ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

On the number of latent representations in deep neural networks for tabular data

Edouard Couplet[1], Pierre Lambert[1],
Michel Verleysen[1], John A. Lee[2,1], Cyril de Bodt[1] *

1-UCLouvain - ICTEAM/ELEN 2-UCLouvain - IREC/MIRO

Abstract. Most recent deep neural network architectures for tabular data operate at the feature level and process multiple latent representations simultaneously. While the dimension of these representations is set through hyper-parameter tuning, their number is typically fixed and equal to the number of features in the original data. In this paper, we explore the impact of varying the number of latent representations on model performance. Our results suggest that increasing the number of representations beyond the number of features can help capture more complex interactions, whereas reducing their number can improve performance in cases where there are many uninformative features.

1 Introduction

To date, ensemble models based on decision trees such as XGBoost [1] still outperform deep learning (DL) models on tabular data in supervised tasks [2]. However, DL remains attractive, notably for building differentiable multi-modal pipelines in domains such as healthcare or robotics which require integrating data, including tables, from a wide variety of sources [3]. As a consequence, a growing number of deep learning models for tabular data have been proposed in the past few years. We can classify these models into two categories: those that operate at the *sample level* and those that operate at the *feature level*. Most recent models fall within the latter category and are based on Transformers [4, 5, 6] or even Graph Neural Networks (GNNs) [7]. Unlike standard Multilayer Perceptrons (MLPs) which use one latent representation per sample, these models use one latent representation per feature: a token in case of transformers, a node vector in case of GNNs. While this enables easier interpretation in terms of contextual embeddings, the choice of using one representation per feature is arbitrary and, as empirically shown in this work, may not be optimal in terms of prediction performance. This paper hence investigates how the number of latent representations affects the performance of DL models on tabular data. We first provide, in Section 2, a general definition of a neural network model for tabular data. In Section 3, we build upon this definition for designing a simple and generic *feature-level* architecture. In Section 4, we test this architecture on several datasets and discuss how performance is affected by varying the number of hidden representations. Finally, Section 5 draws conclusions and outlines directions for future research.

*EC is supported by a FSR grant (UCLouvain). PL is a FRIA grantee of the F.R.S.-FNRS. JAL is a Research Director with the F.R.S.-FNRS. CdB is supported by Service Public de Wallonie Recherche under grant n2010235-ARIAC by DIGITALWALLONIA4.AI.

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

2 Neural network *feature-level* model for tabular data

This work focuses on supervised learning tasks. A data set is denoted $X = \{(\mathbf{x^{(i)}}, y^{(i)})\}_{i=1}^{n}$ where $\mathbf{x^{(i)}} = (x_1^{(i)}, x_2^{(i)}, ..., x_m^{(i)}) \in \mathbb{X}$ represents the features of the ith sample and $y^{(i)} \in \mathbb{Y}$ is the sample label. We consider two types of tasks: binary classification if $\mathbb{Y} = \{0, 1\}$ and regression if $\mathbb{Y} = \mathbb{R}$. We formulate the task of label prediction as an encoding-decoding problem and propose the following general *feature-level* model for tabular data[1]:

$$\mathbf{y} = \rho_\psi(\text{pool}(\phi_\theta(f(\mathbf{x})))) \ ,$$

where $f(\cdot)$ is the *pre-encoder*, $\phi_\theta(\cdot)$ is a neural network *encoder* parameterized by θ and $\rho_\psi(\cdot)$ is a neural network *decoder* parameterized by ψ.

A defining characteristic of tabular data is that the feature space does not need to be homogeneous: for two given features $x_j \in \mathbb{X}_j$ and $x_{j'} \in \mathbb{X}_{j'}$ we usually do not have $\mathbb{X}_j = \mathbb{X}_{j'}$. In particular, we may have a mix of numerical and categorical features. However, a standard neural network only takes vectors of real numbers as input, so we first need a pre-processing step to encode each feature x_j with a specific function $f_j : \mathbb{X}_j \to \mathbb{R}^{d_j}$. When working with *sample-level* models, we typically concatenate all encodings $f_j(x_j)$ into a single sample representation. When working with *feature-level* models, we project all features x_j in a common space $\mathbb{R}^d$ and we keep a separate vector representation for each feature. We define $f : \mathbb{X} \to \mathbb{R}^{d \times m}$ as $f(\mathbf{x}) = (f_1(x_1), f_2(x_2), ..., f_m(x_m))$.

