Deep Boosting vs AdaBoosting: Theoretical Foundations, Computational Efficiency, and Performance Evaluation

Honglai Peng and Xiangyu Wang

ABSTRACT

This proposal outlines a project with the intention of conducting a rigorous and detailed comparison between AdaBoost [2], and Deep Boosting algorithms[1], aiming to understand their theoretical principles, evaluate their computational demands, and empirically assess their performance across datasets. Through this comparative analysis, we seek to uncover the practical trade-offs between the two approaches, providing clear guidelines on their applicability and effectiveness in different machine learning scenarios.

1 INTRODUCTION

Boosting methods significantly enhance the performance by combining multiple predictors which are based on favorable learning guarantees, including AdaBoosting and Deep Boosting. Let $f(x)$ be the prediction function, y be the true output, and $L(y, f(x))$ be a loss function. The goal of boosting is to minimize the expected value of the loss function over the distribution of the data, represented as:

$$\mathbb{E}_{(x,y)\sim D}[L(y, f(x))]$$

Particularly, AdaBoost benefits from the performance guarantees in terms of the margins and training samples. The main idea behind AdaBoosting is to iteratively add functions to minimize the residuals of the previous iterations.

For Deep Boosting, the focus shifts to combining more complex classifiers and balance too complex base classifiers and overfitting issues, so that it can succeed in achieving good performance.

2 MOTIVATION

While the decision between AdaBoosting and Deep Boosting is crucial in machine learning, clear guidelines on when to choose one over the other are not well established. Then, the critical questions arise: Under what conditions does AdaBoosting encounter limitations, and how does Deep Boosting complement or surpass its capabilities? Conversely, in which scenarios does AdaBoosting maintain its superiority, proving itself as the more reliable option? In this project, we aim to address this gap, providing a concise and practical comparative analysis. Through this analysis, we intend to identify scenarios where one algorithm may be more advantageous than the other.

3 THEORETICAL COMPONENTS

In this section, we will explore the mathematical frameworks and algorithmic steps unique to Deep Boosting and AdaBoosting, highlighting their differences and similarities.

Prior to delving into the core principles of boosting, it is essential to establish a foundational understanding of a *weak learner* [3], as defined in Definition 1.

Definition 1 (Weak learning). *A concept class C is said to be weakly PAC-learnable if there exists an algorithm A, $\gamma > 0$, and a polynomial function $poly(\cdot,\cdot,\cdot)$ such that for any $\delta > 0$, for all distributions $\mathcal{D}$ on $\mathcal{X}$ and for any target concept $c \in C$, the following holds for any sample size $m \geq poly(1/\delta, n, size(c))$:*

$$\mathbb{P}_{S\sim\mathcal{D}^m}\left[R(h_S) \leq \frac{1}{2} - \gamma\right] \geq 1 - \delta,$$

where h_S is the hypothesis returned by algorithm A when trained on sample S. When such an algorithm A exists, it is called a weak learning algorithm for C or a weak learner. The hypotheses returned by a weak learning algorithm are called base classifiers.

The key ides of boosting techniques is to use a weak learning algorithm to build a strong learner, which is an accurate PAC-learning algorithm.

3.1 AdaBoosting

3.1.1 ***Main Idea***.

To our best knowledge, AdaBoost is one of the most successful boosting techniques. The main idea behind the design of AdaBoost is that it iteratively trains base classifiers from $\mathcal{H}$ on the training data, like *boosting stumps*, assigning higher weights to misclassified instances in each round, thereby compelling subsequent

classifiers to focus more on the challenging cases. These classifiers are then weighted according to their accuracy and aggregated to form the final model. The result is a powerful PAC-learning algorithm that excels in classification tasks by emphasizing difficult-to-classify instances and combining the strengths of simple decision rules.

3.1.2 *Margin-based Rademacher Complexity.*

To understand the generalization of AdaBoost on unseen data, we always begin by exploring the Rademacher complexity of convex linear combinations of hypotheses. For any given set of hypotheses, denoted by $\mathcal{H}$, consisting of real-valued functions, we denote as $conv(\mathcal{H})$.

The Lemma 1, shows that the empirical Rademacher complexity of $conv(\mathcal{H})$ coincides with that of $\mathcal{H}$ which is the basis for the margin-based analysis.

