A Novel Approach to Training Deep Neural Networks Via Multiple Output Heads and Boosting

Mohamed Hamdy Ponnuthurai Suganthan Abdulaziz Al-Ali
Computer Science and Engineering Department, College of Engineering
Qatar University, Doha 2713, Qatar
{mm1905748, p.n.suganthan, a.alali}@qu.edu.qa

***Abstract*—Deep neural networks have demonstrated strong performance across diverse learning tasks by learning hierarchical feature representations through end-to-end optimization. Nevertheless, conventional training relies on a single output head attached to the final network layer, leaving intermediate representations underutilized and limiting the potential benefits of internal supervision. Although some existing approaches attempt to exploit intermediate outputs through greedy layer-wise training or multi-head ensembles, these methods often constrain the representational capacity of deeper layers or rely on auxiliary heads that are introduced post hoc and do not actively shape the training of the backbone. In this paper, we propose a boosting-guided progressive training and freezing strategy that explicitly leverages multiple intermediate output heads within a single neural network to guide representation learning across depth. The proposed approach progressively incorporates intermediate predictors and employs boosting-inspired sample reweighting to focus the training of deeper layers/blocks on harder examples. We evaluate the proposed method on both vision and tabular classification tasks. Our results show that intermediate output heads and boosting improve the performance of convolutional and transformer-based backbones on four image classification datasets. Multilayer perceptrons, operating as an ensemble using intermediate output layers, outperform state-of-the-art methods on ten tabular classification datasets.**

***Index Terms*—deep neural networks, block-wise training, boosting, ensemble deep neural networks**

I. Introduction

Deep neural networks have become the dominant paradigm for supervised learning across a wide range of application domains, including computer vision, natural language processing, and tabular data analysis. Their success is largely attributed to hierarchical representation learning [1], where successive layers transform raw inputs into increasingly abstract and task-relevant features through end-to-end optimization. Despite their expressive power, conventional neural networks are typically trained using a single output head attached to the final layer, such that supervision is injected only at the deepest representation level [2]. As a result, intermediate representations are optimized indirectly through backpropagation, and their potential contribution to learning efficiency, robustness, and generalization remains largely underexploited [3].

A growing body of work has explored mechanisms for incorporating intermediate supervision into deep networks. These approaches are motivated by several practical and theoretical considerations: improving gradient flow in deep architectures [2], accelerating convergence, enabling resource-aware inference [4], and enhancing performance through multi-level representations [3]. However, existing strategies often introduce trade-offs that limit their effectiveness. Greedy layer-wise training methods optimize network blocks sequentially using private output heads objectives [5], which can restrict the representational capacity of later layers and deviate from global end-to-end optimization. Multi-head or ensemble-based approaches either employ greedy layer-wise training, rely on independently trained models [6], or attach intermediate heads in a post-hoc manner [7], without explicitly shaping backbone representations to support them. Early-exit architectures, while effective for adaptive inference, primarily focus on inference-time efficiency rather than improving the learned backbone itself [8].

In parallel, ensemble learning and boosting methods [9] have long demonstrated strong performance guarantees by combining multiple weak or moderately strong predictors in a stage-wise manner. Boosting techniques adaptively emphasize difficult training examples by reweighting samples based on past errors, leading to progressively improved predictors. Despite their success in classical machine learning, the integration of boosting principles into deep neural network training remains challenging. Standard boosting frameworks assume independent weak learners, whereas deep networks rely on shared representations and joint optimization, making naive combinations inefficient or unstable.

By integrating intermediate supervision with boosting-inspired optimization, the proposed *boosting-guided progressive training and freezing* (BPTF) framework provides a mechanism for guiding representation learning across network depth within a single model. Auxiliary output heads are progressively introduced at intermediate network blocks not only for prediction, but also to assess sample difficulty and adaptively reweight training data in subsequent rounds. This training-and-freezing mechanism enables later blocks to focus on harder examples while earlier blocks retain their ability to model simpler patterns, preserving the full representational capacity of the network and combining the strengths of hierarchical representation learning, ensemble methods, and boosting within a unified training procedure.

