GRADIENT BOOSTING NEURAL NETWORKS: GROWNET

Anonymous authors
Paper under double-blind review

ABSTRACT

A novel gradient boosting framework is proposed where shallow neural networks are employed as "weak learners". General loss functions are considered under this unified framework with specific examples presented for classification, regression and learning to rank. A fully corrective step is incorporated to remedy the pitfall of greedy function approximation of classic gradient boosting decision tree. The proposed model rendered outperforming results against state-of-the-art boosting methods in all three tasks on multiple datasets. An ablation study is performed to shed light on the effect of each model components and model hyperparameters.

1 INTRODUCTION

AI and machine learning pervade every aspect of modern life from email spam filtering and e-commerce, to financial security and medical diagnostics (McKinney et al., 2020; Simeon et al., 2019). Deep learning in particular has been one of the key innovations that has truly pushed the boundary of science beyond what was considered feasible (He et al., 2016; Goodfellow et al., 2014).

However, in spite of its seemingly limitless possibilities, both in theory as well as demonstrated practice, developing tailor-made deep neural networks for new application areas remains notoriously difficult because of its inherent complexity. Designing architectures for any given application requires immense dexterity and often a lot of luck. The lack of an established paradigm for creating an application-specific DNN presents significant challenges to practitioners, and often results in resorting to heuristics or even hacks.

In this paper, we attempt to rectify this situation by introducing a novel paradigm that builds neural networks from the ground up layer by layer. Specifically, we use the idea of gradient boosting (Friedman, 2001) which has a formidable reputation in machine learning for its capacity to incrementally build sophisticated models out of simpler components, that can successfully be applied to the most complex learning tasks. Popular GBDT frameworks like XGBoost (Chen & Guestrin, 2016), LightGBM (Ke et al., 2017) and CatBoost (Prokhorenkova et al., 2018) use decision trees as weak learners, and combine them using a gradient boosting framework to build complex models that are widely used in both academia and industry as a reliable workhorse for common tasks in a wide variety of domains.

In this paper, we combine the power of gradient boosting with the flexibility and versatility of neural networks and introduce a new modelling paradigm called GrowNet that can build up a Deep Neural Network (DNN) layer by layer. Instead of decision trees, we use shallow neural networks as our weak learners in a general gradient boosting framework that can be applied to a wide variety of tasks spanning classification, regression and ranking. We introduce further innovations like adding second order statistics to the training process, and also including a global corrective step that has been shown, both in theory (Zhang & Johnson, 2014) and in empirical evaluation, to provide performance lift and precise fine-tuning to the specific task at hand.

Our specific contributions are summarised below:

- We propose a novel approach to combine the power of gradient boosting to incrementally build complex deep neural networks out of shallow components. We introduce a versatile framework that can readily be adapted for a diverse range of machine learning tasks in a wide variety of domains.

- We develop an off-the-shelf optimization algorithm that is faster and easier to train than traditional deep neural networks. We introduce training innovations including second order statistics and global corrective steps that improve stability and allow finer-grained tuning of our models for specific tasks.
- We demonstrate the efficacy of our techniques using experimental evaluation, and show superior results on multiple real datasets in three different ML tasks: classification, regression and learning-to-rank.

2 Related Work

In this section, we briefly summarize the gradient boosting algorithms with decision trees and general boosting/ensemble methods for training neural nets.

Gradient Boosting Algorithms. Gradient Boosting Machine (Friedman, 2001) is a function estimation method using numerical optimization in the function space. Unlike parameter estimation, function approximation cannot be solved by traditional optimization methods in Euclidean space. Decision Trees are the most common functions (predictive learners) that are used in Gradient Boosting framework. In his seminal paper, Friedman (2001) proposed Gradient Boosting Decision Trees (GBDT) where decision trees are trained in sequence and each tree is modeled by fitting negative gradients. In recent years, there have been many implementations of GBDT in machine learning literature. Among these, Tyree et al. (2011) used GBDT to perform learning to rank, Friedman et al. (2000) did classification and Chen & Guestrin (2016); Ke et al. (2017) generalized GBDT for multi-tasking purposes. In particular, scalable framework of Chen & Guestrin (2016) made it possible for data scientists to achieve state-of-the-art results on various industry related machine learning problems. For that reason, we take XGBoost (Chen & Guestrin, 2016) as our baseline. Unlike these GBDT methods, we propose gradient boosting neural network where we train gradient boosting with shallow neural nets. Using neural nets as base learners also gives our method an edge over GBDT models, where we can correct each previous model after adding the new one, referred to as "corrective step", in addition to the ability to propagate information from the previous predictors to the next ones.

