

LEARNED MIXING WEIGHTS FOR TRANSFERABLE TABULAR DATA AUGMENTATION

Anonymous authors

Paper under double-blind review

ABSTRACT

We present an architecture-agnostic method for tabular data augmentation, which mixes pairs of samples from the training set. The mixing procedure is based on a set of per-feature weights that are assigned by a learned network g that is separate from the primary classification model f . The features are selected between the two samples at random, and the sum of the weights that g assigns the features that are selected from each sample determines the mixing of the target label. g itself is trained based on two loss terms, one that encourages variability in the assigned weights between the features and one that encourages, for every training sample, the model f to be agnostic to the features for which g assigns low weights. Our experiments show that this learned data augmentation method improves multiple neural architectures designed for tabular data. Even more notable is that the network g that was trained on an MLP produces mixed samples that improve non-differentiable methods, including classical methods and gradient-boosting decision tree methods. This is done without any further tuning and with the default parameters of the classifiers. The outcome achieved this way, using the cutting-edge CatBoost method now represents the state of the art.

1 INTRODUCTION

Neural models are generally less successful than tree-based models in the domain of tabular data (Grinsztajn et al., 2022; Shwartz-Ziv & Armon, 2022; Zhu et al., 2023). Motivated by the hypothesis that this stems from the discontinuous input domain of this type of unstructured data, we develop a data mixing (Guo et al., 2019; Huang et al., 2021) technique for performing train-time augmentation.

In contrast to previous mixing techniques, our method employs a learned feature selection model g on top of the primary prediction model f . Network g assigns for every sample x a different importance score per each vector element. The main loss through which g is trained requires that the sample $g(x) \odot x$ obtained by weighing each feature by the associated weight, provides, when f is applied to it, the same logits $f(g(x) \odot x)$ as the logits $f(x)$ of the original sample. Another loss term encourages g to assign different values to different features, preventing uniformity.

When our method mixes two samples, seven-eighths of the features are selected randomly from the first sample, and the rest from the second. The influence of a sample on the classification loss of the virtual sample is determined by the sum of the weights that g assigns to the features it contributes. Put differently, a sample’s label contributes more to the loss of f if more relevant features from it were copied to the virtual sample.

In all of our experiments, g is a vanilla MLP. For both the MLP architecture and the transformer architecture in our experiments, augmenting the dataset with g improves performance by a sizable gap. A further improvement in accuracy is obtained when g is applied at test time. In this case, no mixed samples are generated. Instead, we interpolate between the logits $f(x)$ and the logits of its weighted version $f(g(x) \odot x)$.

The training of g requires a gradient signal from f . When f is non-differentiable one can use a surrogate network to obtain gradients (Athalye et al., 2018) or train g by using gradient-free methods such as REINFORCE (Williams, 1992). However, as we show, the readily available solution of using transfer learning is highly effective. In other words, g that was trained for one classifier f can be used to create virtual samples for another classifier f' .

We, therefore, provide results for gradient boosting decision tree methods and classical methods, that learn after augmenting the training set via network g , which was optimized for an 8-layer MLP classifier, and evaluated using the test-time interpolation with the sample $g(x) \odot x$ mentioned above.

Here, too, we observe an improvement across all methods, with the exception of a low-capacity linear classifier. Specifically, by applying the pre-trained g to augment the data used for training CatBoost, which is the current leading method, a new state-of-the-art is obtained. This is achieved while adhering to the same train/test protocol and with the default set of parameters of CatBoost.

Our main contributions are summarized as:

- Presenting a learned mixup method that is suitable for unstructured data and operates in the input domain. The method and the optimization problem it employs are both novel.
- The application during inference of the importance-assigning function learned by the mixup method to further improve results.
- Using transfer learning for augmenting tabular data in order to apply our method to black-box classifiers, including gradient boosting decision trees.
- Improving multiple tabular data classifiers, including the improvement of the state of the art for tabular data classification by a sizable margin.

2 RELATED WORK

Data mixing-based augmentation methods sample random pairs of examples from the training set and create virtual samples by combining these. The label of the generated sample is a fuzzy label that combines those of the mixed samples. The seminal MixUp method of Guo et al. (2019) linearly interpolates between two different samples and, using the same weights (one per sample), the fuzzy labels are computed.

Mixup does not work well on tabular data, and Contrastive Mixup (Darabi et al., 2021) aims to improve it by projecting the data onto a manifold where the mixture would be performed. This is similar to Manifold Mixup (Verma et al., 2019) except that a contrastive objective is used. In our experience, these methods do not boost classification enough to be competitive with the state of the art. We also note that, unlike these contributions, our work is applied in the original input space, which is crucial for transferring the augmentation to other methods.

Chen et al. (2023) employ dual mixup – in the feature space and the hidden representation manifold – to improve the training of their transformer architecture. This mixup is not learned, and while the authors claim to outperform CatBoost, this is achieved on a set of datasets that do not constitute a known benchmark and, unlike our method, is shown to improve a specific architecture.

