DEBoost: Distillation-Enhanced Boosting for Cost-Efficient Continual Learning

Cailong Hua*, Hankyu Jang*, Di Zhang*, Kaiqiao Li†, Weicheng Zhu*, Xue Hao*,
Bowen Zhi*, Scott Nickleach*, Zhanlong Qiu*, Olcay Boz*, Wenzhuo Zhou*‡
*Amazon, Seattle, WA, USA
†Expedia Group, Seattle, WA, USA
‡University of California Irvine, Irvine, CA, USA

Abstract—Production machine-learning systems retrain periodically to track evolving data, but as accumulated datasets grow to millions of instances, retraining from scratch on the full accumulated history becomes prohibitively expensive, inflating compute, data-pipeline, and storage cost at every refresh. Gradient-boosted decision trees such as LightGBM are widely used for large-scale tabular prediction, yet they are inherently offline: each refresh must either reprocess the entire history or train on recent data alone and discard previously learned patterns. We present DEBoost (Distillation-Enhanced Boosting), which integrates knowledge distillation directly into the gradient-boosting loop to break this trade-off: it trains a fresh student on only the recent data window while transferring historical knowledge from a previously deployed teacher through the teacher's predictions on that window, requiring no historical raw data. We further provide a Bayesian interpretation in which knowledge distillation corresponds to inference under a prior derived from the teacher model. Across nine public benchmarks, including static datasets and time-ordered data streams with naturally evolving feature distributions, DEBoost consistently improves over equal-budget training on only the latest data window, as measured by Area Under the ROC Curve (ROC-AUC) for classification and R-squared (R^2) for regression, while requiring 12–58% of the training time needed to retrain on all data accumulated through the current window. On the time-ordered streams, DEBoost's performance relative to retraining on all accumulated data improves with the measured distribution shift; on the highest-shift stream, it achieves a 0.0324 ROC-AUC gain. Because it consumes only teacher predictions, DEBoost does not require access to teacher internals or shared model parameters.

Index Terms—knowledge distillation, data efficiency, continual learning, gradient boosting

I. Introduction

Modern production systems continually accumulate data and must periodically refresh their models to stay accurate as distributions drift, and at scale this refresh is dominated by a single cost: retraining. The Gradient Boosting Machine (GBM) [1], and its state-of-the-art implementation LightGBM [2], is the workhorse of large-scale tabular prediction—efficient, interpretable, and highly accurate, powering top-ranking solutions across domains including Kaggle competitions. Yet LightGBM is inherently an offline learner: unlike online algorithms [3], [4], it must be retrained from scratch to incorporate new data, discarding previously learned information. This creates a sharp dilemma as accumulated datasets grow to millions of instances. Retraining from scratch on the full accumulated history becomes prohibitively expensive—compute, data-pipeline, and storage cost all scale with total data volume, and every refresh reprocesses the entire history. Training only on recent data avoids this cost but discards knowledge of previously encountered patterns, leading to performance degradation over time. An ideal solution would refresh the model on recent data alone while still retaining the knowledge accumulated in prior models.

Continual learning (CL) research aims to train models to quickly adapt to new patterns while retaining past knowledge. CL has developed along several approaches, including optimization-based methods like GEM [5] that use old gradients to maintain important gradient subspaces, and knowledge distillation (KD) techniques that employ both function and weight regularization. Function regularization preserves the model's previous behavior through various methods such as maintaining predictions on new data (e.g., LwF [6]), feature representations (e.g., EBLL [7]), or generative outputs (e.g., MeRGANs [8]). Weight regularization, exemplified by Elastic Weight Consolidation (EWC) [9], takes a different approach by controlling parameter changes based on their importance to previous tasks, using Fisher Information Matrix to determine parameter significance and apply selective constraints. However, most of these continual learning frameworks are designed for non-tree-based models. An existing tree-based continual-learning method, iGBDT [10], retains and modifies the preceding ensemble, requiring access to its internal tree structure.

In contrast, we propose DEBoost, a continual learning framework for gradient-boosted decision trees that uses knowledge distillation to transfer historical knowledge from a previously deployed teacher into a freshly trained student model. Because it operates through the standard gradient–Hessian interface of the boosting objective, DEBoost applies to any tree-based boosting method, and we instantiate and evaluate it with LightGBM. The key operational distinction is that DEBoost uses the teacher as a scorer: the student needs teacher predictions on the current data window but requires neither historical raw data nor access to teacher internals or shared model

parameters.

A. Our contributions

This paper aims to develop a continual learning version of LightGBM. The major contributions are as follows:

- Continual Learning Framework for Gradient Boosting: We propose a practical framework that integrates gradient-boosted trees (instantiated with LightGBM) with knowledge distillation, enabling the model to update on newly arriving data while preserving knowledge from prior models—overcoming the static-data limitations of traditional offline boosting.
- Empirical Validation: On nine public benchmarks spanning both static and real temporal-drift settings, DEBoost improves over equal-budget recent-window retraining on every dataset while using only 20–25% of the data (12–58% of the training time required by Full-Retrain). On the temporal datasets, its performance relative to Full-Retrain improves with measured drift, and it outperforms Full-Retrain by 0.0324 ROC-AUC on the highest-drift stream.

The structure of this paper is as follows. In §II, we review existing works in continual learning, and their limited application to tree-based models. In §III, we formally state the problem. In §IV, we propose a continual learning version of LightGBM that integrates the offline algorithm with knowledge distillation. This approach enables the model to both leverage newly arriving data and retain valuable information from historical data. In §V, we present the Bayesian interpretation of the proposed method. In §VI, we demonstrate the advantages of our approach over baseline methods through empirical analysis on public static and temporal benchmarks. §VII discusses potential directions for future research. Finally, §VIII summarizes our conclusions.

II. Related work

A. Continual learning

Various threads of research on continual learning (CL) have been conducted, with several promising approaches emerging. One such approach is based on optimization techniques. Exemplar methods in this category include Gradient Episodic Memory (GEM) [5], A-GEM [11], and OGD [12], which leverage old gradients to maintain important gradient subspaces. These methods constrain or project parameter updates to be orthogonal to the gradient directions or input subspaces of old tasks, thereby preventing interference with previously acquired knowledge.

KD represents a significant approach in Continual Learning, encompassing both function and weight regularization techniques. Function regularization focuses on maintaining the previous model's behavior and outputs. This can be achieved in several ways: by preserving the model's predictions on new data (as seen in LwF [6] and iCaRL [13]), by maintaining feature representations (demonstrated in EBLL [7] and PODNet [14]), or by conserving generative outputs (as implemented in MeRGANs [8] and DRI [15]). Through distillation loss, the updated model learns to replicate the responses of its predecessor, effectively preserving knowledge from earlier tasks.

