Technische Universität München

Functional Gene Embeddings

By
Keming Zhang
Marco Basmaji
Niklas Bühler
Supervisor: Shubhankar Londhe

Computational Modeling for System Genetics, Course Project
Wednesday 7th February, 2024

Abstract

Functional gene embeddings, numerical vectors capturing gene functions, can provide useful representations of genes for downstream analyses. In this project, we extracted functional gene embeddings from transcription data and other genome-wide measurements by using principle components, autoencoders and a Variational Deep Tensor Factorization model. We used these embeddings, as well as embeddings from other publications, as gene features in the prediction of genome-wide association study summary statistics for a diverse set of traits and compared their performances. We found that some embeddings could improve the predictions to an extent and compared the impact of using different datasets and embedding-generating architectures.

1 Introduction

A recent study by Brechtmann et al. [1] has shown that the utilization of functional gene embeddings can help with downstream analysis tasks such as disease-gene list prediction, human phenotype ontology prediction, and genome-wide association study (GWAS) signal prediction. This study inspired us to recreate part of it and at the same time include new data sources and embedding-generating models. We generated functional gene embeddings using three approaches based on four datasets, and evaluated the utility of these embeddings in the prediction of GWAS signals of various traits.

2 Datasets

We considered 19,190 human protein-coding genes in total and used the Ensembl gene IDs as their unique identifiers. We used four different datasets to extract functional gene embeddings: the DepMap dataset [2], the GTEx dataset [3], the RNA isoform dataset [4], and the Tabula Sapiens dataset [5].

2.1 DepMap Dataset

The gene effect data from the Cancer Dependency Map Project Achilles 18Q3 release [2] contains CERES scores for 17,585 genes across 485 different cancer cell lines, which measure the dependency between genes and cell lines. We subsetted the dataset according to the aforementioned protein-coding genes, resulting in 16,445 genes and 485 cell lines in total.

For bias correction and to avoid multicollinearity, we followed the preprocessing procedure introduced by Wainberg et al. [6].

2.2 GTEx Gene Expression Dataset

The Genotype-Tissue Expression (GTEx) [3] project is a continuous project aimed at creating a complete public tool for researching gene expression and control in

different tissues. The dataset contains gene expression data from various human tissues across individuals. The samples were collected from 54 non-diseased tissue sites across nearly 1,000 individuals. Since the dataset consists of a big number of genes (+50k) and samples (+10k) and we ran into memory problems, we have decided to subset over the samples in the dataset and train over only 5k samples.

2.3 RNA Isoform Dataset

The RNA Isoform dataset originates from the study of Uhlén et al [4]. We obtained it from The Human Protein Atlas (https://www.proteinatlas.org/about/publicationdata). It contains RNA levels in 32 tissues, based on RNA-seq. These are measured as fragments per kilobase of transcript per million fragments mapped ("FPKM"). We aggregated the dataset based on unique Ensembl gene IDs and subsetted it according to the aforementioned protein-coding genes, resulting in 18,831 genes and 124 subclasses representing the tissues.

2.4 Tabula Sapiens Dataset

The complete dataset from the Tabula Sapiens atlas [5] contains single cell transcriptomics data of 483,152 cells across 58,870 genes.

In order to reduce computational cost and enrich information, we aggregated the initially sparse dataset by cell ontology classes: the normalized count data of all the cells from the same cell ontology class were summed together and normalized by the total number of cells in that ontology class. Each aggregated cell ontology class is called a pseudobulk. We then subsetted the dataset according to the aforementioned protein-coding genes, resulting in 177 pseudobulks and 19,163 genes.

3 Methods

3.1 Generation of Functional Gene Embeddings

We considered three methods for generating functional gene embeddings from the aforementioned data sources: the principal components, an autoencoder model and a Variational Deep Tensor Factorization model [1].

3.1.1 Principal Components

We considered using the principal components of each dataset's matrix as our baseline embeddings. These principal components could explain $80-95\%$ of the variance in the datasets.

3.1.2 Autoencoder Model

An autoencoder is a type of neural network that is trained to accurately reconstruct its input. The basic structure of an autoencoder is shown in figure 1.

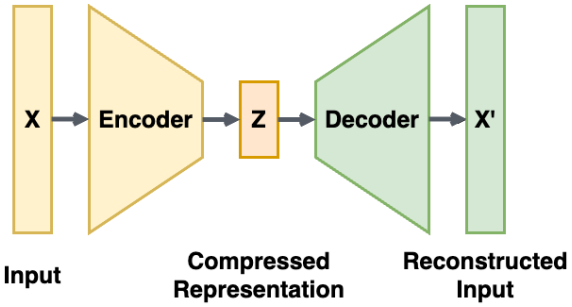

Figure 1: The basic architecture of an autoencoder.

The model consists of two parts, the encoder $\mathbf{f}_e$ and the decoder $\mathbf{f}_d$. Given the expression data $\mathbf{x}_i$ of gene i across all samples in a dataset, the encoder first maps $\mathbf{x}_i$ to its compressed representation $\mathbf{z}_i$ in the latent space, and then the decoder maps $\mathbf{z}_i$ back to reconstruct $\mathbf{x}_i$:

$$\mathbf{z}_i = \mathbf{f}_e(\mathbf{x}_i)$$
$$\mathbf{x}'_i = \mathbf{f}_d(\mathbf{z}_i)$$

Because the latent space of an autoencoder is typically narrower than its input and output space, the model has to condense the information of the input data in the encoder in order to reconstruct it appropriately in the decoder. The condensed latent representations $\{\mathbf{z}_i\}_i$ can thus be extracted and serve as information-dense embeddings of the input data for downstream tasks.

