

Capacity-Aware Gradient-Boosted CNNs and DNNs for Efficient Image Classification

Jujjuvarapu Nagendra¹, Krishna Kumar Singh²

¹Department of Computer Science and Engineering, Rajiv Gandhi University of Knowledge and Technologies, IIIT RK Valley 516330

²Assistant Professor, Department of Computer Science and Engineering, Rajiv Gandhi University of Knowledge and Technologies, IIIT Nuzvid 521202

Abstract

Gradient-boosted convolutional neural networks (GB-CNNs) grow a classifier by sequentially adding and freezing dense layers fitted to pseudo-residuals, yet the parameter cost of this additive head is seldom examined. This article proposes a capacity-aware GB-CNN (CW-GB-CNN) built on a closed-form capacity model of the additive dense head, $P_{\text{head}}(W, M)$, which gives the head's parameter count from the layer width W and the number of boosting rounds M . The width of the boosted dense layers is selected by weighing the modeled parameter cost against the accuracy it returns, rather than being fixed a priori. CW-GB-CNN is compared with jointly trained networks, a gradient-boosted deep network, and classical tree ensembles on five public image datasets using rank-based statistical tests. Boosting reduced the test error of the jointly trained CNN by 49.7% on average, and CW-GB-CNN achieved the best mean accuracy (95.66%) and mean rank (1.6). On CIFAR-10, its 20-neuron head matched a conventional 128-neuron head (87.77% versus 87.86%) with 6.66 times fewer head parameters and 1.70 times fewer total parameters.

Keywords: Convolutional Neural Networks, Deep Neural Networks, Gradient Boosting, GB-CNN, Capacity-Aware GB-CNN (CW-GB-CNN), GB-DNN, Pseudo-Residual Learning, Dense-Layer Width, Parameter Efficiency, Ensemble Learning, Statistical Model Comparison, Image Classification

1. Introduction

Deep learning has become the dominant approach to visual recognition since AlexNet [1] demonstrated that deep convolutional neural networks (CNNs) could surpass hand-engineered pipelines on large-scale benchmarks. Convolutional architectures now underpin image classification [2], object detection [3], anomaly detection [4], and medical image segmentation [5], and increasingly deep designs such as VGG [6] and ResNet [7] have shown that representational depth is a principal driver of accuracy [8]. Depth, however, complicates optimization: vanishing and exploding gradients, internal covariate shift, and overfitting must be counteracted with batch normalization [9], careful initialization [10], residual connections [7], and dropout [11], while all parameters are still optimized jointly by end-to-end back-propagation.

Gradient boosting offers a different route to building accurate predictors. Formulated as functional gradient descent [12, 13], it constructs an additive model stage by stage, each new learner fitting the pseudo-residuals—the negative gradient of the loss—left by the current ensemble. Gradient-boosted

decision trees such as XGBoost [14] and LightGBM [15] remain the reference methods for tabular data [16, 17]. This principle has been transferred to neural networks as gradient-boosted dense (GB-DNN) and convolutional (GB-CNN) networks [18, 19], in which dense layers are added one per boosting iteration and frozen once trained.

Despite encouraging results, the capacity implications and the external validity of gradient-boosted networks have received limited attention. The additive head is precisely the component that boosting enlarges at every iteration, and in CNNs the fully connected block is known to dominate the parameter budget [20, 21]. Yet the width of each additive layer is normally fixed a priori, and its accuracy–capacity trade-off is seldom quantified. Moreover, comparisons are typically restricted to jointly trained neural counterparts, rarely include classical ensembles evaluated under the same protocol, and rarely apply the non-parametric multi-dataset tests recommended for classifier comparison [22]. From this analysis, three research gaps are identified:

- **G1: Cross-paradigm evidence.** There is no unified benchmark placing GB-CNN and GB-DNN alongside jointly trained networks and feature-based tree ensembles across heterogeneous image domains (grayscale digits and apparel, natural color objects, agricultural grains, and hand gestures).
- **G2: Capacity awareness.** The parameter cost of the additive dense head as a function of layer width and number of boosting rounds has not been formalized, and the accuracy obtained per additional parameter has not been measured.
- **G3: Statistical validity.** Claims of superiority of boosted networks are seldom supported by rank-based significance testing over multiple datasets.

In this article, capacity-aware means that the width of the dense layers added by boosting is treated as a design variable whose parameter cost is given by the proposed capacity model of the additive dense head (Eq. (7)) and weighed against the accuracy it returns, instead of being fixed a priori. Applying this model to GB-CNN yields the model proposed in this article, the capacity-aware gradient-boosted convolutional neural network (CW-GB-CNN), which uses a compact additive head of 20 neurons per boosted dense layer. All other classifiers serve as baselines: XGBoost, a decision tree (DT), and a random-forest/decision-tree voting ensemble (RF+DT) built on histogram-of-oriented-gradients (HOG) features, together with the jointly trained CNN and DNN and the gradient-boosted GB-DNN. A conventional GB-CNN with a 128-neuron head (GB-CNN-128) serves as the wide-head reference. HOG [23] is a hand-crafted image descriptor that divides an image into small cells, computes the direction and strength of the local intensity changes (edges) within each cell, and summarizes them as histograms of edge orientations, thereby encoding local shape without any learning. Since a new model is justified only if it either improves performance significantly or achieves similar performance with fewer parameters, CW-GB-CNN is assessed against both criteria: against the baselines for accuracy and against GB-CNN-128 for parameter efficiency.

To close these gaps, this article answers four research questions. **RQ1:** Does gradient boosting consistently improve CNNs and DNNs across diverse image datasets? **RQ2:** How does dense-layer width affect the accuracy and parameter efficiency of GB-CNN? **RQ3:** Are the observed performance differences statistically significant across datasets? **RQ4:** How do boosted networks compare with HOG-based tree ensembles? The main contributions are as follows.

1. A capacity model of the additive dense head: a closed-form expression $P_{\text{head}}(W, M)$ for the parameters of the boosted dense layers (Eq. (7)), together with three efficiency indicators—head share, marginal accuracy yield, and accuracy per 10^5 parameters—and a width-selection rule built

on them (Eq. (11), Algorithm 1). Applied to 20- and 128-neuron heads on CIFAR-10, the model leads to the proposed CW-GB-CNN, which matches the accuracy of GB-CNN-128 with 6.66 times fewer head parameters and 1.70 times fewer total parameters (Sections 3.5 and 5.4).

2. A unified multi-dataset benchmark of seven classifiers from three paradigms (feature-based tree ensembles, jointly trained networks, and gradient-boosted networks) on five public datasets using four complementary metrics (Section 5.1).
3. A statistical assessment based on relative error reduction, Friedman and Iman–Davenport tests, Nemenyi critical-difference analysis, and Wilcoxon signed-rank tests (Section 5.3).
4. A HOG-based decision-tree and random-forest/decision-tree voting extension that serves as a feature-based reference for the boosted networks (Sections 3.6 and 5.5).

The remainder of the article is organized as follows. Section 2 reviews related work and synthesizes the gaps. Section 3 presents the framework and the capacity model. Section 4 describes the experimental protocol. Section 5 reports the results, Section 6 discusses implications and threats to validity, and Section 7 concludes the article.

2. Related Work

2.1 Deep Convolutional Architectures

AlexNet [1] established deep CNNs as the state of the art for large-scale image classification, and subsequent work extended CNNs to transfer learning with web-data augmentation [2], real-time detection with region proposal networks [3], image-encoded network anomaly detection [4], and 3-D lesion segmentation with conditional random fields [5]. VGG [6] showed that stacks of small 3×3 filters capture hierarchical abstractions, deconvolutional visualization clarified what these hierarchies encode [8], and ResNet [7] made networks with hundreds of layers trainable through identity shortcuts. Batch normalization [9], principled initialization [10], dropout [11], and adaptive optimizers [24] complete the toolbox that makes joint end-to-end training practical, although it still requires substantial regularization and computation.