Boosted Neural Nets. Although weak learners, like decision trees, are popular in boosting and ensemble methods, there have been a substantial work done on combining neural nets with boosting/ensemble methods for better performance over single large/deep neural networks. The idea of considering shallow neural nets as weak learners and constructively combining them started with Fahlman & Lebiere (1990). In their pioneering work, fully connected, multi-layer perceptrons are trained in a layer-by-layer fashion and added to get a cascade-structured neural net. Their model is not exactly a boosting model as the final model is a single, multi-layer neural network.

In 1990's, ensemble of neural networks got popular as ensemble methods helped to significantly improve the generalization ability of neural nets. Nevertheless, these methods were simply either majority voting (Hansen & Salamon, 1990) for classification tasks, simple averaging (Opitz & Shavlik, 1996) or weighted averaging (Perrone & Cooper, 1993) for regression tasks. After the introduction of adaptive boosting ($Adaboost$) algorithm (Freund, 1995), Schwenk & Bengio (1997) investigated boosting with multi-layer neural networks for a character recognition task and achieved a remarkable performance improvement. They extended the work to traditional machine learning tasks with variations of Adaboost methods where different weighting schemes are explored (Schwenk & Bengio, 2000). The adaptive boosting can be seen as a specific version of the gradient boosting algorithm where a simple exponential loss function is used (Friedman et al., 2000).

In early 2000's, Hinton et al. (2006) introduced greedy layer-wise unsupervised training for Deep Belief Nets (DBN). DBN is built upon a layer at a time by utilizing Gibbs sampling to obtain the estimator of the gradient on the log-likelihood of Restricted Boltzmann Machines (RBM) in each layer. Bengio et al. (2007) expounded this work for continuous inputs and explained its success on attaining high quality features from image data. They concluded that unsupervised training helped model training by initializing RBM weights in a region close to a good local minimum.

Recently, AdaNet (Cortes et al., 2017) was proposed to adaptively built Neural Network (NN) layer by layer from a singe layer NN to perform image classification task. Beside learning network weights, AdaNet adjusts the network structure and its growth procedure is reinforced by a theo-

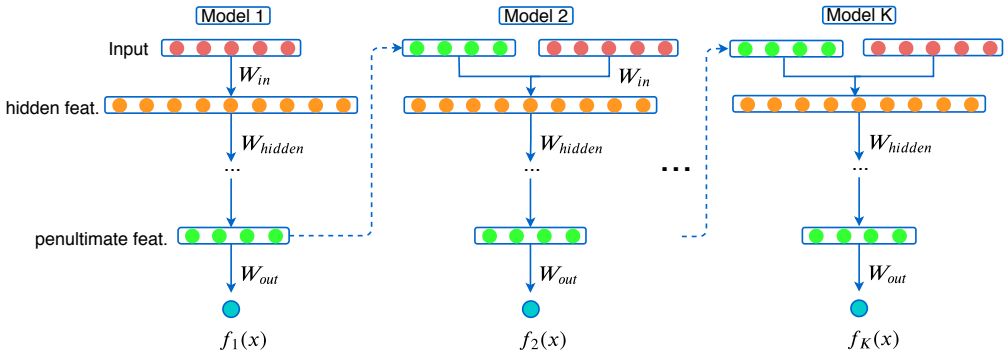

Figure 1: GrowNet architecture. After the first weak learner, each predictor is trained on combined features from original input and penultimate layer features from previous weak learner. The final output is the weighted sum of outputs from all predictors, $\sum_{k=1}^{k=K} \alpha_k f_k(x)$. Here Model K means weak learner K.

retical justification. AdaNet optimizes over a generalization bound that consists of empirical risk and complexity of the architecture. Coordinate descend approach is applied to the objective function, and heuristic search (weak learning algorithm) is performed to obtain $\delta - optimal$ coordinates. Although the learning process is boosting-style, the final model is a single NN whose final output layer is connected to all lower layers. Unlike AdaNet, we train each weak learner in a gradient boosting style, resulting in less entangled training. The final prediction is the weighted sum of all weak learners' output. Our method also renders a unified platform to perform various ML tasks.

In recent years, a few work have been done to explain the success of deep residual neural networks, ResNet, (He et al., 2016) with hundreds of layers by showing that they can be decomposed into a collection of many subnetworks. Huang et al. (2018) extended AdaNet to specifically focus on ResNet architecture to provide a new training algorithm for ResNet.Veit et al. (2016), meanwhile, argued that these deeper layers might serve as a bagging mechanism in a similar spirit to random forest classifier. These studies challenge the common belief that neural networks are too strong to serve as weak learners for boosting methods.

