LEARNING SPARSE STRUCTURED ENSEMBLES WITH STOCHASTIC GRADIENT MCMC SAMPLING AND NETWORK PRUNING

Yichi Zhang, Zhijian Ou

Speech Processing and Machine Intelligence (SPMI) Lab, Tsinghua University, Beijing, China.
zhangyic17@mails.tsinghua.edu.cn, ozj@tsinghua.edu.cn

ABSTRACT

An ensemble of neural networks is known to be more robust and accurate than an individual network, however usually with linearly-increased cost in both training and testing. In this work, we propose a two-stage method to learn Sparse Structured Ensembles (SSEs) for neural networks. In the first stage, we run SG-MCMC with group sparse priors to draw an ensemble of samples from the posterior distribution of network parameters. In the second stage, we apply weight-pruning to each sampled network and then perform retraining over the remained connections. In this way of learning SSEs, we not only achieve high prediction accuracy but also reduce memory and computation cost in both training and testing. We conduct a series of evaluation experiments by learning SSE ensembles with both FNNs and LSTMs. To the best of our knowledge, this work represents the first methodology and empirical study of integrating SG-MCMC, group sparse prior and network pruning together for learning NN ensembles.

Index Terms— learning ensembles, SG-MCMC, group sparse prior, model compression

1. INTRODUCTION

Recently there are increasing interests in using ensembles of *Deep Neural Networks* (DNNs) , which are known to be more robust and accurate than individual networks [1, 2]. An explanation stems from the fact that learning neural networks is an optimization problem with many local minima [3]. Multiple models obtained from applying stochastic optimization, e.g. the widely used Stochastic Gradient Descent (SGD) and its variants, converge to different local minima and tend to make different errors. Due to this diversity, the collective prediction produced by an ensemble is less likely to be in error than individual predictions. The collective prediction is usually performed by averaging predictions of multiple networks.

On the other hand, the improved prediction accuracy of such model averaging can be understood from the principled perspective of Bayesian inference with Bayesian neural networks. Specifically, for each test point $\tilde{x}$, we consider the predictive distribution $P(\tilde{y}|\tilde{x},\mathcal{D}) = \int P(\tilde{y}|\tilde{x},\theta)P(\theta|\mathcal{D})d\theta$, by integrating the model distribution $P(\tilde{y}|\tilde{x},\theta)$ with the posterior distribution over the model parameters $P(\theta|\mathcal{D})$ given training data $\mathcal{D}$. The predictive distribution is then approximated by Monte Carlo integration $P(\tilde{y}|\tilde{x},\mathcal{D}) \approx \frac{1}{M}\sum_{m=1}^{M} P(\tilde{y}|\tilde{x},\theta^{(m)})$, where $\theta^{(m)} \sim P(\theta|\mathcal{D}), m = 1,\cdots,M$, are posterior samples of model parameters. It is well known that such Bayesian model averaging is more accurate in prediction and robust to over-fitting than point estimates of model parameters [4, 5].

This work is supported by NSFC grant 61473168. Correspondence to: Z. Ou (ozj@tsinghua.edu.cn).

Despite the obvious advantages as seen from both perspectives, a practical problem that hinders the use of DNN ensembles is that an ensemble requires too much computation in both training and testing. Traditionally, multiple neural networks are trained, e.g. with different random initialization of model parameters. Recent studies in [2, 6] propose to learn an ensemble which consists of multiple snapshot models along the optimization path within a single training process, by leveraging a special cyclic learning rate schedule. This reduces the training cost, but the testing cost is still high.

In this paper we also aim at learning an ensemble within a single training process, but by leveraging the recent progress in Bayesian posterior sampling, namely the *Stochastic Gradient Markov Chain Monte Carlo* (SG-MCMC) algorithms. Moreover, we apply group sparse priors in training to enforce group-level sparsity on the network's connections. Subsequently we can further use model pruning to compress the networks so that the testing cost is reduced with no loss of accuracy.

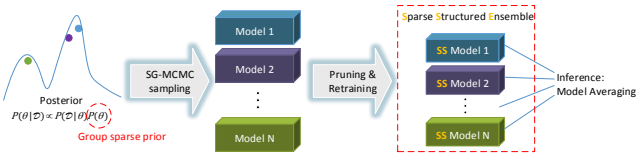


Figure 1: Overview of our two-stage method for learning SSEs.

