77.24
2	Evalue100	66.61	84.41	48.84	78.67	90.34	68.06	65.62	92.54	80.23
DeepGO										
3	BASE	62.60	82.17	46.13	73.90	87.49	59.63	67.12	93.07	81.47
4	+PPI	82.16	90.31	56.97	84.97	92.40	70.19	87.51	96.76	**87.98**
GOAT										
5	BASE	67.69	84.46	47.40	78.67	89.43	60.46	74.94	93.95	82.98
6	+MOTIF	71.04	85.64	48.65	82.53	91.12	66.19	77.62	94.51	83.01
7	+MOTIF+3D	72.62	86.09	50.76	85.38	92.72	69.23	79.57	94.95	84.22
8	+MOTIF+PPI	**83.68**	**90.49**	**57.81**	**88.92**	**93.98**	**72.16**	**90.76**	**97.23**	87.95

Table 5.1.B R@k on the most uncommon 232 BP, 143 MF, and 110 CC labels of the entire label sets.

		BP			MF			CC		
		R@10	R@30	R@50	R@10	R@20	R@30	R@5	R@10	R@20
BLAST Psi-BLAST										
1	Evalue10	25.13	38.34	47.33	51.11	54.93	57.51	25.09	30.91	40.48
2	Evalue100	21.79	36.06	46.29	48.37	55.98	60.98	21.91	29.72	39.20
DeepGO										
3	BASE	11.88	27.23	39.75	27.07	40.38	50.47	13.28	24.57	38.99
4	+PPI	**45.62**	**64.93**	74.55	52.85	63.64	71.98	**53.14**	**64.85**	76.61
GOAT										
5	BASE	13.49	31.73	44.54	30.95	48.31	58.87	17.89	26.47	39.90
6	+MOTIF	17.27	35.58	48.11	41.72	55.80	64.76	19.06	31.03	48.04
7	+MOTIF+3D	23.66	42.90	55.06	47.90	62.49	72.23	27.36	39.72	57.17
8	+MOTIF+PPI	42.84	64.53	**74.78**	**59.49**	**74.77**	**82.67**	48.04	62.14	**77.68**

5.8.1 High-level 3D structures of proteins

High-level structures within a protein may be derived from the patterns within its amino acid sequence. We evaluate the embeddings of Bepler and Berger [9] that capture these high-level

configurations. Bepler and Berger [9] applied a 3-layer Bidirectional Long-Short Term Memory (BiLSTM) to encode an amino acid sequence into a matrix. Bepler and Berger [9] trained their model on the SCOP database and residue-residue contact prediction. The SCOP database describes the major classes of 3D-structures often seen in proteins (not detailed final 3D shapes of proteins). Their model was trained on SCOPe ASTRAL 2.06 dataset with 22,408 amino acid sequences, and each training epoch has 100,000 pairs sampled from these 22,408 sequences. From SCOP, Bepler and Berger [9] predict whether two protein sequences have no relationship, class-level relationship, fold-level relationship, superfamily-level relationship, or family-level relationship (e.g. label $y = 0, 1, 2, 3, 4$) [9]. For example, two proteins with the same Rossmann-fold structural motif have a class-level relationship ($y = 1$ in this case). Residue-residue contact prediction is applied within the same protein sequence; the objective is to predict whether each pair of positions i, j within the same protein are close by (distance less than 8 angstroms) in the 3D structure. Bepler and Berger [9] provide the pretrained encoder which can return a matrix for any amino acid sequence, even those not used in training. We take the mean-pool of this matrix to represent the entire sequence.

Because motif information is a key input for GOAT, we integrate the high-level 3D structure embeddings with $\text{GOAT}_{\text{MOTIF}}$. 3D structures and motifs are targeting two different kinds of information; for example, a kinase domain does not strictly entail a specific folding pattern. $\text{GOAT}_{\text{MOTIF,3D}}$ improves upon $\text{GOAT}_{\text{MOTIF}}$ (Table 5.1.A row 7). For a small top-k label set, GOAT finds fewer correct labels than $\text{MetaGO}_{\text{BLAST}}$. However, for a larger top-k label set, GOAT is better than $\text{MetaGO}_{\text{BLAST}}$, where our recall rates increase by about 9%, 12%, and 17% in BP, MF and CC data respectively (Table 5.1.B row 7).

When we replace the PPI network in DeepGO with the vectors from SCOP in Bepler and Berger [9], the performance significantly decreases. Macro AUC in DeepGO drops from 82.16 to 66.54 in BP, from 84.97 to 77.78 in MF, and from 87.51 to 68.93 in CC data. This decrement is anticipated as we will argue in the next section, that protein interaction network has more impact than 3D structure information.

The embeddings in Bepler and Berger [9] capture less information than the protein 3D structure

data used in MetaGO. Besides its BLAST component, MetaGO uses I-TASSER to derive the final 3D configuration of an input sequence (not just high-level secondary structures) [106, 107]. I-TASSER does not provide embedding for an input protein and cannot yet be integrated into neural network model [106]. We will not compare the neural network models against the MetaGO integrated with I-TASSER.

5.8.2 Protein-protein interaction network

We evaluate protein vectors that capture their relatedness in a protein-protein interaction network. To be consistent with DeepGO, we use the same protein vectors in their paper as input features for our GOAT. These vectors are created following the method in Alshahrani *et al.* [2]. In brief, DeepGO uses vectors representing the protein names in a PPI network that has 8,478,935 proteins, and 11,586,695,610 edges total (derived from STRING database). DeepGO uses DeepWalk to generate sentences from the network, and apply Word2Vec on these sentences to create the vector embedding for the protein names [71, 75]. Effectively, interacting proteins will have similar vectors.