Lemma 1. *Let $\mathcal{H}$ be a set of functions mapping from X to $\mathbb{R}$. Then, for any sample S, we have*

$$\hat{\mathcal{R}}_S(conv(\mathcal{H})) = \hat{\mathcal{R}}_S(\mathcal{H}).$$

As depicted in [3], we have ensemble Rademacher margin bound as shown in Definition 2:

Definition 2. *[Ensemble Rademacher margin bound] Let $\mathcal{H}$ denote a set of real-valued functions. Fix $\rho > 0$. Then, for any $\delta > 0$, with probability at least $1 - \delta$, each of the following holds for all $h \in conv(\mathcal{H})$:*

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{2}{\rho}\hat{\mathcal{R}}_m(\mathcal{H}) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}$$

$$R(h) \leq \hat{R}_{S,\rho}(h) + \frac{2}{\rho}\hat{\mathcal{R}}_m(\mathcal{H}) + 3\sqrt{\frac{\log \frac{2}{\delta}}{2m}}$$

The function f denotes the classifier returned by AdaBoost after T rounds of boosting when trained on sample S as: $f = \sum_{t=1}^{T} \alpha_t h_t$. However, since the inequalities in Definition 2 are not convex combination of base hypotheses which cannot be directly applied to the function f, so its normalized version $\bar{f}$ is proposed:

$$\bar{f} = \frac{\sum_{t=1}^{T} \alpha_t h_t}{\|\alpha\|_1} \in \text{conv}(\mathcal{H}).$$

Then, for any $\delta > 0$, the following holds with probability at least $1 - \delta$:

$$R(f) \leq \hat{R}_{S,\rho}(\bar{f}) + \frac{2}{\rho}\hat{\mathcal{R}}_m(\mathcal{H}) + \sqrt{\frac{\log \frac{1}{\delta}}{2m}}.$$

These bounds can be generalized to hold uniformly for all $\rho \in (0, 1]$, with the price: $O\left(\sqrt{\frac{\log \log_2 \frac{2}{\delta}}{m}}\right)$

3.1.3 *Algorithm.*

The pseudocode of AdaBoost Algorithm is shown in Algorithm 1.

- **Step 2-4**. Weight distribution $D_1(i)$ is created, where each of the m training examples is given an equal weight of $\frac{1}{m}$.
- **Step 5-13**. The algorithm proceeds in T rounds, where in each round t:
  - A weak classifier h_t is selected from a family of classifiers $\mathcal{H}$ with the error ϵ_t.
  - A weight α_t is assigned to the classifier h_t, which is computed based on the error ϵ_t. The weight reflects the importance or confidence of the classifier in the final decision.
  - A normalization factor Z_t is calculated to ensure that the sum of the updated weights of the training examples remains 1. This factor is based on the error ϵ_t.
  - The weight distribution D_t is updated to D_{t+1} for the next round. The weights of the incorrectly classified examples are increased, and those of the correctly classified examples are decreased, which is determined by the classifier's weight α_t, the actual label y_i, and the predicted label $h_t(x_i)$.
- **Step 14**. After all T rounds, the algorithm combines the weak classifiers h_t, each weighted by its corresponding α_t, to form a strong final classifier f which is a linear combination of the weak classifiers.

3.1.4 *Pros and Cons.*

In this section, we will summarize the advantages and challenges in AdaBoosting.

Advantages.

- Simplicity. AdaBoost is straightforward to implement and its methodology is simple to understand.
- Favorable Time Complexity. With decision stumps, its time complexity per round of boosting is $O(mN)$, which is generally efficient. However, this can slow down if the feature space dimension N is very large.
- Rich Theoretical Analysis. AdaBoost benefits from extensive theoretical research, providing a solid

Algorithm 1 AdaBoost Algorithm

$S \leftarrow ((x_1, y_1), \ldots, (x_m, y_m))$
for $i \leftarrow 1$ **to** m **do**
    $D_1(i) \leftarrow \frac{1}{m}$
end for
for $t \leftarrow 1$ **to** T **do**
    $h_t \leftarrow$ base classifier in $\mathcal{H}$ with small error ϵ_t
    $\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
    $Z_t \leftarrow 2\sqrt{\epsilon_t(1-\epsilon_t)}$ // normalization factor
    for $i \leftarrow 1$ **to** m **do**
        $D_{t+1}(i) \leftarrow \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$
    end for
    $f \leftarrow \sum_{t=1}^{T} \alpha_t h_t$
end for
return f

understanding of its generalization properties, as we discussed above.

