SUPPLEMENTARY MATERIAL: IMPROVE DEEP FOREST WITH LEARNABLE LAYERWISE AUGMENTATION POLICY SCHEDULE

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.

A. OVERVIEW

This document serves as the supplementary material for the paper titled "Improve Deep Forest with Learnable Layerwise Augmentation Policy Schedule". Addressing concerns and suggestions raised by the reviewers, the content is structured as follows:

- **Section B**: Detailed explanations for the AugDF architecture figure and the pseudocode of the augmentation policy schedule learning algorithm;
- **Section C**: Additional related work to clarify the relationship between AugDF and existing methods, highlighting its innovative aspects;
- **Section D**: Analysis of experimental results, highlighting the factors contribute to AugDF's superior performance over competing algorithms, complemented by an ablation study.

B. CLARITY ENHANCEMENT

B1. Explanation for the AugDF Architecture Figure

Due to constraints in manuscript length, it is challenging to incorporate a more complex, larger figure within the main text. Therefore, we provide a clear exposition here by juxtaposing our design with the vanilla Deep Forest architecture [1].

As depicted in Figure 1, the vanilla Deep Forest employs a cascading structure, wherein each layer consists of multiple decision forests. These forests, trained sequentially, generate class probability distributions. The output vector of each layer is concatenated with the original input feature vector, serving as the input for the subsequent layer. The final output layer, established upon reaching the predefined maximum layer depth, is determined through cross-validation, leading to the ultimate prediction.

However, as elucidated in the main text, the inherently greedy layer-by-layer training architecture of the vanilla Deep Forest predisposes it towards overfitting. To mitigate this, we propose the CMT data augmentation technique, complemented by an Augmentation Policy Scheduler that optimizes data augmentation policies for each layer, as illustrated in Figure 2. Employing a population algorithm, the scheduler dynamically adjusts the augmentation intensity based on the learning status of each layer, thereby facilitating customized regularization at varying depth levels. This strategy significantly enhances the generalization ability of Deep Forest across diverse datasets. Furthermore, it imbues the Deep Forest with adaptive flexibility, preventing layer homogenization induced by similar learning objectives.

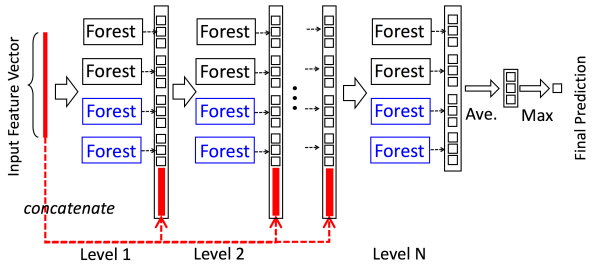

Fig. 1. vanilla Deep Forest architecture [1].

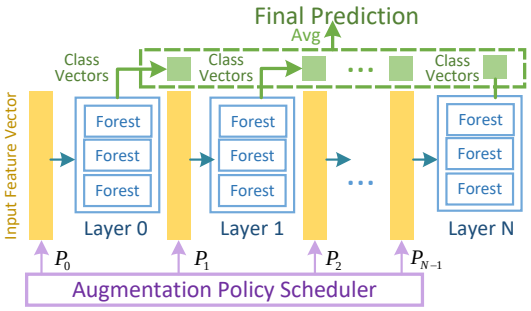

Fig. 2. AugDF architecture with learned policy schedule.

B2. Explanation for the APSL Algorithm

We have attempted to address potential sources of ambiguity in the pseudocode of augmentation policy schedule learning (APSL) algorithm and present our revised version here. Addi-

tionally, we provide a detailed explanation of our pseudocode, complemented by code examples for clarity.

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^*
// initialize
$p[0] \leftarrow$ `grid_search()`;
$p[1:L-1] \leftarrow$ `neigh_search(`*p[0],L - 1*`)`;
$m, p \leftarrow$ `train_eval(`*m,p*`)`;
for $i = 1$ **to** $K-1$ **do**
  // exploit phase
  $m[L/2:L-1] \leftarrow [m[0]] * L/2$;
  // explore phase
  $p[L/2:L-1] \leftarrow$ `neigh_search(`*p[0],L / 2*`)`;
  $m, p \leftarrow$ `train_eval(`*m,p*`)`;
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 `train_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

Algorithm 1 depicts the pseudocode after revisions. We discerned ambiguities particularly in the annotations pertaining to the concepts of "exploit" and "explore." The ambiguity was primarily rooted in the middle line of the pseudocode within the iterative construct. To rectify this, we have introduced line breaks in all relevant comments at this point. This modification is intended to more effectively communicate the underlying intent of these segments.

