Gradient-boosted based structured and unstructured learning

Anonymous Author
Anonymous Institution

Abstract

We propose two frameworks to deal with problem settings in which both structured and unstructured data are available. Structured data problems are best solved by traditional machine learning models such as boosting and tree-based algorithms, whereas deep learning has been widely applied to problems dealing with images, text, audio, and other unstructured data sources. However, for the setting in which both structured and unstructured data are accessible, it is not obvious what the best modeling approach is to enhance performance on both data sources simultaneously. Our proposed frameworks allow joint learning on both kinds of data by integrating the paradigms of boosting models and deep neural networks. The first framework, the boosted-feature-vector deep learning network, learns features from the structured data using gradient boosting and combines them with embeddings from unstructured data via a two-branch deep neural network. Secondly, the two-weak-learner boosting framework extends the boosting paradigm to the setting with two input data sources. We present and compare first- and second-order methods of this framework. Our experimental results on both public and real-world datasets show performance gains achieved by the frameworks over selected baselines by magnitudes of 0.1% - 4.7%.

1 INTRODUCTION

Data in modern machine learning problems can be represented by a variety of modalities or data sources. We consider the setting in which structured data, aka tabular data, and unstructured data are available simultaneously. A common application area of this setting is medical diagnosis, in which the decision making process is supported by unstructured data such as medical imaging and doctors' notes, in combination with patient historical data, lab analyses and blood tests, in the form of structured data.

The settings where structured or unstructured data are available individually have been extensively researched. Deep neural networks (DNNs) have consistently proven successful at solving problems with unstructured data, see e.g. textbooks Balas et al. (2019) and Goodfellow et al. (2016). On the other hand, traditional boosting methods have shown significant advantages over DNNs in modeling structured data inputs (Caruana et al., 2008; Caruana and Niculescu-Mizil, 2006). Examples of such benefits are observed in terms of training time, interpretability, amount of required training data, tuning efforts, and computational expense. This is commonly observed in Kaggle competitions, where better performance is achieved by boosted methods when the available data is structured (Lloyd, 2014; Mangal and Kumar, 2017; Taieb and Hyndman, 2013), and by deep learning models when the available data is unstructured (Graham, 2015; Zou et al., 2017). In particular, LightGBM (Ke et al., 2017) and XGBoost (Chen and Guestrin, 2016), have become de-facto modeling standards for structured data.

Conversely, in the setting in which both structured and unstructured data are accessible ($\mathcal{US}$), it is not obvious what the best modeling approach is to enhance performance on both data sources simultaneously. In general, the simplest method consists of training independent models for each data modality and then combining the results by averaging or voting over the individual predictions. A big caveat is the missed opportunity of capturing any cross-data source interactions or underlying complementary information that might exist in the data. Training concurrently on both modalities of data is deemed crucial if we attempt to learn such relationships. A common approach to joint training consists of using DNNs for representation learning on each data source, concatenating the learned embeddings, and having it as input to a third DNN. This approach serves as a baseline for our experiments, and performs sub-optimally given that boosting algorithms excel on structured data settings.

In this paper, we propose two frameworks for the $\mathcal{US}$ setting

Preliminary work. Under review by AISTATS 2023. Do not distribute.

that address the above-mentioned considerations. Our frameworks aim at better capturing the best and most informative features of each data source, while simultaneously enhancing performance though a joint training scheme. To achieve this, our novel approaches combine the proven paradigms for structured and unstructured data respectively: gradient boosting machines and DNNs.

The first framework is the boosted-feature-vector deep learning network (BFV+DNN). BFV+DNN learns features from the structured data using gradient boosting and combines them with embeddings from unstructured data via a two-branch deep neural network. It requires to train a boosted model on the structured data as an initial step. Then, each neural network branch learns embeddings specific to each data input, which are further fused into a shared trainable model. The post-fusion shared architecture allows the model to learn the complementary cross-data source interactions. The key novelty is the feature extraction process from boosting. Following standard terminology, we refer to model inputs as ”features” and to DNN-learned representations as ”embeddings.” In our proposed framework, BFVs are used as inputs to BFV+DNN and hence, are named features accordingly.

In addition, we propose a two-weak-learner boosting framework (2WL) that extends the boosting paradigm to the USsetting. The framework is derived as a first-order approximation to the gradient boosting risk function and further expanded to a second-order approximation method (2WL2O). It should be noted that this framework can be used in the general multimodal setting and is not restricted to the USuse case. Our experimental results show significant performance gains over the aforementioned baseline. Relative improvements on F1 metrics are observed by magnitudes of 4.7%, 0.1%, and 0.34% on modified Census, Imagenet, and Covertype datasets, respectively. We also consider a real-world dataset from an industry partner where the improvement in accuracy is 0.41%.

The main contributions of this work are as follows.

1. We present a boosted-feature-vector DNN model that combines structured data boosting features with deep neural networks to address the setting in which both structured and unstructured data sources are available.

2. We propose an alternative two-weak-learner-gradient-boosting framework to address the setting in which both structured and unstructured data sources are available.

3. We extend the two-weak-learner-gradient-boosting to a second-order approximation.

4. We show and compare the effectiveness of these approaches on public and real-world datasets.

The rest of this paper is organized as follows. In Section 2, we discuss related work. In Section 3, we formally introduce our proposed frameworks. Experimental results are discussed in Section 4 and we conclude in Section 5.

