Tabular deep learning: Understanding Unsupervised Pre-train Models

Chi Wang
School of Computer Science
University of Adelaide
chi.wang@adelaide.edu.au

Lingqiao Liu
School of Computer Science
University of Adelaide
lingqiao.liu@adelaide.edu.au

Abstract

Deep learning of tabular data is still an unanswered question, and the feasibility of deep learning tabular data is still worth exploring. Recent research has shown a number of supervised and unsupervised deep learning models for tabular data that have demonstrated performance comparable to traditional machine learning. However, unsupervised learning models that have been shown to be effective in CV, and NLP have not achieved desirable performance as they have not been tuned to existing tabular data. In this project, we aim to explore the feasibility of unsupervised learning of tabular data.

1. Introduction

Tabular data is the most widely used datatype in the real world and has a wide range of applications. Examples include house price forecasting, medical diagnostics, home price forecasting, loan approvals and robotics [12]. Currently, however, solving tabular data problems still relies on traditional machine learning methods, such as traditional dominance gradient boosting decision trees. Deep learning has made tremendous progress in recent years, especially in the task with unstructured data such as natural language processing and computer vision. Much research has extended this success to problems with structured data stored in tabular form. There are now significant advances in neural network architectures and training routines for tabular data, with performance comparable to that of traditional algorithms. Deep learning has unparalleled advantages over traditional algorithms, such as its powerful migration learning capability and the current craze for self-supervised learning, where its ability to extract interesting representations from data without relying on explicit supervised signals [12].

Self-supervised learning has achieved impressive performance in industrial computer vision and natural language processing pipelines, where models do not need to be trained with labelled data to obtain the task's features [1]. Some models are even thought to achieve better performance on larger unlabelled datasets. For example, the MoCo [7] self-supervised learning model has demonstrated performance on tasks such as semantic segmentation object detection NLP comparable to supervised models. Furthermore, a large number of experiments have shown that the features learned by self-supervised and supervised learning are very different [12], and as a result, it is often assumed that self-supervised learning models have better transfer capabilities. On the other hand, the ability to learn features from unlabelled data is crucial for tabular data to avoid the high overhead of labelling data, which, for example, has to be done manually in specialist fields such as medicine [9]. As a result, self-supervised learning for tabular data has become an open research problem.

However, according to some preliminary comparisons of existing self-supervised learning to supervised learning models by Levin et al. [12], the application of self-supervised learning brought a considerable performance drop compared to the tremendous improvement of CV and NLP. Potential reasons for this performance degradation could be the difference in application scenarios or the fact that existing self-supervised learning methods are not well tuned to tabular data. Therefore, this project aims to review the existing literature on self-supervised learning to gain a comprehensive understanding to explore the possibilities of unsupervised learning of tabular data.

2. Literature Review

2.1. Traditional machine learning for tabular data

The primary machine learning method for solving tabular data problems has traditionally been dominated by gradient boosting decision trees. It has been recommended as the method of choice for modelling tabular data. Gradient Boosting Decision Tree (GBDT) is a representative algorithm in the boosting family [10], which is an iterative decision tree algorithm consisting of multiple decision trees, with the conclusions of all trees summed up as the final answer. Thanks to its outstanding performance in various data mining and machine learning competitions, GDBT has been the primary method for tabular data for a long time. XGBoost [3], LightGBM [10], and CatBoost [13] are currently the most established GBDT libraries used by most practitioners or machine learning, with only minor differences in their implementation details and both of them perform well on most tasks. The status of traditional algorithms is being challenged by the powerful transfer learning and self-supervised learning capabilities of deep learning.

2.2. Deep learning tabular data

With revolutionary advances with deep learning models in computer vision and natural language processing, there is growing research interest in extending ideas such as transformer in deep learning and attention mechanisms to the domain of tabular data. Some recent studies have shown that some deep learning models demonstrate comparable competitiveness with traditional algorithms on specific tasks, which goes some way toward bridging the performance gap between traditional machine learning and deep learning.