2.2 Gradient Boosting and Tree-Based Ensembles

Boosting originated in the theory of weak learnability [25] and was popularized by AdaBoost [26]. Friedman [12] and Mason et al. [13] reinterpreted it as gradient descent in function space, allowing arbitrary differentiable losses. Efficient implementations such as XGBoost [14] and LightGBM [15] dominate tabular learning [16, 17]. Classification and regression trees [27], random forests [28], and voting combinations of heterogeneous classifiers [29] remain attractive for their low training cost and interpretability; for images, they are usually applied to engineered descriptors such as HOG [23]. Neural architectures designed to rival trees on tabular data, including TabNet [30], DNF-Net [31], and deep neural decision trees [32], further illustrate the convergence of the two families.

2.3 Boosting Deep Neural Networks

Several lines of work have combined boosting with neural networks; Table 1 summarizes representative approaches. Bengio et al. [33] trained convex neural networks by greedily adding hidden units. Moghimi et al. [34] boosted CNNs, and Zhang et al. [35] built gradient-boosted random convolutional networks for remote-sensing scene classification. BoostResNet [36] trained residual blocks sequentially with boosting guarantees, ResFGB [37] related residual networks to functional gradient boosting, AdaNet [38] adaptively grew network structure with learning guarantees, and GrowNet [39] boosted shallow networks for tabular tasks. GB-DNN and GB-CNN [18, 19] are the most direct precursors of this study:

they add one dense layer per iteration, freeze previously trained dense layers, and—for GB-CNN—fine-tune the convolutional backbone on pseudo-residuals.

Table 1: Representative Approaches Combining Boosting and Neural Networks

Approach	Unit added per stage	Primary domain
Convex NN [33]	Hidden neuron (greedy)	Generic
Boosted CNN [34]	CNN weak learner	Images
GBRCN [35]	Random convolutional network	Remote sensing
AdaNet [38]	Subnetwork (adaptive)	Generic
BoostResNet [36]	Residual block	Images
ResFGB [37]	Residual function	Generic
GrowNet [39]	Shallow network	Tabular
GB-DNN / GB-CNN [18]	Dense layer (frozen) + fine-tuned conv. layers	Images, tabular
CW-GB-CNN (this work)	Dense layer with capacity-aware width analysis	Five image domains

2.4 Capacity and Parameter Efficiency of Classification Heads

Fully connected layers account for most parameters of classical CNNs; network-in-network replaced them with global average pooling [20], and deep compression showed that they can be pruned and quantized aggressively with little loss [21]. Efficiency-driven families such as MobileNets [40] and EfficientNet [41] made the accuracy–parameter trade-off an explicit design objective, and over-parameterized networks are known to fit training data far beyond what generalization requires [42]. These observations motivate examining whether the dense head grown by gradient boosting is appropriately sized—a question not addressed by the boosting studies above.

2.5 Synthesis of Research Gaps

Table 2 maps the gaps identified in Section 1 to the research questions, the corresponding contributions, and the sections in which they are addressed.

Table 2: Mapping of Research Gaps to Questions, Contributions, and Sections

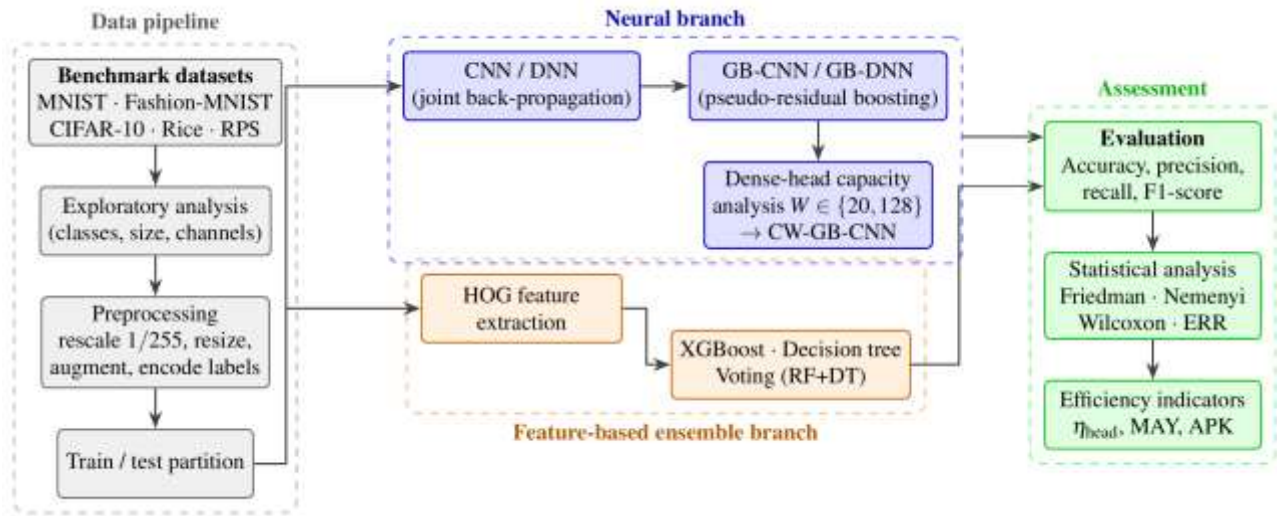
Gap	Limitation in prior work	Response in this article	Sec.
G1/RQ1	Boosted networks compared mainly with neural counterparts	7 models, 3 paradigms, 5 datasets, 4 metrics	5.1, 5.2
G2/RQ2	Head width fixed a priori; cost unquantified	Closed-form $P_{\text{head}}(W, M)$; efficiency indicators	3.5, 5.4
G3/RQ3	No multi-dataset significance testing	Friedman, Nemenyi CD, Wilcoxon tests	5.3
RQ4 (G1)	No feature-ensemble baseline in the same protocol	HOG DT/voting extension	3.6, 5.5

3. Proposed Framework and Methodology

3.1 Framework Overview

Figure 1 outlines the framework. Image datasets are explored, preprocessed, and divided into training and test partitions. Two model branches are trained in parallel: a neural branch, which produces jointly trained CNN and DNN baselines and their gradient-boosted counterparts, and a feature-based branch, which extracts HOG descriptors for XGBoost, a decision tree, and a voting ensemble. All models are evaluated on the held-out test data with identical metrics, the GB-CNN head is subjected to the capacity analysis that yields the proposed CW-GB-CNN.

Figure 1: Proposed Framework Architecture



3.2 Problem Formulation

Let $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$ be a labelled dataset in which $\mathbf{x}_i \in \mathbb{R}^{H \times W \times \text{img} \times C}$ is an image and $\mathbf{y}_i \in \{0, 1\}^K$ is the one-hot encoding of its class among K categories. A classifier outputs a logit vector $F(\mathbf{x}) \in \mathbb{R}^K$ that is mapped to class probabilities by the softmax function,

$$p_k(\mathbf{x}) = \frac{\exp(F_k(\mathbf{x}))}{\sum_{j=1}^K \exp(F_j(\mathbf{x}))}, \quad k = 1, \dots, K, \quad (1)$$

and training minimizes the empirical categorical cross-entropy

$$\mathcal{L}(F) = -\frac{1}{N} \sum_{i=1}^N \sum_{k=1}^K y_{ik} \log p_k(\mathbf{x}_i). \quad (2)$$

3.3 Gradient-Boosted Convolutional Neural Network

GB-CNN represents the classifier as an additive expansion constructed over M boosting iterations [12, 18]:

$$F_m(\mathbf{x}) = F_{m-1}(\mathbf{x}) + \nu \rho_m h_m(\mathbf{x}; \boldsymbol{\theta}_m), \quad m = 1, \dots, M, \quad (3)$$

where h_m is the network hypothesis at iteration m , θ_m collects the parameters of the newly added dense layer and the fine-tuned convolutional filters, ρ_m is a step size, and $\nu \in (0, 1]$ is the shrinkage (learning-rate) factor. The negative gradient of (2) with respect to the k -th logit gives the pseudo-residuals

$$r_{ik}^{(m)} = - \left[\frac{\partial \mathcal{L}(y_i, F(\mathbf{x}_i))}{\partial F_k(\mathbf{x}_i)} \right]_{F=F_{m-1}} = y_{ik} - p_k^{(m-1)}(\mathbf{x}_i), \quad (4)$$

which measure, per class, how far the current ensemble is from the target. The new hypothesis is fitted to these residuals by minimizing the mean squared error

$$\theta_m = \arg \min_{\theta} \frac{1}{NK} \sum_{i=1}^N \sum_{k=1}^K \left(r_{ik}^{(m)} - h_{m,k}(\mathbf{x}_i; \theta) \right)^2. \quad (5)$$

