DeepSeries: Genomic Deep Learning Model for Allelic Series

Iustin Curcean, Stefan Schärdinger, Sonia Castro,
Henrique Colaço e Caldas

TU München.
Helmholtz München.

Abstract

Allelic series are collections of different alleles at a given locus that lead to phenotypic trait variations. Deep learning models based on multiple genomic annotations, such as minor allele frequency, represent a powerful tool for genetics, enhancing the statistical significance between allelic series and phenotypic impacts. Understanding this relationship is of hefty relevance, empowering drug and treatment research, in particular when rare variants are considered. Nevertheless, there are not many instruments and techniques to predict allelic series effects. Hence, DeepSeries, as a multiple genomic annotations deep learning model for allelic series, which aims to overcome this gap and boost possible further research in this field. Our model outperforms current standards in learning the burden score of multiple genomic annotations

Keywords: Deep Learning, Allelic Series, Multiple Genomic Annotations, Rare Variants

1 Introduction

Alleles are defined as variations that can occur through the replacement of a single nucleotide, insertions, or deletions at a specific gene locus. An Allelic series is defined as a collection of alleles in a gene or pathway that lead to a gradation of possible phenotypes (one can also see it as a myriad of variants at a given gene and the associated phenotypic traits variations) [1]. These series are important not only for understanding the relationship between genetic mutations and their effects but also reveal a dose-response relationship between the functionality of a gene and the severity of the resulting phenotype. This implies that pharmacological interventions targeting the gene in question could potentially ameliorate the associated phenotype. This concept is fundamental when studying rare variants, hereafter defined as variants with a Minor Allele Frequency (MAF), lower than 0.01.

Traditional testing methods like Genome Wide Association Studies (GWAS), which are grounded on whole-genome sequencing studies, lack statistical significance when considering assigning a specific phenotypic impact to a rare variant. Even in large cohorts, the presence of these variants is still negligible [1][2]. Hence, locus-specific tests for rare variants such as Coding-Variant Allelic-Series Test (COAST)[1] have been developed to identify genes containing allelic-series, and therefore improve the relationship between a given locus and phenotypic traits.

The identification of genes demonstrating allelic series has been a research topic addressed recently in order to enhance the understanding and scientific knowledge regarding rare variants. Techniques such as the usage of multiple genomic annotations for allelic series, as in [1], have paved the way for testing for allelic series, making use not only of the genomic functional annotations but also of other properties of the concerned single nucleotide polymorphism (SNP), such as the minor allele frequency. Furthermore, deep learning models have been proposed to address the usage of variant annotation on burden score learning [3].

Our research is based on previous work compiling testing methods for allelic series, such as COAST[1] and kernel-based tests SKAT and SKAT-O [4], but also recent deep learning models such as DeepRVAT [3], where different assumptions and methods are proposed to test and identify genes for the presence of allelic series.

Problem definition

Despite the advances in testing methodologies for allelic series and machine learning, for example, with DeepRVAT proposed in [3], there is still a gap to fill in developing models that can map the effect of specific genes to phenotype traits, revealing allelic series. Hence, Deep Series, a genomic deep learning model where allelic series effects are built upon multiple genomic annotations, was developed aiming to achieve better predictions of the expected gene deleteriousness score.

Given this, our aim is to design a model structure which, given a database of individuals for whom we have the SNPs for specific genes, respective annotations, and further context information as their minor allele frequencies, can be trained to map the allelic series to a phenotype score and learn the underlying gene burden score. From such a trained model - particularly from its trained model weights - one can then conclude possible causes for the proteins of interest losing their function.

2 Methods

We first explain the methods for building our model, followed by how we generate training data from our available biological data.

2.1 Model Design of DeepSeries

We start by laying out our assumptions of the model, and then we outline the overall structure of our proposed model. In the paragraphs following that, we explain each of the model's components in more detail.

Model assumptions

To achieve statistically significant results, the following assumptions (concerning both biology and modeling) were considered:

A1 : Different variants on the same functional annotation classification affect the phenotype in the same direction

A2 : Phenotypic traits can be represented by a numeric score

A3 : Less frequent variants have a greater deleterious effect.

Regarding the third assumption we note that, logically, we can make statistical statements about variants which either are common and hence we have a lot of data about them, or about variants which have a large effect. Due to selection, there are no common effects which have a great impact, which implies the fact that the rare variants we know of have a large effect.

Model overview

Our proposed model shown in Figure 1 predicts from a given genotype (denoted as X) a phenotype score. We deterministically create a phenotype score from our genotype dataset and compare that to the phenotype prediction of the model. From this difference, we calculate

a loss and gradient flow, updating the learnable model parameters. From the learned model parameters, we aim to gain more insight into how biology maps a phenotype to a genotype. The model is trained with a large amount of genotype-phenotype pairs. The intuition is that although some data points have low statistical power due to rarety, when using many of them, we can leverage great statistical power over a few model parameters. In our model, we use in total five learnable parameters: three weight parameters in w and two contribution factors α, β for a beta function.

