Modular Deep Neural Networks with residual connections for predicting the pathogenicity of genetic variants in non coding genomic regions

Federico Stacchietti[1 [0009−0009−1945−134X]], Marco Nicolini[1 [0009−0008−5137−2361]], Leonardo Chimirri[2 [0000−0002−6912−8518]], Peter N. Robinson[2 [0000−0002−0736−9199]], Elena Casiraghi[1,3,4,5 [0000−0003−2024−7572]], and Giorgio Valentini[1,3 [0000−0002−5694−3919]]

[1] AnacletoLab, Dipartimento di Informatica, Università degli Studi di Milano, Italy
[2] Berlin Institute of Health at Charite Universitaetsmedizin Berlin, Germany
[3] ELLIS - European Lab for Learning and Intelligent Systems, Milan Unit
[4] Department of Computer Science, Aalto University, Espoo, Finland
[5] Environmental Genomics and Systems Biology Division, Lawrence Berkeley National Laboratory, Berkeley, CA, United States
{federico.stacchietti, giorgio.valentini}@unimi.it

Abstract. Predicting pathogenic single nucleotide variants (SNVs) in non-coding regions of the human genome presents a significant challenge for the extreme class imbalance between pathogenic "positive" variants and physiological "negative" ones, since most machine learning methods are biased toward predicting negative examples. We designed two "block-shaped" tabular-DNN architectures: a Modular Block-Deep Neural Network (*MoB-DNN*) and a tabular Residual Network (*T-ResNet*), able to address the class imbalance problem through a mini-batch balancing strategy. We employed a hierarchical optimization approach to efficiently tune hyper-parameters related to training procedure, architecture, batch size, and mini-batch balancing ratio. Our experimental results demonstrate that *T-ResNet* outperforms and *MoB-DNN* shows competitive performance with a state-of-the-art hyper-ensemble method, suggesting that residual connections provide significant advantages for capturing complex patterns in non coding regions of the human genome.

Keywords: Deep and modular neural models · Residual connections · Pathogenic variant prediction

1 Introduction

The detection of pathogenic variants responsible for genetic diseases is a fundamental challenge in precision medicine [16]. Among approximately 8,000 Mendelian disorders, i.e. genetic diseases caused by mutations at a single genetic locus, the causative gene mutation remains unknown in more than 50% of cases [5]. While disease-associated variants in protein-coding regions of the genome have been extensively studied [14, 3], pathogenic variants in non-coding regions, responsible for a significant proportion of Mendelian disorders, remain largely unknown [8]. In particular, predicting pathogenic single nucleotide variants (SNVs) in non-coding regions is an open problem. These variants constitute only a tiny fraction compared to the vast number of physiological genetic variants in the human genome. This extreme class imbalance presents a significant challenge for machine learning approaches, as the number of positive (pathogenic) variants is vastly outnumbered by negative (physiological) ones [22].

In literature, several machine learning based methods, ranging from Support Vector Machines, to Random Forests, to Neural Networks have been proposed, but most of them fail to correctly predict pathogenic variants, being biased to predict negative examples [18, 12, 25, 11]. Some methods [21, 6] achieve reasonable results by handling the high imbalance through subsampling and oversampling techniques. Recent methods adopted approaches based on the the effect of SNVs to identify non-coding promoter variants which dysregulate gene expression, thus providing insights into the deleteriousness of SNVs in promoter regions [13]. More in general several methods have been developed to infer the regulatory code directly from genomic sequence [2, 15], but the prediction of the pahogenicity or the effects of non-coding genetic variants remains an open challenge [20].

Other approaches [21, 6] improve performance by mitigating class imbalance through subsampling and oversampling techniques.

In particular, HyperSMURF [21], a method nowadays considered as the state-of-the-art model specifically designed to detect deleterious SNV under extreme class imbalance in the dataset, leverages a hyper-ensemble approach and couples subsampling techniques, to reduce the number of neutral ("negative") variants, and oversampling algorithms, to increase the representation of pathogenic ("positive") variants. This enables the model to effectively handle the inherent data imbalance. Today, it is the machine learning core module of Genomizer, a state of the art tool for the diagnosis of genetic diseases [22].

In this work, we investigate the applicability of Deep Neural Networks (DNNs) for predicting pathogenic SNVs. From a machine learning perspective, the genetic, epigenetic, and conservation-based features used for pathogenic SNV prediction are represented in tabular (structured) format [18, 6]. This format poses challenges due to several factors, most notably the presence of hybrid variable types. While traditional models such as random forests are naturally suited to handle such structured data, recent research has also focused on developing deep learning models for tabular data [4]. Inspired by these works, we designed two modular tabular-DNN architectures, each composed of repetitive blocks of layers of equal size. Their modularity facilitates hierarchical hyper-parameter tuning and architectural optimization, while mini-batch balancing techniques help mitigate the class imbalance in pathogenic variant prediction. A comparison with HyperSMURF [21] demonstrates the effectiveness of our approach.