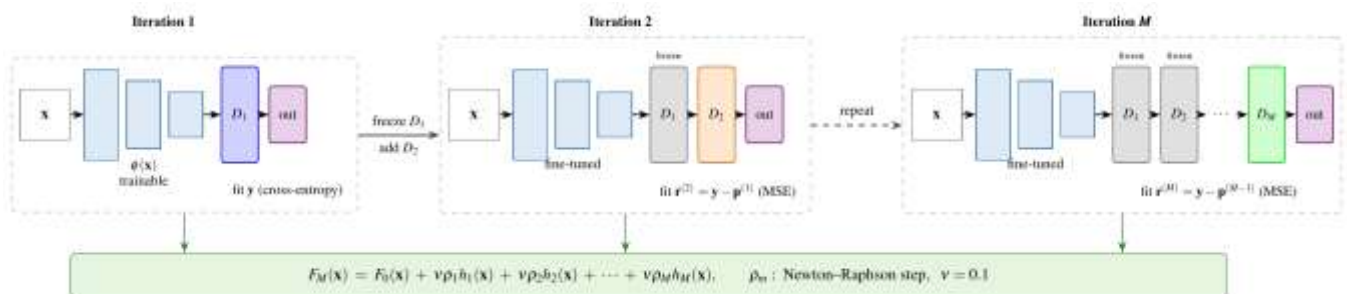
Given h_m , the step size is obtained with a single Newton–Raphson step on the cross-entropy along direction h_m :

$$\rho_m = \frac{\sum_i \sum_k r_{ik}^{(m)} h_{m,k}(\mathbf{x}_i)}{\sum_i \mathbf{h}_m(\mathbf{x}_i)^T (\text{diag}(\mathbf{p}_i) - \mathbf{p}_i \mathbf{p}_i^T) \mathbf{h}_m(\mathbf{x}_i)}, \quad (6)$$

where $\mathbf{p}_i = \mathbf{p}^{(m-1)}(\mathbf{x}_i)$ and the matrix in the denominator is the Hessian of the softmax cross-entropy with respect to the logits.

Architectural realization. The backbone $\phi(\cdot)$ comprises three double-convolution blocks with 32, 64, and 128 filters of size 3×3 , each followed by batch normalization, ReLU activation, 2×2 max pooling, and dropout with rates 0.2, 0.3, and 0.4, respectively. For a $32 \times 32 \times 3$ input, flattening the final $4 \times 4 \times 128$ tensor yields a $d = 2048$ -dimensional representation. At the first iteration, a dense layer D_1 of width W and the K -way output projection are trained on the original targets. At each later iteration, previously trained dense layers are frozen, a new dense layer D_m of width W is inserted immediately before the output projection, and D_m is trained on (5) while the convolutional filters are fine-tuned. Frozen layers preserve what earlier stages learned, so the new layer concentrates on the residual structure that the current model has not yet captured. Figure 2 illustrates the procedure and Algorithm 1 summarizes it.

Figure 2: GB-CNN Training Procedure



Algorithm 1: CW-GB-CNN Training With Additive Dense Layers

Input: Training set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$; candidate widths $\mathcal{W}$; rounds M_W for each $W \in \mathcal{W}$; backbone output size d ; classes K ; backbone parameters P_{backbone} ; tolerance ε ; shrinkage v ; epochs E ; patience P

Output: Selected width W^* ; additive classifier F_M

Stage A: capacity model and width selection

- 1: **for** each $W \in \mathcal{W}$ **do**
- 2: $P_{\text{head}}(W, M_W) \leftarrow (d+1)W + (M_W-1)(W+1)W + (W+1)K$ ▷ capacity model, Eq. (7)
- 3: $P_{\text{total}} \leftarrow P_{\text{backbone}} + P_{\text{head}}; \eta_{\text{head}} \leftarrow P_{\text{head}}/P_{\text{total}}$ ▷ head share, Eq. (8)
- 4: $F^{(W)} \leftarrow \text{BOOSTEDTRAINING}(W, M_W)$ ▷ Stage B
- 5: $\text{Acc}_W \leftarrow \text{accuracy of } F^{(W)}$
- 6: **end for**
- 7: $W^* \leftarrow \text{narrowest } W \text{ with } \text{Acc}_W \geq \max_{W'} \text{Acc}_{W'} - \varepsilon$ ▷ Eq. (11)
- 8: Report $\text{MAY}_{W^* \rightarrow W}$ for each wider W ▷ Eq. (9)
- 9: **return** W^* and $F^{(W^*)}$

Stage B: additive training of width- W dense layers

- 10: **procedure** BOOSTEDTRAINING(W, M)
- 11: Initialize backbone ϕ (3 double-conv blocks: 32/64/128 filters, BN, ReLU, max-pool, dropout)
- 12: Attach dense layer D_1 (width W , $(d+1)W$ parameters) and K -way output; train on $\mathbf{y}$ with cross-entropy for $\leq E$ epochs (early stopping, patience P)
- 13: $F_1(\mathbf{x}) \leftarrow \text{logits of the trained network}$
- 14: **for** $m = 2$ **to** M **do**
- 15: $p_{ik}^{(m-1)} \leftarrow \text{softmax}(F_{m-1}(\mathbf{x}_i))_k$ ▷ Eq. (1)
- 16: $r_{ik}^{(m)} \leftarrow y_{ik} - p_{ik}^{(m-1)}$ ▷ pseudo-residuals, Eq. (4)
- 17: Freeze $D_1, \dots, D_{m-1}$
- 18: Insert new dense layer D_m (width W , $(W+1)W$ parameters) before the output layer
- 19: Train D_m and fine-tune ϕ on $\{(\mathbf{x}_i, \mathbf{r}_i^{(m)})\}$ with MSE (Eq. (5)), $\leq E$ epochs, patience P
- 20: $\rho_m \leftarrow \text{Newton-Raphson step (Eq. (6))}$
- 21: $F_m(\mathbf{x}) \leftarrow F_{m-1}(\mathbf{x}) + v \rho_m h_m(\mathbf{x})$ ▷ Eq. (3)
- 22: **end for**
- 23: **return** F_M
- 24: **end procedure**

3.4 Gradient-Boosted Deep Neural Network

GB-DNN follows the same additive scheme (3)–(6) but operates on the vectorized image, $\mathbf{x} \mapsto \text{vec}(\mathbf{x})$, without a convolutional backbone. Each iteration appends a dense layer that is fitted to the pseudo-residuals while earlier dense layers are frozen [18]. Its jointly trained counterpart, DNN, consists of stacked fully connected layers optimized end-to-end by back-propagation, and the CNN baseline shares the backbone of GB-CNN but is trained jointly in a single stage. Pairing each boosted model with its non-boosted counterpart isolates the effect of the boosting procedure.

