SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

January 2020, Vol. 63 112102:1–112102:12
https://doi.org/10.1007/s11432-018-9944-x

Snapshot boosting: a fast ensemble framework for deep neural networks

Wentao ZHANG[1,2], Jiawei JIANG[3*], Yingxia SHAO[4] & Bin CUI[1,2,5]

[1]*Center for Data Science, Peking University, Beijing* 100871*, China;*
[2]*National Engineering Laboratory for Big Data Analysis and Applications, Beijing* 100871*, China;*
[3]*Department of Computer Science, ETH Zurich, Zurich* 8092*, Switzerland;*
[4]*Beijing Key Lab of Intelligent Telecommunications Software and Multimedia, School of Computer Science, Beijing University of Posts and Telecommunications, Beijing* 100876*, China;*
[5]*Key Lab of High Confidence Software Technologies (MOE), Department of Computer Science, Peking University, Beijing* 100871*, China*

Received 18 December 2018/Revised 27 February 2019/Accepted 15 April 2019/Published online 24 December 2019

Abstract Boosting has been proven to be effective in improving the generalization of machine learning models in many fields. It is capable of getting high-diversity base learners and getting an accurate ensemble model by combining a sufficient number of weak learners. However, it is rarely used in deep learning due to the high training budget of the neural network. Another method named snapshot ensemble can significantly reduce the training budget, but it is hard to balance the tradeoff between training costs and diversity. Inspired by the ideas of snapshot ensemble and boosting, we propose a method named snapshot boosting. A series of operations are performed to get many base models with high diversity and accuracy, such as the use of the validation set, the boosting-based training framework, and the effective ensemble strategy. Last, we evaluate our method on the computer vision (CV) and the natural language processing (NLP) tasks, and the results show that snapshot boosting can get a more balanced trade-off between training expenses and ensemble accuracy than other well-known ensemble methods.

Keywords ensemble learning, deep learning, boosting, neural network, snapshot ensemble

Citation Zhang W T, Jiang J W, Shao Y X, et al. Snapshot boosting: a fast ensemble framework for deep neural networks. Sci China Inf Sci, 2020, 63(1): 112102, https://doi.org/10.1007/s11432-018-9944-x

1 Introduction

Deep neural network (DNN) has been widely used and achieved great performance on many practical applications [1–3]. Usually, we train a single DNN until convergence, but the generalization ability of a single model is limited [4]. To solve this problem, a simple and useful method is ensemble learning that trains a set of neural networks and combines their outputs scientifically [5]. However, such method is rarely used because training a neural network will cost lots of time.

Boosting is a popular ensemble algorithm which is capable of getting high diversity [6] and good interpretability [7, 8]. It increases the weight of the misclassified samples for the next training process, as the weight of each sample can be changed in each cycle and the loss function is closely related to the sample weights, the optimization path is changed correspondingly and the diversity can be enhanced. Note that the traditional boosting algorithm assumes the base models only need to be weak, and it is true that we can finally get an ensemble model with high accuracy if we have plenty of weak base models. Although diversity can be enhanced, such a method may not be appropriate in the deep learning

* Corresponding author (email: jiawei.jiang@inf.ethz.ch)

scenario. Because traditional boosting algorithm trains each base model individually, it needs a long training budget to converge and we cannot get enough number of base models in a limited time.

In order to ensemble the neural networks without significantly increasing the training cost, many new methods have been proposed recently. Deep incremental boosting [9] uses the transfer learning approach to reduce the start-up time in training each incremental base model for the ensemble. Thus, it can reduce the required training time compared with the traditional boosting algorithm. Unlike bagging [10] and boosting, which train each base model independently, snapshot ensemble [11] is proposed to train a single neural network and use stochastic gradient descent with warm restarts (SGDR) [12] to obtain multiple base models. As SGDR periodically resets the learning rate and decays it with a cosine annealing [12], snapshot ensemble is able to escape from the current local minimum and converge to several local optima along its optimization path. Snapshot ensemble saves the model parameters before resetting the learning rate and treats the model replica as a base model. In this way, it has the ability to ensemble multiple neural networks with less training cost compared with traditional ensemble methods.

For ensemble learning using the neural network, it is hard to get plenty of base models due to the high training cost, thus we cannot assume the base model to be weak as the traditional boosting algorithm does. In the condition of a limited training budget, there are four fundamental issues affecting the ensemble accuracy — the number, accuracy, diversity of base models, and the ensemble strategy. Unfortunately, both snapshot ensemble and deep incremental boosting have nonnegligible drawbacks regarding these four aspects.

Number of base models. Both snapshot ensemble and deep incremental boosting use a fixed number of epochs for each base model. To guarantee the convergence, they have to give enough or even more training epochs for each base model. However, the ideal training time for each base model is unpredictable. As a result, if some base models converge faster than the others, some training epochs may be wasted using this fixed schema. Therefore, with a limited training budget, these methods are inefficient in generating base models as many as possible.

Accuracy of base models. These two methods both save the model at the end of each training cycle. However, the saved model may not be the best during the training cycle. Since they both limit the number of training epochs, the saved model may not reach the best local minima. Worse for deep incremental boosting, compared with the previous base model, the next network is trained on different datasets and becomes deeper, thus its accuracy cannot be guaranteed if trained with a limited training budget.

Diversity of base models. For snapshot ensemble, it enhances diversity by using large initial learning rate and then converging to different local minimums. Therefore, the choice of initial learning rate largely affects diversity. As snapshot ensemble decays the learning rate slowly, it is hard to converge quickly if we use a large initial learning rate. Conversely, if given a small initial learning rate and less training epochs, the diversity may not be guaranteed because the base models have the same structure and they are more likely to converge to nearby local minimums. As a result, snapshot ensemble is unable to handle the tradeoff well between training costs and diversity.