The encoder and decoder of the autoencoder are multilayer perceptrons: Four to six linear layers with batch normalization and ReLU activations in between. The model's parameters are trained and updated by minimizing the mean squared error loss calculated between $\mathbf{x}_i$ and $\mathbf{x}'_i$. This is done by performing gradient descent using the Adam optimizer [7]. The training process is visualized using Weights & Biases [8], and the tuning of hyperparameters (including the number of layers in the autoencoder, learning rate, batch size, number of epochs and latent space dimensions) is done using Optuna [9].

3.1.3 Variational Deep Tensor Factorization Model

Furthermore, we eployed the Variational Deep Tensor Factorization Model (VDTFM) introduced in the study by Brechtmann et al. [1]. The basic structure of a VDTFM is shown in figure 2.

For a single modality, suppose the data matrix $\mathbf{X}$ has values $x_{i,j}$, deriving from gene i and sample j. The model learns embeddings of genes and samples to reconstruct $\mathbf{X}$. For each gene i, assume the embedding of gene i follows a normal distribution with mean μ_i and standard deviation σ_i, and the embedding $\mathbf{a}_i$ of gene i is given by sampling from the normal distribution:

$$\mathbf{a}_i \sim \mathbf{N}(\mu_i, \sigma_i^2)$$

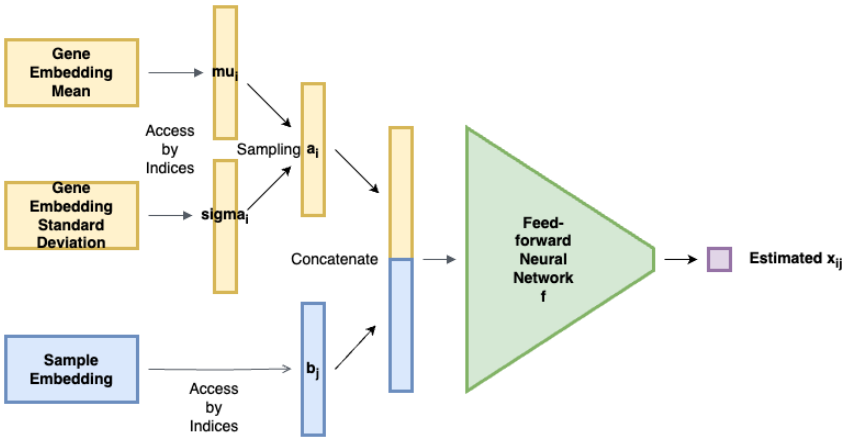

Figure 2: The basic architecture of a VDTFM.

For each sample j, assume the embedding of sample j is non-random and is given by $\mathbf{b}_j$. Then, the reconstructed value $x'_{i,j}$ is obtained by concatenating $\mathbf{a}_i$ and $\mathbf{b}_j$ and applying a feed-forward neural network f to them:

$$x'_{i,j} = f(\mathbf{a}_i, \mathbf{b}_j)$$

The feed-forward neural network f is composed of three to six linear layers and leaky-ReLU activations in between. The model's parameters, the embeddings $\{\mu_i\}_i$, $\{\sigma_i\}_i$, and $\{\mathbf{b}_j\}_j$ are trained and updated by gradient descent using the Adam optimizer [7] minimizing the mean squared error loss calculated between $\mathbf{x}_{ij}$ and $\mathbf{x}'_{ij}$. The training process is visualized using Weights & Biases [8], and the tuning of hyperparameters (including the number of layers in f, learning rate, batch size, number of epochs and embedding dimensions) is done using Optuna [9].

Since the training of VDTFM included training three sets of embeddings and one multilayer perceptron, the training process required much more computational power than the training of the autoencoder or computing the principal components. Therefore, in our project we only trained this model on the Tabula Sapiens dataset and the DepMap dataset.

Also, in the study of Brechtmann et al. [1], a generalized Variational Deep Tensor Factorization model was utilized to generate common variational gene embeddings from multiple different data sources. We did not try this generalized model in our project, but we used the outcome of the study, namely the Omics embeddings generated based on three omics datasets (obtained from https://academic.oup.com/nargab/article/5/4/lqad095/7337624#supplementary-data), as the benchmark embedding for our evaluations and comparisons of embeddings.

3.2 Genome-wide Association Study Signal Predictions

We evaluated the utility of our embeddings by using them as gene features to predict genome-wide association study (GWAS) summary statistics, following the steps

taken in the study of Brechtmann et al. [1].

The Pan UKB study [10] (https://pan.ukbb.broadinstitute.org/) ran GWAS for 7,228 phenotypes using the genotyping data from the UKBiobank [11]. Thirty traits (most of them from the maximally independent 22 trait-set given by [1]) were selected and their GWAS summary statistics were processed through MAGMA [12] using the default parameters and the reference set provided by the PoPS team (https://www.finucanelab.org/data). The selected traits are listed in Table 1. We used the MAGMA z-scores produced by the study of Brechtmann et al. [1], obtained from www.cmm.in.tum.de/public/lecture/ml4rg/gene_embedding_projects/.

