Fan and Waldmann *BMC Bioinformatics* (2024) 25:322
https://doi.org/10.1186/s12859-024-05940-1

BMC Bioinformatics

RESEARCH **Open Access**

Tabular deep learning: a comparative study applied to multi-task genome-wide prediction

Yuhua Fan[1] and Patrik Waldmann[1*]

*Correspondence:
Patrik.Waldmann@oulu.fi

[1] Research Unit of Mathematical Sciences, University of Oulu, P.O. Box 8000, 90014 University of Oulu, Finland

Abstract

Purpose: More accurate prediction of phenotype traits can increase the success of genomic selection in both plant and animal breeding studies and provide more reliable disease risk prediction in humans. Traditional approaches typically use regression models based on linear assumptions between the genetic markers and the traits of interest. Non-linear models have been considered as an alternative tool for modeling genomic interactions (i.e. non-additive effects) and other subtle non-linear patterns between markers and phenotype. Deep learning has become a state-of-the-art non-linear prediction method for sound, image and language data. However, genomic data is better represented in a tabular format. The existing literature on deep learning for tabular data proposes a wide range of novel architectures and reports successful results on various datasets. Tabular deep learning applications in genome-wide prediction (GWP) are still rare. In this work, we perform an overview of the main families of recent deep learning architectures for tabular data and apply them to multi-trait regression and multi-class classification for GWP on real gene datasets.

Methods: The study involves an extensive overview of recent deep learning architectures for tabular data learning: NODE, TabNet, TabR, TabTransformer, FT-Transformer, AutoInt, GANDALF, SAINT and LassoNet. These architectures are applied to multi-trait GWP. Comprehensive benchmarks of various tabular deep learning methods are conducted to identify best practices and determine their effectiveness compared to traditional methods.

Results: Extensive experimental results on several genomic datasets (three for multi-trait regression and two for multi-class classification) highlight LassoNet as a standout performer, surpassing both other tabular deep learning models and the highly efficient tree based LightGBM method in terms of both best prediction accuracy and computing efficiency.

Conclusion: Through series of evaluations on real-world genomic datasets, the study identifies LassoNet as a standout performer, surpassing decision tree methods like LightGBM and other tabular deep learning architectures in terms of both predictive accuracy and computing efficiency. Moreover, the inherent variable selection property of LassoNet provides a systematic way to find important genetic markers that contribute to phenotype expression.

Keywords: Tabular data, Multi-trait, Genome-wide prediction (GWP), Non-linear models

Background

Genome-wide prediction has become widespread as a valuable tool to estimate genetic merit in animal and plant breeding [1, 2]. In humans, GWP has been widely used to predict disease risk of highly polygenic complex human traits [3]. Genomic data is usually represented in matrices with markers in columns and individual observations in rows. Linear mixed models for GWP either model the markers effects directly or via pedigree based covariance models and both approaches often obtain competitive results [4, 5], but comparative studies show that machine learning (ML) methods are better in modelling various interactions in the genome [6]. In recent years, deep neural networks have emerged as powerful tools across various domains, ranging from sound and image analysis to natural language processing [7]. In these problems, the data points are represented as vectors of structured homogeneous features. However, in many other real-world applications, tabular data remains the most prevalent data type, consisting of samples in rows and different features in columns that don't follow any simple structure [8, 9]. Over the past decade, numerous supervised, self-supervised, and semi-supervised learning methods have been proposed to address the specific challenges associated with modeling tabular data.

Decision tree models represent a widely adopted machine learning technique and serve as the foundation for more sophisticated ensemble methods such as random forests [10] and gradient boost decision trees (GBDT) [11–13]. Classical machine learning approaches, including GBDT, have dominated tabular data applications due to their superior performance. Various GBDT methods, such as XGBoost [11], LightGBM [12], and CatBoost [13], construct a robust predictor by ensembling weaker predictors through gradient descent in a functional space. They have been applied with success across diverse domains [14–16]. Despite their ability to model non-linear dependencies, decision trees struggle with out-of-distribution samples [17]. Furthermore, these methods lack flexibility and ability to utilize pre-trained models, thereby limiting their utility. However, they are computationally very efficient and can be utilized on big data sets.

In the context of genomics, a lot of research effort has been invested into prediction of molecular properties of organisms [18]. Single-layer neural networks have been utilized in animal and plant breeding [19, 20]. These shallow neural networks have been found to be prone to overfitting, although they are sometimes competitive with penalized linear models. Recently, several applications of GWP using deep learning have been studied and compared with other methods [21, 22]. The most commonly employed deep learning architectures in genomic prediction are the multi-layer perceptron (MLP) and one-dimensional convolutional neural networks (CNNs), which have been used for GWP in for example humans [23], pigs [24], wheat [25, 26] and soybean [27]. Current research has demonstrated that the CNN-RNN method exhibit superior performance compared to models based on random forest, deep fully connected neural networks, and conventional Lasso regression [28]. A deep neural network was designed to predict maize yield across 2,247 locations from 2008 to 2016 using both genotype and environmental data in the yield testing stage [29]. Furthermore, a number of studies have used multi-omics data to predict complex traits in various organisms, including animals, humans and plants [30–32].

The tabular representation of genomic data offers a structured framework for organizing, analyzing, and interpreting complex biological information [33]. Except for the usually underperforming MLP, few tabular deep learning models have been applied to GWP tasks. One exception is the GPtransformer which is a Transformer-based deep learning algorithm that was applied to predicting Fusarium related traits in Barley [34]. Neural networks offer an end-to-end pipeline of hierarchical feature extractors and a final estimator for tabular data, with components trained together to minimize loss functions using gradient-based strategies. Unlike decision trees, neural networks can to some extent maintain performance on out-of-distribution data [17]. Recent research has highlighted deep learning as a promising alternative approach to decision trees [9]. Based on Lasso regularization, three prominent models, the CNNGWP [24], the LassoNet [35] and the Neural Lasso [36], underscore the flexibility and efficiency of regularized deep learning methods. Developed by researchers at Google Cloud AI, TabNet [37] leverages the advantages of decision trees and attention mechanisms to effectively learn from tabular data, making it well-suited for tasks such as classification and regression. Additionally, NODE [38] combines neural oblivious decision trees with dense connections, applying the ensemble method of oblivious decision trees to neural networks using differentiable trees based on the entmax function. TabR [39] introduces a retrieval-augmented tabular deep learning architecture, achieving strong performance through attention-based retrieval components. AutoInt [40] reduces data sparsity by transforming high-dimensional data into a low-dimensional space using an embedding layer with a gating mechanism for feature representation learning and feature selection. Moreover, GANDALF [41] proposes a gated adaptive network for deep automated feature learning, incorporating a gating mechanism for feature representation learning. Tab-Transformer [42] utilizes self-attention based transformers to transform categorical feature embeddings into robust contextual embeddings for improved prediction accuracy. FT-Transformer [43], adapted from the transformer architecture, embeds all features before applying a stack of transformer layers to operate on the feature level, specifically designed for tabular data. By leveraging several mechanisms to overcome the difficulties of training on tabular data, SAINT [44] was proposed to boost performance for semi-supervised problems with a self-attention and intersample attention transformer.

Compared to typical linear statistical models, multi-task deep learning for genomic prediction tasks presents a more flexible and powerful framework by leveraging feature representation learning and facilitating learning across tasks. Although multi-task deep learning models perform well for classification or regression tasks on homogeneous input data (e.g., image, audio, and text data), its application to tabular genomic data still pose a challenge in the high-dimensional setting where the number of feature variables is considerably higher than the number of observations. In this paper, we present a comparative GWP study of tabular deep learning together with LightGBM as a GBDT representative using five real datasets for both regression and classification.