Cutmix (Yun et al., 2019) relies on the structure images. It extracts a rectangular box from one image and pastes it onto the second. The resulting label is proportional to the area of the rectangle. Since the area is a crude estimate of the importance of an image region, SnapMix (Huang et al., 2021) replaces it with the sum of the CAM activations (Zhou et al., 2016) within the extracted and the masked patches. As we demonstrate in our ablation study, replacing network g with a relevancy score derived from backpropagation is not as effective for tabular data as our method.

Learned data augmentation techniques have been widely applied in computer vision, e.g., Cubuk et al. (2019); Zoph et al. (2020). However, we are not aware of any learned augmentation techniques for tabular data. Similarly, we are not aware of any learned mixup technique.

Learning the data augmentation network on the same training dataset can be compared to pertaining techniques Yoon et al. (2020); Ucar et al. (2021); Somepalli et al. (2021); Bahri et al. (2021); Majmundar et al. (2022); Rubachev et al. (2022); Wang & Sun (2022); Zhu et al. (2023), which also perform a preliminary training step, although with the completely different goal of initiating the network weights in a favorable way. However, our method is much more efficient than pretraining techniques and can be applied outside the realm of deep neural networks. Being from an orthogonal domain of performance-enhancing techniques, it can be applied together with pretraining methods.

Gradient Boosting Decision Tree (GBDT) methods (Friedman, 2001) are currently considered the tool of choice in tabular data classification (Gorishniy et al., 2021; Shwartz-Ziv & Armon, 2022;

Grinsztajn et al., 2022; McElfresh et al., 2023). These include XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017), and CatBoost (Prokhorenkova et al., 2018), as notable samples.

The leading position of GBDT is despite a considerable effort to develop suitable deep learning methods, including architecture search (Kadra et al., 2021; Egele et al., 2021; Yang et al., 2022), specifically designed transformers (Huang et al., 2020; Wang & Sun, 2022; Gorishniy et al., 2021), and methods that are inspired by decision trees (Popov et al., 2019; Hazimeh et al., 2020; Arik & Pfister, 2021; Katzir et al., 2020), to name a few approaches and a few representative samples of each approach.

In contrast to these contributions, we are architecture-agnostic and offer a method for training data augmentation. In order to emphasize that our contribution is not architectural, in our experiments we use a simple feed-forward network for g .

3 METHOD

We are given a training dataset $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ of n training samples, with feature vectors $x_i \in \mathbb{R}^d$ and labels $y_i \in \mathcal{Y}$. Our goal is to train a model $f : \mathbb{R}^d \rightarrow \mathcal{Y}$ that generalizes well. In our method, we also train an auxiliary network $g : \mathbb{R}^d \rightarrow \{0, 1\}^d$, which, given a sample x , assigns a weight to each coordinate k in $[d] = \{1, 2, \dots, d\}$. These weights are positive and sum to one, e.g., obtained by employing a softmax as the last layer of a neural network implementing g . Using superscripts as the index vector elements, and defining $m = g(x)$, this is written as $\forall k, 0 \leq m^k \leq 1$ and $\sum_{k=1}^d m^k = 1$.

The training procedure mixes two random training samples x_i and x_j to create a virtual sample \bar{x} . The mixing is entirely random and involves a random selection of k indices in $[d]$. These k indices are copied to the virtual sample from the second sample, while the rest are obtained from the first. Throughout our experiments, k is one-eighth of the number of features, or, more precisely, $k = \max(1, \lfloor d/8 \rfloor)$ (rounded and never zero). Therefore, the first sample selected dominates the mixture.

In order to weigh the importance of x_i and x_j in \bar{x} , we sum the weights assigned by network g that are associated with the relevant coordinates of each sample. Let $m_i = g^c(x_i)$, and $m_j = g^c(x_j)$, where g^c is a copy of g such that gradients from the loss \mathcal{L}_f below do not backpropagate to network g . We define

$$\begin{aligned} \gamma_i &= \sum_{u=a}^b m_i^u \\ \gamma_j &= \sum_{u \in [d] \setminus \{a..b\}} m_j^u. \end{aligned} \tag{1}$$

We focus on classification problems and the classifier f outputs a vector of logits. Denote the logits obtained for the virtual sample as $\bar{y} = f(\bar{x})$. The primary model f is trained using the cross-entropy loss (CE) of both labels y_i and y_j , weighted by γ_i and γ_j :

$$\mathcal{L}_f = \gamma_i \text{CE}(\bar{y}, y_i) + \gamma_j \text{CE}(\bar{y}, y_j). \tag{2}$$

This loss is applied over batches of random virtual samples \bar{x} , created on-the-fly from the training set.