Because motif information is a key input of GOAT, we integrate PPI network data on top of our $\text{GOAT}_{\text{MOTIF}}$. $\text{GOAT}_{\text{MOTIF,PPI}}$ exceeds the other Transformer models by large margins (Table 5.1.A row 8). On the entire label sets, with PPI network information GOAT and DeepGO perform similarly (Table 5.1.A row 4, 8). Only for rare MF labels does GOAT exceed DeepGO by noticeable margins (Table 5.1.B row 4, 8).

Intuitively, it is reasonable that PPI network dominates the information from amino acid sequences at classifying BP and CC labels, but not for MF labels. For example, two interacting proteins can have distinct 3D structures and sequences (and thus motifs); yet, they involve in the same biological process and sometimes found at the same cellular components. The same two interacting proteins however can have dissimilar molecular functions because they can induce very different chemical reactions.

PPI network embeddings in DeepGO do not need amino acid sequences to retrieve vectors representing the proteins. However, this method will not return vector representations for novel proteins not yet existed in the database. In practice, for proteins not yet well studied, we may need

a different approach and other metadata for such proteins. We reserve this topic for future research work.

5.9 Evaluation on sparse GO labels

Many GO terms annotate only a few proteins because protein functions can be very unique. Parametric predictors must handle sparse labels to predict terms closely resemble the true protein functions. In practice, parametric models can fail when the training data has too many sparse labels. In such cases, neural network accuracy will drop as we need each label to have enough samples to reliably train the model parameters. It is important to evaluate neural network models against $\text{MetaGO}_{\text{BLAST}}$ which does not have trainable parameters.

To create datasets containing more rare labels, we reuse the same proteins in the preprocessed Uniprot data of Section 5.5. Now, we include labels with at least 50, 10, and 10 occurrences in the BP, MF and CC train data, whereas the same criteria before were 250, 50 and 50. Our larger datasets now have 2980 BP, 1697 MF and 989 CC labels, respectively (versus the original 932 BP, 589 MF, and 439 CC labels). For each added term, we include its ancestors as the gold-standard labels, so that most of the labels in the original data now have higher occurrence frequencies.

We train the models on the entire larger dataset. We will include DeepGOCNN, GOAT_{PPI} and $\text{GOAT}_{\text{MOTIF,3D,PPI}}$ in this experiment. We do not include PPI network into DeepGOCNN because the DeepGOCNN code does not accept PPI network data [59]. GOAT_{PPI} evaluates the contribution of the PPI network alone, and $\text{GOAT}_{\text{MOTIF,3D,PPI}}$ evaluates our most complete model. We will evaluate the accuracy for the labels found in the original preprocessed Uniprot data. However, we are less interested in these common labels, and will also evaluate the added 2048 BP, 1108 MF and 550 CC labels which are sparse. For example, 95% of the added labels occur below 252, 34, and 68 times in the BP, MF and CC training data, respectively. Table 5.2 shows the results for the common and sparse labels. We include Macro AUC to judge whether a classifier obtains higher recall rates by making too many false positives. We discuss four keys points in these results.

First, $\text{MetaGO}_{\text{BLAST}}$ is very competitive against the base implementations of deep learning

Table 5.2: We increase the label sets in the preprocessed Uniprot data from 932 BP, 589 MF, and 439 CC labels to 2980 BP, 1697 MF and 989 CC labels. Models are trained on the entire expanded label sets.

Table 5.2.A Evaluating the original 932 BP, 589 MF, and 439 CC labels. Largest R@k is about 10% of BP, MF and CC labels.

		BP				MF				CC			
		Macro	R@k			Macro	R@k			Macro	R@k		
		AUC	40	70	100	AUC	10	30	60	AUC	10	20	40
BLAST Psi-BLAST													
1	Evalue10	66.66	43.38	54.94	62.16	75.31	52.21	68.92	76.91	65.07	56.02	72.92	82.96
2	Evalue100	67.12	43.19	55.27	62.91	77.47	49.07	66.87	76.63	66.37	56.31	74.61	85.26
DeepGO													
3	BASE	64.49	40.89	53.02	61.38	73.39	41.38	58.17	70.22	68.61	57.52	75.82	86.23
4	+PPI	81.58	49.86	63.48	71.75	83.73	48.48	68.44	79.86	88.05	61.37	80.00	91.11
DeepGOPlus													
5	DeepGOCNN	69.92	41.50	55.15	63.77	78.56	45.70	63.12	71.46	75.01	60.21	77.57	87.27
GOAT													
6	BASE	66.74	40.41	52.87	61.38	76.79	41.00	58.23	70.53	74.75	59.82	77.20	87.16
7	+MOTIF+3D	69.93	43.12	55.54	63.54	84.40	48.83	67.56	78.51	78.34	59.50	77.31	87.98
8	+PPI	83.02	50.40	63.87	72.10	87.03	48.74	68.44	79.91	91.34	**62.68**	**80.90**	91.68
9	+MOTIF+PPI	82.82	50.26	63.57	71.96	85.89	47.86	67.92	79.63	90.41	61.50	80.08	91.35
10	+MOTIF+3D+PPI	**84.02**	**51.50**	**65.31**	**73.61**	**88.61**	**53.75**	**73.62**	**84.12**	**91.41**	62.24	80.86	**91.72**

Table 5.2.B Evaluating the added 2048 BP, 1108 MF and 550 CC labels which are sparse. We evaluate only 7850, 2671, and 1848 proteins having these labels out of the 9095, 6294, and 8886 samples in BP, MF and CC testing data. Largest R@k is about 5% of BP, and 10% of MF and CC labels.