The goal of the encoder is to learn an appropriate representation of the data, enabling accurate label predictions by the decoder. It is a neural network with an input layer $\phi_0 : \mathbb{R}^{d \times m} \to \mathbb{R}^{d_0 \times m_0}$ and l hidden layers $\phi_k : \mathbb{R}^{d_{k-1} \times m_{k-1}} \to \mathbb{R}^{d_k \times m_k}$ such that $\phi = \phi_l \circ \phi_{l-1} \circ ... \circ \phi_1 \circ \phi_0$. The final m_l representations learned by the encoder are aggregated together via a pooling operator into $\mathbf{z} \in \mathbb{R}^{d_l}$.

The decoder is another neural network $\rho : \mathbb{R}^{d_l} \to \mathbb{Y}$ with $\rho(\mathbf{z}) = \hat{y}$, where $\hat{y}$ is the predicted label for a given sample $\mathbf{x}$.

While the decoder is most often composed of one or two fully connected linear layers with appropriate nonlinear activations depending on the task, what really differentiates recent deep learning models for tabular data is the design of the hidden layers in the encoder. For instance, [5] uses transformer blocks with multi-head attention, [4] also uses transformer blocks but with an additional inter-sample attention mechanism, [6] uses transformers with attention coefficients computed based on a feature graph, [7] uses graph message passing convolutional layers instead, etc. A shared aspect of all these models, however, is that the dimension m_k of the hidden layers is always equal to m, the number of features in the original data. Although this allows for a nice interpretation of hidden representations as contextual feature embeddings, one may wonder whether m_k must necessarily be fixed to m? We then ask the following question: how does taking values of $m_k > m$ or $m_k < m$ affect the performance of *feature-level* models on tabular data?

[1]The chosen vocabulary and notations originate in the literature on deep sets and graph neural networks.

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

3 A simple *feature-level* architecture

To carry out our investigation, we design a simple and generic *feature-level* architecture for tabular data. This architecture is depicted in Fig. 1.

The *pre-encoder* is defined feature-wise; for a given feature index j, we have:

$$f_j(\cdot) = \text{pad}\left(g_j(\cdot)\right), \text{ with } g_j(\cdot) = \begin{cases} \text{gaussianize}(\cdot) & \text{if } j \text{ is numerical} \\ \text{one-hot-encode}(\cdot) & \text{if } j \text{ is categorical} \end{cases}$$

where gaussianize$(\cdot)$ applies a quantile transformation such that the jth feature follows a normal distribution, one-hot-encode$(\cdot)$ transforms its input into a binary vector, and pad$(\cdot)$ appends zeroes to the obtained encoding such that $\sum_j f_j(x_j) = \text{concatenate}_j(g_j(x_j))^T$ for a given sample $\mathbf{x}$. The *encoder* ϕ is composed of three layers. It takes as input $\mathbf{e} = f(\mathbf{x})$ and outputs $\mathbf{h}_2 = \phi(\mathbf{e})$. The input layer ϕ_0 can be interpreted as a static embedding layer. The two hidden layers ϕ_1 and ϕ_2 first combine latent feature representations linearly, then apply the same nonlinear transformation to all new combined representations. To reduce the number of hyper-parameters, the dimension of the latent space is constant: $d_0 = d_1 = d_2 = d'$, as well as the number of latent representations in both hidden layers: $m_1 = m_2 = m'$ with m' not necessarily equal to the number m of original features. The equations for each layer are then:

$$\mathbf{h_0} = \phi_0(\mathbf{e}) \quad = \text{ReLU}(W_0\mathbf{e} + \mathbf{b_0}\mathbf{1}^T) \tag{1}$$

$$\mathbf{h_1} = \phi_1(\mathbf{h}_0) = \text{ReLU}(W_1\tilde{\mathbf{h}}_0 + \mathbf{b}_1\mathbf{1}^T) \text{ with } \tilde{\mathbf{h}}_0 = \mathbf{h}_0 A_1 \tag{2}$$

$$\mathbf{h_2} = \phi_2(\mathbf{h}_1) = \text{ReLU}(W_2\tilde{\mathbf{h}}_1 + \mathbf{b}_2\mathbf{1}^T) \text{ with } \tilde{\mathbf{h}}_1 = \mathbf{h}_1 A_2 \tag{3}$$

where W_k, $\mathbf{b}_k$ and A_k are trainable parameters and $\mathbf{1}$ is a vector of d' ones. The pooling operator is simply a sum : $\mathbf{z} = \sum_{j=1}^{m'} h_{2j}$. The *decoder* takes $\mathbf{z}$ as input and outputs a label prediction $\hat{y}$. It consists of a unique linear layer: $\hat{y} = W\mathbf{z}$. For regression tasks, we minimize the mean square error. For classification tasks, we add a sigmoid activation function and minimize the binary cross entropy.