3 Model

In this section, we first describe the basic framework of GrowNet for general loss functions, and then we show how the corrective step is incorporated. The key idea in gradient boosting is to take simple, lower-order models as weak learners and use them as fundamental building blocks to build a powerful, higher-order model by sequential boosting using first or second order gradient statistics. We use shallow neural networks (e.g., with one or two hidden layers) as weak learners in this paper. As each boosting step, we augment the original input features with the output from the penultimate layer of the current iteration (see Figure 1). This augmented feature-set is then fed as input to train the next weak learner via a boosting mechanism using the current residuals. The final output of the model is a weighted combination of scores from all these sequentially trained models.

3.1 Gradient Boosting Neural Network: GrowNet

Let us assume a dataset with n samples in d dimensional feature space $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)_{i=1}^n | \boldsymbol{x}_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}$. GrowNet uses K additive functions to predict the output,

$$\hat{y}_i = \mathcal{E}(\boldsymbol{x}_i) = \sum_{k=0}^{K} \alpha_k f_k(\boldsymbol{x}_i), f_k \in \mathcal{F} \tag{1}$$

where $\mathcal{F}$ is the space of multilayer perceptrons and α_k is the step size (boost rate). Each function f_k represents an independent, shallow neural network with a linear layer as an output layer. For a given sample $\boldsymbol{x}$, the model calculates the prediction as a weighted sum of f_k's in GrowNet.

Let l be any differentiable convex loss function. Our objective is to learn a set of functions (shallow neural networks) that minimize the following equation: $\mathcal{L}(\mathcal{E}) = \sum_{i=0}^{n} l(y_i, \hat{y}_i)$.

We may further add regularization terms to penalize the model complexity but it is omitted for simplicity in this work. As the objective we are optimizing is over the functions and not on the parameters, traditional optimization techniques will not work here. Analogous to GBDT Friedman (2001), the model is trained in an additive manner.

Let $\hat{y}_i^{(t-1)} = \sum_{k=0}^{t-1} \alpha_k f_k(\boldsymbol{x}_i)$ be the output of GrowNet at stage $t-1$ for the sample $\boldsymbol{x}_i$. We greedily seek the next weak learner $f_t(\mathbf{x})$ that will minimize the loss at stage t which can be summized as,

$$\mathcal{L}^{(t)} = \sum_{i=0}^{n} l(y_i, \hat{y}_i^{(t-1)} + \alpha_t f_t(\mathbf{x}_i)) \tag{2}$$

In addition, Taylor expansion of the loss function l was adopted to ease the computational complexity. As second-order optimization techniques are proven to be superior to first-order ones and require less steps to converge, we train models with Newton-Raphson steps. Consequently, regardless of the ML task, individual model parameters are optimized by running weighted least squares regression on the second order gradients of the GrowNet's outputs. Objective function for the weak learner f_t can be simplified as follows,

$$\mathcal{L}^{(t)} = \sum_{i=0}^{n} h_i(\tilde{y}_i - \alpha_t f_t(\boldsymbol{x}_i))^2 \tag{3}$$

where $\tilde{y}_i = -g_i/h_i$, and g_i & h_i are the first and second order gradients of the objective function l at $\boldsymbol{x}_i$, w.r.t. $\hat{y}_i^{(t-1)}$. (See the pseudo-code in part 1 of Algorithm 1)

3.2 Corrective Step (CS)

In a traditional boosting framework, each weak learner is greedily learned. This means that only the parameters of t^{th} weak learner are updated at boosting step t where all the parameters of previous $t-1$ weak learners remain unchanged. The myopic learning procedures may cause the model to get stuck in a local minima, and a fixed boosting rate α_k aggravates the issue (Friedman, 2001). Therefore, we implemented a corrective step to address this problem. In the corrective step, instead of fixing the previous $t-1$ weak learners, we allow update of the parameters of the previous $t-1$ weak learners through back-propagation. Moreover, we incorporated the boosting rate α_k into parameters of the model and it is automatically updated through the corrective step. Beyond getting better performance, this move saves us from tuning a delicate parameter. CS can also be interpreted as a regularizer to mitigate the correlation among weak learners, as during corrective step, the main training objective becomes task specific loss function on just original inputs. The usefulness of this step is theoretically investigated in Zhang & Johnson (2014) for gradient boosting decision tree models and we applied these developments for our model. The Algorithm 3 from Zhang & Johnson (2014) introduces the Regularized Greedy Forest (RGF) where the coefficients of the all decision rules so far added as well as coefficients of basis functions are repeatedly re-optimized. Regularization is imposed on the nonlinear function class $\mathcal{H}$ where the non-linearity is achieved by additive models. Thus, the theoretical proofs therein can easily be adopted for our method. Our experiments in the ablation study 6.2 empirically validate the necessity of corrective step in our model as well. The corrective step is summarized in the second part of Algorithm 1.