		BP				MF				CC			
		Macro	R@k			Macro	R@k			Macro	R@k		
		AUC	40	70	100	AUC	40	70	100	AUC	10	30	50
BLAST Psi-BLAST													
1	Evalue10	64.10	30.30	34.84	37.88	68.79	38.67	41.09	43.42	60.20	20.80	29.15	33.13
2	Evalue100	64.44	29.53	34.57	38.02	69.93	38.58	43.32	45.81	60.03	23.47	35.23	39.78
DeepGO													
3	BASE	63.53	23.14	27.98	32.01	68.27	16.62	23.83	29.60	60.45	22.49	31.53	37.96
4	+PPI	84.66	**48.06**	55.86	61.02	80.22	41.61	48.61	53.39	83.17	47.89	60.77	66.48
DeepGOPlus													
5	DeepGOCNN	69.35	27.32	33.34	37.92	70.31	17.92	24.38	28.87	61.55	24.38	33.36	39.99
GOAT													
6	BASE	65.68	23.78	28.82	32.69	73.58	19.41	27.88	34.19	68.22	25.22	35.98	42.53
7	+MOTIF+3D	70.00	27.70	33.50	38.03	79.50	30.52	38.74	45.44	69.61	25.73	37.82	44.42
8	+PPI	86.77	43.51	52.52	58.54	85.61	41.77	52.46	58.51	88.12	41.98	60.02	68.70
9	+MOTIF+PPI	87.32	46.22	55.06	61.08	84.93	45.69	55.32	61.11	90.01	**50.08**	**67.15**	73.89
10	+MOTIF+3D+PPI	**88.08**	46.78	**56.24**	**62.49**	**87.57**	**50.92**	**60.77**	**66.64**	**90.07**	47.96	65.30	**74.36**

methods, especially for small top-k label sets. Only for CC labels, we observe that neural network models are better for both common and sparse labels. In the future work, we will explore ways to create an ensemble between a BLAST-based and a neural network model. For example, the combination weights of the predictions for these two models can be trained with the neural network parameters. The neural network would then primarily focus on labels that are hard to predict by a BLAST-based method.

Second, amino acid motifs and high-level structural motifs, which can be inferred from the sequence data, increase the accuracy of GOAT on BP and CC labels, and has the highest effect on MF labels. For BP and MF labels, motifs and high-level 3D structure data enable GOAT to come close to $\text{MetaGO}_{\text{BLAST}}$ (Table 5.2 row 6, 7). Future neural network model should leverage these two data types either by following our approach or any other means.

Third, PPI network has the most impact on the prediction accuracy. With PPI, even the simple DeepGO surpasses $\text{MetaGO}_{\text{BLAST}}$ (Table 5.2 row 4, 8–10). This evidence supports our earlier hypothesis that PPI network can dominate sequence information; that is, two interacting proteins should be involved in the same biological processes even when their sequences display dissimilar motifs, 3D structures, or any other types of information extracted by the neural network. BLAST search can infer motifs and high-level 3D structure from amino acid sequences, but not the PPI network. It is unfair to compare $\text{MetaGO}_{\text{BLAST}}$ against deep learning models with PPI embeddings; however, our results highlight the major impact of having the PPI network embeddings.

Fourth, within the three neural network approaches, DeepGO has the lowest accuracy but understandably, it is also the simplest model. For BP labels, DeepGOCNN outperforms $\text{GOAT}_{\text{BASE}}$ and is about the same as $\text{GOAT}_{\text{MOTIF,3D}}$. For BP labels, having many CNN components may be enough to capture key amino acid patterns. This is not the case for $\text{GOAT}_{\text{MOTIF,3D}}$ on MF and CC labels, where $\text{GOAT}_{\text{MOTIF,3D}}$ is better than DeepGOCNN and the improvement is most prominent for rare labels (Table 5.2 row 5, 7). Between DeepGOCNN and the base architecture of GOAT, for MF and CC labels, there is a trade-off in the accuracy of common and sparse labels. DeepGOCNN is better than $\text{GOAT}_{\text{BASE}}$ at classifying common labels, but $\text{GOAT}_{\text{BASE}}$ is better for sparse labels (Table 5.2 row 5, 6). Because sparse labels are closer to the true protein functions, $\text{GOAT}_{\text{BASE}}$

would provide more precise predictions than DeepGOCNN. Both DeepGOCNN and $\text{GOAT}_{\text{BASE}}$ apply their own large neural network architectures on the amino acid sequences. DeepGOCNN uses 8192 components of CNN, and GOAT uses 12 layers of Transformer. The main differences are that GOAT uses an attention-based network, and that GOAT takes label embeddings. Both factors are likely to have improved the accuracy for rare MF and CC labels.

5.10 Summary and discussion

We introduce **GO a**nnotation method with **T**ransformer (GOAT), a deep learning classifier based on the neural network with attention mechanism. Besides the raw amino acid sequences, GOAT can be trained with three types of metadata: motif information, high-level 3D structures and PPI network. Each metadata increases the accuracy of GOAT, but the PPI network has the highest impact. GOAT is better than the most recent CNN-based classifier DeepGOCNN for rare MF and CC labels [59]. With metadata, in some cases GOAT can be at least equivalent to $\text{MetaGO}_{\text{BLAST}}$ [107].

MetaGO of Zhang *et al.* [107] has also combined sequence data, 3D structure and PPI network information to annotate GO labels. In MetaGO, each type of metadata is used to build its own classifier, and then these independent classifiers are combined to produce the final prediction. For example, Zhang *et al.* [107] built their $\text{MetaGO}_{\text{BLAST}}$ as an independent unit from the their two classifiers that uses PPI network and 3D structure data. The reason for their strategy is that BLAST algorithm, which is similar to Smith-Waterman, cannot take information about the interacting partners and 3D structure of the input sequences [3]. Unlike MetaGO, GOAT can jointly analyze the raw amino acid sequences, motifs, protein 3D structures and PPI network. In the future work, we wish to integrate components of MetaGO into GOAT, and vice versa.

