HOPULAR: MODERN HOPFIELD NETWORKS FOR TABULAR DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

While Deep Learning excels in structured data as encountered in vision and natural language processing, it failed to meet its expectations on tabular data. In real world, however, most machine learning applications face tabular data with less than 10,000 samples. For tabular data, Support Vector Machines (SVMs), Random Forests, and Gradient Boosting are the best performing techniques, where Gradient Boosting has the lead. Recently, we saw a surge of Deep Learning methods that were tailored to tabular data. However, these methods still underperform compared to Gradient Boosting. We suggest "Hopular" to learn from tabular data with hundreds or thousands of samples. Hopular is a Deep Learning architecture, where each layer is equipped with continuous modern Hopfield networks. The modern Hopfield networks can store two types of data: (i) the whole training set and (ii) the feature embedding vectors of the actual input. The stored data allow the identification of feature-feature, feature-target, sample-sample, and sample-target dependencies. The stored training set enables to find similarities across input vectors and targets, while the stored actual input enables to determine dependencies between features and targets. Hopular's novelty is that the original training set and the original input are provided at each layer. Therefore, Hopular can improve the current prediction at every layer by re-accessing the original training set like standard iterative learning algorithms. In experiments on small-sized tabular datasets with less than 1,000 samples, Hopular surpasses Gradient Boosting, Random Forests, SVMs, and in particular several Deep Learning methods. In experiments on medium-sized tabular data with about 10,000 samples, Hopular outperforms XGBoost and a state-of-the art Deep Learning method designed for tabular data. Although Hopular needs more training time than Gradient Boosting, Random Forests, and SVMs, it is a strong alternative to these methods on small-sized and medium-sized tabular datasets as it yields higher performance.

1 INTRODUCTION

Deep Learning has led to tremendous success in vision and natural language processing, where it excelled on large image and text corpora (LeCun et al., 2015; Schmidhuber, 2015). On small tabular data, however, Deep Learning could not convince so far. In real world, small tabular datasets with less than 10,000 samples are ubiquitous. They are found in life sciences, when building a model for a certain disease with a limited number of patients, for bio-assays in drug design, or for the effect of environmental soil contamination. The same situation appears in most industrial applications, when a company wants to predict customer behaviour, to control processes, to optimize its logistics, to market new products, or to employ predictive maintenance. The omnipresence of small datasets can also be witnessed at Kaggle challenges. On small-sized and medium-sized tabular datasets with less than 10,000 samples, Support Vector Machines (SVMs) (Boser et al., 1992; Cortes & Vapnik, 1995; Schölkopf & Smola, 2002), Random Forests (RF) (Ho, 1995; Breiman, 2001) and, in particular, Gradient Boosted Decision Trees (GBDT) (Friedman, 2001) typically lead to higher performances than Deep Learning methods. GBDT have the edge over other methods on most small-sized and medium-sized tabular datasets. In real world applications, the best performing and most prevalent GBDT variants are XGBoost (Chen & Guestrin, 2016), CatBoost (Dorogush et al., 2017; Prokhorenkova et al., 2018), and LightGBM (Ke et al., 2017). For small-sized and medium-sized tabular data, these methods dominate real world applications.

Recently, research on extending Deep Learning methods to tabular data has been intensified. Some approaches to tabular data are only remotely related to Deep Learning. AutoGluon-Tabular stacks small neural networks for tabular data (Erickson et al., 2020). Neural Oblivious Decision Ensembles (NODE) generalizes ensembles of oblivious decision trees by hierarchical representation learning (Popov et al., 2019). NODE is a hybrid of differentiable decision trees and neural networks. DNF-Net builds neural structures corresponding to logical Boolean formulas in disjunctive normal forms, which enable localized decisions using small subsets of the features (Abutbul et al., 2020).

However, most research focused on adapting established Deep Learning techniques to tabular data. For tabular data, modifications to deep neural networks like introducing leaky gates or skip connections can improve their performance on tabular data (Fiedler, 2021). Even plain MLPs that are well-regularized work well on tabular data (Kadra et al., 2021). Different regularization coefficients to each weight improve the performance of Deep Learning architectures on tabular data (Shavitt & Segal, 2018). TabularNet consists of three modules (Du et al., 2021). First, it uses handcrafted cell-level feature extraction with a language model for textual data. Secondly, it uses both row and column-wise pooling via bidirectional gated recurrent units. Thirdly, a graph convolutional network captures dependencies between cells of the table.

Many approaches that adapt Deep Learning methods to tabular data use attention mechanisms from transformers (Vaswani et al., 2017) and BERT (Devlin et al., 2019). The TabTransformer learns contextual embeddings of categorical features (Huang et al., 2020). However continuous features are not fed into the self-attention module, therefore the feature-feature interaction is limited. The FT-Transformer maps features to tokens that are fed into a transformer (Gorishniy et al., 2021). The FT-Transformer performs well on tabular data but all considered datasets have more than 10,000 samples. TabNet uses an attentive transformer for sequential attention to predict masked features (Arik & Pfister, 2021). Therefore, TabNet does instance-wise feature selection, that is, can select the relevant features for each input differently. TabNet also utilizes feature masking for pre-training, which was very successful in natural language processing when pre-training the BERT model. Also semi-supervised learning has been proposed for tabular data using projection of the features and contrastive learning (Darabi et al., 2021). The contrastive loss is low if pairs of the same class have high similarity. Value Imputation and Mask Estimation (VIME) uses self- and semi-supervised learning of deep architectures for tabular data (Yoon et al., 2020). Like BERT, the network has to predict the values of the masked feature vectors, where the target is always masked. The success of BERT feature masking confirms that Deep Learning techniques must employ strong regularization to be successful on tabular data (Kadra et al., 2021). A multi-head self-attentive neural network for modeling feature-feature interactions was also used in AutoInt (Song et al., 2019). So far we mentioned work, where attention mechanisms extract feature-feature and feature-target relations. However, also inter-sample attention can be implemented, if the whole training set is given at the input. TabGNN uses a graph neural network for tabular data to model inter-sample relations (Guo et al., 2021). However, the authors focus on large tabular datasets with more than 40,000 samples. SAINT contains both self-attention and inter-sample attention and embeds both categorical and continuous features before feeding them into transformer modules (Somepalli et al., 2021). SAINT also uses self-supervised pre-training with a contrastive loss to minimize the difference between original and mixed samples. Non-Parametric Transformers (NPTs) also use feature self-attention and inter-sample attention (Kossen et al., 2021). The feature self-attention identifies dependencies between features, while inter-sample attention detects relations between samples. As in previous approaches, BERT masking is used during training, where the masked feature values and the target have to be predicted.