Results

Model comparison

Multi-trait regression: The results from the regression tasks are summarized in the Table 1. The comparative analysis of different models for multi-trait regression tasks across three datasets-Mice, Pig, and Wheat-reveals notable variations in performance, primarily measured by Mean Squared Error (MSE) and correlation coefficient (r). LassoNet consistently outperforms other models, achieving the lowest MSE and highest correlation across all datasets (e.g., Mice data: MSE $= 0.135 \pm 0.003$, $r = 0.735$; Pig data: MSE $= 0.151 \pm 0.002$, $r = 0.722$). This indicates LassoNet's superior ability to capture the important information in the underlying data structure. In contrast, models like NODE and TabR show relatively poor performance, with higher MSE and lower correlation values, indicating less accurate predictions (e.g., NODE on Pig data: MSE $= 0.242 \pm 0.007$, $r = 0.531$). GANDALF demonstrates competitive results, especially in the Wheat dataset, where it records an MSE of 0.191 ± 0.004 and a correlation of 0.651, close to LassoNet's performance. The TabNet models, both supervised and unsupervised, perform moderately well across the datasets, with supervised TabNet slightly outperforming the unsupervised version. For instance, on the Pig data, supervised TabNet achieves an MSE of 0.225 ± 0.006 and $r = 0.607$, whereas the unsupervised version shows a slightly higher MSE of 0.243 ± 0.004 and $r = 0.621$. FT-Transformer, Tab-Transformer and SAINT offer a middle ground in terms of performance, with MSE values consistently lower than NODE but higher than LassoNet, reflecting a decent trade-off between complexity and predictive power. The results clearly demonstrate that LassoNet is the most effective model for multi-trait regression tasks over these datasets.

Multi-class classification: The experimental results for multi-class classification for GWD are reported in Table 2. In the multi-trait classification tasks, the performance of various models was evaluated based on their accuracy, area under the curve (AUC), and Brier score across two datasets: the 14-cancer microarray data and the Loblolly pine data (subset). Among the models, LassoNet once again emerged as the top performer, achieving the highest accuracy (90.1 ± 0.17) and AUC (91.5 ± 0.152) on the microarray data, and further outperforming other models on the Loblolly pine data with an accuracy of 93.5 ± 0.19 and an AUC of 96.7 ± 0.190, coupled with the lowest Brier scores (0.174 and 0.183, respectively). It is clear that LassoNet has strong capability in both predictive power and reliability. In contrast, TabNet (unsupervised) displayed the weakest performance, with the lowest accuracy (81.2 ± 0.37) and AUC (84.5 ± 0.205) on the microarray data as well as worse metrics on the Loblolly pine data (accuracy of 90.3 ± 0.25 and AUC of 93.6 ± 0.233). This highlights the challenges faced by unsupervised approaches in these prediction tasks. GANDALF and FT-Transformer also showed competitive results, with GANDALF reaching an accuracy of 87.4 ± 0.33 and an AUC of 89.7 ± 0.163 on the microarray data. On the Loblolly pine data the accuracy was 91.5 ± 0.26 and AUC 95.1 ± 0.194, indicating relatively robust performance. FT-Transformer also positioned itself as a strong contender.

Interpretability analysis

In the context of multi-trait genomic prediction, interpretability of models is becoming an increasingly important issue for understanding the influence of specific genomic features on model predictions [45, 46]. Feature selection of input variables provides a straightforward interprebablity of the evaluated models in this study. Compared with other methods, LassoNet offers a high interpretability due to its use of L_1 regularization (Lasso penalty), which by its inherent sparsity operation identify which input features are most relevant for making predictions (139 on the mice data, 952 on the pig data, 55 on the wheat data, 764 on the 14-cancer microarray data, and 99 on the Loblolly pine data). By selecting a subset of features, LassoNet can focus on the most influential variables while ignoring irrelevant or redundant ones. LightGBM can also provide interpretable results in terms of feature importance and decision paths within the trees. However, LightGBM does not rely on L_1 regularization for interpretation, but instead provides two different types of feature importance: split and gain importance. On the other side, the tabular deep learning methods lack the inherent interpretabily due to their more complex neural network structures. This complexity enables them to capture the patterns present in genomic data but also make interpretation and understanding of model decision more challenging. We only visualize the selected coefficients for the five datasets using LassoNet due to the superiority of this model. Figures 1 and 2 shows the selected coefficients for multi-trait regression and multi-class classification over five datasets.

Computation time

To improve the computing efficiency, we use parallel computing with 5 GPUs for the different models. The time required for one BO iteration of the models are summarized in Table 3. We can clearly see that LightGBM consistently outperforms other models, achieving the fastest execution times across all datasets. For instance, LightGBM completes one Bayesian Optimization (BO) iteration on the Mice dataset in only 4.9 ± 0.4 seconds (s), significantly faster than any other model. Similarly, it maintains a clear advantage on larger datasets such as Pig, Wheat, and 14-cancer microarray data, with times of 33.2 ± 3.5, 25 ± 2.6, and 8.4 ± 0.52 s, respectively. In contrast, NODE and Tab-R exhibit the slowest performance, particularly on the Pig and Wheat datasets where NODE takes 207 ± 11.7 and 240 ± 14.3 s, respectively, and Tab-R requires 175 ± 10.7 and 271 ± 11.6 s. This indicates that these models are not optimized for computational speed, especially in a multi-GPU setting. LassoNet and AutoInt show competitive performance with LassoNet nearly matching LightGBM on the Loblolly pine dataset, finishing in 4.1 ± 1.1 s. However, they generally lag behind LightGBM on the other datasets. Models like TabNet (both unsupervised and semi-supervised), FT-Transformer, Tab-Transformer, and SAINT offer moderate performance, with execution times ranging between those of the fastest and slowest models. These models, while not as fast as LightGBM, are notably quicker than NODE and Tab-R, particularly on smaller datasets like the Mice and 14-cancer microarray data.

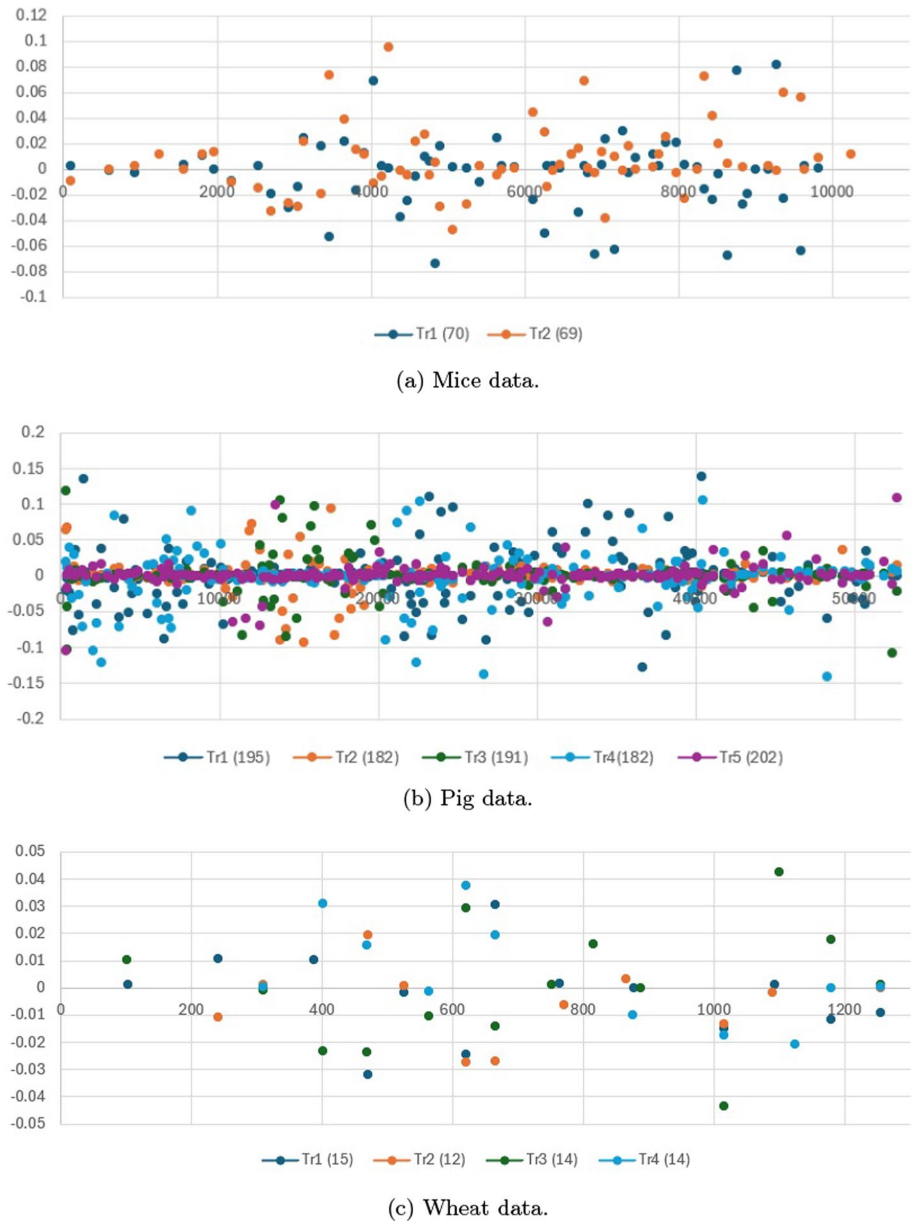
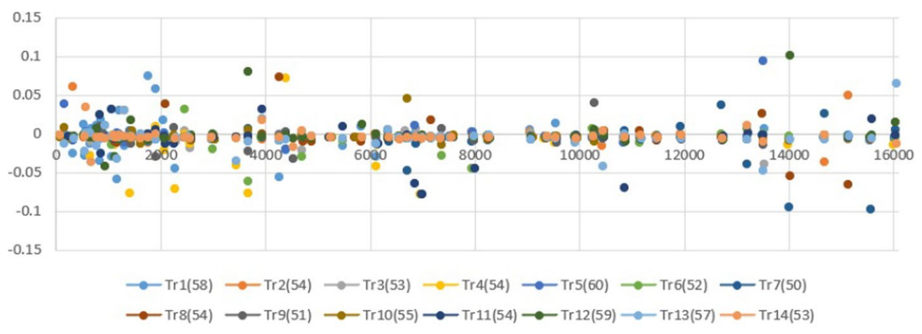