We discuss a key property of our Transformer in the context of GO embeddings. This Transformer learns the co-occurrences among the labels; for example, the last layer in Transformer returns comparable vectors for a child and parent GO label (Fig. 5.1). GO embeddings produced by our Transformer are not equivalent to the embeddings produced by factorizing the co-occurrence

matrix of GO labels, because GO embeddings from our Transformer are also affected by information from the amino acid sequence (Fig. 5.2). For our future work, we will integrate embedding learned from co-occurrence frequencies into our Transformer framework.

Next, GOAT is capable of zeroshot learning, where it classifies labels not yet observed in the training data. The reasons are because (1) the model parameters are just the matrices V, Q, and K in Eq. 5.5 and (2) the input GO embeddings is the pretrained $\text{BERT}_{\text{LAYER12}}$ embeddings which will provide vector representation for any GO label from its definition. Unfortunately, zeroshot learning requires GOAT to handle a much larger label set, because for every added label, we must also include all the parent labels of that label. Theoretically, this is not a problem, but in practice, due to GPU limitation, we will need to devise a better training strategy.

We outline two limitations of our adaptation of Transformer. First, to make our software GOAT accessible to many users, we have reduced the number of parameters that Transformer often assume in other machine learning applications. GOAT fits in one GPU having 11GB memory, whereas Rives *et al.* [82] trained a language model on protein sequences using a 36-layer Transformer with multiple GPUs. We expect that our GO annotation accuracy to increase if we train our model with more parameters and on more samples. Second, we do not pretrain our Transformer. For example, before predicting GO labels, Transformer can be pretrained only on protein sequences with the following objective. We can remove amino acids from a sequence, and then use Transformer to retrieve these missing amino acids. Pretraining helps the parameters in Transformer to converge better for the downstream tasks. However, pretraining requires a lot of data; for example Rives *et al.* [82] pretrained their 36-layer Transformer on 250 million sequences. For our future work, we will consider training a large-scale Transformer to predict GO labels for protein sequences.

CHAPTER 6

Conclusion

Since the past two decades, many biological datasets have emerged, capturing different aspects of gene expressions and protein functions. This thesis presents techniques to jointly analyze these datasets, so that we gain more holistic views of the biological mechanisms and obtain higher detection power for biological signals of interest. In particular, we study the two following problems: associating genotypes and gene expressions, and predicting protein functions.

The first problem focuses on ways to better discover eGenes in the GTEx data. eGenes are considered as potential causal factors for many phenotypes of interest [50, 66]. An eGene is defined as a gene that has at least one SNP associated with its expression. eGenes are identified via eQTL studies, where the objective is to correlate the genotypes with the gene expressions. We present approaches to increase the number of discovered eGenes when analyzing a single tissue and then many tissues altogether.

In a single tissue, we leverage the Roadmap data to obtain extra knowledge about the SNPs in the eQTL studies; for example, SNPs located in DNase hypersensitive sites are likely to affect gene expressions. We evaluate the effects of the following genomic regions in the eQTL studies: $\pm$150kb from transcription start sites, DNase hypersensitive sites, and histone modification sites. Each type of genomic annotation increases the number of discovered eGenes by at least 16% more than the baseline without genomic annotation, and transcription start sites have the most impact (57% increment).

For many tissues, the GTEx data contains few sample sizes, making it difficult to detect eGenes in each tissue. To overcome this problem, we design a meta-analysis that combines the eQTL results of the 44 tissues in the GTEx data. Our model also has a new parameter to capture the

correlations of SNP effects on the gene expressions in these tissues. Our method discovers 20% more eGenes than the standard tissue-by-tissue approach.

On the topic of predicting protein functions, since 2015 there has been many deep learning methods to analyze different aspects of the protein sequences [2, 9, 16, 59]. Recently, neural network models were built to annotate functions of proteins based on their amino acid sequences. To gain more knowledge about the proteins, besides the amino acid sequences, these classifiers also include various protein metadata like the protein 3D structures and the protein-protein interaction network.

These deep learning classifiers have not yet leveraged the metadata presented in the function labels. Protein functions are described by GO labels which contain two types of metadata, the definitions of the GO labels and their hierarchical relationships. We design and compare ways to embed the two types of GO metadata by using the recent neural network methods of BiLSTM, ELMo, BERT, and GCN. Embedding GO label definitions via our BERT implementation most faithfully captures the relationships of the GO labels.

Next, we introduce a new GO label classifier based on the Transformer architecture (named GOAT) that uses the GO label embeddings and the protein metadata. GOAT models all pairwise interactions of the amino acids in the same sequence, surpassing the constraint of CNN approaches [59]. GOAT takes the following protein metadata: key amino acid patterns, high-level 3D structures, and PPI network. Together with the label embeddings, these metadata increase the prediction accuracy, and the PPI network data has the highest impact. For rare MF and CC labels which are closer to the true protein functions, GOAT obtains higher accuracy than the recent CNN models.

In conclusion, for association studies and protein function prediction, borrowing information from related data resources improves the results.

CHAPTER 7

Future Work

Many genetic datasets contain individuals primarily of European ancestry, and the GTEx data is no exception. Only in its most recent release v8 in Aug 2019, there are about 15% of non-European donors [37]. Recent publications have been analyzing the impact of population structure on the eQTL analysis, but not so much work has been done to find eGenes [109]. Because eGenes are viewed as candidate genes for phenotypic variations, it is important to characterize the differences for the lists of eGenes discovered in each population. This result would provide deeper understanding of the genetic factors for a phenotype in each population. To further improve the detection power for the eGenes in each population, we may integrate population-specific epigenomic data into the eQTL analysis.