In the implemented code, we have configured the size of population (number of models or policies) to 8 and the depth of AugDF to 15. Upon initialization, a total of 8 untrained models are generated and assembled into the list designated as m. The global optimal augmentation policy bp_1 for the first layer is derived through grid search. Employing bp_1, the procedure explores 7 adjacent policies, subsequently enlisting them in p. Following this, the models in m undergo training, each aligned with a policy corresponding to its respective position. This process results in the list m containing models that have undergone a single training cycle. The models within m are then ranked based on their performance in the validation set, arranging them in descending order. This ranking is concurrently applied to sort p. Consequently, this furnishes us with the refined lists m and p, primed for the ensuing round of training. Subsequently, we perpetuate the aforementioned steps iteratively until we reach the predetermined maximum of 15 rounds:

- Substitute the four least effective models in the population (ranging from $m[5]$ to $m[8]$) with the model exhibiting the highest performance ($m[0]$).
- Utilize the policy corresponding to the best-performing model, specifically $p[0]$, to explore for 4 neighboring policies. Replace the 4 underperforming policies ($p[5]$ to $p[8]$) with these newly obtained policies.
- Train models in the whole population with updated policies from p and sort models associate with their policies in descending order based on their performance on the validation set.

We hope that this detailed explanation will facilitate a deeper understanding of our pseudocode. Note that the concise functions 'grid search' and 'neighbor search', as referenced in the pseudocode, are expounded upon in Section 2.2 of the mian text.

C. NOVELTY AND RELATIONSHIP TO EXISTING METHODS

To the best of our knowledge, our AugDF introduces the first data augmentation technique tailored for Deep Forest in tabular signal processing. As delineated in the main text, the tabular domain lacks appropriate data augmentation methods due to its inherent absence of invariance. Previously, the mixup strategy—a modality-agnostic and broadly effective data augmentation approach[2]—has been employed in the training of deep neural networks for tabular classification tasks [2, 3, 4], achieving noteworthy results. However, the interpolation methodology of mixup can generate unrealistic inputs, potentially leading to shifts in decision boundaries [5]. Moreover, its linear interpolation approach may not be apt for tabular data characterized by irregular patterns. In this paper, we highlight the semantic ambiguity in category variables induced by mixup in tabular data and introduce a novel nonlinear interpolation method, CMT, which constructs augmented samples with all feature values sampled from the original dataset. This approach effectively circumvents the aforementioned issues, with related ablation experiments exemplified in Section D.

Following the inception of the vanilla Deep Forest, advancements in Deep Forest literature have primarily concentrated on three aspects: architecture, computation overhead,

and application. As a quintessential architectural improvement, csDF [6, 7] employs a confidence screening mechanism at each layer, theoretically proven to enhance generalization performance by discarding a subset of samples and incrementally increasing forest complexity. The mdDF [8] introduces a margin distribution reweighting approach, optimizing the marginal loss at each forest layer, thereby reducing the overall margin ratio. Meanwhile, hiDF [9] refines the forests at each layer into high-order feature interactions, enriching the feature representation and reformulating deep forest to some extent as a tool for automated feature engineering. Concurrently, the sample selection in csDF offers a degree of training speed enhancement, whereas hiDF reduces inference overhead by distilling numerous decision paths in the forest layers into fewer feature generation rules. Additionally, BLB-gcForest [10] enhances both training and inference speeds of Deep Forest through high-performance distributed algorithms.

In terms of applications, Deep Forests are increasingly employed in a wide array of fields, such as weak label learning [11], multi-label learning [12], anomaly detection [13, 14], website fingerprinting [15], recommendation systems [16], image segmentation [17], and liveness detection [18]. However, none of these applications have incorporated data augmentation, even in fields like image processing where data augmentation is well-developed [17, 18]. This oversight is presumably due to shallow decision forests, the fundamental component of deep forests, being a classic algorithm for tabular data processing and seldom discussed in conjunction with data augmentation. Our work bridge this gap and draws inspiration from learnable data augmentation [19] and snapshot ensembling [20], previously exclusive to deep networks. We devise a layerwise augmentation policy schedule learning algorithm and checkpoint ensemble strategy tailored to deep forest characteristics, culminating in the AugDF architecture. Detailed ablation studies on the efficacy of these three strategies are presented in Section D.