Figure 1 presents a high-level overview of our two-stage method to learn Sparse Structured Ensembles (SSEs) for DNNs. Specifically, in the first stage, we run SG-MCMC with group sparse priors to draw an ensemble of samples from the posterior distribution of network parameters. In the second stage, we apply weight-pruning to each sampled network and then perform retraining over the remained connections as fine-tuning. In this way of learning SSEs with SG-MCMC and pruning, we reduce memory and computation cost significantly in both training and testing of NN ensembles, while maintaining high prediction accuracy. This is empirically verified in our experiments of learning SSE ensembles of both FNNs and LSTMs.

We evaluate the performance of the proposed method on two experiments with different types of neural networks. The first is an image classification experiment, which uses *Feed-forward Neural Networks* (FNNs) on the well-known MNIST dataset ([7]). Second, we experiment with the more challenging task of *Long Short-term Memory* (LSTM, [8]) based language modeling, which is conducted on the Penn Treebank dataset ([9]). It is found that the proposed method works well across both tasks. For example, we obtain 12% relative reduction (from 78.4 to 68.6) in LM perplexity by learning a SSE of 4 large LSTM models, which has only 40% of model parameters and 90% of computations in total, as compared to the large

978-1-5386-5477-4/18/$31.00

LSTM LM in [10]. Furthermore, when the embedding weights of input and output are shared as in [11], we obtain a perplexity of 62.1 (achieving 21% reduction from 78.4) by 4 large LSTMs with only 30% of model parameters and 70% of computations in total.

2. RELATED WORK

This work draws inspiration from three recent research findings, namely running SG-MCMC for efficient and scalable Bayesian posteriori sampling, applying group sparse priors to enforce network sparsity, and network pruning. To the best of our knowledge, this work represents the first methodology and empirical study of integrating these three techniques and demonstrates its usefulness to learning and using ensembles. More discussions on related studies are given next.

SG-MCMC sampling: SG-MCMC represents a family of MCMC sampling algorithms developed in recent years, e.g. *Stochastic Gradient Langevin Dynamics* (SGLD) ([12], *Stochastic Gradient Hamiltonian Monte Carlo* (SGHMC) [13], mainly for Bayesian learning from large scale datasets. SG-MCMC has the following favorable properties for learning ensembles. *(i)* SG-MCMC works by adding a scaled gradient noise during training, and thus enhances exploration of the model-parameter space. This is beneficial for finding diverse sample models for ensembles. *(ii)* Scalable and simple: the basic SG-MCMC algorithm, e.g. SGLD, is just a noisy *Stochastic Gradient Descent* (SGD), which means the same training cost as SGD on large datasets. The effectiveness of applying SG-MCMC to Bayesian learning of RNNs is shown in [5] but without considering model pruning to reduce the cost of model averaging.

Sparse structure learning: Group Lasso penalty ([14]) has been widely used to regularize likelihood function to learn sparse structures. It was applied with SGD in [15] and in [16] to learn structurally sparse DNNs and LSTMs respectively. But all focus on point estimates and are not in the context of learning ensembles. Group Lasso idea has been studied in Bayesian learning, which is known as applying group sparse priors [17, 18]; but these previous works use variational method. Applying group sparse priors with SG-MCMC has not been explored.

Model compression: Model pruning and retraining has been studied to compress CNNs [19]. Recently, [20] apply model pruning to LSTM models for automatic speech recognition task. We use similar model pruning and retraining method in the experiments. We find that model averaging can enable the ensemble with heavily-pruned networks to be more robust in prediction.

Learning ensembles: Some efforts have been made to reduce the training and testing cost for ensembles. For reducing the training time cost of ensembles, a special cyclic learning rate schedule is developed in [2], which restarts the learning rate periodically to attempt to visit multiple local minima along its optimization path and saves snapshot models. In contrast to relying on such empirical setting of the learning rate to explore model space, theoretical consistency properties of SG-MCMC methods in posterior sampling have been established [21].