We suggest "Hopular" to learn from tabular data. Hopular is a Deep Learning architecture, where each layer is equipped with continuous modern Hopfield networks (Ramsauer et al., 2021; 2020; Widrich et al., 2020). Continuous modern Hopfield networks can store two types of data: (i) the whole training set or (ii) the feature embedding vectors of the actual input. Like SAINT and NPTs, Hopular can detect feature-feature, feature-target, sample-sample, and sample-target dependencies via modern Hopfield networks. In each layer, the stored training set enables similarity-, prototype-, or quantization-based learning methods like nearest neighbor. In each layer, the stored actual input enables the identification of dependencies between the features and the target. Consequently, the current prediction can be improved at every layer via direct access to both the training set and the actual input. Therefore, a pass through a Hopular model is similar to standard learning algorithms, which iteratively improve the current solution by re-accessing the training set. The number of

iterations is fixed by the number of layers in the Hopular architecture. As previous methods, Hopular uses a feature embedding and BERT masking, where masked features have to be predicted. Hopular is most closely related to SAINT (Somepalli et al., 2021) and Non-Parametric Transformers (NPTs) (Kossen et al., 2021), but in contrast to SAINT and NPTs, the whole training set and the actual input are provided via Hopfield networks at every layer of the network and not only at the input.

Recently, it was reported that Random Forests still outperforms standard Deep Learning techniques on tabular datasets with up to 10,000 samples (Xu et al., 2021). In (Shwartz-Ziv & Armon, 2021), the authors show that XGBoost outperforms various Deep Learning methods that are designed for tabular data on datasets that did not appear in the original papers. Therefore, we test Hopular on exactly those datasets to see whether it performs as well as XGBoost. Furthermore, we test Hopular on UCI datasets (Ramsauer et al., 2021; 2020; Klambauer et al., 2017; Wainberg et al., 2016; Fernández-Delgado et al., 2014). Hopular surpasses Gradient Boosting, Random Forests, and SVMs but also state-of-the-art Deep Learning approaches to tabular data like NPT.

2 Brief Review of Modern Hopfield Networks

We briefly review continuous modern Hopfield networks. Their main features are their property to retrieve stored patterns with only one update and their exponential storage capacity. Modern Hopfield networks have much higher storage capacity (Demircigil et al., 2017; Krotov & Hopfield, 2016) than classical Hopfield networks (Hopfield, 1982; 1984). These binary modern Hopfield networks have been generalized to continuous modern Hopfield networks that are differentiable and allow for gradient descent if embedded in Deep Learning architectures (Ramsauer et al., 2021; 2020).

We assume a set of patterns $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathbb{R}^d$ that are stacked as columns to the matrix $\boldsymbol{X} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N)$ and a state pattern (query) $\boldsymbol{\xi} \in \mathbb{R}^d$ that represents the current state. The largest norm of a stored pattern is $M = \max_i \|\boldsymbol{x}_i\|$. Continuous modern Hopfield networks with state $\boldsymbol{\xi}$ have the energy

$$\mathrm{E} \ = \ - \beta^{-1} \ \log \left(\sum_{i=1}^{N} \exp(\beta \boldsymbol{x}_i^T \boldsymbol{\xi}) \right) \ + \ \beta^{-1} \log N \ + \ \frac{1}{2} \, \boldsymbol{\xi}^T \boldsymbol{\xi} \ + \ \frac{1}{2} \, M^2 \ . \tag{1}$$

For energy E and state $\boldsymbol{\xi}$, the update rule

$$\boldsymbol{\xi}^{\mathrm{new}} \ = \ f(\boldsymbol{\xi}; \boldsymbol{X}, \beta) \ = \ \boldsymbol{X} \, \boldsymbol{p} \ = \ \boldsymbol{X} \, \mathrm{softmax}(\beta \boldsymbol{X}^T \boldsymbol{\xi}) \tag{2}$$

has been proven to converge globally to stationary points of the energy E, which are almost always local minima (Ramsauer et al., 2021; 2020). The update rule Eq. (2) is also the formula of the well-known transformer attention mechanism (Ramsauer et al., 2021; 2020), therefore Hopfield retrieval and transformer attention coincide.

The *separation* Δ_i of a pattern $\boldsymbol{x}_i$ is defined as its minimal dot product difference to any of the other patterns: $\Delta_i \ = \ \min_{j, j \neq i} \left(\boldsymbol{x}_i^T \boldsymbol{x}_i - \boldsymbol{x}_i^T \boldsymbol{x}_j \right)$. A pattern is *well-separated* from the data if $\Delta_i \geq \frac{2}{\beta N} + \frac{1}{\beta} \log \left(2(N-1) N \beta M^2 \right)$. If the patterns $\boldsymbol{x}_i$ are well separated, the iterate Eq. (2) converges to a fixed point close to a stored pattern. If some patterns are similar to one another and, therefore, not well separated, the update rule Eq. (2) converges to a fixed point close to the mean of the similar patterns. This fixed point is a *metastable state* of the energy function and averages over similar patterns.

The next theorem states that the update rule Eq. (2) typically converges after one update if the patterns are well separated. Furthermore, it states that the retrieval error is exponentially small in the separation Δ_i (proof see (Ramsauer et al., 2021; 2020)):

Theorem 1. *With query $\boldsymbol{\xi}$, after one update the distance of the new point $f(\boldsymbol{\xi})$ to the fixed point $\boldsymbol{x}_i^*$ is exponentially small in the separation Δ_i. The precise bounds using the Jacobian $\mathrm{J} = \frac{\partial f(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$ and its value J^m in the mean value theorem are:*

$$\| f(\boldsymbol{\xi}) \ - \ \boldsymbol{x}_i^* \| \ \leqslant \ \| \mathrm{J}^m \|_2 \ \| \boldsymbol{\xi} \ - \ \boldsymbol{x}_i^* \| \ , \tag{3}$$

$$\| \mathrm{J}^m \|_2 \ \leqslant \ 2 \, \beta \, N \, M^2 \, (N-1) \exp(- \, \beta \, (\Delta_i \ - \ 2 \, \max\{ \| \boldsymbol{\xi} \ - \ \boldsymbol{x}_i \|, \| \boldsymbol{x}_i^* \ - \ \boldsymbol{x}_i \| \} \, M)) \ . \tag{4}$$

