ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 979-8-3503-4485-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICASSP48485.2024.10446501

IMPROVE DEEP FOREST WITH LEARNABLE LAYERWISE AUGMENTATION POLICY SCHEDULES

Hongyu Zhu[1†], Sichu Liang[2†], Wentao Hu[1†], Fang-Qi Li[3], Yali Yuan[1], Shi-Lin Wang[3], Guang Cheng[1]

[1] School of Cyber Science and Engineering, [2] School of Artificial Intelligence, Southeast University.
[3] School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University.

ABSTRACT

As a modern ensemble technique, Deep Forest (DF) employs a cascading structure to construct deep models, providing stronger representational power compared to traditional decision forests. However, its greedy multi-layer learning procedure is prone to overfitting, limiting model effectiveness and generalizability. This paper presents AugDF, an optimized Deep Forest featuring learnable, layerwise data augmentation policy schedules. Specifically, We introduce the Cut Mix for Tabular data (CMT) augmentation technique to mitigate overfitting and develop a population-based search algorithm to tailor augmentation intensity for each layer. Additionally, we propose to incorporate outputs from intermediate layers into a checkpoint ensemble for more stable performance. Experimental results show that AugDF sets new state-of-the-art (SOTA) benchmarks in various tabular classification tasks, outperforming shallow tree ensembles, deep forests, deep neural network, and AutoML competitors. The learned policies also transfer effectively to Deep Forest variants, underscoring its potential for enhancing non-differentiable deep learning modules in tabular signal processing.

Index Terms— Deep Forest, Data Augmentation, Tabular Signal Classification

1. INTRODUCTION

In recent years, deep neural networks (DNNs) have achieved remarkable success in perception tasks such as vision, speech, and language [1]. However, heterogeneous tabular data in industrial scenarios still poses significant challenges, making it the "last unconquered castle" for DNNs [2]. In tabular modeling tasks, shallow decision tree ensembles like random forests and gradient boosting trees remain the preferred tools for data mining specialists [3]. However, these methods lack the ability to learn deep representations, which limits their potential to benefit from larger models and more training data [4, 5, 6].

Drawing on the core principles of deep learning—layer-by-layer processing, intra-model feature transformations, and sufficient capacity—Zhou et al. introduced Deep Forest (DF), a tree-based deep learning approach [7]. Deep Forest constructs a cascading structure by training random forests layer by layer. In classification tasks, each layer's forests generate class probability distributions, which are then merged with original features for the subsequent layer. Upon reaching the maximum layer depth, optimal layer selected by cross-validation is used to produce final output. When sufficiently deep, Deep Forest aims to achieve robust and powerful feature representations through this iterative refinement [8, 9, 10].

However, due to its supervised greedy multi-layer architecture, Deep Forest is prone to overfitting [11], and such issue in one lay can propagate to subsequent layers [12]. While increased depth should theoretically enhance representational power [11], it frequently leads to severe overfitting in practice, undermining generalizability [13]. Moreover, the similar learning objectives across layers diminish the benefits of building deeper models.

Overfitting in Deep Forest stems from its intrinsic learning procedure, which is difficult to suppress through hyperparameter tuning [12]. Reducing the model size, on the other hand, would compromise its primary design principle. Data augmentation offers an alternative for regularization without limiting model capacity [14]. However, tabular data augmentation remains underexplored due to the absence of invariances in heterogeneous features [15]. Furthermore, uniform augmentation intensity across layers result in similar feature representations and introduce high variance, complicating selection of the optimal layer based on validation error.

In this work, we propose to improve Deep Forest with learnable data augmentation policy schedules. Firstly, we introduce a simple yet potent data augmentation technique called CMT (**C**ut **M**ix for **T**abular data) to regularize Deep Forest. Furthermore, a population-based algorithm is proposed to globally search for augmentation policy schedules, enabling layer-specific adjustments in augmentation intensity. Finally, we utilize outputs from intermediate layers to construct a checkpoint ensemble, serving as a variance reducer to ensure more stable and robust results.