Meanwhile, the main loss of the auxiliary network g encourages the classification obtained on a weighted sample $dg(x) \odot x$, where \odot denotes the Hadamard product, to be similar to that obtained on the unweighted sample (the factor d ensures that the L1 norm of the sample $g(x) \odot x$ does not decrease in comparison to x since g returns a pseudo-probability). This loss is expressed as:

$$\mathcal{L}_g = \text{CE}(f^c(x), f^c(dg(x) \odot x)), \tag{3}$$

where f^c is a frozen copy of f , such that the gradients from this loss do not propagate to network f . The loss \mathcal{L}_g is applied over batches of random samples from the original training set.

Recall that the values in $m = g(x)$ are all positive and sum to one. In order to encourage g to select important features and not assign all weights uniformly, we add the following loss term:

$$\mathcal{L}_m = -\max(m). \tag{4}$$

This loss is computed on the same samples that are used for the loss \mathcal{L}_g .

The overall training loss for g is given by

$$\mathcal{L}_{gm} = \mathcal{L}_g + \lambda \mathcal{L}_m, \quad (5)$$

where the parameter λ scales the non-uniformity loss term, since it is on a different scale. In all of our experiments $\lambda = 0.1$. Note that none of the terms in \mathcal{L}_{gm} propagates to f , and this classifier is co-trained with g simply by observing the virtual (mixed) samples \bar{x} .

During inference, we would like to benefit from the feature weighting. Therefore, we mix $f(x)$ and $f(g(x) \odot x)$ and the predicted label for an input sample x is given by the combined logit vector:

$$\hat{y} = f(x) + \alpha f(dg(x) \odot x), \quad (6)$$

where in all of our experiments $\alpha = 0.1$.

4 EXPERIMENTS

In our experimental evaluation, we assess the efficacy of our novel approach across a range of supervised tabular learning tasks, encompassing both binary and multiclass classification challenges. To ensure the reliability of our findings, we meticulously follow a consistent experimental protocol across all scenarios. We leverage the best-performing model checkpoint, determined based on validation scores during training, to evaluate the model’s performance on test datasets.

4.1 DATASETS

To conduct our experiments, we employ the publicly available OpenML-AutoML Benchmark (AMLB), a recently introduced benchmark tailored for automated machine learning. This benchmark comprises 84 tabular classification tasks and 33 regression tasks. We focus on classification tasks and leave regression tasks for future work.

AMLB’s standard practice involves reserving 10% of the tabular data for testing. The remaining data, is allocated 87.5% (7/8) to the training set and 12.5% (1/8) for validation purposes. Five independent trials are performed, each involving distinct test folds, across all tabular datasets. To maintain consistency, each trial adheres to the same data split strategy.

4.2 BASELINES

We employ multiple neural models, two leading GBDT models, and a few classical methods. The neural models include an MLP and a tabular transformer. Each baseline is subjected to the same experimental conditions.

CatBoost (Prokhorenkova et al., 2018) is a gradient-boosting algorithm tailored for tabular data classification. It excels in handling categorical features without extensive preprocessing, making it a strong contender for tabular data tasks.

XGBoost Chen & Guestrin (2016) is a widely adopted gradient-boosting framework that offers speed and accuracy in handling structured data.

Multi-Layer Perceptron (MLP) is a neural network architecture with multiple layers of nodes. Our adaptation of the model featured eight layers with a hidden dimension of 512 and a GLU activation.

Tab-Transformer Huang et al. (2020) is a recent deep learning architecture designed explicitly for tabular data. Leveraging transformer-based mechanisms, it captures complex relationships within structured data.

K-Nearest Neighbors (KNN) is a straightforward classification algorithm that assigns labels based on the majority class among the nearest neighbors. We employ it with $k = 5$.

Logistic Regression is a linear classification algorithm that models the probability of binary outcomes.

Due to a lengthy run-time, we were not able to add FT-Transformer (Gorishniy et al., 2021) by the time of submission.