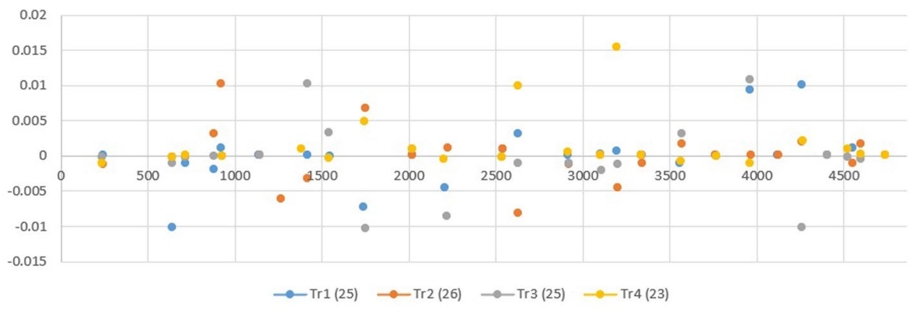
Fig. 1 Selected coefficients using LassoNet for multi-trait regression over three datasets, with the number of features selected on a specific trait noted in parentheses (zero values in the coefficient matrix have been removed)

Discussion

In one earlier study [47], we evaluated several linear sparse proximal multi-task learning (SPMTL) methods on the same genomic datasets as in the current study. The regularization methods there include Lasso, group Lasso, sparse group Lasso, nuclear norm, and the $L_{2,\frac{1}{2}}$ norm. Among the evaluated methods, the $L_{2,\frac{1}{2}}$ norm achieved the best out-of-sample prediction performance across all datasets with the smallest test MSE and the highest r on the mice data ($\mathrm{MSE} = 0.151, \mathrm{r} = 0.702$), pig data ($\mathrm{MSE} = 0.160, \mathrm{r} = 0.71$), and wheat data ($\mathrm{MSE} = 0.180, \mathrm{r} = 0.620$).



(a) 14-cancer microarray data.



(b) Loblolly pine data.

Fig. 2 Selected coefficients using LassoNet over two datasets for multi-class classification, with the number of features selected on a specific task noted in parentheses (zero values in the coefficient matrix have been removed)

Table 1 Comparison of model performance across different datasets for multi-trait regression tasks

	Mice data		**Pig data**		**Wheat data**	
Model	**MSE (mean ± stddev)**	**r**	**MSE (mean ± stddev)**	**r**	**MSE (mean ± stddev)**	**r**
LightGMB	0.208 ± 0.005	0.572	0.203 ± 0.006	0.652	0.160 ± 0.004	0.657
NODE	0.231 ± 0.008	0.501	0.242 ± 0.007	0.531	0.223 ± 0.006	0.514
LassoNet	**0.135 ± 0.003**	**0.735**	**0.151 ± 0.002**	**0.722**	**0.149 ± 0.003**	**0.691**
AutoInt	0.253 ± 0.007	0.571	0.267 ± 0.006	0.633	0.228 ± 0.005	0.641
TabNet (unsupervised)	0.301 ± 0.006	0.566	0.243 ± 0.004	0.621	0.218 ± 0.004	0.635
TabNet (supervised)	0.252 ± 0.008	0.554	0.225 ± 0.006	0.607	0.216 ± 0.005	0.626
FT-Transformer	0.221 ± 0.005	0.531	0.219 ± 0.004	0.585	0.206 ± 0.004	0.602
Tab-Transformer	0.223 ± 0.006	0.526	0.221 ± 0.005	0.563	0.209 ± 0.005	0.558
TabR	0.241 ± 0.009	0.503	0.230 ± 0.007	0.534	0.211 ± 0.006	0.541
GANDALF	0.211 ± 0.006	0.571	0.214 ± 0.005	0.637	0.191 ± 0.004	0.651
SAINT	0.224 ± 0.007	0.537	0.227 ± 0.006	0.581	0.210 ± 0.005	0.576

Bold indicates the best results

Compared with $L_{2,\frac{1}{2}}$ norm from SPMTL methods, the current study presents an obvious reduction in test MSE with LassoNet across three datasets: a decrease of 10.6% for the mice datasets, 5.7% for the pig data, and 17.2% for the wheat dataset, respectively. Moreover, employing Markov Chain Monte Carlo (MCMC)-based

Table 2 Comparison of different models for multi-class classification tasks on two datasets

Model	14-cancer microarray data			Loblolly pine data (subset)		
	Accuracy (mean ± stddev)	AUC (mean ± stddev)	Brier score	Accuracy (mean ± stddev)	AUC (mean ± stddev)	Brier score
LightGMB	85.7 ± 0.32	88.9 ± 0.171	0.202	91.5 ± 0.27	93.7 ± 0.201	0.206
NODE	85.1 ± 0.35	88.7 ± 0.237	0.243	91.1 ± 0.31	93.4 ± 0.223	0.301
LassoNet	**90.1 ± 0.17**	**91.5 ± 0.152**	**0.174**	**93.5 ± 0.19**	**96.7 ± 0.190**	**0.183**
AutoInt	85.2 ± 0.33	88.6 ± 0.263	0.381	91.5 ± 0.30	94.3 ± 0.207	0.372
TabNet (unsupervised)	81.2 ± 0.37	84.5 ± 0.205	0.406	90.3 ± 0.25	93.6 ± 0.233	0.383
TabNet (supervised)	82.3 ± 0.37	86.1 ± 0.207	0.355	90.1 ± 0.33	93.6 ± 0.246	0.361
FT-Transformer	84.7 ± 0.34	88.8 ± 0.181	0.235	90.8 ± 0.34	94.1 ± 0.252	0.245
Tab-Transformer	83.4 ± 0.34	88.5 ± 0.177	0.238	90.6 ± 0.32	94.0 ± 0.243	0.253
TabR	85.1 ± 0.35	88.6 ± 0.183	0.244	91.4 ± 0.27	93.5 ± 0.221	0.355
GANDALF	87.4 ± 0.33	89.7 ± 0.163	0.227	91.5 ± 0.26	95.1 ± 0.194	0.241
SAINT	84.6 ± 0.32	88.8 ± 0.175	0.241	90.7 ± 0.27	93.5 ± 0.185	0.263

Bold indicates the best results

Table 3 Time report in seconds (s) (mean ± stddev) of one BO iteration using 5 GPUs for different models on the five data sets