For given ϵ and sufficient large Δ_i, we have $\| f(\boldsymbol{\xi}) \ - \ \boldsymbol{x}_i^* \| < \epsilon$, that is, retrieval with one update. The retrieval error $\| f(\boldsymbol{\xi}) \ - \ \boldsymbol{x}_i \|$ of pattern $\boldsymbol{x}_i$ is bounded by

$$\| f(\boldsymbol{\xi}) \ - \ \boldsymbol{x}_i \| \ \leqslant \ 2 \, (N-1) \ \exp(- \, \beta \, (\Delta_i \ - \ 2 \, \max\{ \| \boldsymbol{\xi} \ - \ \boldsymbol{x}_i \|, \| \boldsymbol{x}_i^* \ - \ \boldsymbol{x}_i \| \} \, M)) \, M \ . \tag{5}$$

The main requirement to modern Hopfield networks to be suited for tabular data is that they can store and retrieve enough patterns. We want to store a potentially large training set in every layer of a Deep Learning architecture. We first define what we mean by storing and retrieving patterns from a modern Hopfield network.

Definition 1 (Pattern Stored and Retrieved)**.** *We assume that around every pattern $\boldsymbol{x}_i$ a sphere S_i is given. We say $\boldsymbol{x}_i$* is stored *if there is a single fixed point $\boldsymbol{x}_i^* \in \mathrm{S}_i$ to which all points $\boldsymbol{\xi} \in \mathrm{S}_i$ converge, and $\mathrm{S}_i \cap \mathrm{S}_j = \emptyset$ for $i \neq j$. We say $\boldsymbol{x}_i$* is retrieved *for a given ϵ if iteration (update rule) Eq. (2) gives a point $\tilde{\boldsymbol{x}}_i$ that is at least ϵ-close to the single fixed point $\boldsymbol{x}_i^* \in \mathrm{S}_i$. The retrieval error is $\|\tilde{\boldsymbol{x}}_i - \boldsymbol{x}_i\|$.*

As with classical Hopfield networks, we consider patterns on the sphere, i.e. patterns with a fixed norm. For randomly chosen patterns, the number of patterns that can be stored is exponential in the dimension d of the space of the patterns (proof see (Ramsauer et al., 2021; 2020)):

Theorem 2. *We assume a failure probability $0 < p \leqslant 1$ and randomly chosen patterns on the sphere with radius $M := K\sqrt{d-1}$. We define $a := \frac{2}{d-1}(1 + \ln(2\beta K^2 p(d-1)))$, $b := \frac{2K^2\beta}{5}$, and $c := \frac{b}{W_0(\exp(a+\ln(b)))}$, where W_0 is the upper branch of the Lambert W function (Olver et al., 2010, (4.13)), and ensure $c \geq \left(\frac{2}{\sqrt{p}}\right)^{\frac{4}{d-1}}$. Then with probability $1-p$, the number of random patterns that can be stored is:*

$$N \;\geq\; \sqrt{p}\, c^{\frac{d-1}{4}} \;. \tag{6}$$

Therefore it is proven for $c \geq 3.1546$ with $\beta = 1$, $K = 3$, $d = 20$ and $p = 0.001$ ($a + \ln(b) > 1.27$) and proven for $c \geq 1.3718$ with $\beta = 1$, $K = 1$, $d = 75$, and $p = 0.001$ ($a + \ln(b) < -0.94$).

This theorem motivates to use continuous modern Hopfield networks for tabular data, where we want to store the training set in each layer of a Deep Learning architecture. Even for hundreds of thousands of training samples, the continuous modern Hopfield network is able to store the training set if the dimension of the pattern is large enough.

3 HOPULAR: MODERN HOPFIELD NETWORKS FOR TABULAR DATA

Hopular architecture. The Hopular architecture consists of a Hopular input layer, then several stacked Hopular Blocks, and a Hopular output layer. (i) The input layer of a Hopular architecture is a feature embedding layer. Categorical features are encoded as one-hot vectors, whereas continuous features are normalized to zero mean and unit variance. Then a linear mapping to an e-dimensional embedding space is applied. The index of a feature w.r.t. the position inside the sample as well as the feature type itself are conserved by separate e-dimensional learnable embeddings. All three embedding vectors are element-wise summed and serve as the final representation of an input feature. The input sample is represented by the concatenation of all of its feature representations. (ii) The last layer of a Hopular architecture is a sparsely-connected layer which maps the output of the last Hopular block to the output layer. (iii) In the following we explain the main architectural concept of Hopular, the Hopular Block.

Figure 1 depicts a Hopular Block. A Hopular Block consecutively applies two different Hopfield modules, where the results of each Hopfield module are combined with its respective input via a residual block before passing the results on. The sample representation before and after each Hopular Block is kept. The predicted target is at the position of the originally masked target and intermediate results are at the unmasked feature positions. Figure 1 illustrates the forward-pass of a single input sample with the masked target indicated by the question mark symbol (**?**). The masked target is transformed by the Hopular block to a corresponding prediction as indicated by a check mark symbol (✓). Also other input features can be masked and predicted as with BERT pre-training. In the following we describe the components (I)–(III) of a Hopular Block.

(I) Hopfield Module H_s. The first Hopfield module H_s implements a modern Hopfield network via the layer `HopfieldLayer` (Ramsauer et al., 2021; 2020) with the training set as fixed stored patterns. The current input (the state vector) $\boldsymbol{\xi}$ to Hopfield module H_s is interacting with the whole training data as described in Eq. (7). This is the update rule of continuous modern Hopfield networks as given in Eq. (2). Therefore, the Hopfield module H_s identifies sample-sample relations and can perform similarity searches like a nearest-neighbor search in the whole training data. H_s can also average over training data that are similar to a mapping of the current input $\boldsymbol{\xi}$.