Our contributions are summarized as follows:

- We introduce tabular data augmentation, specifically CMT, to mitigate overfitting in Deep Forest.
- We optimize augmentation policy schedules through population-based algorithm, with moderate overhead.
- We improve Deep Forest by integrating outputs from

†Equal contribution.

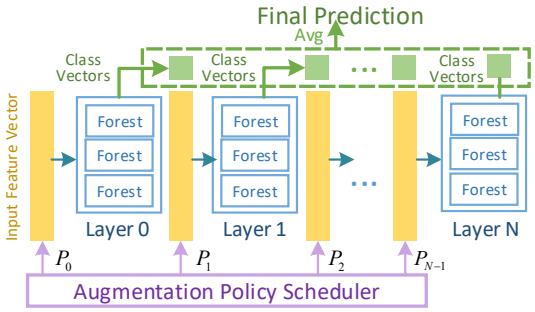

Fig. 1: AugDF architecture with learned policy schedule.

each layer to form a stable checkpoint ensemble.

- Our method sets SOTA benchmarks in tabular classification and enables policy transfer to DF variants.

2. METHODOLOGY

2.1. Cut Mix for Tabular Data (CMT)

Data Augmentation, guided by the Vicinal Risk Minimization principle [16], has significantly advanced regularization of DNNs in fields like vision and language [14]. However, the tabular domain still lacks effective augmentation techniques [15]. The mixup method [17], widely used across data modalities, generate new samples through linear combination of original instances. However, mix-up has two primary limitations in this context. First, convex combination of categorical variables can lead to semantic ambiguity. Second, tabular data exhibits irregular patterns [3], and samples constructed through simple linear interpolation may introduce bias.

We thus design Cut Mix for Tabular data (CMT) to address these issues. The new samples are formulated as:

$$\begin{aligned} \tilde{x} &= \mathbf{w} \odot x_i + (\mathbf{1} - \mathbf{w}) \odot x_j \\ \tilde{y} &= c \cdot y_i + (1 - c) \cdot y_j \end{aligned} \tag{1}$$

Here, $x_i, x_j \in \mathbf{R}^d$ are original samples, and y_i,y_j are their one-hot encoded labels. $\mathbf{w} \in \{0,1\}^d$ is a random mask vector with $\|\mathbf{w}\|_1 = \lambda \cdot d$, $\lambda \sim \text{Beta}(\alpha, \alpha)$. The coefficient c is calculated by the Feature Importance (FI) vector derived from decision forests in the preceding layer, where $FI(k)$ indicates importance of the kth feature:

$$c = \frac{\sum_{k=1}^{d} w_k \cdot FI(k)}{\sum_{k=1}^{d} FI(k)} \tag{2}$$

The blending strategy of CMT swaps partial features between samples[18], drawing all values from the original dataset to avoid semantic ambiguity in categorical variables. Moreover, this method extends beyond linear combinations, better accommodating diverse patterns in heterogeneous tabular data.

2.2. Augmentation Policy Schedule Learning

The intensity of data augmentation is a pivotal factor affecting model performance, as excessive or inappropriate augmentation can introduce biases [19, 20]. To regulate this, an augmentation policy θ is optimized as follows:

$$\theta^* = \arg\max_{\theta \in \Theta} \text{eval}(\theta) \tag{3}$$

where θ consists of hyperparameters $prob$ and mag, representing the likelihood of sample selection for augmentation and the perturbation magnitude for each selected sample (α in CMT). $prob$ and mag are respectively taken from lists P = $[p_1, p_2 \dots p_Q]$ and M = $[m_1, m_2 \dots m_N]$, resulting in a searching space of size $Q \times N$. $\text{eval}(\theta)$ represents the objective to optimize, i.e. accuracy score on the validation set.

While a static policy provides certain regularization benefits, it is limited in mitigating inter-layer convergence. We thus reformulate the problem to identify an optimal sequence of policy combinations, establishing a layerwise policy schedule for Deep Forest. The policy for the k^{th} layer is denoted by $\theta^k = (prob^k, mag^k)$. In a DF with K layers, the complete policy schedule is represented as $\Theta = (\theta^1, \theta^2, \dots, \theta^K)$. This approach enlarges the search space to $(Q \times N)^K$, offering much greater flexibility for augmentation.