- Outlier Detection. In the context of noise, AdaBoost behavior can be utilized to identify outliers, as examples that are hard to classify tend to gain larger weights in successive rounds.

Challenges.

- Parameter Selection. Choosing the right number of boosting rounds T and the appropriate base classifiers is crucial and can be challenging. As this process repeats over multiple rounds, the algorithm increasingly focuses on the samples that are harder to classify which can lead to an overemphasis on very hard or noisy samples. And if the base classifier is complicated, it will lead to overfitting and reduce its overall accuracy on general data.
- High-Dimensional Feature Spaces. The risk of overfitting increases because the algorithm may start to fit to noise or irrelevant features in the data. Although AdaBoost has some inherent mechanisms to prevent overfitting, these may not be sufficient in very high-dimensional spaces.
- Vulnerability to Noise. AdaBoost is particularly susceptible to noise in the data. This sensitivity increases with more boosting rounds, leading to potential misclassification and reduced accuracy. Also, empirical studies show that AdaBoost accuracy suffers significantly in the presence of uniform noise.

3.2 Deep Boosting

As we discussed in 3.1, AdaBoost combine weak learners from a base classifier hypothesis set $\mathcal{H}$ in which $\mathcal{H}$ is reduced to the so-called *boosting stumps*. Simple boosting stumps are not sufficient to achieve good performance on complicated tasks like image or speech processing, which consist of many high-dimensional features. Also, the learning bounds will become looser using too complex base classifier sets $\mathcal{H}$, which might lead to overfitting. As a result, deep boosting is proposed to balance too complex base classifiers and overfitting issues, so that it can succeed in achieving good performance.

3.2.1 *Main Idea.*

Assume that the set of base classifiers $\mathcal{H}$ can be decomposed as the union of p disjoint families $\mathcal{H}_1, \ldots, \mathcal{H}_p$ ordered by increasing complexity, where $\mathcal{H}_k$, $k \in [1, p]$, could be, for example, the set of decision trees of depth k.

The key idea behind the design of deep boosting is that an ensemble based on hypotheses drawn from $\mathcal{H}_1, \ldots, \mathcal{H}_p$ can achieve higher accuracy by making use of hypotheses drawn from $\mathcal{H}_k$ with large k if it allocates more weights to hypotheses drawn from $\mathcal{H}_k$ with a small k.

3.2.2 *Margin-based Rademacher Complexity.*

In [1], they consider base hypotheses taking arbitrary real values but assume that they can be selected from several distinct hypothesis sets $H_1, \ldots, H_p$ with $p \geq 1$ and present margin-based learning in terms of the Rademacher complexity of these sets, which admits an explicit dependency on the mixture coefficients defining the ensembles. The ensemble family is

$$\mathcal{F} = \operatorname{conv}\left(\bigcup_{k=1}^{p} H_k\right)$$

that is the family of functions f of the form, $f = \sum_{t=1}^{T} \alpha_t h_t$, where $\alpha = (\alpha_1, \ldots, \alpha_T)$ is in the simplex Δ and where, for each $t \in [1, T]$, h_t is in H_{k_t} for some $k_t \in [1, p]$.

Definition 3 gives a margin-based Rademacher complexity bound of deep boosting.

Definition 3. *Assume $p > 1$. Fix $\rho > 0$. Then, for any $\delta \geq 0$, with probability at least $1 - \delta$ over the choice of a sample S of size m drawn i.i.d. according to $\mathcal{D}^m$, the*

following inequality holds for all $f = \sum_{t=1}^{T} \alpha_t h_t \in \mathcal{F}$*:*

$$R(f) \leq \hat{R}_{S,\rho}(f) + \frac{4}{\rho} \sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(H_{k_t}) + \frac{2}{\rho}\sqrt{\frac{\log p}{m}} + \sqrt{\frac{4}{\rho^2}\log\left(\frac{\rho^2 m}{\log p}\right)\frac{\log p}{m} + \frac{\log^2\left(\frac{2}{\delta}\right)}{2m}}.$$