2 Methods

We designed two deep "block-shaped" neural models for their relatively simplicity and modularity that make them easily tunable using appropriate model selection procedures. The idea is to construct DNNs composed by repetitive blocks of the same size, using at the same time mini-batch balancing techniques to allow the model to learn the pathogenic variants, largely underrepresented in the available genetic data. More precisely we considered the two following deep learning models:

1. *MoB-DNN*: a Modular Blocks-Deep-NN, a deep neural network composed by block modules, i.e. set of layers having the same number of hidden neurons.
2. *T-ResNet*: a ResNet model (borrowed from Convolutional Neural Networks (CNNs) for image processing [9]), working on tabular data, composed by groups of blocks connected by residual connections.

Both models are trained with a tabular SNV-dataset $D = \{\mathbf{x}^i, y^i\}_{i=1}^n$, being n the sample-size, $\mathbf{x}^i \in \mathbb{R}^k$, the vector storing k Mendelian features and $y^i \in \{0, 1\}$ the label associated to the i^{th} genetic variant. All the features are first normalized by standard scaling.

2.1 *MoB-DNN*: A Modular Block-Deep Neural Network.

MoB-DNN is a modular deep neural network designed for ease of use, featuring a relatively small number of hyper-parameters that facilitate model selection.

Its overall architecture is constituted by sequential blocks, all sharing the same structure, i.e. each block is composed by 3 layers (Fig. 1): 1. Linear layer; 2. Activation Layer; 3. Dropout layer.

More precisely, *MoB-DNN* computes the following function, where $\mathbf{x} \in \mathbb{R}^k$ is the k-dimensional vector of the SNV features, $\mathbf{z} \in \mathbb{R}^d$, and d is the dimension, shared by all blocks, of the hidden layers:

$$\mathbf{z} = Input\_layer(\mathbf{x}) \tag{1}$$

$$Block(\mathbf{z}) = Dropout(Act\_layer(Linear\_layer(\mathbf{z}))) \tag{2}$$

$$MoB\text{-}DNN(\mathbf{x}) = Class\_layer(Block_N(Block_{N-1}(\dots(Block_1(Input\_layer(\mathbf{x})))))) \tag{3}$$

where the $Input\_layer$ transforms the input vector of the SNV features into a d-dimensional vector given in input to the first $Block$, and $Class\_layer$ estimates the probability that the SNV variant is pathogenic:

- $Input\_layer : \mathbb{R}^k \to \mathbb{R}^d$
- $Block : \mathbb{R}^d \to \mathbb{R}^d$
- $Class\_layer : \mathbb{R}^d \to \mathbb{R}$

The activation layer Act_layer introduces non-linearity in the prediction through the ReLU function [1], and the $Dropout$ layer regularizes the neural network [23].

Notably, all blocks maintain the same hidden dimension d, allowing the overall blocks to be optimized by tuning a single hyper-parameter.

2.2 *T-ResNet*: Tabular ResNet.

The *T-ResNet* model is structurally similar to the *MoB-DNN* model, but explicitly incorporates residual connections, borrowed from CNN for image processing [9]. Residual connections help mitigate the vanishing gradient problem commonly encountered in deep neural network architectures by adding a direct connection between the input and output of specific sequences of layers. More precisely, if $F(x) \in \mathbb{R}^d$ represents the output of a set of layers with input $x \in \mathbb{R}^d$, a residual connection can be implemented by simply summing the input with the output:

$$\text{res}(x) = F(x) + x \tag{4}$$

T-ResNet is structured into repetitive modules called *groups*. Each group consists of consecutive *blocks*, where each block is identical to the *MoB-DNN* block, maintaining a uniform hidden dimension d. A single residual connection spans the entire group, meaning the input to a group is directly added to its output, effectively creating a shortcut across multiple layers.