Trait	Max. ind. set	Trait	Max. ind. set	Trait	Max. ind. set
Alanine Aminotransferate	True	Albumin	True	Apolipoprotein A	True
Apoliprotein B	True	C-reactive Protein	True	Calcium 30680	True
Calcium 100024	True	Cholesterol	True	Creatinine	True
Direct Bilirubin	False	Glucose	Flase	HDL Cholesterol	True
HDL	False	IGF-1	True	LDL Direct Adjusted by Medication	Flase
LDL Direct	True	LDLC	False	Lipoprotein	False
MCH	False	Mean Corpuscular Haemoglobin	False	Phosphate	True
RBC	False	Red Blood Cell Erythrocyte Count	False	SHBG	True
Testosterone	Flase	Total Bilirubin	True	Total Protein	True
Triglycerides	True	Urate	False	Vitamin D	True

Table 1: The 30 traits that were examined in GWAS Signal Prediction Evaluation. Most of them are included in the maximally independent trait set.

For each trait, let $\mathbf{y}$ be the MAGMA z-scores of genes, which measure the gene-trait association. To avoid the influence of the linkage disequilibrium between nearby genes during the process of model fitting and prediction, we projected $\mathbf{y}$ to $\mathbf{l}$ using a Cholesky decomposition of the covariance matrix that explained the linkage disequilibrium, and use this projected $\mathbf{l}$ to do model fitting and prediction.

We first considered the null model $\mathbf{f}_0$ that predicts the MAGMA z-scores based on six covariates $\mathbf{C}$: gene density, effective gene size and inverse of the mean minor allel count of single-nucleotide polymorphisms in the gene, as well as the logarithmic values of them:

$$\mathbf{l}_0 = \mathbf{f}_0(\mathbf{C})$$

Then, we considered the full model $\mathbf{f}_1$ that predicts the MAGMA z-scores based on these six covariates $\mathbf{C}$ and the gene embeddings $\mathbf{X}$:

$$\mathbf{l}_1 = \mathbf{f}_1(\mathbf{C}, \mathbf{X})$$

We considered two different model architectures for $\mathbf{f}_0$ and $\mathbf{f}_1$: linear models and XGBoost regression models [13].

To evaluate the performance of our embeddings, we used Leave-One-Chromosome-Out cross validation. For each of the 22 autosomes, the genes on it were held out for evaluation and the rest of genes were used to fit the model. After fitting the models

and computing the predictions of the held out genes, the predicted $\mathbf{l}'_0$ and $\mathbf{l}'_1$ were projected back to $\mathbf{y}'_0$ and $\mathbf{y}'_1$. We evaluated the performances of the models by using the square of the Pearson correlation coefficient, and evaluated the performances of the embeddings by using the improvement of the Pearson correlation coefficient of the full model upon the null model:

$$R_0^2 = (\rho_{\mathbf{y}'_0,\mathbf{y}})^2 = (\frac{\mathrm{Cov}(\mathbf{y}'_0, \mathbf{y})}{\sigma_{\mathbf{y}'_0}\sigma_{\mathbf{y}}})^2,$$

$$R_1^2 = (\rho_{\mathbf{y}'_1,\mathbf{y}})^2 = (\frac{\mathrm{Cov}(\mathbf{y}'_1, \mathbf{y})}{\sigma_{\mathbf{y}'_1}\sigma_{\mathbf{y}}})^2,$$

$$\Delta R^2 = R_1^2 - R_0^2.$$

Genes that were not represented in a specific dataset, i.e. for which we could not generate embeddings from that dataset, were imputed in the evaluation step, by setting their GWAS signal prediction value to the mean target value. This was done in order to balance out the unfair advantage of small datasets over bigger datasets in the evaluation step.

4 Results

4.1 Generation of Functional Gene Embeddings

We created multiple functional gene embeddings from four different data sources, as shown in Table 2. These different embeddings were then utilized for GWAS signal prediction and compared against each other, as well as against the baseline null model, predicting only on the basis of covariates.

Embedding Source	Embedding-Generating Model or Method	Dimension
DepMap Dataset	Principal Components	81
DepMap Dataset	Autoencoder	128
DepMap Dataset	VDTFM	128
GTEx Dataset	Principal Components	64
GTEx Dataset	Autoencoder	128
GTEx Dataset	Autoencoder	256
RNA Isoform Dataset	Principal Components	4
RNA Isoform Dataset	Autoencoder	32
Tabula Sapiens Dataset	Principal Components	5
Tabula Sapiens Dataset	Autoencoder	64
Tabula Sapiens Dataset	VDTFM	256

Table 2: Functional gene embeddings of different dimensions were created from multiple data sources.

4.2 GWAS Signal Prediction Results

The XGBoost regression model produced better general results (measured in achieved R^2-values) than the linear regression model and also lent itself to more interesting analyses of the obtained results. Therefore, and due to space constraints, we only analyse the XGBoost predictions in this report.