However, navigating through such a vast search space is computationally intractable. Therefore, we propose a population-based search algorithm optimized for the unique attributes of DF while ensuring low computational overhead. Initially, we formulate two single-layer search strategies:

Grid Search: This method scans all hyperparameter combinations to maximize current layer's validation accuracy, but is feasible only for a single layer due to complexity.

Neighbour Search: Given a selected policy $\theta_{i,j} = (p_i, m_j)$ from the set $P \times M$, a neighboring policy $\theta_{x,y} = (p_x, m_y)$ is chosen with probability of $\frac{1}{|i-x|+|j-y|+1}$. After normalizing these probabilities, multiple sampling rounds are performed to obtain the required number of different policies.

In the population, $2k$ DFs are considered, each acting as an individual. We initialize the best policy θ^0 for the first layer with grid search and acquire $2k-1$ neighboring policies via neighbour search. These $2k$ policies are used to train the first layer of each DF. For subsequent layers, the procedure iteratively continues as follows:

Explore: Leveraging the policies of the current half top-performing individuals, an exploration by neighbour search is conducted to generate a set of k new policies. These new policies replace the existing k inferior ones, and each individual is then trained with this updated policy set.

Exploit: The current validation accuracy of each individual is assessed and ranked. The upper half of top-performing individuals, along with their associated policies, are preserved, while the lower-performing half is substituted by the current best individual with explored policies.

Upon iterating through each layer, a complete policy schedule is finalized. See Algorithm 1 for pseudocode.

2.3. Checkpoint Ensemble (CE)

The concept of using "historical" information in deep learning is well-established [21]. Averaging snapshots from the SGD trajectory often improves generalization [22, 23, 24].

Algorithm 1: The process of Augmentation Policy Schedule Learning. Functions grid search and neighbour search are detailed in section 2.2. Explore and exploit phases are outlined in the same section.

Input : list of policies p, list of models m, number of models (policies) L, max layer K
Output: Optimal model m^*

$p[0] \leftarrow$ `grid_search()`; `// initialize`
$p[1:L-1] \leftarrow$ `neigh_search(`*p[0],L - 1*`)`;
$m, p \leftarrow$ `eval(`*m,p*`)`;
for $i = 1$ **to** $K - 1$ **do**
  $m[L/2 : L-1] \leftarrow [m[0]] * L/2$; `// exploit`
  $p[L/2 : L-1] \leftarrow$ `neigh_search(`*p[0],L / 2*`)`;
  $m, p \leftarrow$ `eval(`*m,p*`)`; `// explore`
return $m[0]$;
Function `grid_search()`**:**
  find the best policy *bp* using grid search;
  Result: bp
Function `neigh_search(`*bp, X*`)`**:**
  generate X new policies $[new\_policies]$ nearby current best policy *bp*;
  Result: [new_policies]
Function `eval(`*m, p*`)`**:**
  Train each $model$ in m corresponding to p, evaluate the accuracy of each $model$, and sort m and p by accuracy in descending order;
  Result: m,p

While snapshot ensemble enhances performance, it increases storage and inference latency due to multiple weight sets and evaluations. In AugDF, we harness the hierarchical diversity introduced by the augmentation policy schedule to construct a checkpoint ensemble, as illustrated in Figure 1, which incurs neither additional training cost nor significant inference overhead. This strategy serves as a variance reducer, while further boosting the representational capability of the model, thereby achieving ensemble benefits (almost) for free.

