# Sparsely Connected Layers for Financial Tabular Data

**Mohammed Abdulrahman***
Vector Institute
University of Waterloo
m3abdulr@uwaterloo.ca

**Hui Chen**
Aven Inc.
hui.chen@aven.com

**Yin Wang**
Aven Inc.
yin@aven.com

## Abstract

While neural networks are the standard for unstructured data, such as images and text, their performance often lags behind more traditional machine learning models, like gradient-boosted trees, for tabular data. This is supported by academic studies, industry practices, and Kaggle competitions. In particular, there is no straightforward way to increase the number of layers in neural networks applied to tabular data — a key factor in their success with unstructured data. Deep fully connected networks suffer from the vanishing gradient problem, and convolutional layers and transformers are generally not directly applicable to tabular data. Special constructs, such as skip layers and attention mechanisms, have been adapted to tabular data models with limited success. In this paper, we show that for consumer financial tabular data, while standard two-layer neural networks typically underperform when compared to gradient-boosted trees, sparsely connected layers can increase network depth and reliably outperform gradient-boosted trees. The superior performance appears to stem from the sparse layers' ability to reduce correlations in the input data, a common challenge in high-dimensional tabular data. Therefore, we are hopeful that this method could be applicable to other domains facing similar challenges.

## 1 Introduction

The past decade has witnessed rapid progress in neural networks for unstructured data, such as images and text, much of which is due to specialized network constructs that enable very deep architectures without suffering from the vanishing gradient problem. In particular, convolutional neural networks have come to dominate nearly all vision tasks [12], and more recently, transformers have become predominant in language modeling [21].

On the other hand, while tabular data is arguably simpler and easier to process, no generic deep neural architecture is available for it. Standard two-layer neural networks often lag behind modern gradient-boosted tree models (GBDT) such as XGBoost [4], LightGBM [11], and CatBoost [15]. For instance, GBDTs dominated the 2022 Kaggle competitions, winning 7 out of 10 challenges [6]. Another survey paper examining academic datasets concluded that decision tree models remain the state of the art, outperforming others on 7 out of 9 datasets [3].

Motivated by this performance gap, a substantial amount of research has focused on developing deep neural networks for tabular data, which can be broadly divided into three categories. First, some approaches combine neural networks with tree-based methods. For instance, [13] proposes a network structure that imitates a binary decision tree, while [18] argues that an ensemble of the two methods performs better. Second, the wide and deep network architecture [5] is applicable for certain datasets. Building on this, DeepFM [9] incorporates a learned Factorization Machine [16] into a deep network. Finally, recent work has explored using attention mechanisms to capture higher-order

*Work done while interning at Aven Inc.

Table Representation Learning Workshop at NeurIPS 2024

interactions between features. TabNet [2] applies sequential attention for implicit feature selection, where each layer emphasizes only a few features, and another method leverages both inter-sample and inter-feature attention [19].

While working with our industrial-scale financial tabular data, we found that XGBoost significantly outperforms standard two-layer neural networks. Adding additional fully connected layers further degrades performance, and other, more sophisticated network architectures struggle to meet baseline performance in our problem setting. However, a simple sparsely connected layer that organizes input data into meaningful groups surprisingly outperforms XGBoost by a reliable margin. Further analysis reveals that our high-dimensional financial data, unsurprisingly, contains highly correlated features. The sparsely connected layer appears especially effective in automatically reducing this correlation. We believe this method could be applicable in other domains with high-dimensional tabular data.

## 2 Our Method

### 2.1 Financial tabular data

Machine learning is widely applied in the consumer finance domain, for example, propensity models to acquire customers and risk models for underwriting. A person's financial history is undoubtedly valuable for predicting future financial behaviours. Consequently, numerous data providers featurize and standardize personal financial data for model-building purposes [8]. Example attributes in these datasets include the number of inquiries made over the past six months, the number of open credit cards, and the maximum balance over the past three months.