2 Related work

2.1 Boosting Methods

Boosting methods combine base models (referred to as weak learners) as a means to improve the performance achieved by individual learners (Ridgeway, 1999). AdaBoost (Freund and Schapire, 1997) is one of the first concrete adaptive boosting algorithms, whereas Gradient Boosting Machines (GBM) (Friedman, 2000) derive the boosting algorithm from the perspective of optimizing a loss function using gradient descent, see Mayr et al. (2014). A formulation of gradient boosting for the multi-class setting and two algorithmic approaches are proposed in Saberian and Vasconcelos (2011). LightGBM (Ke et al., 2017) incorporates techniques to improve GBM's efficiency and scalability. Traditionally, trees have been the base learners of choice for boosting methods, but the performance of neural networks as weak learners for AdaBoost has also been investigated in Schwenk and Bengio (2000). More recently, CNNs were explored as weak learners for GBM in Moghimi et al. (2016), integrating the benefits of boosting algorithms with the impressive results that CNNs have obtained at learning representations on visual data (Jia et al., 2014; Krizhevsky et al., 2012; Redmon et al., 2016). Second-order information is employed in boosting algorithms such as Logitboost (Friedman et al., 2000), Taylorboost (Saberian et al., 2011), and XGBoost (Chen and Guestrin, 2016). However, unlike our proposed second-order model, all these algorithms consider a single family of weak learners and individual data inputs, whereas we handle two families of weak learners and both structured and unstructured data simultaneously.

Boosting approaches have also been applied to the setting in which more than one data source is available as input. For instance, a multiview boosting algorithm based on PAC-Bayesian theory is presented in Goyal et al. (2018) and a cost-based multimodal approach that introduces the notion of weak and strong modalities in Koço et al. (2012). In both cases, final classification is performed using majority or weighted voting. Similarly, a model that assigns a different contribution of each data input to the final classification is proposed in Peng et al. (2018), where a shared weight distribution among modalities is used. None of these algorithms make use of DNN approaches, as they employ traditional decision stumps as weak learners regardless of the data input sources. In contrast, the multimodal reward-penalty-based voting boosting model proposed in Lahiri et al. (2018) uses DNNs as weak learners, but overlooks the benefits of tree-based approaches for structured data. Common to all methods reviewed in this paragraph, is the notion of addressing the setting where multiple data inputs are available. However, they do not take into account the underlying properties of these different data sources, nor do they consider specific algorithmic approaches that better suit each one of them.

2.2 Structured & Unstructured Data Setting

Approaches that directly target the *US*setting are scarce, more so those that address the structured data characteristics. In Chen et al. (2017), the authors deal with demographics, living habits, and examination results from patients in the form of structured data, and with doctor's records and patient's medical history presented as unstructured text data. An intuitive DNN approach is used, with the drawbacks that have already been discussed as no special treatment is given to structured data.

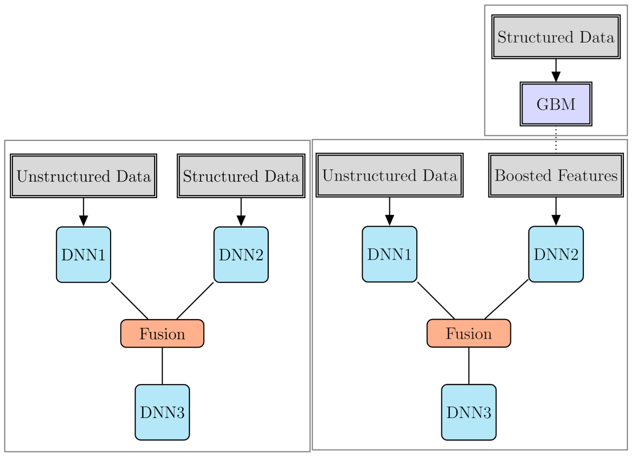
Conversely, in Li et al. (2019) the *US*setting is tackled by combining the benefits of tree-based models and DNNs. To do so, they use stacking and boosted stacking of independently trained models. The core idea is similar to our proposed boosted-feature-vector DNN in the sense that a first model is trained and then used as an input for joint training. Their approaches differ from our BFV+DNN in two main aspects. First, their models are heavily tailored for the learning-to-rank use case and second, they use direct outputs from the first model as input to the second model, whereas we propose a novel way to extract boosted-feature vectors from the first model, rather than using its direct output.

3 Proposed models

In this section, we propose two models to tackle the *US*setting. The models address the inherent nature of each source of data by exploiting the specific benefits of boosted algorithms and neural networks as learners on each of them.

3.1 Boosted-feature-vector Deep Learning Network (BFV+DNN)

The boosted-feature-vector deep learning network aims at using DNNs as the primary learning method, while incorporating boosted-feature vectors (BFV) from the structured data source. As a means of comparison, the baseline DNN approach to the *US*setting is shown in Figure 1a and the BFV+DNN architecture in Figure 1b. Both contain two branches, a fusion stage and a joint learning architecture. Each branch learns representations from one data source (see DNN1 and DNN2 in Figure 1). Then, a fusion yields a joint embedding that combines the data-source-specific representations. Finally, the combined vector is used as



(a) Baseline architecture (b) BFV+DNN architecture

Figure 1: *US*deep neural networks

input to a trainable DNN to model cross-data source interactions (DNN3 in Figure 1).