The processing procedure of a test sample x in Deep Forest can be recursively defined as follows:

$$F_i(x) = \begin{cases} h_i(x), & i = 1 \\ h_i(x; F_{i-1}(x)), & 1 < i \leq K \end{cases} \tag{4}$$

where h_i represents the i^{th} layer of the Deep Forest, and F_i denotes the model up to the i^{th} layer. K is the maximum number of layers in DF. Unlike the conventional approach where the final output layer o is determined through cross-validation, resulting in $F_o(x)$ as the output, the checkpoint ensemble can be represented as:

$$F_{CE}(x) = \frac{1}{K} \sum_{i=0}^{K-1} F_{K-i}(x) \tag{5}$$

Note that intermediate layers in Deep Forest already require computation; thus, adding a checkpoint ensemble incurs no extra training overhead. While early-stopping is possible through cross-validation, deeper models can be more powerful [11]. Therefore, DF in practice often approach maximum depth [25]. Accordingly, the inference overhead of a checkpoint ensemble is essentially the same as a vanilla DF.

3. EXPERIMENTS

Dataset. Experiments are conducted on eight benchmark datasets for binary and multi-class classification tasks from the UCI Machine Learning Repository [26]. These datasets vary significantly in size, ranging from 452 to 1,516,948 instances, and span multiple application domains such as ECG signals for cardiac arrhythmia diagnosis, remote sensing signals for covertype recognition, air production unit signals for failure prediction, and mobile phone signals for activity recognition. These datasets feature tabular data with a mix of categorical and continuous variables. Detailed statistical characteristics are presented in Table 1.

Table 1: Statistics of the datasets in terms of number of examples, number of features, number of classes.

Datasets	# of examples	# of features	# of classes
adult	48842	14	2
arrhythmia	452	279	16
crowdsourced	10546	29	6
nsl-kdd	148517	42	5
academic	4424	36	3
accelerometer	31991	8	2
metro	1516948	15	2
diabetes	101766	47	3

Settings. For comparison, we widely choose 10 models from diverse families, including Random Forest (RF) [27], prevalent gradient boosting trees such as XGBoost [28], LightGBM [29], and CatBoost [30]. We also compare with DANET [31], a state-of-the-art neural network model for tabular modeling, as well as AutoGluon [32], which constructs crazy ensembles with thousands of base learners across model families with AutoML technique. Additionally, we evaluated the vanilla Deep Forest alongside improved variants csDF [33], mdDF [25], and hiDF [13].

Furthermore, due to the limited support for multi-output training in RF, which is essential for CMT module, we replaced RF with the SketchBoost [34] in AugDF, which is a fast implemention for multioutput GBDT. To isolate the effect, we introduced a variant named skDF, which simply replaces the RF in DF with SketchBoost.

For all experiments, same train-test splits are maintained, ensuring consistency in evaluation. We conduct each experiment five times with different random seeds and report the average performance along with the standard deviation. All DFs have a max layer of 15, and the number of individuals is set to 8 during the Augmentation Policy Schedule Learning.

For the sake of reproducibility, we have made the source code and parameter settings for this work publicly available at `https://github.com/dbsxfz/AugDF`.

Table 2: Comparison of test accuracy of each models across datasets. The best accuracy is highlighted in **Bold** type. • indicates the second-best. The average rank is listed at the bottom.