Ensemble strategy. As we analyzed above, snapshot ensemble is unable to handle the tradeoff well between training costs and diversity. Therefore, snapshot ensemble may produce similar base models if given less training costs. Since it directly averages the softmax outputs of all base models without model selection, these similar base models may incur bias and seriously decrease the accuracy of the ensemble model [13]. Deep incremental boosting learns the weight of each base model with the training set, but this model weight can be improper for inference as the base model is easy to overfit if using the DNN [14].

The boosting algorithm can significantly improve the diversity of the ensemble system, and snapshot ensemble is capable of reducing the training cost. Inspired by the ideas of these two methods, we propose an ensemble method named snapshot boosting. To address the above four issues, snapshot boosting has made a series of optimizations. First, in order to obtain enough base models, we propose to dynamically adjust the learning rate instead of using fixed cosine annealing, thus the epochs for training each base model are not fixed. Second, we use a validation set to identify the best base model in each training cycle and therefore improve the model accuracy. Third, for the purpose of increasing model diversity,

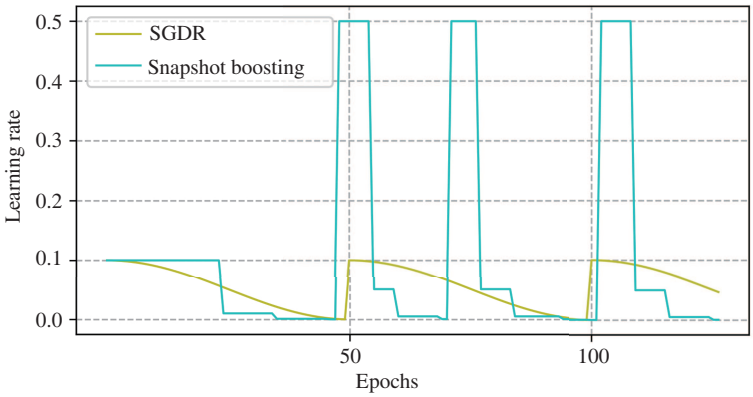

Figure 1 (Color online) The learning rate schedule of a 40-layer DenseNet on CIFAR100 using snapshot ensemble and snapshot boosting. Note that SGDR is used in snapshot ensemble and its learning rate restarts every 50 epochs.

we propose to reset the learning rate with a large value and introduce dynamic weights for misclassified samples during training. As shown in Figure 1, we dynamically adjust the learning rate according to the accuracy on the validation set, choosing a large learning rate does not cause a significantly slow convergence which can be observed in the cosine annealing strategy. Finally, our ensemble strategy outperforms these two ensemble methods via model selection, feature engineering, and a meta-learner on top of base models.

The main contributions of our work can be summarized as follows:

• We design a boosting-based framework for training. Combined with the advantage of snapshot ensemble and boosting, snapshot boosting can generate more accurate and diverse base models with low training cost.

• We propose an effective ensemble strategy. By combining model selection, feature engineering, and meta-learner techniques, snapshot boosting achieves superior ensemble accuracy.

• The experiment results on computer vision (CV) and natural language processing (NLP) tasks show that snapshot boosting achieves better ensemble accuracy than the competitors and gets a balanced trade-off between training expenses and model accuracy. Specifically, compared with other state-of-the-art solutions, snapshot boosting only needs half of the training budget to achieve the same accuracy using ResNet-32 on CIFAR-100.

2 Related work

There are many strategies to ensemble the neural networks, in which one widely used method is bagging. It uses the bootstrap aggregation to train multiple models while reducing variance. In order to combine these base learners, lots of bagging strategies have been proposed. One method is the majority voting [15], it counts the votes from all base learners and makes the prediction using the most voted label. Another method is averaging [16], which averages the softmax outputs of all base learners. By averaging the outputs of six residual networks, He et al. [17] won the first place in ILSVRC 2015. Unlike bagging which trains each base model independently, snapshot ensemble trains a single neural network, converges to several local optimums along its optimization path, and saves the model parameter at each local minimum as one base model. After that, it averages the softmax outputs of base models as the final result. Our work differs from them in that we use a boosting method to get the base models and a specific ensemble strategy to combine their outputs.

In contrary to bagging which aims at reducing model's variance, boosting is a method combining weak learners into a single strong learner in an iterative fashion, so that it can effectively reduce the bias of a model [18]. Schwenk and Bengio [19] used AdaBoost to improve the ensemble accuracy of neural networks. Some other methods [20, 21] are proposed to reduce the prediction time. Unlike them, deep incremental boosting uses the transfer learning to accelerate the initial warm-up phase of training