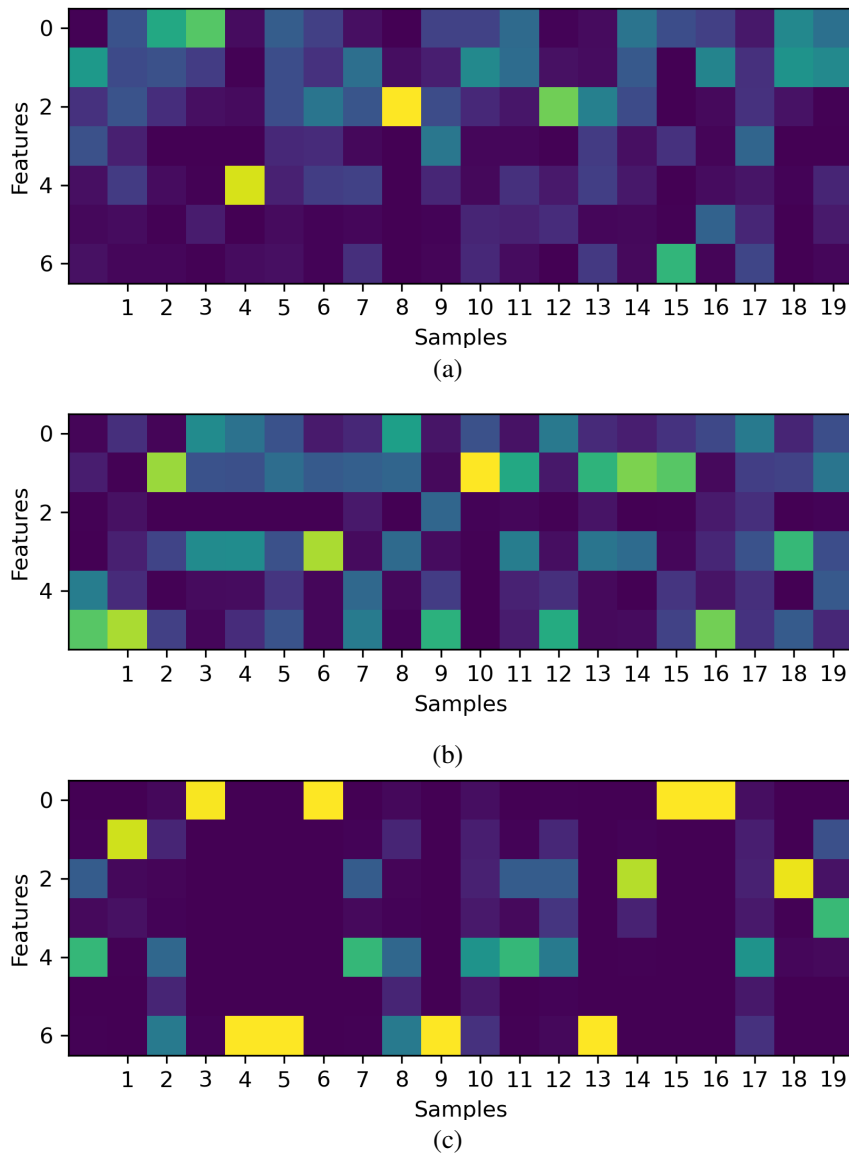


Figure 1: The weights associated by g trained for an MLP classifiers to random samples of the (a) artificial-character dataset, (b) the chess dataset, and (c) the LED-display-domain-7digit dataset.

4.3 IMPLEMENTATION DETAILS

Network g employs an MLP architecture, and is trained entirely from scratch. The network’s input dimension corresponds to the number of features in the tabular data. It comprises of an 8-layer MLP with a hidden dimension set to be 512.

The AdamW optimizer is used with a learning rate of 0.0003. The networks undergo training for a duration of 100 epochs, employing a batch size of 128 throughout the training process, utilizing the NVIDIA GeForce RTX 2080Ti GPU for accelerated model running.

For the purpose of obtaining g for transfer learning, we retrained f and g for 4000 steps on a given training set. The architecture we used for f was identical to that of g . This training effort is independent of the dataset size (only on the number of features) and takes less than 90 seconds per dataset on a mid-range GPU. This is often negligible in comparison to the runtime of CatBoost.

Table 1: Performance of various tabular data classification methods on the AMLB benchmark. *Indicates that the g model used is the one of the MLP and was not trained for these methods.

Algorithm	Class	Rank				Accuracy	
		min	max	mean	median	mean± sd	median
CatBoost (Prokhorenkova et al.)	GBDT	1	12	4.2±2.4	4	0.84±0.17	0.91
CatBoost + Ours*	GBDT	1	9	3.0±2.2	3	0.86±0.16	0.93
XGBoost (Chen & Guestrin)	GBDT	1	12	5.9±2.9	6	0.84±0.17	0.92
XGBoost + Ours*	GBDT	1	13	4.9±3.0	4	0.85±0.17	0.92
MLP	Neural	1	13	7.4±3.0	8	0.77±0.23	0.85
MLP+SnapMix (ablation)	Neural	1	13	5.8±3.1	6	0.80±0.22	0.90
MLP+Ours	Neural	1	13	5.4±3.4	5	0.81±0.20	0.88
TabTransformer (Huang et al.)	Neural	2	13	10.7±2.8	12	0.68±0.26	0.71
TabTransformer+Ours	Neural	1	13	9.3±3.7	11	0.71±0.27	0.75
k Nearest Neighbours	Classical	3	13	8.8±2.6	9	0.77±0.21	0.76
k Nearest Neighbours + Ours*	Classical	1	13	7.8±3.4	8	0.79±0.21	0.86
Logistic Regression	Classical	1	13	8.7±3.1	9	0.74±0.22	0.81
Logistic Regression + Ours*	Classical	1	13	9.1±3.4	9	0.75±0.23	0.81