Model	Data sets				
	Mice data (s)	Pig data (s)	Wheat data (s)	14-cancer microarray data (s)	Loblolly pine data (subset) (s)
LightGBM	**4.9 ± 0.4**	**33.2 ± 3.5**	**25 ± 2.6**	**8.4 ± 0.52**	**4.1 ± 0.27**
NODE	46.3 ± 6.3	207 ± 11.7	240 ± 14.3	33.1 ± 3.7	9.1 ± 2.1
LassoNet	5.6 ± 1.1	36.1 ± 3.3	32 ± 3.6	8.8 ± 2.7	**4.1 ± 1.1**
AutoInt	9.3 ± 1.7	56.8 ± 4.4	53 ± 3.6	9.1 ± 2.2	4.4 ± 1.2
TabNet (unsupervised)	19.1 ± 5.1	75.5 ± 4.3	123 ± 9.1	10.9 ± 2.3	5.7 ± 1.7
TabNet (semi-supervised)	16.9 ± 5.4	67.3 ± 6.1	113 ± 7.7	10.1 ± 4.1	5.2 ± 1.3
FT-Transformer	14.1 ± 3.9	62.3 ± 5.6	147 ± 10.4	13.7 ± 4.0	6.8 ± 2.2
Tab-Transformer	16.3 ± 3.6	85.3 ± 4.8	93 ± 6.6	11.2 ± 5.3	5.9 ± 0.9
Tab-R	55.1 ± 5.5	175 ± 10.7	271 ± 11.6	35.4 ± 7.1	11.2 ± 2.3
GANDALF	16.5 ± 3.1	52.3 ± 6.2	98 ± 5.7	10.2 ± 2.3	5.0 ± 1.0
SAINT	15.3 ± 2.1	67.1 ± 4.2	97.3 ± 4.9	14.1 ± 3.7	7.5 ± 2.1

Bold indicates the best results

multi-task Bayesian ridge regression (BRR) and Bayesian spike-and-slab analysis further confirms the efficacy of LassoNet. Specially, the test MSE exhibits significant improvements with a reduction of 48.8% compared to BRR on the mice dataset, 50.3% compared to spike-and-slab on the pig dataset, and 43.5% compared to BRR on the wheat data. These results demonstrate that LassoNet is very competitive in comparison to both linear SPMTL methods, Bayesian GWP methods and other tabular neural network methods in performing joint predictions across correlated traits.

In [48], linear classifiers with eight different regularization methods were assessed on the microarray data that contains 14 cancer classes. These techniques included nearest shrunken centroids, L_2-penalized discriminant analysis, support vector classifier, Lasso regression (one vs all), k-nearest neighbors, L_1/L_2 penalized multinomial, and elastic-net penalized multinomial regression. Among these methods, the elastic-net penalized multinomial method achieved the highest accuracy of 78.1%. In contrast, LassoNet attained an accuracy of 90.1% on the same 14-cancer microarray data, clearly outperforming the evaluated linear classifiers employing different regularization approaches in facilitating joint predictions across multiple traits.

LassoNet can capture complex interaction effects between features via its neural network module. When input features are correlated or interact with each other in predicting the target phenotype, LassoNet assigns non-zero coefficients to these features to account for their joint effect. However, because of the model architecture and the regularization that encourage sparsity, LassoNet's ability to capture interactions is smaller compared to the interaction-based models like AutoInt, TabR and TabNet. Additionally, the evaluations indicate that some tabular deep learning models can achieve comparable or competitive results in classification tasks across our datasets (GANDALF). For the models based on transformers (AutoInt, TabTransformer, TabNet and SAINT), we can notice that a self-attention mechanism within the encoder can capture more complex interactions, but it doesn't improve results compared to LassoNet since they lack efficient regularization procedures.

Conclusion

In this paper, we provide an overview and comparison of recent deep learning architectures tailored for tabular data and their application to multi-trait genome-wide prediction (MTGWP). Through extensive evaluations on real-world genomic datasets, the study identifies LassoNet as a standout performer, surpassing decision tree methods like LightGBM and other tabular deep learning architectures in terms of both predictive accuracy and computing efficiency. Moreover, the inherent variable selection property of LassoNet provides a systematic way to find important genetic markers that contribute to phenotype expression.

Methods

Definitions

The key part of tabular deep learning is a deep neural network and usually in the form of a feed-forward network. A deep neural network defines a mapping $\hat{f}$ [7] following

$$Y = f(X) \approx \hat{f}(X; W), \tag{1}$$

that learns the model parameters W (i.e. the weights of a neural network) that results in the best approximation of true underlying and unknown function f. In this case, $X \in \mathcal{R}^p$ is the input feature data with corresponding output target $Y \in \mathcal{Z}^k$ for k classes or $Y \in \mathcal{R}^k$ for k regression tasks represented as a set of tuples $\{x_i, y_i\}_{i \in n}$. Throughout this study, we focus on genomic data X of dimension $n \times p$ with associated multiple traits Y of dimension $n \times k$ with the restriction that the all traits are of the same modality (i.e. either continuous or binary). However, the input data can be of different modalities (for example a mixture between DNA marker and RNA expression data) which fundamentally contrasts with the homogeneous nature of image, audio or text data. The network used in this work is feed-forward, which means that the input information flows in one direction to the output without any feedback connections [7]. Multi-task tabular deep learning involves training a model to simultaneously minimize a multivariate loss function $\mathcal{L}(\hat{f}(X; W), Y)$. The goal is to capitalize on possible shared representations across related tasks in order to enhance the model's performance.

Tabular neural networks

NODE

Inspired by CatBoost [13], the neural oblivious decision ensemble (NODE) [38] is designed to leverage the interpretability of decision trees while benefiting from the expressive prediction power of neural networks.

Each oblivious decision module in NODE consists of several layers of decision nodes. At each layer l, each node makes a decision based on a splitting feature g and a learnable threshold b. For the l-th layer and node j, the decision function can be expressed as

$$d_{l,j}(X) = \mathcal{H}(g_{l,j} - b_{l,j}), \tag{2}$$

where $\mathcal{H}(\cdot)$ denotes the Heaviside step function and $d_{l,j}(X)$ is the output of the decision node. Then all r decision nodes in the l-th layer will produce a decision vector $v_l = [d_{l,1}(X), d_{l,2}(X), \cdots, d_{l,r}(X)]$. NODE uses an ensemble of such decision trees through these output nodes.

To make the oblivious decision trees (ODTs) differentiable, the splitting feature choice $g_{l,j}$ and the comparison operator $\mathcal{H}(g_{l,j} - b_{l,j})$ are replaced by their differentiable counterparts. The choice function is replaced by a weighted sum of features, with weights computed based on the $\alpha-$ entmax transformation [49] over the learnable feature selection matrix. The Heaviside function $\mathcal{H}(\cdot)$ is relaxed to a two-class entmax, which is denoted as $\sigma_\alpha(v) = \text{entmax}_\alpha([v, 0])$. Its scaled version $c_{l,j}(X) = \sigma_\alpha(\frac{g_{l,j}(v) - b_{l,j}}{\tau_{l,j}})$ is used due to potential variations in feature scales by using learnable parameters. The computed $c_{l,j}(v)$ is combined into a "choice" tensor $C \in \mathcal{R}^{2^d}$. The final prediction is then computed as a weighted linear combination of response tensor entries R with weights from the entries of choice tensor C. Assume that the tree outputs are one-dimensional $\hat{h}(v)$ and each NODE layer contains several trees whose outputs are concatenated by m individual trees $[\hat{h}_1(v), \cdots, \hat{h}_m(v)]$. Then the NODE layer can be trained alone or within a complex structure, just like fully-connected layers that serve as input for the subsequent layers. Similar to DenseNet [50], this architecture is a sequence of l NODE layers, where each layer uses a concatenation of previous layers as its input. This aggregated output is then passed through a final fully connected layer to produce the final prediction $\hat{Y}$, with the structure of this final layer depending on the task (regression or classification).

TabR

TabR [39] is a feed-forward network incorporating a customized t-Nearest-Neighbors-like component in the middle layer to produce a better prediction. Its main idea is to utilize the self-attention mechanism of transformers to capture complex interactions between features in tabular data.

With the feature matrix X, a feed-forward retrieval-free network $f(X) = P(E(X))$ is first partitioned into two parts: an encode $E : X \to \mathcal{R}^{p'}$ part and a predictor $P : \mathcal{R}^{p'} \to P\hat{Y}$ part. To make the model incrementally retrieval-based, a retrieval module R in a residual branch is added after E, where $\tilde{X} \in \mathcal{R}^{p'}$ is the intermediate representation of the target object, $\{\tilde{x}_i\}_{i \in I_{cand}} \subset \mathcal{R}^{p'}$ are the intermediate representations of the candidates and $\{y_i\}_{i \in I_{cand}} \subset Y$ are the labels of the candidates.