As noted, the common baseline ignores the structured or unstructured nature of the data source and directly learns representations via DNNs, whereas BFV+DNN uses BFVs as input to DNN2. To do so, we assume that a GBM model is first trained on the structured data. For the multiclass setting with M classes and N iterative GBM stages, M CARTs (Breiman et al., 1984) are fitted per iteration. Let $R_{i,j,k}$ be the region defined by class i, tree j, and leaf k, and $w_{i,j,k}$ the value representing the raw prediction of a sample falling in the corresponding region $(1 \leq i \leq M, 1 \leq j \leq N)$. Moreover, let each fitted tree j of class i have a number of leaves $l_{i,j}$. We define the boosted-feature vector of the structured portion of a sample x as $BFV(x) \in \mathbb{R}^{M \text{x} N}$, $BFV(x) =$

$$\begin{bmatrix} \sum_{k=1}^{l_{1,1}} w_{1,1,k}\mathbb{1}\{x \in R_{1,1,k}\} \ , & \dots & , \sum_{k=1}^{l_{M,1}} w_{M,1,k}\mathbb{1}\{x \in R_{M,1,k}\} \\ \sum_{k=1}^{l_{1,2}} w_{1,2,k}\mathbb{1}\{x \in R_{1,2,k}\} \ , & \dots & , \sum_{k=1}^{l_{M,2}} w_{M,2,k}\mathbb{1}\{x \in R_{M,2,k}\} \\ \vdots & & \vdots \\ \sum_{k=1}^{l_{1,N}} w_{1,N,k}\mathbb{1}\{x \in R_{1,N,k}\} \ , & \dots & , \sum_{k=1}^{l_{M,N}} w_{M,N,k}\mathbb{1}\{x \in R_{M,N,k}\} \end{bmatrix}.$$

Note that for the binary case, a single tree is fitted in each iteration and the boosted-feature vector of x is simplified to $BFV(x) \in \mathbb{R}^N$, $BFV(x) =$

$$\left[\sum_{k=1}^{l_1} w_{1,k}\mathbb{1}\{x \in R_{1,k}\} \ , \quad \dots \quad , \ \sum_{k=1}^{l_N} w_{N,k}\mathbb{1}\{x \in R_{N,k}\} \right],$$

where $w_{j,k}$ is the raw prediction of a sample falling in region $R_{j,k}$ of tree j and leaf k.

The outputs of DNN1 and DNN2 are combined into a joint representation using a fusion method. In our experiments in Section 4, fusing steps are performed by concatenatation or element-wise multiplication of the DNNs' embeddings. The best method per dataset is reported in Table 1.

3.2 Two-Weak-Learner-Gradient-Boosting Framework

Multi-class boosting aims at finding a classifier $F(x) = \arg\max_k \langle y^k, f(x)\rangle$ where f is some predictor, y^k is the k^{th} class unit vector identifier, and $\langle \cdot,\cdot \rangle$ is the standard dot product. Following the GD-MCBoost (Saberian and Vasconcelos, 2011) multi-class boosting approach, f is a boosted predictor trained to minimize classification risk $R(f) = \mathbb{E}_{X,Y}[L(y, f(x))] \approx \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i))$ where n is the number of training samples and $L(y, f(x)) = \sum_{k=1}^{M} e^{-\frac{1}{2}[<f(x), y-y^k>]}$ is the M-class loss function. At each iteration t, the update of the predictor is given by $f^{t+1}(x) = f^t(x) + g(x)$ with $g(x)$ a weak learner. Although the most common choices for weak learners are decision trees, we posit that weak learners must be chosen according to the available data source, such that they best capture their specific properties. In the $\mathcal{US}$ setting, each training sample is of the form $((x^U, x^S), y)$, and we have two families of weak learners denoted by $g = g(x^U)$ and $h = h(x^S)$.

3.2.1 Two-Weak-Learner-First-Order-Gradient-Boosting Framework (2WL)

The two-weak-learner-gradient-boosting framework integrates the boosting paradigm to the $\mathcal{US}$ setting by including two families of weak learners that target each specific data input.

In the two-weak-learner case, given f^t we have weak learners g and h. We update the predictor at iteration $t+1$ to $f^{t+1}((x^U, x^S)) = f^t((x^U, x^S)) + \epsilon g^*(x^U) + \delta h^*(x^S)$. The optimization step is taken via gradient descent along directions g and h of largest decrease of $R(f)$. We have that (see Appendix A):

$$R(f^t + \epsilon g + \delta h) \approx R(f^t) + \frac{\partial R}{\partial \epsilon}\Big|_{\substack{\epsilon=0\\ \delta=0}}\epsilon + \frac{\partial R}{\partial \delta}\Big|_{\substack{\epsilon=0\\ \delta=0}}\delta,$$

where

$$\frac{\partial R}{\partial \epsilon}\Big|_{\substack{\epsilon=0\\ \delta=0}}\epsilon = -\epsilon\sum_{i=1}^{n} < g(x_i^U), w_i >,$$

$$\frac{\partial R}{\partial \delta}\Big|_{\substack{\epsilon=0\\ \delta=0}}\delta = -\delta\sum_{i=1}^{n} < h(x_i^S), w_i >,$$

$$w_i = \frac{1}{2}e^{-\frac{1}{2}<f^t(x_i^U, x_i^S), y_i>}\sum_{k=1}^{M}(y_i - y^k)e^{\frac{1}{2}<f^t(x_i^U, x_i^S), y^k>}, \tag{1}$$

which yields optimization problems:

$$g^* \in \arg\min_g \qquad ||g - w||^2 = \sum_{i=1}^{n} ||g(x_i) - w_i|| \tag{2}$$

$$h^* \in \arg\min_h \qquad ||h - w||^2 = \sum_{i=1}^{n} ||h(x_i) - w_i|| \tag{3}$$

$$(\epsilon^*, \delta^*) \in \arg\min_{\epsilon,\delta} \qquad R(f + \epsilon g^* + \delta h^*). \tag{4}$$

These problems are solved iteratively using Algorithm 1. At each iteration, weak learners g and h are fitted to minimize the expressions shown in (2) and (3) for w_i as in (1). Risk function $R(f)$, evaluated in the learned values, is optimized with respect to ϵ and δ. Problems 2 and 3 are solved by using standard mean squared error algorithms. Optimization 4 can be approximated in different ways such as heuristics, grid search, randomized search, or Bayesian optimization. Our experimental study, detailed in Section 4, uses heuristic values or Bayes optimization.