The main contributions of this work are:

- We propose a novel *boosting-guided progressive training*

Supplementary material is available at https://github.com/P-N-Suganthan/PDFs/blob/master/wcci26_supp.zip.

and freezing (BPTF) strategy that integrates multiple intermediate output heads with boosting-inspired sample reweighting to explicitly supervise and guide depth-wise representation learning within a single unified neural network.
- We adapt the proposed BPTF framework to tabular learning by applying it to multilayer perceptrons (MLP) with intermediate output heads.
- We extend the BPTF strategy to vision tasks by incorporating it into both convolutional and transformer-based architectures.
- We conduct extensive empirical evaluations on ten tabular benchmarks and four small vision datasets, validating the proposed approach across multiple architectures and showing consistent performance improvements over standard end-to-end training and state-of-the-art models.

II. Related Work

Standard end-to-end training of deep neural networks relies on a single output head in the final layer. Although effective in many settings, this paradigm underutilizes intermediate representations and can suffer from optimization challenges such as vanishing gradients and slow convergence [1], [2]. To mitigate these issues, a growing body of work has explored mechanisms for incorporating intermediate supervision into deep neural networks, motivated by improving gradient flow in deep architectures, accelerating convergence, enabling resource-aware inference, and enhancing performance through multi-level representations [2], [3]. Nevertheless, existing strategies often introduce trade-offs that limit their overall effectiveness.

Early greedy layer-wise training methods optimize network blocks sequentially using private objectives, and perform inference at test time using only the final output head [5]. However, greedily training the network block by block is known to restrict the representational capacity of later layers and deviate from global end-to-end optimization. Additionally, relying solely on the final output head discards the potential of the other trained intermediate heads. To avoid restricting representational capacity, some work explored training the full network with an objective that minimizes the loss imposed by all heads (joint optimization) [2]. However, only the final output head is used for inference.

More recently, multi-output and ensemble-deep architectures have been proposed to exploit intermediate representations by attaching auxiliary output heads to hidden layers and maintaining them during inference. SimCNN, [10] greedily trains each layer using an auxiliary classifier and allows predictions from intermediate layers to be used at inference time. Similarly, ensemble deep MLP (edMLP) associates a linear classifier (auxiliary output head) with each hidden layer of the backbone MLP. Unlike standard greedy layer-wise training, SimCNN and edMLP aggregate the predictions from all output heads during inference, leading to strong empirical performance. However, the greedy nature of both SimCNN and edMLP constrains later layers and increases training cost. More recent variants, such as MO-MLP, attach auxiliary output heads to a pre-trained MLP without retraining the backbone [7], improving efficiency and maintaining the backbone's representation capacity but still relying on representations not explicitly optimized to support intermediate prediction.

A related line of work explores early-exit and anytime prediction architectures, which introduce intermediate classifiers to enable adaptive inference under different computational budgets [4], [8]. While effective for reducing inference cost, early-exit models primarily focus on runtime efficiency rather than improving performance and representation learning within the backbone, and typically do not use intermediate outputs to guide subsequent training stages.

In contrast to the above methods, BPTF integrates intermediate supervision and boosting-inspired sample reweighting directly into the training dynamics of a single deep network. Unlike greedy layer-wise approaches, it preserves the full representational capacity of deeper network components, and unlike post-hoc multi-output models, it explicitly uses intermediate predictors to guide the optimization of deeper representations, combining hierarchical representation learning, multi-output architectures, and boosting within a single unified training procedure.

III. Boosting-guided Progressive Training and Freezing (BPTF) Strategy

A. Problem Setup and Notation

We consider a supervised classification setting with a training dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{M}$, where $\mathbf{x}_i \in \mathcal{X}$ denotes an input sample, $\mathbf{y}_i \in \{0,1\}^C$ is its corresponding one-hot encoded class label, and $y_i \in \{1, \ldots, C\}$ denotes the associated class index, M is the number of training samples, and C denotes the number of classes. In the case of tabular data, $\mathbf{x}_i \in \mathbb{R}^d$ represents a d-dimensional feature vector, whereas for vision tasks $\mathbf{x}_i \in \mathbb{R}^{H \times W \times C_{\text{in}}}$ corresponds to an image.

Let $f(\cdot;\theta)$ denote a neural network backbone composed of N sequential blocks,$\{B_1, B_2, \ldots, B_N\}$, followed by a final output head O_N. Each block B_k maps its input representation to a higher-level embedding, and the composition of the backbone can be written as

$$\mathbf{h}_k = B_k(\mathbf{h}_{k-1}), \quad k = 1, \ldots, N, \tag{1}$$

where $\mathbf{h}_0 = \mathbf{x}_i$ and $\mathbf{h}_k$ denotes the latent representation produced by block B_k.