On predicting protein functions, we propose the following directions. Having PPI network data significantly increases the accuracy of GO label classifiers. Yet, providing a classifier with this information is counter-intuitive because interacting partners of a new protein are not identified without experimental evidence. Future work should focus on building method that predict relationships of proteins using just their amino acid sequences. These outputs can then be passed to a downstream GO label classifier.

We may also shift our attention from proteins to non-coding RNAs (ncRNAs). ncRNAs have been found to play critical roles in cancer development [57, 67, 108]. We may design a function classifier for RNA sequences, by following the same idea for classifier of protein functions. Finally, we desire a classifier that will scale well to a large label set (e.g. 44,531 labels in the GO database as of March 2020). At this moment, deep learning classifiers based on CNN or Transformer cannot scale to this many labels. For future endeavor, this problem is challenging in both the mathematical

and hardware perspective.

BIBLIOGRAPHY

[1] Abraham, B. and Ledolter, J. (2006). Introduction to regression modeling.

[2] Alshahrani, M., Khan, M.A., Maddouri, O., Kinjo, A.R., Queralt-Rosinach, N. and Hoehndorf, R. (2017). Neuro-symbolic representation learning on biological knowledge graphs. *Bioinformatics*, **33**(17), 2723–2730.

[3] Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J. (1990). Basic local alignment search tool. *Journal of molecular biology*, **215**(3), 403–410.

[4] Altschul, S.F., Madden, T.L., Schäffer, A.A., Zhang, J., Zhang, Z., Miller, W. et al (1997). Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, **25**(17), 3389–3402.

[5] Bao, X.R., Zhu, Y.H. and Yu, D.J. (2019). Deeptf: Accurate prediction of transcription factor binding sites by combining multi-scale convolution and long short-term memory neural network. In *International Conference on Intelligent Science and Big Data Engineering*, pages 126–138. Springer.

[6] Begum, F., Ghosh, D., Tseng, G.C. and Feingold, E. (2012). Comprehensive literature review and statistical considerations for gwas meta-analysis. *Nucleic acids research*, **40**(9), 3777–3784.

[7] Belanger, D. and McCallum, A. (2016). Structured prediction energy networks. In *International Conference on Machine Learning*, pages 983–992.

[8] Benner, C., Spencer, C.C., Havulinna, A.S., Salomaa, V., Ripatti, S. and Pirinen, M. (2016). FINEMAP: efficient variable selection using summary data from genome-wide association studies. *Bioinformatics*, page btw018.

[9] Bepler, T. and Berger, B. (2019). Learning protein sequence embeddings using information from structure. *arXiv preprint arXiv:1902.08661*.

[10] Berman, H.M., Battistuz, T., Bhat, T.N., Bluhm, W.F., Bourne, P.E., Burkhardt, K. et al (2002). The protein data bank. *Acta Crystallographica Section D: Biological Crystallography*, **58**(6), 899–907.

[11] Bernstein, B.E., Kamal, M., Lindblad-Toh, K., Bekiranov, S., Bailey, D.K., Huebert, D.J. et al (2005). Genomic maps and comparative analysis of histone modifications in human and mouse. *Cell*, **120**(2), 169–181.

[12] Binns, D., Dimmer, E., Huntley, R., Barrell, D., O'donovan, C. and Apweiler, R. (2009). Quickgo: a web-based tool for gene ontology searching. *Bioinformatics*, **25**(22), 3045–3046.

[13] Bycroft, C., Freeman, C., Petkova, D., Band, G., Elliott, L.T., Sharp, K. et al (2018). The uk biobank resource with deep phenotyping and genomic data. *Nature*, **562**(7726), 203–209.

[14] Cantor, R.M., Lange, K. and Sinsheimer, J.S. (2010). Prioritizing gwas results: a review of statistical methods and recommendations for their application. *The American Journal of Human Genetics*, **86**(1), 6–22.

[15] Cao, J. (2019). *A Case Study for Predicting in-Hospital Mortality by Utilizing the Hyperbolic Embedding of ICD-9 Medical Ontology*. Ph.D. thesis.

[16] Chen, M., Ju, C.J.T., Zhou, G., Chen, X., Zhang, T., Chang, K.W. et al (2019). Multifaceted protein–protein interaction prediction based on siamese residual rcnn. *Bioinformatics*, **35**(14), i305–i314.

[17] Conneau, A., Kiela, D., Schwenk, H., Barrault, L. and Bordes, A. (2017). Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364*.

[18] Consortium, E.P. et al (2012). An integrated encyclopedia of dna elements in the human genome. *Nature*, **489**(7414), 57–74.

[19] Dabney, A., Storey, J.D. and Warnes, G. (2010). qvalue: Q-value estimation for false discovery rate control. *R package version*, **1**(0).

[20] Darnell, G., Duong, D., Han, B. and Eskin, E. (2012). Incorporating prior information into association studies. *Bioinformatics*, **28**(12), i147–i153.

[21] De Castro, E., Sigrist, C.J., Gattiker, A., Bulliard, V., Langendijk-Genevaux, P.S., Gasteiger, E. et al (2006). Scanprosite: detection of prosite signature matches and prorule-associated functional and structural residues in proteins. *Nucleic acids research*, **34**(suppl_2), W362–W365.