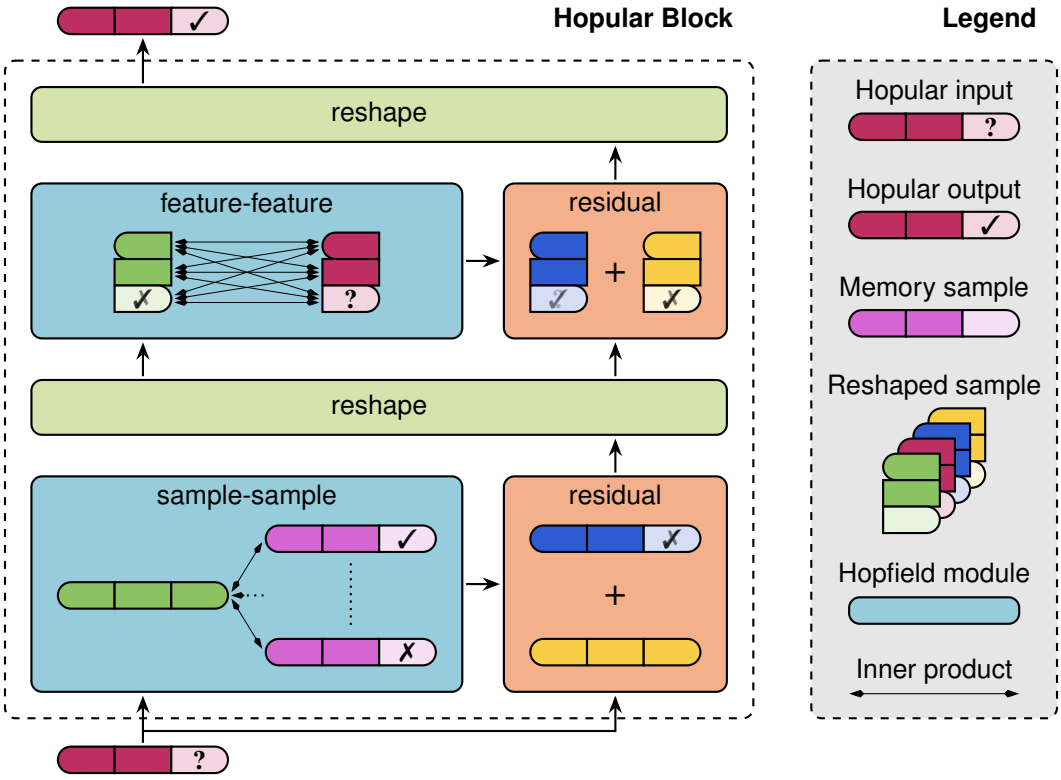

Figure 1: A Hopular Block. The first Hopfield module stores the whole training set and identifies sample-sample relations. The second Hopfield module stores the embedded input features and extracts feature-feature and feature-target relations. Residual blocks combine the Hopfield retrievals with the input to refine the actual predicted target.

Next, we describe Hopfield Module H_s in more detail. Let d be the number of features, e the embedding dimension of each single feature, and h the dimension of the Hopfield embedding space. The forward-pass for module H_s with state $\boldsymbol{\xi} \in \mathbb{R}^{d \cdot e}$, learned weight matrices $\boldsymbol{W_\xi}, \boldsymbol{W_X} \in \mathbb{R}^{h \times (d \cdot e)}$, $\boldsymbol{W_S} \in \mathbb{R}^{(d \cdot e) \times h}$, the stored training set $\boldsymbol{X} \in \mathbb{R}^{(d \cdot e) \times n}$, and a fixed scaling parameter β is given as

$$H_s\left(\boldsymbol{\xi}\right) := \boldsymbol{W_S}\, \boldsymbol{W_X}\, \boldsymbol{X}\, \mathrm{softmax}\left(\beta\, \boldsymbol{X}^T\, \boldsymbol{W_X^T}\, \boldsymbol{W_\xi}\, \boldsymbol{\xi}\right). \tag{7}$$

The hyperparameter β allows to steer the type of fixed point the update rule Eq. (2) converges to, hence it may be used to further amplify the nearest-neighbor-lookup of the sample-sample Hopfield module H_s.

H_s may contain more than one continuous modern Hopfield network. In this case, the respective results are combined and projected, serving as the modules final output. Let $(H_s^i)_{i=1}^N$ be the ordered sequence of N separate Hopfield networks of the same Hopfield module H_s, then the corresponding module output is defined as

$$H_s\left(\boldsymbol{\xi}\right) \;=\; \boldsymbol{W_G}\, \left[H_s^1\left(\boldsymbol{\xi}\right) \ldots\, H_s^N\left(\boldsymbol{\xi}\right)\right] \tag{8}$$

with block matrix $\left[H_s^1\left(\boldsymbol{\xi}\right) \ldots\, H_s^N\left(\boldsymbol{\xi}\right)\right]$ and a learnable weight matrix $\boldsymbol{W_G} \in \mathbb{R}^{(d \cdot e) \times (d \cdot e)}$.

(II) Hopfield Module H_f. The second Hopfield module H_f implements a modern Hopfield network via the layer `Hopfield` (Ramsauer et al., 2021; 2020) with the embedded input features as stored patterns. The current input (the state vector) $\boldsymbol{\xi}$ is reshaped and transposed to the matrix $\boldsymbol{\Xi}$, which serves as input to the Hopfield module H_f. $\boldsymbol{\Xi}$ interacts with the actual embedded input features of the corresponding original sample in the dataset as described in Eq. (9). Again, this is the update rule of continuous modern Hopfield networks as given in Eq. (2). Therefore, the Hopfield module H_f extracts and models feature-feature and feature-target relations. Feature are adjusted and refined by directly associating the intermediate result with the original input sample features.

Next, we describe Hopfield Module H_f in more detail. The state matrix $\boldsymbol{\Xi} \in \mathbb{R}^{e \times d}$ is a transposed and reshaped version of state vector $\boldsymbol{\xi}$ with respect to the embedding dimension e. Using the learned weight matrices $\boldsymbol{W_\Xi}, \boldsymbol{W_Y} \in \mathbb{R}^{h \times e}$, $\boldsymbol{W_F} \in \mathbb{R}^{e \times h}$, the embedded input $\boldsymbol{Y} \in \mathbb{R}^{e \times d}$, and a fixed scaling parameter β the forward-pass is

$$H_f\left(\boldsymbol{\Xi}\right) := \boldsymbol{W_F}\ \boldsymbol{W_Y}\ \boldsymbol{Y}\ \mathrm{softmax}\left(\beta\ \boldsymbol{Y}^T\ \boldsymbol{W_Y^T}\ \boldsymbol{W_\Xi}\ \boldsymbol{\Xi}\right). \tag{9}$$

Of course also H_f may contain more than one continuous modern Hopfield network, which leads to analog formulae as for H_s.