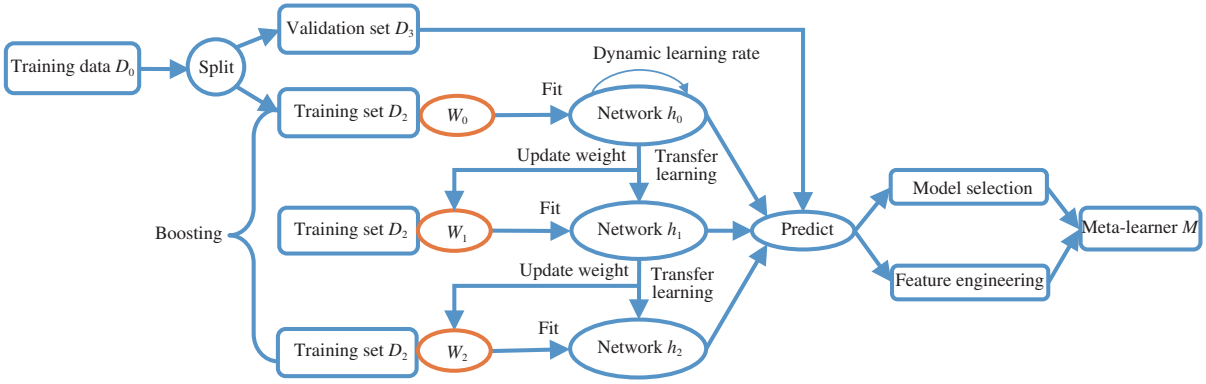

Figure 2 (Color online) The procedure of snapshot boosting.

each boosting round. Similarly, our proposed method also uses the boosting method to train each base network. However, the way we use the sample weights and the strategy we combine the base models are totally different.

Another popular ensemble method is stacking [22], it trains multiple base models and combines their outputs via a meta-learner. Based on the same idea, Laan [23] proposed super learner, which used cross-validation to estimate the performance of base models. Deep super learner [24] used the traditional machine learning in a deep learning architecture, it circularly appended the outputs of base models to the training data and used the super learner to combine their outputs. Ju et al. [25] adopted super learner from convolution neural network perspective, in which a CNN network was used to combine each output of the base model. However, all methods above train each base model independently and do not take the training budget into consideration. That is the difference between our method and these prior studies.

3 Snapshot boosting

In this section, we first give an overview of snapshot boosting and then describe each component in detail.

3.1 Overview of snapshot boosting

Two fundamental factors which have a great influence on ensemble learning are the method that generates base models and the method that combines them. Figure 2 shows the procedure of snapshot boosting ensembling three neural networks. This method can be divided into two parts. The first part generates many base models with high diversity and accuracy, and the other one aims at combining their outputs scientifically.

3.2 Method to get the base model

We set D_0 and D_1 as the original training set and test set, and T as the number of base models. Besides, we use the parameter α to control the split ratio and the parameters r_1 and r_2 as the initial learning rate of the first base model and other base models, respectively. Furthermore, we use p_1, p_2, δ and f to adjust the learning rate during the training process. Algorithm 1 shows the full procedure in detail.

Getting more base models. To get more base models given a fixed budget and improve the generalization of the model, pre-training methods are used to initialize model parameters to a region near a local minimum [26]. In this way, the time for a neural network to jump from one local minimum to another is usually less than that required from the initial point to the first local minimum. In other words, training the first base model is generally slower than base models trained afterward. Considering this observation, training each base model with the same epochs, which snapshot ensemble proposes, is not efficient enough. We therefore study a more efficient strategy for the circular training of base models.

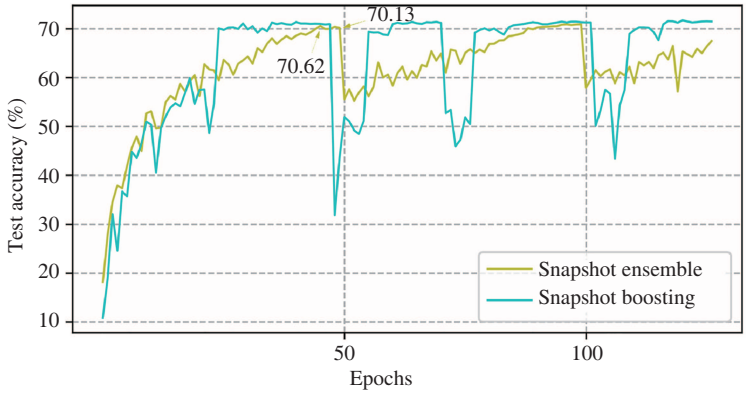

Figure 3 (Color online) The test accuracy of a 40-layer DenseNet on CIFAR100 using snapshot ensemble and snapshot boosting. Note that SGDR is used in snapshot ensemble and its learning rate restarts every 50 epochs.

Algorithm 1 Snapshot boosting

Input: D_0: the original training set; D_1: the test set; T: the number of base models; α: the split ratio; r_1: the initial learning rate for training the first base model; r_2: the initial learning rate for training other base models; k: the number of classes; p_1; p_2; δ; f;

Output: The prediction on the test set, R;

1: $|D_2| = |D_0|(1-\alpha)$; $|D_3| = |D_0|\alpha$;
2: $m = |D_2|$; $n = |D_3|$; $t = 0$;
3: $W_0(i) = 1/m$;
4: $h_0 \leftarrow F(D_2, W_0, r_1, p_1, p_2, \delta, f)$;
5: **for** $t = 1$ to T **do**
6: Set the learning rate to r_2 and decay it according to the validation accuracy on D_3;
7: $h_t \leftarrow F(D_2, W_t, h_{t-1}, r_2, p_1, p_2, \delta, f)$;
8: Save the best mode h_t before the validation accuracy on D_3 is no significant increased;
9: Get the error ϵ_t on D_3, $\epsilon_t = \frac{1}{n}\sum_{i=1}^{n} I(h_t(x_i) \neq y_i)$;
10: $\beta_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t} + \frac{1}{10}\log(k-1)$;
11: $W_{t+1}(i) = \frac{1/m}{Z_t}\mathrm{e}^{-\beta_t y_i h_t(x_i)}, i = 1, 2, \ldots, m$;
12: Where Z_t is a normalization factor;
13: $V_t \leftarrow$ use h_t to predict the softmax outputs of D_3;
14: $S_t \leftarrow$ use h_t to predict the softmax outputs of D_1;
15: $t = t + 1$;
16: **end for**
17: $V \leftarrow$ merge each base model's outputs V_t, $t = 1, 2, \ldots, T$;
18: $S \leftarrow$ merge each base model's outputs S_t, $t = 1, 2, \ldots, T$;
19: Use V to do the model selection and feature engineering;
20: $V^* \leftarrow V$'s remaining features;
21: $S^* \leftarrow S$'s remaining features;
22: $M \leftarrow$ fit a meta-learner on V^*;
23: $R \leftarrow M(S^*)$;

With the help of the validation data D_3, we can adjust the learning rate adaptively and test whether the base model has converged during the training process.