3.5 Capacity Model of the Additive Dense Head

To address G2, this article proposes a capacity model of the additive dense head: the parameter count of the head is expressed as a function of the layer width W , the number of boosting rounds M , the backbone output dimension d , and the number of classes K . The first dense layer maps $d \rightarrow W$, each of the $M - 1$ additive layers maps $W \rightarrow W$, and the output layer maps $W \rightarrow K$. Counting weights and biases gives

$$P_{\text{head}}(W, M) = \underbrace{(d+1)W}_{D_1} + \underbrace{(M-1)(W+1)W}_{D_2, \dots, D_M} + \underbrace{(W+1)K}_{\text{output}} \quad (7)$$

Equation (7) is the capacity model of the additive dense head. It makes the cost of each design choice explicit before any training is done and serves as the criterion for choosing the width of the boosted layers, because the fully connected block dominates the parameter budget of CNNs [20, 21]. For CIFAR-10 ($d = 2048$, $K = 10$), (7) yields $P_{\text{head}}(20, 3) = 40,980 + 2 \times 420 + 210 = 42,030$ and $P_{\text{head}}(128, 2) = 262,272 + 16,512 + 1,290 = 280,074$, in exact agreement with the model summaries. Equation (7) exposes two properties. First, because $d \gg W$, the term $(d+1)W$ dominates for practical widths, so head size grows almost linearly with W and only weakly with M . Second, the additive layers contribute $\mathcal{O}(MW^2)$ parameters, so extra boosting rounds are inexpensive for narrow heads but increasingly costly for wide ones.

The following indicators quantify efficiency. With $P_{\text{total}} = P_{\text{backbone}} + P_{\text{head}}$, the head share is

$$\eta_{\text{head}} = \frac{P_{\text{head}}}{P_{\text{total}}} \times 100\%. \quad (8)$$

For two configurations a (smaller) and b (larger) with test accuracies Acc_a , Acc_b in percent, the marginal accuracy yield is the accuracy gained per 10^5 additional parameters,

$$\text{MAY}_{a \rightarrow b} = \frac{\text{Acc}_b - \text{Acc}_a}{(P_{\text{total},b} - P_{\text{total},a})/10^5}, \quad (9)$$

and the accuracy per 10^5 parameters is $\text{APK} = \text{Acc}/(P_{\text{total}}/10^5)$. Finally, the benefit of boosting on each dataset is summarized by the relative error reduction of a boosted model over its baseline,

$$\text{ERR} = \frac{(1 - \text{Acc}_{\text{base}}) - (1 - \text{Acc}_{\text{GB}})}{1 - \text{Acc}_{\text{base}}} \times 100\%, \quad (10)$$

which, unlike absolute accuracy differences, is comparable between easy and difficult datasets.

Proposed model. CW-GB-CNN is the gradient-boosted CNN whose dense-layer width is chosen by the capacity model (7) and the indicators (8)–(9) (Algorithm 1), rather than fixed a priori. For a set of candidate widths $\mathcal{W}$, with Acc_W the accuracy of the boosted network of width W and ϵ an accuracy tolerance in percentage points, the selected width is the narrowest one that preserves accuracy,

$$W^* = \min \left\{ W \in \mathcal{W} : \text{Acc}_W \geq \max_{W' \in \mathcal{W}} \text{Acc}_{W'} - \varepsilon \right\}, \quad (11)$$

because the $(d + 1)W$ projection dominates P_{head} and head size therefore grows almost linearly with W . On CIFAR-10 with $\mathcal{W} = \{20, 128\}$, the 0.09 pp accuracy difference lies within a tolerance of $\varepsilon = 0.1$ pp, which gives $W^* = 20$ with $M = 3$ boosting rounds. The configuration with $W = 128$ and $M = 2$ (GB-CNN-128) represents a conventional wide head and is evaluated only as a reference.

3.6 HOG-Based Tree Ensemble Extension

The feature-based branch represents each image by a HOG descriptor [23]. Gradient magnitudes $g = \sqrt{g_x^2 + g_y^2}$ and orientations $\vartheta = \arctan(g_y/g_x)$ are accumulated into orientation histograms over local cells, which are contrast-normalized over overlapping blocks and concatenated into a feature vector $\mathbf{z}$. Three learners operate on $\mathbf{z}$.

Decision tree. A CART tree [27] recursively partitions the feature space by choosing, at node t , the split that maximizes the decrease of the Gini impurity $G(t) = 1 - \sum_{k=1}^K \pi_{tk}^2$, where π_{tk} is the proportion of class k among the samples reaching t .

XGBoost. XGBoost [14] fits an additive ensemble of T regression trees, $\hat{y}_i = \sum_{t=1}^T f_t(\mathbf{z}_i)$, by minimizing the regularized objective $\sum_i \ell(y_i, \hat{y}_i) + \sum_t (\gamma L_t + \lambda/2 \|\mathbf{w}_t\|^2)$, where L_t and $\mathbf{w}_t$ are the number of leaves and the leaf weights of tree f_t .

Voting ensemble. The voting classifier combines a random forest [28] and a decision tree. For B base classifiers producing class-probability estimates $\hat{p}_{b,k}(\mathbf{z})$, the ensemble predicts [29]

$$\hat{y}(\mathbf{z}) = \arg \max_k \frac{1}{B} \sum_{b=1}^B \hat{p}_{b,k}(\mathbf{z}), \quad (12)$$

which reduces variance relative to its members when their errors are not perfectly correlated.

4. Experimental Setup

4.1 Datasets

Five public datasets with different resolutions, color channels, class counts, and application domains were used (Table 3). MNIST [43] contains handwritten digits, Fashion-MNIST [44] contains grayscale apparel images, CIFAR-10 [45] contains natural color images of ten object categories, the Rice Image dataset [46] contains photographs of five rice varieties (Arborio, Basmati, Ipsala, Jasmine, and Karacadag), and Rock-Paper-Scissors (RPS) [47] contains computer-generated images of three hand gestures. This selection spans low-resolution grayscale images, small natural color images with high intra-class variability, and higher-resolution domain-specific images.

Table 3: Characteristics of the Benchmark Datasets

Dataset	Cls.	Images	Size	Ch.	Domain
MNIST [43]	10	70,000	28×28	1	Digits

Dataset	Cls.	Images	Size	Ch.	Domain
Fashion-MNIST [44]	10	70,000	28×28	1	Apparel
CIFAR-10 [45]	10	60,000	32×32	3	Objects
Rice Image [46]	5	75,000	250×250	3	Agriculture
RPS [47]	3	2,892	300×300	3	Gestures

4.2 Preprocessing

Pixel intensities were rescaled to $[0, 1]$ by $x_{\text{norm}} = x/255$ to aid convergence. Images were resized to a common resolution within each dataset, and class labels were integer-encoded and, for the neural models, one-hot encoded. For the neural branch, training images were streamed through the Keras ImageDataGenerator, which applied shear, zoom, and horizontal-flip augmentation to the training partition only. For the feature-based branch, images were read, resized, converted to the required color space, and described by HOG vectors that were stored with their labels as NumPy arrays. For CIFAR-10, the complete official training (50,000) and test (10,000) partitions were used without subsampling.

4.3 Implementation and Hyperparameters

The neural models were implemented in TensorFlow/Keras and the tree-based models in scikit-learn [48] and the XGBoost library [14]. Table 4 lists the GB-CNN settings used for the dense-head study. Both configurations share the same backbone ($P_{\text{backbone}} = 296,992$ parameters) and training schedule; they differ only in the head. The 20-neuron head was grown for $M = 3$ rounds and the 128-neuron head for $M = 2$ rounds, so the wider configuration is also the one with the larger head, as quantified by (7).

Table 4: GB-CNN Configuration for the Dense-Head Study (CIFAR-10)

Component	Setting
Input	32×32×3, rescaled to $[0, 1]$
Conv. blocks	3 blocks of 2×Conv3×3, filters 32 / 64 / 128
Per block	BN, ReLU, MaxPool 2×2, dropout 0.2 / 0.3 / 0.4
Flattened features	$d = 2048$
Dense width W	20 (CW-GB-CNN) 128 (GB-CNN-128)
Boosting rounds M	3 2
Epochs per round	150, early stopping (patience 10)
Residual loss	Mean squared error, Eq. (5)
Step size	Newton–Raphson, Eq. (6)
Shrinkage ν	0.1

4.4 Evaluation Metrics

Performance was measured on the held-out test partitions with accuracy, precision, recall, and F1-score. For class k with TP_k , FP_k , and FN_k true positives, false positives, and false negatives,

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\hat{y}_i = y_i), \quad (13)$$

$$\text{Precision}_k = \frac{TP_k}{TP_k + FP_k}, \quad \text{Recall}_k = \frac{TP_k}{TP_k + FN_k}, \quad (14)$$

$$F1_k = \frac{2 \text{Precision}_k \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k} \quad (15)$$

Class-wise values were aggregated by support-weighted averaging. Under this averaging, weighted recall equals accuracy, which provides an internal consistency check on the reported tables.