Weight regularization represents another major CL approach by controlling changes to model parameters based on their significance to previous tasks. Elastic Weight Consolidation (EWC) [9], for example, introduces a penalty term that preserves important neural-network weights from the previous model. The importance of each parameter is estimated using the Fisher Information Matrix, which measures sensitivity to parameter changes. Such methods assume a shared differentiable parameterization across tasks, whereas our setting targets boosted trees refreshed through teacher-student response distillation [16].

Note that the majority of continual learning frameworks are designed for non-tree-based models. Our work focuses on enabling continual learning for tree-based models, which we discuss next.

B. Continual learning for tree-based models

LightGBM [2] is a high-performance gradient boosting framework, but like other GBDTs it does not natively support online learning or incremental updates as new data arrives. Several works adapt tree-based models for streaming data: [17] studies online bagging and boosting that update without retraining from scratch, and [18] develops an online LightGBM for wind-turbine fault detection. iGBDT [10] retains the prior ensemble and appends residual-fitting trees on each new window. This requires access to prior tree parameters and causes the ensemble to grow across updates, unlike DEBoost, which trains a fresh student using only teacher predictions. These efforts motivate a robust continual-learning approach for LightGBM that overcomes its offline limitations.

Knowledge distillation (KD), popularized by [19], trains a smaller student model to mimic a larger teacher's softened output probabilities. Although developed for neural networks and widely used in deep learning and NLP, we show the same principle transfers effectively to gradient boosting models such as LightGBM, yielding an efficient continual-learning algorithm that updates incrementally while maintaining high predictive performance.

III. Problem Statement

We study a production model-refresh setting. At refresh cycle s, the learner observes a current training window $\mathcal{D}_s = \{(x_i, t_i)\}_{i=1}^{n_s}$ and can query a previously deployed teacher for soft predictions $\hat{y}_i$ on these current examples; as noted, no historical raw data or teacher internals are required. The goal is to train a student model that performs well on the current serving distribution while retaining useful information encoded in the teacher's predictions. This reflects production refresh practice, where model

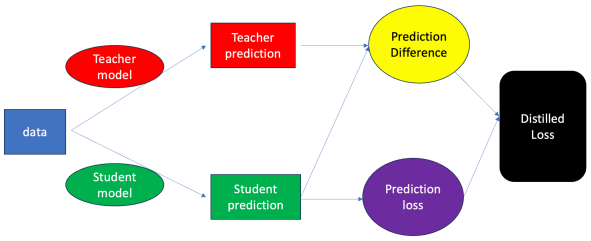


Fig. 1. Response KD

quality is evaluated on the current or near-future window rather than by revisiting old windows.

As illustrated in Figure 1, the KD loss function comprises two components: the prediction loss of the student model on new data (L_{new}) and the prediction difference between teacher and student models (L_{KL}), formulated as:

$$L = (1-\alpha)\, L_{new} + \alpha L_{KL} \tag{1}$$

where α controls the trade-off between fitting the current labels and matching the teacher scores. When $\alpha = 0$, the approach reduces to ordinary supervised training on the current window; when $\alpha = 1$, the student is trained to mirror the teacher scores on that window.

A. Regression

For regression problems, we employ square loss for the prediction loss L_{new} and Gaussian KL divergence for the prediction difference. Thus, the loss function is given by:

$$L = \frac{1-\alpha}{2}(y-t)^2 + \frac{\alpha}{2}(y-\hat{y})^2 \tag{2}$$

where t represents the true label of the continuous response (any real number), and $\hat{y}$ denotes the predicted value from the teacher model. For continuous value prediction, no temperature parameter is required.

B. Binary classification

For binary classification, we employ logistic loss for the prediction loss L_{new} and KL divergence for the prediction difference. Thus, the loss function is given by:

$$\begin{aligned} L = & -(1-\alpha)\left[t\ln y + (1-t)\ln(1-y)\right] \\ & + \alpha T^2\left[\hat{y}_T \ln\frac{\hat{y}_T}{y_T} + (1-\hat{y}_T)\ln\frac{1-\hat{y}_T}{1-y_T}\right] \\ \propto & -(1-\alpha)\left[t\ln y + (1-t)\ln(1-y)\right] \\ & - \alpha T^2\left[\hat{y}_T \ln y_T + (1-\hat{y}_T)\ln(1-y_T)\right] \end{aligned} \tag{3}$$

where t represents the true label of response (0 or 1), $y = 1/(1+\exp(-z))$ is the predictive value from the student model to be trained, and z represents the raw predictive value. The temperature parameter T controls predictive-distribution softness and entropy. The term $y_T = 1/(1+\exp(-z/T))$ represents the predictive value from the student model with temperature T, while $\hat{y}_T = 1/(1+\exp(-\hat{z}/T))$ denotes the predictive value from the teacher model with temperature T. Consequently, we have $y_T = \text{logistic}\left(\frac{1}{T}\text{logit}(y)\right)$ and $y = \text{logistic}(T\text{logit}(y_T))$.

When the temperature $T = 1$, we have $y_T = y$ and $\hat{y}_T = \hat{y}$. The loss function is simplified as

$$\begin{aligned} L \propto & -(1-\alpha)\left[t\ln y + (1-t)\ln(1-y)\right] \\ & -\alpha\left[\hat{y}\ln y + (1-\hat{y})\ln(1-y)\right]. \end{aligned}$$

a) Multi-class Classification (K classes): The loss function is given by

$$\begin{aligned} L = & -(1-\alpha)\textstyle\sum_{k=1}^{K}(t_k \ln y_k) \\ & +\alpha T^2 \textstyle\sum_{k=1}^{K}\left(\hat{y}_{k,T}\ln\frac{\hat{y}_{k,T}}{y_{k,T}}\right) \\ \propto & -(1-\alpha)\textstyle\sum_{k=1}^{K}(t_k \ln y_k) \\ & -\alpha T^2 \textstyle\sum_{k=1}^{K}(\hat{y}_{k,T}\ln y_{k,T}), \end{aligned} \tag{4}$$

where t_k is the true label for the kth class ($\sum_{k=1}^{K} t_k = 1$), $y_k = \exp(z_k)/\Sigma_{j=1}^{K}\exp(z_j)$ is the student model's predictive probability ($\Sigma_{k=1}^{K} y_k = 1$), z_k is the raw predictive value for the kth class, T is the temperature parameter controlling predictive-distribution softness and entropy, $y_{k,T} = \exp(z_k/T)/\Sigma_{j=1}^{K}\exp(z_j/T)$ is the student model's temperature-scaled prediction, and $\hat{y}_{k,T} = \exp(\hat{z}_k/T)/\Sigma_{j=1}^{K}\exp(\hat{z}_j/T)$ is the teacher model's temperature-scaled prediction.