Algorithm 1 Two-Weak-Learner-Gradient-Boosting

Input: Number of classes M, number of boosting iterations N and training dataset $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$, where x_i are training samples of the form $x_i = (x_i^1, x_i^2)$, with x_i^1 corresponding to one modality, x_i^2 corresponding to the second modality, and y_i are the class labels. In our use case, $x_i = (x_i^U, x_i^S)$.

```
Initialization: Set $f^0 = 0 \in \mathbb{R}^M$
for $t = 0$ to $N$ do
    Compute $w_i$ as in (1).
    Fit learners $g^*$ and $h^*$ as in (2) and (3).
    Find $\epsilon^*$ and $\delta^*$ as in (4).
    Update $f^{t+1}(x) = f^t(x) + \epsilon^* g^*(x_{i_1}) + \delta^* h^*(x_{i_2})$ .
end for
```

Output: $F(x) = \arg\max_k \langle y^k, f^N(x)\rangle$

3.2.2 Two-Weak-Learner-Second-Order Gradient Boosting Framework (2WL2O)

The two-weak-learner-gradient-boosting framework is derived from the first-order approximation to the multi-class risk function R. In order to improve the estimation, we use second-order Taylor approximation as follows (details are provided in Appendix B):

$$R_M(f^t + \epsilon g + \delta h) \approx R(f^t) + \frac{\partial R}{\partial \epsilon}\Big|_{\substack{\epsilon=0\\ \delta=0}}\epsilon + \frac{\partial R}{\partial \delta}\Big|_{\substack{\epsilon=0\\ \delta=0}}\delta$$
$$+ \frac{1}{2}\frac{\partial^2 R}{\partial \epsilon^2}\Big|_{\substack{\epsilon=0\\ \delta=0}}\epsilon^2 + \frac{1}{2}\frac{\partial^2 R}{\partial \delta^2}\Big|_{\substack{\epsilon=0\\ \delta=0}}\delta^2$$
$$+ \frac{\partial^2 R}{\partial \epsilon \partial \delta}\Big|_{\substack{\epsilon=0\\ \delta=0}}\epsilon\delta$$

where

$$\frac{\partial^2 R}{\partial \epsilon^2}\Big|_{\substack{\epsilon=0\\ \delta=0}}\epsilon^2 = \frac{\epsilon^2}{4}\Bigg[\sum_{i=1}^{n}\Big(< g(x_i^1), g(x_i^1) > + 8 < g(x_i^1), \tilde{w}_i > + \hat{w}_i\Big)\Bigg],$$

$$\frac{\partial^2 R}{\partial \delta^2}\Big|_{\substack{\epsilon=0\\ \delta=0}}\delta^2 = \frac{\delta^2}{4}\Bigg[\sum_{i=1}^{n}\Big(< h(x_i^2), h(x_i^2) > + 8 < h(x_i^2), \tilde{w}_i > + \hat{w}_i\Big)\Bigg],$$

$$\frac{\partial^2 R}{\partial \epsilon \partial \delta}\Big|_{\substack{\epsilon=0\\ \delta=0}}\epsilon\delta = \frac{\epsilon\delta}{2}\sum_{i=1}^{n}\Big(< g(x_i^1), w_i > + < h(x_i^2), w_i > \Big),$$

$$\tilde{w}_i = \sum_{k=1}^{M}\Big[(y_i - y^k)(e^{-\frac{1}{2}<f^t(x_i^U, x_i^S), y_i - y^k>})^{\frac{1}{2}}\Big], \quad (5)$$

$$\hat{w}_i = e^{-\frac{1}{2}<f^t(x_i^U, x_i^S), y_i>}\sum_{k=1}^{M}||y_i - y^k||^2 e^{\frac{1}{2}<f^t(x_i^U, x_i^S), y^k>},$$

and w_i as in (1).

We now have that:

$$(\epsilon^*, \delta^*) \in \arg\min_{\epsilon,\delta} \quad R(f + \epsilon g^*(\epsilon,\delta) + \delta h^*(\epsilon,\delta))$$

$$\text{s.t.} \quad g^* \in \arg\min_{g} \quad \Big|\Big|g - (\epsilon w - \frac{\epsilon^2}{4}\tilde{w} - \frac{\epsilon\delta}{2}w)\Big|\Big|^2$$

$$h^* \in \arg\min_{h} \quad \Big|\Big|h - (\delta w - \frac{\delta^2}{4}\tilde{w} - \frac{\epsilon\delta}{2}w)\Big|\Big|^2,$$

which we solve using Algorithm 2. At each iteration, w_i and $\tilde{w}_i$ are computed as in (1) and (5). An inner loop jointly optimizes g, h, ϵ, and δ for these fixed w and $\tilde{w}$: weak learners g and h are fitted to minimize the expressions shown in (3.2.2) and (3.2.2) and $R(f)$, evaluated in the learned values, is optimized with respect to ϵ and δ.

Optimization problems 3.2.2, 3.2.2, and 3.2.2 are solved as stated for Algorithm 1. In the experimental study in Section 4, the initialization values for ϵ_0^* and δ_0^* are set to 0.1, mimicking the default learning rate used in standard GBM implementations.