3. LEARNING ENSEMBLES WITH SG-MCMC AND NETWORK PRUNING

We consider the classification problem under Bayesian inference framework. Given training data $\mathcal{D} \triangleq \{(x_i, y_i)\}_{i=1}^N$ with input feature $x_i \in \mathbb{R}^D$ and class label $y_i \in \mathcal{Y}$, where $\mathcal{Y}$ is the set of classes. We view a neural network as a conditional probabilistic model $P(y_i|x_i, \theta)$. Denote the network parameters by θ, with $P(\theta)$ a prior distribution. We compute the posterior distribution over the model parameters, $P(\theta|\mathcal{D}) \propto P(\theta)\prod_{i=1}^N P(y_i|x_i, \theta)$. For testing, given a test input $\tilde{x}$, the Bayesian predictive distribution for its label $\tilde{y}$ is given by $P(\tilde{y}|\tilde{x}, \mathcal{D}) = \mathbb{E}_{P(\theta|\mathcal{D})}[P(\tilde{y}|\tilde{x}, \theta)]$, which can be viewed as model averaging across parameters with distribution $P(\theta|\mathcal{D})$. However, the integration over the posterior is analytically intractable for deep neural networks (DNNs). Thus it is approximated by Monte Carlo integration as in the following:

$$P(\tilde{y}|\tilde{x}, \mathcal{D}) \approx \frac{1}{M}\sum_{m=1}^{M} P(\tilde{y}|\tilde{x}, \theta^{(m)}), \quad \theta^{(m)} \sim P(\theta|\mathcal{D}) \tag{1}$$

where $\{\theta^{(m)}\}_{m=1}^M$ is a set of posterior samples drawn from $P(\theta|\mathcal{D})$, e.g. by the popular Markov Chain Monte Carlo (MCMC) methods. Traditional MCMC methods either have low-efficiency for high dimensional sampling or scale poorly with dataset. Fortunately, the recently developed SG-MCMC methods work on stochastic gradients over small mini-batches, which alleviate these problems and can be applied for posterior sampling for DNNs.

3.1. Sampling via Stochastic Gradient Langevin Dynamics

Specifically, we choose the simplest and most widely used SG-MCMC algorithm - *Stochastic Gradient Langevin Dynamics* (SGLD) [12] as the sampling method in our first stage of learning ensembles. Extension by using other high-order SG-MCMC algorithms is straightforward. SGLD calculates a stochastic gradient of negative log posterior based on S_t, small mini-batch of training data:

$$\widetilde{g}_t \triangleq \nabla_\theta \widetilde{U}_t(\theta) = -\frac{N}{|S_t|}\sum_{i\in S_t} \nabla_\theta \log P(y_i|x_i, \theta) - \nabla_\theta \log P(\theta) \tag{2}$$

where $U(\theta) \triangleq -\log P(\mathcal{D}|\theta) - \log P(\theta)$ is known as the potential energy in SG-MCMC sampling and $\widetilde{U}_t(\theta)$ is its approxmation over the t-th mini-batch. The updating rule of SGLD is as simple as SGD with an additional Gaussian noise $\xi \sim \mathcal{N}(0, \epsilon_t \mathrm{I})$ as following:

$$\theta_{t+1} = \theta_t - \frac{\epsilon_t}{2}\widetilde{g}_t + \xi \tag{3}$$

where ϵ_t is the learning rate or step size. By using gradient information and stochastic mini-batch updating, SGLD overcomes the problems in traditional MCMC methods and thus leads to efficient posterior sampling.

In the following, we provide three discussions about applying SGLD to learning ensembles: (1) We use constant learning rate in this work instead of annealing learning rate as in [12]. Since constant learning rate is found to give better mixing rate and make more extensive exploration of parameter space [22], which meets our aim of collecting diverse models for a good ensemble. (2) We choose thinned collection of samples $\{\theta_{\frac{kT}{M}}\}_{k=1}^M$ from the parameter updating sequence $\{\theta_t\}_{t=1}^T$, since there are lower correlations of thinned samples which is supported by our preliminary results as well as the results from [5]. (3) We need to consider how long to run the sampling algorithm and how many models are used for model averaging. The fixed-scale additional noise in SGLD generally reduces overfitting, thus longer running can be allowed in order to better explore the parameter space. As shown by the empirical result in Fig. 1(b), the SGLD learning method indeed can improve performance by averaging more models than other traditional methods of learning ensembles.

3.2. Pruning and Retraining

After all the models are collected, we come to the second stage of learning DNN ensembles - network pruning and retraining. We use a simple pruning rule, i.e. finding the network connections whose weights are below certain threshold and removing them away, as did in [19]. The threshold is determined by the configured overall sparsity or pruning ratio, e.g. 90%, after sorting the weights by their absolute values.