In addition to the final output head O_N, multiple auxiliary output heads $\{O_1, O_2, \ldots, O_{N-1}\}$ are associated with intermediate blocks of the backbone. Each output head O_k produces a prediction $\hat{\mathbf{y}}_i^{(k)}$ based on the corresponding representation $\mathbf{h}_k$. Formally, the prediction of output head O_k for sample $\mathbf{x}_i$ is given by

$$\hat{\mathbf{y}}_i^{(k)} = \begin{cases} O_k\big([r(\mathbf{h}_k), \mathbf{x}_i]\big)\,, \\ O_k(r(\mathbf{h}_k))\,, \end{cases} \tag{2}$$

where $r(\cdot)$ denotes an optional reduction function applied to the intermediate representation (e.g., pooling or projection), and $[\cdot,\cdot]$ denotes feature concatenation. An overview of the

resulting architecture and the placement of intermediate output heads is shown in Fig. 1.

The loss incurred by output head O_k on sample $(\mathbf{x}_i, \mathbf{y}_i)$ is denoted by $\ell\big(\mathbf{y}_i, \hat{\mathbf{y}}_i^{(k)}\big)$. To enable boosting-style optimization, each training sample is associated with a non-negative weight $w_i^{(k)}$, which reflects its relative difficulty at training round k. The weighted empirical risk induced by the sample weights $w^{(k)}$ is defined as

$$\mathcal{L}^{(k)} = \sum_{i=1}^{M} w_i^{(k)}\, \ell\big(\mathbf{y}_i, \hat{\mathbf{y}}_i^{(k)}\big) \tag{3}$$

where $\hat{\mathbf{y}}_i^{(k)}$ denotes the prediction of the model being optimized under the current weighting.

In this work we use the multi-class AdaBoost framework [9] to update both estimator weights and sample weights. Formally, given a predictor producing class score vectors $\hat{\mathbf{y}}_i^{(k)}$, we define the corresponding predicted class label as

$$\tilde{y}_i^{(k)} = \arg \max_{c \in \{1,\dots,C\}} \hat{y}_{i,c}^{(k)}.$$

Its weighted classification error is then defined as

$$\epsilon_k = \frac{\sum_{i=1}^{M} w_i^{(k)}\, \mathbb{I}\big(\tilde{y}_i^{(k)} \neq y_i\big)}{\sum_{i=1}^{M} w_i^{(k)}}, \tag{4}$$

where $\mathbb{I}(\cdot)$ denotes the indicator function. The corresponding estimator weight is then given by

$$\alpha_k = \log\left(\frac{1-\epsilon_k}{\epsilon_k}\right) + \log(C-1). \tag{5}$$

The sample weights are subsequently updated according to

$$w_i^{(k+1)} = w_i^{(k)} \exp\big(\alpha_k\, \mathbb{I}(\tilde{y}_i^{(k)} \neq y_i)\big), \tag{6}$$

followed by normalization such that $\sum_{i=1}^{M} w_i^{(k+1)} = 1$. These definitions establish a weighted empirical risk that emphasizes misclassified samples and serves as the basis for the training strategy introduced in the following section.

B. Proposed Training Strategy

Unlike greedy layer-wise training [5], which sequentially trains blocks using private objectives and discards intermediate heads, the proposed training strategy optimizes a single backbone in a progressive and boosted manner that avoids limiting the representational capacity of the network while incrementally improving performance on harder training examples.

The training procedure starts with a standard end-to-end optimization of the full backbone $B_1 \to \cdots \to B_N$ together with the final output head O_N, using uniform sample weights $w_i^{(0)} = 1/M$. This initial round corresponds to conventional neural network training and serves as a baseline as well as an initialization for subsequent stages. Following this stage, auxiliary output heads are introduced at intermediate blocks of the backbone to assess the difficulty of training samples at different representation depths.

At a given training round k, an auxiliary output head O_k is trained while keeping the backbone parameters fixed.