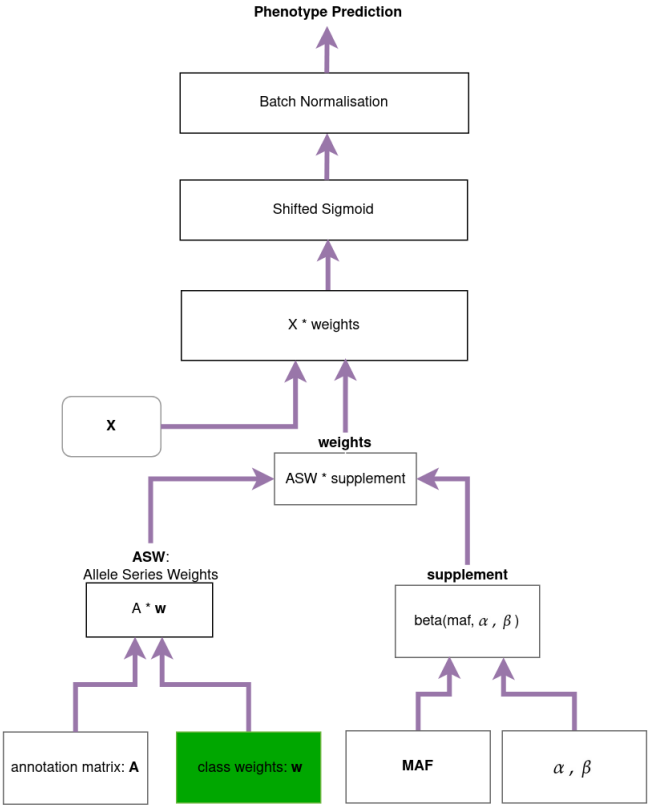

Fig. 1: Overview of DeepSeries

Creating a phenotype score with allelic series

The core of the *weights* path in 1 is the inclusion of the multiple variant annotations: We treat each variant as a member of one of the annotation groups. For our use case, we group the variants into the three groups called synonymous, missense and protein loss of function, hereafter referred to as pLoF. For synonymous variants, we know that the corresponding mutation likely has an insignificant impact on protein functionality, if any. The pLoF variants have a detrimental effect on the protein function and of missense variants, the effect is mainly unknown.

As further explained in Section 2.2, the simulated training data uses hardcoded effect sizes for these three classes. We aim for the model to learn these effect sizes through training. Referring to our model shown in Figure 1, the effect sizes are located in the learnable parameters w. In its most basic form, DeepSeries is tasked to learn w when given as training data for each individual the genetic makeup (so $X = (SNP_1, SNP_2, ...SNP_{n-SNPs})$ with $SNP_j = 1$ if that individual has the jth SNP) and the phenotype score (which in the following we call y). This phenotype score should be a measure of what a SNP (or a combination of SNPs) mean biologically.

Our precursor model implements a naive approach for this phenotype score: The model takes all the SNPs of the individual, obtains the learned weights of the variant class of these SNPs and then sums up these weights:

$$Y \leftarrow X \ * A \cdot w \tag{1}$$

where X is a one-hot-encoded input of an individual, and A is the $(N_v \times 3)$ one-hot encoding for the class of each variant.

Note that X contains the genetics of a large set of individuals and Y the corresponding phenotypic scores. With this setup, we train w with a large set of individuals, the idea is that by training with a lot of data points (X) and very few parameters (w), we get good statistical power, even if each of our datapoints has low statistical power (e.g., through noise). This precursor predictor $Y \leftarrow X \ * A \cdot w$ serves as the core of our proposed model as is henceforth referred to as ASW (allele series weights).

In particular, we aim for w to learn the ratios of the actual class-specific effect sizes, not necessarily the actual values. This means that the model learns the relative importance of one annotation group compared to another.

Incorporating the MAF with a learnable beta function

The MAF (minor allele frequency) of each variant is included in the model as an input since it holds significant information regarding the deleterious score of variants, which follows from assumption A3. This is our motivation to include it in the precursor model of the $Y \leftarrow X \ * \ A \cdot w$ explained in the earlier paragraph 2.1.

In our implementation, MAF_j corresponds to the minor allele frequency of variant j. The details of our dataset and preprocessing can be checked in the section 2.2

We incorporate MAF into the $weights$ to be used together with the allelic series weights ASW. We expect rare variants to have a larger effect on the phenotype, which, as a first approach, we found could be incorporated into our model with

$$weights_j = ASW_j * \frac{1}{MAF_j(1 - MAF_j)} \tag{2}$$

This is based on the SKAT model as introduced in Equation 4 by McCaw et al. [1]. The same authors offer a generalization of this term using the beta distribution [4]. It can be shown that $1/MAF(1 - MAF)$ is the case where $\alpha = \beta = 0.5$.

For our use case, we thus propose

$$weights_j = ASW_j * \sqrt{Beta(MAF_j, \alpha, \beta)} \tag{3}$$

using the square root as used in SKAT [5]. This helped prevent excessively high values that might dominate the score prediction and could lead to the saturation of the sigmoid function (see 2.1), causing everything to be mapped to a singular output.