Formally, the *T-ResNet* model consists of the following modular components:

- **Input layer** : $\mathbb{R}^k \to \mathbb{R}^d$
- **Block** : $\mathbb{R}^d \to \mathbb{R}^d$ (Linear + ReLU + Dropout)
- **Group** : $\mathbb{R}^d \to \mathbb{R}^d$ (sequence of multiple Blocks + residual connection)

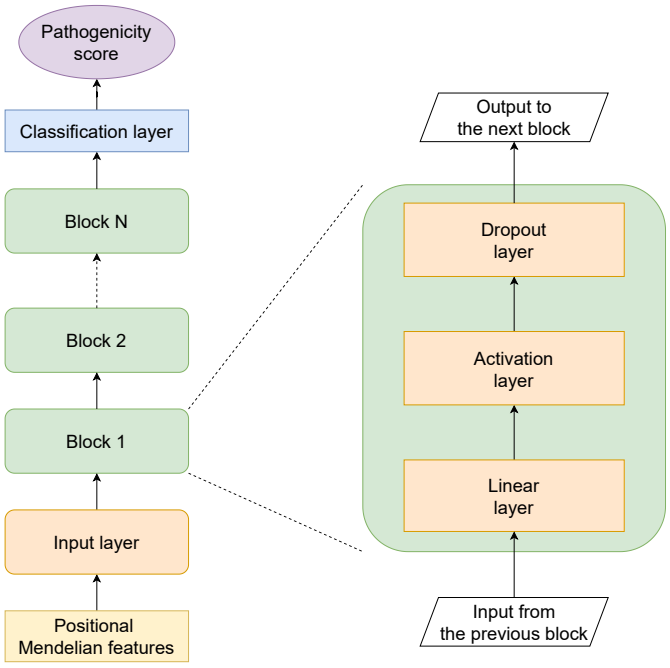

Fig. 1: High-level scheme of the *MoB-DNN* model. On the left a scheme of the overall modular model including N blocks. On the right the structure of each block, composed by a sequence of Linear, Activation and Droupout layers.

– **Class layer** : $\mathbb{R}^d \to \mathbb{R}$

Given an input vector $x \in \mathbb{R}^k$, the *T-ResNet* model computes the following functions, with intermediate representation $z \in \mathbb{R}^d$:

$$z = Input\_layer(x) \tag{5}$$

$$Block(z) = Dropout\big(ReLU(Linear(z))\big) \tag{6}$$

$$Group(z) = \underbrace{Block(Block(\dots Block}_{\text{number of blocks per group}}(z))) + z \tag{7}$$

$$T\text{-}ResNet(x) = Class\_layer\big(Group_N(Group_{N-1}(\dots(Group_1(Input\_layer(x))))\big) \tag{8}$$

The dimension d is maintained constant across all hidden layers, and each group contains an identical number of blocks. Fig. 2 illustrates the high-level scheme of the *T-ResNet* architecture.

As an example, if the *T-ResNet* model has 4 groups with 3 blocks per group, the resulting network consists of 12 blocks in total, with residual connections between the four groups.

By introducing residual connections, *T-ResNet* facilitates more effective gradient propagation during training, alleviating issues commonly encountered in deep neural network architectures, such as vanishing gradient effects. Again, its modularity diminishes the number of architectural parameters to be optimized.

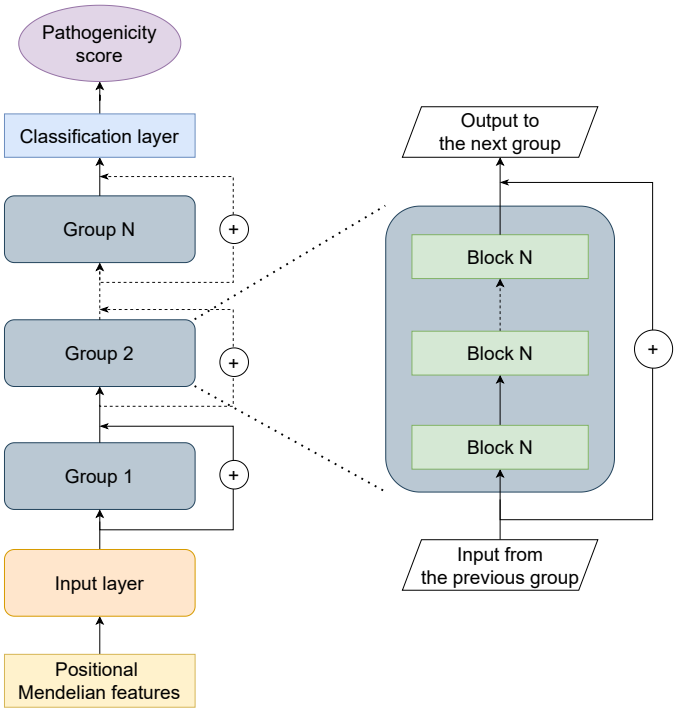

Fig. 2: High-level scheme of the *T-ResNet* model. On the left a scheme of the overall modular model including N groups. On the right the structure of each group, composed by several blocks.

2.3 Mini-Batch Balancing Strategy.

Given the extreme class imbalance in our dataset (negative-to-positive ratio on the order of 10^4), standard training procedures tend to overfit negative examples. To counteract this, we implemented a mini-batch balancing strategy that ensures a comparable number of positive and negative examples in each mini-batch.