4 Applications

In this section, we show how GrowNet can be adapted for regression, classification and ranking problems.

GrowNet for Regression. We employ mean squared error (MSE) loss function for the regression task. Let us assume l is the MSE loss; then we can easily obtain $\tilde{y}_i$, first order, and second order statistics at stage t, as follows:

$$g_i = 2(\hat{y}_i^{(t-1)} - y_i), \quad h_i = 2 \implies \tilde{y}_i = y_i - \hat{y}_i^{(t-1)}$$

Algorithm 1 Complete GrowNet training

Input: $f_0(x) = log(\frac{n_+}{n_-})$, α_0, Training data $\mathcal{D}_{tr}$
Output: GrowNet $\mathcal{E}$
for $k = 1$ **to** M **do**
  # Part 1 - Individual model training
  Initialize model $f_k(x)$
  Calculate 1^{st} and 2^{nd} order gradients:
  $g_i = \partial_{\hat{y}_i^{(k-1)}} l(y_i, \hat{y}_i^{(k-1)})$ & $h_i = \partial^2_{\hat{y}_i^{(k-1)}} l(y_i, \hat{y}_i^{(k-1)}), \forall x_i \in \mathcal{D}_{tr}$
  Train $f_k(\cdot)$ by weighted least square regression on $\{x_i, -g_i/h_i\}$
  Add the model $f_k(x)$ into the GrowNet $\mathcal{E}$
  # Part 2 - Corrective step
  for $epoch = 1$ **to** T **do**
    Calculate GrowNet output: $\hat{y}_i^{(k)} = \sum_{m=0}^{k} \alpha_m f_m(x_i), \forall x_i \in \mathcal{D}_{tr}$
    Calculate Loss from GrowNet: $\mathcal{L} = \frac{1}{n} \sum_i^n l(y_i, \hat{y}_i^{(k)})$
    Update step size α_k and model f_m parameters through back-propagation $\forall m \in \{1, ...k\}$
  end for
end for

We train next weak learner f_t by least square regression on $\{\boldsymbol{x}_i, \tilde{y}_i\}$ for $i = 1, 2, ..., n$. In the corrective step, all model parameters in GrowNet are updated again using the MSE loss.

GrowNet for Classification. For the illustration purposes, let us consider the binary cross entropy loss function; however, note that any differentiable loss function can be used. Choosing labels $y_i \in \{-1, +1\}$ (this notation has an advantage of $y_i^2 = 1$, which will be used in the derivation), the first and second order gradients, g_i and h_i , respectively, at stage t can be written as follows,

$$g_i = \frac{-2y_i}{1 + e^{2y_i \hat{y}_i^{(t-1)}}}, \quad h_i = \frac{4y_i^2 e^{2y_i \hat{y}_i^{(t-1)}}}{(1 + e^{2y_i \hat{y}_i^{(t-1)}})^2}; \quad \tilde{y}_i \quad = -g_i/h_i = y_i(1 + e^{-2y_i \hat{y}_i^{(t-1)}})/2 \quad (4)$$

The next weak learner f_t is fitted by least square regression using second order gradient statistics on $\{x_i, \tilde{y}_i\}$. In the corrective step, parameters of all the added predictive functions are updated by retraining the whole model using the binary cross entropy loss. This step slightly corrects the weights according to the main objective function of the task at hand, i.e. classification in this case.

GrowNet for Learning to Rank. In this part, we demonstrate how the model is adapted to learning to rank (L2R) with a pairwise loss. In the L2R framework, there are queries and documents associated with each query. A document can be associated with many different queries. Then for each query, the associated documents have relevance scores. Assume for a given query, a pair of documents U_i and U_j is chosen. Assume further that we have a feature vector for these documents, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Let $\hat{y}_i$ and $\hat{y}_j$ denote the output of the model for samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ respectively. According to Burges (2010), a common pairwise loss for a given query can be formulated as follows,

$$l(\hat{y}_i, \hat{y}_j) = \frac{1}{2}(1 - S_{ij})\sigma_0(\hat{y}_i - \hat{y}_j) + log(1 + e^{-\sigma_0(\hat{y}_i - \hat{y}_j)})$$

where $S_{ij} \in \{0, -1, +1\}$ denotes the documents' relevance difference; it is 1 if the U_i has a relevance score greater than U_j, -1 vice-versa and 0 if both document have been labeled with the same relevance score. σ_0 is the sigmoid function. Note that the cost function l is symmetric and its gradients can be easily computed as follows (for the details, readers can refer to Burges (2010)),