In order to enable the adaptability of the Minor Allele Frequency (MAF) contribution within the $weights$, we make the parameters α and β learnable.

Extracting burden score with non-linearity

We aim to predict a burden genetic score within the range [0,1]. To achieve this, we incorporate a non-linear layer in our model, employing a shifted sigmoid function:

$$sig(x - 6) = \frac{1}{1 + e^{-(x-6)}} \tag{4}$$

The x-axis shift applied to the input ensures that entries with a score of 0 (indicating an individual with no deleterious variants) are accurately mapped to 0. This not only facilitates the correct mapping of scores within the desired range but also enables our model to learn non-linearities. This approach allows the model to capture more complex patterns in the data, enhancing its predictive capabilities.

Output standardization

To ensure better stability [6] and allow further scaling of the output of the non-linear layer from our model, a last batch normalization layer was added. With this, the standardization is performed over each mini-batch according to

$$y = \frac{x - \mathbb{E}[x]}{Var[x] + \epsilon} * \gamma + \beta \tag{5}$$

where ϵ is a fixed parameter ensuring numerical stability to the computations and γ and β are additional learnable parameters.

2.2 Dataset Generation

Data Preprocessing

The used data set is based on the sequence alignment of 5 different genes (ALB, APOB, PCSK9, SHBG, and SLC22A2) from 343486 individuals, from which the variants annotations were taken from [7] and the rest of the details of the variants from the UK Biobank [8]. Nevertheless, due to the current data privacy framework, handling with direct real individual data sets was not possible, i.e., it was not possible to evaluate the direct genotype and phenotype for real individuals. Instead, the initial data set was a set of SNPs and additional details for the sample.

For each SNP contained in each gene, the following details were disclosed: chromosome, position index, specific nucleotide polymorphism, genomic functional annotation, allele count and sample size. As for the genomic functional annotation, we have three categories for each SNP as already explained in the paragraph 2.1: These include synonymous variants, indicating a likely insignificant impact on protein functionality; missense variants, for which there is no definitive statement, and protein loss-of-function (pLof) variants, known to have a detrimental effect on protein function.

Also, from our available data set, we have access to the allele counts of the variants from which we form

$$MAF = AC_j/N \tag{6}$$

with N individuals contained in our dataset in total and AC_j the count of the variant j.

Simulating genotype from allelic counts

From the allele counts of each variants we generate a genotype library X of structure $\{0,1\}^{(N\times N_V)}$, where N is the number of individuals and N_V is the number of different variants present in the dataset. In our simulation context the individual $i \in [0, N-1]$ has the variant $j \in [0, N_V - 1]$ if $X_{i,j} = 1$. This means that the entries in $X_{i,:}$ are a one-hot encoded genotype of individual i.

For constructing this matrix, we require the condition that the simulated variant frequencies align with those observed in the real world: To do this, firstly the matrix is initially initialized to zero. Subsequently, for each column j, 1s are introduced in an amount proportionally to the occurrences of the corresponding variant in the real world; i.e. $AC_j = \sum_{i\in[0,N-1]} X_{i,j}$. Finally, we permutate each column j randomly, so the array $X_{:,j}$ is permutated for every variant j which means that all the occurrences of variant j are randomly distributed among the individuals. With the resulting $X_{i,j}$ we have preserved the authentic frequencies of all variants of our used dataset.

Simulating a phenotypic score from a genotype library

In order to generate a phenotype library from the previously generated genotype library, we do:

$$score_i = \sum_{j=0}^{N_V-1} (X_{i,j} \cdot w_j) \tag{7}$$

where i is the individual index $[0, N-1]$, and j is the rare variant index $[0, N_V-1]$. The assigned weight for each variant is represented by w_j , and this can vary depending on the assumptions made.

In our case, we split our variants to their categorical functional annotation, following a similar procedure as in COAST [1]: We define the three variant classes' specific score values, with synonymous variants corresponding to 0, missense variants to 1, and pLof variants to 2. Thus, the weights come from:

$$w_j = A \cdot AS \tag{8}$$

where A is the $(N_v \times 3)$-one-hot encoding for the class of each variant, and AS is the (3×1) tensor containing the allele class-specific scores. We choose the class-specific scores $[0, 1, 2]$. With this, we obtain for each variant j the corresponding allelic weight $W_j \in \{0, 1, 2\}$.

As explained in Paragraph 2.1 the model is expanded by incorporating the MAF. We also cover this case in the phenotype generation as done in [5]:

$$w_j = (A \cdot AS) \odot \sqrt{Beta\ pdf(MAF_j, \alpha, \beta)} \tag{9}$$

This assumption is based on the premise that less frequent variants are more deleterious due to natural selection. We chose the commonly used values $\alpha = 1$ and $\beta = 25$ (see [5]) to ensure that only the rarer variants receive higher values.

As we are interested in developing a burden genetic score, we subsequently pass the calculated $score_i$ through the shifted sigmoid (check eq. (4)) previously mentioned in Paragraph 2.1 to obtain a score within the range [0,1]. Then, we obtain $score_i = sig(score_i - 6)$. The x-axis shift to the input is employed to guarantee that entries with a score of 0 (indicating an individual with no deleterious variants) are mapped to 0.