A comprehensive approach to deep learning of tabular data is systematically discussed based on a survey published by Vadim et al [2]. The authors find that there is only a little work in the areas of supervised learning, unsupervised learning, and data synthesis that covers some of these aspects. However, no research explicitly applies deep neural networks to tabular data. In addition, the authors evaluate state-ofthe-art deep learning methods on real world data, for small datasets with heterogeneous data which less than 1 million samples, deep learning-based methods are still inferior to traditional machine learning algorithms. Only on large datasets with a large number of homogeneous data samples can some of the SOTA models outperform these classical methods. Yury et al [6]. empirically evaluated several state-ofthe-art deep learning methods, which include XGBoost, CatBoost, SNN [11] and so froths and provided a benchmark to measure model performance for tabular data on a wide range of datasets. Prior to this, deep learning methods for tabular data have not been adequately compared to traditional machine learning methods due to the lack of benchmark support. Therefore, there has been no clear conclusive conclusion as to whether deep learning methods outperform the traditional methods represented by GDBT. Interestingly, the authors demonstrate that tuned deep neural network models with ResNet-like architectures and established GBDT implementations still outperform most deep learning models.

Recently, Levin et al. published a study on transfer learning of several different deep models for tabular data, exploring the value of deep learning for solving tabular data and the applicability of transfer learning, and comparing supervised and supervised pre-training strategies [12]. Some key insights into deep learning for tabular data are presented. Notably, the authors also find that supervised pre-training leads to significantly better performance than the selfsupervised alternative. And the authors also mention that the tabular domain of the SSL approach has not been thoroughly explored. Self-supervised learning has been a popular research area in the past few years, which aims to improve the feature extraction ability of models by designing proxy tasks for unlabeled data and mining the representational properties of the data itself as supervised information. Well-known Selfsupervised learning methods such as SimCLR [4], approach based on contrast learning and MoCo [7] have revolutionised the field of computer vision, making self-supervised learning the dominant pretraining method. Recent research has shown that unsupervised (or self-supervised) pre-training can produce more transferable features than supervised pre-training [7]. As a result, there has been much research interest in trying to extend this success to tabular data. However, some preliminary investigations have concluded that the opposite is true for solving tabular data problems compared to the revolutionary improvements that self-supervised learning has brought in areas such as computer vision.

3. Project Goals and Challenges

For the project goal to be facilitated, this project aims to apply self-supervised learning methods to solve tabular data problems by analysing existing deep learning methods for tabular data. Self-supervised learning is becoming increasingly dominant in areas such as computer vision and natural language processing, so it is an open question whether this can be

Table 1. Samples form income 1995 dataset

Age	Sex	workclass	Education	marital-status	...	relationship	$\geq 50K$
32	Male	State-gov	Bachelors	Never-married		Not in family	1
18	Male	Private	HS-grad	Never-married	...	Own children	0
32	Female	Private	Doctorate	Married-civ-spouse	...	Wife	1
51	Male	Private	HS-grad	Divorced	...	Not in family	1
...	...	...	...	...	...	...	...

successfully extended to solve tabular data problems. Even though some recent research has shown that self-supervised learning may not necessarily lead to better performance than unsupervised learning, it is not enough to prove that self-supervised learning is not as successful in tabular data problems as it is in other areas, and this attempt to discover some valuable phenomena during the project will e a meaningful contribution to the field. This project goal will be addressed by reviewing recent self-supervised learning approaches and by developing experiments and tests based on the tabular neural network literature. From reviewing the contemporary literature on deep learning of tabular data, current models based on self-supervised learning are still proposed for tabular transformer architectures, such as Masked Language Models (MLM) pre-training using recently adapted tabular data [8] and a tabular version of contrast learning [14], which are not representative of SSL performance on tabular data. Since SSL is significantly less explored and tuned in the tabular domain than in CV or NLP, this results in poorer performance. Therefore, it was a major challenge for this project to better target and adapt the neural network to the tabular data. Time was of the essence for this project, so the number of self-supervised learning methods that can be attempted will be limited. Furthermore, as the field is relatively new and there are few references available, many original and creative approaches will need to be proposed to solve the difficulties. We assume that self-supervised learning methods can be adapted well to the tabular data problem and bring the same performance gains as in other areas.

4. Methodology

4.1. Self-supervised learning in language model

The initial attempt on self-supervised training for tabular data involved designing a pre-training task. Our exploration began with a masked language model of linguistic self-supervised models, an approach first proposed by the designers of the BERT model, Devlin et al.[5] BERT is a linguistic representation model. The model structure is a stack of encoders in Transformer, which is a 2-stage framework for pretraining, and for finetuning on individual specific tasks. In the pre-training phase, BERT introduced the Masked language model. Specific approach is randomly masks or replaces any word or phrase in a sentence, and then allows the model to predict the masked or replaced part by the context, followed by the calculation of the loss of the masked part with the original input only. The advantage of this is that BERT does not know which word is masked, and any word could be the one that is being replaced. This forcing the model not to rely too much on the current word at the moment when encoding it, but to consider its context and even correct it according to that context. MLM works on linguistic data because the language model inherently carries information about contextual semantics as well as temporal order. Whether such contextual semantic information is also present in tabular data then becomes a question worth investigating.