$$\partial_{\hat{y}_i} l(\hat{y}_i, \hat{y}_j) = \sigma_0(\tfrac{1}{2}(1 - S_{ij}) - \tfrac{1}{1+e^{\sigma_0(\hat{y}_i - \hat{y}_j)}}), \; \partial^2_{\hat{y}_i} l(\hat{y}_i, \hat{y}_j) = \sigma_0^2(\tfrac{1}{1+e^{\sigma_0(\hat{y}_i - \hat{y}_j)}})(1 - \tfrac{1}{1+e^{\sigma_0(\hat{y}_i - \hat{y}_j)}})$$

where I denotes the set of pairs of indices $\{i, j\}$, for which U_i is desired to be ranked differently from U_j for a given query. Then for a particular document U_i, the loss function and its first and second order statistics can be derived as follows,

$$l = \sum_{j:\{i,j\} \in I} l(\hat{y}_i, \hat{y}_j) + \sum_{j:\{j,i\} \in I} l(\hat{y}_i, \hat{y}_j)$$

$$g_i = \sum_{j:\{i,j\} \in I} \partial_{\hat{y}_i} l(\hat{y}_i, \hat{y}_j) - \sum_{j:\{j,i\} \in I} \partial_{\hat{y}_i} l(\hat{y}_i, \hat{y}_j), \quad h_i = \sum_{j:\{i,j\} \in I} \partial^2_{\hat{y}_i} l(\hat{y}_i, \hat{y}_j) - \sum_{j:\{j,i\} \in I} \partial^2_{\hat{y}_i} l(\hat{y}_i, \hat{y}_j)$$

Table 1: L2R results in Normalized Discounted Cumulative Gain for top 5 and 10 queries (NDCG@5 & 10), on Microsoft Learning to Rank with 10K queries and Yahoo LTR datasets. GrowNet results are average of 5 iterations and the values in the parenthesis represents the standard deviation. Note: XGBoost results on Yahoo dataset are from their own paper (Chen & Guestrin, 2016).

	MSLR-WEB 10K		Yahoo LTR	
	NDCG@5	NDCG@10	NDCG@5	NDCG@10
XGBoost	0.4677(0.0287)	0.4858(0.0245)	0.7618	0.7913
GrowNet (pairwise loss)	**0.5106**(0.0011)	**0.5203**(0.0015)	**0.7726**(0.0006)	**0.8101**(0.0003)
GrowNet (Gen. I div. loss)	0.5044(0.0072)	0.5137(0.0070)	0.7713(0.0006)	0.8088(0.0005)

5 EXPERIMENTS

Experiment Setup. All predictive functions added to the model are multilayer perceptrons with two hidden layers. We generally set the number of hidden layer units to roughly half of the input feature dimension. More hidden layers degraded the performance as the model starts overfitting. 40 additive functions were employed in the experiments for all three tasks and the number of weak learners in test time is chosen by the validation results. Boosting rate is initially set to 1 and automatically adjusted during the corrective step.

We trained each predictive function for just 1 epoch and the entire model is also trained for 1 epoch during the corrective step by stochastic gradient descent with Adam optimizer. The number of epochs is increased to 2 for the ranking task. We also employed $2D$ batch normalization on the hidden layers. We compared the proposed model performance with (1) XGBoost, the state of the art in traditional boosting methods, as it yields similar results compared with LightGBM and CatBoos and (2) AdaNet, the state of the art in structure learning for NNs. Tuning and model details of all 3 methods are provided in supplementary material.

Datasets. We evaluate our model on 5 datasets with 3 different tasks. Higgs Bozon dataset is used for classification. Higgs data is created using Monte Carlo simulations on high energy physics events.

To perform regression, 2 datasets from UCI machine learning repository are selected. The first one is Computed Tomography (CT) slice localization data where the aim is to retrieve the location of CT slices on axial axis. The second regression dataset is YearPredictionMSD, a subset of Million Song dataset. The goal is to predict the release year of a song from its audio features.

We choose Yahoo LTR dataset Chapelle & Chang (2011) for the learning to rank task as it is a well-known benchmark dataset and also used in the original XGBoost paper. The dataset has $20K$ queries each associated with approximately 22 documents. Train-test split from the original paper is preserved. The second benchmark ranking dataset we used is MSLR-WEB $10K$ in which there are $10K$ queries, each corresponding to list of $100 - 200$ documents. Detailed statistics of each dataset can be found in the supplementary material.

5.1 RESULTS

Regression. Table 2 reports regression performance on two UCI datasets. GrowNet outperforms both methods on Music dataset where AdaNet delivers the worse result. On CT slice localization dataset, our model obtains on par results with Adanet and displays 21% decrease in RMSE compared to XGBoost.

Table 2: Regression results in root mean square error (RMSE). GrowNet results are average of 5 iterations and the values in the parenthesis represent standard deviation.