More concretely, we calculate the model's accuracy on the validation set D_3 after each epoch. During the training process F, if the improvement of accuracy is less than δ in p_1 epochs, we multiply the learning rate r by a decaying factor f. Besides, if such condition lasts for p_2 epochs, we reset the learning rate to r_2. At last, we save the model parameter which has the highest validation accuracy during this cycle as one base model h_t (lines 6–8).

As shown in Figure 3, although snapshot boosting needs more time to reach the first local minimum, the model can converge to the next local minimum in a shorter time. As we save each base model early and then restart the training process with the pre-trained model for the next base model, we can get more base models than snapshot ensemble with the same training budget.

Getting base models with high accuracy. We use two methods to increase the model accuracy.

First, we set the first initial learning rate r_1 to a small value, thus the model can converge to a good local minimum. Second, we use the validation set to identify and save the best model before resetting the learning rate. As shown in Figure 3, the model before the last epoch has a test accuracy of 70.62%, which is higher than the last epoch's 70.13%. However, snapshot ensemble cannot save the best model during the first cycle as it directly saves the model parameters of the last epoch. But snapshot boosting is able to overcome this problem as it can assess the entire training process with the validation data D_3.

Getting base models with high diversity. As aforementioned, we train the first base model with a small learning rate r_1. After that, we reset the learning rate with a large value r_2, thus there is more chance to escape from the local minimum and explore more parameter space. Therefore, we can increase the diversity of base models.

We further study the techniques giving large weights to those samples who are misclassified in each training cycle. Thus, the optimization path is changed correspondingly and the diversity can be further increased. To that end, we use a boosting framework. We define that n is the number of validation samples, m is the number of training samples, k is the number of classes, x_i is the feature of the i-th sample, and y_i is the label of the i-th sample. After getting the base model $h_t(x)$, we use the validation set to calculate its error ϵ_t (line 9):

$$\epsilon_t = \frac{1}{n}\sum_{i=1}^{n} I(h_t(x_i) \neq y_i). \tag{1}$$

Then, we calculate the coefficient β_t (line 10):

$$\beta_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t} + \frac{1}{10}\log(k-1), \tag{2}$$

where β_t has the ability to measure the generalization ability of the base model since higher ϵ_t leads to smaller β_t. Next, let Z_t be a normalization factor, we update the weight of each training sample for next base model $W_{t+1}(i)$ as follows (line 11):

$$W_{t+1}(i) = \frac{1/m}{Z_t}\mathrm{e}^{-\beta_t y_i h_t(x_i)}, \quad i = 1, 2, \ldots, m. \tag{3}$$

Our boosting method differs from the traditional AdaBoost in three aspects. First, we use the validation set D_3 to calculate the generalization error of each base model, while AdaBoost gets this error on the training data D_0 with the risk of overfitting. Next, AdaBoost uses the training data to calculate the training error and update weights based on the former weights, thus the weight of each sample will not change severely and the diversity between base models is limited. In snapshot boosting, however, we calculate the sample weight $W_{t+1}(i)$ for the next base model with the initial weights $W_0 = \frac{1}{m}$, instead of the former weight $W_t(i)$. Since the sample weights of the former base model have no impact on the latter one, the diversity between base models can be enhanced. Last, AdaBoost assigns the weight of each base model according to the training error, while snapshot boosting achieves better ensemble accuracy by using a meta-learner to learn the base model's outputs with a validation set.

3.3 Method to combine the base models

Model selection. It is unwise to ensemble all the available neural networks for prediction when some of them are similar [27]. Therefore, it is necessary to do model selection before combining their outputs. With the help of the validation set, we can try lots of model selection methods, such as AIC and BIC [28]. Especially, in snapshot boosting, we use the cross-validation method [29] to select the base models.

Feature engineering. The quality and quantity of the features will have a great influence on the model's ensemble accuracy [30]. Thus, it is worthwhile to conduct feature engineering with the validation set. For example, if one model performs poorly in classifying a specific class, we remove the corresponding softmax outputs before feeding it to the meta-learner to avoid a bad influence on the meta-learner. Since the output of each base model can be regarded as the input feature for the following meta-learner, we

call it feature engineering in our context. A lot of feature engineering methods can be used in this process, such as feature selection and feature transformation. In this experiment, we use the embedded method [31] to select the effective features and use them to fit the meta-learner.

Meta-learner. By concatenating each base model's output V_t on the validation set, we get an input dataset V for the meta-learner (line 17). After model selection and feature engineering, V is transformed into a new dataset denoted by V^* (lines 19 and 20). We fit a meta-learner M on V^* to combine the outputs of base models (line 22). Note that, the time needed to fit the meta-learner is generally negligible compared with the training budget for the neural networks.