Once the network is pruned, the posterior changes from $P(\theta|\mathcal{D})$ to the reduced posterior $Q^{(m)}(\phi^{(m)}|\mathcal{D})$, where m is the index of the pruned network. Retraining is then performed for each pruned network:

$$\hat{\phi}^{(m)} = \arg\max_{\phi^{(m)}} \log Q^{(m)}(\phi^{(m)}|\mathcal{D}), \quad m = 1, \ldots, M \tag{4}$$

We thus obtain an ensemble of networks $\{\hat{\phi}^{(m)}\}_{m=1}^{M}$, which are in fact maximum a posterior (MAP) estimates under the reduced posteriors.

The effect of pruning is to reduce the model size as well as the computation cost. Interestingly, it is found in our experiments that retraining of the sampled models, whether being pruned or not, significantly improve the performance of the ensemble. There are two justifications for the retraining phase. First, theoretically (namely with infinite samples), model averaging according to Equ. (1) does not need retraining. However, the actual number of samples used in practice is rather small for computational efficiency. So retraining essentially compensates for the limited size of samples for model averaging. Second, if we denote by $\bar{\phi}^{(m)}$ the network obtained just after pruning but before retraining, it can be seen that the MAP estimate $\hat{\phi}^{(m)}$ is more likely than $\bar{\phi}^{(m)}$ under the reduced posterior. Note that the probability of $\bar{\phi}^{(m)}$ under the reduced posterior is close to the probability of $\bar{\phi}^{(m)}$ under the original posterior, since we only prune small network weights. So retraining increases the posteriori probabilities of the networks in the ensemble and hopefully improves the prediction performance of the networks in the ensemble.

4. SPARSE STRUCTURED ENSEMBLES

The main computation for training or testing a DNN is the large amount of matrix calculations, which are commonly accelerated by using a GPU hardware. However, a randomly pruned network is not friendly for GPUs to handle, since the randomly positioned zeros in the weight matrices still require *floating-point operations* (FLOPs) without special treatment. In our *Sparse Structured Ensembles* (SSEs), we take this into consideration and aim at learning structures for reducing FLOPs in the sense of matrix calculations.

4.1. Group Sparse Prior (GSP)

In optimization, a regularization term is often used as a penalty to the objective function to do trade-off between minimizing a loss function and choosing a desirable model with certain constraints. The group Lasso regularization [14] proposes to do feature selection in group level, which means keeping or removing all the parameters in a group simultaneously to achieve structured sparsity corresponding to grouping strategy. It can be formulated as:

$$R(\theta) = \lambda \sum_{g=1}^{G} \sqrt{dim(\theta_g)} \left\|\theta_g\right\|_2 \tag{5}$$

where θ_g is a group of weights in θ, G is the number of groups, $dim(\theta_g)$ denotes the number of weights in θ_g and $\|\cdot\|_2$ denotes the l_2 norm. The term $\sqrt{dim(\theta_g)}$ ensures that each group gets regularized uniformly corresponding to its dimension. The coefficient λ, called the strength coefficient, is a hyperparameter to do trade off between gaining group sparsity and minimizing the loss function. While in training, the gradient of each component can be calculated by

$$\frac{\partial \sqrt{dim(\theta_g)} \left\|\theta_g\right\|_2}{\partial \theta_g} = \sqrt{dim(\theta_g)} \frac{\theta_g}{\|\theta_g\|_2} \tag{6}$$

A small constant could be added to $\|\theta_g\|_2$ in order to avoid the denominator being zero. In our experiments, we find $\|\theta_g\|_2$ fluctuate near zero and thus do not add the constant.

In Bayesian inference framework, the regularization term corresponds to the negative log prior term $-\log P(\theta)$ in the potential energy $U(\theta)$, thus the group Lasso regularization can be converted into a specific prior as follows:

$$P(\theta) = \frac{1}{Z} \exp(-R(\theta)) \tag{7}$$

where Z is a normalization constant. The gradient term $-\nabla_\theta \log P(\theta)$ in Equ. (2) for SGLD parameter updating can be directly calculated via Equ. (7), without the use of complex hierarchical decomposition form for the prior as the variational methods do [17, 18]. We call it a *Group Sparse Prior* (GSP) as named in [18].