	Music Year Pred.	Slice Localz.
XGBoost	8.9301	6.6744
AdaNet	12.1778	5.3824
GrowNet	**8.8156** (0.0061)	**5.3112** (0.3512)

Classification. To make a fair comparison with XGBoost, we tested our model on Higgs bozon dataset as it is used in XGBoost's paper Chen & Guestrin (2016). Classification results are presented in the Table 3. GrowNet clearly outperforms XGBoost using all the data. Subsampling 10%

of the data for training each weak learner also renders better performance. We used 30 weak learners (multilayer perceptrons with two hidden layers of 16 units) and the number of weak learners to be used at test time is chosen by validation results. In all 3 experiments, this number was 30.

Learning to Rank. Ranking experiment results on Yahoo and MSLR datasets are presented in Table 1. We evaluated GrowNet with 2 different loss functions, namely pairwise loss and generalized I-divergence loss. In both scenarios, GrowNet outperforms XGBoost on both datasets; in particular, it delivers $6-8\%$ increase on Microsoft data in NDCG@5 and NDCG@10, and 2.5% performance boost on Yahoo dataset in NDCG@10. For our model to achieve these results, 30 weak learners were enough.

Table 3: Classification results, in AUC, on Higgs bozon dataset. For our model, we preset 3 different results: using all the data, 10% of the data ($1M$), and 1% of the data ($100K$).

XGBoost	0.8304
GrowNet (all data)	**0.8510**
GrowNet (data sampling= 10%)	0.8439
GrowNet (data sampling= 1%)	0.8180

6 Ablation Study

We investigated different components of GrowNet. We picked 2 datasets for these experiments: Higgs and Microsoft. For Higgs dataset, we randomly selected $1M$ points for training and 5% of the remaining as the validation set. The original test data was used as the test set. For Microsoft dataset, we used Fold 1 and the original split was preserved. In all upcoming experiments, only the component that is being analyzed, is altered while the rest of the parameters remain unchanged. All ablation experiments are reported in Table 4, and the third column (GrowNet) represents the results from final version of our model on these datasets.

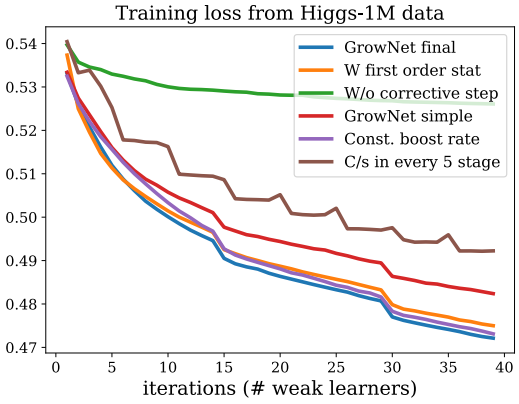

Figure 2: Classification training losses

6.1 Stacked versus simple version

As seen in Figure 1, every weak learner except the first one is trained on the combined features of the original input and penultimate layer's features from previous predictive function. It is worth to note that the input dimension does not grow by iteration; indeed, it is always the dimension of hidden layer plus the dimension of the original input. This idea of stacked features has a resemblance to auto-context Tu & Bai (2010) in literature, where the authors utilized the direct output of the classifier, along with the original inputs, to boost the image segmentation performance. The work in Becker et al. (2013) extended this idea to not only use the output of the classifier, namely class probabilities, but also the raw prediction image itself. Our model is significantly different from these methods, as we do not simply use the previous model's output but more expressive representation at the penultimate layer. These features leverage our model by propagating more complex information from previous model to the new one. Moreover, we utilize stacked features to introduce a novel boosting scheme. Instead of weighted sampling, we used the penultimate layer features of the previous weak learner and let the next learner to decide on selecting or fusing any features. In addition to empirical evidence, Theorem 1 in Tu & Bai (2010) proves that stacked features, referred to as auto-context, monotonically decrease the training error. The results of the theorem is not bounded by any particular classifier type, thus, the proof can also be applied to the proposed model.

To empirically test the advantage of this stacked model, we compared the proposed model against its simpler version in which the original input features are used for all learners. The sixth column in Table 4 presents the results from the simpler version. In both tasks, the stacked model outperforms the simpler version; especially, the difference is noteworthy in the ranking task. Training loss in

Table 4: Ablation study experiment results on Higgs $1M$ and Microsoft (Fold 1) datasets. All models have two-layer shallow networks as weak learners. Hidden layer dimension is 16 for classification and 64 for ranking task. The third column is the final GrowNet model that all different versions are compared against. Reported results are AUC scores for classification and NDCG for ranking.