Because these datasets are designed for general purposes, they contain thousands of attributes that span various financial industries, time periods, and types of behaviours. For example, loans are categorized into auto, personal, mortgage, student, retail, and other types, while inquiry attributes cover different time frames, such as the number of inquiries in the past 14 days, 1 month, 3 months, and 6 months. Derogatory behaviours include categories like delinquency, charge-offs, collections, and bankruptcies.

### 2.2 The data correlation problem

These thousands of financial attributes are often correlated. For example, the 3-month inquiry count is a subset of the 6-month count, and charge-off accounts are frequently preceded by delinquency.

It is well known that correlated features can dramatically impair a model's performance, especially in neural networks compared to GBDTs. In neural networks, correlations can incorrectly amplify gradients, while decision trees are able to ignore redundant features altogether. This is likely a major reason why GBDTs often outperform neural networks, as data correlation is very common in high-dimensional tabular data.

However, reducing data correlation is nontrivial. First, with large-scale, high-dimensional data, the correlation among attributes typically follows a smooth distribution, making it difficult to establish a clear cutoff threshold for feature removal. This means that correlation will likely persist regardless of how many features we eliminate. Second, with sparse labels, even subtle differences between two correlated features can significantly impact model performance. Finally, as business objectives or macro conditions change, feature reduction analysis must be repeated, which is both tedious and prone to error. Ideally, we would like the model to automatically select features and eliminate correlations.

### 2.3 Our model architecture

Given information about correlated features, we can place those features side by side in the neural network's input layer. The first layer of this model consists of small MLPs for each group of input features, with an output size of $G$, a hyperparameter we can tune. We refer to this as a sparse layer, as only the neurons within the group are connected. Figure 1 illustrates a visualization of our proposed model, specifically in the case where we have only two hidden layers. In practice, any number of fully connected hidden layers can be used, as the model becomes a regular MLP after the input layer.

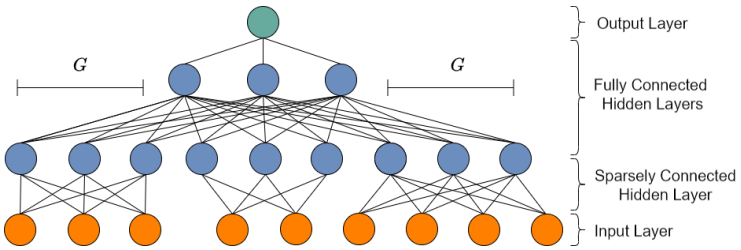


Figure 1: Our proposed model architecture.

## 3 Experiments

We evaluate all models on a proprietary dataset, with labels representing consumer propensity. The dataset consists of tens of millions of samples and thousands of features, with a positive sample ratio in the single digits. For all neural networks, numerical features were scaled to have zero mean and unit variance, categorical features were one-hot encoded, and missing values were mean imputed. To ensure a fair comparison, we conducted hyperparameter tuning for each model with at least 100 trials using Bayesian search with SyneTune [17]. All experiments were performed on AWS using GPU instances, with training taking a few hours for each model.

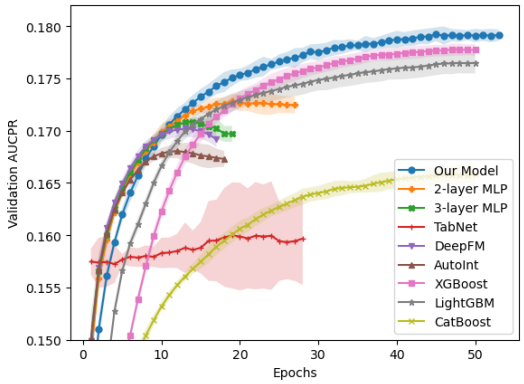


Figure 2: Various models' validation AUCPR on proprietary classification dataset.

### 3.1 Results

Figure 2 compares our model with several well-known models applicable to tabular data, including GBDT models (XGBoost, LightGBM, and CatBoost) [4, 11, 15] and neural networks (TabNet, DeepFM, and AutoInt) [2, 9, 20]. We employ open source implementations for all these models [7, 22, 4, 14, 23]. For GBDT models, one epoch on the x-axis represents 10 estimators. Error bars are reported with five runs per experiment to ensure consistency.