4.5 Statistical Analysis Protocol

Following Demšar [22], the $k = 7$ classifiers were ranked by accuracy on each of the $N = 5$ datasets (rank 1 is best; ties receive average ranks), giving mean ranks R_j . The Friedman statistic [49]

$$\chi_F^2 = \frac{12N}{k(k+1)} \left[\sum_{j=1}^k R_j^2 - \frac{k(k+1)^2}{4} \right] \quad (16)$$

and its less conservative Iman–Davenport correction $F_F = (N-1)\chi_F^2 / (N(k-1) - \chi_F^2)$, distributed as F with $(k-1)$ and $(k-1)(N-1)$ degrees of freedom, test the null hypothesis that all classifiers are equivalent. When this hypothesis is rejected, the Nemenyi post-hoc test declares two classifiers different if their mean ranks differ by at least

$$CD = q_\alpha \sqrt{\frac{k(k+1)}{6N}}, \quad (17)$$

with $q_{0.05} = 2.949$ for $k = 7$. Paired boosted/non-boosted comparisons were additionally assessed with one-sided Wilcoxon signed-rank tests over the five datasets.

5. Results

5.1 Multi-Dataset Benchmark (RQ1, RQ4)

Table 5 reports the four metrics for all models and datasets, and Figure 3 compares test accuracies. CW-GB-CNN was the most accurate model on MNIST (99.7%), Rice (99.6%), and RPS (99.1%), and GB-DNN ranked second on each of these three datasets. On the two datasets on which all neural models were least accurate—CIFAR-10 and Fashion-MNIST—the HOG-based voting ensemble obtained the highest accuracy (94.1% and 94.9%), with CW-GB-CNN the best neural model (87.8% and 92.1%). Precision, recall, and F1-score closely track accuracy for every model, differing from it by at most 0.3 percentage points (pp), which indicates that no model achieved its accuracy by favoring majority classes. The F1 heat map in Figure 4 makes the dataset dependence of the classical models visible: XGBoost dropped to 45.5% F1 on CIFAR-10 while reaching 88.5–97.3% on the other datasets, and the tree ensembles fell to about 85% on MNIST, where every neural model reached at least 98.7%.

Table 5: Test Performance of All Models on the Five Datasets (Best per Dataset in Bold)

Dataset	Model	Accuracy	Precision	Recall	F1-score
CIFAR-10	XGBoost	0.458	0.456	0.458	0.455
	DT (HOG)	0.904	0.906	0.904	0.903
	Voting (RF+DT)	0.941	0.943	0.941	0.941
	CNN	0.843	0.846	0.843	0.842

Dataset	Model	Accuracy	Precision	Recall	F1-score
	CW-GB-CNN	0.878	0.880	0.878	0.878
	DNN	0.842	0.844	0.842	0.841
	GB-DNN	0.869	0.871	0.869	0.869
Fashion-MNIST	XGBoost	0.885	0.885	0.885	0.885
	DT (HOG)	0.918	0.919	0.918	0.918
	Voting (RF+DT)	0.949	0.950	0.949	0.949
	CNN	0.903	0.904	0.903	0.903
	CW-GB-CNN	0.921	0.922	0.921	0.921
	DNN	0.896	0.897	0.896	0.896
	GB-DNN	0.914	0.915	0.914	0.914
MNIST	XGBoost	0.972	0.972	0.972	0.972
	DT (HOG)	0.847	0.847	0.847	0.847
	Voting (RF+DT)	0.850	0.850	0.850	0.850
	CNN	0.992	0.992	0.992	0.992
	CW-GB-CNN	0.997	0.997	0.997	0.997
	DNN	0.987	0.987	0.987	0.987
	GB-DNN	0.995	0.995	0.995	0.995
Rice Image	XGBoost	0.974	0.975	0.974	0.973
	DT (HOG)	0.951	0.952	0.951	0.951
	Voting (RF+DT)	0.981	0.982	0.981	0.981
	CNN	0.985	0.986	0.985	0.985
	CW-GB-CNN	0.996	0.996	0.996	0.996
	DNN	0.972	0.973	0.972	0.972
	GB-DNN	0.992	0.992	0.992	0.992
RPS	XGBoost	0.948	0.949	0.948	0.948
	DT (HOG)	0.901	0.902	0.901	0.901
	Voting (RF+DT)	0.948	0.949	0.948	0.948
	CNN	0.968	0.969	0.968	0.968
	CW-GB-CNN	0.991	0.991	0.991	0.991
	DNN	0.944	0.945	0.944	0.944
	GB-DNN	0.984	0.985	0.984	0.984
Mean over datasets	XGBoost	0.847	0.847	0.847	0.847
	DT (HOG)	0.904	0.905	0.904	0.904
	Voting (RF+DT)	0.934	0.935	0.934	0.934
	CNN	0.938	0.939	0.938	0.938
	CW-GB-CNN	0.957	0.957	0.957	0.957
	DNN	0.928	0.929	0.928	0.928
	GB-DNN	0.951	0.952	0.951	0.951

Figure 3: Test Accuracy Comparison

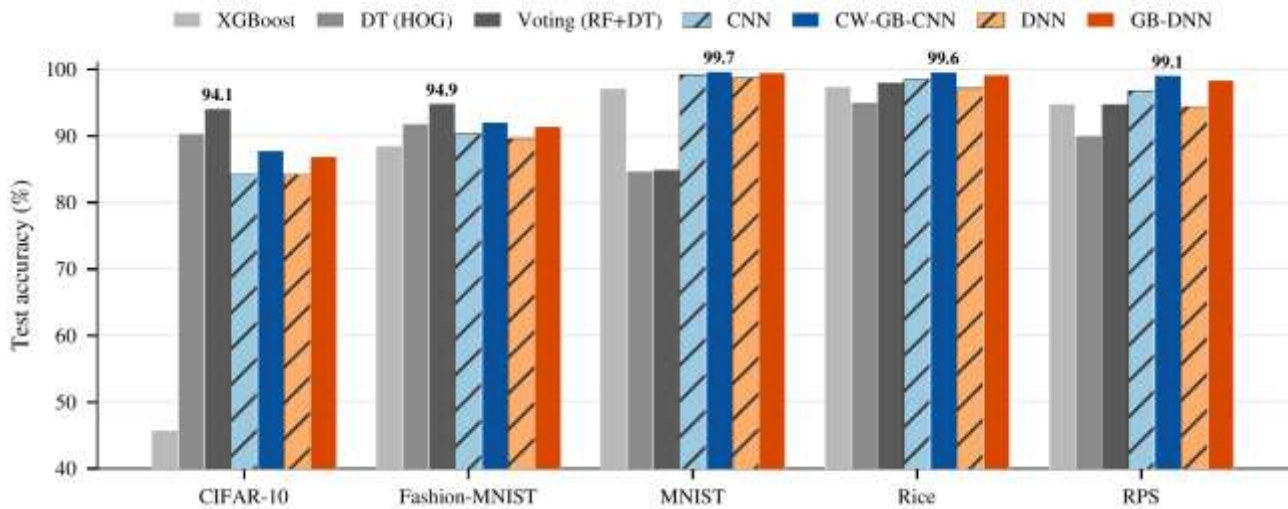
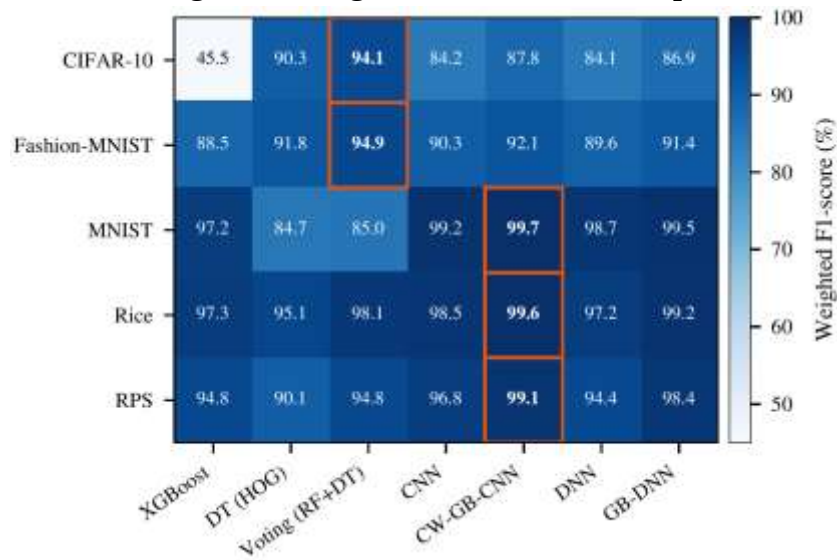