Datasets	DANET	RF	LightGBM	XGboost	DF	mdDF	hiDF	skDF	csDF	Catboost	AutoGluon	AugDF
adult	85.11±0.15	85.60±0.03	87.26±0.05	86.82±0.08	86.08±0.07	86.87±0.09	86.18±0.07	87.17±0.06	86.20±0.13	87.29±0.00	87.42±0.04•	**87.58±0.02**
arrhythmia	65.54±1.57	72.07±0.57	71.17±1.61	74.59±0.67	75.50±0.36	71.71±1.22	74.77±0.57	74.95±1.75	75.50±1.20	75.14±0.72	77.48±0.00•	**77.66±0.88**
crowd	61.33±1.80	64.27±0.44	65.27±0.80	62.53±1.07	62.87±0.62	65.47±0.62	63.13±0.62	63.53±1.75	63.67±0.87	66.27±0.98•	65.73±0.49	**67.20±0.58**
kdd	76.11±0.53	74.32±0.16	75.01±0.22	77.09±0.08	77.11±0.52	76.88±0.21	76.73±0.54	76.13±0.16	77.10±0.31	77.33±0.13	77.41±0.07•	**78.89±0.12**
academic	74.12±0.78	77.20±0.37	76.43±0.15	76.36±0.34	76.61±0.34	76.47±0.19	76.97±0.55	77.04±0.39	76.50±0.10	76.41±0.50	77.20±0.24•	**77.92±0.18**
accelero	98.36±0.08	98.52±0.02	98.49±0.02	98.45±0.05	98.54±0.02	98.54±0.03	98.55±0.02	98.51±0.06	98.54±0.04	98.55±0.05•	98.54±0.01	**98.57±0.01**
metro	98.23±0.37	98.73±0.01	98.51±0.04	98.74±0.05	98.41±0.29	98.42±0.01	98.68±0.14	98.56±0.21	98.62±0.13	98.78±0.02•	98.60±0.04	**99.15±0.14**
diabetes	58.11±0.14	58.59±0.05	59.31±0.05	59.20±0.09	58.64±0.09	58.95±0.06	58.73±0.05	59.28±0.10	58.75±0.13	59.10±0.07	59.54±0.05•	**59.70±0.07**
Avg.Rank	11.75	7.94	7.75	7.75	7.31	7.06	6.88	6.63	6.38	4.125	3.44•	**1.00**

3.1. Test Accuracy on Benchmark Datasets

Results in Table 2 demonstrate that AugDF consistently outperforms other models in terms of classification accuracy across all eight datasets. The performance gap between AugDF and the vanilla DF is particularly noteworthy. While skDF, which replaces the base learners, outperforms vanilla DF, it does not surpass csDF and Catboost due to overfitting and the challenge to determine the optimal output layer. AugDF demonstrates a robust performance uplift, irrespective of base learner replacement, achieving an average improvement of approximately 2% over skDF on most datasets. Interestingly, even the extensive AutoML ensemble, AutoGluon, fails to match AugDF's performance, underscoring the efficacy of AugDF's deep representation and the benefits derived from its learnable data augmentation policy schedule.

3.2. Comparison of Model Training and Inference Time

In this section, we juxtapose the training and inference latencies of all considered models and employ scatter plots to visualize average performance metrics across the eight datasets. As depicted in Figure 2, it is evident that AugDF achieves optimal performance with a moderate computational cost. AutoGluon, while achieving suboptimal performance compared to AugDF, incurs over five times the training cost and exponentially higher inference latency. When compared to skDF, which utilizes the same base learners, AugDF's training cost is only about tenfold higher within an expansive search space of 10^{30}. Considering that the search process is highly parallelizable, the time complexity may only slightly increase when executed on efficient parallel computing frameworks [35]. It is worth noting that AugDF demonstrates superior inference efficiency compared to DF and its variants, primarily due to the shallower trees used by SketchBoost in AugDF.

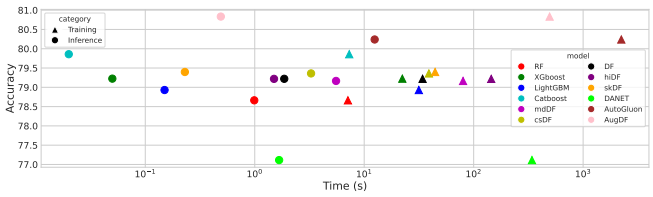

Fig. 2: Scatter Plot of Model Accuracy and Latency (Logarithmic Scale). The values displayed represent averages across these eight datasets.

3.3. Transferring Policy Schedule to Variants of DF

Tree-based models are inherently non-parametric, posing challenges for transfer learning [36]. In this section, we directly apply the policy schedules discovered by AugDF to three variants of DF to further demonstrate the effectiveness and generalizability of our approach. As illustrated in Figure 3, the policy schedules are universally beneficial, with all DF variants exhibiting positive improvements across all datasets, albeit less so than the gains seen with AugDF over DF due to AugDF's exhaustive policy search. The transfer of augmentation policy schedules introduces a novel facet of transfer learning, which is especially advantageous for high-cost training models like hiDF.