(III) Residual Block. The results of each Hopfield module are combined via a residual block with the input to the Hopfield module. The residual blocks allow additional information to be passed through directly from the previous layer.

Hopular's Objective, Pre-Training, and Training Method. Hopular's objective is a weighted sum of a loss for predicting masked features (BERT masking) and the standard supervised target loss.

Feature Masking. We follow state-of-the-art Deep Learning methods like SAINT (Somepalli et al., 2021) and Non-Parametric Transformers (NPTs) (Kossen et al., 2021) that are tailored to tabular data and use BERT masking (Devlin et al., 2019) of the input features. Masked input features must be predicted during training. Feature masking is an especially beneficial self-supervised approach when handling small datasets as it exerts a strong regularizing effect on the training procedure. The amount of masked attributes during training is determined by the masking probability, which is a hyperparameter of the model. In Hopular, both features and targets can be masked during training, while for inference only the target is masked.

Objective. Hopular's objective is a weighted sum of the masked feature loss L_f and the supervised target loss L_t. The overall loss L is

$$\mathrm{L}\ =\ \gamma\,\mathrm{L}_f\ +\ (1\ -\ \gamma)\mathrm{L}_t\ , \tag{10}$$

where L_t and L_f are the negative logloss in case of discrete attributes and the mean squared error in case of continuous attributes with γ as a hyperparameter. In our default hyperparameter setting γ is annealed using a cosine scheduler starting at 1 with a final value of 0. Another essential hyperparameter for Hopular is β in Eq. (7) and Eq. (9). A small β retrieves a pattern close to the mean of the stored patterns, while a large β retrieves the stored pattern that is closest to the initial state pattern (Ramsauer et al., 2021). For module H_s a large β value emphasizes a nearest-neighbor lookup mechanics. For module H_f a large β value leads to less diluted features. Thus, large β values seem to be beneficial for Hopular. Experiments confirm this assumption (see Section 4).

4 Experiments

We want to know whether Hopular can produce competitive results on small tabular datasets. In particular, we compare Hopular to XGBoost and NPT (Kossen et al., 2021). XGBoost has the lead on tabular data when excluding Deep Learning methods. NPT represents state-of-the-art Deep Learning methods for tabular data, as NPT yielded very good results on small tabular datasets.

4.1 Small-Sized Tabular Datasets

In these experiments, we compare Hopular to other Deep Learning methods and to XGBoost on small-sized tabular datasets.

Methods Compared. We compare Hopular, XGBoost, and NPT with the 24 machine learning methods and self-normalizing networks discussed in (Klambauer et al., 2017). Following (Klambauer et al., 2017; Wainberg et al., 2016), 17 methods were selected from their respective method group as the model with the median performance over all datasets within each method group. NPT is used in a non-transductive setting for a fair comparison.

Hyperparameter Selection. For NPT we use the hyperparameters reported in (Kossen et al., 2021), which also form the basis of Hopular's hyperparameter selection process. For XGBoost we used the package `hyperopt`. More details about the hyperparameter selection and the precise hyperparameter settings can be found in the Appendix Table A1. Hopular is very robust w.r.t. the hyperparameters, since it is possible to obtain good results with only few hyperparameter adjustments.

Datasets. In line with (Klambauer et al., 2017), we consider UCI machine learning repository datasets with less than 1,000 samples as being *small*. To assess the performance of Hopular and other Deep Learning methods on such small datasets, we selected a subset of 16 datasets from (Klambauer et al., 2017). The sizes of these datasets range from 200 to 1,000 samples. We put the focus on smaller sizes, therefore we selected 13 datasets with 500 samples or less. Additionally, we selected two datasets with 500 to 750 samples and one dataset with 750 to 1,000 samples.

Small datasets typically have small test sets, which introduce a high variance in evaluating methods if they are too small or unbalanced. Furthermore, some test sets seem not to be sampled iid from the whole population. Thus, the method evaluation may be highly dependent on the chosen train/test split and performance estimates may be skewed. Problematic datasets in (Klambauer et al., 2017) are characterized by having an accuracy range of 0.5 or higher across well established methods. We have excluded the problematic datasets *seeds*, *spectf*, *libras*, *dermatology*, *arrythmia*, and *conn-bench-vowel-deterding*. The dataset *spect* was excluded as its description in (Fernández-Delgado et al., 2014) is in conflict with the available UCI version regarding the number of attributes and samples. The dataset *heart-hungarian* was excluded as the dataset description is insufficient to distinguish between categorical and continuous attributes, which is required by some methods. Since *breast-cancer-wisc* is solved (0.9859 accuracy), it is excluded as it does not distinguish the performance of methods. We drop *heart-va*, since the best reported method has only a low accuracy of 0.4. The datasets and train/test splits were taken from (Fernández-Delgado et al., 2014). Next, we give a short description of the selected datasets that we used for comparison.

conn-bench-sonar-mines-rocks: Also known as *Connectionist Bench (Sonar, Mines vs. Rocks)*. A classification setting of 208 instances with 60 continuous features per instance. The task is to discriminate between sonar sounds from metal vs. rocks.

glass: Also known as *Glass Identification*. A classification setting of 214 instances with 9 continuous features per instance. The task is to discriminate between 6 types of glass.

statlog-heart: A classification setting of 270 instances with 6 continuous and 7 categorical features per instance. The task is to predict the presence or absence of a heart disease.

breast-cancer: A classification setting of 286 instances with 9 categorical features per instance. The task is to predict the presence or absence of breast cancer.

heart-cleveland: Also known as *Heart Disease*. A classification setting of 303 instances with 6 continuous and 7 categorical features per instance. The task is to predict the presence or absence of a heart disease.

haberman-survival: A classification setting of 306 instances with 3 continuous features per instance. The task is to predict whether patients survived longer than 5 years or not.

vertebral-column2, vertebral-column3: Also know as *Vertebral Column Dataset*. Two classification settings of 310 instances each with 6 continuous features per instance. The task is to classify patients into either 2 or 3 classes.