[22] De Jong, S., Van Eijk, K.R., Zeegers, D.W., Strengman, E., Janson, E., Veldink, J.H. et al (2012). Expression qtl analysis of top loci from gwas meta-analysis highlights additional schizophrenia candidate genes. *European Journal of Human Genetics*, **20**(9), 1004–1008.

[23] Degner, J.F., Pai, A.A., Pique-Regi, R., Veyrieras, J.B., Gaffney, D.J., Pickrell, J.K. et al (2012). Dnase i sensitivity qtls are a major determinant of human expression variation. *Nature*, **482**(7385), 390–394.

[24] Devlin, B. and Roeder, K. (1999). Genomic control for association studies. *Biometrics*, **55**(4), 997–1004.

[25] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[26] Duong, D., Zou, J., Hormozdiari, F., Sul, J.H., Ernst, J., Han, B. et al (2016). Using genomic annotations increases statistical power to detect eGenes. *Bioinformatics*, **32**(12), i156–i163.

[27] Duong, D., Ahmad, W.U., Eskin, E., Chang, K.W. and Li, J.J. (2018). Word and sentence embedding tools to measure semantic similarity of gene ontology terms by their definitions. *Journal of Computational Biology*, **26**(1), 38–52.

[28] Duong, D., Uppunda, A., Ju, C., Zhang, J., Chen, M., Eskin, E. et al (2019). Evaluating representations for gene ontology terms. *bioRxiv*, page 765644.

[29] Duong, D.B., Gai, L., Uppunda, A., Le, D., Eskin, E., Li, J.J. et al (2020). Annotating gene ontology terms for protein sequences with the transformer model. *bioRxiv*.

[30] Dworkin, J. (2015). Ser/thr phosphorylation as a regulatory mechanism in bacteria. *Current opinion in microbiology*, **24**, 47–52.

[31] Dynomant, E., Darmoni, S.J., Lejeune, É., Kerdelhué, G., Leroy, J.P., Lequertier, V. et al (2019). Doc2vec on the pubmed corpus: study of a new approach to generate related articles. *arXiv preprint arXiv:1911.11698*.

[32] Eskin, E. (2008). Increasing power in association studies by using linkage disequilibrium structure and molecular function as prior information. *Genome Research*, **18**(4), 653–660.

[33] Eskin, E. (2015). Discovering genes involved in disease and the mystery of missing heritability. *Communications of the ACM*, **58**(10), 80–87.

[34] Fei-Fei, L., Fergus, R. and Perona, P. (2006). One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, **28**(4), 594–611.

[35] Flutre, T., Wen, X., Pritchard, J. and Stephens, M. (2013). A statistical framework for joint eQTL analysis in multiple tissues. *PLoS Genetics*, **9**(5), e1003486.

[36] Gamazon, E.R., Segrè, A.V., van de Bunt, M., Wen, X., Xi, H.S., Hormozdiari, F. et al (2018). Using an atlas of gene regulation across 44 human tissues to inform complex disease-and trait-associated variation. *Nature genetics*, **50**(7), 956–967.

[37] Gay, N.R., Gloudemans, M., Antonio, M.L., Balliu, B., Park, Y., Martin, A.R. et al (2019). Impact of admixture and ancestry on eqtl analysis and gwas colocalization in gtex. *bioRxiv*, page 836825.

[38] Gene Ontology Consortium (2019). The gene ontology resource: 20 years and still going strong. *Nucleic acids research*, **47**(D1), D330–D338.

[39] Giral, H., Landmesser, U. and Kratzer, A. (2018). Into the wild: Gwas exploration of non-coding rnas. *Frontiers in cardiovascular medicine*, **5**, 181.

[40] Guillemette, B., Drogaris, P., Lin, H.H.S., Armstrong, H., Hiragami-Hamada, K., Imhof, A. et al (2011). H3 lysine 4 is acetylated at active gene promoters and is regulated by h3 lysine 4 methylation. *PLoS genetics*, **7**(3).

[41] Han, B. and Eskin, E. (2011). Random-effects model aimed at discovering associations in meta-analysis of genome-wide association studies. *The American Journal of Human Genetics*, **88**(5), 586–598.

[42] Han, B. and Eskin, E. (2012). Interpreting meta-analyses of genome-wide association studies. *PLoS genetics*, **8**(3).

[43] Han, B., Kang, H.M. and Eskin, E. (2009a). Rapid and accurate multiple testing correction and power estimation for millions of correlated markers. *PLoS Genetics*, **5**(4), e1000456.

[44] Han, B., Kang, H.M. and Eskin, E. (2009b). Rapid and accurate multiple testing correction and power estimation for millions of correlated markers. *PLoS Genet*, **5**(4), e1000456.

[45] Han, B., Duong, D., Sul, J.H., de Bakker, P.I.W., Eskin, E. and Raychaudhuri, S. (2016). A general framework for meta-analyzing dependent studies with overlapping subjects in association mapping. *Hum. Mol. Genet.*, **25**(9), 1857–1866.

[46] Hernández-Ramírez, L.C., Gam, R., Valdés, N., Lodish, M.B., Pankratz, N., Balsalobre, A. et al (2017). Loss-of-function mutations in the cables1 gene are a novel cause of cushing's disease. *Endocrine-related cancer*, **24**(8), 379–392.

[47] Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, **6**(02), 107–116.

[48] Hormozdiari, F., Kostem, E., Kang, E.Y., Pasaniuc, B. and Eskin, E. (2014). Identifying causal variants at loci with multiple signals of association. *Genetics*, **198**(2), 497–508.

[49] Hormozdiari, F., Kichaev, G., Yang, W.Y., Pasaniuc, B. and Eskin, E. (2015). Identification of causal genes for complex traits. *Bioinformatics*, **31**(12), i206–i213.