Algorithm 1 BPTF Strategy

Input: Training data $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{M}$, backbone blocks $\{B_1, \dots, B_N\}$, output heads $\{O_1, \dots, O_N\}$
Output: Trained backbone and output heads
Initialization:
Initialize sample weights $w_i^{(0)} = 1/M$
Train full backbone $B_1 \to \cdots \to B_N$ with final head O_N using $\mathcal{L}^{(0)}$
Boosting-guided Depth-adaptive Training:
for $k = 1$ to $N-1$ **do**
  Freeze backbone blocks $\{B_1, \dots, B_k\}$
  Train auxiliary head O_k using representations $\mathbf{h}_k$ and loss $\mathcal{L}^{(k-1)}$
  Compute weighted error ϵ_k and estimator weight α_k
  Update and normalize sample weights $\{w_i^{(k)}\}$
  Train blocks $\{B_{k+1}, \dots, B_N\}$ and final head O_N using $\mathcal{L}^{(k)}$
end for
return Trained backbone and output heads

The predictions of O_k are used to compute the weighted classification error and update the sample weights according to the boosting formulation defined in (4)–(6). These updated weights define a new weighted objective $\mathcal{L}^{(k)}$ that emphasizes samples that are misclassified by the current representation.

Subsequently, the parameters of the backbone block B_k and the preceding ones are frozen, and the remaining blocks $B_{k+1}, \dots, B_N$, together with the final output head O_N, are optimized with respect to the updated weighted loss $\mathcal{L}^{(k)}$. In this way, earlier blocks preserve their ability to represent easier examples, while later blocks are encouraged to represent harder samples given their higher representational capacity. This process is repeated progressively over blocks, resulting in a single network that integrates boosting principles and internal training dynamics. By sequentially training and freezing the intermediate output heads from the input-end to the output-end, the output heads act as boosting-inspired probes that guide depth-wise optimization by identifying harder samples at each representation stage, thereby steering deeper blocks toward more challenging aspects of the task. The complete procedure is summarized in Figure 1 and Algorithm 1.

At inference time, the final prediction is obtained by aggregating the predictions of all output heads using their corresponding boosting weights. Given a test sample $\mathbf{x}$, each output head O_k produces a class score vector $\hat{\mathbf{y}}^{(k)}$. The aggregated class score for class c is computed as

$$\hat{y}_c = \sum_{k=1}^{N} \alpha_k\, \hat{y}_c^{(k)}, \tag{7}$$

where α_k denotes the boosting weight associated with output head O_k. The final predicted label is then given by