D. ANALYSIS OF EXPERIMENTAL RESULTS

D1. Why AugDF Performs Better

Due to inductive biases unsuitable for tabular data, DNN classifiers like DANET still fall short of decision forest, aligning with recent benchmarks [21, 22]. Shallow decision forests like RF, XGBoost, LightGBM, and CatBoost, as elaborated in our main text, are efficacious 'out-of-the-box' tools for tabular data. However, their lack of distributed representation learning capability restricts benefits from more data and larger models. Deep forest and the variants such as csDF, mdDF, and hiDF, while embodying both the inductive bias of decision forests suitable for tabular data and the representational power of deep models, are prone to overfitting due to their inherently greedy layer-wise training architecture and similar inter-layer learning objectives. This propensity hampers both representational effectiveness and generalization performance. Our proposed AugDF builds upon deep forest architecture, employing CMT and learnable layerwise augmentation policy schedule. This approach adaptively mitigates overfitting and utilizes checkpoint ensemble for an improved and more stable representation, thereby improving Deep Forest generalization.

Remarkably, AutoGluon, an ensemble comprising thousands of sub-models from diverse model families like KNN, decision forests, and neural networks, constructed using AutoML techniques, also underperformed compared to AugDF. We hypothesize that AutoGluon's performance is constrained by several factors: 1) The use of diverse but slightly weaker sub-models such as KNN and MLP for ensemble diversity, which stabilizes performance but also potentially caps its upper limit; 2) Analogous to vanilla DF, AutoGluon suffers from overfitting due to deep stacking and greedy layer-wise training. Although enhancing a single layer in deep stacking, such as the transition from vanilla DF to skDF, aids overall model performance, this improvement is limited—evident as skDF still underperforming compared to csDF and CatBoost, and significantly lower than AugDF; 3) The model scale and performance of AutoGluon are contingent on time investment. Higher time budgets may yield superior models, but under our current time constraints, AutoGluon requires on average five times the training overhead and over ten times the inference latency of AugDF. Consequently, larger models become progressively less practical. Additionally, our APSL approach may potentially be effective for larger deep stacking models like AutoGluon, which we earmark for future exploration.

D2. Ablation Study

To further elucidate the reasons behind AugDF's enhanced performance, we have conducted ablation experiments to quantify the contribution of each component in AugDF.

The ablation experiments are primarily divided into two parts, as shown in Table 1:

- While data augmentation for decision forests is uncommon, there exists a plethora of mix-style augmentation methodologies analogous to CMT. Is our custom-designed CMT justified for tabular data?

- Are APSL (Augmentation Policy Schedule Learning) and CE (Checkpoint Ensemble) necessary? How does the efficacy of APSL compare to a baseline where policies are randomly assigned to each layer? If CE is not performed, how would the performance change?

In our experiments, substituting CMT with the conventional mix up [2] resulted in a notable decline in performance. Nevertheless, the results still surpassed those of vanilla Deep Forest and skDF. This suggests that not all augmentation ap-

Table 1. Comparisons between the performance after removing a specific component of AugDF and the baseline.

model	adult	arrh	crowd	kdd	academic	acceler	metro	diabetes
DF	86.08 ± 0.07	75.50 ± 0.36	62.87 ± 0.62	77.11 ± 0.52	76.61 ± 0.34	98.54 ± 0.02	98.41 ± 0.29	58.64 ± 0.09
skDF	87.17 ± 0.06	74.95 ± 1.75	63.53 ± 1.75	76.13 ± 0.16	77.04 ± 0.39	98.51 ± 0.06	98.56 ± 0.21	59.28 ± 0.10
w/o CMT	87.52 ± 0.05	76.40 ± 0.75	66.13 ± 0.80	76.02 ± 0.13	77.69 ± 0.34	98.55 ± 0.01	98.74 ± 0.02	59.64 ± 0.10
w/o APSL	87.57 ± 0.03	76.76 ± 1.17	66.33 ± 0.85	77.38 ± 0.15	77.72 ± 0.40	98.56 ± 0.06	99.13 ± 0.20	59.59 ± 0.05
w/o CE	87.27 ± 0.07	76.22 ± 1.64	66.07 ± 0.64	76.63 ± 0.47	77.31 ± 0.39	98.56 ± 0.03	98.80 ± 0.29	59.37 ± 0.08
AugDF	87.58 ± 0.02	77.66 ± 0.88	67.20 ± 0.58	78.89 ± 0.12	77.92 ± 0.18	98.57 ± 0.01	99.15 ± 0.14	59.70 ± 0.07

proach effective in other domains can be directly applied to tabular data.