Datasets	Eval. metric	GrowNet	1^{st} order grad.	Constant α_t	Simple version	No CS	CS in every 5 stage
Higgs $1M$	AUC	**0.8401**	0.8363	0.8397	0.8326	0.8093	0.8315
MSLR Fold1	NDCG@5	**0.5106**	0.5001	0.5020	0.4836	0.4743	0.4881
	NDCG@10	**0.5195**	0.5104	0.5115	0.4972	0.4872	0.4998

Figure 2 also supports the information gain while the stacked version is utilized. Unlike tree boosting methods, our model makes this architecture possible through its flexible weak learners.

6.2 Analyzing corrective step

Among all components of the model, the corrective step is presumably the most vital one. In this step, the parameters of all weak learners, that are added to the model, are updated by training the whole model on the original inputs without the penultimate layer features. The loss function used in this step is a task specific one. This procedure allows the model to rectify the parameters to specifically better accommodate the task at hand rather than fitting negative gradients. CS also alleviates the potential correlation among weak learners. Moreover, within this step, we incorporated the boosting rate α_t and it is automatically adjusted without requiring any tuning. The last two columns of Table 4 present the classification and learning to rank results from GrowNet without using any corrective step and using a corrective step in every 5 stages, respectively. The performance severely degraded in the former one, and the model hardly learned any information after a couple of predictive functions added. The flat training loss in Figure 2 confirms this phenomenon as well. Running the corrective step in every 5 steps rendered much better performance, yet was not as good as GrowNet's results. The stair-like loss curve in the Figure 2 evidently displays the influence of the corrective step on model training.

Dynamic boost rate. Within the corrective step, we are able to dynamically update the boost rate α_t (at stage t). Taking this measure saved us from tuning one more parameter as well as yielded a mild performance increase in all tasks. Moreover, the model obtained better training loss convergence, compared to the fixed boost rate version (see Fig. 2). In our setup, starting with $\alpha_0 = 1$, the boost rate is automatically updated each time the corrective step is executed (see Fig. 3) . Results of the best constant boost rate ($\alpha_t = 0.1$), coarsely tuned in a set of $\{0.01, 0.1, 1\}$, are reported in fifth column of Table 4.

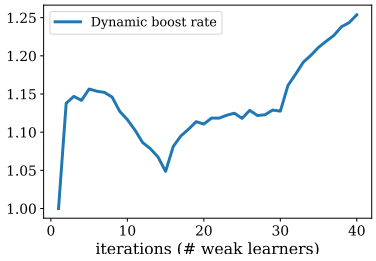

Figure 3: Boosting rate evolution

6.3 First order statistics vs second order statistics

In this experiment, we explored the impact of first and second order statistics on model performance as well as on the convergence of training loss. As the forth column of Table 4 displays, using the second order (third column in the Table 4) renders a slight performance boost over the first order in classification and almost 2% increase in learning to rank task. Figure 2 displays the effects of first and second order statistics on training loss. The final model (with second order statistics) again shows slightly better convergence on classification yet the difference is more apparent on ranking. As the learning rate is decreased by a rate of $1/2$ per 15 weak learners, sudden drops are observed in classification loss curve at 15^{th} and 30^{th} stages in Figure 2.

6.4 Analyzing the effect of hidden layers

As the literature suggests, boosting algorithms work best with weak learners, thus we utilized a shallow neural network with two hidden layers as a weak predictor for our model. While adding more hidden layers yields stronger predictors, they are not weak learners any-

more. To explore this weak learner limit on the number of hidden layers, we assayed weak learners with 1, 2, 3, and 4 hidden layers. Each hidden layer had 16 units. Although weak learners with more hidden layers render better training loss convergence as expected, the overall model starts saturating on performance and overfitting. Weak learners with 1 and 2 hidden layers attain the best scores, yet the latter one outperforms the former. The worst test AUC score is from the model with 4 hidden layers (See Fig 2 in Supp. material).

Altering the number of hidden units has a lesser effect on performance. To illustrate the impact of hidden layer dimensions, we tested the final model (weak learner with two hidden layers) with various hidden units. Higgs data has 28 features and we tested the model with 2, 4, 8, 16, 32, 64, 128 and 256 hidden units. The smaller the hidden layer dimension is, the less information propagation the weak learners get. On the other hand, having a large number of units also leads to overfitting after a certain point. Figure 4 displays test AUC scores from this experiment on Higgs 1M data. The highest AUC of 0.8478 is achieved with 128 units, yet the performance suffers when the number is increased to 256.