More precisely, let D be the dataset containing n examples, with the subset of positives denoted as $P \subset D$, where $|P| = n_+$, and the subset of negatives as $N \subset D$, where $|N| = n_-$. A mini-batch B of size $|M| = m$ is constructed by sampling:

- m_+ positive examples, drawn *with replacement* from P using a uniform distribution.
- m_- negative examples, drawn *without replacement* from N using a uniform distribution.

The degree of imbalance in B is controlled by the ratio r of negative examples to the total mini-batch size:

$$r = \frac{m_-}{m_- + m_+}, \quad \text{such that} \quad m_- + m_+ = m. \tag{9}$$

The ratio r ranges within $0 < r < 1$, where $r = 0.5$ indicates perfect balancing, while $r = 0.9$ implies that 90% of the mini-batch consists of negatives.

Since the number of positive examples is limited, we sample them *with replacement* to allow for sufficiently small values of r (i.e., a larger proportion of positives) even for large m. Conversely, negative examples are sampled *without replacement* to ensure a more extensive coverage of the available negative examples across training iterations.

3 Results

3.1 Datasets

For training and testing our neural models, we used a dataset including more than 14 million single nucleotide variants included in the non coding regions of the human genome, previously prepared and analyzed in [22]. Therein, only 406 were pathogenic, thus resulting in an imbalance between pathogenic (positive) and physiological (negative) SNVs of about $1 : 35,000$. Each SNV is characterized by 26 numerical features that include genomic attributes downloaded from public data bases (UCSC, NCBI and others). The main features represent conservation scores, transcriptional, regulation and epigenomic features (e.g. DNAse features, histone methylation and acetylation, transcription factor binding sites), and other features relevant to assess the potential pathogenicity of the genetic variants (see [22] for more details).

3.2 Hyper-parameter optimization

In order to optimize hyper-parameter configurations and maintain computational efficiency, both DNNs and HyperSMURF models were optimized by applying a grid search strategy within a cross-validation framework. First, the full dataset was stratified and split into a stratified external holdout, with 80% of the positive and negative instances assigned to the training set and 20% to the test set. The external training set was further used to run an internal 5-fold stratified cross-validation and the best hyper-parameter configurations were chosen by maximizing the mean AUPRC scores across the internal validation folds.

For optimizing the neural network models (both *MoB-DNN* and *T-ResNet*), due to the high number of hyper-parameters to be tuned, we adopted a hierarchical optimization approach (Section 3.3). In contrast, the HyperSMURF models underwent a single comprehensive grid search to determine the best hyper-parameter settings (Section 3.4).

3.3 DNNs hyper-parameter optimization procedure

Hierarchical Hyper-parameter Optimization. The advantage of the proposed DNN models lies in their modularity, which significantly reduces the number of architectural hyper-parameters to optimize. However, deep neural networks involve a large set of hyper-parameters, some of which are crucial yet often overlooked. Beyond evaluating DNNs for SNV pathogenicity classification, this work aims to determine whether a hierarchical and computationally efficient optimization procedure can yield promising results.

We defined a three-step hierarchical optimization process: (1) hyper-parameters optimization related to the training procedure, which are often preset and overlooked in literature; (2) by fixing the selected training hyper-parameters set, further selection of a set of optimal architectural hyper-parameters; (3) having narrowed the range of architectural hyper-parameters, final selection of the optimal batch size and mini-batch balancing ratio.

Preliminary selection of learning hyper-parameters on data subsamples. Given the substantial computational cost that a hyper-parameter optimization via grid-search would require, we implemented a preliminary hyper-parameter selection procedure on a data subsample. To this aim, we created five datasets, each containing all the positive samples and $500,000$ randomly selected negative samples. Each subsample was split into stratified train and validation sets (75:25 train:test ratio) and the optimal hyper-parameters were chosen by applying grid search to maximize the mean AUPRC over the five validation sets.

The aim of this phase was to analyze the effect of (often overlooked) model-training hyper-parameters to discard evidently suboptimal configurations at an early stage, thereby mitigating computational costs. In particular, by training simple *MoB-DNN* architectures (comprising 3–5 blocks with 128–256 units per layer), we set the learning rate to 10^{-3}, and we set a linear decay learning rate scheduling. We also chose to apply the Adam optimization algorithm and set the batch size to $8,192$ with a negative/positive ratio per batch of 0.8. Training continued until 100 epochs (with early stopping implemented), and we applied a dropout rate of 0.3.

Convergence analysis showed that the best test AUPRC scores were achieved within 30 to 50 epochs, with minimal improvement beyond this range. Thus, we chose to set a minimum of 30 epochs and a maximum of 50, applying early stopping with a patience of 10 to prevent overfitting while maintaining computational efficiency.