Figure 4: Weighted F1-Score Heatmap

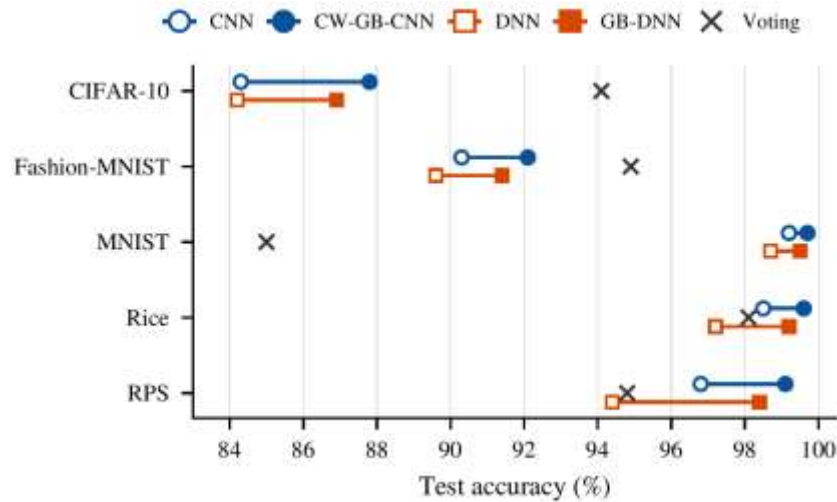


5.2 Effect of Gradient Boosting (RQ1)

Gradient boosting improved its non-boosted counterpart on all ten dataset–architecture pairs (Figure 5). Absolute gains for CW-GB-CNN over CNN ranged from 0.5 pp on MNIST to 3.5 pp on CIFAR-10, and gains for GB-DNN over DNN ranged from 0.8 pp on MNIST to 4.0 pp on RPS. Absolute differences understate the effect on datasets that are already nearly solved, so the relative error reduction (10) is also considered. Boosting removed 62–73% of the residual errors of the baselines on MNIST, Rice, and RPS, and 17–22% on CIFAR-10 and Fashion-MNIST; averaged over the datasets, the error fell by 49.7% for CW-GB-CNN and 47.8% for GB-DNN. The larger relative reductions on datasets with less intra-class variability suggest that the additive head is most effective at correcting a small number of systematic confusions, whereas the harder natural-image errors of CIFAR-10 are limited more by the representational power of the shallow backbone than by the classifier head.

Notably, GB-DNN outperformed the jointly trained CNN on all five datasets and on mean accuracy (95.1% versus 93.8%) even though it lacks convolutional layers, underlining that the training procedure can matter as much as the architecture at this model scale.

Figure 5: Boosting Accuracy Gains



5.3 Statistical Ranking (RQ3)

Table 6 lists the mean ranks. CW-GB-CNN obtained the best mean rank (1.6), followed by GB-DNN (2.8), the voting ensemble (3.3), and CNN (3.8); DT, DNN, and XGBoost occupied the last three positions. The Friedman test rejected the hypothesis of equal performance ($\chi^2_F = 15.66$, $df = 6$, $p = 0.016$), as did the Iman–Davenport correction ($F_F = 4.37$, $df = (6, 24)$, $p = 0.004$). With $N = 5$ and $k = 7$, the Nemenyi critical difference is $CD = 4.03$. On this basis, CW-GB-CNN is significantly better than XGBoost (rank difference 4.1), while its advantage over DNN (4.0) falls just short of the threshold. The paired Wilcoxon tests confirm that CW-GB-CNN outperforms CNN and GB-DNN outperforms DNN ($p = 0.031$ in both cases, the smallest p -value attainable with five paired observations). Hence, the ordering of the boosted networks relative to their direct baselines is statistically supported, whereas finer distinctions among the upper-ranked models would require more datasets to resolve.

Table 6: Mean Accuracy, Standard Deviation Across Datasets, and Mean Friedman Rank

Model	Mean acc. (%)	Std. (pp)	Mean rank
CW-GB-CNN	95.66	5.43	1.6
GB-DNN	95.08	5.66	2.8
Voting (RF+DT)	93.38	4.93	3.3
CNN	93.82	6.38	3.8
DT (HOG)	90.42	3.76	5.2
DNN	92.82	5.94	5.6
XGBoost	84.74	22.06	5.7
Friedman $\chi^2_F = 15.66$ ($p = 0.016$)			
Iman–Davenport $F_F = 4.37$ ($p = 0.004$)			
Nemenyi CD ($\alpha = 0.05$, $k = 7$, $N = 5$) = 4.03			

5.4 Dense-Layer Width and Parameter Efficiency (RQ2)

Table 7 compares the proposed CW-GB-CNN ($W = 20$) with the wide-head GB-CNN-128 on CIFAR-10, and Figure 6 decomposes their parameter budgets. Widening the dense layers from 20 to 128 neurons raised test accuracy from 87.77% to 87.86%, a gain of 0.09 pp that corresponds to a relative error reduction of only 0.74%. Over the same change, the dense head grew from 42,030 to 280,074 parameters (6.66 $\times$), the total model from 339,022 to 577,066 parameters (1.70 $\times$), and the head's share of the network from 12.4% to 48.5%. The marginal accuracy yield (9) is therefore 0.038 pp per 10^5 additional parameters, and accuracy per 10^5 parameters falls from 25.9 to 15.2—a 41% loss of parameter efficiency.

The training accuracies of 99.55% and 99.47% show that both heads fit the training set almost perfectly, and their train–test gaps are nearly identical (11.78 pp and 11.61 pp). The additional capacity of the wider head is thus not converted into better generalization: the two configurations are limited by the same factor, namely the 2048-dimensional representation delivered by the backbone. This interpretation is consistent with the layer-wise decomposition in Figure 6(b), where the first dense layer D_1 alone accounts for 97.5% (CW-GB-CNN) and 93.6% (GB-CNN-128) of the head, so the cost of widening is incurred almost entirely in the projection of backbone features rather than in the boosting layers.

Evaluating (7) over the range of widths and boosting rounds shows that, for narrow heads, the head size is almost independent of M ; at $W = 20$, adding one further round costs only 420 parameters. At $W = 128$, by contrast, each round costs 16,512 parameters, and a third round would bring the head to 296,586 parameters. Narrow heads therefore make additional boosting rounds nearly free, whereas wide heads make both width and depth expensive.