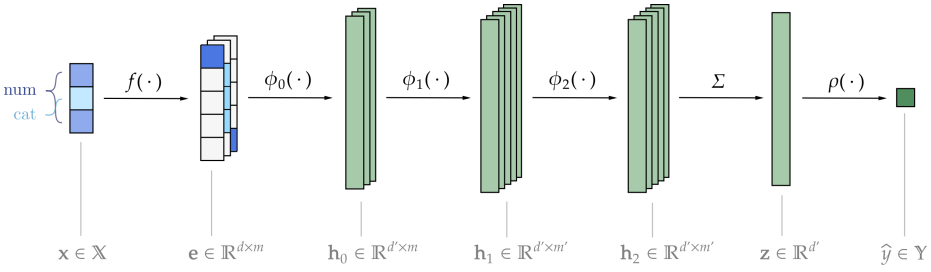

Fig. 1: Generic *feature-level* architecture for tabular data. Equations for computing latent representations $\mathbf{e}, \mathbf{h}_0, \mathbf{h}_1, \mathbf{h}_2, \mathbf{z}$, as well as the final prediction $\hat{y}$ are detailed in the main text of Section 3.

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

4 Experiments and discussion

This section evaluates the performance of our generic architecture on 12 different data sets and discusses how it is affected by m', the number of latent representations. The data sets all come from the benchmark introduced in [2]. We consider both regression and classification tasks, as well as data sets with both categorical and numerical features. For the *pre-encoder*, we use ScikitLearn's `OneHotEncoder` and `QuantileTransformer`. We fix d' to 128 and test the following values for m' : $\{1, 2, 4, 8, 16, 32, 64, 128\} \cup \{m\}$. We also compare our architecture to a standard *sample-level* MLP with a latent space of dimension 192, such that the MLP always has more parameters than our architecture. We split the data into training and test sets (70% and 30%, respectively). We use a batch size of 256 and train each model for a maximum of 100 epochs, using early stopping with a patience of 20. We repeat these operations for each data set using 20 different random seeds. The averaged test results are reported in Fig. 2. For classification tasks, we look at the test accuracy; for regression tasks, at the R2 score. For easier interpretation, we multiply each score by 100 and display the relative performance with respect to the case $m' = m$.

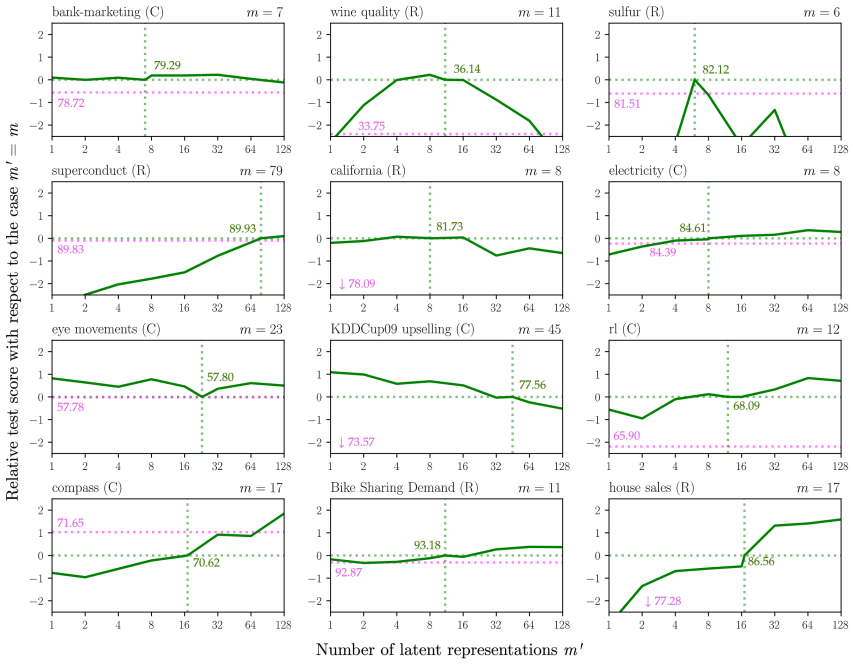

Fig. 2: Relative performances of our *feature-level* architecture on various data sets, for varying values of m'. We display the absolute scores ($\times 100$) for the case $m' = m$ (green) and the MLP (magenta). C stands for classification, R for regression, m for the original number of features and m' for the number of latent representations.