Specifically, we build m different models on V^*, including random forest, SVM, and Logistic regression. Then, we select the model M_t with the highest cross-validated prediction accuracy on V^*:

$$\arg\min_{M_t} \sum_{i=1}^{n} I(M_t(x_i) \neq y_i), \quad t = 1, 2, \ldots, m, \tag{4}$$

where n is the sample number in V^*, x is the feature of V^*, and y is the sample label. We use the methods above to select the best meta-learner, and we finally use the logistic regression in our experiments.

4 Experiments

To validate our proposed method, we conduct extensive experiments on the tasks of CV and NLP.

4.1 Experimental settings

Models. For the CV task, we train a residual neural network (ResNet) with 32 layers and a densely connected convolutional networks (DenseNet) with 40 layers [32]. The growth rate of DenseNet is 12. For the NLP task, we train a one-layer LSTM [33] network. We set the dimension of the dense embedding to 128 and set both the dropout rate and the recurrent dropout rate to 0.2. We use a mini-batch size of 64 for these networks.

Datasets. The two CIFAR datasets [34], CIFAR-10 and CIFAR-100, contain colored images with 32×32 pixels, and both of them include a training set of 50000 images and a test set of 10000 images. CIFAR-10 (C10) consists of images drawn from 10 classes and CIFAR-100 (C100) from 100 classes. We adopt a widely used data augmentation scheme [17, 35] before training. Besides, we use a standard preprocessing method [17], in which the images are normalized with the channel means and standard deviations.

We use the IMDB movie dataset to train the LSTM network. This dataset is used for sentiment classification [36]. It contains 25000 movie reviews for training and 25000 for testing, which have been labeled as positive or negative. We use the 4000 most frequent words to calculate the TF-IDF feature and set the max length of each sentence to 400.

Ensemble methods. We firstly run single model as the baseline. It is trained with a standard learning rate schedule, which starts at 0.1 and is divided by 10 when the training is at 50% and 75% of the total number of training epochs [32]. Besides, in order to assess the traditional ensemble methods, we also run experiments with bagging and AdaBoost, in which each base model uses the same learning rate schedule as described above. For AdaBoost, we use the original version for the IMDB dataset and use AdaBoost.M1 [37] for CIFAR dataset, as the latter one is a multi-label classification task. Furthermore, we also consider some recent methods, such as snapshot ensemble and deep incremental boosting (DIB). To evaluate the performance of our combining strategy, we calculate the ensemble accuracy of snapshot boosting using the same average method of snapshot ensemble, we name this variant snapshot boosting (average). Finally, we evaluate our snapshot boosting method.

Protocol. We use stochastic gradient descent (SGD) to train these networks. For snapshot boosting, we set the initial learning rate r_1 to 0.1 and the split ratio α to 0.05. We then set the resetting learning rate r_2 to 0.2 for ResNet and 0.5 for DenseNet. We divide the learning rate by 10 if the validation accuracy has not improved by 0.0001 within 3 epochs for ResNet and 4 epochs for DenseNet. Besides, if

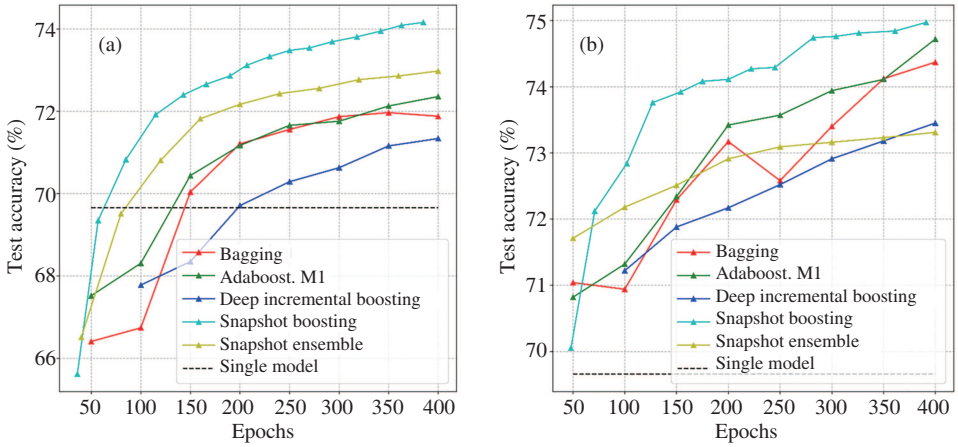

Figure 4 (Color online) Test accuracy of ensembles on CIFAR-100 using ResNet-32 (a) and DenseNet-40 (b). For the single model, the test accuracy is directly calculated on the test set in the last epoch. For the ensemble method, the test accuracy is the ensemble accuracy which is calculated with the base models already trained.

such situation lasts for 6 epochs, we save the base model with the highest validation accuracy ever seen and reset the learning rate to the max value r_2, then continue the training of the next boosting network. For snapshot ensemble, we use the same settings as the original paper, in which the max learning rate r_2 is set to 0.1 for ResNet and 0.2 for DenseNet. For deep incremental boosting, we add a two-layer block with 64 filters during each round for ResNet-32 and add one layer at the head of the third block for DenseNet-40, and each base model uses the standard learning rate schedule. Particularly, for the NLP task, we use all these methods above except deep incremental boosting as it is designed for image classification.