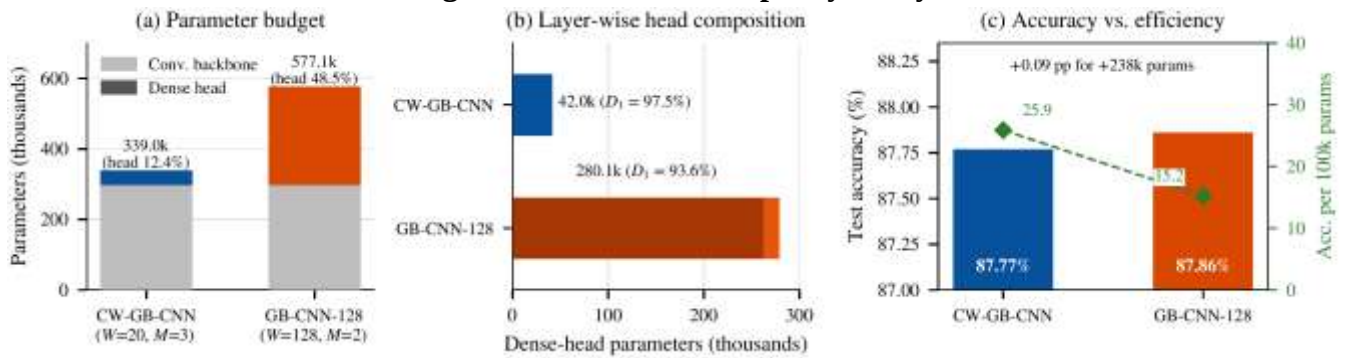
Significance of the additional parameters. The 1.70-fold increase in total parameters (6.66-fold in the head) from CW-GB-CNN to GB-CNN-128 has no practical significance for performance: the 0.09 pp difference corresponds to nine of the 10,000 CIFAR-10 test images, lies within typical seed-to-seed variation (Section 6), and is not accompanied by a meaningfully smaller generalization gap. The wider configuration is therefore not proposed; it is reported only as the conventional wide-head reference against which the capacity-aware choice is tested. CW-GB-CNN satisfies the criterion of similar performance at reduced parameters with respect to GB-CNN-128, achieving practically the same accuracy with 238,044 fewer parameters, and the criterion of significant improvement with respect to its jointly trained counterpart, which it outperforms on all five datasets (Wilcoxon $p = 0.031$) with an average error reduction of 49.7% (Section 5.2).

Table 7: Proposed CW-GB-CNN ($W = 20$) Versus Wide-Head GB-CNN-128 on CIFAR-10: Accuracy and Capacity

Quantity	CW-GB-CNN	GB-CNN-128	Change
Dense width W / rounds M	20 / 3	128 / 2	—
Dense-head parameters	42,030	280,074	$\times 6.66$
of which D_1	40,980 (97.5%)	262,272 (93.6%)	$\times 6.40$
Backbone parameters	296,992	296,992	—
Total parameters	339,022	577,066	$\times 1.70$
Head share η_{head} (%)	12.4	48.5	+36.1 pp
Training accuracy (%)	99.55	99.47	−0.08 pp
Test accuracy (%)	87.77	87.86	+0.09 pp

Quantity	CW-GB-CNN	GB-CNN-128	Change
Train–test gap (pp)	11.78	11.61	–0.17 pp
APK (%/10 ⁵ params)	25.9	15.2	–41.2%
MAY _{20→128} (pp/10 ⁵ params)	0.038		—

Figure 6: Dense-Head Capacity Study



5.5 Boosted Networks Versus Feature-Based Ensembles (RQ4)

The HOG-based models exhibit markedly stronger dataset dependence than the neural models (Table 6): the standard deviation of accuracy across datasets is 22.1 pp for XGBoost, compared with 5.4–6.4 pp for the four neural models. The voting ensemble exceeded CW-GB-CNN by 6.3 pp on CIFAR-10 and 2.8 pp on Fashion-MNIST but trailed it by 14.7 pp on MNIST, 1.5 pp on Rice, and 4.3 pp on RPS. Combining a random forest with a decision tree was consistently beneficial within the classical branch, improving on the single tree on all five datasets and raising mean accuracy from 90.4% to 93.4%. Nevertheless, the voting ensemble's reversal between MNIST and CIFAR-10—two datasets on which neural models behave in the opposite way—indicates that the performance of HOG-based ensembles depends strongly on how well the fixed descriptor matches the visual structure of each dataset, whereas the learned features of the boosted networks transfer more uniformly. In terms of expected performance on an unseen image domain, CW-GB-CNN is therefore the most reliable choice, as reflected in its best mean accuracy and mean rank.

6. Discussion

6.1 Principal Findings

The results support three conclusions. First, gradient boosting is a consistently beneficial training strategy for both convolutional and fully connected networks: it improved every baseline, roughly halved the error on average, and allowed a convolution-free GB-DNN to surpass a jointly trained CNN on every dataset. Second, the accuracy of GB-CNN is governed mainly by the backbone representation rather than by head width. The 128-neuron head increased the parameter count by more than two hundred thousand for less than one-tenth of a percentage point, whereas the 20-neuron head of CW-GB-CNN delivered practically the same accuracy with a head that is only 12.4% of the model. Third, when rank-based multi-dataset statistics are applied, the advantage of boosted networks over their direct counterparts is significant, while the placement of the strongest models relative to one another remains within the Nemenyi critical difference.

6.2 Practical Design Guidelines

Equation (7) and the width study translate into concrete recommendations for practitioners. (1) Start with a narrow additive head ($W \approx 20$); because the $(d + 1)W$ projection dominates, width is the main cost driver, and narrow heads keep boosting rounds inexpensive. (2) Invest the parameter budget in the backbone or in additional boosting rounds rather than in wider dense layers, since the two heads studied here share the same generalization gap. (3) Evaluate candidates with efficiency indicators such as MAY and APK alongside accuracy, particularly for edge deployment where memory and latency are constrained [40, 41]. (4) Where the backbone output is large, reducing d —for example by global average pooling [20]—would shrink D_1 directly and compound the savings of a narrow head.

6.3 Threats to Validity and Limitations

Internal validity. The two heads in the width study differ in both width and number of boosting rounds ($M = 3$ versus $M = 2$). Equation (7) shows that the effect of M on head size is minor, but its effect on accuracy was not isolated; a full factorial design over W and M would separate the two. **Statistical conclusion validity.** Results derive from a single training run per configuration on fixed hold-out splits, so run-to-run variance is not reported, and the 0.09 pp width difference corresponds to nine test images out of 10,000, which lies within typical seed-to-seed variation for CIFAR-10. With five datasets, the Nemenyi test has limited power. **Construct validity.** The HOG-based ensembles and the neural models use different input representations, so their comparison reflects complete pipelines rather than learners alone; the unusually large dataset dependence of these ensembles, especially their strong CIFAR-10 and weak MNIST results, warrants confirmation under repeated and cross-validated partitions. **External validity.** The backbone is intentionally compact and the datasets are of small to moderate scale; whether the width findings hold for deeper backbones or large-scale datasets remains to be verified.

7. Conclusion and Future Work

This article presented CW-GB-CNN, a capacity-aware gradient-boosted convolutional neural network, together with a study of gradient-boosted convolutional and deep neural networks for image classification, motivated by gaps in cross-paradigm evidence, capacity analysis, and statistical validation. Across five heterogeneous datasets, gradient boosting improved all ten network baselines, reduced the test error by 49.7% (CW-GB-CNN) and 47.8% (GB-DNN) on average, and made CW-GB-CNN the best-ranked of seven classifiers, with its advantage over jointly trained counterparts confirmed by Wilcoxon signed-rank tests. A closed-form model of the additive dense head showed that head size is dominated by the projection from backbone features, and the CIFAR-10 width study showed that enlarging each dense layer from 20 to 128 neurons multiplied head parameters by 6.66 for a 0.09-percentage-point accuracy gain, reducing accuracy per parameter by 41%; consequently, CW-GB-CNN attains similar accuracy with 1.70 times fewer total parameters than GB-CNN-128. Compact heads should therefore be the default configuration for gradient-boosted image classifiers. A HOG-based voting extension provided a feature-based reference against which the neural models were compared. Future work will (1) perform a factorial study over width, boosting rounds, and shrinkage with repeated runs and confidence intervals; (2) couple narrow boosted heads with global-average-pooled and pretrained backbones; (3) develop adaptive schemes that select the width and stopping round of each iteration from validation residuals; (4) extend the benchmark to larger datasets and corrupted-image robustness benchmarks such as MNIST-C [50]; and (5) profile latency and memory of the trained models on edge hardware.