primary-tumor: A classification setting of 330 instances with 17 categorical features per instance. The task is to predict the class of primary tumors.

ecoli: A classification setting of 336 instances with 5 continuous and 2 categorical features per instance. The tasks is to classify proteins into 8 classes.

horse-colic: A classification setting of 368 instances with 8 continuous and 19 categorical features per instance. The task is to predict the survival or death of a horse.

congressional-voting: A classification setting of 435 instances with 16 categorical features per instance. The task is to predict political affiliation.

cylinder-bands: A classification setting of 512 instances with 20 continuous and 19 categorical features per instance. The task is to classify the band type.

credit-approval: A classification setting of 690 instances with 6 continuous and 9 categorical features per instance. The task is to determine positive or negative feedback for credit card applications.

blood-transfusion: Also known as *Blood Transfusion Service Center*. A classification setting of 748 instances with 3 continuous and 1 categorical feature per instance. The task is to predict whether a person donated blood or not.

mammographic: Also known as *Mammographic Mass*. A classification setting of 961 instances with 1 continuous and 5 categorical features per instance. The task is to discriminate between benign and malignant mammographic masses.

Table 1: Median rank of compared methods across the datasets of the UCI machine learning repository. Methods were ranked for each dataset according to the accuracy. 17 method groups have been compared previously (Wainberg et al., 2016). We added XGBoost (Chen & Guestrin, 2016), Non-Parametric Transformers (Kossen et al., 2021), Self-Normalizing Networks (Klambauer et al., 2017), and our Hopular. "Partial Least Squares" abbreviates "Partial Least Squares and Principal Component Regression". Deep Learning methods are indicated by "(DL)" and are not grouped. Hopular achieves the lowest median rank of 6.75, therefore is the best performing method across datasets.

Method	Rank	Method	Rank
Hopular (DL)	**6.75**	MSRAinit (DL)	14.00
Support Vector Machines	8.25	Other Ensembles	14.00
Logistic and Multinomial Regression	9.00	Rule-Based Methods	14.50
Self-Normalizing Networks (DL)	9.75	LayerNorm (DL)	14.50
Random Forest	11.25	Highway Networks (DL)	14.50
Multivariate Adaptive Regression Splines	11.50	Non-Parametric Transformers	14.50
WeightNorm (DL)	11.50	Other Methods	17.25
Generalized Linear Models	11.75	ResNet (DL)	17.50
Neural Networks (DL)	12.00	Partial Least Squares	18.25
Boosting Methods	12.00	Bayesian Methods	19.50
Decision Trees	12.75	XGBoost	20.00
BatchNorm (DL)	13.25	Nearest Neighbour	23.00
Bagging Methods	13.75	Stacking (Wolpert)	26.00
Discriminant Analysis	13.75		

Results. Table 1 gives the median rank of compared methods across the datasets of the UCI machine learning repository. Methods were ranked for each dataset according to the accuracy. 17 method groups have been compared previously (Wainberg et al., 2016), to which we added XGBoost (Chen & Guestrin, 2016), Non-Parametric Transformers (Kossen et al., 2021), Self-Normalizing Networks (Klambauer et al., 2017), and our Hopular. Hopular has a median rank of 6.75, followed by Support Vector Machines with 8.25, while Non-Parametric Transformers and XGBoost have a median rank of 14.5 and 20, respectively. Hopular with modern Hopfield networks as memory performs better than other Deep Learning methods and in particular better than the closely-related Non-Parametric Transformers (NPT). Hopular outperforms other Deep Learning methods and XGBoost. **Across the considered UCI datasets, Hopular is the best performing method.**

4.2 Medium-Sized Tabular Datasets

In these experiments, we compare Hopular to other Deep Learning methods and to XGBoost on medium-sized tabular datasets. In (Shwartz-Ziv & Armon, 2021), the authors show that XGBoost outperforms various Deep Learning methods that are designed for tabular data on datasets that did not appear in the original papers. We want to know whether XGBoost still has the lead on these medium-sized datasets.

Methods Compared. We compare Hopular, NPT, and XGBoost, where NPT is used in a non-transductive setting for a fair comparison.

Hyperparameter Selection. We used the default hyperparameters for Hopular as well as NPT, which were found during the hyperparameter selection process for the UCI dataset in previous Subsection 4.1. For XGBoost, we applied the same Bayesian hyperparameter optimization procedure as described in (Shwartz-Ziv & Armon, 2021). For more details see Appendix Table A1.

Datasets. We selected the datasets of (Shwartz-Ziv & Armon, 2021), where XGBoost performed better than Deep Learning methods that have been designed for tabular data.

shrutime: A classification setting of 10,000 instances with 2 continuous and 9 categorical features per instance. The task is to predict whether a bank account is closed or not.

blastchar: A classification setting of 7,048 instances with 3 continuous and 17 categorical features per instance. The task is to predict customer behavior.

gesture-phase: Also known as *Gesture Phase Segmentation*. A classification setting of 9,873 instances with 31 continuous features per instance. The task is to classify gesture phases.

eye-movements: A classification setting of 10,936 instances with 19 continuous and 3 categorical features per instance. The task is to discriminate between correct, irrelevant or relevant answers.

Table 2: Results of all compared methods on the subset of medium-sized tabular datasets (Shwartz-Ziv & Armon, 2021). Reported values are the *logloss* on the respective test sets, averaged over *three* replicates, as well as the corresponding *standard error of the mean*.

Method	**Eye**	**Gesture**	**Shrutime**	**Blastchar**
Hopular	90.40 ± 0.16	113.14 ± 0.18	**34.61** ± 0.05	**40.99** ± 0.06
NPT	95.16 ± 0.84	141.86 ± 6.36	34.70 ± 0.12	41.11 ± 0.02
XGBoost	**88.30** ± 0.00	**91.93** ± 0.31	35.93 ± 0.00	46.60 ± 0.00

Table 3: Results of all compared methods on the subset of medium-sized tabular datasets (Shwartz-Ziv & Armon, 2021). Reported values are the *misclassification error* on the respective test sets, averaged over *three* replicates, as well as the corresponding *standard error of the mean*.