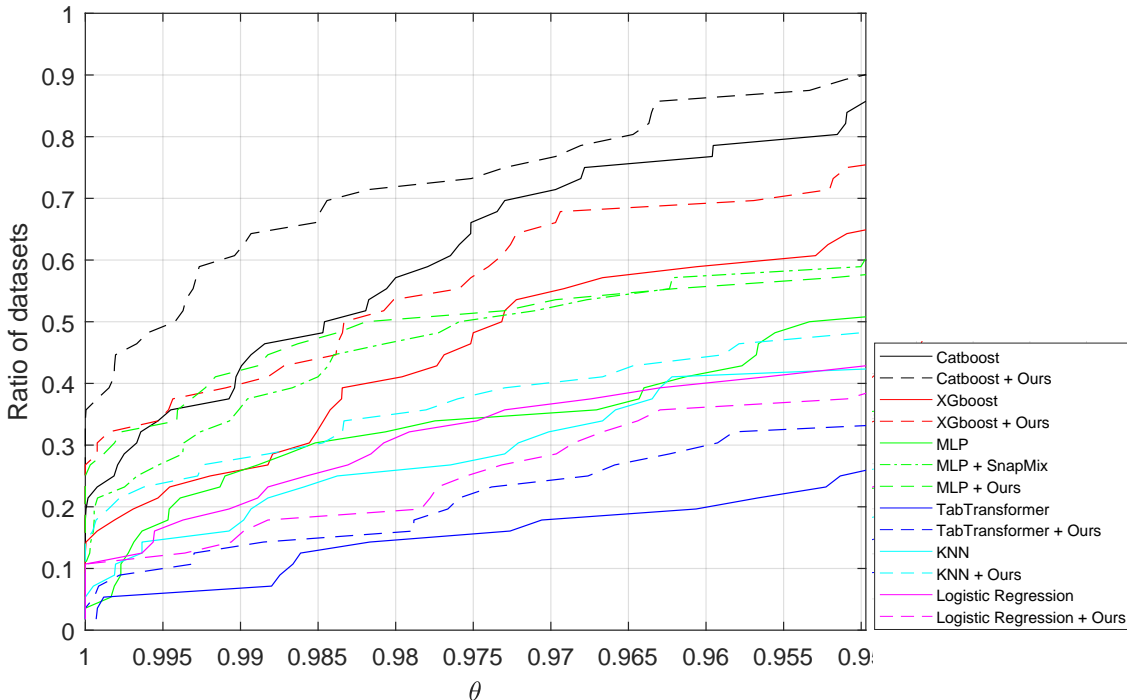


Figure 2: A Dolan-More performance profile (Dolan & Moré, 2002) comparing accuracy scores of the various methods on the AMLB benchmark. For each method and each value of θ (x -axis), the graph presents the ratio of datasets for which the method performs better or equal to θ multiplied by the best accuracy score for the corresponding dataset. An algorithm that achieves the best score on all datasets would reach the top left corner of the plot for $\theta = 1$. The combination of CatBoost with our method yields better results than all baselines, and CatBoost is the 2nd best method.