The retrieval module R is defined in the spirit of k-nearest neighbors. For the target object's representation, the retrieval module takes the $x_{1,\ldots,t}$ nearest neighbors among the candidates $\tilde{x}_i$ according to the similarity module $\mathcal{S}$ and aggregates their values produced by the value module $\mathcal{V}$ with the definitions

$$\mathcal{S}(\tilde{X}, \tilde{x}_i) = W_Q(\tilde{X})^T W_K(\tilde{x}_i) \qquad \mathcal{V}(\tilde{X}, \tilde{x}_i, y_i) = W_V(\tilde{x}_i), \tag{3}$$

where W_Q, W_K, W_V are the weights for the corresponding transformation. They play a critical role in transforming inputs to better capturing the similarities between entries, contributing to the model's ability to learn complex patterns and relationships within the data. By adding context labels, the performance of the similarity $\mathcal{S}$ and the value module $\mathcal{V}$ can be improved. Finally, the formal complete description of TabR which implements the R module is

$$\mathcal{S}(\tilde{X}, \tilde{x}_i) = -\|t - t_i\|^2 \qquad \mathcal{V}(\tilde{X}, \tilde{x}_i, y_i) = W_{y_i} + O(t - t_i), \tag{4}$$

where $t = W_K(\tilde{X})$, $t_i = W_K(\tilde{x}_i)$and the operation of O is defined as $O(\cdot) = \text{LinearWithoutBias}(\text{Dropout}(\text{Relu}(\text{Linear}(\cdot))))$. The retrieval module R enriches the target object's representation by retrieving and processing relevant objects from the candidates. Finally, the predictor P makes a prediction.

TabNet

TabNet [37] combines the strengths of both tree-based methods and deep neural networks using a sequential attention mechanism. It emerges as a deep learning model embodying the feature selection principles of decision trees, with its encoder comprising a feature transformer, an attentive transformer, and feature masking.

The features in X will be the input to a batch normalization layer which yields $X' \in \mathcal{R}^{w \times z}$, where w is the batch size and z is the dimension. Assume the number of hidden layers is l, the output of the feature transformer then becomes $w \times l$ which is split into two parts to construct a gated linear unit: a standard decision step $\rho[i] \in \mathcal{R}^{w \times l_a}$ and a shared across decision step $a[i] \in \mathcal{R}^{w \times l_\rho}$. The former is used for the final output of TabNet, and the latter is used as an input of the attentive transformer. Each block is composed of a fully-connected (FC) layer, batch normalization (BN) and a gated linear unit (GLU). For the attentive transformer, the main function is to get a learnable mask layer $M[j] \in \mathcal{R}^{w \times z}$ according to

$$M[j] = \text{sparsemax}(P[j-1] \cdot \gamma_i(a[j-1])), \tag{5}$$

where $\gamma_i(a[i-1]))$ is from the FC to BN, and *sparsemax* is a mapping from the vector to a simplex that obtains sparsity. The scaling prior - $P[j\text{-}1]$ has a close connection to the mask $M[j]$ via the features used in previous steps and one can notice that the initial value of $P[0]$ equals 1. To ensure the sparsity of $M[j]$, a regularized constraint is given to the parameters to make the distribution of $M[j]$ more reasonable.

All the output from the earlier steps are summed to give the final output through the FC layer. For the multi-task learning, each task-specific branch ends with an output layer that produces a scalar output for the regression task. The shared layers can facilitate the extraction of relevant features for multi-task learning and the task-specific branches capture patterns specific to each multi-task regression target. The TabNet decoder is composed of a feature transformer block at each step. After reconstructing the features from the encoded representation, the aggregated features will be passed through a fully connected layer to do the predictions $\hat{Y}$.

TabTransformer

Considering the characteristics of context embeddings, TabTransformer [42] is built upon self-attention based transformers. This model comprises a column embedding layer, a stack of l Transformer layers, and a multi-layer perceptron. Each Transformer layer consists of a multi-head self-attention layer followed by a position-wise feed-forward layer. For the tuples $\{x_i, y_i\}_{i\in n}$, each of the x_i is embedded into a parametric embedding of dimension s using column embedding. Let $e_{\phi_i}(x_i) \in \mathcal{R}^s$ be the embedding of the x_i feature, and $E_\phi(x_{cat}) = \{e_{\phi_1}(x_1), \cdots, e_{\phi_{s'}}(x_{s'})\}$ is the set of embeddings for all the categorical features. Then $E_\phi(x_{cat})$ serves as input to the sequential Transformer layers f_θ, which operate on parametric embeddings and return the corresponding contextual embeddings $h_{s'}$ where $h \in \mathcal{R}^s$. These contextual embeddings are concatenated along with the features to first form a vector which serves as the input to an MLP that is used to predict the target $\hat{Y}$.

A self-attention layer in TabTransformer comprises three parametric matrices - Key (K), Query (Q) and Value (V). Each input embedding is projected on to these matrices to generate the corresponding vectors and attends to all other embeddings through an attention head, which is computed as $\text{Attention}(K, Q, V) = A \cdot V$, where $A = \text{softmax}((QK^T)/\sqrt{k'})$($k'$ is the dimension of Key). The output of the attention head is projected back to the embedding through a FC, which in turn is passed through two position-wise feed-forward layers. The contextual embeddings are concatenated to form the feature x_{cont}. If we let δ be the cross-entropy for classification and the mean square error for regression tasks, the prediction $\hat{Y}$ can be obtained by minimizing the loss function $\mathcal{L}(\hat{f}(X; W), Y) = \delta(MLP(\text{Transformer}(E_\phi(x_{cat})), x_{cont}), Y)$.

FT-Transformer

FT-Transformer performs feature transformations that enhance the model's ability to capture complex patterns [43]. It handles individual features independently before combining them to make predictions. There are two important parts of FT-Transformer: the feature tokenizer and the transformer.

The feature Tokenizer component first transforms the input feature X to embeddings $G \in \mathcal{R}^{m' \times n'}$. The embedding for the feature x_i is computed as

$$G_i = f_i(x_i) + b_i, \tag{6}$$

where b_i is the i-th feature bias, f_i is implemented as the element-wise multiplication with the weight matrix W_i. There is also a function $f_i^{(cat)}$ implemented as a lookup table $W_i^{(cat)}$ for categorical features with one-hot vectors of the corresponding categorical features e_i^T. Then, the vectors are stacked as $G = stack[G_1, \cdots, G_i, G_1^{(cat)}, \cdots, G_i^{(cat)}]$ and the embedding of the [CLS] token (or "output token") is appended to the G and l Transformer layers $F_1, F_2, \cdots, F_l$ as

$$G_0 = \text{stack}[[CLS], G] \qquad G_i = F_i(G_{i-1}). \tag{7}$$

After using the PreNorm setting, the final representation of the [CLS] token is used for prediction. For the multi-task learning situation, the initial layer of FT-Transformer consists of a shared transform encoder that will process the input feature that are propagated as the task-specific heads for each regression task. These heads are small MLPs that take the output of the shared encoder and generate task-specific predictions. Denoting the final representation of the $[CLS]$ token as $G_l^{[CLS]}$, then the prediction is $\hat{Y} = f(G_l^{[CLS]}; W)$.

AutoInt

Given the limited ability of shallow networks to model interactions, AutoInt [40] is designed based on transformer mechanisms that enhance the modelling capabilities for feature interactions. The main idea of AutoInt is mapping of the original features to sparse low dimensions and modeling of the interactions among the high-order features.

With an embedding vector υ_i for field i, the original feature x_i is embedded into low dense vectors through the embedding layer as $\sigma_i = \upsilon_i x_i$. The output of the embedding layer is a concatenation of multiple embedding vectors, which are the input of an interaction layer. For the following interaction layer, a multi-head mechanism is utilized to map the feature into multiple subspaces and generate the different feature interaction pattern in these spaces. Further on, more high-order interactions will be produced through stacking of interaction layers. For the feature σ_i in attention space I, there are three vectors: W_Q for query, W_K for key, and W_V for value, respectively. The similarity between the feature σ_i and feature σ_j is first obtained as $\phi^I =< W_Q^I \sigma_i, W_K^I \sigma_j >$, and then the distribution of the attention is produced using softmax. With a weighted sum, the new feature of σ_i can be acquired as $\hat{\sigma}_i{}^I$.

For multiple attention spaces, the new feature from each space can be concatenated to get the final representation of σ_i as $\hat{\sigma}_i$. To preserve the learned combinatorial features, including raw individual features, a standard connection is added to the network

$$\sigma_i^{Res} = \text{ReLU}(\hat{\sigma}_i + W_{Res}\sigma_i), \tag{8}$$

where W_{Res} is the projection matrix and $\text{ReLU}(z) = \max(0, z)$ is the standard non-linear activation function. Thus, the representation of each feature σ_i is updated into a new representation σ_i^{Res}. By stacking multiple such layers, an arbitrary order of $\hat{\sigma}_i$ can be modeled. The output of the interacting layer is a set of feature vectors $\{\sigma_i^{Res}\}_{i=1}^{p}$. By concatenating all of the learned feature interactions, the aggregated representation will be passed through a final layer for the predictions of $\hat{Y}$.