Algorithm 2 Two-Weak-Learner-Gradient-Boosting-Second-Order

Input: Number of classes M, number of boosting iterations N_1, number of inner iterations N_2 and training dataset $\mathcal{D} = \{(x_1, y_1), ..., (x_n, y_n)\}$, where x_i are training samples of the form $x_i = (x_i^1, x_i^2)$, with x_i^1 corresponding to one modality, x_i^2 corresponding to the second modality, and y_i are the class labels.

```
Initialization: Set $f^0 = 0 \in \mathbb{R}^M$
for $t = 0$ to $N_1$ do
    Compute $w_i$ and $\tilde{w}_i$ as in (1), and (5).
    Initialize $\epsilon_0^*$, $\delta_0^*$.
    for $j = 0$ to $N_2$ do
        Fit learners $g_j^*$ and $h_j^*$ as in (3.2.2) and (3.2.2) by using $\epsilon_j^*$, $\delta_j^*$ .
        Find $\epsilon_{j+1}^*$ and $\delta_{j+1}^*$ as in (3.2.2).
        Compute risk function value $R_j$ at point $(g_j^*, h_j^*, \epsilon_{j+1}^*, \delta_{j+1}^*)$.
    end for
    $j^* = \arg\min_j R_j$
    $g^* = g_{j*}, h^* = h_{j^*}$
    $\epsilon^* = \epsilon_{j^*}, \delta^* = \delta_{j^*}$
    Update $f^{t+1}(x) = f^t(x) + \epsilon^* g^*(x) + \delta^* h^*(x)$.
end for
```

Output: $F(x) = \arg\max_k \langle y^k, f^{N_1}(x)\rangle$

4 Computational study

The computational study of the proposed models was conducted on five datasets: two subsets of the structured Census-Income (KDD) dataset (Dua and Graff, 2017), modified versions of Imagenet (Deng et al., 2009) and UCI Forest Covertype (Blackard, 1999), and a real-world proprietary dataset.

4.1 Datasets

Census-Income Dataset (CI) The census-income dataset contains 40 demographic and employment related features and is used to predict income level, presented as a binary classification problem. Approximately 196,000 samples were used for training and almost 50,000 for validation. All of the features are presented in the form of structured data. We adjust it to the *US*setting in two ways: CI-A) by randomly splitting the set of features and assigning them to two sets $\mathcal{S}$ and $\mathcal{U}$, representing the structured and unstructured modalities, respectively; CI-B) by using backward elimination to identify the most informative features and assigning them to one of the sets ($\mathcal{S}$), while the rest of the features were assigned to the other ($\mathcal{U}$). The latter setting CI-B represents the case of one modality being much stronger correlated to the labels than the other.

Modified Imagenet Dataset (MI) We sample from Imagenet ($\mathcal{I}$) and construct $\mathcal{U}$ with two classes: $\mathcal{C}_0 = \{x_U | x_U \in \mathcal{I} \text{ and } x_U \text{ is a dog}\}$, which accounts for 47% of the total samples in the resulting dataset and $\mathcal{C}_1 = \{x_U | x_U \in \mathcal{I} \text{ and } x_U \text{ is a feline, primate, reptile or bird}\}$, which accounts for the remaining 53%. These classes were selected so that the dataset has a reasonable size and it is balanced. Approximately 313,000 samples were used for training and 12,000 for validation. We adjust it to the *US*setting as follows: we generate $\mathcal{S}$ by creating a structured sample $x_S \in \mathbb{R}^{500}$ for each image x_U in $\mathcal{U}$ such that, for a fixed $w \in \mathbb{R}^{500}$ we have that $w^T x_S > 0$ if $x_U \in \mathcal{C}_0$ and $w^T x_S < 0$ otherwise. Since there are many such x_S, we select one at random. Finally, we randomly switch 9% of the labels in $\mathcal{S} \cup \mathcal{U}$ which provides a balance between further introducing noise to the data, while keeping more than 90% of the dataset's deterministic label assignment unchanged.

Forest Covertype Dataset (CT) We construct $\mathcal{S}$ with the 3 most represented classes in the highly imbalanced Forest Covertype dataset, resulting in approximately 424,000 and 53,000 training and validation samples, respectively. Conversely, we adjust it to the $\mathcal{US}$setting by generating an image $x_U \in \mathbb{R}^{128\times128}$ for each structured sample x_S in $\mathcal{S}$ as follows. Each x_U consists of a white background and a random number in $\{1, ..., 10\}$ of randomly positioned:

- mixed type shapes if $x_s \in \mathcal{C}_0$,
- triangles if $x_s \in \mathcal{C}_1$,
- rectangles if $x_s \in \mathcal{C}_2$.

The shapes were generated using scikit-image (van der Walt et al., 2014) with maximum bounding box sizes of 128 pixels and minimums of 10, 20, and 15 pixels, respectively. Again, we randomly switch 9% of the labels in $\mathcal{S}\cup\mathcal{U}$ to introduce noise, while keeping more than 90% of the dataset's labels unchanged.

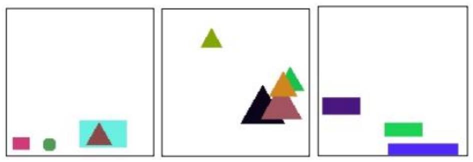

(a) Class $\mathcal{C}_0$ (b) Class $\mathcal{C}_1$ (c) Class $\mathcal{C}_2$

Figure 2: Examples of generated images

Real-World Multimodal Dataset (RW) For this experiment, we use a proprietary dataset with both structured and unstructured data inputs, which allows us to test our models in a real-world $\mathcal{US}$setting. The dataset constitutes a binary classification problem with two data sources: one is presented in the form of images $\mathcal{U}$ and the other as structured data $\mathcal{S}$ where GBM works very well. Tens of thousands of samples were curated for training and validation, with each structured data sample containing approximately 100 features.