Training budget. The total training budget is $B = 400$ epochs for CIFAR dataset and $B = 80$ epochs for IMDB dataset. In bagging and AdaBoost, each base model is trained with a budget of 50 epochs for CIFAR dataset and 10 epochs for IMDB dataset. For snapshot ensemble on CIFAR dataset, we use the same settings in [11], in which snapshot variants are trained with $M = 8$ cycles ($B/M = 50$ epochs per cycle) for DenseNets, and $M = 10$ cycles ($B/M = 40$ epochs per cycle) for ResNets. For snapshot ensemble on IMDB dataset, snapshot variants are trained with $M = 8$ cycles ($B/M = 10$ epochs per cycle). For deep incremental boosting, the first member is trained with 100 epochs, while each cycle after the first one is only trained with 50 epochs. For snapshot boosting, the training budget for each base model is adaptively allocated considering that the time required for the network to converge to the next local minimum is uncertain.

4.2 Experimental results

End-to-end comparisons. To estimate the end-to-end performance of each ensemble method, we calculate and present the test accuracy of each method using the same fixed epochs during the training process. In Figure 4, we observe that snapshot boosting gets a higher test accuracy than other methods in most cases. More concretely, snapshot boosting achieves 72.98% ensemble accuracy within 207 epochs using ResNet-32. Thus it only uses nearly half of the training budget to reach the same accuracy achieved by other state-of-art ensemble methods. Besides, it achieves 73.76% ensemble accuracy within 127 epochs using DenseNet-40, which is 2 times faster than other best method. The result proves that snapshot boosting can achieve a good ensemble accuracy in a shorter time, and such ability is extremely useful when the training budget is insufficient.

Diversity of base models. In order to explore the diversity between base models, we compute the pairwise correlation of softmax outputs for each pair of snapshot ensemble and snapshot boosting. Since the earlier models have higher errors, we choose the last 4 models which are more likely to be selected as the base models for the final ensemble. Besides, we calculate the mean accuracy of the last 4 models for each method. The mean accuracy of snapshot ensemble is 71.64%, and that of snapshot boosting is

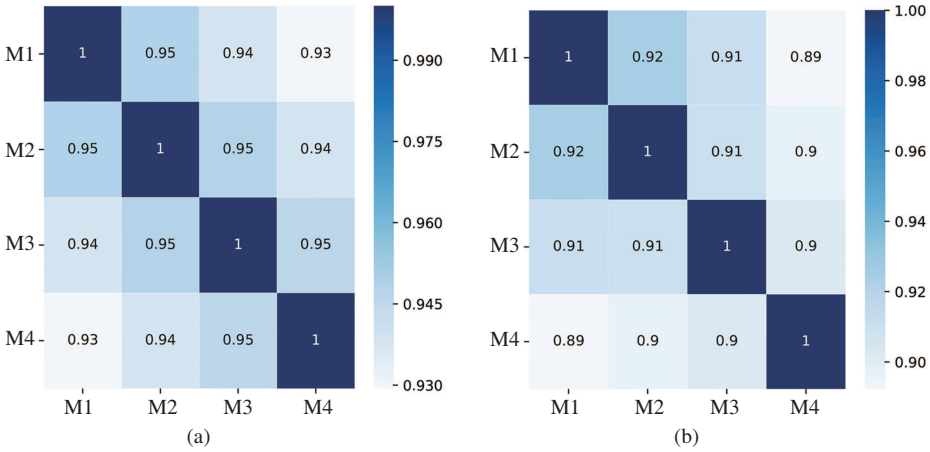

Figure 5 (Color online) Pairwise correlation of softmax outputs between base models for DenseNet-40 on CIFAR-100. (a) Snapshot ensemble; (b) snapshot boosting.

Table 1 Comparisons on CV tasks[a)]

Model	Method	Number of base model	Best base model accuracy (%)	Ensemble accuracy (%)	Increased accuracy (%)
ResNet-32	Single model	1	69.66	–	–
	Deep incremental boosting	7	67.78	71.34	3.56
	Bagging	8	67.86	71.88	4.02
	AdaBoost.M1	8	68.12	72.36	4.24
	Snapshot ensemble	10	69.83	72.98	3.15
	Snapshot boosting	16	68.75	**74.16**	5.41
DenseNet-40	Single model	1	72.07	–	–
	Deep incremental boosting	7	71.22	73.45	2.23
	Bagging	8	71.34	74.37	3.03
	AdaBoost.M1	8	71.35	74.72	3.37
	Snapshot ensemble	8	71.71	73.31	1.60
	Snapshot boosting	14	71.55	**74.97**	3.42

a) For each ensemble method, we present the number and best accuracy of the base models, the final ensemble accuracy, and accuracy improvement brought by the ensemble strategy. The ResNet-32 and the DenseNet-40 are trained on CIFAR-100. Note that the best ensemble accuracy for each network is bolded.

71.39%. Obviously, the base models generated by these two methods have similar accuracy. However, Figure 5 shows that snapshot ensemble has a large pair-wise correlation value, meaning that the diversity of its base models is not enough. Actually, this is the reason why snapshot boosting has better ensemble accuracy than snapshot ensemble in most cases.

Ensemble accuracy. In order to show the performance of each ensemble strategy, we train different ensemble methods with different networks and different datasets. As shown in Tables 1–3, snapshot boosting achieves the highest ensemble accuracy than all other methods with the same budgets. In particular, it yields an accuracy of 74.16% on CIFAR-100 and ResNet-32, which significantly outperforms other ensemble methods with the same training budget. This result demonstrates that snapshot boosting is superior to other ensemble methods in terms of the trade-off between training expenses and accuracy.

Number of base models. As Table 1 illustrates, snapshot boosting can get 16 base models for ResNet-32 and 14 base models for DenseNet-40, while snapshot ensemble only gets 10 and 8 base models, respectively, using the same training resources. The reason is that snapshot boosting can dynamically decay the learning rate and converge to the next local minimum in a shorter time.