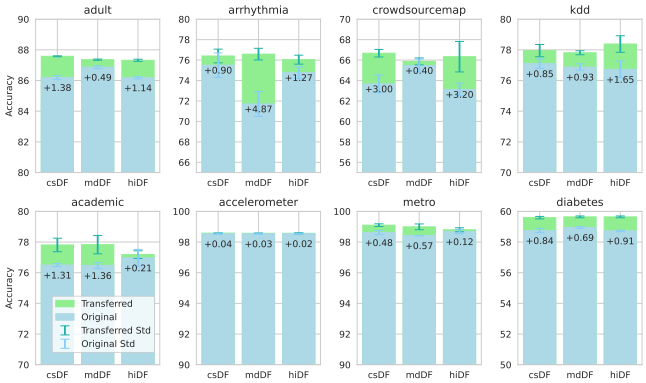

Fig. 3: Comparative Analysis of Accuracy Improvement with Transferred Augmentation Policy Schedules.

4. CONCLUSION

In conclusion, this paper introduces an improved Deep Forest specifically designed for tabular signal classification. Through the integration of CMT data augmentation technique, population-based augmentation policy schedule learning and checkpoint ensemble, we successfully mitigate overfitting and elevate model performance. These advancements yield SOTA results across a variety of benchmarks. Notably, the learned augmentation policy schedules are not only effective but also transferable, allowing application to variants of Deep Forest. This demonstrates its potential for broader impact in the field of non-differentiable deep learning.

5. ACKNOWLEDGEMENTS

This work was supported by National Natural Science Foundation of China /NSF China (Grant No. U22B2025).

References

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] Arlind Kadra, Marius Lindauer, Frank Hutter, and Josif Grabocka, "Well-tuned simple nets excel on tabular datasets," *Advances in neural information processing systems*, vol. 34, pp. 23928–23941, 2021.

[3] Leo Grinsztajn, Edouard Oyallon, and Gael Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?," in *Advances in Neural Information Processing Systems*. 2022, vol. 35, pp. 507–520, Curran Associates, Inc.

[4] Ji Feng, Yang Yu, and Zhi-Hua Zhou, "Multi-layered gradient boosting decision trees," *Advances in neural information processing systems*, vol. 31, 2018.

[5] Sergei Popov, Stanislav Morozov, and Artem Babenko, "Neural oblivious decision ensembles for deep learning on tabular data," in *International Conference on Learning Representations*, 2020.

[6] Hongyu Fu, Yijing Yang, Vinod K. Mishra, and C.-C. Jay Kuo, "Classification via subspace learning machine (slm): Methodology and performance evaluation," in *ICASSP*, 2023, pp. 1–5.

[7] Zhi-Hua Zhou and Ji Feng, "Deep forest: Towards an alternative to deep neural networks.," in *IJCAI*, 2017, pp. 3553–3559.

[8] Jie Song, Liang Xiao, Mohsen Molaei, and Zhichao Lian, "Sparse coding driven deep decision tree ensembles for nucleus segmentation in digital pathology images," *IEEE Transactions on Image Processing*, vol. 30, pp. 8088–8101, 2021.

[9] Ziqing Zhang, Cuicui Kang, Gang Xiong, and Zhen Li, "Deep forest with lrrs feature for fine-grained website fingerprinting with encrypted ssl/tls," in *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, 2019, pp. 851–860.

[10] Hong Wen, Jing Zhang, Quan Lin, Keping Yang, and Pipei Huang, "Multi-level deep cascade trees for conversion rate prediction in recommendation system," in *proceedings of the AAAI conference on artificial intelligence*, 2019, vol. 33, pp. 338–345.

[11] Shen-Huan Lyu, Yi-Xiao He, and Zhi-Hua Zhou, "Depth is more powerful than width with prediction concatenation in deep forest," *Advances in Neural Information Processing Systems*, vol. 35, pp. 29719–29732, 2022.