The multi-class formulation generalizes the binary case. When $K = 2$, the loss function simplifies due to $t_2 = 1-t_1$, $y_2 = 1-y_1$, $y_{2,T} = 1-y_{1,T}$, and $\hat{y}_{2,T} = 1-\hat{y}_{1,T}$.

$$\begin{aligned} L \propto & -(1-\alpha)(t_1\ln y_1 + t_2 \ln y_2) \\ & -\alpha T^2(\hat{y}_{1,T}\ln y_{1,T} + \hat{y}_{2,T}\ln y_{2,T}) \\ = & -(1-\alpha)\left[t_1\ln y_1 + (1-t_1)\ln(1-y_1)\right] \\ & -\alpha T^2\left[\hat{y}_{1,T}\ln y_{1,T} + (1-\hat{y}_{1,T})\ln(1-y_{1,T})\right]. \end{aligned} \tag{5}$$

When the temperature $T = 1$, we have $y_k = y_{k,T}$ and $\hat{y}_k = \hat{y}_{k,T}$.

$$L \propto -(1-\alpha)\textstyle\sum_{k=1}^{K}(t_k \ln y_k) - \alpha\sum_{k=1}^{K}(\hat{y}_k \ln y_k).$$

In large tabular data prediction problems, boosting-based methods often demonstrate superior performance compared to neural network-based approaches. Therefore, we propose integrating this response-based KD method with boosting-based techniques to enhance tabular data prediction. This integration enables the construction of a continual learning framework for tabular data prediction tasks within a boosting-based approach.

IV. Methodology

In this paper, we propose distillation-enhanced boosting (DEBoost), which integrates response-based knowledge distillation into the gradient-boosting objective. The teacher may be any previously deployed scoring model, while the student is a freshly trained gradient-boosted model—instantiated with LightGBM—optimized on the current data window with an additional distillation term.

A. Knowledge distilled boosting

We utilize Equation 2 and Equation 3 as the loss functions and derive the corresponding gradient and diagonal entries of the Hessian matrix.

1) Regression: The gradient and diagonal entries of the Hessian matrix for the regression model are:

$$L' = (1-\alpha)(y-t) + \alpha(y-\hat{y}) \quad \text{and} \quad L'' = 1. \tag{6}$$

2) Binary classification: The gradient and diagonal entries of the Hessian matrix for the binary classification model are:

$$\begin{aligned} L' &= (1-\alpha)(y-t) + \alpha T(y_T - \hat{y}_T) \\ L'' &= (1-\alpha)\, y(1-y) + \alpha y_T(1-y_T). \end{aligned} \tag{7}$$

When the temperature $T = 1$, we have $y_T = y$ and $\hat{y}_T = \hat{y}$. The gradient and diagonal entries of the Hessian matrix simplify to

$$L^{'} = (1-\alpha)(y-t) + \alpha(y-\hat{y})$$

$$L^{''} = y(1-y).$$

a) Multi-class Classification (K classes): For a multi-class classification problem with K classes, we derive the gradient and Hessian expressions for each class k. The gradient and diagonal entries of the Hessian matrix are:

$$L'_k = (1-\alpha)(y_k - t_k) + \alpha T(y_{k,T} - \hat{y}_{k,T})$$

$$L''_k = (1-\alpha)\, y_k(1-y_k) + \alpha y_{k,T}(1-y_{k,T}).$$

These expressions simplify when the temperature parameter $T = 1$, as this makes $y_{k,T} = y_k$ and $\hat{y}_{k,T} = \hat{y}_k$. Under these conditions, the gradient and diagonal entries of the Hessian matrix reduce to the simpler form

$$L'_k = (1-\alpha)(y_k - t_k) + \alpha(y_k - \hat{y}_k)$$

$$L''_k = y_k(1-y_k).$$

We provide the detailed derivation in Appendix A. Let z denote the predictor value from training data. The response value y is given by $y = 1/(1+\exp(-z))$ and $y_T = 1/(1+\exp(-z/T))$, obtained via a standard logistic function. We then fit a LightGBM tree on each iteration for L'/L'' on z [20].

By aggregating these LightGBM trees, we obtain our DEBoost. We present the complete algorithm in Algorithm 1.

Algorithm 1 DEBoost Algorithm

Input: Given training data, pretrained teacher model
Use the teacher model to generate soft labels $\hat{y}$ on the training data
for $t = 1$ to N do
    Compute gradient $g_t = \nabla L(\hat{y}, y)$ and Hessian diagonal $h_t = \nabla^2 L(\hat{y}, y)$
    Fit LightGBM tree $f_t(x)$ to the pseudo response $-L'/L''$
    Update $\tilde{z} \leftarrow \tilde{z} + \eta_t f_t(x)$
    if validation performance does not improve then
        break
    end if
end for
y *or* $y_k = \begin{cases} \tilde{z} & \text{regression} \\ 1/(1+\exp(-\tilde{z})) & \text{binary} \\ \exp(\tilde{z}_k) / \sum_{i=1}^{K} \exp(\tilde{z}_i) & \text{multiclass} \end{cases}$

B. Continual learning

We apply the proposed DEBoost model to capture historical information. First, we build a teacher model following classical lightGBM model building procedures and save it to serve as the teacher model in DEBoost. When updating the model, we collect the new training data, and then use both the teacher model and the new training data to build a student model using DEBoost. This approach enables student model to learn from both new training data and inherit knowledge from the teacher model, effectively incorporating information from teacher model's training data. The student model is built at significantly reduced computational cost, since it requires only the new data window. By applying this strategy for model refreshing over time, we establish a continual learning procedure that retains information while eliminating redundant computations. The detailed implementation is presented in Algorithm 2.

Algorithm 2 Continual Learning with DEBoost

Input: training data, latest saved model

1) Train a LightGBM model from scratch
2) Iteratively model build
    a) Collect data later than latest saved model's data
    b) Build DEBoost model with new collected data and latest saved model
    c) Save and use the new trained DEBoost model
    d) End until the model no longer needed

C. Hyper-parameter tuning

The DEBoost framework requires tuning of two distinct sets of hyperparameters: the distillation parameters (α and T, where T applies only to classification tasks) and the

LightGBM tree construction parameters. In our experiments, we use two complementary approaches to optimize the distillation parameters. The first is a grid search with α between 0 and 1, while T is searched on a logarithmic scale (geometric progression) with values greater than or equal to 1. The second is the tree-structured Parzen estimator (TPE) [21], implemented with Optuna [22]; TPE estimates separate densities from better- and lower-performing trials and selects the next (α, T) combination by favoring values that are more likely under the better-performing density. The optimal value of α is dataset-dependent and is best selected by validation, reflecting how much the current window benefits from the teacher. For the LightGBM tree construction parameters, we recommend following standard optimization approaches used in classical LightGBM implementations, such as Bayesian optimization.

V. Bayesian Interpretation

The KD loss in DEBoost admits a natural Bayesian interpretation as maximum a posteriori (MAP) estimation. In this view, the teacher's predictions define a prior distribution over the student's output space, and the observed labels provide the likelihood. Minimizing the KD loss is then equivalent to finding the MAP estimate under this prior-likelihood model.