The accuracy of base models. As shown in Table 1, during training ResNet-32 on CIFAR-100, the accuracy of the best base model in snapshot boosting is 1.08% lower than that in snapshot ensemble. This is because snapshot boosting sets 5% of the input dataset as the validation set, thus less data is

Table 2 Ensemble accuracy on IMDB dataset[a)]

Model	Method	Accuracy (%)
LSTM	Single model	89.02
	Bagging	89.17
	AdaBoost	90.13
	Snapshot ensemble	90.71
	Snapshot boosting (average)	91.17
	Snapshot boosting	**91.52**

a) All methods are trained with 400 epochs. Note that the best accuracy for each method is bolded.

Table 3 Ensemble accuracy on CIFAR-10 dataset[a)]

Model	Method	Accuracy (%)
ResNet-32	Single model	93.50
	DIB	92.78
	Bagging	93.24
	AdaBoost.M1	93.47
	Snapshot ensemble	94.26
	Snapshot boosting (average)	94.87
	Snapshot boosting	**95.12**
DenseNet-40	Single model	93.43
	DIB	93.26
	Bagging	93.74
	AdaBoost.M1	93.12
	Snapshot ensemble	93.57
	Snapshot boosting (average)	93.96
	Snapshot boosting	**94.31**

a) All methods are trained with 400 epochs. Note that the best accuracy for each method is bolded.

used to train each base model. Besides, the gap is only 0.16% when training DenseNet-40. Although the best accuracy of the base model is slightly lower than snapshot ensemble, snapshot boosting achieves a better ensemble accuracy since it can get more base models with high diversity.

Effects of the ensemble strategy. To further evaluate the performance of our ensemble strategy, we compare the ensemble accuracy of snapshot boosting (average) with snapshot boosting. As shown in Table 3, our proposed ensemble strategy can increase the ensemble accuracy by 0.25% for ResNet-32 and 0.35% for DenseNet-40 on CIFAR-10 dataset. Similarly, Table 2 shows that the ensemble accuracy can be increased by 0.35% for LSTM on IMDB dataset. The results above show that snapshot boosting can increase the ensemble accuracy through the meta-learner.

Analysis of other ensemble methods. For AdaBoost and Bagging, as shown in Table 1, the increased accuracy is relatively higher than deep incremental boosting and snapshot ensemble because the base models are more diverse. However, their final ensemble accuracy is worse than snapshot boosting during each training period, because they cannot simultaneously get a large number of base models with high diversity and accuracy in a limited time.

For snapshot ensemble, Table 1 shows that its best base model can get high accuracy. For example, the accuracy of its base model is the highest on ResNet-32, but the number of base models and the increased accuracy is smaller than snapshot boosting. This is because snapshot ensemble uses fixed training epochs to train each base model and lots of training resources have been wasted. Besides, the diversity between their base models is lower than snapshot boosting as it cannot use a large resetting learning rate.

As shown in Tables 1 and 3, the ensemble accuracy of the deep incremental boosting is relatively lower than other methods in most cases. Although it uses the pre-trained model to accelerate the training process, it has to train a deeper network with a new dataset to get the next base model, so the base model cannot converge in a short time.

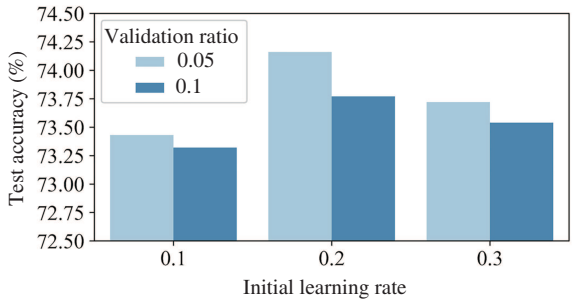

Figure 6 (Color online) The test accuracy of snapshot boosting on CIFAR-100 using ResNet-32 with different resetting learning rate r_2 and split ratio α.

Exploration of impact factors. In order to investigate the influences of related factors on snapshot boosting, we vary the value of the resetting learning rate r_2 and the split ratio for validation α, and then assess the change of test accuracy.

• Effects of resetting learning rate. For analyzing the effect of the resetting learning rate, we set r_2 to 0.1, 0.2 and 0.3. We can observe from Figure 6 that snapshot boosting with the resetting learning rate of 0.2 performs the best because we can explore more local optima and the diversity between base models can be increased accordingly. However, when setting r_2 to 0.3, the ensemble accuracy is decreased. The reason is that the model escapes too far from the current local optimum, and it needs a long time to converge to the next local optimum. With too large resetting learning rate, we may get fewer base models with a fixed training budget.

• Effects of the split ratio for validation. Intuitively, with a higher split ratio for validation, the validation data can be more representative, thus we can find the base model in each training cycle and adjust the learning rate in a more reasonable way. Besides, as we can use more data to fit the meta-learner, the combining strategy can work better. However, Figure 6 shows that the test accuracy is relatively lower when the split ratio is 0.1 than that of 0.05. This empirical result indicates that it is unsuitable to set a very large value for α. The reason is that a higher split ratio leads to fewer data used for training each base model, and therefore the base model's accuracy degrades. Thus, the value of the split ratio needs to be carefully tuned.

As shown in Figure 6, the ensemble accuracy with the worst hyper parameters is 73.32%, which is still higher than all the competitors. Besides, we find there is little change in speedup ratio if we use different hyper parameters. In conclusion, our method is very robust to different hyper parameters.

5 Conclusion

We proposed snapshot boosting, an ensemble method which could get a balanced trade-off between training expenses and ensemble accuracy. With a novel boosting method and a special learning rate schedule, many accurate and diverse base models were found with a limited training budget. Besides, with the help of the validation data, snapshot boosting conducted model selection and feature engineering, then trained a meta-learner. Such a procedure made the combining strategy more effective and robust.