As described in the main text, mix up has two primary limitations in the tabular domain. First, convex combination of categorical variables can lead to semantic ambiguity. Second, tabular data exhibits irregular patterns, and samples constructed through simple linear interpolation may introduce bias. Our experimental results substantiate the correctness of our hypothesis.

Additionally, we contrast the performance of AugDF, which employs Augmentation Policy Schedule Learning (APSL), with models adopting random policy selection. The findings reveal that APSL consistently yields superior and more stable results. Removal of checkpoint ensemble (CE) also resulted in an obvious decrease in performance. Furthermore, it is notable that both alternatives display a significant rise in standard deviation.

Conceptually, APSL has proven effective in mitigating the performance fluctuations and potential decreases caused by overly aggressive augmentation policies. It adeptly customizes the augmentation intensity for each layer, fostering a more harmonious equilibrium. Concurrently, the role of CE is in sync with our articulated objectives. This strategy not only acts as a variance reducer but also enhances the model's representational capability, thereby achieving ensemble benefits with minimal additional cost.

In summary, this subsection comprehensively analyzes the impact of three key components in AugDF: CMT, APSL, and CE. It is evident that the removal of each component individually leads to a decline in performance. However, the results still outperform both the vanilla DF and skDF, which only replaces the base learner. This underscores the efficacy of each component and highlights their synergistic contribution to improving generalization.

E. REFERENCES

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

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

[3] 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.

[4] Jintai Chen, Kuanlun Liao, Yanwen Fang, Danny Z. Chen, and Jian Wu, "Tabcaps: A capsule neural network for tabular data classification with bow routing," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. 2023, OpenReview.net.

[5] Ziqi Wang, Yuexin Wu, Frederick Liu, Daogao Liu, Le Hou, Hongkun Yu, Jing Li, and Heng Ji, "Augmentation with projection: Towards an effective and efficient data augmentation paradigm for distillation," in *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. 2023, OpenReview.net.

[6] 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.

[7] Ming Pang, Kai Ming Ting, Peng Zhao, and Zhi-Hua Zhou, "Improving deep forest by screening," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 9, pp. 4298–4312, 2022.

[8] 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.

[9] 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.

[10] Zexi Chen, Ting Wang, Haibin Cai, Subrota Kumar Mondal, and Jyoti Prakash Sahoo, "Blb-gcforest: A high-performance distributed deep forest with adaptive sub-forest splitting," *IEEE Trans. Parallel Distributed Syst.*, vol. 33, no. 11, pp. 3141–3152, 2022.

[11] Qian-Wei Wang, Liang Yang, and Yu-Feng Li, "Learning from weak-label data: A deep forest expedition," in *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*. 2020, pp. 6251–6258, AAAI Press.

[12] Liang Yang, Xi-Zhu Wu, Yuan Jiang, and Zhi-Hua Zhou, "Multi-label learning with deep forest," in *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, Giuseppe De Giacomo, Alejandro Catalá, Bistra Dilkina, Michela Milano, Senén Barro, Alberto Bugarín, and Jérôme Lang, Eds. 2020, vol. 325 of *Frontiers in Artificial Intelligence and Applications*, pp. 1634–1641, IOS Press.

[13] Haolong Xiang, Hongsheng Hu, and Xuyun Zhang, "Deepiforest: A deep anomaly detection framework with hashing based isolation forest," in *IEEE International Conference on Data Mining, ICDM 2022, Orlando, FL, USA, November 28 - Dec. 1, 2022*, Xingquan Zhu, Sanjay Ranka, My T. Thai, Takashi Washio, and Xindong Wu, Eds. 2022, pp. 1251–1256, IEEE.

[14] Diana Laura Aguilar, Miguel Angel Medina-Pérez, Octavio Loyola-González, Kim-Kwang Raymond Choo, and Edoardo Bucheli-Susarrey, "Towards an interpretable autoencoder: A decision-tree-based autoencoder and its application in anomaly detection," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 2, pp. 1048–1059, 2023.

[15] 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.

[16] 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.

[17] 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.

[18] Jiucui Lu, Yuezun Li, Jiaran Zhou, Bin Li, and Siwei Lyu, "Forensics forest: Multi-scale hierarchical cascade forest for detecting gan-generated faces," in *IEEE International Conference on Multimedia and Expo, ICME 2023, Brisbane, Australia, July 10-14, 2023*. 2023, pp. 2309–2314, 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] 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.

[21] 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*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds. 2022, vol. 35, pp. 507–520, Curran Associates, Inc.

[22] Duncan C. McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Ganesh Ramakrishnan, Micah Goldblum, and Colin White, "When do neural nets outperform boosted trees on tabular data?," in *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.