4.2. Grouping Strategies

To learn sparse structured networks for our SSE, it is necessary to specify grouping strategy according to the characteristics of different types of neural networks. In this paper, we show how to learn SSE for both FNN and LSTM. Their grouping strategies are described separately in the following.

Feed-forward Neural Network: For FNN, we group all the outgoing connections from a single neuron (input or hidden) together following [23]. Since FNN's simple hierarchical structure, if a neuron's outputs are all zeros, it makes no contribution to the next layer and can be removed. This leads to node pruning instead of random connection pruning, which reduces the rows and columns of weight matrices between layers, thus leading to lower matrix-level FLOPs as expected. We can also group the incoming connections of a neuron, but the neurons with no incoming weights are still required to shift their biases to the next layer, which is a bit more complex than the above strategy we choose.

Long Short-term Memory: The case is not that simple for LSTM, since the input and hidden units are used four times when calculating the input gate, forget gate, cell updates and output gate as follows:

$$\begin{aligned} \boldsymbol{f}_t &= \sigma([\boldsymbol{x}_t, \boldsymbol{h}_{t-1}]W_f + \boldsymbol{b}_f) & \boldsymbol{i}_t &= \sigma([\boldsymbol{x}_t, \boldsymbol{h}_{t-1}]W_i + \boldsymbol{b}_i) \\ \boldsymbol{u}_t &= \tanh([\boldsymbol{x}_t, \boldsymbol{h}_{t-1}]W_u + \boldsymbol{b}_c) & \boldsymbol{o}_t &= \sigma([\boldsymbol{x}_t, \boldsymbol{h}_{t-1}]W_o + \boldsymbol{b}_o) \\ \boldsymbol{c}_t &= \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{u}_t & \boldsymbol{h}_t &= \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \end{aligned} \tag{8}$$

where all the vectors are row vectors, $\sigma(\cdot)$ is the sigmoid function, $[\cdot,\cdot]$ denotes concatenating horizontally and $\odot$ is element-wise multiplication. Removing an input or hidden unit is difficult for LSTM since every unit affects all the updating steps. However, note that the weight matrix between units and each gate is fully-connected, it is still beneficial to reduce the matrix size by removing a row or column. Thus a straightforward grouping strategy is to group each row and each column for the four weight matrices separately in Equ. (8), which we call an *untied* strategy. Specifically, we keep two index lists during pruning to record the remained rows and columns

for each weight matrix. When doing computations, we just use partial units to update partial dimensions of the gates according to the index lists. This is flexible for different units to provide updating for different gate dimensions.

In a concurrent work of learning structurally sparse LSTMs [16] using SGD, a grouping strategy, called *Intrinsic Sparse Structures* (ISS), is proposed to reduce the hidden size by grouping all the weights associated with a certain hidden unit together and removing them simultaneously. LSTMs learned by ISS can be reconstructed easily with the pruned smaller hidden size, without the need to keep original model size and index lists. However, the embedding size is not reduced in [16], which leads to high cost in computing the input for the 1st LSTM layer. To overcome this, there are two schemes. (a) Each column of the input embedding matrix is grouped to further reduce the input size of the 1st LSTM layer; (b) The weights from the embedding layer and the softmax layer are shared, as proposed in [11], thus the embedding size is the same as hidden size of the last LSTM layer.

5. EXPERIMENTS

In our experiments, we implemented the proposed method in TensorFlow, and present the results in two parts: *(i)* learn SSE of FNNs for image classification task on MNIST; *(ii)* learn SSE of LSTM models, which is more challenging, for word-level language modeling task on Penn TreeBank corpus.

The sparsity of a network in this paper means the percentage of the pruned weights from the total parameters, FLOPs for a matrix W is calculated as the size of the smallest sub-matrix formed by such rows and columns in W that contain all non-zero elements in W, and FLOPs for a network is the sum of FLOPs for all its weight matrices. The parameters and FLOPs presented in the following tables are the total size considering all the models in an ensemble, unless otherwise indicated. PR, GSP and SSE denotes Pruning and Retraining, Group Sparse Prior and Sparse Structured Ensemble respectively.