The same hyper-parameters were also used for the *T-ResNet* architecture.

Additionally, we consistently employed the Rectified Linear Unit (ReLU) activation function and the Cross Entropy Loss function due to their well-established effectiveness in deep neural network classification tasks [10, 17].

Selection of the model architecture. Once the training hyper-parameters were chosen, the second step applied the grid search procedure detailed in Section 3.2 to narrow the range of optimal architectural hyper-parameters, i.e. we applied a 5-fold internal cross-validation on the full training set.

For the *MoB-DNN* architecture, we explored configurations varying in depth (number of blocks) and width (number of hidden units per linear layer in a block). Specifically, the following architectural parameters were examined for the *MoB-DNN* model: number of blocks in $\{3, 5, 7, 9, 11\}$; and number of units per linear layer in a block (hidden layer size) in $\{64, 128, 256, 512\}$.

For *T-ResNet*, we explored all combinations of the following architectural hyper-parameters: number of groups in $\{3, 5, 7, 9\}$; number of blocks per group in $\{1, 2, 3\}$; and number of units per layer in $\{128, 256, 512\}$.

Selecting the configurations that achieved the highest mean AUPRC across the internal validation sets, we can narrow the set of architectural hyper-parameters to explore. These sets of hyper-parameters were then considered in the next step for batch size and mini-batch balancing optimization.

For the *MoB-DNN* model the best architectural hyper-parameters were a number of blocks equal to 3 or 7, and a number of units per layer equal to 256 or 512. For the *T-ResNet* model, the set of architectural parameters included a number of groups equal to 3 or 5, a number of blocks per group equal to 1 or 3, a number of units per hidden layer equal to 128.

Refinement of the models with respect the batch size and the mini-batch imbalance. After the architectural selection described above, we performed a further refinement step focusing on batch-related hyper-parameters: the batch size and the mini-batch balancing ratio (see Section

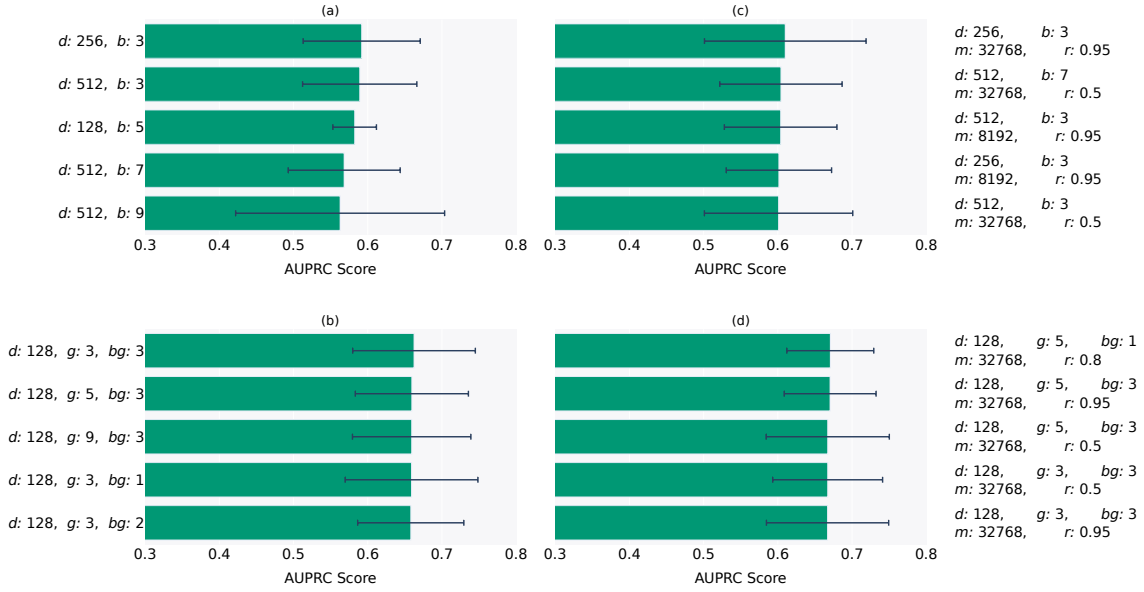

Fig. 3: AUPRC of top 5 configurations for *MoB-DNN* and *T-ResNet* models obtained by internal cross-validation on the training set. First row refers to the *MoB-DNN*, second row to *T-ResNet*. (a) Best 5 *MoB-DNN* models resulting from the selection of the model architecture; (b) Best 5 *T-ResNet* models resulting from the selection of the model architecture; (c) Best 5 *MoB-DNN* models resulting from the further selection of the batch size and the ratio of the mini-batch balancing; (d) Best 5 *T-ResNet* models resulting from the further selection of the batch size and the ratio of the mini-batch balancing. Error bars represent the standard deviation. *d*: number of neural units for each layer, *b*: number of blocks, *g*: number of groups, *bg*: number of blocks per group, *m*: batch size, *r*: mini-batch balancing (negative-to-positive ratio).