4.2 Implementation and hyperparameters

The experiments were implemented in Python and ran using GeForce RTX 2080 Ti GPU and Intel(R) Xeon(R) Silver 4214 CPU @ 2.20GHz for all datasets except RW, for which Tesla V100 GPU and Intel Xeon CPU E5-2697 v4 @2.30Hz were used. For the BFV+DNN models, scikit-learn's GradientBoosterClassifiers (Pedregosa et al., 2011) are trained and used to generate the BFVs. We employ Bayesian Optimization (BO) (Snoek et al., 2012) with 10 random exploration points and 20 iterations to find ϵ^* and δ^* in steps (4) of Algorithm 1 and (3.2.2) of Algorithm 2. The tracked metric is F1 for all datasets, except for RW, where accuracy is used.

The dataset-specific hyperparameters used for BFV+DNN and two-weak-learner experiments can be found in Tables 1 and 2, respectively. These hyperparameters were selected as follows. For DNNs, we used a fully connected layer with k neurons (FCk) or two fully connected layers with k_1 and k_2 layers (FCk_1+k_2), where the number of layers and neurons were chosen based on the number of samples and features of each dataset. Regarding image datasets, VGG16(Simonyan and Zisserman, 2015) and Resnet50(He et al., 2016) convolutional architectures were compared and the best one was selected. For optimizers, we chose the best performing between RMSPROP and stochastic gradient descent with learning rate 10^{-j}, $j \in \{3, 4, 5\}$ (SGD/LR). Matrix multiplication and embedding concatenation were compared in order to select the fusion method for each dataset. The number of BFV trees is the best in $\{1000, 1500, 2000, 3000\}$, while the maximum tree depth is the best in $\{3, 4, 5, 6\}$. Values N, N_1, and N_2 vary according to the number of iterations each dataset took until convergence. Batch sizes were chosen based on the number of input features and pretraining was used for datasets with image data.

4.3 Experimental Results

Model Comparison

In Table 3, we summarize the results of the conducted experiments. For each dataset, we compare the performance of the boosted-feature vector DNN (BFV$_\mathcal{S}$ +DNN$_\mathcal{U}$), the two-weak-learner-gradient-boosted model with BO (2WL), and the two-weak-learner-second-order-gradient-boosted model with BO (2WL2O). Additionally, we analyze the impact of finding optimal steps ϵ^* and δ^* for the two-weak-learner models and conduct the same experiments with fixed $\epsilon^* = \delta^* = 0.1$ (2WL_Fix and 2WL2O_Fix). The value of 0.1 was chosen following the same reasoning as before regarding default hyperparameters used in GBM implementations. All results are given as percentage of relative improvement over the chosen baseline (see Figure 1 for reference). To account for randomization, we ran 5 identical experiments of each BFV+DNN and report their average. Their coefficients of variation were smaller than 0.005, 0.005, 0.0001, 0.0001, and 0.001 for CI-A, CI-B, CT, MI, and RW datasets, respectively. Given the low variability shown by the DNN-based model and the additional computational time needed to run two-

Table 1: Boosted-feature-vector Deep Learning Network Hyperparameters

	CI-A	**CI-B**	**CT**	**MI**	**RW**
DNN1	FC32	FC 100+50	VGG16+FC1024+200	Resnet50+FC1024+256	VGG16
DNN2	FC256+32	FC100+50	FC1024+200	FC256	FC512
DNN3	FC256+32	FC25	FC64	FC128	FC16
Fusion	Product	Concat	Concat	Concat	Product
Optimizer	SGD/0.001	RMSPROP	SGD/0.00001	SGD/0.00001	SGD/0.001
Batch size	128	128	32	32	16
BFV Trees	2000	1500	3000	3000	2000

Table 2: Two-Weak-Learner-Gradient-Boosting Hyperparameters

	CI-A	**CI-B**	**CT**	**MI**	**RW**
DNN	FC100+50	FC100+50	VGG16+FC1024	Resnet50+FC64	VGG16
DT max depth	3	3	6	3	5
Optimizer	RMSPROP	RMSPROP	SGD/0.01	SGD/0.0001	SGD/0.0001
Batch size	512	128	32	32	16
N , N_1, N_2	2000,2100,1	3500, 750,1	135,25,1	200,10,1	20,70,1

Table 3: Best relative performance per dataset

	% of Relative Metric Improvement				
Model	**CI-A**	**CI-B**	**CT**	**MI**	**RW**
BFV$_\mathcal{S}$ +DNN$_\mathcal{U}$	1.87	**4.70**	0.08	**0.34**	0.11
2WL	3.50	0.14	-0.37	-0.09	-0.60
2WL_Fix	0.68	-2.10	-0.36	-3.45	-2.37
2WL2O	**3.97**	0.25	**0.10**	0.13	**0.41**
2WL2O_Fix	-7.87	-19.85	-0.23	-0.14	-5.37

weak-learner experiments, we report a single instance for each boosting model.

In general, BFV$_\mathcal{S}$ +DNN$_\mathcal{U}$ and 2WL2O models exhibit the best performance. The results in the different considered datasets help to differentiate the individual strengths of each of our proposed models.