Thus,

$$R(f) \leq \hat{R}_{S,\rho}(f) + \frac{4}{\rho} \sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(H_{k_t}) + C(m,p)$$

$$\textit{with } C(m,p) = O\left(\sqrt{\frac{\log p}{\rho^2 m}\log\left[\frac{\rho^2 m}{\log p}\right]}\right).$$

These bounds can also be generalized to hold uniformly for all $\rho \in (0,1]$, with the price: $O\left(\sqrt{\frac{\log\log_2 \frac{2}{\delta}}{m}}\right)$ In addition, since the complexity term of the bound would be the Rademacher complexity of the family of hypotheses $\mathcal{F} = \text{conv}\left(\bigcup_{k=1}^{p} H_k\right)$, so the standard Rademacher complexity analysis is not applicable. However, since $\mathfrak{R}_m(\text{conv}(\bigcup_{k=1}^{p} H_k)) = \mathfrak{R}_m(\bigcup_{k=1}^{p} H_k)$, the following lower bound holds for any choice of the non-negative mixtures weights α_t summing to one:

$$\mathfrak{R}_m(\mathcal{F}) \geq \max_{k=1}^{m} \mathfrak{R}_m(H_k) \geq \sum_{t=1}^{T} \alpha_t \mathfrak{R}_m(H_{k_t}).$$

Therefore, definition 3 gives a finer learning bound. Also, we can tell that, for $\mathbf{p} = 1$, that is the case of single hypothesis set, the analysis coincides with the standard ensemble margin bounds, as discussed in 3.1.

3.2.3 ***Algorithm***.

The pseudocode of Deep Boosting Algorithm is shown in Algorithm 2.

- **Step 2-4.** Weight distribution $D_1(i)$ is created, where each of the m training examples is given an equal weight of $\frac{1}{m}$.
- **Step 5-13.** Calculate a direction for each hypothesis and select the one with the maximal $|d_j|$, DeepBoost decides which base learner to focus on next. This direction is based on the gradient of the loss with respect to the weights, reflecting how much impact each learner has on the overall performance.
- **Step 14-22.** Set the step size η_t that aims to minimize the loss. It is calculated using a closed-form expression when certain conditions related to the margin are met, which ties the minimization directly to improving the confidence of the classification, thereby contributing to the ensemble's generalization ability.
- **Step 23.** Update the weights α_t of the hypotheses. The algorithm assigns higher weights to more influential base learners in the ensemble.
- **Step 24-27.** Re-weight the data points. The reweighting of the data points adjusts the focus of the learning process, emphasizing misclassified data which helps to prevent overfitting by ensuring the ensemble does not become too confident on the training data and remains robust to variations.
- **Step 29-30.** Return the final model by DeepBoost which is an ensemble of weighted base learners.

3.2.4 ***Pros and Cons***.

In this section, we will summarize the advantages and challenges in Deep Boosting.

Advantages.

- Handle Complex Hypothesis Sets. DeepBoost can manage complex base learners and control their influence on the final model through the mixture coefficients, allowing for a robust ensemble.
- Improved Generalization. By using margin-based updates and a convex surrogate loss function, DeepBoost can potentially achieve finer generalization bounds than using standard Rademacher complexity.
- Regularization. DeepBoost includes a regularization term in its objective function, which helps to prevent overfitting which is particularly useful when dealing with noisy data or data with many high-dimensional features.

Challenges.

- Computational Complexity. The iterative nature of Deep Boosting, along with the need to calculate margins and update distributions, can make it computationally intensive, especially with large datasets or complex base learners.
- Parameter Tuning. The performance of DeepBoost is dependent on the correct tuning of its hyperparameters, such as the regularization parameters and the number of boosting rounds, which can be challenging.