$$\hat{y} = \arg \max_{c \in \{1,\dots,C\}} \hat{y}_c. \tag{8}$$

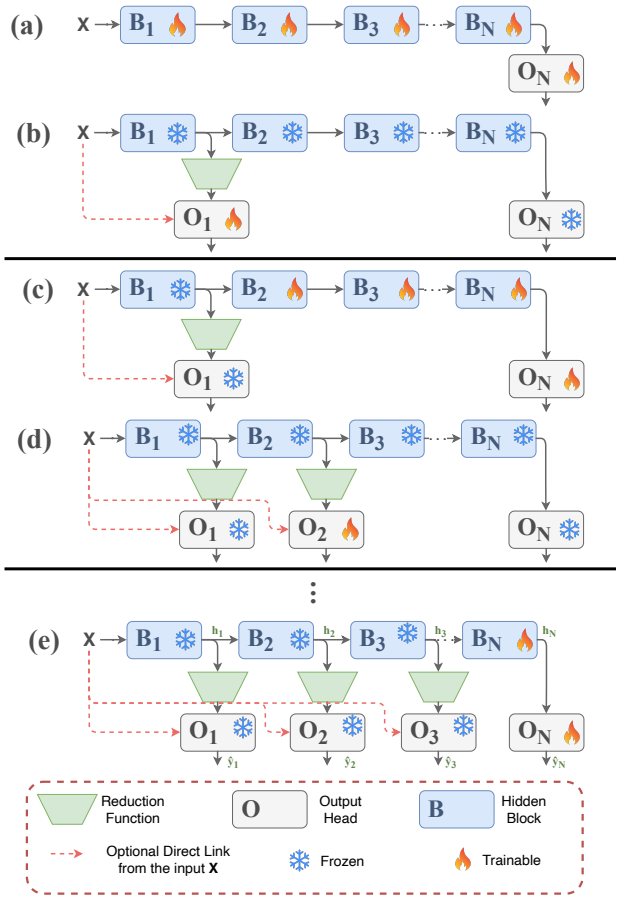

Fig. 1. Overview of the Boosting-guided Progressive Training and Freezing (BPTF) training procedure. The network is first trained end-to-end using only the final output head (a). Auxiliary output heads are then progressively attached from shallow to deeper blocks and trained on frozen representations to estimate sample difficulty (b,d), after which sample weights are updated and deeper backbone blocks together with the final head are optimized under the reweighted objective (c,e). Earlier blocks and heads are frozen to preserve representations for easier samples, while later blocks are progressively specialized to harder examples. Snowflake and flame icons denote frozen and trainable components, respectively, and dashed arrows indicate optional direct links from the input.

This aggregation scheme integrates information from multiple representation depths within a single network, without requiring multiple independently trained models.

IV. Experimental Setup

a) Datasets: We evaluated the proposed method on ten public tabular datasets from the UCI Machine Learning Repository and four widely used small vision benchmarks. Details about the datasets are summarized in the supplementary material. For the tabular datasets, we adopt the same preprocessing pipeline introduced in [11], which has since been employed by several subsequent studies, including [6], [12]. All tabular experiments are conducted using 4-fold cross-validation following [11], leveraging the publicly available split indices[1]. The vision datasets are used with their standard training and evaluation splits.

A. Network Architectures

We consider three backbone architectures covering tabular and vision modalities. For tabular data, we employ an MLP, where the network depth and number of hidden units are tunable hyperparameters. Output heads are attached after every hidden layer, and direct connections from the input features to all output heads are enabled, following common practice in multi-output tabular models [13]. Further, we compare the proposed BPTF-MLP against state-of-the-art tabular models: edRVFL [13], MO-MLP [7], and RFRBoost [14].

For vision tasks, we evaluate one convolutional and one transformer-based architecture using their smallest standard variants, namely DenseNet-121 and Swin Transformer V2-Tiny. Both models are based on the official *torchvision* implementations[2] and are modified to support intermediate output heads. For DenseNet-121 [15], exit points are placed after the initial convolutional block and after each transition layer, with the final exit corresponding to the classification head. For Swin Transformer V2-Tiny [16], exit points are introduced after each hierarchical stage of the network, followed by a final exit after the classification head.

To handle high-dimensional vision features, we apply a reduction function (LayerNorm $\rightarrow$ adaptive average pooling $\rightarrow$ flattening) before each output head, where the pooling resolution is chosen per exit to yield an embedding size on the order of a target budget (set to 4,096). No reduction is used for tabular models (identity mapping).

B. Hyperparameters Tuning

For vision experiments, hyperparameters are tuned using a validation split of the training data. Backbone optimization uses stochastic gradient descent with momentum, where the learning rate, weight decay, and number of epochs are tuned using Bayesian optimization (SMAC [17]) during the initial training and then fixed for all subsequent boosted rounds.

Auxiliary output heads are implemented as linear classifiers, trained using multinomial logistic regression with cross-entropy loss. For each head, the regularization strength is tuned using Bayesian optimization, and sample weights derived from the boosting procedure are incorporated during training. All tuning decisions are made based on validation accuracy.

All competing models are tuned using SMAC under the same computational budget, with hyperparameter search ranges matching those reported in their respective original papers. More details about the hyperparameters and their tuning ranges can be found in the supplementary material.

V. Results and Discussion

A. Vision Datasets

Table I reports test accuracy for two vision backbones—DenseNet-121 (DN-121) and Swin Transformer V2-Tiny

[1]https://github.com/lingping-fuzzy/UCI-data-correct-split/tree/main
[2]https://github.com/pytorch/vision

(SwinV2-T)—under three settings: Standard training, BPTF training with inference using only the final backbone head (BPTF-Backbone), and BPTF with ensemble inference over all block-wise heads (BPTF-Ens). In both architectures and all four datasets, BPTF consistently improves over standard training. In particular, BPTF-Backbone yields systematic gains over the Standard baseline on most datasets, demonstrating that the proposed BPTF optimization improves the learned backbone itself rather than solely benefiting ensemble inference. This effect is more pronounced for Swin V2-Tiny, where BPTF-Backbone improves accuracy on all tested datasets.