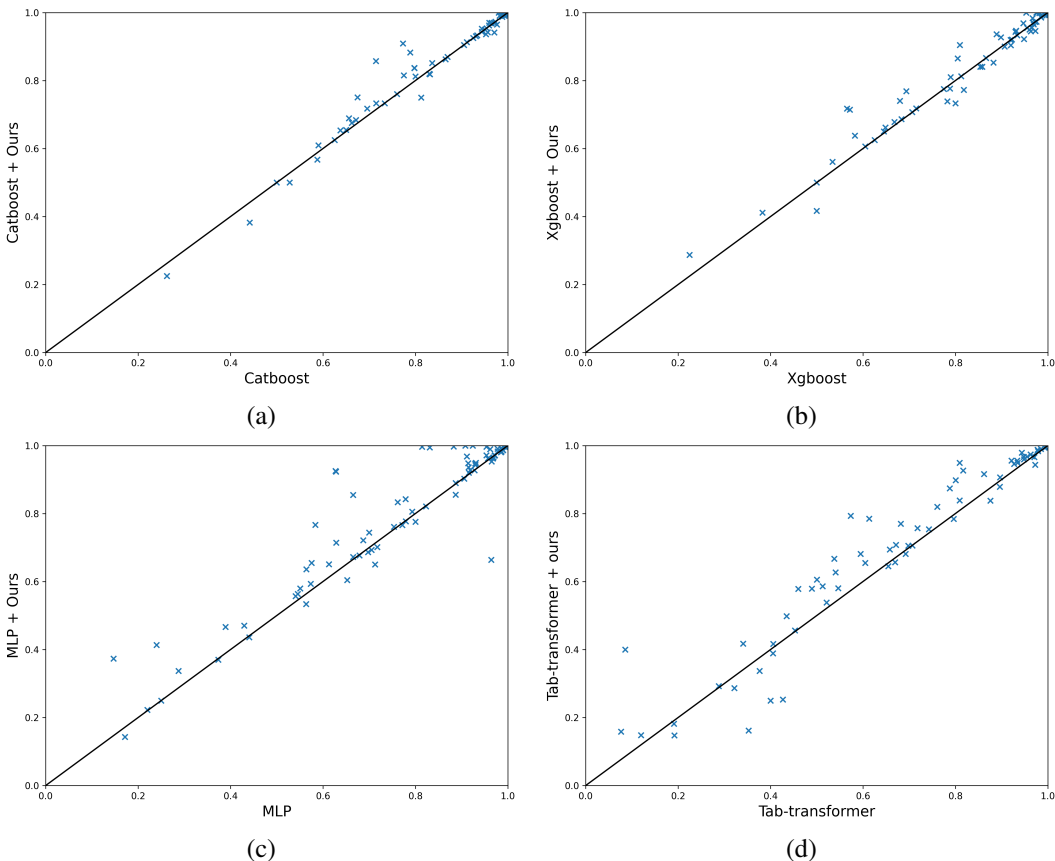


Figure 3: A comparison of the accuracy per tabular dataset in the AMLB benchmark with (y-axis) and without (x-axis) our method. The augmentation network g for the boosting methods is based on transfer learning, while the classifier f for neural methods is co-trained with g . (a) CatBoost, (b) XGBoost, (c) MLP, (d) TabTransformer.

4.4 RESULTS

As desired, post-training, network g assigns a different weight vector to each sample, and much of the weight is concentrated in a relatively small subset of the features of the sample. See Fig. 1 for examples from datasets with relatively few features.

Our results are reported in Tab. 1. Following previous work, we report for each method the statistics of its rank, with mean rank being the most important metric considered. For completeness, we also report statistics on the obtained accuracy.

Consistent with previous work, the GBDT methods are outperforming all other methods. It is evident that our method (using g that is trained for 90 seconds to optimize a simple MLP classifier f) is able to improve both the ranking and accuracy of both CatBoost and XGBoost. Since the CatBoost method is the leading method in multiple benchmarks, including AMLB, this constitutes a new state-of-the-art result.

The same improvement, in both mean ranking and in mean accuracy, is observed for the neural methods, whether MLP-based or attention-based. In this case, g is trained together with f for the entire training duration.

Finally, even for the classical k Nearest Neighbours (kNN), applying the same transfer-learning g we employ for the GBDT methods improves the mean rank and accuracy. In this case, we do not apply the test time augmentation, since there is no direct equivalent of Eq. 6.

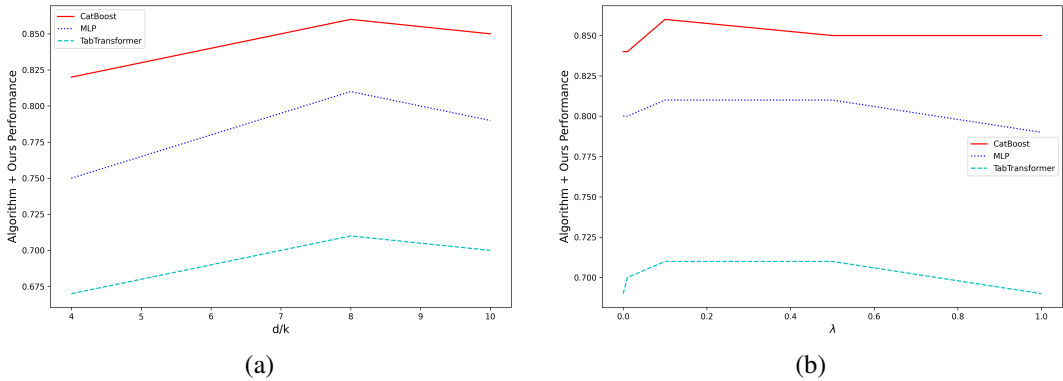


Figure 4: The parameter sensitivity study for the training parameters. (a) varying d/k the ratio of features selected from the second sample, (b) varying λ that balances the loss terms used to train g .

A single exception to the positive contribution of our method is the logistic regression method, for which applying our method hurts performance. This is likely to be due to the limited capacity of this method (recall that kNN is simple but has infinite capacity).

To further visualize these multiple-methods multiple-dataset results, we employ a Dolan-More profile (Dolan & Moré, 2002). In such profiles, there is a single graph per method, depicting the ratio of benchmarks (y-axis) for which the method obtains up to a fraction θ of the maximal score obtained by any of the methods. Unlike conventional Dolan-More profiles, that aim at reducing cost and in which the x-axis is increasing, when a higher score is preferable, θ is presented along the x-axis in a decreasing order, starting with a value of one. A dominating method would obtain a ratio of 1.0 closer to $\theta = 1$, i.e., it would be within a narrow range of the best results obtained.

Fig. 2 presents this type of profile based on the accuracy scores on the AMLB benchmark. Evidently, our method boosts the performance of each of the classification methods, except for Logistic Regression. The dominance of the CatBoost method, when it is augmented by our method, is clear.