[12] Ludovic Arnould, Claire Boyer, and Erwan Scornet, "Analyzing the tree-layer structure of deep forests," in *International Conference on Machine Learning*. PMLR, 2021, pp. 342–350.

[13] Yi-He Chen, Shen-Huan Lyu, and Yuan Jiang, "Improving deep forest by exploiting high-order interactions," in *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2021, pp. 1030–1035.

[14] Jonas Geiping, Micah Goldblum, Gowthami Somepalli, Ravid Shwartz-Ziv, Tom Goldstein, and Andrew Gordon Wilson, "How much data are augmentations worth? an investigation into scaling laws, invariance, and implicit regularization," in *The Eleventh International Conference on Learning Representations*, 2023.

[15] Elliott Gordon-Rodriguez, Thomas Quinn, and John P Cunningham, "Data augmentation for compositional data: Advancing predictive models of the microbiome," *Advances in Neural Information Processing Systems*, vol. 35, pp. 20551–20565, 2022.

[16] Olivier Chapelle, Jason Weston, Léon Bottou, and Vladimir Vapnik, "Vicinal risk minimization," in *Advances in Neural Information Processing Systems*, T. Leen, T. Dietterich, and V. Tresp, Eds. 2000, vol. 13, MIT Press.

[17] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz, "mixup: Beyond empirical risk minimization," in *6th International Conference on Learning Representations, ICLR, Vancouver, BC, Canada, April 30 - May 3, 2018*.

[18] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *ICCV, Seoul, Korea (South), October 27 - November 2, 2019*. 2019, pp. 6022–6031, IEEE.

[19] 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*, 2019, pp. 113–123.

[20] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel, "Population based augmentation: Efficient learning of augmentation policy schedules," in *International Conference on Machine Learning*. PMLR, 2019, pp. 2731–2741.

[21] Xiang Li, Ge Wu, Lingfeng Yang, Wenhai Wang, Renjie Song, and Jian Yang, "A survey of historical learning: Learning models with learning history," *CoRR*, vol. abs/2303.12992, 2023.

[22] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger, "Snapshot ensembles: Train 1, get m for free," in *International Conference on Learning Representations*, 2017.

[23] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson, "Loss surfaces, mode connectivity, and fast ensembling of dnns," *Advances in neural information processing systems*, vol. 31, 2018.

[24] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[25] Shen-Huan Lyu, Liang Yang, and Zhi-Hua Zhou, "A refined margin distribution analysis for forest representation learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[26] Dheeru Dua and Casey Graff, "UCI machine learning repository," 2017.

[27] Leo Breiman, "Random forests," *Machine learning*, vol. 45, pp. 5–32, 2001.

[28] 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*, 2016, pp. 785–794.

[29] 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*, vol. 30, 2017.

[30] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin, "Catboost: unbiased boosting with categorical features," *Advances in neural information processing systems*, vol. 31, 2018.

[31] Jintai Chen, Kuanlun Liao, Yao Wan, Danny Z Chen, and Jian Wu, "Danets: Deep abstract networks for tabular data classification and regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2022, vol. 36, pp. 3930–3938.

[32] Jonas Mueller, Xingjian Shi, and Alexander Smola, "Faster, simpler, more accurate: practical automated machine learning with tabular, text, and image data," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020, pp. 3509–3510.

[33] Ming Pang, Kai-Ming Ting, Peng Zhao, and Zhi-Hua Zhou, "Improving deep forest by confidence screening," in *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 2018, pp. 1194–1199.

[34] Leonid Iosipoi and Anton Vakhrushev, "Sketchboost: Fast gradient boosted decision tree for multioutput problems," *Advances in Neural Information Processing Systems*, vol. 35, pp. 25422–25435, 2022.

[35] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I. Jordan, and Ion Stoica, "Ray: A distributed framework for emerging AI applications," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, Carlsbad, CA, Oct. 2018, pp. 561–577, USENIX Association.

[36] Zeyi Wen, Qinbin Li, Bingsheng He, and Bin Cui, "Challenges and opportunities of building fast gbdt systems.," in *IJCAI*, 2021, pp. 4661–4668.