For regression, the combined loss (Eq. 2) corresponds to a Gaussian likelihood with a teacher-centered Gaussian prior:

$$t \mid y \sim \mathcal{N}(y,\, \sigma^2), \quad y \sim \mathcal{N}\big(\hat{y},\, \tfrac{1-\alpha}{\alpha}\,\sigma^2\big) \tag{8}$$

where the teacher's prediction $\hat{y}$ centers the prior, and α controls its precision—larger α yields a tighter prior (stronger teacher trust).

For binary classification with temperature T, the loss corresponds to a temperature-transformed Bernoulli likelihood with a teacher-centered Beta prior:

$$\begin{aligned} t \mid y_T &\sim \text{Bernoulli}(F_T(y_T)), \\ y_T &\sim \text{Beta}\Big(\tfrac{\alpha T^2}{1-\alpha}\hat{y}_T + 1,\ \tfrac{\alpha T^2}{1-\alpha}(1-\hat{y}_T) + 1\Big) \end{aligned} \tag{9}$$

where $F_T(p) = \text{logistic}(T\,\text{logit}(p))$ maps a temperature-scaled probability back to the untempered probability. This generalizes to a Categorical likelihood with a teacher-centered Dirichlet prior for K-class classification.

This Bayesian perspective provides three practical insights: (1) α, which appears as a mixing weight in the loss function, controls the precision of the teacher-induced prior; (2) temperature T, originally a heuristic for softening teacher outputs, modulates the concentration of the Beta/Dirichlet prior; and (3) the same MAP perspective extends naturally across squared, logistic, and multi-class losses through the corresponding teacher-centered prior family. Full derivations are provided in Appendix B.

VI. Experiments

In production machine learning pipelines, models are periodically refreshed to adapt to evolving data distributions. Full retraining on all accumulated data becomes increasingly expensive as data volumes grow, requiring historical data assembly, storage, transfer, and compute that scale with data size. Training only on the current data window is cheaper but may discard useful historical patterns. DEBoost targets the middle ground: it refreshes the model using only the current data window while inheriting historical knowledge from teacher scores on that same window.

To validate this approach, we conduct experiments in two complementary public-data settings: (1) controlled static benchmarks with UCI datasets (§VI-A) and (2) multi-cycle model refresh under real temporal drift (§VI-B). For both settings, we quantify DEBoost's training-cost savings (§VI-C); we also conduct a controlled sensitivity analysis of the distillation hyperparameters on MAGIC (§VI-D). All experiments use LightGBM with early stopping (50 rounds without validation improvement). Each configuration is repeated 25 times with different random seeds; we report mean ± standard error. In all settings, DEBoost trains on the same current-window data as the equal-budget baseline, additionally incorporating teacher knowledge through the KD loss. We report Full-Retrain (LightGBM trained on all available data) as a high-cost reference point, while DEBoost trains on only 20–25% of the data per refresh (Fig. 2).

A. Static Evaluation

We evaluate DEBoost on six public UCI datasets—three for classification (MAGIC [23], Diabetes [24], SUSY [25]) and three for regression (Power Plant [26], Superconductivity [27], Energy Farm [28])—spanning 9.6K to 5M instances. Each dataset is split 75%/25% to simulate a historical/recent data scenario: the teacher is trained on the historical 75%, while both the baseline and DEBoost train exclusively on the recent 25%. Thus, the baseline and DEBoost have the same supervised-data budget; DEBoost differs only by access to teacher scores on the recent data.

1) Classification: Table Ia shows that DEBoost outperforms the equal-budget baseline on all three classification datasets, improving ROC-AUC by 0.0008–0.0037 across them. On Diabetes and SUSY, it nearly matches Full-Retrain despite training on only the recent partition. The Self-Distill column isolates the source of this gain: a teacher trained on the same recent data gives no improvement over the baseline, whereas DEBoost's teacher—trained on historical data—improves on every dataset. The benefit therefore comes from transferring historical knowledge, not from the KD loss acting only as a regularizer.

2) Regression: The same pattern holds for regression (Table Ib). DEBoost improves over the equal-budget baseline on all three datasets, raising R^2 by 0.0003–0.0061 across them. On Energy Farm, where the baseline already achieves $R^2 = 0.9937$, the headroom is minimal, yet DEBoost still provides a small consistent improvement.

TABLE I

Static evaluation (25 repeats, mean ± SE). Baseline trains on the recent window only; Self-Distill distills a teacher trained on that same recent data; DEBoost distills a teacher trained on historical data; Full-Retrain trains on all data (a high-cost reference).

(a) Classification (ROC-AUC)

Dataset	Instances	Features	Baseline	Self-Distill	DEBoost	Full-Retrain
MAGIC	19K	10	0.9203 ±0.0015	0.9152 ±0.0020	0.9240 ±0.0018	0.9370 ±0.0013
Diabetes	254K	21	0.8281 ±0.0011	0.8288 ±0.0011	0.8310 ±0.0011	0.8311 ±0.0011
SUSY	5M	18	0.8766 ±0.0001	0.8766 ±0.0001	0.8774 ±0.0001	0.8777 ±0.0001

(b) Regression (R^2)

Dataset	Instances	Features	Baseline	Self-Distill	DEBoost	Full-Retrain
Power Plant	9.6K	4	0.9510 ±0.0010	0.9487 ±0.0010	0.9543 ±0.0011	0.9701 ±0.0012
Superconductivity	21K	81	0.8870 ±0.0018	0.8864 ±0.0019	0.8931 ±0.0016	0.9294 ±0.0012
Energy Farm	54K	148	0.9937 ±0.0002	0.9939 ±0.0002	0.9940 ±0.0001	0.9970 ±0.0001

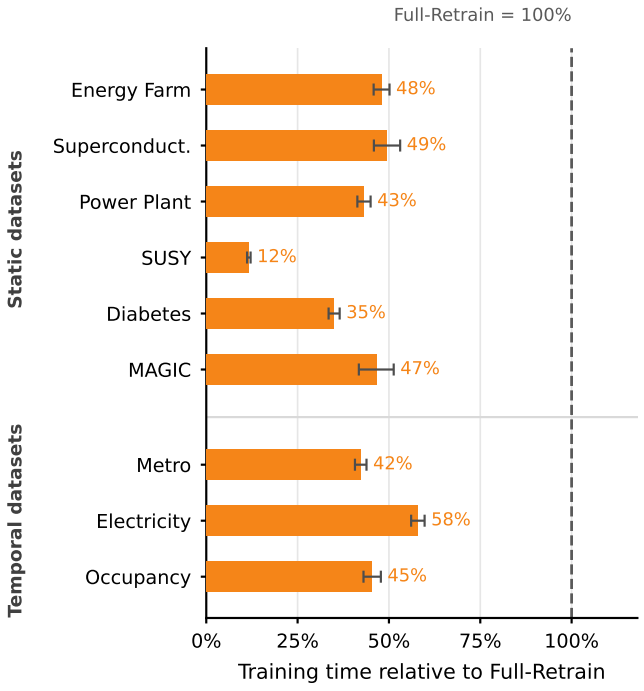


Fig. 2. DEBoost's single-call training time relative to Full-Retrain (= 100%, dashed; lower is better) on all nine public datasets (mean ± SE over 25 seeds); DEBoost trains in 12–58% of the training time required by Full-Retrain.