LassoNet

LassoNet is based on the sparsity idea of the Lasso and achieves feature sparsity by allowing a feature to participate in a hidden unit only if its input connection is active [35]. The features X and a residual feed-forward neural network $\mathcal{F}$ with an arbitrary width and depth [51] can be described as

$$\mathcal{F} = \{\hat{f} \equiv \hat{f}_{\theta,W} : X \mapsto \theta^T X + g_W X\}, \tag{9}$$

where g_W is a feed-forward network with wights W (fully connected). The object function of LassoNet for multi-task learning is

$$\arg\min_{\theta,W} \mathcal{L}(\theta, W) + \lambda\|\theta\|_1 \quad s.t. \quad \left\|W_i^{(1)}\right\|_\infty \leq \nu|\theta_i|, i = 1, \cdots, p, \tag{10}$$

where $\mathcal{L}(\theta, W)$ is the loss on the training data set, and W_i^1 denotes the weights for feature i in the first hidden layer. The constraint means that the total amount of non-linearity involving feature i according to the relative effect importance of x_i as a main effect. The residual link and the first hidden layer jointly pass through a hierarchical soft-thresholding optimizer $\mathcal{S}(x) = sign(x) \cdot max\{|x| - \lambda, 0\}$. For the multi-task learning, the layers of the neural network remain common across all tasks to capture shared representations. The sparsity of the input layer weights gives complete control of the feature sparsity of the network. When $\nu = 0$, all the hidden units are inactive and only the skip connection remains which means that the formulation recovers exactly the Lasso. On the other hand, when $\nu \to \infty$, one recovers a standard unregularized feed-forward neural network. The linear and nonlinear components are optimized jointly to capture arbitrary nonlinearity.

GANDALF

Inspired by gated recurrent units (GRUs) [52] for representation learning, the gated adaptive network for deep automated learning of features (GANDALF) is designed for tabular data based on a gating mechanism and in-built feature selection called Gated Feature Learning Unit (GFLU) [41].

A learnable mask $M_n \in \mathcal{R}^p$ is used for the soft sparse selection of important features for each stage n of feature learning in GFLU. The mask is constructed by applying a sparse transformation on a learnable parameter vector $\Im_n \in \mathcal{R}^p$ combined with t-softmax activation for encouraging sparsity selection [53]. Here, let X_n be the input features and $M_n = t\text{-softmax}(\Im_n)$ the mask, the feature selection can be defined by

$$X_n = M_n \odot X \tag{11}$$

$$M_n = t\text{-softmax}(\Im_n, t), \tag{12}$$

where $\Im_n$ and t are learnable parameters and $\odot$ denotes an element-wise multiplication operation. The weight matrix W depends on the value of t. The gating mechanism has a reset gate r_n and an update gate z_n. The update gate decides how much information to update in its internal feature representation, which can be defined as

$$z_n = \sigma(W_n^z \cdot [\varphi_{n-1}; x_n]), \tag{13}$$

where φ_{n-1} is the $(n-1)$-th stage of the GFLU and W_n^z is a learnable parameter for the weight at stage n. Then the candidate feature representation $\hat{\varphi}_n$ is computed as

$$\hat{\varphi}_n = tanh(W_n^O \cdot [r_n \odot \varphi_{n-1}; X]), \tag{14}$$

where r_n decides how much information to forget from the previous feature representation, [] represents a concatenation operation, and W_n^O represents a learnable parameter. The reset gate can be computed in a similar way as the update gate: $r - n = \sigma(W_n^r \cdot [\varphi_{n-1}; X_n])$.

GANDALF can be viewed as a stack of GFLUs arranged in a sequence mannerthat at each stage n selects a subset of features and learns a representations of features and therefore multiple stages act in a hierarchical way to built up the optimal representation for the prediction task. Then this representation is fed to a multi-layer perceptron for the final prediction.

SAINT

SAINT (self-attention and intersample attention transformer) [44] is inspired by the transformer encoder, where the model takes in a sequence of feature embeddings and outputs contextual representations of the same dimension. Its main idea is to leverage several mechanisms to overcome the difficulties of training on tabular data. For the embedding layer, each feature in the input row is embedded into a e-dimensional space as

$$\mathbf{E} = \text{Embedding}(X), \tag{15}$$

where $\mathbf{E} \in \mathbb{R}^{n \times p \times e}$, and e is the embedding dimension. In the stacking of L identical stages, each stage consists of one self-attention transformer block and one intersample attention transformer block. The contextual representations of the input of batch b can be given as $\{\mathbf{r_i}\}_{i=1}^b = \mathbf{S}(\{\mathbf{E}(x_i)\}_{i=1}^b)$. When $L = 1$, $\mathbf{r_i}$ can be obtained as the following procedure

$$\begin{aligned}
\mathbf{z}_i^{(1)} &= \text{LN}(\text{MSA}(\mathbf{E}(x_i))) + \mathbf{E}(x_i) \\
\mathbf{z}_i^{(2)} &= \text{LN}(\text{FF}_1(\mathbf{z}_i^{(1)})) + \mathbf{z}_i^{(1)} \\
\mathbf{z}_i^{(3)} &= \text{LN}(\text{MISA}(\{\mathbf{z}_i^{(2)}\}_{i=1}^b)) + \mathbf{z}_i^{(2)'} \\
\mathbf{r}_i &= \text{LN}(\text{FF}_2(\mathbf{z}_i^{(3)})) + \mathbf{z}_i^{(3)}
\end{aligned} \tag{16}$$

where MSA is a multi-head self-attention layer with h heads, FF is a fully-connected feed-forward layer with a GELU non-linearity, LN is a normalization layer with skip connection and MISA is an intersample attention transformer block. For the intersample attention, it is computed across the different data points (i.e. rows of the tabular data matrix) in the batch. This can be helpful to improve the representation of a given data point by inspecting other points. For the self-supervised pretraining method, CutMix is used to augment samples in the input space and mixup is used in the embedding space for the augmented representation. At the final prediction stage, the corresponding embedding is passed through a single layer MLP with ReLU activation to get the output $\hat{Y}$.

Implementation details

Tuning

For each dataset, we tuned the hyperparameters of each model using Bayesian optimization (BO) with 100 iterations. The hyperparameter search was conducted on the validation folds of the training set, ensuring that the test set remained untouched and independent. To optimize the hyperparameters, we used a 5-fold cross-validation (CV) approach on the training set. For each fold, the model was trained on 4 folds and validated on the remaining fold. This process was repeated 5 times, with each fold serving as the validation set once. The performance metrics from the 5 folds were averaged to obtain a single performance measure for the given set of hyperparameters. Various combinations of hyperparameters were evaluated, and the set that provided the best average performance across the 5 folds was selected as the optimal set. Subsequently, we executed models in parallel across each fold and independently calculating the test MSE or test accuracy. The performance metric was then collected and averaged from this parallelized execution to facilitate the Bayesian Optimization process using Tree Parzen Estimator (TPE) for parameter suggestions. The best hyperparameters were selected based on the loss criteria (i.e. MSE or accuracy) of the validation set. This iterative process continued until the predefined stopping criterion was reached. For the TPE method, we relied on the stochasticity inherent in draws from the models, ensuring diverse candidate suggestions from one iteration to the next while incorporating new recommendations from BO [54]. To obtain a balance between time consumption and precision of the performance metric results, we set the BO stopping criterion to 1e-5. The experiments were conducted using 5 NVIDIA Tesla V100 GPUs. Each GPU is equipped with 32 GB of HBM2 memory. The initial parameter ranges of the hyperparameters of the models are public available online along with our code.

There are two important hyperparamters in LassoNet: the l_1-penalty coefficient λ and the hierarchy coefficient M, which control the complexity of the fitted model and the relative strength of the linear and nonlinear components, respectively. First, we performed some initial test runs to determine a suitable range of M and λ. For the λ, we made sure that the initial dense model with $\lambda = 0$ trained well before starting the regularization path. Then the stepsize over λ was implemented following the same strategy as the original paper.