Given that in real-world scenarios the presence of a rare variant accounts for only a small fraction of the phenotype variance, the final steps in generating our phenotype are:

$$Y_g = \frac{y_g - \bar{y_g}}{\sigma_{y_g}} * \sqrt{genetic\ share} \tag{10}$$

where the genetic component of the phenotype is standardized.
To consider a realistic data set, gaussian noise is then also added to the phenotype:

$$Y = Y_g + Y_{noise} \tag{11}$$

with:

$$Y_n = y_{noise} * \sqrt{(1 - genetic\ share)}\ ,\ y_{noise} \sim N(0, 1) \tag{12}$$

3 Experimentation and Results

In order to better analyze the performance of the proposed DeepSeries model, we compare the results and performance with other standard baseline models.

3.1 Baseline Models

For the baseline models, we employ commonly used allelic series models, specifically the max and sum variants. These are explained in more detail in [1]. In both models, a weight w_l is defined for each variant category l, in order to specify the pattern of effect sizes sought. In our case, we have three categories (synonymous, missense, and pLof) as in DeepSeries. However, we have implemented the two most widely used versions that only consider pLof or pLof and missense variant annotations.

Allelic-series sum model

As for the sum variant of the model, we have the following:

$$score_i = \alpha + \left(\sum_{l=1}^{L} w_l \cdot N_{i,l} \right) \cdot \beta \tag{13}$$

where i is the individual, l is the variant category and $N_{i,l}$ is the category l allele counts of the individual i.

In our implementation, α and β are learned as parameters of a linear layer. The sum model will have better power the closer the prespecified allelic-series weights (w_l) are proportional to the true pattern of effect sizes.

Allelic-series max model

In the max variant of the model, we have:

$$score_i = \alpha + \max_{l \in \{0,L\}} (w_l N_{i,l}) \cdot \beta \tag{14}$$

where i is the individual, l is the variant category and $N_{i,l}$ is the category l allele counts of the individual i.

In our implementation, α and β are learned as parameters of a linear layer. This is motivated by a genetic architecture in which the expected phenotype is determined by the most deleterious variant present.

3.2 DeepSeries Model Results

We fit our proposed model to the gene *Albumin* (ALB). To do this, we simulate a dataset consisting of 343,486 different individuals. From the original biologically accurate dataset, we can sample from a pool of 352 different alleles that are observed in the original dataset in a total of 11,613 individuals. For every displayed plot, a new genotype was generated by randomly adding an SNP from the 352 available ones to the individuals until a total of 11613 was reached.

As for the model comparison, the R2 metric was taken into account, as it not only gives easy interpretability across the models (i.e., the score represents how much of the proportion of the phenotypic score is predicted by the genetic setup) but also robustness, not being affected by the range of the data set.

From Figure 2 we see that for different variances on the phenotype explained by genetics, at levels of 5%, 10%, 15%, and 20% DeepSeries shows a higher R^2 score, which proves good results for the developed model.

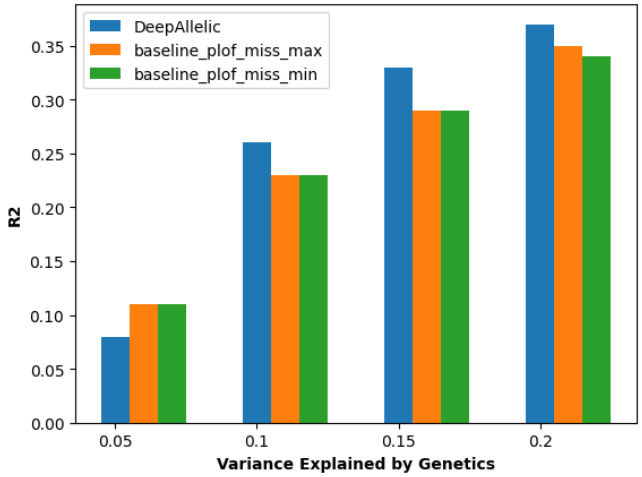

Fig. 2: R2 score comparison between our DeepAllelic model and two baseline models for different levels of variance explained by genetics.

We analyse how the weights of the annotations change over the process of training: From Figure 3, one can see that the trained weights converge after around 350 training steps to the expected ratio of [0, 1, 2]. The predicted values are not exact, because of the integration of the minor allele frequency and non-linearity. To ensure the reliability of our results, we conduct an investigation by training our DeepSeries model a total of 1131 separate times. We expect that the third weight (pLoF) is double the second weight (Missense).

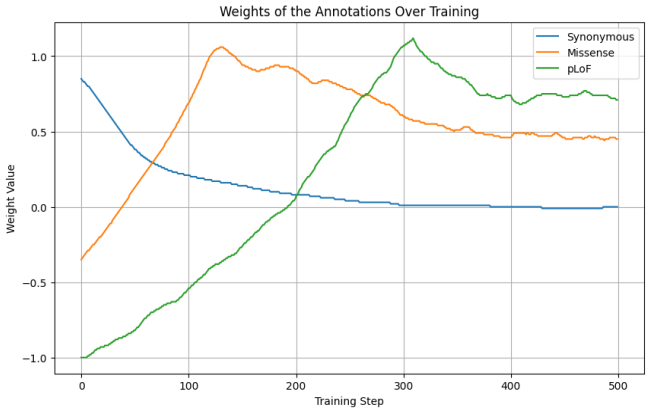

Fig. 3: Values of annotation weights w as training progresses for 500 training steps.