Method	**Eye**	**Gesture**	**Shrutime**	**Blastchar**
Hopular	45.91 ± 0.17	36.16 ± 0.03	**14.23** ± 0.08	**19.41** ± 0.15
NPT	48.59 ± 0.30	44.33 ± 0.67	14.35 ± 0.04	19.63 ± 0.09
XGBoost	**43.06** ± 0.00	**33.92** ± 0.00	14.94 ± 0.00	21.14 ± 0.00

Table 2 gives the *logloss* and Table 3 the *misclassification error* of Hopular, NPT, and XGBoost on the medium-sized datasets. We report both losses, since the package `hyperopt` changed the default loss type in version `1.3.0` (Bergstra et al., 2013). The evaluation procedure is from (Shwartz-Ziv & Armon, 2021). Hopular is second best on two datasets and best on the other two datasets. Over the 4 datasets NPT has average rank 2.5, XGBoost has average rank 2, and Hopular has average rank 1.5. **On average over all 4 datasets, Hopular performs better than NPT and XGBoost.**

4.3 Runtime Estimates

Table 4: Runtime estimates of Hopular and XGBoost based on the dataset *gesture-phase*, specified in *seconds per step*. One *step* refers to one complete pass through of the training set for *Training*, and one complete pass through of the test set for *Inference*. Although it takes more time to train Hopular, this difference is negligible during inference, as the absolute runtime is still in the seconds.

	XGBoost		**Hopular**	
	Training	Inference	Training	Inference
$[sec/step]$	0.18	0.115	3.6	0.116

5 Conclusion

Hopular is a novel Deep Learning architecture which poses as a strong contender to current state-of-the-art methods like XGBoost and other Deep Learning methods specialized in small- and medium-sized datasets. Hopular's performance is not only on par with all compared methods, but it even surpasses them on average. One advantage of Hopular compared to XGBoost is the capability to handle data with missing values without the need for additional imputation techniques. This makes Hopular highly relevant for real world scenarios where incomplete data is a common companion.

ETHICS STATEMENT

Our research has the potential to positively impact a wide domain of fields in real life. However, all new developments in machine learning can be applied for both good and bad. Our system could be used for medical applications to save lives but it could also be used for malevolent systems. It is the society that decides how new technology is employed but it is our responsibility as scientists to inform the public about the possibilities and limitations of new technologies. A big danger is that practitioners apply our method to their problem settings in an unreflected way and too heavily rely on the outcomes in a non-questioning way. For example, in medical practitioners might rely on the technical system and shift the liability towards the machine. Depending on the area of application an error of our method might of proportionally less concern like the prediction of customer behavior. If our method is used for medical data an incorrect prediction could cause greater harm. However, this is true for almost all machine learning methods, and usage and testing falls within the responsibility of the application area. Since our method relies on humanly annotated data it could learn undesirable biases and propagate them to downstream applications. Therefore, the responsible use of our method depends on a careful selection of the training data and the awareness of the potential biases.

REPRODUCIBILITY STATEMENT

To ensure that the presented results are reproducible we will make the source code available to the public after the reviewing period. This includes all material and information such as hyperparameter optimizations scripts, dataset preprocessing pipelines as well as corresponding usage guidelines.

REFERENCES

A. Abutbul, G. Elidan, L. Katzir, and R. El-Yaniv. DNF-Net: A neural architecture for tabular data. *ArXiv*, 2006.06465, 2020. URL https://openreview.net/forum?id=73WTGs96kho. 9th International Conference on Learning Representations (ICLR).

S. Ö. Arik and T. Pfister. TabNet: Attentive interpretable tabular learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(8):6679–6687, 2021.

James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pp. 115–123. PMLR, 2013.

B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pp. 144–152. ACM Press, Pittsburgh, PA, 1992.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pp. 785–794, New York, NY, USA, 2016. Association for Computing Machinery. doi: 10.1145/2939672.2939785.

C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

S. Darabi, S. Fazeli, A. Pazoki, S. Sankararaman, and M. Sarrafzadeh. Contrastive Mixup: self- and semi-supervised learning for tabular domain. *ArXiv*, 2108.12296, 2021.

M. Demircigil, J. Heusel, M. Löwe, S. Upgang, and F. Vermet. On a model of associative memory with huge storage capacity. *Journal of Statistical Physics*, 168(2):288–299, 2017. doi: 10.1007/s10955-017-1806-y.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186. Association for Computational Linguistics, 2019. doi: 10.18653/v1/N19-1423.

A. V. Dorogush, A. Gulin, G. Gusev, N. Kazeev, L. O. Prokhorenkova, and A. Vorobev. CatBoost: unbiased boosting with categorical features. *ArXiv*, 1706.09516, 2017.

L. Du, F. Gao, X. Chen, R. Jia, J. Wang, J. Zhang, S. Han, and D. Zhang. TabularNet: A neural network architecture for understanding semantic structures of tabular data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, pp. 322–331, New York, NY, USA, 2021. Association for Computing Machinery. doi: 10.1145/3447548.3467228.

N. Erickson, J. Mueller, A. Shirkov, H. Zhang, P. Larroy, M. Li, and A. Smola. AutoGluon-Tabular: Robust and accurate AutoML for structured data. *ArXiv*, 2003.06505, 2020.

M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems? *The Journal of Machine Learning Research*, 15(1): 3133–3181, 2014.

J. Fiedler. Simple modifications to improve tabular neural networks. *ArXiv*, 2108.03214, 2021.

J. H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, 2001. doi: 10.1214/aos/1013203451.

Y. Gorishniy, I. Rubachev, V. Khrulkov, and A. Babenko. Revisiting deep learning models for tabular data. *ArXiv*, 2106.11959, 2021.

X. Guo, Y. Quan, H. Zhao, Q. Yao, Y. Li, and W. Tu. TabGNN: Multiplex graph neural network for tabular data prediction. *ArXiv*, 2108.09127, 2021.

T. K. Ho. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, pp. 278–282, 1995. doi: 10.1109/ICDAR.1995.598994.

J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, 81(10):3088–3092, 1984. doi: 10.1073/pnas.81.10.3088.

X. Huang, A. Khetan, M. Cvitkovic, and Z. Karnin. TabTransformer: Tabular data modeling using contextual embeddings. *ArXiv*, 2012.06678, 2020.

A. Kadra, M. Lindauer, F. Hutter, and J. Grabocka. Regularization is all you need: Simple neural nets can excel on tabular data. *ArXiv*, 2106.11189, 2021.