Evaluation

For each tuned configuration, ensemble predictions were generated by conducting 10 experiments with different random seeds, and the average results are reported on the test set. For the multi-trait classification task, evaluation metrics include average classification accuracy with standard deviation (stddev), Bries scores and the area under the curve (AUC) with standard deviation. For the regression task, the metrics reported are the test mean squared error (MSE) with standard deviation and the Pearson correlation coefficient *r*, averaged across traits for each dataset.

Material

Mice data

The first data of our study is the mice data which is part of the BGLR package in R [55], but originally comes from the Wellcome Trust (http://gscan.well.ox.ac.uk) and has been used for whole-genome regression in several other studies [56, 57]. It consists of genotypes and phenotypes of 1,814 mice. Each mouse was genotyped at 10,346 single nucleotide polymorphisms (SNPs) that were coded as 0, 1 and 2. Here we use two continuous traits, body length (BL) and body mass index (BMI). The entire dataset was divided into a training set (70%), a validation set (10%) and an independent test set (20%).

Pig data

The largest tabular data set in our study is the pig data [58] which contains 3534 individuals with high-density genotypes and continuous phenotypes of five anonymized traits. After cleaning some missing data, we finally obtain 2314 samples and each sample contains 52, 843 SNPs. The data was anonymised by randomising the map order and cording of the SNP genotypes were 0, 1, and 2. The dataset was partitioned into three subsets for training, validation and independent testing using the same approach as for the mice data.

Wheat data

The wheat data set originates from CIMMYT's Global Wheat Program and is also a part of the BGLR package [55]. It comprises 599 wheat lines from the CIMMYT Global Wheat Program evaluated in four international environments representing four basic agroclimatic regions (mega-environments). The wheat lines were genotyped using 1,447 Diversity Array Technology (DArT) markers. As a quality control, all the markers with a minor allele frequency below 0.05 were eliminated, and any missing genotypes were imputed using samples from the marginal distribution of marker genotypes. Following these procedures, the dataset was reduced to 1,279 DArT markers which are coded as 0 and 1. We used the data from the different environments as multiple traits, resulting in a total of four traits. The whole datasets was divided into three datasets for training, validation and testing following the procedure of the pig and mice data.

14-Cancer microarray data

The cancer data originates from a study by [59] and has been used in for example [48]. The data uses oligonucleotide microarrays containing 16,063 oligonucleotide probe sets for the gene expression. It contains 16,063 gene expression feature values and 198 tumor samples, which were divided into 144 training samples and 54 test samples. Of the training samples, 100 were used for training and 44 allocated for validation. Each feature represents the expression level of a specific gene across various samples. The binary traits constitutes 14 common human cancers, including Breast (BR), Prostate (PR), Lung

(LU), Leukemia (LE), Renal (RE), Pancreas (PA), Ovarym (OV), Mesothelioma (ME) and CNS cancers. For further details see [59].

Loblolly pine data

The lobolly pine population is derived from 32 parent trees representing a wide range of accessions from the Atlantic coastal plain, Florida, and the lower Gulf of the United States. Parents were crossed in a circular mating design with additional off-diagnal crosses, resulting in 70 full-sib families with an average of 13.5 individuals per family [60, 61]. It was originally composed of 951 individuals from 61 families that was genotyped using an Illumina Infinium assay [62]. A subset of 4,853 SNPs (encoded as 0, 1, 2) were polymorphic and used in our study. By discretizing the values of the deregressed breeding values > 0 to 1 and deregressed breeding values < 0 to 0, we recreated two binary traits: presence or absence of rust (Rustbin) and presence or absence of roots (Rootnumbin). Then the traits were recoded to four multi-classes ([0,0],[0,1],[1,0] and [1,1]). After cleaning some missing data, we finally got 806 samples and each sample contains 4,853 SNPs. The dataset was divided into three subsets-training, validation and testing - using the same percentage as for the mice, pig and wheat data.

Acknowledgements
Not applicable.

Author contributions
Y.F. performed the analyses. Y.F. and P.W. wrote the main text and reviewed the manuscript.

Funding
Open Access funding provided by University of Oulu (including Oulu University Hospital). This work was supported by the University of Oulu & The Research Council of Finland Profi5/HiDyn funding for mathematics and AI: data insight for high dimensional dynamics [Grant 326291].

Availability of data and materials
The code for the tabular deep learning models and the datasets are available at https://github.com/angelYHF/Tabular-deep-learning-for-GWP. The original data sets are available at: Mice data: https://cran.r-project.org/web/packages/BGLR/index.html Pig data: https://academic.oup.com/g3journal/article/2/4/429/6026060/4294_ FileS1.zip Wheat data: https://cran.r-project.org/web/packages/BGLR/index.html 14-cancer microarray data: https://www.kaggle.com/datasets/tranthinhuy/14cancermicroarraydata Loblolly pine data: https://academic.oup.com/genetics/article/190/4/1503/6064084/Loblolly_Pine_Resende_.zip.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 6 June 2024 Accepted: 19 September 2024
Published online: 04 October 2024

References

1. Meuwissen TH, Hayes BJ, Goddard M. Prediction of total genetic value using genome-wide dense marker maps. Genetics. 2001;157(4):1819–29.
2. Jubair S, Domaratzki M. Crop genomic selection with deep learning and environmental data: a survey. Front Artif Intell. 2023;5:1040295.
3. Abraham G, Inouye M. Genomic risk prediction of complex human disease and its clinical application. Curr Opin Genet Dev. 2015;33:10–6.