We carefully evaluated snapshot boosting and several popular ensemble methods with different tasks and datasets. The results showed that snapshot boosting achieved the top-tier ensemble accuracy using a much smaller training budget than the competitors.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. 61832001, 61702015, 61702016, 61572039), National Key Research and Development Program of China (Grant No. 2018YFB1004403), and PKU-Tencent Joint Research Lab.

References

1 Liu L, Du X, Zhu L, et al. Learning discrete hashing towards efficient fashion recommendation. Data Sci Eng, 2018, 3: 307–322

2 Abdelatti M, Yuan C Z, Zeng W, et al. Cooperative deterministic learning control for a group of homogeneous nonlinear uncertain robot manipulators. Sci China Inf Sci, 2018, 61: 112201
3 Arun K S, Govindan V K. A hybrid deep learning architecture for latent topic-based image retrieval. Data Sci Eng, 2018, 3: 166–195
4 Zhang C, Bengio S, Hardt M, et al. Understanding deep learning requires rethinking generalization. 2016. ArXiv: 1611.03530
5 Opitz D, Maclin R. Popular ensemble methods: an empirical study. J Artif Intell Res, 1999, 11: 169–198
6 Melville P, Mooney R J. Creating diversity in ensembles using artificial data. Inf Fusion, 2005, 6: 99–111
7 Jiang J, Cui B, Zhang C, et al. DimBoost: boosting gradient boosting decision tree to higher dimensions. In: Proceedings of the 2018 International Conference on Management of Data. New York: ACM, 2018. 1363–1376
8 Gao W, Zhou Z H. On the doubt about margin explanation of boosting. Artif Intell, 2013, 203: 1–18
9 Mosca A, Magoulas G D. Deep incremental boosting. 2017. ArXiv: 1708.03704
10 Quinlan J R. Bagging, boosting, and C4. 5. In: Proceedings of the 13th National Conference on Artificial Intelligence and 8th Innovative Applications of Artificial Intelligence Conference, Portland, 1996. 725–730
11 Huang G, Li Y, Pleiss G, et al. Snapshot ensembles: train 1, get M for free. 2017. ArXiv: 1704.00109
12 Loshchilov I, Hutter F. Sgdr: stochastic gradient descent with warm restarts. 2016. ArXiv: 1608.03983
13 Zhou Z H. Ensemble methods: foundations and algorithms. Chapman and Hall/CRC, 2012
14 LeCun Y, Bengio Y, Hinton G. Deep learning. Nature, 2015, 521: 436–444
15 Dietterich T G. Ensemble methods in machine learning. In: Proceedings of the International Workshop on Multiple Classifier Systems. Berlin: Springer, 2000. 1–15
16 Naftaly U, Intrator N, Horn D. Optimal ensemble averaging of neural networks. Netw-Comput Neural Syst, 1997, 8: 283–296
17 He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016. 770–778
18 Friedman J, Hastie T, Tibshirani R. The Elements of Statistical Learning. New York: Springer, 2001
19 Schwenk H, Bengio Y. Training methods for adaptive boosting of neural networks. In: Proceedings of the Advances in Neural Information Processing Systems, 1998. 647–653
20 Bucilu C, Caruana R, Niculescu-Mizil A. Model compression. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM, 2006. 535–541
21 Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. 2015. ArXiv: 1503.02531
22 Breiman L. Stacked regressions. Mach Learn, 1996, 24: 49–64
23 van der Laan M J, Polley E C, Hubbard A E. Super learner. Stat Appl Genets Mol Biol, 2007, 6: 1
24 Young S, Abdou T, Bener A. Deep super learner: a deep ensemble for classification problems. In: Proceedings of the 31st Canadian Conference on Artificial Intelligence, Toronto, 2018. 84–95
25 Ju C, Bibaut A, van der Laan M. The relative performance of ensemble methods with deep convolutional neural networks for image classification. J Appl Stat, 2018, 45: 2800–2818
26 Seyyedsalehi S Z, Seyyedsalehi S A. A fast and efficient pre-training method based on layer-by-layer maximum discrimination for deep neural networks. Neurocomputing, 2015, 168: 669–680
27 Zhou Z H, Wu J, Tang W. Ensembling neural networks: many could be better than all. Artif Intell, 2002, 137: 239–263
28 Aho K, Derryberry D W, Peterson T. Model selection for ecologists: the worldviews of AIC and BIC. Ecology, 2014, 95: 631–636
29 Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, 1995. 1137–1145
30 Brownlee J. Discover feature engineering, how to engineer features and how to get good at it. Machine Learning Process, 2014
31 Guyon I, Elisseeff A. An introduction to variable and feature selection. J Mach Learn Res, 2003, 3: 1157–1182
32 Huang G, Liu Z, van der Maaten L, et al. Densely connected convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 4700–4708
33 Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks. In: Proceedings of Advances in Neural Information Processing Systems, 2014. 3104–3112
34 Krizhevsky A, Hinton G. Learning Multiple Layers of Features From Tiny Images. Technical Report, University of Toronto, 2009
35 Lin M, Chen Q, Yan S. Network in network. 2013. ArXiv: 1312.4400
36 Maas A L, Daly R E, Pham P T, et al. Learning word vectors for sentiment analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 2011. 142–150
37 Freund Y, Schapire R E. Experiments with a new boosting algorithm. In: Proceedings of the 13th International Conference on ML, 1996. 96: 148–156