Algorithm 2 DEEPBOOST algorithm

$S \leftarrow ((x_1, y_1), \ldots, (x_m, y_m))$
for $i \leftarrow 1$ **to** m **do**
  $D_1(i) \leftarrow \frac{1}{m}$
end for
for $t \leftarrow 1$ **to** T **do**
  for $j \leftarrow 1$ **to** N **do**
    if $(\alpha_{t-1,j} \neq 0)$ **then**
      $d_j \leftarrow \left(e_{t,j} - \frac{1}{2}\right) + \operatorname{sgn}(\alpha_{t-1,j})\frac{\Lambda_j m}{2S_t}$
    else if $\left|e_{t,j} - \frac{1}{2}\right| \leq \frac{\Lambda_j m}{2S_t}$ **then**
      $d_j \leftarrow 0$
    else
      $d_j \leftarrow \left(e_{t,j} - \frac{1}{2}\right) - \operatorname{sgn}\left(e_{t,j} - \frac{1}{2}\right)\frac{\Lambda_j m}{2S_t}$
    end if
    $k \leftarrow \arg\max_{j \in [1,N]} |d_j|$
    $\epsilon_t \leftarrow \epsilon_{t,k}$
    if $|(1-\epsilon_t)e^{\alpha_{t-1,k}} - \epsilon_t e^{-\alpha_{t-1,k}}| \leq \frac{\Lambda_k m}{S_t}$ **then**
      $\eta_t \leftarrow -\alpha_{t-1,k}$
    else if $|(1-\epsilon_t)e^{\alpha_{t-1,k}} - \epsilon_t e^{-\alpha_{t-1,k}}| > \frac{\Lambda_k m}{S_t}$ **then**
      $\eta_t \leftarrow \log\left(-\frac{\Lambda_k m}{2\epsilon_t S_t} + \sqrt{\left(\frac{\Lambda_k m}{2\epsilon_t S_t}\right)^2 + \frac{1-\epsilon_t}{\epsilon_t}}\right)$
    else
      $\eta_t \leftarrow \log\left(\frac{\Lambda_k m}{2\epsilon_t S_t} + \sqrt{\left(\frac{\Lambda_k m}{2\epsilon_t S_t}\right)^2 + \frac{1-\epsilon_t}{\epsilon_t}}\right)$
    end if
    $\alpha_t \leftarrow \alpha_{t-1} + \eta_t e_k$
    $S_{t+1} \leftarrow \sum_{i=1}^{m} \Phi'(1-y_i) \sum_{j=1}^{N} \alpha_{t,j} h_j(x_i)$
    for $i \leftarrow 1$ **to** m **do**
      $D_{t+1}(i) \leftarrow \frac{\Phi'(1-y_i)\sum_{j=1}^{N} \alpha_{t,j} h_j(x_i)}{S_{t+1}}$
    end for
  end for
  $f \leftarrow \sum_{j=1}^{N} \alpha_{T,j} h_j$
  return f
end for

- Potential Suboptimal Steps. The step size in the update rule is critical for the algorithm's performance. If the step size is not optimally chosen, the algorithm may take suboptimal steps, leading to bad performance.
- Risk of Overfitting with Many Rounds. Despite regularization, like other boosting methods, there's a potential risk of overfitting if the number of boosting rounds is too large.

4 COMPUTATION COMPONENTS

To assess the computational performance, we use the Receiver Operating Characteristic curve (ROC) and its corresponding Area Under the Curve (AUC) as key metrics. These are derived from the confusion matrix, which tracks True Positives (TP), False Positives (FP), False Negatives (FN), and True Negatives (TN). The ROC is plotted by setting the false positive rate (FPR)

$$FPR = \frac{FP}{TN + FP}, \; TPR = \frac{TP}{TP + FN}$$

against the true positive rate (TPR). The AUC, ranging between 0 and 1, measures a classifier's ability to distinguish between fraudulent and legitimate cases, with higher values indicating better performance.

4.1 Adaboost

In AdaBoost, we employ decision stumps — the most basic form of decision trees with a depth of one — as the weak learners for AdaBoost, due to decision stumps' simplicity and computational efficiency.

4.1.1 Computational Efficiency.

AdaBoost is evaluated by its quick training and evaluation time, with processing completed in 1.40 seconds for the "Fraudulent" dataset and 1.13 seconds for the "Bank Marketing" dataset. This efficiency is attributable to the use of decision stumps, which are computationally straightforward due to their simplicity.

4.1.2 Computational Accuracy.

The performance of AdaBoost is evaluated via the ROC curve, as we mentioned before. For the "Fraudulent" dataset, AdaBoost achieves an AUC score of 0.56, indicating a moderate ability to distinguish between classes. On the "Bank Marketing" dataset, the algorithm performs better, with an AUC of 0.85, suggesting a high degree of separability between classes. These results are visually illustrated in Fig. 1.

4.2 Deepboosting

In contrast to AdaBoost's decision stumps, DeepBoost utilizes decision trees of depth 4, allowing for the capture of more complex patterns in the data.