Notably, the auxiliary heads introduce only a modest parameter overhead, ranging from 5–10% for Swin-V2-T and 20–38% for DN-121 (Table I). The absolute auxiliary parameter count is comparable across backbones for the same dataset; the higher relative overhead for DN-121 reflects its smaller backbone. These extra parameters reside entirely in the auxiliary heads and incur no additional cost when inference is performed via BPTF-Backbone alone.

Further performance improvements are obtained with BPTF-Ens, which outperforms both Standard and BPTF-Backbone in the majority of cases. This indicates that intermediate output heads capture complementary information at different representation depths, and that their boosting-weighted aggregation provides a more robust classifier than the final head alone. While BPTF-Backbone occasionally matches the Standard baseline (DN-121 on Caltech101) or underperforms it (DN-121 on Flowers), BPTF-Ens achieves higher accuracy than BPTF-Backbone in most settings and improves over the Standard baseline in all but one case (DN-121 on Flowers), demonstrating the benefit of aggregating intermediate output heads trained under the BPTF procedure.

To gain deeper insight into how predictive performance evolves across depth, Fig. 2 visualizes, for selected backbone–dataset pairs, three quantities as a function of head index: (i) per-head error, (ii) cumulative error from the first head (i.e., progressively adding heads), and (iii) reverse cumulative error from the last head (i.e., progressively removing earlier heads). The per-head error curve (blue) reports the error of each output head in isolation; it is highest for the earliest heads reflecting that shallow representations are still largely generic and not yet sufficiently task-specialized. The cumulative error curve (green) shows the error obtained by progressively aggregating predictions from the first head up to head N using the learned boosting weights, with the final point corresponding to the full BPTF-Ens predictor. In contrast, the reverse cumulative error curve (red) starts from the final head alone (i.e., BPTF-Backbone) and then successively incorporates earlier heads. Notably, for some experiments (e.g., Swin-V2-T on dogs), even adding the very first head–despite its poor standalone performance–consistently reduces error, albeit slightly, as reflected by the slope between the last two points of this curve. Overall, these trends demonstrate that while early heads are weak predictors individually, they encode complementary information that meaningfully improves the boosted ensemble, validating the use of all intermediate heads in BPTF inference.

TABLE I

TEST ACCURACY (%) ON VISION DATASETS FOR STANDARD END-TO-END TRAINING, BPTF USING ONLY THE FINAL BACKBONE HEAD (BPTF-BACKBONE), AND BPTF WITH ENSEMBLE INFERENCE OVER ALL BLOCK-WISE HEADS (BPTF-ENS). BOLD DENOTES THE BEST RESULT PER ROW; UNDERLINE MARKS RESULTS THAT OUTPERFORM STANDARD BUT ARE NOT THE BEST. EXTRA PARAMS REPORTS THE TOTAL ADDITIONAL PARAMETERS INTRODUCED BY ALL AUXILIARY OUTPUT HEADS (ABSOLUTE AND AS A PERCENTAGE OF THE BACKBONE).

Backbone	Dataset	Standard	BPTF-Backbone	BPTF-Ens	Extra Params
DN-121	birds	75.388	**76.648**	76.631	2.7M (38%)
	caltech101	94.315	94.315	**94.351**	1.4M (20%)
	dogs	78.893	79.254	**79.289**	1.6M (23%)
	flowers	**94.503**	94.113	94.080	1.4M (20%)
Swin-V2-T	birds	81.533	82.050	**82.240**	2.6M (10%)
	caltech101	94.679	95.590	**95.700**	1.4M (5%)
	dogs	88.485	89.336	**89.441**	1.6M (6%)
	flowers	95.284	96.243	**96.292**	1.4M (5%)

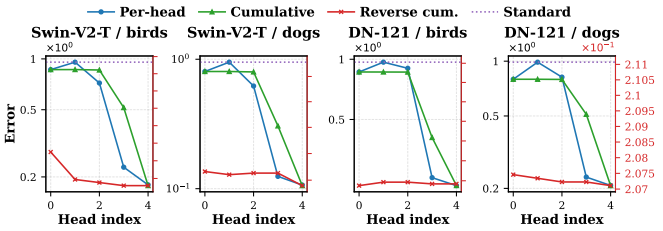

Fig. 2. Per-head, cumulative, and reverse cumulative test error (1–accuracy) as a function of head index for two backbones and four datasets. Both y-axes are shown in log scale: the left axis corresponds to per-head and cumulative error, while the right (red) axis corresponds to reverse cumulative error together with the Standard baseline error. The final point on both cumulative curves corresponds to the full BPTF-Ens predictor.