For datasets CI-A and CI-B, we observe that given the underlying structured nature of the data, the BFVs used for BFV$_\mathcal{S}$ +DNN$_\mathcal{U}$ are responsible for a large portion of the predictive power, making this model perform significantly better than the baseline. This behavior is notably exhibited in CI-B, were the most informative features have been grouped in $\mathcal{S}$ and used to generate the BFVs. On the other hand, due to the random variable split in CI-A, not all of the most informative features are used for generating the BFV, which is reflected in the performance gap between both datasets for this model and in the two-weak-learner models outperforming BFV+DNNs for CI-A.

In datasets CT, MI and RW, which contain both structured and unstructured data, we observe closer gaps between the performances of BFV$_\mathcal{S}$ +DNN$_\mathcal{U}$ and 2WL2O, but quite large improvements of these over 2WL in all cases. BFV$_\mathcal{S}$ +DNN$_\mathcal{U}$ achieves the best performance for CI-B and MI. On the other hand, 2WL2O outperforms all models for CI-A, CT, and RW. The predictive power and complexity of each data input vs the other appears to play an important role both in the best model's performance and in the usefulness of the second order approximation. We further observe this in Figures 3 and 4. In Figure 3, we compare the first and second order weak learners performance. Given that we have sufficient time for convergence, the second order model outperforms the first order in all our experiments. However, we observe that for some datasets such as MI and RW, the performance of two-weak-learner models may abruptly drop after reaching its maximum. To further explore this, we compare 2WL and 2WL2O with their corresponding one weak learner models 1WL-DT, trained on $\mathcal{S}$, and 1WL-CNN, trained on $\mathcal{U}$. Accordingly, this comparison is conducted on datasets that have both structured and image data availables (MI, CT, and RW) and is shown in Figure 4. As can be seen, the drop in performance observed for the two-weak-learner models in datasets MI and RW is consistently observed in their corresponding 1WL-CNN models. The unstructured data weak learner seems to be driving the two-weak-learner models for the aforementioned datasets. Interestingly, the deterioration is shown only in one of the two-weak-learner models per dataset, possibly as a result of conducting independent optimizations to find ϵ^* and δ^* in step (4) of Algorithm 1 and (3.2.2)

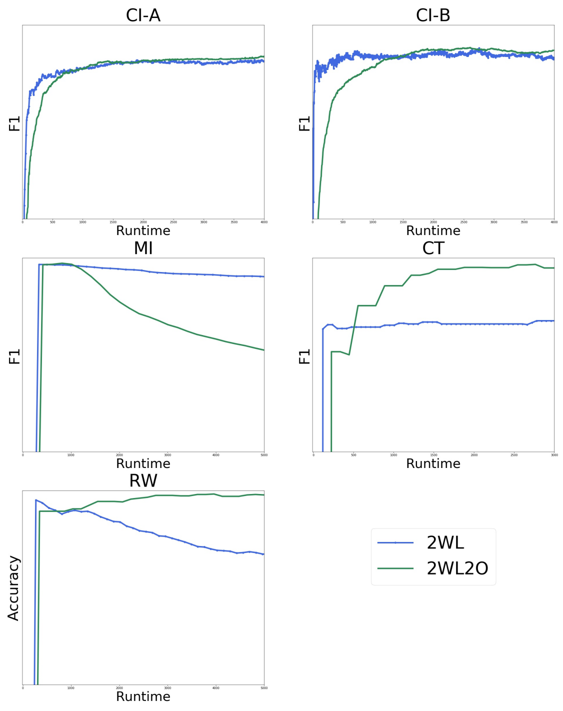


Figure 3: Two-weak-learners performance vs runtime

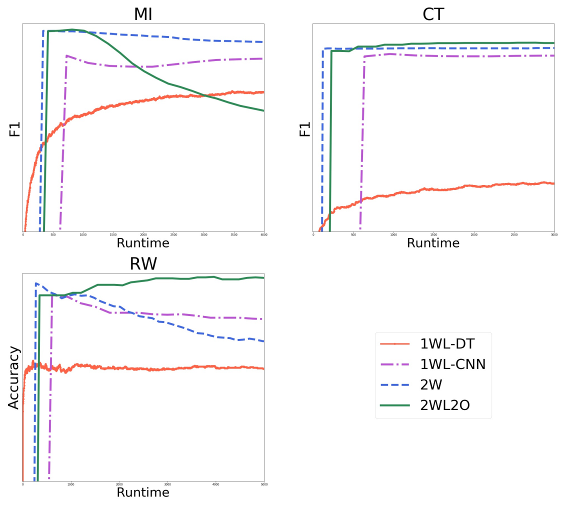


Figure 4: One- and two-weak-learners performance vs runtime

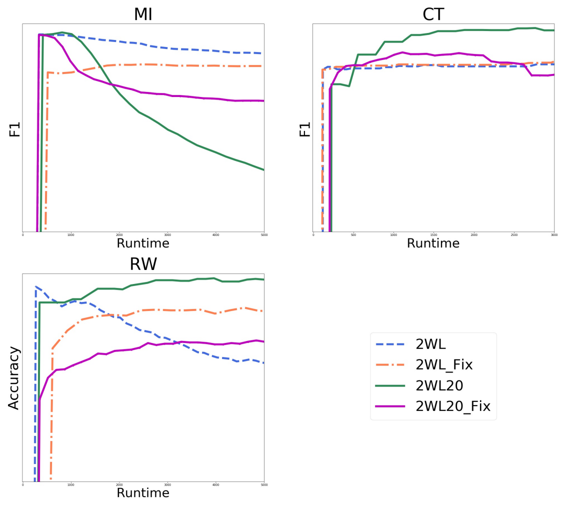


Figure 5: Fixed vs Optimized LR

of Algorithm 2. In Figure 5, we compare 2WL and 2WL2D with their corresponding 2WL_Fix and 2WL2_Fix runs. In all of our experiments, fixing the values of ϵ^* and δ^* results in a significant drop in performance, further emphasizing the key role played by the Bayes optimization steps and careful choice of learning rates ϵ and δ.