Overall, GBDT models such as XGBoost and LightGBM demonstrate outstanding performance, significantly outperforming two-layer fully connected neural networks. Adding an additional fully connected layer to the neural network actually diminishes performance; however, incorporating our sparse layer reliably surpasses the performance of both GBDT models.

These performance differences remain consistent across a variety of financial behaviour models we built using the same feature set, including risk, balance transfer, and refinancing models.

### 3.2 The effect of the sparse layer

To understand the effect of the sparse layer, Figure 3 presents the covariance matrices of the input data, the output from the first hidden layer of a three-layer network, and the output from our sparse layer. Both the hidden layer and our sparse layer contain the same number of neurons for a fair comparison.

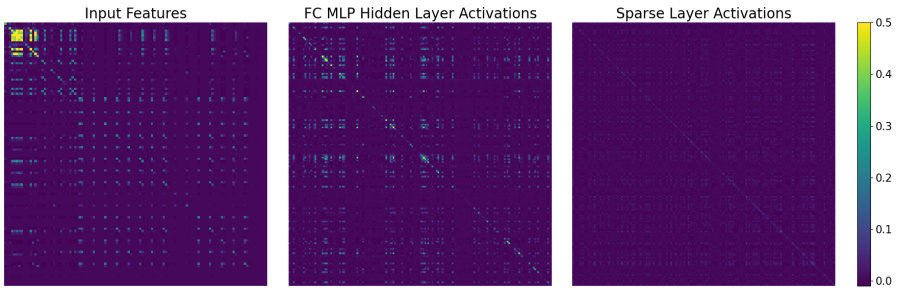


Figure 3: The covariance matrices of the input features, output of a 3-layer MLP first hidden layer, and the output of the sparse layer in our model. Note activation here refers to the value computed after applying the activation function.

The input data is highly correlated, as indicated by the bright spots. While the hidden layer of the three-layer network still shows many clusters, our sparse layer displays no visible clusters.

In our model, we group the input features for the sparse layer by their financial industries, such as automobile, mortgage, and personal loans. When we instead randomly group the features, the AUCPR drops from 0.1794 to 0.1730. We also experimented with other grouping criteria, including data types (balance, payment, time, percentage, etc.) and time periods, and found that their performance falls between industry-based grouping and random grouping, underscoring the importance of feature group selection for the sparse layer. Automated and optimal grouping strategies present an interesting direction for our future studies.

## 4 Related Works

Tabular data presents unique challenges for deep learning, as traditional neural networks have struggled to outperform GBDT methods like XGBoost, LightGBM, and CatBoost [4, 11, 15]. These tree-based methods excel at handling tabular data due to their ability to capture complex feature interactions and manage missing values effectively. However, a limitation of tree models is that they do not naturally incorporate domain knowledge, which can be crucial in scenarios where feature relationships are known but not explicitly provided to the model.

Recent efforts have explored adapting neural networks to work more effectively with tabular data. TabNet [2] leverages attention mechanisms to dynamically focus on relevant features. DeepFM [9] and AutoInt [20] aim to explicitly model feature interactions; the former combines factorization machines with deep neural networks to capture both first- and second-order interactions, while the latter employs self-attention mechanisms to model higher-order feature interactions without requiring manual engineering. Group-Connected Multilayer Perceptron Networks (GMLP) [10] also focus on exploiting feature groupings, but they achieve this by dynamically learning groups during the training phase using techniques like temperature annealing and entropy loss to encourage sparsity. However, these models rely on the network to discover feature groupings and relationships during training, rather than directly encoding domain-specific knowledge into the model.