As seen in 4 the distribution's bulk is around that value, which shows that the expected ratio is achieved and is consistent between runs.

We furthermore investigate the exact contribution of the MAF in our model by calculating the burden score of all the variants grouped by annotation class. In Figure 5 we see that the MAF effect, based on the learned weights and parameters, is not acquiring a big impact on increasing the deleteriouness weight for the regarding SNPs.
Our final hyperparameters are listed in Table 1.

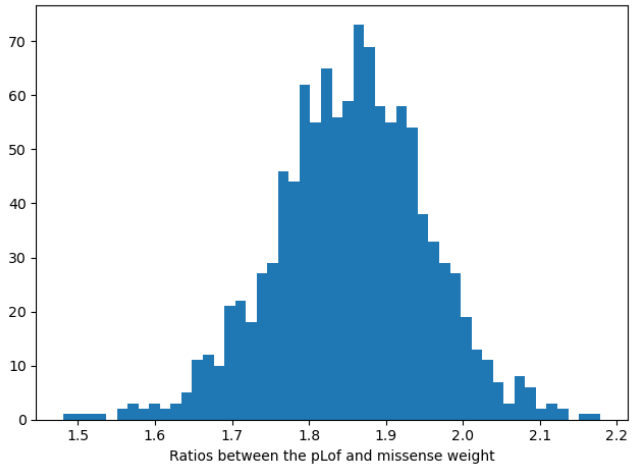

Fig. 4: Distribution of the ratio w_2/w_1 over 1131 separate training runs over at least 30 epochs.

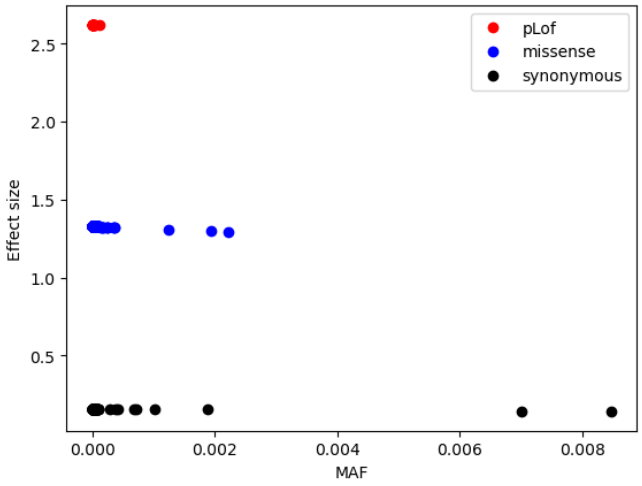

Fig. 5: Estimated phenotypic effect of our model $\sqrt{Beta(maf, \alpha, \beta)} * w_{annotation}$ as a function of MAF.

Weight initialization	Xavier Uniform
MAF bundling	Beta (MAF, α, β)
α *Initialization*	1
β *Initialization*	25
Loss Function	L1
Optimizer	Adam Optimizer
Activation Function	Sigmoid (x - 6)
Phenotype Generation pLoF weight	2
Phenotype Generation Missence weight	1
Phenotype Generation Synonomous weight	0
Loss / Dataset Balancing	No
Data split	[0.4, 0.2, 0.4]
Batch Size	1000
Training time	30 epochs

Table 1: Experimental parameters setup

4 Discussion and Future Works

Bundling MAF with Beta probability density function

From Figure 5 we see the necessity to further develop methods to increase the effect of MAF. One way we propose to do this is with hyperparameter testing, for instance increasing the β or not applying a square root to the output of the function. But that would imply other modelling changes since the higher values would make all the points different to 0 to map to 1 after the sigmoid.

Batch balancing the training data to have variants in every batch

The difficulty of working with rare variants is that they are extremely underrepresented in the population. For this reason, the data shows to be highly unbalanced, especially when training the model batch-wise. To address this issue in future works, we devised a way to balance the data in the training dataset amongst all batches, thus ensuring that every batch will have a minimum of n individuals with at least a single nucleotide polymorphism (SNP) present. In the context of our model, these individuals are the ones that show $sum(present\ SNPs) \neq 0$.