As a final remark, an important factor to consider when evaluating and comparing the proposed models is computational time. In Algorithms 1 and 2 for the two-weak-learner frameworks, we have that one DNN is trained per iteration for the first order approximation, yielding a total of N trained DNNs per run. For the second order approximation, N_2 DNNs are trained per iteration, for a total of $N_1 N_2$ DNNs per run. On the other hand, we have that the BFV+DNN model trains a single DNN (plus a previously trained GBM). Hence, BFV+DNN has a clear advantage in terms of runtime, whereas the two-weak-learner-boosted frameworks can be leveraged to improve performance when time does not pose a hard constraint.

5 Conclusion

Traditionally, boosted models have shown stellar performance when dealing with structured data, whereas DNNs excel in unstructured data problems. However, in many real-world applications both structured and unstructured data are available. In this paper, we presented two frameworks that address these scenario. The proposed models are compared to a standard baseline model and demonstrate strong results, outperforming the baseline approach when data is presented as a combination of these two data sources.

References

References

Balas, V. E., Roy, S. S., Sharma, D., and Samui, P. (2019). *Handbook of deep learning applications*, volume 136. Springer.

Blackard, J. A. (1999). UCI machine learning repository. `https://archive.ics.uci.edu/ml/machine-learning-databases/covtype/covtype.info`.

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth and Brooks.

Caruana, R., Karampatziakis, N., and Yessenalina, A. (2008). An empirical evaluation of supervised learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, pages 96–103.

Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 161–168.

Chen, M., Hao, Y., Hwang, K., Wang, L., and Wang, L. (2017). Disease prediction by machine learning over big data from healthcare communities. *IEEE Access*, volume 5, pages 8869-8879.

Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.

Dua, D. and Graff, C. (2017). UCI machine learning repository. `https://archive.ics.uci.edu/ml/datasets/Census-Income+%28KDD%29`.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, volume 55, pages 119–139.

Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, volume 28, pages 337 – 407.

Friedman, J. H. (2000). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, volume 29, pages 1189–1232.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

Goyal, A., Morvant, E., Germain, P., and Amini, M. (2018). Multiview boosting by controlling the diversity and the accuracy of view-specific voters. *arXiv preprint arXiv:1808.05784*.

Graham, B. (2015). Kaggle diabetic retinopathy detection competition report. Technical report, University of Warwick. `https://kaggle-forum-message-attachments.storage.googleapis.com/88655/2795/competitionreport.pdf`.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., Guadarrama, S., and Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems 30*, pages 3146–3154.

Koço, S., Capponi, C., and Béchet, F. (2012). Applying multiview learning algorithms to human-human conversation classification. In *Conference of the International Speech Communication Association*, pages 2322–2325.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*, pages 1097–1105.

Lahiri, A., Paria, B., and Biswas, P. K. (2018). Forward stagewise additive model for collaborative multiview boosting. *IEEE Transactions on Neural Networks and Learning Systems*, volume 29, pages 470–485.

Li, P., Qin, Z., Wang, X., and Metzler, D. (2019). Combining decision trees and neural networks for learning-to-rank in personal search. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2032–2040.

Lloyd, J. R. (2014). GEFCom2012 hierarchical load forecasting: Gradient boosting machines and Gaussian processes. *International Journal of Forecasting*, volume 30, pages 369–374.

Mangal, A. and Kumar, N. (2017). Using big data to enhance the Bosch production line performance: A Kaggle c
arXiv preprint arXiv:1701.00705.

Mayr, A., Binder, H., Gefeller, O., and M, S. (2014). The evolution of boosting algorithms - from machine learning to statistical modelling. In *Methods of Information in Medicine 53*, pages 419–427.

Moghimi, M., Belongie, S., Saberian, M., Yang, J., Vasconcelos, N., and Li, L.-J. (2016). Boosted convolutional neural networks. In *Proceedings of the British Machine Vision Conference*, pages 24.1–24.13.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, volume 12, pages 2825–2830.

Peng, J., Aved, A. J., Seetharaman, G., and Palaniappan, K. (2018). Multiview boosting with information propagation for classification. *IEEE Transactions on Neural Networks and Learning System.*, volume 29, pages 657–669.

Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788.

Ridgeway, G. (1999). The state of boosting. In *Computing Science and Statistics*, volume 31, pages 172–181.

Saberian, M. J., Masnadi-Shirazi, H., and Vasconcelos, N. (2011). Taylorboost: First and second-order boosting algorithms with explicit margin control. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2929–2934.

Saberian, M. J. and Vasconcelos, N. (2011). Multiclass boosting: Theory and algorithms. In *Advances in Neural Information Processing Systems 24*, pages 2124–2132.

Schwenk, H. and Bengio, Y. (2000). Boosting neural networks. *Neural Computation*, volume 12, pages 1869–1887.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations*.

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2951–2959.

Taieb, S. B. and Hyndman, R. J. (2013). A gradient boosting approach to the kaggle load forecasting competition. *International Journal of Forecasting*.

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T., and the scikit-image contributors (2014). Scikit-image: image processing in Python. *PeerJ*, volume 2, pages e453.

Zou, H., Xu, K., and Li, J. (2017). The YouTube-8M Kaggle Competition: Challenges and methods. *arXiv preprint arXiv:1706.09274*.