5.1. Classification on MNIST

First we use our method on FNNs for classification on the well-known MNIST dataset. We choose a commonly used network structure of 2 hidden layers with 300 and 100 hidden neurons respectively and ReLU activations, denoted as FNN-784-300-100-10. We run our experiments without any additional tricks such as dropout, batch normalization etc. Such basic setting allows easy reproduction of the results. All the results reported in table 1 are averaged results from 10 independent runs.

The baseline FNN-784-300-100-10 is trained by Stochastic Gradient Descent (SGD). Specifically it is trained for 100 epochs with an annealing learning rate of 0.5 which decays by a factor of 2 every 10 epoch. The baseline obtains 1.66% test error rate, and an ensemble of 18 independently trained FNN-784-300-100-10 networks decrease the error to 1.49%.

In the first group of experiments with SGLD learning, we use Laplace priors, similar to adding L1 regularization. We train also for 100 epochs but with a constant learning rate of 0.5. Network samples are collected every 5 epoch after a 10 epoch burn in. The ensemble learned by SGLD gives 1.53% test error, which is slightly worse than the independently trained ensemble, since each sample drawn by constant-step-size SGLD sampling is not as accurate as the sample trained through SGD optimization. Adding L1 could enforce sparse structure learning and allows us to prune the network weights. After pruning, we retrain each network in the ensemble for 20 epochs with a small learning rate of 0.01 which decay by factor 1.15 every epoch. The resulting ensembles are denoted by SGLD+L1+PR in Table 1. For these ensembles, the highest sparsity without losing accuracy is 90%. When 96% of the parameters are pruned, the performance is worsened obviously.

Table 1. MNIST results of various models based on FNN-784-300-100-10. The number of parameters and FLOPs are shown as multiples of the baseline FNN trained by SGD. PR: Pruning and Retraining, GSP: Group Sparse Prior.

Method	Model	Params	FLOPs	Test Error (%)
SGD (baseline)	1 model	1*	1*	1.66
SGD	18 models	18×	18×	1.49
SGLD	18 models	18×	18×	1.53
SGLD+L1+PR	18 models†	1.8×	3.4×	1.26
SGLD+L1+PR	18 models‡	0.7×	3.0×	1.39
SGLD+GSP+PR	18 models†	1.8×	2.5×	**1.26**
SGLD+GSP+PR	18 models‡	**0.7**×	**2.2**×	1.29

* The baseline model has 266K parameters and 532K FLOPs.
† indicates 90% sparsity and ‡ indicates 96% sparsity for each model.

In the second group of experiments with SGLD learning, our new method, SGLD with group sparse prior (GSP) is applied. The resulting ensembles are denoted by SGLD+GSP+PR in Table 1. When compared to SGLD+L1+PR, the new method achieves larger sparsity (up to 96%) and FLOP reduction without losing accuracy, presumably because applying GSP forces the pruned connections to be aligned, thus removes more neurons. When compared to the baseline FNN, the SSE of 18 networks learned by the new method decreases test error from 1.66% to 1.29% with 70% of parameters and $2.2\times$ computational cost.

5.2. Language Modeling

Next, we experiment with the more challenging task of learning ensembles of LSTMs, which represent a widely used type of recurrent neural networks (RNNs) for sequence learning tasks. Specifically, we study the task of LSTM-based language modeling (LM), which basically is to predict the next word given previous words. The prediction performance is measured by perplexity (PPL), which is defined as the exponential of negative log-probability per token. A popular LM benchmarking dataset - Penn TreeBank (PTB) corpus [9] is used, with a vocabulary of 10K words and 929K/73K/10K words in training, development and test sets respectively.

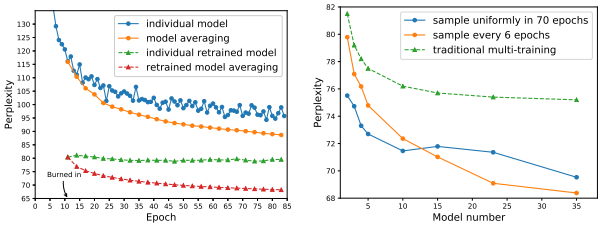
We use [10] as the baseline and follow their LSTM architectures to make comparable results. We test different methods on the medium (2 layers with 650 hidden units each) and large (2 layers with 1500 hidden units each) LSTM models as used in [10]. The dimension of word embedding as input is the same as the size of hidden units. All the models are trained with the dropout technique introduced in [10].