The data in Table 1 are taken from income 1995, a dataset that investigates whether an individual has an income greater than $50,000. On closer inspection, age and education, marital-status, and relation feature are likely to be related; if a person is less than or equal to 18 years of age, there is a high probability that he or she is high school graduated education, is in the never-married status, and so forth. Therefore, there may be some contextual relationships in the tabular data and a masked tabular model might help the model to learn this language-like contextual relationship form the tabular data.

4.2. Mask and prediction

In the MLM pre-training task of the BERT, words that need to be masked are masked with a special token [mask], and the same embedding layer is used for embedding. However, this approach of sharing the same embedding layer is not applicable to tabular

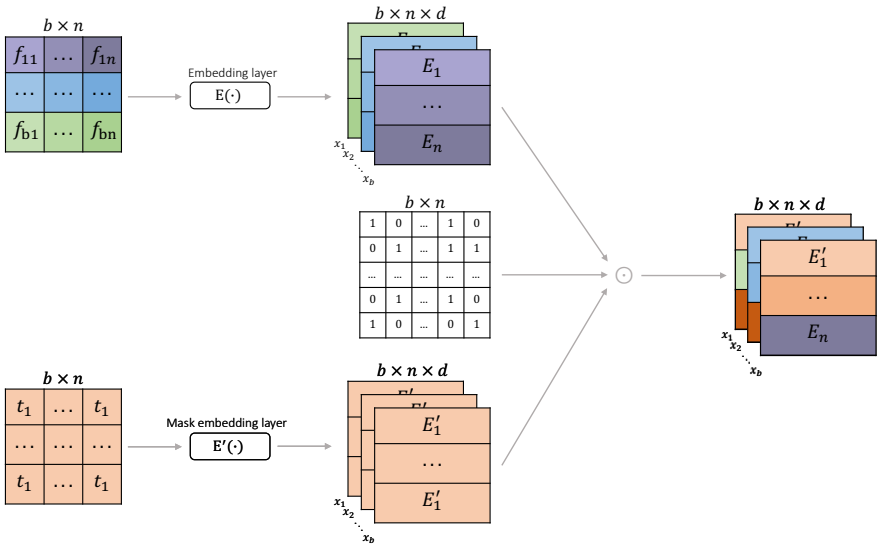

Figure 1. Unified token embedding in latent space.

data. The reason for this is that, assuming we use a special value such as 0 or -1 as the mask value, this may work for categorical data, but for continuous data it becomes a tricky problem to ensure that the original value of the masked feature is different from the mask value. Therefore, an additional learnable mask token embedding layer is needed to encode the mask token for tabular data. We therefore devise two possible mask strategies.

4.3. Mask in tabular model

Similar to the use of [mask] tokens for masking in BERT. Firstly, the input and token are fed into the embedding layer and the mask embedding layer respectively to get their respective representations in the latent space. A position matrix of zeros or ones is generated, with zero representing the position where the feature embedding will be replaced by the mask embedding and one representing the retention. Based on this position matrix, the representation of the original data is masked with the embedding.

For the second strategy, the only difference is that we want to use different mask embedding for different positions. Therefore, different tokens with the same number of features are fed into the embedding layer and masked according to the position matrix. This is a more accurate approach which expects mask embedding to learn how to mask over features at different locations.

4.4. Network design

The network is designed using the encoder component of the Transformer [15], which consists of multiple layers of encoders. Each encoder consists of two sub-layers, the self-attention layer, and the feed forward network. This is shown in (I) of Figure 3. Each encoder has the same structure but have the different weighting parameters. A batch of samples first feds into a multi-head self-attention layer. This allows the encoder to use information from other features in the input samples when encoding a particular feature. Meanwhile, each multi-head mapping the input to a different sub-representation space, which allows the model to focus on different positions in the different sub-representation spaces.

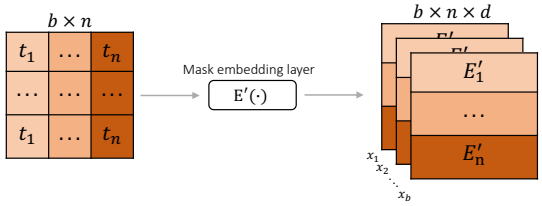

Figure 2. Multiple masking in latent space.