To observe the distribution of accuracy scores per dataset before and after applying our method, we provide a per-method visualization in Fig. 3. Each panel presents, for one specific method, the accuracy without our method (x-axis) and with it (y-axis). As can be seen, most data points (each representing a single tabular dataset from AMLB) are above the diagonal. The few cases where the accuracy without our method is preferable occur for low-accuracy datasets. It is possible that for such datasets g , which is co-trained with the classifier f , is less reliable.

4.5 ABLATION ANALYSIS

Snapmix (Huang et al., 2021) employs CAM (Zhou et al., 2016) of the trained classifier f in order to determine the mixing weights. In order to check the performance of this method for tabular data, we apply it to the MLP classifier f . After trying a few options, we employ Grad-CAM (Selvaraju et al., 2017), which seems to outperform CAM. As can be seen in Tab. 1 this combination provides a significant improvement over the MLP baseline, which is still lower than what our method is able to achieve. These conclusions are further supported by the Dolan-More profile in Fig. 2. Note that this application is done for both training time and inference time (as in Eq. 6), otherwise, the improvement is lower.

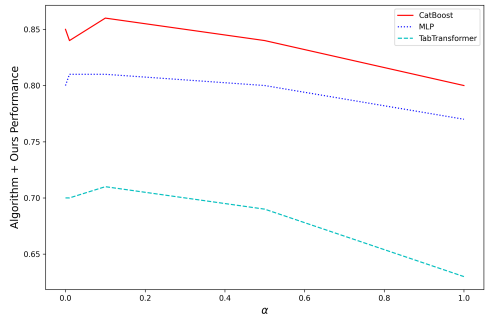


Figure 5: Sensitivity analysis for α .

Two additional ablation experiments are performed to check whether the regularizer \mathcal{L}_m is helpful and the inference time augmentation. These are done as part of a parameter sensitivity study when setting the associated coefficients to zero.

The parameter sensitivity study that explores the behavior of our method when modifying its parameters is depicted in Fig. 4 and Fig. 5. Three parameters are considered: the ratio of features obtained from the second sample of the pair of mixed samples, which has a default value of $1/8$, and the coefficients of the loss term \mathcal{L}_m in Eq. 5 and the inference-time mixing coefficient α (Eq. 6), both of which have a default value of 0.1. The results are shown for MLP, TabTransformer, and CatBoost, where for the latter, the matrix g of the MLP is used.

As can be seen, the default values are close to optimal for the three methods tested. However, one can obtain a slightly better value of α for TabTransformer. Reassuringly, the value of $\lambda = 0$ ($\alpha = 0$) is not optimal, showing the benefit of regularizing g (inference-time augmentation).

5 LIMITATIONS

Motivated by (i) simplicity, (ii) fairness when comparing between methods, (iii) the need to avoid multiple hypotheses, and (iv) limited resources, we do not optimize the parameters of the classifiers to the datasets, and instead employ the default parameters used in the tabular classification literature. When allowing the selection of per dataset parameters, one can obtain better accuracy, especially given the heterogeneous nature of the benchmark. However, we note that for the SOTA method of CatBoost, which we improve, per-dataset selection seems deleterious Zhu et al. (2023). Therefore, our experiments do show SOTA results even with the single set of default hyperparameters.

The auxiliary network g provides a soft feature selection that differs between samples, see Fig. 1. We focus on the prediction outcome and do not attempt to validate the selected features, or present cases in which having a between-sample variability of features importance score would be beneficial. This may be possible, for example, on genetic datasets, where the importance of each gene in a certain context can be justified.

Likewise, the work does not yet explore the more intricate question of how the primary network f is affected by being co-trained with g . Clearly, based on the improvement in performance, the primary model becomes more powerful. However, in what ways? does it become more linear in some embedding space (due to the mixing of samples), less reliant on specific features (due to the random augmentation), better able to grasp multiple modes of data in each class, or, conversely, becomes more regularized and effectively loses capacity?

Lastly, the method can be improved in various ways that require further exploration. Network g may not need to be a separate network, and could be a 2nd prediction head of the classifier network f . Warmup rounds, where f is trained without g may also help.

6 CONCLUSIONS

Mixing-based data augmentation techniques are an effective tool in computer vision. However, they have been only sporadically shown to be successful in tabular data. Indeed, as we demonstrate in our ablation study, the existing mixing methods are only partly effective for tabular data.

In this work, we add an auxiliary network that is used to measure how much of the relevant information in a given sample is passed to the mixed sample. This information is then used to define the loss of the virtual sample that is created by mixing.

The method is generic, does not make any assumptions about the data or the primary model, and is efficient to compute. On an accepted and comprehensive benchmark of tabular data, our method is shown to improve the performance of a varied set of neural methods.

In Tabular data analysis, neural methods still lag in performance in comparison to boosting methods. Network g cannot be learned directly in these cases, since its training requires a gradient signal that is not provided by these non-differentiable methods. However, using the network g learned for an MLP to augment the training data of the leading boosting methods leads to a sizable improvement in their performance and to new state of the art results.