B. Tabular Datasets

Table II reports average test accuracy on ten tabular benchmarks, comparing BPTF against Standard training (MLP) as well as state-of-the-art tabular models, including edRVFL [13], MO-MLP [7], and RFRBoost [14]. Results for BPTF are shown for both the final backbone head (BPTF-MLP) and the boosting-weighted ensemble of all intermediate heads (BPTF-Ens). Consistent with the trends observed in vision tasks, BPTF-Ens achieves the best overall performance, obtaining a higher rank on all datasets compared to standard training and the lowest (best) mean rank. However, in contrast to the vision setting, BPTF-MLP performs worse than both Standard training and BPTF-Ens on most datasets, often obtaining the lowest rank. This suggests that, for tabular MLPs, the primary gains of BPTF arise from leveraging intermediate output heads rather than from improvements to the final backbone representation alone. This can be attributed, in part, to the fact that direct links are enabled for the backbone MLP.

When compared against state-of-the-art tabular models, BPTF-Ens achieves the lowest overall average rank over all benchmarks, indicating strong and consistent performance in diverse datasets. Though different methods achieve the best result on individual tasks, BPTF-Ens maintains high accuracy across all benchmarks, reflecting the effectiveness of boosting-weighted aggregation of intermediate representations.

TABLE II
AVERAGE TEST ACCURACY (%) ON TABULAR BENCHMARKS FOR STANDARD END-TO-END TRAINING, BPTF USING ONLY THE FINAL BACKBONE HEAD (BPTF-MLP), AND BPTF WITH ENSEMBLE INFERENCE OVER ALL BLOCK-WISE HEADS (BPTF-ENS). NUMBERS IN PARENTHESES DENOTE AVERAGE RANK ACROSS 4-FOLD CROSS-VALIDATION, AND THE LAST ROW REPORTS THE MEAN RANK OVER ALL DATASETS.

Datasets	edRVFL [13]	MO-MLP [7]	RFRBoost [14]	MLP	**BPTF-MLP**‡	**BPTF-Ens**‡
BC-Wisc-Prog	**79.85% (1)**	75.41% (3)	75.05% (5)	76.99% (2)	72.30% (6)	75.31% (4)
Breast-Tissue	**73.41% (1)**	70.80% (3)	68.52% (5)	69.32% (4)	66.70% (6)	70.80% (2)
CTG-10	81.52% (4)	**84.92% (1)**	83.36% (3)	81.05% (5)	80.23% (6)	83.42% (2)
Chess-KRVKP	98.80% (4)	**99.16% (1)**	98.49% (6)	98.93% (3)	98.79% (5)	99.01% (2)
Contraceptive	53.49% (4)	54.19% (2)	**55.48% (1)**	51.94% (5)	45.09% (6)	53.91% (3)
Ionosphere	93.21% (2)	91.76% (4)	91.11% (6)	92.81% (3)	91.48% (5)	**94.15% (1)**
Iris	97.43% (2)	96.76% (4)	**97.84% (1)**	95.95% (5)	88.18% (6)	97.09% (3)
Oocytes-5b	92.84% (4)	**94.25% (1)**	91.58% (6)	93.03% (3)	92.77% (5)	93.62% (2)
Seeds	91.78% (5)	89.90% (6)	93.37% (2)	92.45% (4)	92.84% (3)	**93.70% (1)**
Synth-Control	98.12% (5)	**99.38% (1)**	98.50% (4)	98.75% (2)	92.22% (6)	98.68% (3)
Avg. Rank	3.20	2.60	3.90	3.60	5.40	**2.30**

‡ Indicates models proposed in this study. Full dataset names are given in the supplementary material.