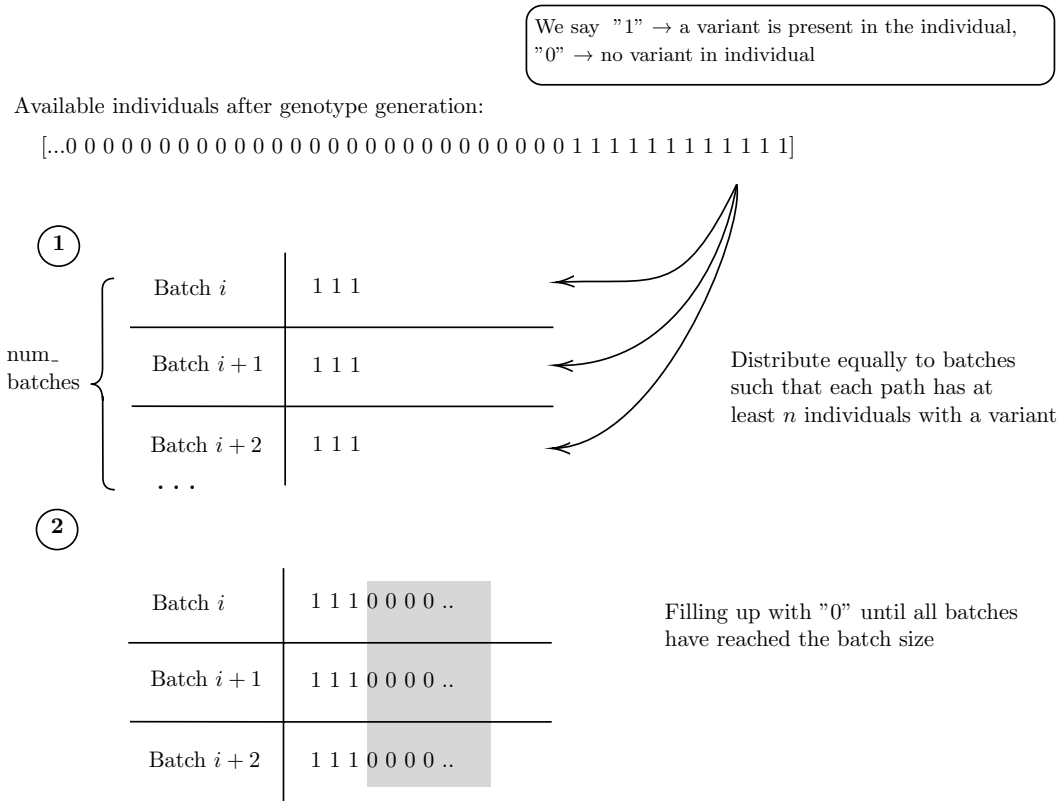

Fig. 6: Visual description of batch balancing such that all batches have a minimum of n individuals with a SNP.

The balancing algorithm as it can be seen in 6 follows these steps:

1. Find the number of batches (num_batches)
2. Find the indices of the individuals that have mutations and the indices of the ones with no mutations
3. Calculate a balanced distribution of individuals that have genetic mutations amongst all batches - $len(individuals)/num\_batches$

4. Create batches of indices following the rule that every batch should have an n number of non-zero indices and the rest until the batch size is reached, are zero indices. This can be observed in 6
5. Based on a parameter defined at the start, remove an $m\%$ of the zeros in every batch
6. Shuffle these indices inside every batch
7. Combine the batches into the dataset again, subsampling the original dataset with the new index order.

At the end of this algorithm, the dataset will be constructed so that for every batch, there will be n number of individuals that have at least 1 SNP.
Our version of this batch balancing does not improve results, however we argue that this method can, with adequate modification and tuning, be made to bring benefits.

Loss balancing function increases the contribution of rare variants

Individuals with pLoF variants tend to be less common than individuals with synonymous variants. To address the imbalance in the number of variants within each annotation category we implemented a modification to the loss function. This technique, commonly employed in classification contexts, has been adapted to the Mean Squared Error (MSE) regression loss used in our model. The approach entails assigning weights to the loss values:

$$loss_b = \sum_{i=0}^{BS} |\text{prediction}_i - \text{target}_i| \cdot w_i \tag{15}$$

where BS is the batch size and b is the index of the current batch.
In our implementation we define three classes of individuals: Those who have at least one pLof, those with at least one missense and no pLofs, and individuals with no missenses nor pLofs. $z \in 0, 1, 2$. We use the above formula defining w_i as:

$$w_i = \frac{1}{|\{j \,|\, \text{class}_j = \text{class}_i \,\text{and}\, j \in \text{training dataset}\}|} \tag{16}$$

where class_i represents the class of individual i, and the set $\{j \,|\, \text{class}_j = \text{class}_i \,\text{and}\, j \in \text{training dataset}\}$ represents the set of individuals in the training dataset that have the same class as individual i.

This adjustment ensures equal contributions from all classes to the overall loss. This addresses the potential issue where a class with a majority of data might disproportionately influence the loss due to minimal errors, overshadowing a less common class with higher errors. However, without this correction, the model may struggle to learn effectively, achieving favorable metrics by accurately predicting the dominant class.

For implementing the loss balancing we introduced these changes:

- Added three columns to our input data X as a one-hot encoding of the individual class.
- Counted the number of times each class appears in the training set and saved its inverse in a loss weight tensor, which is passed as an additional input to the training function
- Created a loss function based on the individual one-hot encoded class, multiplying the value in the loss by the appropriate weight.

Our version of this loss balancing does not improve results, however we argue that this method can, with adequate modification and tuning, be made to bring benefits.