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

We observe several kinds of behaviors as the number of latent representation m' varies: (a) performance increases when increasing m', (b) performance increases when decreasing m', (c) performance decreases for $m' > m$ and for $m' < m$, and (d) performance stays approximately the same for different values of m'. Additionally we observe that (e) in general, the *feature-level* architecture outperforms the standard *sample-level* MLP. We formulate two hypotheses to explain our observations:

Hypothesis 1 A higher number of latent representations makes a model more expressive and allows to capture more complex feature-target relationships.

Hypothesis 2 A higher number of latent representations makes a model more sensitive to uninformative features, which are common in tabular data [2].

In fact, (a) would correspond to cases where the feature-target relationship can be better approximated with a higher m' and where there are not too many uninformative features; (b) to cases where there are a significant proportion of uninformative features but the feature-target relationship can still be well captured with a low m'; (c) to a similar case but where the feature-target relationship is too complex to capture with a low m' such that performances also decrease as m' decreases; (d) to simpler cases where there are no uninformative features and the feature-target relationship is well captured for any m', or to in-between cases where the effects of uninformative features and complexity of the feature-target relationship cancel out. Finally, (e) suggests that a *sample-level* MLP behaves like a *feature-level* architecture with a high m': it is thus more sensitive to uninformative features, with the major drawback that it has much fewer degrees of freedom for reducing this sensitivity without compromising expressiveness.

We conduct an additional experiment to verify whether our hypotheses are plausible. We consider the *compass* data set that, based on the previous results, has few uninformative features and a feature-target relationship that is best captured with higher values of m', as in situation (a). We add to this dataset m random uninformative features and conduct the same experiment as described above. Figure 3 displays the results. We observe that the behavior of the model completely changes with a more important proportion of uninformative features: now reducing m enables to significantly increase accuracy, suggesting less sensitivity but still some level of expressiveness (more like situation (b)). This is in line with our hypotheses; more in depth experiments will be conducted in further works to provide stronger evidences.

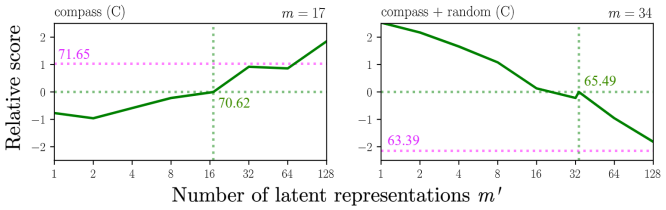

Fig. 3: Performances of our *feature-level* architecture on the *compass* data set, with and without the addition of uninformative features.

ESANN 2023 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 4-6 October 2023, i6doc.com publ., ISBN 978-2-87587-088-9. Available from http://www.i6doc.com/en/.

5 Conclusion and perspectives

This work investigates the impact of the number of latent representations in deep neural networks for tabular data. We show that having one representation per feature of the original data may not be optimal and adjusting this number based on task complexity and the proportion of uninformative features can enhance model performance. Interestingly, in cases with many uninformative features, performance can be improved by reducing the number of latent representations, suggesting a kind of regularization phenomenon. This could have important implications for tasks involving high dimensional data, where the number of latent representations raises computational limitations. An intriguing research direction would be to quantitatively define the concepts of "task complexity" and "uninformativeness of a feature" in the context of tabular data and explore more in detail their relationship with model performance. By doing so, the number of latent representations could potentially be set *a priori*, reducing the need for expensive hyper-parameter tuning. A related research direction would be to investigate also the relationship with latent space dimensionality; this would help build neural networks that are better suited to the data they intend to model.

References

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[2] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? In *NeurIPS 2022 Datasets and Benchmarks Track*, Advances in Neural Information Processing, New Orleans, United States, November 2022.

[3] Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[4] Gowthami Somepalli, Micah Goldblum, Avi Schwarzschild, C Bayan Bruss, and Tom Goldstein. Saint: Improved neural networks for tabular data via row attention and contrastive pre-training. *arXiv preprint arXiv:2106.01342*, 2021.

[5] Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

[6] Jiahuan Yan, Jintai Chen, Yixuan Wu, Danny Z Chen, and Jian Wu. T2g-former: Organizing tabular features into relation graphs promotes heterogeneous feature interaction. *arXiv preprint arXiv:2211.16887*, 2022.

[7] Mario Villaizán-Vallelado, Matteo Salvatori, Belén Carro Martinez, and Antonio Javier Sanchez Esguevillas. Graph neural network contextual embedding for deep learning on tabular data. *arXiv preprint arXiv:2303.06455*, 2023.