4.3 GWAS Signal Prediction Using XGBoost Regression

For a direct comparison of the full models against their respective null models based only on covariates, we plotted the GWAS signal predictions of the full models against the predictions of the null models for all traits. This plot is shown in figure 3 for a few selected embeddings (one per dataset). A full plot containing all embeddings can be found in figure 7 in the supplement.

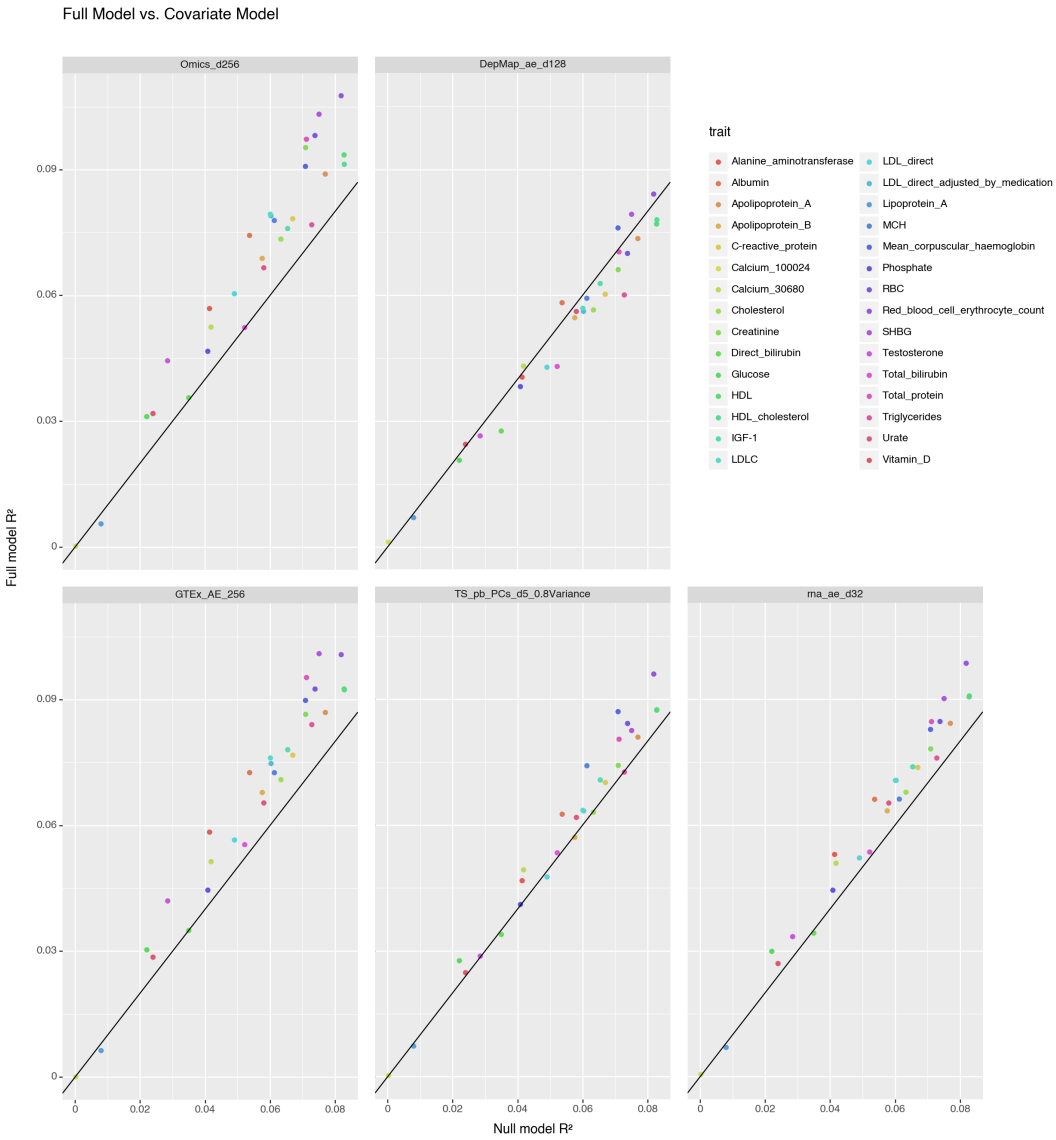

Figure 3: XGBoost Regression: Comparing predictions of the full evaluation model (using covariates and embeddings) against the null evaluation model (using only covariates) to determine the improvement on GWAS signal prediction per embedding across all different traits. All embeddings are plotted in figure 7.

To further examine the influence of each embedding on the GWAS signal prediction of different traits, we created a heatmap showing the improvement of the achieved

R^2-value of the full models over their respective null models (ΔR^2). This heatmap can be seen in figure 4. It also serves as a way to compare different embeddings (i.e. embedding-generating architectures and data sources) against each other.