Each layer has a skip connection and layer normalisation, which returns a set of attention matrices that are merged with the actual input and performs layer/batch normalisation. Finally, a feed-forward network, consisting of two fully connected layers, the first with a ReLU activation function, passes the transformed attention values to the next decoder layer

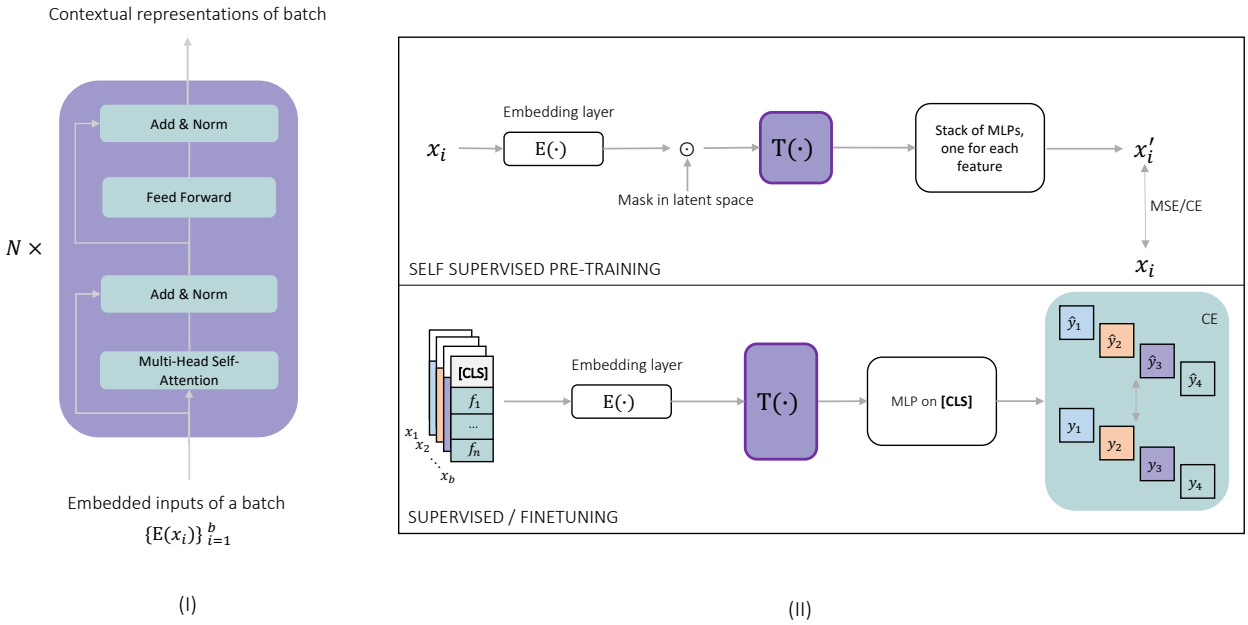

Figure 3. The encoder architecture (I). Unsupervised pre-taining and Supervised training/Finetuning pipelines (II).

or linear layer on top.

4.5. Self supervised pre-training

In the pre-training phase, a batch of sample is first fed to the embedding layer for embedding to obtain a representation of the features in the latent space. In the latent space we perform mask and prediction operations such as mask with one token or mask with multiple tokens as mentioned in the previous section. An encoder consisting of N self-attention blocks receives the input to compute the contextual representation of the batch. The structure of MLP has been changed to accommodate tabular data. In BERT, a word dictionary is used to represent all the different tokens, so a single classifier can satisfy the classification task of MLM. However, the table data consists continuous data and categorical data, and the same classifier is clearly not sufficient for the classification needs of MLM. Therefore, we designed different classification layers for different features. For continuous features an MLP with one output is sufficient for continuous data prediction. For categorical features we use a stacking of multiple MLPs so that the model can classify and predict multiple possible outputs for different categorical features. We use mean square error loss as the optimization objective for continuous data and cross-entropy loss for categorical data.