G. Ke, A. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems*, pp. 971–980, 2017.

J. Kossen, N. Band, C. Lyle, A. N. Gomez, T. Rainforth, and Y. Gal. Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. *ArXiv*, 2106.02584, 2021.

D. Krotov and J. J. Hopfield. Dense associative memory for pattern recognition. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, pp. 1172–1180. Curran Associates, Inc., 2016.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.

F. W. J. Olver, D. W. Lozier, R. F. Boisvert, and C. W. Clark. *NIST handbook of mathematical functions*. Cambridge University Press, 1 pap/cdr edition, 2010. ISBN 9780521192255.

S. Popov, S. Morozov, and A. Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *ArXiv*, 1909.06312, 2019. URL https://openreview.net/forum?id=r1eiu2VtwH. 8th International Conference on Learning Representations (ICLR).

L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin. CatBoost: unbiased boosting with categorical features. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, V. Greiff, D. Kreil, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. Hopfield networks is all you need. *ArXiv*, 2008.02217, 2020.

H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, V. Greiff, D. Kreil, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. Hopfield networks is all you need. In *9th International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=tL89RnzIiCd.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. doi: 10.1016/j.neunet.2014.09.003.

B. Schölkopf and A. J. Smola. *Learning with kernels - Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, 2002.

I. Shavitt and E. Segal. Regularization learning networks: Deep learning for tabular datasets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

R. Shwartz-Ziv and A. Armon. Tabular Data: Deep learning is not all you need. *ArXiv*, 2106.03253, 2021. URL https://openreview.net/forum?id=vdgtepS1pV. AutoML Workshop of International Conference on Machine Learning (ICML).

G. Somepalli, M. Goldblum, A. Schwarzschild, C. B. Bruss, and T. Goldstein. SAINT: Improved neural networks for tabular data via row attention and contrastive pre-training. *ArXiv*, 2106.01342, 2021.

W. Song, C. Shi, Z. Xiao, Z. Duan, Y. Xu, M. Zhang, and J. Tang. AutoInt: Automatic feature interaction learning via self-attentive neural networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pp. 1161–1170, New York, NY, USA, 2019. Association for Computing Machinery. doi: 10.1145/3357384.3357925.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. Curran Associates, Inc., 2017.

M. Wainberg, B. Alipanahi, and B. J. Frey. Are random forests truly the best classifiers? *The Journal of Machine Learning Research*, 17(1):3837–3841, 2016.

M. Widrich, B. Schäfl, M. Pavlović, H. Ramsauer, L. Gruber, M. Holzleitner, J. Brandstetter, G. K. Sandve, V. Greiff, S. Hochreiter, and G. Klambauer. Modern Hopfield networks and attention for immune repertoire classification. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2020.

H. Xu, M. Ainsworth, Y.-C. Peng, M. Kusmanov, S. Panda, and J. T. Vogelstein. When are deep networks really better than random forests at small sample sizes? *ArXiv*, 2108.13637, 2021.

J. Yoon, Y. Zhang, J. Jordon, and M. vanDerSchaar. VIME: Extending the success of self- and semi-supervised learning to tabular domain. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 11033–11043. Curran Associates, Inc., 2020.

Y. You, J. Li, S. Reddi, J. Hseu, S. Kumar, S. Bhojanapalli, X. Song, J. Demmel, K. Keutzer, and C. Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020. ArXiv 1904.00962.

M. Zhang, J. Lucas, J. Ba, and G. E. Hinton. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems 32*, 2019. ArXiv 1907.08610.

A APPENDIX

A.1 HYPERPARAMETER SELECTION PROCESS

All evaluated hyperparameter settings are shown in Table A1 for both Hopular and NPT. Since NPT constitutes an important benchmark for our model comparison we took exactly the same hyperparameter settings that were successfully used among several datasets in (Kossen et al., 2021). As Hopular provides an additional adjustable scaling factor for β, we also tested scaling factors of 100 and 1000 to further emphasize the nearest-neighbor search. The LAMB (You et al., 2020) optimizer was used for all Hopular and NPT experiments, extended by a Lookahead (Zhang et al., 2019) wrapper with fixed values. For LAMB we used $\beta_L = (0.9, 0.999)$, $\epsilon = 1e{-}6$ and for Lookahead $\alpha = 0.5$, $k = 6$. NPT and Hopular were both trained for 10,000 epochs with early stopping.

Table A1: Complete listing of all evaluated hyperparameter settings for Hopular. Settings with a scaling factor of 1 were additionally performed for NPT.

learn. rate	**#Hop. blocks**	**dropout prob.**	**#Hop. nets**	**label mask. prob.**	**feature mask. prob.**	**learn. rate scheduler**	**scaling factor**
0.001	8	0.1	8	1.0	0.15	cosine	1
0.001	16	0.1	8	1.0	0.15	cosine	1
0.001	8	0.1	16	1.0	0.15	cosine	1
0.001	16	0.1	16	1.0	0.15	cosine	1
0.001	8	0.1	8	0.1	0.15	cosine	1
0.001	8	0.1	8	0.5	0.15	cosine	1
0.001	8	0.1	8	1.0	0.2	cosine	1
0.001	8	0.1	8	1.0	0.15	cosine cyclic	1
0.001	8	0.1	8	1.0	0.15	cosine	100
0.001	16	0.1	8	1.0	0.15	cosine	100
0.001	8	0.1	16	1.0	0.15	cosine	100
0.001	16	0.1	16	1.0	0.15	cosine	100
0.001	8	0.1	8	0.1	0.15	cosine	100
0.001	8	0.1	8	0.5	0.15	cosine	100
0.001	8	0.1	8	1.0	0.2	cosine	100
0.001	8	0.1	8	1.0	0.15	cosine cyclic	100
0.001	8	0.1	8	1.0	0.15	cosine	1000
0.001	16	0.1	8	1.0	0.15	cosine	1000
0.001	8	0.1	16	1.0	0.15	cosine	1000
0.001	16	0.1	16	1.0	0.15	cosine	1000
0.001	8	0.1	8	0.1	0.15	cosine	1000
0.001	8	0.1	8	0.5	0.15	cosine	1000
0.001	8	0.1	8	1.0	0.2	cosine	1000
0.001	8	0.1	8	1.0	0.15	cosine cyclic	1000

A.2 SOURCE CODE

Source code will be made available for the public after the reviewing period.