Additional Scores as Input

While our current model successfully utilizes the Minor Allele Frequency (MAF) of a variant to extract valuable information from the data set, future work could focus on incorporating additional valuable inputs to the model. For instance, the deleteriousness scores of the variants calculated by other deep learning initiatives, such as CADD [9], PrimateAI [10], and

Alphamissence [11].

Different variants or fine-tuning hyperparameters

As a preparation for possible future studies, the model was designed in such a way to be able to be easily adapted, so that multiple transformations and variations can be used and tested (e.g. substitution of the shifted sigmoid by a hyperbolic tangent layer).

5 Conclusion

We developed a deep learning model that was able to learn the deteriousness score ratio regarding multiple genomic annotations for variants of a specific gene that outperforms baseline models (such as the allelic series sum and allelic series max) in explaining the variation on the phenotype trait associated with a gene containing allelic series, according to an R^2 analysis. These are promising results when looking further and aiming to incorporate more data and increase the complexity of the allelic series modelation, and hence the better understanding of rare variants and the effect that allelic series show to have on phenotypic traits through the learned deteriousness scores.

References

[1] Zachary R McCaw et al. "An allelic-series rare-variant association test for candidate-gene discovery". In: *The American Journal of Human Genetics* 110.8 (2023), pp. 1330–1342.

[2] Paul L. Auer and Guillame Lettre. "Rare variant association studies: considerations, challenges and opportunities". In: *Genome Medicine* 7 (2015), p. 16.

[3] Brian Clarke et al. "Integration of variant annotations using deep set networks boosts rare variant association genetics". In: *bioRxiv* (2023). DOI: 10.1101/2023.07.12.548506. eprint: https://www.biorxiv.org/content/early/2023/10/26/2023.07.12.548506.full.pdf. URL: https://www.biorxiv.org/content/early/2023/10/26/2023.07.12.548506.

[4] Seunggeun Lee et al. "Optimal unified approach for rare-variant association testing with application to small-sample case-control whole-exome sequencing studies". In: *The American Journal of Human Genetics* 91.2 (2012), pp. 224–237.

[5] Michael C Wu et al. "Rare-variant association testing for sequencing data with the sequence kernel association test". In: *The American Journal of Human Genetics* 89.1 (2011), pp. 82–93.

[6] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[7] Konrad J. Karczewski et al. "Systematic single-variant and gene-based association testing of thousands of phenotypes in 394,841 UK Biobank exomes". In: *Cell Genomics* 2.9 (2022), p. 100168. ISSN: 2666-979X. DOI: https://doi.org/10.1016/j.xgen.2022.100168. URL: https://www.sciencedirect.com/science/article/pii/S2666979X22001100.

[8] *UK Biobank - UK Biobank*. Jan. 12, 2024. URL: https://www.ukbiobank.ac.uk (visited on 02/07/2024).

[9] Max Schubach et al. "CADD v1.7: using protein language models, regulatory CNNs and other nucleotide-level scores to improve genome-wide variant predictions". In: *Nucleic Acids Research* 52.D1 (2024), pp. D1143–D1154.

[10] Laksshman Sundaram et al. "Predicting the clinical impact of human mutation with deep neural networks". In: *Nature genetics* 50.8 (2018), pp. 1161–1170.

[11] Jun Cheng et al. "Accurate proteome-wide missense variant effect prediction with AlphaMissense". In: *Science* 381.6664 (2023), eadg7492.

Code reference

https://github.com/AIH-SGML/allelic-series-a

.1 Ablation Study

We conducted a comparison (check figure 7) between three stages of our model:

- DeepAllelic - Simple Linear Layer
- DeepAllelic2 - Linear Layer + Incorporating MAF via beta pdf
- DeepAllelic3 - Linear + BetaPdf + Non-Linear Layer (Sigmoid)

The remaining parameters are the same as in the Experimentation and Results section. All three models were run for a total of 25 times and the learned weight values were clustered together. The final stage of our model achieves precise results, as seen in the clustering of the data.

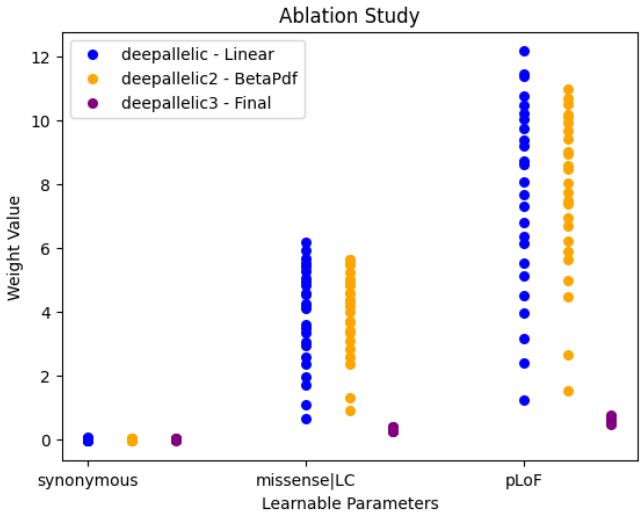

Fig. 7: Clustering of the annotation weights for 25 runs of three different models. The blue model is a classic linear model, followed by an extension taking the minor allelic frequency into account, and finally, our final model uses a sigmoid non-linearity.