B. Temporal Evaluation

We evaluate DEBoost on three temporal datasets that exhibit real covariate drift: Electricity (ELEC2) [29], Occupancy Detection [30], and Metro Interstate Traffic [31]. These are time-ordered data streams in which the feature distributions shift over time, so patterns learned in earlier windows partially break in later ones—a genuine continual-learning challenge rather than i.i.d. resampling.

Each dataset is partitioned into 4 equal contiguous time-ordered windows (W_0–W_3), yielding 3 update cycles that simulate successive production model refreshes. The teacher is chained: cycle k's teacher is cycle $(k-1)$'s student, starting from a teacher trained on W_0. Both DEBoost and the baseline (Naive-Refresh) train exclusively on the current window—identical data budgets—while Full-Retrain accumulates all windows seen so far. We additionally compare with iGBDT, described in Section II-B, using its published 250 initial trees and 25 trees per update. We measure the average ROC-AUC (classification) or R^2 (regression) on the current window's test set across the 3 cycles, capturing how well the refreshed model serves incoming data.

To confirm these windows carry genuine distribution shift, we measure the Jensen-Shannon (JS) divergence—a symmetric, bounded $[0, 1]$ distance—between each feature's distribution in consecutive windows, excluding calendar features. Table II jointly reports the drift and performance results. The average maximum JS divergence ranges from 0.13 (Metro) to 0.49 (Occupancy), confirming clear covariate drift across all three streams.

DEBoost outperforms the equal-budget Naive-Refresh baseline by +0.0262 ROC-AUC on Occupancy, +0.0041 ROC-AUC on Electricity, and +0.0111 R^2 on Metro. Relative to iGBDT, DEBoost gains +0.0312 ROC-AUC on Electricity and +0.0033 R^2 on Metro, with a −0.0038 ROC-AUC difference on Occupancy, where DEBoost already achieves a high ROC-AUC of 0.9954. Moreover, iGBDT requires access to and carries forward the previous tree structures and parameters, whereas DEBoost uses only teacher predictions to train a fresh model.

Across these three streams, we observe a trend: higher measured drift coincides with a stronger DEBoost advantage over Full-Retrain. Specifically, DEBoost's standing relative to Full-Retrain improves in the same order as JS divergence: it trails by 0.0024 R^2 on Metro (JS 0.13), matches on Electricity (JS 0.32; 0.8544 vs. 0.8544), and outperforms by 0.0324 ROC-AUC on Occupancy (JS 0.49; 0.9954 vs. 0.9630). At the highest drift level, Full-Retrain is the worst arm, consistent with accumulated historical windows diluting the model toward stale patterns.

TABLE II

Temporal evaluation (25 repeats, mean ± SE). Drift is the average maximum JS divergence between consecutive windows, excluding calendar features. iGBDT retains the preceding ensemble, initialized with 250 trees, and appends 25 trees per update. Full-Retrain is the high-cost reference.

Task	Dataset (instances, features)	Avg. max JS (most drifting features)	Naive-Refresh	iGBDT	DEBoost	Full-Retrain
Classification (ROC-AUC)	Occupancy (20K, 5)	0.49 CO_2 (0.62), Temperature (0.55)	0.9692 ±0.0054	0.9992 ±0.0001	0.9954 ±0.0012	0.9630 ±0.0095
	Electricity (45K, 8)	0.32 vicdemand (0.48), transfer (0.48)	0.8503 ±0.0002	0.8232 ±0.0008	0.8544 ±0.0008	0.8544 ±0.0003
Regression (R^2)	Metro (48K, 11)	0.13 clouds (0.26), weather (0.21)	0.9500 ±0.0001	0.9578 ±0.0001	0.9611 ±0.0001	0.9635 ±0.0001

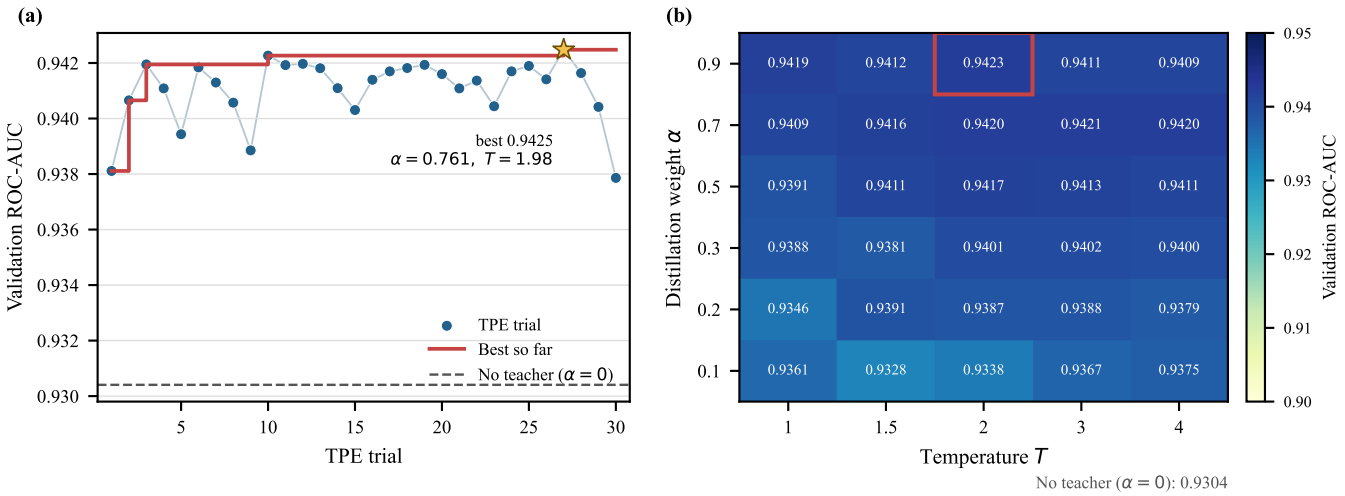


Fig. 3. Controlled tuning of the distillation parameters on MAGIC with all LightGBM parameters fixed. At $\alpha = 0$, the teacher term is removed and no teacher knowledge is used. (a) TPE optimization history over 30 trials; every trial lies above the same-split no-teacher control (dashed, ROC-AUC 0.9304), and the best reaches 0.9425 at $\alpha = 0.761$ and $T = 1.98$. (b) Validation ROC-AUC over the 30-point $\alpha \times T$ grid; every cell exceeds the same control, and the box marks the best setting ($\alpha = 0.9$, $T = 2$, ROC-AUC 0.9423).