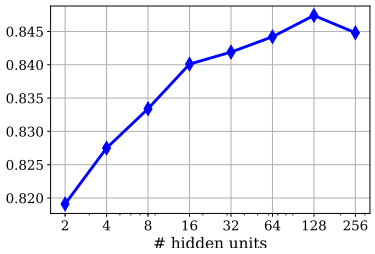

Figure 4: Effect of # neurons on classification performance

6.5 GrowNet versus DNN

One might ask what would happen if we just combine all these shallow networks into one deep neural network. There are a couple of issues with this approach: (1) it is very time-consuming to tune the parameters of the DNN, such as the number of hidden layers, the number of units in each hidden layer, the overall architecture, batch normalization, dropout level, and etc., (2) DNNs require a huge computational power and in general run slower. We compared our model (with 30 weak learners) against DNN with 5, 10 , 20, and 30 hidden-layer configurations. The best DNN (with 10 hidden layers) produced 0.8342 on Higgs 1M data in 1000 epochs, and each epoch took 11 seconds. The DNN achieved this score (its best) at epoch 900. GrowNet rendered 0.8401 AUC on the same configuration with 30 weak learners. The average stage training time, including the corrective step, took 50 seconds. On top of performance boost, GrowNet has a clear advantage of 6x faster training time ($11 * 900 = 9900$ vs $50 * 30 = 1500$) compared to vanilla DNNs. Both models are run on the same machine with NVIDIA Tesla V100 (16GB) GPU.

Further details and illustrations from the ablation study and the code are provided in the supplementary material.

7 Conclusion

In this work, we propose *GrowNet*, a novel approach to leverage shallow neural networks as "weak learners" in a gradient boosting framework. This flexible network structure allows us to perform multiple machine learning tasks under a unified framework while incorporating second order statistics, corrective step and dynamic boost rate to remedy the pitfalls of traditional gradient boosting decision tree. Ablation study is conducted to explore the limits of neural networks as weak learners in the boosting paradigm and analyze the effects of each GrowNet component on the model performance and convergence. We show that the proposed model achieves better performance in regression, classification and learning to rank on multiple datasets, compared to state-of-the-art boosting methods. We further demonstrate that GrowNet is a better alternative to DNNs in these tasks as it yields better performance, requires less training time and is less cumbersome in hyperparameter tuning.

REFERENCES

C. J. Becker, R. Rigamonti, V. Lepetit, and P. Fua. Kernelboost: Supervised learning of image features for classification. In *Technical Report*, 2013.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Proceedings of the 19th International Conference on Neural Information Processing Systems*, 2007.

C. J. C. Burges. From ranknet to lambdarank to lambdamart: An overview. In *Microsoft Research Technical Report*, 2010.

O. Chapelle and Y. Chang. Yahoo! learning to rank challenge overview. *Journal of Machine Learning Research - W & CP*, 14:1–24, 2011.

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

C. Cortes, X. Gonzalvo, V. Kuznetsov, M. Mohri, and S. Yang. AdaNet: Adaptive structural learning of artificial neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

S. E. Fahlman and C. Lebiere. The cascade-correlation learning architecture. In *NIPS*, 1990.

Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, pp. 256–285, 1995.

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

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

L. Hansen and P. Salamon. Neural network ensembles. *TPAMI*, 12:993–1001, 1990.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pp. 770–778, 2016.

G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

F. Huang, J. Ash, J. Langford, and R. Schapire. Learning deep ResNet blocks sequentially using boosting theory. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

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. In *NIPS*, 2017.

S.M. McKinney, M. Sieniek, and V. Godbole et. al. International evaluation of an ai system for breast cancer screening. *Nature*, 577:89–94, 2020.

D.W. Opitz and J.W. Shavlik. Actively searching for an effective neural network ensemble. *Connection Science*, 8:337–353, 1996.

M. P. Perrone and L. N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. pp. 126–142. Chapman and Hall, 1993.

L. Prokhorenkova, G. Gusev, A. Vorobev, A.V. Dorogush, and A. Gulin. Catboost: unbiased boosting with categorical features. In *NeurIPS*, 2018.

H. Schwenk and Y. Bengio. Training methods for adaptive boosting of neural networks for character recognition. In *NIPS*, 1997.

H. Schwenk and Y. Bengio. Boosting neural networks. *Neural Computation*, 12:1869–1887, 2000.

S. Simeon, M.L. David, D.M. Marco, and D. Christopher et. al. A multimodality test to guide the management of patients with a pancreatic cyst. *Science Translational Medicine*, 11(501), 2019. doi: 10.1126/scitranslmed.aav4772.

Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(10):1744–1757, 2010.

S. Tyree, K. Weinberger, K. Agrawal, and J. Paykin. Parallel boosted regression trees for web search ranking. In *20th International Conference on World Wide Web*, 2011.

A. Veit, M. Wilber, and S. Belongie. Residual networks behave like ensembles of relatively shallow networks. In *NIPS*, 2016.

T. Zhang and R. Johnson. Learning nonlinear functions using regularized greedy forest. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.