2.3). Using the selected architectural hyper-parameters, we run the final grid search by internal cross-validation on the training set, to evaluate the following search spaces for both *MoB-DNN* and *T-ResNet*: batch size in $\{1024, 8192, 32768\}$; and mini-batch balancing (negatives/positives) in $\{0.50, 0.8, 0.95\}$.

Figure 3 shows the top 5 configurations by average AUPRC across the 5 folds of the internal cross-validation.

Note that *MoB-DNN* and *T-ResNet* without mini-batch balancing and with a relatively small mini-batch size (1024) cannot learn anything (their AUPRC is close to 0). By enlarging the mini-batch size to $32,768$, we can obtain better results, e.g. AUPRC of 0.454 for MoB-DNN, since the larger size improves the probability of including some positives in the mini-batch. Nevertheless, the results are relatively poor with respect to the same models trained using mini-batch balancing.

3.4 HyperSMURF hyper-parameter Selection

HyperSMURF hyper-parameters were tuned using grid search to maximize the mean AUPRC computed by internal cross-validation (Section 3.2).

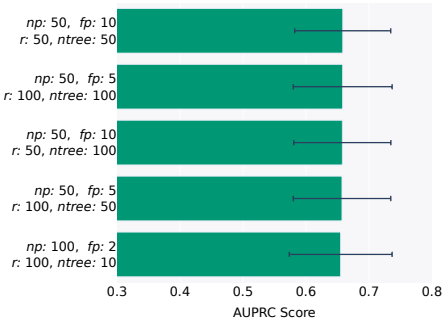

Fig. 4: AUPRC of top 5 configurations for the HyperSMURF models. Configuration labels: *np*: number of partitions, *fp*: oversampling factor, *r*: undersampling ratio, *ntree*: number of trees.

The hyper-parameter search space was chosen based on the hyper-parameter optimization results reported by authors of [21] using the same dataset. In particular, the grid search space was defined as follows.

- *Number of partitions (np)*: each partition includes all positive examples and a unique subset of negatives (i.e., a maximum of N_-/np negatives); we tested the following values: $np \in \{50, 100, 150\}$.
- *Oversampling factor for positive examples (fp)*: if N_+ is the number of positives, the cardinality of the over-sampled positive set, obtained through the SMOTE algorithm [7], is $n_+ = N_+ + (\text{fp} \times N_+)$; we tested the following values: $fp \in \{2, 5, 10, 15, 20\}$.
- *Negative to positive ratio (r)*: this parameter controls the number of negatives in each partition. The number of negatives is $min(r \times n_+, N_-/np)$; We tested the following values: $r \in \{2, 3, 10, 50, 100\}$.
- *Number of trees (ntree)* in each random forest; we tested with $ntree \in \{10, 50, 100\}$.

All other HyperSMURF and random forest settings were kept at their defaults. The top 5 configurations are shown in Figure 4.

3.5 Results on the Test Set

The AUPRC scores on the external test set across all top $k = 5$ configurations for each type of model ranged between approximately 0.5 and 0.6, reflecting the challenges posed by the highly imbalanced dataset. Note that AUROC values are always larger than 0.9 for all the considered models, but we reported only AUPRC results, since this metric is better suited to evaluate very imbalanced data [19].

In Table 1, we present the final results obtained from the best configurations for each model type. Notably, *T-ResNet* achieved the highest Test AUPRC of 0.624, followed by HyperSMURF with 0.581, and *MoB-DNN* with 0.561. The difference in favor of *T-ResNet* is always statistically significant (Wilcoxon paired rank sum test, $\alpha = 10^{-4}$). This outcome is particularly striking considering that our dataset is tabular in nature, a domain where tree-based methods such as HyperSMURF