DEBoost instead retains historical information through teacher scores while fitting the current window directly.

C. Cost Analysis

DEBoost refreshes on only the current data window—roughly 25% of the data used by full retraining—so each update is substantially cheaper to train. Figure 2 reports its single-call training time relative to Full-Retrain on all nine public datasets: DEBoost trains in only 12–58% of the training time required by Full-Retrain, a 42–88% reduction.

These savings become increasingly important as the accumulated history grows: avoiding a full-history retrain at every cycle reduces compute and eliminates the data-pipeline overhead of assembling, transferring, and storing the entire historical dataset for each update.

D. Hyperparameter Sensitivity

Using MAGIC as a representative controlled example, we fix all LightGBM parameters and apply both search procedures on a predeclared validation split (Fig. 3). The no-teacher control ($\alpha = 0$), which reduces to ordinary supervised training on the recent window, obtains 0.9304 ROC-AUC. Every configuration evaluated by both methods improves on this control: the 30 TPE trials span 0.9379–0.9425, and the 30 grid settings span 0.9328–0.9423. TPE reaches 0.9425 at $\alpha = 0.761$ and $T = 1.98$, while the grid identifies the same high-performing region, reaching 0.9423 at $\alpha = 0.9$ and $T = 2$. Thus, across all evaluated $\alpha > 0$ and T settings, incorporating teacher knowledge improves validation ROC-AUC, while the grid shows that performance remains strong over a broad parameter range.

VII. Future Work

We will explore additional applications in machine learning acceleration. Specifically, given a high-performing LightGBM model, we can leverage DEBoost to construct a compact model with a lighter structure (e.g., fewer trees) for latency optimization. Future work will also investigate the use of output-level distillation for cold-start settings and foundation-model fine-tuning for tabular data.

VIII. Conclusion

We presented DEBoost, a continual learning framework that integrates knowledge distillation into the gradient-boosting loop (instantiated with LightGBM in our experiments). By training only on the recent data window while distilling teacher predictions, DEBoost refreshes models without reprocessing historical data, retaining prior knowledge at a fraction of the cost of full retraining. On public benchmarks, it improves over equal-budget recent-window retraining in both ROC-AUC and R^2, and a self-distillation ablation confirms that the gain comes from historical teacher information. Under real temporal drift, DEBoost outperforms Naive-Refresh on every dataset. We also observe that its performance relative to Full-Retrain improves with measured drift, despite Full-Retrain using all accumulated data and DEBoost using only the current window. Across these public benchmarks, DEBoost achieves this while training on only 20–25% of the data, in 12–58% of the training time required by Full-Retrain (a 42–88% reduction). Because it consumes only teacher predictions, DEBoost provides a practical path to cost-efficient continual refresh for production boosted-tree systems without requiring historical raw data, access to teacher internals, or shared model parameters.

References

[1] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," Annals of statistics, pp. 1189–1232, 2001.

[2] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," Advances in neural information processing systems, vol. 30, 2017.

[3] D. Zhang and S. Sen, "The stochastic conjugate subgradient algorithm for kernel support vector machines," SIAM Journal on Optimization, vol. 35, no. 2, pp. 1194–1215, 2025.

[4] D. Zhang and Y. Zhang, "Beyond first-order: Training llms with stochastic conjugate subgradients and adamw," arXiv preprint arXiv:2507.01241, 2025.

[5] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," Advances in neural information processing systems, vol. 30, 2017.

[6] Z. Li and D. Hoiem, "Learning without forgetting," IEEE transactions on pattern analysis and machine intelligence, vol. 40, no. 12, pp. 2935–2947, 2017.

[7] A. Rannen, R. Aljundi, M. B. Blaschko, and T. Tuytelaars, "Encoder based lifelong learning," in Proceedings of the IEEE international conference on computer vision, 2017, pp. 1320–1328.

[8] C. Wu, L. Herranz, X. Liu, J. Van De Weijer, B. Raducanu et al., "Memory replay gans: Learning to generate new categories without forgetting," Advances in neural information processing systems, vol. 31, 2018.

[9] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska et al., "Overcoming catastrophic forgetting in neural networks," Proceedings of the national academy of sciences, vol. 114, no. 13, pp. 3521–3526, 2017.

[10] K. Wang, A. Liu, J. Lu, G. Zhang, and L. Xiong, "An elastic gradient boosting decision tree for concept drift learning," in AI 2020: Advances in Artificial Intelligence. Springer, 2020, pp. 420–432.

[11] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," arXiv preprint arXiv:1812.00420, 2018.

[12] M. Farajtabar, N. Azizan, A. Mott, and A. Li, "Orthogonal gradient descent for continual learning," in International Conference on Artificial Intelligence and Statistics. PMLR, 2020, pp. 3762–3773.

[13] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2017, pp. 2001–2010.

[14] A. Douillard, M. Cord, C. Ollion, T. Robert, and E. Valle, "Podnet: Pooled outputs distillation for small-tasks incremental learning," in Computer vision–ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, proceedings, part XX 16. Springer, 2020, pp. 86–102.

[15] Z. Wang, L. Liu, Y. Duan, and D. Tao, "Continual learning through retrieval and imagination," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, no. 8, 2022, pp. 8594–8602.

[16] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," International Journal of Computer Vision, vol. 129, no. 6, pp. 1789–1819, 2021.

[17] N. C. Oza and S. J. Russell, "Online bagging and boosting," in International workshop on artificial intelligence and statistics. PMLR, 2001, pp. 229–236.

[18] M. Tang, Q. Zhao, S. X. Ding, H. Wu, L. Li, W. Long, and B. Huang, "An improved lightgbm algorithm for online fault detection of wind turbine gearboxes," Energies, vol. 13, no. 4, p. 807, 2020.

[19] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," arXiv preprint arXiv:1503.02531, 2015.

[20] K. Li, S. Yao, Z. Zhang, B. Cao, C. M. Wilson, D. Kalos, P. F. Kuan, R. Zhu, and X. Wang, "Efficient gradient boosting for prognostic biomarker discovery," Bioinformatics, vol. 38, no. 6, pp. 1631–1638, 2022.

[21] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in Advances in Neural Information Processing Systems, vol. 24, 2011, pp. 2546–2554.

[22] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. Association for Computing Machinery, 2019, pp. 2623–2631.

[23] R. Bock, "Magic gamma telescope," UCI Machine Learning Repository, 2004, DOI: https://doi.org/10.24432/C52C8B.

[24] "CDC Diabetes Health Indicators," UCI Machine Learning Repository, 2017, DOI: https://doi.org/10.24432/C53919.

[25] D. Whiteson, "Susy," UCI Machine Learning Repository, 2014, DOI: https://doi.org/10.24432/C54606.