4.6. Supervised / finetuning

Once the pre-training phase is complete. We take L samples from the dataset and fine-tune the model on the target prediction task. The ratio of positive to negative samples is 1:1 for the L samples that are randomly drawn. See Figure b for a detailed flow. For prediction, inspired by the BERT, where we add a [CLS] token for classification at the beginning of each sample. As shown in the figure, a batch of samples with labels are fed into the embedding layer to obtain embeddings and the model learns the contextual representation based on these feature embeddings. In the prediction stage, only the embeddings tagged with [CLS] are fed into the classifier to compute the predicted output. The prediction classifier consists of a simple MLP with a hidden layer with ReLU activation.

5. Experiment Setup

The performance of the model was evaluated on six tabular datasets in supervised and self-supervised scenarios.

5.1. Datasets

The datasets we use are all binary task datasets, which cover real data from various fields such as finance, biology, security, etc. These datasets were

Table 2. Experiment results and relevant information of datasets. Green represents the performance improvement, red for the performance for the performance degradation, bold indicates the best accuracy among thes unsupervised pre-training compare to the supervised training on the corresponding datasets.

Data size	7,043	1,055	32,561	12,330	45,211
Features	20	41	14	17	16
Cat	17	0	8	2	9
Con	3	41	6	15	7
Name	Blast char	Qsar bio	Income 1995	Online shopper	Bank market
% of Positives	26.54	33.74	24.08	15.47	11.70
Total trainset	82.40	93.32	91.69	93.40	93.31
50	78.57	87.67	85.41	84.61	82.03
50 + pt(unified)	78.33 (0.24)	**87.79 (0.12)**	**87.45 (2.04)**	84.20 (0.41)	83.43 (1.40)
50 + pt(multi)	**78.68 (0.11)**	86.82 (0.85)	86.89 (1.48)	84.48 (0.13)	**83.95 (1.92)**
200	79.62	90.91	87.27	89.57	85.69
200 + pt(unified)	80.13 (0.51)	91.00 (0.09)	**88.67 (1.4)**	89.17 (0.40)	**86.92 (1.32)**
200 + pt(multi)	**80.29 (0.67)**	**91.15 (0.24)**	86.82 (0.45)	**91.32 (0.75)**	85.41 (0.28)
500	80.64	91.76	89.47	91.36	88.21
500 + pt(unified)	80.96 (0.32)	**91.87 (0.11)**	89.53 (0.06)	91.17 (0.19)	89.19 (0.98)
500 + pt(multi)	**81.21 (0.57)**	91.75 (0.01)	**89.74 (0.21)**	**93.28 (2.11)**	**89.24 (1.03)**

chosen because they are diverse, have sample sizes ranging from 1000 to 45,000, and contain multiple continuous data or categories of data, both uniformly and unevenly distributed. Some datasets have a balanced distribution of positive and negative samples, while others have a highly skewed class distribution of positive and negative samples. In the pre-processing stage of the dataset all continuous data is first normalized, and all categorical features are encoded with labels.

5.2. Baseline

The comparison baseline is the supervised training of the corresponding dataset. All datasets are trained supervised. In the pre-training stage, all data are pre-trained with a mask and prediction task, in which the pre-trained label is unknown and the optimization objective is to minimize the cross- entropy and mean square error loss of the output with respect to the original sample. As the prediction tasks are binary and some of the datasets have a highly unbalanced distribution of positive and negative samples, the metric used to evaluate the performance of the model is the AUC, which has the advantage of being insensitive to the distribution of positive and negative samples.

5.3. Training

All datasets were divided into 65%, 15% and 25%, training set, validation set and test set, with a batch size of 256. The AdamW optimization function was used for all models including the pre-training and fine-tuning stages, with parameters set to $\beta 1 = 0.9$, $\beta 2 = 0.999$, decay = 0.01 and a learning rate of 0.0001. For the mask ratio, too small a ratio resulted in the model not learning the contextual relationships, too large a ratio resulted in the dataset being corrupted, with set of experiments mask ratio of 0.5 usually provide the promising results.

6. Results

6.1. Mask with unified token

Experiment results can be found in Table 2. The initial results of using the unified token for mask are promising and provide some performance gains after the mask and prediction pre-training task. Although pre-training with the full dataset and then fine-tuning with limited data sample on multiple datasets yielded higher performance metrics than supervised training with corresponding dataset, these performance gains were tiny. For some larger datasets such as income

Table 3. Experiment results on the multiclass classification tasks.