.2 Adding alpha missense to the model

We propose an extension to our model by incorporating the results of the deep learning project AlphaMissense, a model which can predict the pathogenic impact of a missense variant [11]. We have extended our used dataset (see 2.2) with predicted pathogenicity scores from AlphaMissense. In Figure 8 we show our suggestion of implementing AlphaMissense into the *weigths* component of our model. The component **?** marks a function which with increasing input values also has increasing outputs. The first consequence we see is that more parameters hence are included (and hence trained) which expands on possible (biological) insights we can draw upon from the model. The second consequence is that this may improve the accuracy of the predicted parameter w.

However, the exact way of optimally integrating alpha-missense into DeepSeries is not clear to us, as we do not know with what mechanism the respective contribution of MAF or alphamissense to *weights* should be managed. This constitutes possible future work as besides of joining the datasets of ALB, we do not have implemented this functionality.

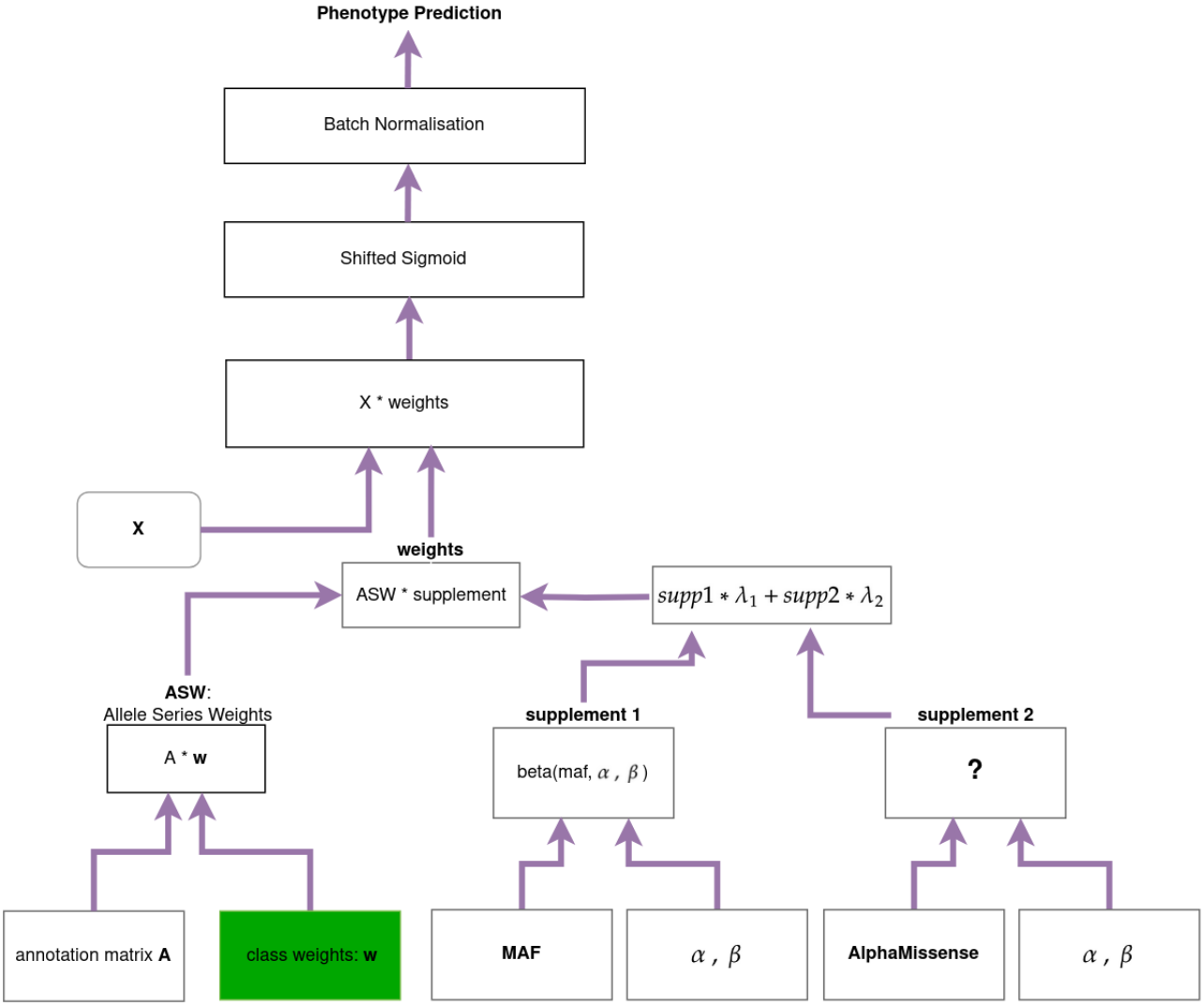

Fig. 8: Proposed inclusion of AlphaMissense into DeepSeries.