[26] P. Tfekci and H. Kaya, "Combined Cycle Power Plant," UCI Machine Learning Repository, 2014, DOI: https://doi.org/10.24432/C5002N.

[27] K. Hamidieh, "Superconductivty Data," UCI Machine Learning Repository, 2018, DOI: https://doi.org/10.24432/C53P47.

[28] M. Neshat, B. Alexander, N. Y. Sergiienko, and M. Wagner, "Optimisation of large wave farms using a multi-strategy evolutionary framework," in Proceedings of the 2020 genetic and evolutionary computation conference, 2020, pp. 1150–1158.

[29] M. Harries, "Splice-2 comparative evaluation: Electricity pricing," Technical Report, University of New South Wales, 1999.

[30] L. M. Candanedo and V. Feldheim, "Accurate occupancy detection of an office room from light, temperature, humidity and CO2 measurements using statistical learning models," Energy and Buildings, vol. 112, pp. 28–39, 2016.

[31] J. Hogue, "Metro interstate traffic volume data set," UCI Machine Learning Repository, 2019.

Appendix A
Gradient and Hessian Derivations

This appendix derives the per-example gradients and Hessians used by DEBoost. Labels and teacher predictions are fixed; all derivatives are taken with respect to the student's raw score. We write $g = \partial L/\partial z$ and $h = \partial^2 L/\partial z^2$ in the scalar cases. For multiclass classification, $\boldsymbol{H}$ denotes

the full Hessian with respect to the logit vector, and $h_k = H_{kk}$ denotes the diagonal entry supplied to LightGBM.

A.1 Regression

Regression uses the identity link, so the raw score is the prediction: $z = y$. From (2), the per-example objective is

$$L = \frac{1-\alpha}{2}(y-t)^2 + \frac{\alpha}{2}(y-\hat{y})^2. \tag{10}$$

Because t and $\hat{y}$ do not depend on the student score, differentiation gives

$$\begin{aligned} g &= (1-\alpha)(y-t) + \alpha(y-\hat{y}) \\ &= y - \big[(1-\alpha)t + \alpha\hat{y}\big], \\ h &= (1-\alpha) + \alpha = 1. \end{aligned} \tag{11}$$

Thus, the Newton target is centered on a convex combination of the observed label and the teacher prediction, while the curvature remains constant.

A.2 Binary Classification

For a soft target $q \in [0,1]$ and prediction $p \in (0,1)$, define binary cross-entropy as

$$\mathrm{CE}(q,p) = -\left[q \ln p + (1-q)\ln(1-p)\right].$$

After dropping terms independent of the student, (3) becomes

$$L = (1-\alpha)\,\mathrm{CE}(t,y) + \alpha T^2\,\mathrm{CE}(\hat{y}_T, y_T), \tag{12}$$

where $y = \sigma(z)$, $y_T = \sigma(z/T)$, and $\sigma(u) = 1/(1+\exp(-u))$. The basic logistic identities are

$$\frac{\partial y}{\partial z} = y(1-y), \qquad \frac{\partial y_T}{\partial z} = \frac{1}{T} y_T(1-y_T). \tag{13}$$

More generally, if $p = \sigma(u)$, then

$$\begin{aligned} \frac{\partial\,\mathrm{CE}(q,p)}{\partial u} &= \left(-\frac{q}{p} + \frac{1-q}{1-p}\right) p(1-p) \\ &= p - q. \end{aligned} \tag{14}$$

Applying (14) first with $u = z$ and then with $u = z/T$ yields

$$\begin{aligned} g &= (1-\alpha)(y-t) + \alpha T^2 \frac{1}{T}(y_T - \hat{y}_T) \\ &= (1-\alpha)(y-t) + \alpha T(y_T - \hat{y}_T), \\ h &= (1-\alpha)y(1-y) + \alpha y_T(1-y_T). \end{aligned} \tag{15}$$

The conventional T^2 scaling leaves one factor of T in the distillation gradient; the second factor cancels when differentiating again, producing the Hessian in (15).

A.3 Multiclass Classification

Let $\boldsymbol{z} \in \mathbb{R}^K$ be the student logits, and define

$$y_i = \frac{\exp(z_i)}{\sum_{j=1}^K \exp(z_j)}, \qquad y_{i,T} = \frac{\exp(z_i/T)}{\sum_{j=1}^K \exp(z_j/T)}.$$

For probability vectors $\boldsymbol{q}$ and $\boldsymbol{p}$, define $\mathrm{CE}(\boldsymbol{q},\boldsymbol{p}) = -\sum_{i=1}^K q_i \ln p_i$. After removing teacher-only constants, (4) is

$$L = (1-\alpha)\,\mathrm{CE}(\boldsymbol{t},\boldsymbol{y}) + \alpha T^2\,\mathrm{CE}(\hat{\boldsymbol{y}}_T, \boldsymbol{y}_T). \tag{16}$$

Using the Kronecker delta δ_{ik}, the softmax Jacobians are

$$\frac{\partial y_i}{\partial z_k} = y_i(\delta_{ik} - y_k), \qquad \frac{\partial y_{i,T}}{\partial z_k} = \frac{1}{T} y_{i,T}(\delta_{ik} - y_{k,T}). \tag{17}$$

Since $\sum_i t_i = \sum_i \hat{y}_{i,T} = 1$, these identities give

$$\begin{aligned} \frac{\partial\,\mathrm{CE}(\boldsymbol{t},\boldsymbol{y})}{\partial z_k} &= -\sum_{i=1}^K t_i(\delta_{ik} - y_k) = y_k - t_k, \\ \frac{\partial\,\mathrm{CE}(\hat{\boldsymbol{y}}_T,\boldsymbol{y}_T)}{\partial z_k} &= -\frac{1}{T}\sum_{i=1}^K \hat{y}_{i,T}(\delta_{ik} - y_{k,T}) \\ &= \frac{1}{T}(y_{k,T} - \hat{y}_{k,T}). \end{aligned} \tag{18}$$

Therefore, the gradient for class k is

$$g_k = (1-\alpha)(y_k - t_k) + \alpha T(y_{k,T} - \hat{y}_{k,T}). \tag{19}$$

Differentiating g_k with respect to every logit z_j gives the full Hessian:

$$\begin{aligned} H_{kj} &= \frac{\partial g_k}{\partial z_j} \\ &= (1-\alpha)y_k(\delta_{kj} - y_j) + \alpha y_{k,T}(\delta_{kj} - y_{j,T}). \end{aligned} \tag{20}$$

The implementation uses the diagonal entries

$$h_k = H_{kk} = (1-\alpha)y_k(1-y_k) + \alpha y_{k,T}(1-y_{k,T}). \tag{21}$$

For completeness, the off-diagonal entries are $H_{kj} = -(1-\alpha)y_k y_j - \alpha y_{k,T} y_{j,T}$ for $k \neq j$. When $T = 1$, the classification formulas reduce to the expressions given in Section IV.