C. Discussion

From a representation learning perspective, BPTF can be viewed as introducing a form of depth-wise specialization. Early blocks are trained under the initial uniform sample distribution and therefore tend to capture broadly separable patterns, whereas later blocks are progressively optimized under reweighted distributions that emphasize harder examples. In this sense, intermediate output heads serve not merely as auxiliary predictors, but as probes that shape the optimization of deeper representations. A possible explanation for the different behavior of BPTF-Backbone across vision and tabular settings is the reweighting mechanism, which increasingly emphasizes harder samples and may bias the final head toward difficult cases. This effect may be more pronounced in tabular MLPs due to their less hierarchical representations, while BPTF-Ens can help mitigate it by combining predictions from heads trained at different depths.

VI. CONCLUSION

In this paper, we proposed *boosting-guided progressive training and freezing* (BPTF), a unified training framework that combines intermediate supervision from multiple auxiliary output heads with boosting-inspired sample reweighting to guide depth-wise representation learning within a single deep network. Unlike prior greedy or multi-output approaches, BPTF preserves the full representational capacity of the backbone while emphasizing harder examples through intermediate heads. Across vision and tabular benchmarks, BPTF consistently outperforms standard end-to-end training and state-of-the-art methods, yielding stronger backbone representations in vision via multi-head guidance and competitive tabular performance mainly through boosting-weighted ensemble inference. A limitation of BPTF is the additional training rounds introduced by the boosting procedure, which increase overall training time; however, this overhead remains moderate in practice since later rounds are initialized from previously trained weights and therefore typically converge in fewer epochs, albeit under a reweighted data distribution. At inference time, BPTF-Backbone incurs no additional cost relative to Standard training, whereas BPTF-Ens introduces extra computation proportional to the number of exit points, the dimensionality of intermediate representations, and the number of classes. Moreover, progressively reweighting toward harder samples may bias later rounds toward over-specialization on difficult examples, but this effect is mitigated in BPTF-Ens by combining predictions from earlier, more uniformly trained heads. Future work will investigate head-pruning strategies to reduce inference cost, richer non-linear intermediate heads, faster training schemes for auxiliary heads such as closed-form solutions, more adaptive reweighting strategies, head placement strategies, and evaluation on additional benchmarks.

REFERENCES

[1] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[2] C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Artificial intelligence and statistics*. Pmlr, 2015, pp. 562–570.

[3] C. Li, M. Z. Zia, Q.-H. Tran, X. Yu, G. D. Hager, and M. Chandraker, "Deep supervision with intermediate concepts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 41, no. 8, pp. 1828–1843, 2018.

[4] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Branchynet: Fast inference via early exiting from deep neural networks," in *2016 23rd international conference on pattern recognition (ICPR)*. IEEE, 2016, pp. 2464–2469.

[5] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, 2006.

[6] H. Aly, A. K. Al-Ali, and P. N. Suganthan, "Boosted multilayer feedforward neural network with multiple output layers," *Pattern Recognition*, vol. 156, p. 110740, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320324004916

[7] M. Hamdy, A. Al-Ali, P. N. Suganthan, and H. Aly, "Hybrid training of deep neural networks with multiple output layers for tabular data classification," *Pattern Recognition*, p. 112035, 2025.

[8] H. Yu, H. Li, G. Hua, G. Huang, and H. Shi, "Boosted dynamic neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 37, no. 9, 2023, pp. 10 989–10 997.

[9] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[10] E. Belilovsky, M. Eickenberg, and E. Oyallon, "Greedy layerwise learning can scale to imagenet," in *International conference on machine learning*. PMLR, 2019, pp. 583–593.

[11] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[12] Q. Shi, M. Hu, P. N. Suganthan, and R. Katuwal, "Weighting and pruning based ensemble deep random vector functional link network for tabular data classification," *Pattern Recognition*, vol. 132, p. 108879, 2022.

[13] Q. Shi, R. Katuwal, P. Suganthan, and M. Tanveer, "Random vector functional link neural network based ensemble deep learning," *Pattern Recognition*, vol. 117, p. 107978, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320321001655

[14] N. Zozoulenko, T. Cass, and L. Gonon, "Random feature representation boosting," in *Forty-second International Conference on Machine Learning, ICML*, 2025.

[15] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[16] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong *et al.*, "Swin transformer v2: Scaling up capacity and resolution," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 12 009–12 019.

[17] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter, "Smac3: A versatile bayesian optimization package for hyperparameter optimization," *Journal of Machine Learning Research*, vol. 23, no. 54, pp. 1–9, 2022.