4.2.1 Computational Efficiency.

DeepBoost exhibits a longer training and evaluation time compared to AdaBoost, with durations of 196.68

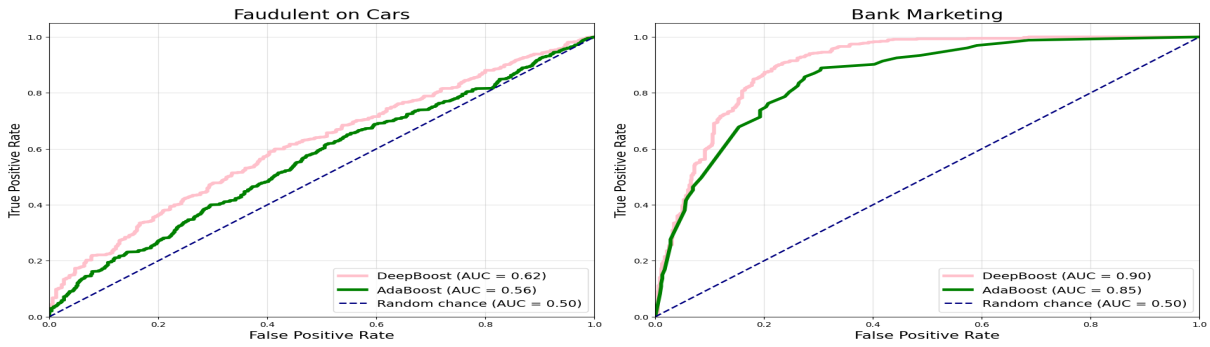

Figure 1: Comparison of Adaboost and DeepBoost

seconds for the "Fraudulent" dataset and 203.22 seconds for the "Bank Marketing" dataset.

4.2.2 Computational Accuracy.
DeepBoost's ROC curve analysis demonstrates its robust performance. For the "Fraudulent" dataset, it achieves an AUC score of 0.62, indicating a good ability to distinguish between classes. On the "Bank Marketing" dataset, DeepBoost performs well with an AUC of 0.9, indicating a strong capability to separate the target class, as shown in Fig. 1.

5 DISCUSSION AND CONCLUSION

5.1 Generalization Bounds

Both AdaBoost and DeepBoost provide interpretable generalization bounds using Rademacher complexity, which offers a theoretical guarantee of their performance on unseen data. These bounds give insight into how each algorithm might perform in practice, with the understanding that lower complexity generally leads to better generalization.

5.2 Computational Efficiency

AdaBoost is computationally more efficient due to the simplicity of decision stumps, making it a preferable choice for scenarios where computational resources are limited or when rapid model training is required. In contrast, DeepBoost's use of decision trees of depth 4 and the integration of neural network elements make it computationally intensive, which could be a limiting factor for large datasets or resource-constrained environments.

5.3 Computational Accuracy

DeepBoost typically outperforms AdaBoost in computational accuracy, as evidenced by higher AUC scores on both the "Fraudulent" and "Bank Marketing" datasets. However, this improved accuracy comes with the caveat that it is highly dependent on the choice of parameters. The tuning of DeepBoost's parameters can be a complex and sensitive task, potentially requiring extensive cross-validation to achieve optimal performance.

5.4 Scenario Analysis

AdaBoost may be the better in scenarios where interpretability is paramount, computational resources are not enough, or the data patterns are relatively simple. In such cases, the simplicity and speed of AdaBoost can be more valuable than the potential gains in accuracy from a more complex model.
DeepBoost is highly required in scenarios where data is complex, with intricate, nonlinear relationships and high-dimensional feature spaces, especially when class imbalance is present. Its sophisticated approach, capable of capturing deeper patterns and nuances in the data, makes it well-suited for challenging problems where traditional methods fall short.

5.5 Conclusion

In conclusion, while DeepBoost offers a compelling advantage in terms of accuracy, this comes at the cost of computational efficiency and ease of use. The choice between AdaBoost and DeepBoost should be decided by the specific requirements of the task at hand, balancing the need for accuracy against the available computational resources and the complexity of the data.

REFERENCES

[1] Corinna Cortes, Mehryar Mohri, and Umar Syed. 2014. Deep boosting. In *International conference on machine learning*. PMLR, 1179–1187.

[2] Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences* 55, 1 (1997), 119–139.

[3] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. *Foundations of machine learning*. MIT press.