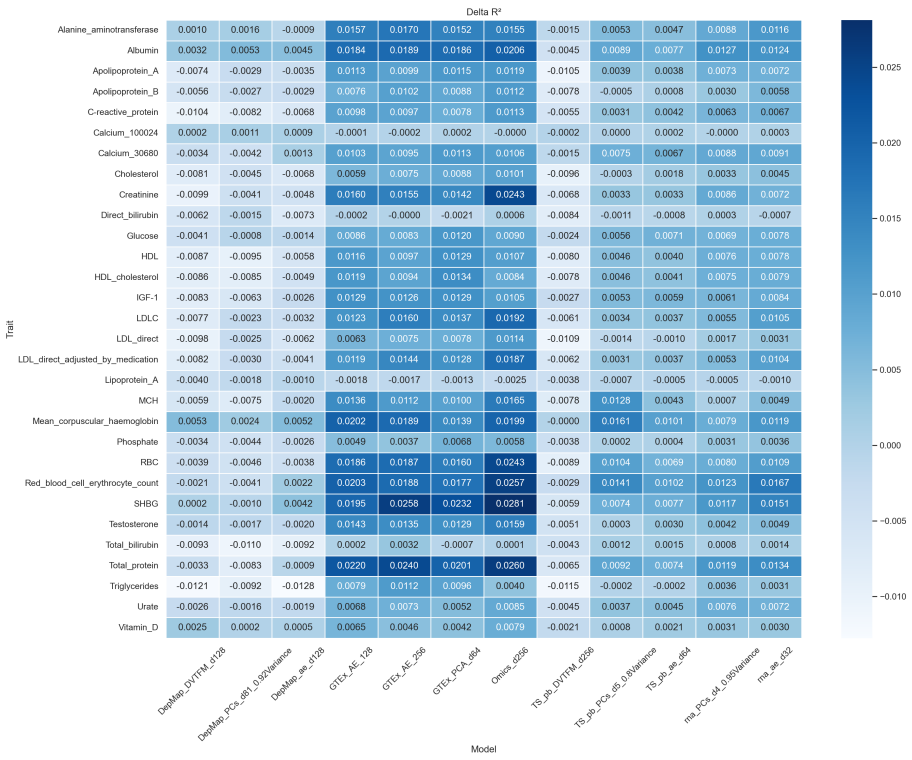

Figure 4: XGBoost Regression: Heatmap, showing the improvement on the achieved R^2-values (i.e. ΔR^2) for each embedding on each trait.

Another way to compare different embeddings against each other, is to compare the distribution of their improvements on the R^2-values per trait. This is shown in the boxplots in figure 5, which already contains the ΔR^2 of the full models against the respective null models.

Furthermore, we tested for a significant difference between the distributions of achieved R^2-values across all traits (full models versus null models) using the paired Wilcoxon signed-rank test ($\alpha = 0.05$). The results showed that all differences were statistically significant. In order to correct for multiple testing, we've also applied Bonferroni correction by dividing the original α-value by the amount of tests performed, leading to a new $\alpha^* \approx 0.0042$. This didn't change the significance of the differences.