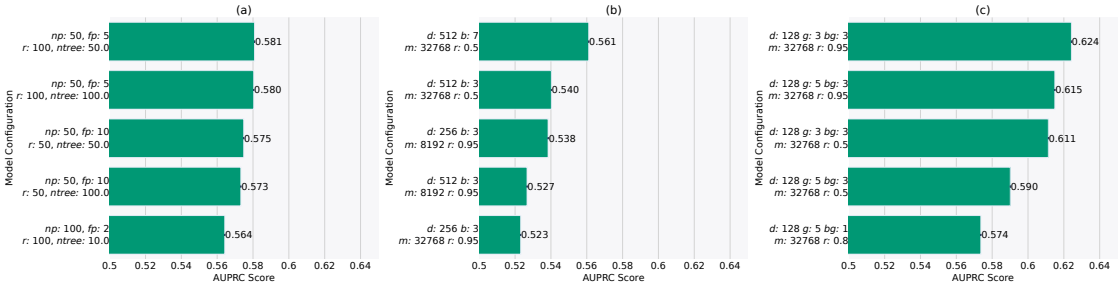

Fig. 5: Comparison of the AUPRC results on the test set for the 5 best models of (a) HyperSMURF, (b) *MoB-DNN* and (c) *T-ResNet*. Configuration labels: *np*: number of partitions, *fp*: oversampling factor, *r*: undersampling ratio, *ntree*: number of trees, *d*: number of neural units for each layer, *b*: number of blocks, *g*: number of groups, *bg*: number of blocks per group, *m*: batch size, *r*: mini-batch balancing (negative-to-positive ratio)

are traditionally expected to have an edge over deep learning models. The ResNet's superior performance suggests that the use of residual layers is an important addition compared to *MoB-DNN* models.

4 Conclusions

Our results show that well-tuned deep neural networks can be competitive with state-of-the-art methods to predict pathogenic variants in non coding regions of the human genome. Moreover balancing techniques are crucial to achieve competitive results in this challenging problem, where the number of "negative" physiological genetic variants largely outnumber "positive" pathogenic SNVs.

These findings pave the way for further research, such as integrating ensemble methods with deep neural network models, to potentially enhance predictive performance beyond current benchmarks.

Model Type	Best Model Config	Test AUPRC
T-ResNet	N. Units: 128, N. Groups: 3, Blocks per Group: 3, Batch Size: 32,768, Neg/Pos Ratio: 0.95	0.624
HyperSMURF	N. Partitions: 50, Oversampling factor: 5, Undersampling ratio: 100, N. Trees: 50	0.581
MoB-DNN	N. Blocks: 7, N. Units: 512, Batch Size: 32,768, Neg/Pos Ratio: 0.5	0.561

Table 1: AUPRC scores for the best configuration of each model type on the test set.

Beyond raw performance, deep networks offer several key advantages over traditional tree-based methods. Indeed, deep network architectures can be more easily integrated with other neural network models, like transformer models [24], enabling the creation of hybrid systems that leverage the strengths of both architectures for multi-modal data, including nucleotide sequence data and tabular genomic and epigenomic features for the prediction of pathogenic single nucleotide variants in non coding regions of human genome.

Acknowledgments. This work was supported by National Center for Gene Therapy and Drugs Based on RNA Technology—MUR (Project no. CN_00000041) funded by NextGeneration EU program, and by FAIR (Future Artificial Intelligence Research) project, funded by the NextGenerationEU program within the PNRR-PE-AI scheme (M4C2, Investment 1.3, Line on Artificial Intelligence) - AIDH – FAIR - PE0000013.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Codebase. The *MoB-DNN* and *T-ResNet* models code and scripts to reproduce the experiments are available from GitHub: `https://github.com/AnacletoLAB/T-ReMM-FFNN`.