The results are organized into four parts. (1) For ablation analysis, we study the contribution of each component in the new method SGLD+GSP+PR through a series of comparison experiments, as shown in Table 2. (2) We show the effects of the number of model samples and sampling strategies for the method SGLD+GSP+PR, as given in Fig. 1. (3) Main results are summarized and compared in Table 3. Table 2 lists the model ablation results of SGLD training with different combinations of pruning, retraining and GSP (untied grouping strategy as default). It is found that applying PR or GSP alone lead to worse performance than vanilla SGLD for learning LSTM ensembles. When they are applied together, GSP forces

Table 2. Model ablations for the proposed method based on the medium LSTM models over PTB. All the ensembles have 10 models, and the column of single model denotes the lowest PPL obtained by a single model in the ensemble. The number of parameters and FLOPs are shown as multiples of the baseline medium LSTM. The grouping strategy is the untied strategy by default, and ISS denotes *Intrinsic Sparse Structures* as in [16].

Method	Params	FLOPs	Single model		Ensemble	
			Dev.	Test	Dev.	Test
SGD [10]	1*	1*	86.2	82.1	-	-
SGD [10]	10×	10×	-	-	75.2	72.0
SGLD	10×	10×	87.0	83.7	80.5	78.9
SGLD+PR	1×	10×	103.8	100.2	91.1	89.4
SGLD+GSP	10×	10×	98.8	97.0	88.0	86.9
SGLD+GSP+R	10×	10×	80.0	76.3	70.8	69.1
SGLD+GSP+P	1×	4×	103.8	101.9	96.1	94.7
SGLD+GSP+PR	1×	4×	79.8	76.6	71.5	69.5
SGLD+GSP+PR†	1×	3×	80.9	77.4	71.8	69.9

* The baseline LSTM model has 19.8M parameters and 26.5M FLOPs.
† indicates the ISS group strategy.

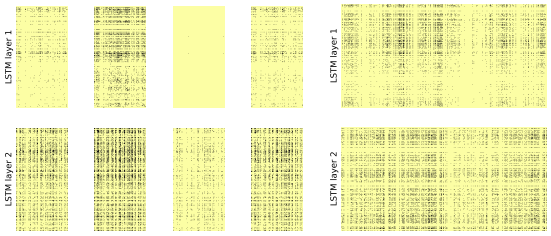


(a) (b)

Fig. 1. (a) The PPL curves along the learning of LSTM ensemble by SGLD+GSP+PR over the PTB development set. (b) The PPLs over development set v.s. the number of models in the LSTM ensembles by SGLD+GSP+PR.

the network to learn group sparse structures which are highly robust to pruning, thus leading to a better result. With GSP, applying pruning only leads to negligible loss of performance but greatly reduces the model size, as can been seen from comparing SGLD+GSP+PR with SGLD+GSP+R; pruning without retraining produces inferior result. Remarkably, compared to the medium LSTM ensemble obtained by multiple training, the ensemble learned by the new method SGLD+GSP+PR reduces the PPL from 72.0 to 69.5, and with only 10% parameters and 40% FLOPs in total. SGLD indeed provides a good approach to finding diverse sample models for ensembles.

Fig.1(a) presents the PPL curves along the learning of LSTM ensemble by the new method SGLD+GSP+PR over the development set. It clearly shows the performance gains brought by model averaging and PR. The relationship between the performance of an ensemble and the number of models in an ensemble is examined in Fig.1(b), together with a comparison between different sampling strategies. We test the performances of different ensembles on the PTB development set, each consisting of 2 to 35 medium LSTMs. The blue curve shows the result of SGLD+GSP+PR, which runs 10-epoch burn-in and then samples uniformly for 70 epochs. The orange curve is obtained by sampling every 6 epochs; thus the more models collected, the longer run of SGLD. It can be seen from comparing the two curves that for a fixed number of models, sampling with



(a) Untied weights (b) ISS

Fig. 2. The sparse structure patterns of LSTM weight matrices. Yellow areas are all zeros while black dots stand for non-zero weights. The patterns are plotted with a sub-sampling by a factor of 5.

Table 3. Comparison of various models based on LSTMs on PTB dataset. The number of parameters and FLOPs are shown as multiples of the baseline medium LSTM trained by SGD, the specifics of which are shown in parentheses. The bold line denotes the best result obtained with shared embeddings (denote as SE).