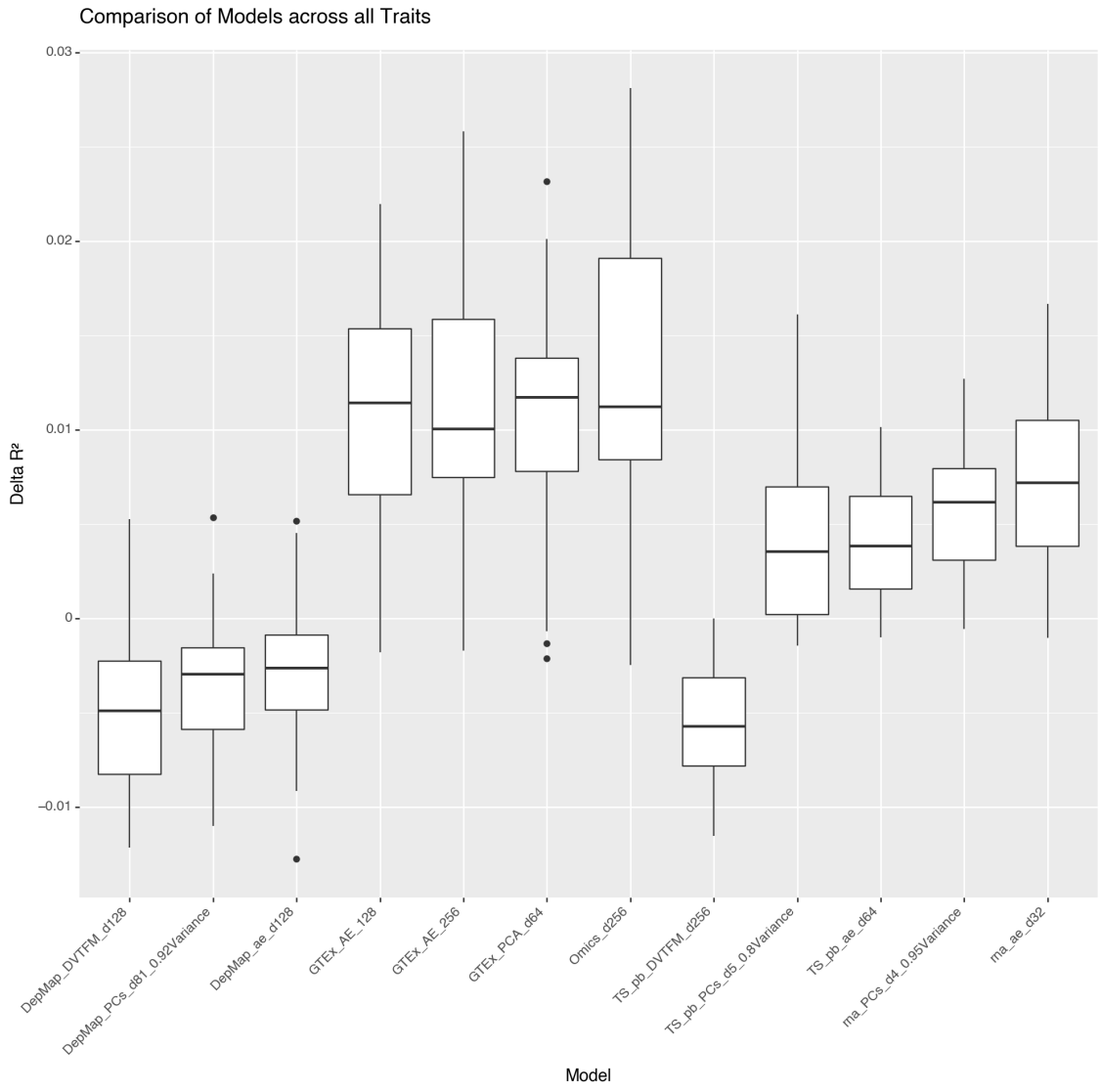

Figure 5: XGBoost Regression: comparing the achieved improvement on R^2-values (i.e. ΔR^2) of all embeddings across all traits.

4.4 GWAS Signal Prediction Using Linear Regression

For the GWAS signal prediction using linear regression, we did the same analyses as for the XGBoost regression. Due to space constraints, we only included a figure showing the distribution of improvements on the R^2-values of all embeddings across all traits in this report, as shown in figure 6.

5 Discussion

5.1 Findings based on XGBoost Regression

The scatter plots in figures 3 and 7 show that the R^2-value could in general be most improved for traits which already exhibited a high R^2-value in the null model. This

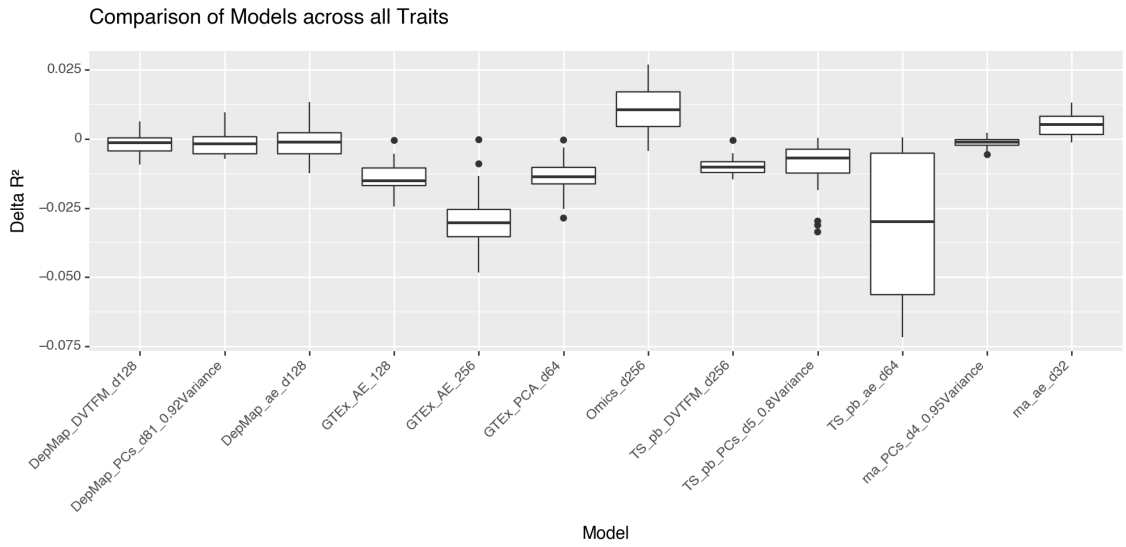

Figure 6: Linear Regression: Comparing the achieved improvement on R^2-values (i.e. ΔR^2) of all embeddings across all traits.

might be linked to the fact that some traits are influenced more heavily by genetic factors than others.

The heatmap in figure 4 highlights the improvement achieved through the different embeddings for each trait. For example, it can be seen that in general, the Omics and GTEx embeddings could increase R^2-values the most. On the other hand, the DepMap embeddings did not help to improve R^2 on most traits. Furthermore, the predictions of some traits were not improved by any embedding, like for example Lipoprotein A. Others however, like the predictions of SHBG, were greatly improved by the GTEx and Omics embeddings. It can also be seen that the VDTF model didn't help with improving the R^2-values for most traits across datasets.

The boxplots in figure 5 again highlight the comparatively strong improvements achieved by the Omics and GTEx embeddings. While the RNA and Tabula Sapiens embeddings (except for the TS VDTF model) also increased the achieved R^2-values for most traits, all embeddings extracted from the DepMap dataset and the VDTFM embeddings of the Tabula Sapiens dataset didn't improve R^2 on most traits.

Furthermore, the results of the paired Wilcoxon signed-rank test show that all distributions of R^2-values differ significantly from the distribution generated by the null evaluation model using only the covariates. In order to control the family-wise error rate, we applied Bonferroni correction, but all differences remained significant.