References

1. Agarap, A.: Deep learning using rectified linear units (relu). CoRR **abs/1803.08375** (2018), `http://arxiv.org/abs/1803.08375`
2. Avsec, Z., Agarwal, V., Visentin, D., et al.: Effective gene expression prediction from sequence by integrating long-range interactions. Nat Methods **18**, 1196–1203 (2021). https://doi.org/10.1038/s41592-021-01252-x
3. Bendl, J., Musil, M., Stourac, J., Zendulka, J., Damborsky, J.and Brezovsky, J.: Predictsnp2: A unified platform for accurately evaluating snp effects by exploiting the different characteristics of variants in distinct genomic regions. PLOS Computational Biology **e100496** (2016)
4. Borisov, V., Leemann, T., Seßler, K., Haug, J., Pawelczyk, M., Kasneci, G.: Deep neural networks and tabular data: A survey. IEEE Transactions on Neural Networks and Learning Systems **35**(6), 7499–7519 (2024). https://doi.org/10.1109/TNNLS.2022.3229161
5. Boycott, K., Rath, A., Chong, J., et al.: International cooperation to enable the diagnosis of all rare genetic diseases. The American Journal of Human Genetics **100**(5), 695–705 (2017). https://doi.org/https://doi.org/10.1016/j.ajhg.2017.04.003, `https://www.sciencedirect.com/science/article/pii/S0002929717301477`
6. Caron, B., Luo, Y., Rausell, A.: Ncboost classifies pathogenic non-coding variants in mendelian diseases through supervised learning on purifying selection signals in humans. Genome Biology **20**(1), 32 (Feb 2019). https://doi.org/10.1186/s13059-019-1634-2, `https://doi.org/10.1186/s13059-019-1634-2`
7. Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W.: SMOTE: Synthetic Minority Over-sampling Technique. Journal of Artificial Intelligence Research **6**, 321–357 (2002)
8. Edwards, S.L., Beesley, J., French, J.D., Dunning, A.M.: Beyond gwass: illuminating the dark road from association to function. American Journal of Human Genetics **93 5**, 779–97 (2013)
9. He, K., Zhang, X., Ren, S., Sun.J.: Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 770–778 (2016), `https://api.semanticscholar.org/CorpusID:206594692`
10. Hu, X., Niu, P., Wang, J., Zhang, X.: A dynamic rectified linear activation units. IEEE Access **7**, 180409–180416 (2019)
11. Huang, Y.F., Gulko, B., Siepel, A.: Fast, scalable prediction of deleterious noncoding variants from functional and population genomic data. Nature Genetics **49**, 618–624 (2017). https://doi.org/10.1038/ng.3810
12. Ionita-Laza, I., McCallum, K., Xu, B., Buxbaum, J.D.: A spectral approach integrating functional genomic annotations for coding and noncoding variants. Nat Genet **48**(2), 214–20 (Feb 2016). https://doi.org/10.1038/ng.3477
13. Jaganathan, K., Ersaro, N., Novakovsky, G., et al.: Predicting expression-altering promoter mutations with deep learning. Science p. eads7373 (2025). https://doi.org/10.1126/science.ads7373
14. Kumar, P., Henikoff, S., Ng, P.: Predicting the effects of coding non-synonymous variants on protein function using the sift algorithm. Nat Protoc. **4**(7), 1073–81 (2009)

15. Linder, J., Srivastava, D., Yuan, H., et al.: Predicting RNA-seq coverage from DNA sequence as a unifying model of gene regulation. Nat Genet **57**, 949–961 (2025). https://doi.org/10.1038/s41588-024-02053-6
16. Maddirevula, S., Kuwahara, H., Ewida, N., et al.: "analysis of transcript-deleterious variants in mendelian disorders: implications for rna-based diagnostics". Genome Biol **21**(145) (2020). https://doi.org/10.1186/s13059-020-02053-9
17. Mao, A., Mohri, M., Zhong, Y.: Cross-entropy loss functions: Theoretical analysis and applications. In: International conference on Machine learning. pp. 23803–23828. PMLR (2023)
18. Rentzsch, P., Witten, D., Cooper, G., Shendure, J., Kircher, M.: Cadd: predicting the deleteriousness of variants throughout the human genome. Nucleic Acids Res. **47**(D1), D886–D894 (2019). https://doi.org/10.1093/nar/gky1016
19. Saito, T., Rehmsmeier, M.: The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. PLoS ONE **10**(3) (2015). https://doi.org/10.1371/journal.pone.0118432
20. Sasse, A., Ng, B., Spiro, A., et al.: Benchmarking of deep neural networks for predicting personal gene expression from DNA sequence highlights shortcomings. Nat Genet **55**, 2060–2064 (2023). https://doi.org/10.1038/s41588-023-01524-6
21. Schubach, M., Re, M., Robinson, P.N., Valentini, G.: Imbalance-aware machine learning for predicting rare and common disease-associated non-coding variants. Scientific Reports **7**(1), 2959 (2017). https://doi.org/10.1038/s41598-017-03011-5
22. Smedley, D., Schubach, M., Jacobsen, J.O., Köhler, S., Zemojtel, T., Spielmann, M., Jäger, M., Hochheiser, H., Washington, N.L., McMurry, J.A., Haendel, M.A., Mungall, C.J., Lewis, S.E., Groza, T., Valentini, G., Robinson, P.N.: A Whole-Genome Analysis Framework for Effective Identification of Pathogenic Regulatory Variants in Mendelian Disease. The American Journal of Human Genetics **99**(3), 595–606 (sep 2016). https://doi.org/10.1016/j.ajhg.2016.07.005, `http://linkinghub.elsevier.com/retrieve/pii/S0002929716302786`
23. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**(56), 1929–1958 (2014), `http://jmlr.org/papers/v15/srivastava14a.html`
24. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. p. 6000–6010. NIPS'17, Curran Associates Inc., Red Hook, NY, USA (2017)
25. Zhou, J., Troyanskaya, O.G.: Predicting effects of noncoding variants with deep learning-based sequence model. Nature Methods **12**(10), 931–934 (Aug 2015). https://doi.org/10.1038/nmeth.3547, `http://dx.doi.org/10.1038/nmeth.3547`