[50] Hormozdiari, F., Van De Bunt, M., Segre, A.V., Li, X., Joo, J.W.J., Bilow, M. et al (2016). Colocalization of gwas and eqtl signals detects target genes. *The American Journal of Human Genetics*, **99**(6), 1245–1260.

[51] Huang, J., Osorio, C. and Sy, L.W. (2019). An empirical evaluation of deep learning for icd-9 code assignment using mimic-iii clinical notes. *Computer methods and programs in biomedicine*, **177**, 141–153.

[52] Huang, Z., Tomitaka, A., Raymond, A. and Nair, M. (2017). Current application of crispr/cas9 gene-editing technique to eradication of hiv/aids. *Gene therapy*, **24**(7), 377–384.

[53] Isenberg, I. (1979). Histones. *Annual review of biochemistry*, **48**(1), 159–191.

[54] Jones, C.E., Schwerdt, J., Bretag, T.A., Baumann, U. and Brown, A.L. (2008). Gosling: a rule-based protein annotator using blast and go. *Bioinformatics*, **24**(22), 2628–2629.

[55] Kipf, T.N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

[56] Koch, G., Zemel, R. and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille.

[57] Kotake, Y., Nakagawa, T., Kitagawa, K., Suzuki, S., Liu, N., Kitagawa, M. et al (2011). Long non-coding rna anril is required for the prc2 recruitment to and silencing of p15 ink4b tumor suppressor gene. *Oncogene*, **30**(16), 1956–1962.

[58] Kuhlman, B. and Bradley, P. (2019). Advances in protein structure prediction and design. *Nature Reviews Molecular Cell Biology*, **20**(11), 681–697.

[59] Kulmanov, M. and Hoehndorf, R. (2020). Deepgoplus: improved protein function prediction from sequence. *Bioinformatics*, **36**(2), 422–429.

[60] Kulmanov, M., Khan, M.A. and Hoehndorf, R. (2017). Deepgo: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, **34**(4), 660–668.

[61] Lambert, J.C., Ibrahim-Verbaas, C.A., Harold, D., Naj, A.C., Sims, R., Bellenguez, C. et al (2013). Meta-analysis of 74,046 individuals identifies 11 new susceptibility loci for alzheimer's disease. *Nature genetics*, **45**(12), 1452.

[62] Lau, J.H. and Baldwin, T. (2016). An empirical evaluation of doc2vec with practical insights into document embedding generation. *arXiv preprint arXiv:1607.05368*.

[63] Ledford, H. (2016). Crispr: gene editing is just the beginning. *Nature News*, **531**(7593), 156.

[64] Lencz, T., Lam, M., Consortium, C. et al (2019). Large-scale gwas meta-analysis and multi-trait analysis yields dozens of novel loci and novel genetic correlates for general cognitive ability. *European Neuropsychopharmacology*, **29**, S808–S809.

[65] Liu, L., Tang, L., He, L., Yao, S. and Zhou, W. (2017). Predicting protein function via multi-label supervised topic model on gene ontology. *Biotechnology & Biotechnological Equipment*, **31**(3), 630–638.

[66] Marees, A.T., Gamazon, E.R., Gerring, Z., Vorspan, F., Fingal, J., van den Brink, W. et al (2020). Post-gwas analysis of six substance use traits improves the identification and functional interpretation of genetic risk loci. *Drug and Alcohol Dependence*, **206**, 107703.

[67] Mattick, J.S. (2005). The functional genomics of noncoding rna. *Science*, **309**(5740), 1527–1528.

[68] Mazandu, G.K. and Mulder, N.J. (2013). Information content-based gene ontology semantic similarity approaches: toward a unified framework theory. *BioMed research international*, **2013**.

[69] Mazandu, G.K. and Mulder, N.J. (2014). Information content-based gene ontology functional similarity measures: Which one to use for a given biological data type? *PLoS ONE*, **9**(12), e113859.

[70] Mazandu, G.K., Chimusa, E.R. and Mulder, N.J. (2016). Gene ontology semantic similarity tools: survey on features and challenges for biological knowledge discovery. *Briefings in Bioinformatics*, page bbw067.

[71] Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S. and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

[72] Mullenbach, J., Wiegreffe, S., Duke, J., Sun, J. and Eisenstein, J. (2018). Explainable prediction of medical codes from clinical text. *arXiv preprint arXiv:1802.05695*.

[73] Obozinski, G., Lanckriet, G., Grant, C., Jordan, M.I. and Noble, W.S. (2008). Consistent probabilistic outputs for protein function prediction. *Genome Biology*, **9**(1), S6.

[74] Orphanides, G. and Reinberg, D. (2002). A unified theory of gene expression. *Cell*, **108**(4), 439–451.

[75] Perozzi, B., Al-Rfou, R. and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

[76] Pesaranghader, A., Matwin, S., Sokolova, M. and Beiko, R.G. (2015). simdef: definition-based semantic similarity measure of gene ontology terms for functional similarity analysis of genes. *Bioinformatics*, **32**(9), 1380–1387.

[77] Pesquita, C., Faria, D., Bastos, H., Ferreira, A.E., Falcão, A.O. and Couto, F.M. (2008). Metrics for go based protein semantic similarity: a systematic evaluation. *BMC bioinformatics*, **9**(5), S4.

[78] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. et al (2018). Deep contextualized word representations. In *Proc. of NAACL*.

[79] Plackett, R.L. (1958). Studies in the history of probability and statistics: Vii. the principle of the arithmetic mean. *Biometrika*, **45**(1-2), 130–135.

[80] Profiti, G., Martelli, P.L. and Casadio, R. (2017). The bologna annotation resource (bar 3.0): improving protein functional annotation. *Nucleic acids research*, **45**(W1), W285–W290.