5.2 Comparison between XGBoost and Linear Regressions

Figures 5 and 6 show different performances of embeddings between the linear regression and XGBoost regression evaluation. While the GTEx embeddings help to

improve the R^2-values in XGBoost regression, they do not help much in linear regression, the R^2-values even decrease when including the embeddings. This is also true for the Tabula Sapiens embeddings. The DepMap embeddings show better performance in the linear regression evaluation than with XGBoost regression, but still help little with improvements of the R^2-values.

It's worth to notice that the Omics embedding is robust against the regression model selection, showing strong performances in both linear and XGBoost regression. Also, the RNA-autoencoder-embeddings persist in showing relatively good performances in both regression setups, being able to improve the R^2-values on most traits.

To explain the different performances of embeddings between regression models, we hypothesize that different regression models would prefer different input structures and have different error (noise) handling methods. For example, the linear regression model assumes a linear relationship between the response variable (projected z-score) and the input (embeddings), with the error term following a normal distribution. The GTEx embeddings might not satisfy these model assumptions and therefore are not suitable to fit a linear regression model.

5.3 Further Research

We created various functional gene embeddings from multiple data sources, and showed that they could be used to improve GWAS signal prediction to some extent. However, we encountered several challenges, which offer directions for further research.

5.3.1 Comparable Dimensions of Embeddings

The dimensions of the embeddings were determined through hyperparameter tuning, which means that for different data sources and embedding-generating methods, the embeddings' dimensions were different, complicating direct comparability.

For the autoencoder model, data sources with larger dimensions (e.g. the GTEx dataset) had embedding dimensions up to 256, while data sources with smaller dimensions (e.g. the RNA Isoform dataset) had embedding dimensions of up to 32. In the model training process, these embedding dimensions could both achieve good reconstruction accuracy on test data (coefficient of determination at around 0.9).

However, in the GWAS signal prediction task, direct comparison between embeddings of different dimensions was complicated, because on the one hand, embeddings of larger dimension might include more information on genes, but on the other hand, embeddings of smaller dimension might contain more enriched information and include less noise.

5.3.2 Embeddings generated from VDTFM

Figures 5 and 6 show that the embeddings generated from the VDTFM do not lead to good performances in GWAS signal prediction. This might be because of a mistake in specifying the embedding-dimension-ranges when training the model. When we initially trained the model, we did not realize that we should choose the embedding-dimension according to the datasets' dimensions. We only considered embedding-dimensions 256 and 128, and did hyperparameter tuning. Later we found that for datasets (such as the Tabula Sapiens dataset) with a relatively small dimension, it would be better to consider smaller embedding dimensions, such as 32 or 64.

Because of time limitations of our project, we didn't have the chance to try out a broader range of embedding-dimensions for the VDTF model. We believe it would be a promising direction for further research to explore a broader range of embedding-dimensions for the VDTF model and investigate how the embedding-dimension influences the training of VDTFM, as well as performance in further downstream tasks.

5.3.3 Combining Embeddings from Different Data Sources

One of our further experiments was concatenating promising embeddings in order to obtain gene embeddings stemming from different data sources.

One such combined embedding consists of three extracted latent spaces from a GTEx autoencoder, Tabula Sapiens principal components and RNA Isoform autoencoder, each with an embedding dimension of 128, 32 and 5, respectively.

The choice of this combination was based on the performance of each embedding when evaluated on its own. For almost all evaluated traits, the R^2-values increased slightly.

In further studies, more combinations could be tested. Another example of an interesting, more comparable, result could be to sum up the dimension of the concatenated embeddings to exactly 256, making it directly comparable to the Omics embeddings.

Supplement

Figure 7 contains the direct comparison of the full models against the null models for all generated embeddings.