4. Lee SH, Van Der Werf JH, Hayes BJ, Goddard ME, Visscher PM. Predicting unobserved phenotypes for complex traits from whole-genome SNP data. PLoS Genet. 2008;4(10):1000231.
5. Waldmann P, Ferenčaković M, Mészáros G, Khayatzadeh N, Curik I, Sölkner J. AUTALASSO: an automatic adaptive LASSO for genome-wide prediction. BMC Bioinform. 2019;20:1–10.
6. Momen M, Mehrgardi AA, Sheikhi A, Kranis A, Tusell L, Morota G, Rosa GJ, Gianola D. Predictive ability of genome-assisted statistical models under various forms of gene action. Sci Rep. 2018;8(1):12309.
7. Goodfellow I, Bengio Y, Courville A. Deep learning. Cambridge, MA, USA: MIT Press; 2016.
8. Ryan M. Deep learning with structured data. New York: Manning Publications; 2020.
9. Borisov V, Leemann T, Seßler K, Haug J, Pawelczyk M, Kasneci G. Deep neural networks and tabular data: a survey. IEEE Trans Neural Netw Learn Syst. 2024;30:7499–519.
10. Breiman L. Random forests. Mach Learn. 2001;45:5–32.
11. Chen T, Guestrin C. Xgboost: a scalable tree boosting system. In: Proceedings of the 22nd Acm Sigkdd international conference on knowledge discovery and data mining; 2016, p. 785–94
12. Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu T-Y. Lightgbm: a highly efficient gradient boosting decision tree. Adv Neural Inf Process Syst. 2017;30
13. Prokhorenkova L, Gusev G, Vorobev A, Dorogush AV, Gulin A. CatBoost: unbiased boosting with categorical features. Adv Neural Inf Process Syst. 2018;31
14. Wang Y, Feng D, Li D, Chen X, Zhao Y, Niu X. A mobile recommendation system based on logistic regression and gradient boosting decision trees. In: 2016 International joint conference on neural networks (IJCNN); 2016, p. 1896–902
15. Wen H, Zhang J, Lin Q, Yang K, Huang P. Multi-level deep cascade trees for conversion rate prediction in recommendation system. In: Proceedings of the AAAI conference on artificial intelligence; 2019, vol. 33, p. 338–45
16. Roe BP, Yang H-J, Zhu J, Liu Y, Stancu I, McGregor G. Boosted decision trees as an alternative to artificial neural networks for particle identification. Nucl Instrum Methods Phys Res Sect A Accel Spectrometers Detect Assoc Equip. 2005;543(2–3):577–84.
17. Bondarenko I. More layers! End-to-end regression and uncertainty on tabular data with deep learning. arXiv preprint arXiv:2112.03566 (2021)
18. Li Y, Shi W, Wasserman WW. Genome-wide prediction of cis-regulatory regions using supervised deep learning methods. BMC Bioinform. 2018;19:202.
19. Gianola D, Okut H, Weigel KA, Rosa GJM. Predicting complex quantitative traits with Bayesian neural networks: a case study with Jersey cows and wheat. BMC Genet. 2011;12:1–14.
20. Pérez-Rodríguez P, Gianola D, González-Camacho JM, Crossa J, Manès Y, Dreisigacker S. Comparison between linear and non-parametric regression models for genome-enabled prediction in wheat. G3: Genes Genomes Genet. 2012;2(12):1595–605.
21. Montesinos-López OA, Montesinos-López A, Pérez-Rodríguez P, Barrón-López JA, Martini JW, Fajardo-Flores SB, Gaytan-Lugo LS, Santana-Mancilla PC, Crossa J. A review of deep learning applications for genomic selection. BMC Genom. 2021;22:1–23.
22. Lee H-J, Lee JH, Gondro C, Koh YJ, Lee SH. DeepGBLUP: joint deep learning networks and GBLUP framework for accurate genomic prediction of complex traits in Korean native cattle. Genet Sel Evol. 2023;55(1):56.
23. Bellot P, Los Campos G, Pérez-Enciso M. Can deep learning improve genomic prediction of complex human traits? Genetics. 2018;210(3):809–19.
24. Waldmann P, Pfeiffer C, Mészáros G. Sparse convolutional neural networks for genome-wide prediction. Front Genet. 2020;11: 499643.
25. Ma W, Qiu Z, Song J, Li J, Cheng Q, Zhai J, Ma C. A deep convolutional neural network approach for predicting phenotypes from genotypes. Planta. 2018;248:1307–18.
26. Sandhu K, Patil SS, Pumphrey M, Carter A. Multitrait machine-and deep-learning models for genomic selection using spectral information in a wheat breeding program. Plant Genome. 2021;14(3):20119.
27. Zhang Q, Sun T, Wang J. Genome-wide association study and high-quality gene mining related to soybean protein and fat. BMC Genom. 2023;24(1):596.
28. Khaki S, Wang L, Archontoulis SV. A CNN-RNN framework for crop yield prediction. Front Plant Sci. 2020;10: 492736.
29. Nevavuori P, Narra N, Lipping T. Crop yield prediction with deep convolutional neural networks. Comput Electron Agric. 2019;163: 104859.
30. Azodi CB, Pardo J, VanBuren R, Los Campos G, Shiu S-H. Transcriptome-based prediction of complex traits in maize. Plant Cell. 2020;32(1):139–51.
31. Fu Y, Xu J, Tang Z, Wang L, Yin D, Fan Y, Zhang D, Deng F, Zhang Y, Zhang H, et al. A gene prioritization method based on a swine multi-omics knowledgebase and a deep learning model. Commun Biol. 2020;3(1):502.
32. Hu H, Campbell MT, Yeats TH, Zheng X, Runcie DE, Covarrubias-Pazaran G, Broeckling C, Yao L, Caffe-Treml M, Gutiérrez L, et al. Multi-omics prediction of oat agronomic and seed nutritional traits across environments and in distantly related populations. Theor Appl Genet. 2021;134:4043–54.
33. Khurana E, Fu Y, Chen J, Gerstein M. Interpretation of genomic variants using a unified biological network approach. PLoS Comput Biol. 2013;9(3):1002886.
34. Jubair S, Tucker JR, Henderson N, Hiebert CW, Badea A, Domaratzki M, Fernando WD. GPTransformer: a transformer-based deep learning method for predicting Fusarium related traits in barley. Front Plant Sci. 2021;12: 761402.
35. Lemhadri I, Ruan F, Abraham L, Tibshirani R. Lassonet: a neural network with feature sparsity. J Mach Learn Res. 2021;22(127):1–29.
36. Mathew B, Hauptmann A, Léon J, Sillanpää MJ. NeuralLasso: neural networks meet lasso in genomic prediction. Front Plant Sci. 2022;13: 800161.
37. Arik S.Ö, Pfister T. Tabnet: attentive interpretable tabular learning. In: Proceedings of the AAAI conference on artificial intelligence; 2021, vol. 35, p. 6679–87
38. Popov S, Morozov S, Babenko A. Neural oblivious decision ensembles for deep learning on tabular data. arXiv preprint arXiv:1909.06312 (2019)

39. Gorishniy Y, Rubachev I, Kartashev N, Shlenskii D, Kotelnikov A, Babenko A. Tabr: unlocking the power of retrieval-augmented tabular deep learning. arXiv preprint arXiv:2307.14338 (2023)
40. Song W, Shi C, Xiao Z, Duan Z, Xu Y, Zhang M, Tang J. Autoint: automatic feature interaction learning via self-attentive neural networks. In: Proceedings of the 28th ACM international conference on information and knowledge management; 2019, p. 1161–70
41. Joseph M, Raj H. GANDALF: gated adaptive network for deep automated learning of features. arXiv preprint arXiv:2207.08548 (2024)
42. Huang X, Khetan A, Cvitkovic M, Karnin Z. Tabtransformer: tabular data modeling using contextual embeddings. arXiv preprint arXiv:2012.06678 (2020)
43. Gorishniy Y, Rubachev I, Khrulkov V, Babenko A. Revisiting deep learning models for tabular data. Adv Neural Inf Process Syst. 2021;34:18932–43.
44. Somepalli G, Goldblum M, Schwarzschild A, Bruss CB, Goldstein T. Saint: improved neural networks for tabular data via row attention and contrastive pre-training. arXiv preprint arXiv:2106.01342 (2021)
45. Watson D. Interpretable machine learning for genomics. Hum Genet. 2022;141:1499–513.
46. Conard AM, DenAdel A, Crawford L. A spectrum of explainable and interpretable machine learning approaches for genomic studies. WIREs Comput Stat. 2023;15:1617.
47. Fan Y, Launonen I, Sillanpää MJ, Waldmann P. Evaluation of sparse proximal multi-task learning for genome-wide prediction. IEEE Access. 2024;12:51665–75.
48. Hastie T, Tibshirani R, Friedman JH, Friedman JH. The elements of statistical learning: data mining, inference, and prediction, vol. 2. New York: Springer; 2009.
49. Peters B, Niculae V, Martins AF. Sparse sequence-to-sequence models. arXiv preprint arXiv:1905.05702 (2019)
50. Huang G, Liu Z, Van Der Maaten L, Weinberger K.Q. Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2017, p. 4700–8
51. He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016, p. 770–8
52. Cho K, Van Merriënboer B, Bahdanau D, Bengio Y. On the properties of neural machine translation: encoder-decoder approaches. arXiv preprint arXiv:1409.1259 (2014)
53. Zhao S, Liang Z, Wen J, Chen J. Sparsing and smoothing for the seq2seq models. IEEE Trans Artif Intell. 2022
54. Bergstra J, Yamins D, Cox DD, et al. Hyperopt: a Python library for optimizing the hyperparameters of machine learning algorithms. SciPy. 2013;13:20.
55. Pérez P, Los Campos G. Genome-wide regression and prediction with the BGLR statistical package. Genetics. 2014;198(2):483–95.
56. Legarra A, Robert-Granié C, Manfredi E, Elsen J-M. Performance of genomic selection in mice. Genetics. 2008;180(1):611–8.
57. Okut H, Gianola D, Rosa GJ, Weigel KA. Prediction of body mass index in mice using dense molecular markers and a regularized neural network. Genet Res. 2011;93(3):189–201.
58. Cleveland MA, Hickey JM, Forni S. A common dataset for genomic analysis of livestock populations. G3: Genes Genomes Genet. 2012;2(4):429–35.
59. Ramaswamy S, Tamayo P, Rifkin R, Mukherjee S, Yeang C-H, Angelo M, Ladd C, Reich M, Latulippe E, Mesirov JP, et al. Multiclass cancer diagnosis using tumor gene expression signatures. Proc Natl Acad Sci. 2001;98(26):15149–54.
60. Resende M Jr, Munoz P, Resende MD, Garrick DJ, Fernando RL, Davis JM, Jokela EJ, Martin TA, Peter GF, Kirst M. Accuracy of genomic selection methods in a standard data set of loblolly pine (Pinus taeda L.). Genetics. 2012;190(4):1503–10.
61. Baltunis BS, Huber DA, White TL, Goldfarb B, Stelzer HE. Genetic analysis of early field growth of loblolly pine clones and seedlings from the same full-sib families. Can J For Res. 2006;37(1):195–205.
62. Eckert AJ, Van Heerwaarden J, Wegrzyn JL, Nelson CD, Ross-Ibarra J, González-Martínez SC, Neale DB. Patterns of population structure and environmental associations to aridity across the range of loblolly pine (Pinus taeda L., Pinaceae). Genetics. 2010;185(3):969–82.

Publisher's Note