8. Data and Code Availability

All datasets used in this study are publicly available from their original sources [43, 44, 45, 46, 47]. The analysis scripts used to compute the statistics and generate the figures are provided with this manuscript.

9. Acknowledgment

The authors thank the providers of the public datasets used in this study.

10. References

1. A. Krizhevsky, I. Sutskever, G.E. Hinton, “ImageNet classification with deep convolutional neural networks”, Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2012, 25, 1097–1105.
2. D. Han, Q. Liu, W. Fan, “A new image classification method using CNN transfer learning and web data augmentation”, Expert Systems with Applications, 2018, 95, 43–56.
3. S. Ren, K. He, R. Girshick, J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks”, Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2015, 28, 91–99.
4. T. Kim, S.C. Suh, H. Kim, J. Kim, J. Kim, “An encoding technique for CNN-based network anomaly detection”, Proceedings of the IEEE International Conference on Big Data, 2018, 2960–2965.
5. K. Kamnitsas et al., “Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation”, Medical Image Analysis, 2017, 36, 61–78.
6. K. Simonyan, A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2015.
7. K. He, X. Zhang, S. Ren, J. Sun, “Deep residual learning for image recognition”, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, 770–778.
8. M.D. Zeiler, R. Fergus, “Visualizing and understanding convolutional networks”, Proceedings of the 13th European Conference on Computer Vision (ECCV), 2014, 818–833.
9. S. Ioffe, C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, Proceedings of the 32nd International Conference on Machine Learning (ICML), 2015, 448–456.
10. X. Glorot, Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks”, Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS), 2010, 249–256.
11. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting”, Journal of Machine Learning Research, 2014, 15 (1), 1929–1958.
12. J.H. Friedman, “Greedy function approximation: A gradient boosting machine”, Annals of Statistics, 2001, 29 (5), 1189–1232.
13. L. Mason, J. Baxter, P. Bartlett, M. Frean, “Boosting algorithms as gradient descent”, Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 1999, 12, 512–518.
14. T. Chen, C. Guestrin, “XGBoost: A scalable tree boosting system”, Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, 785–794.

15. G. Ke et al., “LightGBM: A highly efficient gradient boosting decision tree”, Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2017, 30, 3146–3154.
16. R. Shwartz-Ziv, A. Armon, “Tabular data: Deep learning is not all you need”, Information Fusion, 2022, 81, 84–90.
17. C. Bentéjac, A. Csörgő, G. Martínez-Muñoz, “A comparative analysis of gradient boosting algorithms”, Artificial Intelligence Review, 2021, 54, 1937–1967.
18. S. Emami, G. Martínez-Muñoz, “Sequential training of neural networks with gradient boosting”, IEEE Access, 2023, 11, 42738–42750.
19. S. Emami, G. Martínez-Muñoz, “Multioutput regression neural network training via gradient boosting”, Proceedings of the 30th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN), 2022, 145–150.
20. M. Lin, Q. Chen, S. Yan, “Network in network”, Proceedings of the 2nd International Conference on Learning Representations (ICLR), 2014.
21. S. Han, H. Mao, W.J. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding”, Proceedings of the 4th International Conference on Learning Representations (ICLR), 2016.
22. J. Demšar, “Statistical comparisons of classifiers over multiple data sets”, Journal of Machine Learning Research, 2006, 7, 1–30.
23. N. Dalal, B. Triggs, “Histograms of oriented gradients for human detection”, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2005, 1, 886–893.
24. D.P. Kingma, J. Ba, “Adam: A method for stochastic optimization”, Proceedings of the 3rd International Conference on Learning Representations (ICLR), 2015.
25. R.E. Schapire, “The strength of weak learnability”, Machine Learning, 1990, 5 (2), 197–227.
26. Y. Freund, R.E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting”, Journal of Computer and System Sciences, 1997, 55 (1), 119–139.
27. L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, “Classification and Regression Trees”, Wadsworth, Belmont, CA, USA, 1984.
28. L. Breiman, “Random forests”, Machine Learning, 2001, 45 (1), 5–32.
29. L.I. Kuncheva, “Combining Pattern Classifiers: Methods and Algorithms”, 2nd Edition, Wiley, Hoboken, NJ, USA, 2014.
30. S.Ö. Arık, T. Pfister, “TabNet: Attentive interpretable tabular learning”, Proceedings of the AAAI Conference on Artificial Intelligence, 2021, 35 (8), 6679–6687.
31. A. Abutbul, G. Elidan, L. Katzir, R. El-Yaniv, “DNF-Net: A neural architecture for tabular data”, arXiv preprint arXiv:2006.06465, 2020.
32. Y. Yang, I.G. Morillo, T.M. Hospedales, “Deep neural decision trees”, arXiv preprint arXiv:1806.06988, 2018.
33. Y. Bengio, N. Le Roux, P. Vincent, O. Delalleau, P. Marcotte, “Convex neural networks”, Proceedings of Advances in Neural Information Processing Systems (NeurIPS), 2005, 18, 123–130.
34. M. Moghimi, S.J. Belongie, M.J. Saberian, J. Yang, N. Vasconcelos, L.-J. Li, “Boosted convolutional neural networks”, Proceedings of the British Machine Vision Conference (BMVC), 2016.

35. F. Zhang, B. Du, L. Zhang, “Scene classification via a gradient boosting random convolutional network framework”, IEEE Transactions on Geoscience and Remote Sensing, March 2016, 54 (3), 1793–1802.
36. F. Huang, J. Ash, J. Langford, R. Schapire, “Learning deep ResNet blocks sequentially using boosting theory”, Proceedings of the 35th International Conference on Machine Learning (ICML), 2018, 2058–2067.
37. A. Nitanda, T. Suzuki, “Functional gradient boosting based on residual network perception”, Proceedings of the 35th International Conference on Machine Learning (ICML), 2018, 3819–3828.
38. C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, S. Yang, “AdaNet: Adaptive structural learning of artificial neural networks”, Proceedings of the 34th International Conference on Machine Learning (ICML), 2017, 874–883.
39. S. Badirli, X. Liu, Z. Xing, A. Bhowmik, K. Doan, S.S. Keerthi, “Gradient boosting neural networks: GrowNet”, arXiv preprint arXiv:2002.07971, 2020.
40. A.G. Howard et al., “MobileNets: Efficient convolutional neural networks for mobile vision applications”, arXiv preprint arXiv:1704.04861, 2017.
41. M. Tan, Q.V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks”, Proceedings of the 36th International Conference on Machine Learning (ICML), 2019, 6105–6114.
42. C. Zhang, S. Bengio, M. Hardt, B. Recht, O. Vinyals, “Understanding deep learning requires rethinking generalization”, Proceedings of the 5th International Conference on Learning Representations (ICLR), 2017.
43. Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, “Gradient-based learning applied to document recognition”, Proceedings of the IEEE, November 1998, 86 (11), 2278–2324.
44. H. Xiao, K. Rasul, R. Vollgraf, “Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms”, arXiv preprint arXiv:1708.07747, 2017.
45. A. Krizhevsky, “Learning multiple layers of features from tiny images”, Technical Report, University of Toronto, Toronto, ON, Canada, 2009.
46. M. Koklu, I. Cinar, Y.S. Taspinar, “Classification of rice varieties with deep learning methods”, Computers and Electronics in Agriculture, 2021, 187, Article 106285.
47. L. Moroney, “Rock, paper, scissors dataset”, February 2019. https://www.tensorflow.org/datasets/catalog/rock_paper_scissors
48. F. Pedregosa et al., “Scikit-learn: Machine learning in Python”, Journal of Machine Learning Research, 2011, 12, 2825–2830.
49. M. Friedman, “The use of ranks to avoid the assumption of normality implicit in the analysis of variance”, Journal of the American Statistical Association, 1937, 32 (200), 675–701.
50. N. Mu, J. Gilmer, “MNIST-C: A robustness benchmark for computer vision”, arXiv preprint arXiv:1906.02337, 2019.

Licensed under [Creative Commons Attribution-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-sa/4.0/)