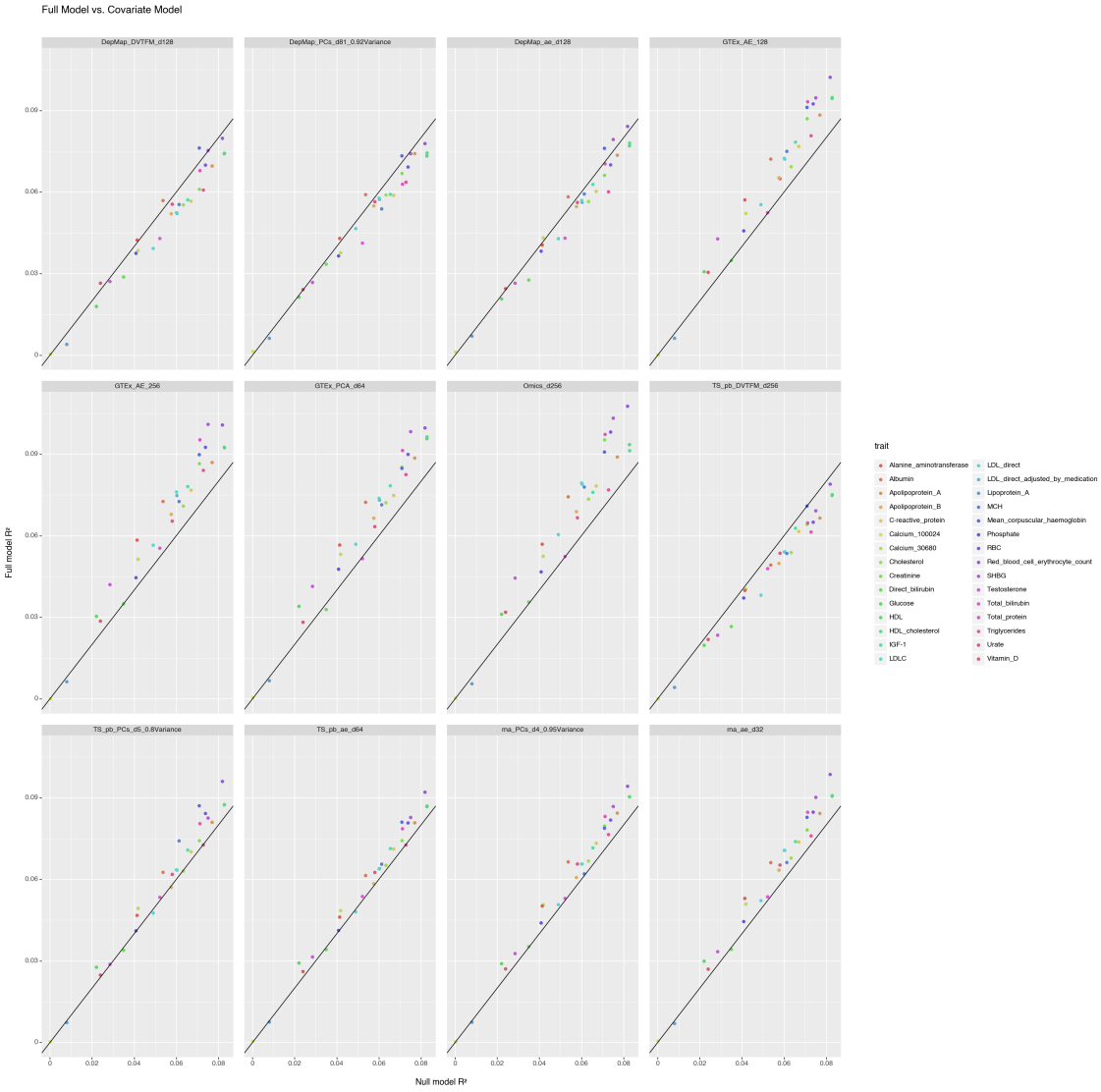

Figure 7: XGBoost Regression: Comparing predictions of the full evaluation model (using covariates and embeddings) against the null evaluation model (using only covariates) to determine the improvement on GWAS signal prediction per embedding across all different traits.

References

[1] F. Brechtmann, T. Bechtler, S. Londhe, C. Mertes, and J. Gagneur, "Evaluation of input data modality choices on functional gene embeddings," *NAR Genomics and Bioinformatics*, vol. 5, no. 4, p. lqad095, 2023.

[2] A. Tsherniak, F. Vazquez, P. G. Montgomery, B. A. Weir, G. Kryukov, G. S. Cowley, S. Gill, W. F. Harrington, S. Pantel, J. M. Krill-Burger, *et al.*, "Defining a cancer dependency map," *Cell*, vol. 170, no. 3, pp. 564–576, 2017.

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

[4] M. Uhlén, L. Fagerberg, B. M. Hallström, C. Lindskog, P. Oksvold, A. Mardinoglu, Å. Sivertsson, C. Kampf, E. Sjöstedt, A. Asplund, *et al.*, "Tissue-based map of the human proteome," *Science*, vol. 347, no. 6220, p. 1260419, 2015.

[5] T. T. S. Consortium*, R. C. Jones, J. Karkanias, M. A. Krasnow, A. O. Pisco, S. R. Quake, J. Salzman, N. Yosef, B. Bulthaup, P. Brown, *et al.*, "The tabula sapiens: A multiple-organ, single-cell transcriptomic atlas of humans," *Science*, vol. 376, no. 6594, p. eabl4896, 2022.

[6] M. Wainberg, R. A. Kamber, A. Balsubramani, R. M. Meyers, N. Sinnott-Armstrong, D. Hornburg, L. Jiang, J. Chan, R. Jian, M. Gu, *et al.*, "A genome-wide atlas of co-essential modules assigns function to uncharacterized genes," *Nature genetics*, vol. 53, no. 5, pp. 638–649, 2021.

[7] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[8] L. Biewald, "Experiment tracking with weights and biases," 2020. Software available from wandb.com.

[9] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[10] Pan-UKB team, "Pan-uk biobank," 2020.

[11] C. Bycroft, C. Freeman, D. Petkova, G. Band, L. T. Elliott, K. Sharp, A. Motyer, D. Vukcevic, O. Delaneau, J. O'Connell, *et al.*, "The uk biobank resource with deep phenotyping and genomic data," *Nature*, vol. 562, no. 7726, pp. 203–209, 2018.

[12] C. A. de Leeuw, J. M. Mooij, T. Heskes, and D. Posthuma, "Magma: generalized gene-set analysis of gwas data," *PLoS computational biology*, vol. 11, no. 4, p. e1004219, 2015.

[13] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, ACM, Aug. 2016.