Data size	1,728	1,728
Features	22	7
Cat	22	7
Con	0	0
Name	car-evaluation	car
Total trainset	81.3187	78.2967
50	73.35	73.90
50 + pt(unified)	70.88 (2.47)	73.35 (0.55)
50 + pt(multi)	71.70 (1.65)	73.08 (0.82)
200	83.52	77.75
200 + pt(unified)	**84.89 (1.37)**	81.04 (3.29)
200 + pt(multi)	81.59 (1.93)	**83.79 (6.04)**
500	82.69	75.82
500 + pt(unified)	**85.99(3.33)**	77.47 (1.65)
500 + pt(multi)	82.14(0.55)	**79.95 (4.13)**

1995, online shopper, bank market, the performance improvement is more obvious because more training samples are available for the model to learn the representations. The performance of the model with the same mask token is more consistent for most of the datasets, but and degradation for some of the datasets was observed. The reasons for the performance degradation are not clear at this time, as the number of datasets involved in the experiments is still not much enough.

6.2. Mask with multiple token

Using different mask tokens for different feature is a more accurate strategy of masking. Learnable token embedding matrices can be used to learn how to mask features for the feature at different locations. On most datasets, mask with different token embedding impressively consistently achieve reasonable accuracy, although they fail to outperform supervised training on the corresponding training set. This method seems to be more prone to performance degradation than the mask strategy using the unified token. Similarly, the performance improvement is more obvious for larger data sets when the number of samples is limited. With these two strategies, masking and predictive pre-training has identified some contextual representations present on the tabular data, and resulted in some performance gains, but a huge enough outperformance was not observed compared to supervised training on the corresponding dataset.

6.3. On the multiclass classification task

In addition to the binary classification dataset, we also performed the same experiment on the multi classification dataset. The result can be found in Table 3. Both unified and multi masking demonstrated the same performance degradation when the number of samples was limited (50). When a certain number of samples were available (200, 500), a significant improvement was observed, with a maximum improvement of 6 percentage points, which outperformed the corresponding supervised learning training. On the other hand, the prediction accuracy remained within a reasonable range, even though performance degradation occurred.

7. Code link

Code available at: `https://github.com/iamabin/AdvTopic2022`

8. Conclusion

In this project we explore pre-training methods for tabular data. Despite the fact that tabular data is an extremely common data format used in the real-world task, deep learning methods still lag behind tree-based boosting methods in production. With mask and prediction, our tables can often help neural model models learn certain contextual relationships. This model provides an improvement on a widely used domain dataset, and our approach performs well on the various tabular datasets studied here. However, real-world tasks can contain a wide range of datasets that can be very noisy or unbalanced, so further experiments are required to evaluate its effectiveness .

References

[1] Luca Biggio and Iason Kastanis. “Self-supervised pre-training on industrial time-series”. In: *2021 8th Swiss Conference on Data Science (SDS)*. IEEE. 2021, pp. 56–57.

[2] Vadim Borisov et al. “Deep neural networks and tabular data: A survey”. In: *arXiv preprint arXiv:2110.01889* (2021).

[3] Tianqi Chen et al. “Xgboost: extreme gradient boosting”. In: *R package version 0.4-2* 1.4 (2015), pp. 1–4.

[4] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[5] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805* (2018).

[6] Yury Gorishniy et al. “Revisiting deep learning models for tabular data”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 18932–18943.

[7] Kaiming He et al. “Momentum contrast for unsupervised visual representation learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9729–9738.

[8] Xin Huang et al. “Tabtransformer: Tabular data modeling using contextual embeddings”. In: *arXiv preprint arXiv:2012.06678* (2020).

[9] Ashish Jaiswal et al. “A survey on contrastive self-supervised learning”. In: *Technologies* 9.1 (2020), p. 2.

[10] Guolin Ke et al. “Lightgbm: A highly efficient gradient boosting decision tree”. In: *Advances in neural information processing systems* 30 (2017).

[11] Günter Klambauer et al. “Self-normalizing neural networks”. In: *Advances in neural information processing systems* 30 (2017).

[12] Roman Levin et al. “Transfer Learning with Deep Tabular Models”. In: *arXiv preprint arXiv:2206.15306* (2022).

[13] Liudmila Prokhorenkova et al. “CatBoost: unbiased boosting with categorical features”. In: *Advances in neural information processing systems* 31 (2018).

[14] Gowthami Somepalli et al. “Saint: Improved neural networks for tabular data via row attention and contrastive pre-training”. In: *arXiv preprint arXiv:2106.01342* (2021).

[15] Ashish Vaswani et al. “Attention is all you need”. In: *Advances in neural information processing systems* 30 (2017).