[81] Rios, A. and Kavuluru, R. (2018). Few-shot and zero-shot multi-label learning for structured label spaces. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2018, page 3132. NIH Public Access.

[82] Rives, A., Goyal, S., Meier, J., Guo, D., Ott, M., Zitnick, C.L. et al (2019). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, page 622803.

[83] Schwartzberg, P.L. (1998). The many faces of src: multiple functions of a prototypical tyrosine kinase. *Oncogene*, **17**(11), 1463.

[84] Self, S.G. and Liang, K.Y. (1987). Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions. *Journal of the American Statistical Association*, **82**(398), 605–610.

[85] Sigrist, C.J., De Castro, E., Cerutti, L., Cuche, B.A., Hulo, N., Bridge, A. et al (2012). New and continuing developments at prosite. *Nucleic acids research*, **41**(D1), D344–D347.

[86] Smaili, F.Z., Gao, X. and Hoehndorf, R. (2018). Onto2vec: joint vector-based representation of biological entities and their ontology-based annotations. *Bioinformatics*, **34**(13), i52–i60.

[87] Socher, R., Ganjoo, M., Manning, C.D. and Ng, A. (2013). Zero-shot learning through cross-modal transfer. In *Advances in neural information processing systems*, pages 935–943.

[88] Song, X., Li, L., Srimani, P.K., Yu, P.S. and Wang, J.Z. (2014). Measure the semantic similarity of GO terms using aggregate information content. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, **11**(3), 468–476.

[89] Sterner, D.E. and Berger, S.L. (2000). Acetylation of histones and transcription-related factors. *Microbiol. Mol. Biol. Rev.*, **64**(2), 435–459.

[90] Storey, J.D. and Tibshirani, R. (2003). Statistical significance for genomewide studies. *Proceedings of the National Academy of Sciences*, **100**(16), 9440–9445.

[91] Stranger, B.E., Brigham, L.E., Hasz, R., Hunter, M., Johns, C., Johnson, M. et al (2017). Enhancing gtex by bridging the gaps between genotype, gene expression, and disease. *Nature genetics*, **49**(12), 1664.

[92] Struhl, K. (1989). Helix-turn-helix, zinc-finger, and leucine-zipper motifs for eukaryotic transcriptional regulatory proteins. *Trends in biochemical sciences*, **14**(4), 137–140.

[93] Sul, J.H., Han, B., Ye, C., Choi, T. and Eskin, E. (2013). Effectively identifying eQTLs from multiple tissues by combining mixed model and meta-analytic approaches. *PLoS Genetics*, **9**(6), e1003491.

[94] Sul, J.H., Raj, T., de Jong, S., de Bakker, P.I., Raychaudhuri, S., Ophoff, R.A. et al (2015). Accurate and fast multiple-testing correction in eQTL studies. *The American Journal of Human Genetics*, **96**(6), 857–868.

[95] Szklarczyk, D., Gable, A.L., Lyon, D., Junge, A., Wyder, S., Huerta-Cepas, J. et al (2019). String v11: protein–protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets. *Nucleic acids research*, **47**(D1), D607–D613.

[96] The UniProt Consortium (2018). UniProt: the universal protein knowledgebase. *Nucleic Acids Research*, **46**(5), 2699–2699.

[97] The ENCODE project Consortium (2007). *Nature*, **447**(7146), 799–816.

[98] The GTEx Consortium (2015). The genotype-tissue expression (GTEx) pilot analysis: Multitissue gene regulation in humans. *Science*, **348**(6235), 648–660.

[99] The Roadmap Epigenomics Mapping Consortium (2015). Integrative analysis of 111 reference human epigenomes. *Nature*, (518), 317–330.

[100] Thompson, S.G. and Sharp, S.J. (1997). Explaining heterogeneity in meta-analysis: A comparison of methods. *Statist. Med.*, **18**(3), S82.

[101] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N. et al (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

[102] Wang, S., Cho, H., Zhai, C., Berger, B. and Peng, J. (2015). Exploiting ontology graph for predicting sparsely annotated gene function. *Bioinformatics*, **31**(12), i357–i364.

[103] Wang, Y. and Yao, Q. (2019). Few-shot learning: A survey. *arXiv preprint arXiv:1904.05046*.

[104] Xiao, H. (2018). bert-as-service. github.com/hanxiao/bert-as-service.

[105] Xiong, W., Yu, M., Chang, S., Guo, X. and Wang, W.Y. (2018). One-shot relational learning for knowledge graphs. *CoRR*, **abs/1808.09040**.

[106] Yang, J. and Zhang, Y. (2015). I-tasser server: new development for protein structure and function predictions. *Nucleic acids research*, **43**(W1), W174–W181.

[107] Zhang, C., Zheng, W., Freddolino, P.L. and Zhang, Y. (2018). Metago: Predicting gene ontology of non-homologous proteins through low-resolution protein structure prediction and protein–protein network mapping. *Journal of molecular biology*, **430**(15), 2256–2265.

[108] Zhang, H., Chen, Z., Wang, X., Huang, Z., He, Z. and Chen, Y. (2013). Long non-coding rna: a new player in cancer. *Journal of hematology & oncology*, **6**(1), 37.

[109] Zhong, Y., Perera, M.A. and Gamazon, E.R. (2019). On using local ancestry to characterize the genetic architecture of human traits: Genetic regulation of gene expression in multiethnic or admixed populations. *The American Journal of Human Genetics*, **104**(6), 1097–1115.

[110] Zhou, J. and Troyanskaya, O.G. (2015). Predicting effects of noncoding variants with deep learning–based sequence model. *Nature methods*, **12**(10), 931–934.