REFERENCES

- Sercan Ö Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 6679–6687, 2021.
- Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pp. 274–283. PMLR, 2018.
- Dara Bahri, Heinrich Jiang, Yi Tay, and Donald Metzler. Scarf: Self-supervised contrastive learning using random feature corruption. *arXiv preprint arXiv:2106.15147*, 2021.
- Jintai Chen, Jiahuan Yan, Danny Ziyi Chen, and Jian Wu. Excelformer: A neural network surpassing gbdts on tabular data. *arXiv preprint arXiv:2301.02819*, 2023.
- 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*, pp. 785–794, 2016.
- Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 113–123, 2019.
- Sajad Darabi, Shayan Fazeli, Ali Pazoki, Sriram Sankararaman, and Majid Sarrafzadeh. Contrastive mixup: Self-and semi-supervised learning for tabular domain. *arXiv preprint arXiv:2108.12296*, 2021.
- Elizabeth D Dolan and Jorge J Moré. Benchmarking optimization software with performance profiles. *Mathematical programming*, 91:201–213, 2002.
- Romain Egele, Prasanna Balaprakash, Isabelle Guyon, Venkatram Vishwanath, Fangfang Xia, Rick Stevens, and Zhengying Liu. Agebo-tabular: joint neural architecture and hyperparameter search with autotuned data-parallel training for tabular data. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–14, 2021.
- Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pp. 1189–1232, 2001.
- 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.
- Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data? *arXiv preprint arXiv:2207.08815*, 2022.
- Hongyu Guo, Yongyi Mao, and Richong Zhang. Mixup as locally linear out-of-manifold regularization. In *AAAI Conference on Artificial Intelligence*, pp. 3714–3722, 2019.
- Hussein Hazimeh, Natalia Ponomareva, Petros Mol, Zhenyu Tan, and Rahul Mazumder. The tree ensemble layer: Differentiability meets conditional computation. In *International Conference on Machine Learning*, pp. 4138–4148. PMLR, 2020.
- Shaoli Huang, Xinchao Wang, and Dacheng Tao. Snapmix: Semantically proportional mixing for augmenting fine-grained data. In *AAAI Conference on Artificial Intelligence*, 2021.
- Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.
- Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka. Well-tuned simple nets excel on tabular datasets. *Advances in neural information processing systems*, 34:23928–23941, 2021.
- Liran Katzir, Gal Elidan, and Ran El-Yaniv. Net-dnf: Effective deep modeling of tabular data. In *International Conference on Learning Representations*, 2020.

- 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. *Advances in neural information processing systems*, 30, 2017.
- Kushal Majmundar, Sachin Goyal, Praneeth Netrapalli, and Prateek Jain. Met: Masked encoding for tabular data. *arXiv preprint arXiv:2206.08564*, 2022.
- Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Ganesh Ramakrishnan, Micah Goldblum, Colin White, et al. When do neural nets outperform boosted trees on tabular data? *arXiv preprint arXiv:2305.02997*, 2023.
- Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *CoRR*, abs/1909.06312, 2019. URL <http://arxiv.org/abs/1909.06312>.
- Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.
- Ivan Rubachev, Artem Alekberov, Yury Gorishniy, and Artem Babenko. Revisiting pretraining objectives for tabular deep learning. *arXiv preprint arXiv:2207.03208*, 2022.
- Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, 2017.
- Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.
- 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.
- Talip Ucar, Ehsan Hajiramezani, and Lindsay Edwards. Subtab: Subsetting features of tabular data for self-supervised representation learning. *Advances in Neural Information Processing Systems*, 34:18853–18865, 2021.
- Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, pp. 6438–6447. PMLR, 2019.
- Zifeng Wang and Jimeng Sun. Transtab: Learning transferable tabular transformers across tables. *arXiv preprint arXiv:2205.09328*, 2022.
- Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.
- Chengrun Yang, Gabriel Bender, Hanxiao Liu, Pieter-Jan Kindermans, Madeleine Udell, Yifeng Lu, Quoc V Le, and Da Huang. Tabnas: Rejection sampling for neural architecture search on tabular datasets. *Advances in Neural Information Processing Systems*, 35:11906–11917, 2022.
- Jinsung Yoon, Yao Zhang, James Jordon, and Mihaela van der Schaar. Vime: Extending the success of self-and semi-supervised learning to tabular domain. *Advances in Neural Information Processing Systems*, 33:11033–11043, 2020.
- Sangdoon Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6023–6032, 2019.
- Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.
- Bingzhao Zhu, Xingjian Shi, Nick Erickson, Mu Li, George Karypis, and Mahsa Shoaran. Xtab: Cross-table pretraining for tabular transformers. *arXiv preprint arXiv:2305.06090*, 2023.

Barret Zoph, Ekin D Cubuk, Golnaz Ghiasi, Tsung-Yi Lin, Jonathon Shlens, and Quoc V Le. Learning data augmentation strategies for object detection. In *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, pp. 566–583. Springer, 2020.