Additionally, Sparsely-Connected Neural Networks (SCNN) [1] employs stochastic binary masks to generate sparsity, reducing the number of connections in neural networks to save memory and energy consumption while maintaining competitive accuracy. Our model shares the goal of introducing sparsity, but differs in that its sparsity is driven by the natural structure of the data — specifically, by grouping related features — rather than relying on random mask generation or pruning.

## 5 Conclusion

Sparsely connected layers represent a new approach to handling high-dimensional tabular data by explicitly incorporating domain-driven feature groupings into the neural network architecture. This positions it as a simple yet effective solution compared to more complicated neural network architectures for tabular data. Since large-scale high-dimensional tabular data often exhibits similar behaviours as our consumer financial data, we hope to see more use cases of our proposed solution.

## References


[1] Arash Ardakani, Carlo Condo, and Warren J. Gross. Sparsely-connected neural networks: Towards efficient vlsi implementation of deep neural networks, 2017. URL https://arxiv.org/abs/1611.01427.

[2] Sercan O. Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In Proceedings of the AAAI conference on artificial intelligence, volume 35, pages 6679–6687, 2021.

[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, 35(6):7499–7519, June 2024. ISSN 2162-2388. doi: 10.1109/tnnls.2022.3229161. URL http://dx.doi.org/10.1109/TNNLS.2022.3229161.

[4] 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, volume 11 of KDD '16, page 785–794. ACM, August 2016. doi: 10.1145/2939672.2939785. URL http://dx.doi.org/10.1145/2939672.2939785.

[5] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In Proceedings of the 1st workshop on deep learning for recommender systems, pages 7–10, 2016.

[6] ML Contests. Tabular Data Competitions - Winning Strategies. https://mlcontests.com/tabular-data/, March 2023. Accessed 2 September, 2024.

[7] Dreamquark-AI. Tabnet, 2023. URL https://github.com/dreamquark-ai/tabnet.

[8] Experian. Premier Attributes. https://www.experian.com/business/products/premier-attributes, 2024. Accessed 2 September, 2024.

[9] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: A factorization-machine based neural network for ctr prediction, 2017. URL https://arxiv.org/abs/1703.04247.

[10] Mohammad Kachuee, Sajad Darabi, Shayan Fazeli, and Majid Sarrafzadeh. Group-connected multilayer perceptron networks, 2020. URL https://arxiv.org/abs/1912.09600.

[11] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[12] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4): 541–551, 1989. doi: 10.1162/neco.1989.1.4.541.

[13] Haoran Luo, Fan Cheng, Heng Yu, and Yuqi Yi. Sdtr: Soft decision tree regressor for tabular data. IEEE Access, 9:55999–56011, 2021.

[14] Microsoft. Lightgbm, 2024. URL https://github.com/microsoft/LightGBM.

[15] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features, 2019. URL https://arxiv.org/abs/1706.09516.

[16] Steffen Rendle. Factorization machines. In 2010 IEEE International conference on data mining, pages 995–1000. IEEE, 2010.

[17] David Salinas, Matthias Seeger, Aaron Klein, Valerio Perrone, Martin Wistuba, and Cedric Archambeau. Syne tune: A library for large scale hyperparameter tuning and reproducible research. In International Conference on Automated Machine Learning, AutoML 2022, 2022. URL https://proceedings.mlr.press/v188/salinas22a.html.

[18] Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. Information Fusion, 81:84–90, 2022.

[19] 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.

[20] Weiping Song, Chence Shi, Zhiping Xiao, Zhijian Duan, Yewen Xu, Ming Zhang, and Jian 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. ACM, November 2019. doi: 10.1145/3357384.3357925. URL http://dx.doi.org/10.1145/3357384.3357925.

[21] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Advances in Neural Information Processing Systems, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[22] Shen Weichen, Zan Shuxun, Wang Ze, Zhang Wutong, Zhang Yuefeng, Huo Junyi, Zeng Kai, Chen K, Cheng Weiyu, and Tang. Deepctr-torch, 2022. URL https://github.com/shenweichen/DeepCTR-Torch.

[23] Yandex. Catboost, 2024. URL https://github.com/catboost/catboost.