Appendix B
MAP Interpretation of the Distillation Objectives

This appendix makes the Bayesian interpretation in Section V precise. In each case, the DEBoost objective equals a negative log posterior up to a positive multiplicative factor and an additive constant; neither changes the optimizer. We assume $0 < \alpha < 1$. The endpoints $\alpha = 0$ and $\alpha = 1$ follow as limiting cases. Throughout this appendix, C and C' denote terms that do not depend on the student prediction being optimized.

B.1 Regression

Consider the Gaussian likelihood and teacher-centered Gaussian prior

$$t \mid y \sim \mathcal{N}(y, \sigma^2), \qquad y \sim \mathcal{N}\left(\hat{y}, \frac{1-\alpha}{\alpha}\sigma^2\right). \tag{22}$$

Bayes' rule gives the log posterior

$$\begin{aligned} \ln p(y \mid t) = C &- \frac{(y-t)^2}{2\sigma^2} \\ &- \frac{\alpha(y-\hat{y})^2}{2(1-\alpha)\sigma^2}. \end{aligned} \tag{23}$$

Multiplying its negative by the positive factor $(1-\alpha)\sigma^2$ yields

$$\begin{aligned} -(1-\alpha)\sigma^2 \ln p(y \mid t) &= \frac{1-\alpha}{2}(y-t)^2 \\ &\quad + \frac{\alpha}{2}(y-\hat{y})^2 + C' \\ &= L + C', \end{aligned} \tag{24}$$

which is exactly the regression objective in (2) up to an additive constant. Consequently, minimizing that objective produces the MAP estimate. Its closed-form value,

$$y_{\text{MAP}} = (1-\alpha)t + \alpha\hat{y},$$

also matches the blended target identified by the gradient in (11).

B.2 Binary Classification

Define the classification prior strength

$$c = \frac{\alpha T^2}{1-\alpha}. \tag{25}$$

For a temperature-scaled probability p, define the map back to the untempered probability by

$$\begin{aligned} F_T(p) &= \sigma\left(T \operatorname{logit}(p)\right) \\ &= \frac{p^T}{p^T + (1-p)^T}. \end{aligned} \tag{26}$$

Because $y_T = \sigma(z/T)$, this map satisfies $F_T(y_T) = \sigma(z) = y$. Consider the likelihood and teacher-centered prior

$$\begin{aligned} t \mid y_T &\sim \text{Bernoulli}\left(F_T(y_T)\right), \\ y_T &\sim \text{Beta}\left(1 + c\hat{y}_T,\, 1 + c(1-\hat{y}_T)\right). \end{aligned} \tag{27}$$

The likelihood and prior densities are

$$p(t \mid y_T) = y^t(1-y)^{1-t} \tag{28}$$

and

$$\begin{aligned} p(y_T) = &\frac{1}{\text{B}\left(1 + c\hat{y}_T,\, 1 + c(1-\hat{y}_T)\right)} \\ &\times y_T^{c\hat{y}_T}(1-y_T)^{c(1-\hat{y}_T)}, \end{aligned} \tag{29}$$

where $\text{B}(\cdot,\cdot)$ is the beta function. Their log posterior is therefore

$$\begin{aligned} \ln p(y_T \mid t) = C &+ t \ln y + (1-t)\ln(1-y) \\ &+ c\left[\hat{y}_T \ln y_T + (1-\hat{y}_T)\ln(1-y_T)\right]. \end{aligned} \tag{30}$$

Using $(1-\alpha)c = \alpha T^2$ gives

$$\begin{aligned} -(1-\alpha)\ln p(y_T \mid t) &= (1-\alpha)\,\text{CE}(t, y) \\ &\quad + \alpha T^2\,\text{CE}(\hat{y}_T, y_T) + C' \\ &= L + C', \end{aligned} \tag{31}$$

which proves the MAP equivalence for (3). The Beta prior has mode $\hat{y}_T$ and total concentration $c+2$, so α and T jointly control how strongly the posterior favors the teacher. When $T=1$, F_T is the identity and this reduces to standard Beta–Bernoulli conjugacy. For general T, the likelihood is transformed by F_T; the MAP equivalence remains exact, although the pair is not strictly conjugate.

B.3 Multiclass Classification

For a probability vector $\mathbf{p}$ on the K-simplex, define the multiclass temperature map componentwise by

$$\left[\mathbf{F}_T(\mathbf{p})\right]_k = \frac{p_k^T}{\sum_{j=1}^K p_j^T}. \tag{32}$$

For $\mathbf{y}_T = \text{softmax}(\mathbf{z}/T)$, the map recovers the untempered prediction:

$$\mathbf{F}_T(\mathbf{y}_T) = \text{softmax}(\mathbf{z}) = \mathbf{y}.$$

For a one-hot label vector $\mathbf{t}$, consider

$$\begin{aligned} \mathbf{t} \mid \mathbf{y}_T &\sim \text{Categorical}\left(\mathbf{F}_T(\mathbf{y}_T)\right), \\ \mathbf{y}_T &\sim \text{Dirichlet}\left(\mathbf{1} + c\hat{\mathbf{y}}_T\right). \end{aligned} \tag{33}$$

The likelihood is

$$p(\mathbf{t} \mid \mathbf{y}_T) = \prod_{k=1}^K y_k^{t_k}, \tag{34}$$

and the Dirichlet density is

$$p(\mathbf{y}_T) = \frac{1}{\text{B}(\mathbf{1} + c\hat{\mathbf{y}}_T)} \prod_{k=1}^K y_{k,T}^{c\hat{y}_{k,T}}, \tag{35}$$

where the multivariate beta function is

$$\text{B}(\boldsymbol{\beta}) = \frac{\prod_k \Gamma(\beta_k)}{\Gamma\left(\sum_k \beta_k\right)}.$$

Combining the likelihood and prior gives

$$\begin{aligned} \ln p(\mathbf{y}_T \mid \mathbf{t}) = C &+ \sum_{k=1}^K t_k \ln y_k \\ &+ c\sum_{k=1}^K \hat{y}_{k,T} \ln y_{k,T}. \end{aligned} \tag{36}$$

Therefore,

$$\begin{aligned} -(1-\alpha)\ln p(\mathbf{y}_T \mid \mathbf{t}) &= (1-\alpha)\,\text{CE}(\mathbf{t}, \mathbf{y}) \\ &\quad + \alpha T^2\,\text{CE}(\hat{\mathbf{y}}_T, \mathbf{y}_T) + C' \\ &= L + C', \end{aligned} \tag{37}$$

which recovers (4). The Dirichlet prior has mode $\hat{\mathbf{y}}_T$ and concentration $K + c$. As in the binary case, $T=1$ gives standard Categorical–Dirichlet conjugacy, while general T retains the exact MAP equivalence through the temperature map.