Method	Model	Params	FLOPs	Dev.	Test
SGD [10]	1 large	1*	1*	82.2	78.4
SGD [10]	38 large	38×	38×	71.9	68.7
VD [24]	10 large	10×	-	-	68.7
VD+SEAL [11]	individual	51M	-	71.1	68.5
AWD [25]	individual	24M	-	60.0	57.3
SGD+GSP+PR	1 large	0.1×	0.2×	81.9	77.8
SGD+GSP+PR	4 large	0.4×	0.9×	69.9	66.7
SGLD+GSP+PR	20 large	2.0×	4.5×	68.6	66.4
SGLD+GSP+PR	4 large	0.4×	0.9×	70.9	68.7
SGLD+GSP+PR†	4 large	0.4×	0.5×	72.2	69.7
SGLD+GSP+PR+SE†	4 large	**0.3**×	**0.7** ×	**64.4**	**62.1**

* The baseline LSTM model has 66M parameters and 102M FLOPs.
† indicates the ISS group strategy.

larger intervals performs better. It is also clear from Fig.1(b) that the traditional ensemble learning method by SGD training of multiple models is inferior to the SGLD learning method.

Fig. 2 shows the sparse structured patterns of the weight matrices from a single LSTM model sample in the ensemble trained by applying SGLD with GSP, separately for two different grouping strategies of weight matrices.

Comparison of various models based on LSTMs on PTB dataset are summarized in Table 3. We investigate to use small number of models to achieve trade off between cost and performance. An attractive model is the SSE of 4 large LSTMs, which only requires 40% of parameters and 90% of computational cost in total, compared to the baseline large LSTM [10], but decrease the perplexity from 78.4 to 68.7. This result is also better than those obtained by [10] (38 independently trained large LSTMs) and [24] (10 independently trained large LMs with costly MC dropout for testing), not only in terms of PPL reduction but also in term of reducing memory and computing costs.

As suggested by a referee, we compare SGD (1 model)+GSP+PR with SGLD (ensemble)+GSP+PR. SGD+GSP+PR represents the SGD training with group sparse prior and model pruning/retraining. SGD (1 model)+GSP+PR can reduce the model size but the PPL is much worse than the ensemble, which clearly shows the improvement provided by the ensemble. Additionally, we compare SGLD (4 models)+GSP+PR with SGD (4 models)+GSP+PR, namely the classic ensemble training method by multiple independent runs with

different initializations. The two ensembles achieve close PPLs. However, SGD ensemble learning requires 30×4 epochs training and 15×4 epochs retraining, SGLD ensemble learning takes 80 epochs training plus 15×4 epochs retraining, which reduces about 30% training time.

Note that a number of better model architectures [11, 25] have emerged than the baseline large LSTM LM, which we used as a baseline. Our SGLD+GSP+PR in principle can be applied to those new model architectures. As an example, we apply SGLD+GSP+PR to models that share input and output embeddings [11]. With shared embeddings, we further reduce the perplexity to 62.1 by using the SSE of 4 large LSTMs, which can be regarded as an ensemble version of [11] without variational dropout (VD) and augmented loss (AL). As a side note, without shared embeddings, the lowest perplexity achieved is 66.4 by the SSE of 20 large LSTMs, which is also among the top models obtained with standard LSTMs to the best of our knowledge.

6. CONCLUSION AND FUTURE WORKS

In this work, we propose a novel method of learning NN ensembles efficiently and cost-friendly by integrating three mutually enhanced techniques: SG-MCMC sampling, group sparse prior and network pruning. The resulting SGLD+GSP+PR method is easy to implement, yet surprisingly effective. This is thoroughly evaluated in the experiments of learning SSE ensembles of both FNNs and LSTMs. The Sparse Structured Ensembles (SSEs) learned by our method gain better prediction performance with reduced training and test cost when compared to traditional methods of learning NN ensembles. Moreover, by proper controlling the number of models used in the ensemble, the method can also be used to produce SSE, which outperforms baseline NN significantly without increasing the model size and computation cost.

Some interesting future works: (1) interleaving model sampling and model pruning; (2) application of this new method, as a new powerful tool of learning ensembles, to more tasks.

7. REFERENCES

[1] Cheng Ju, Aurélien Bibaut, and Mark J van der Laan, "The relative performance of ensemble methods with deep convolutional neural networks for image classification," *arXiv preprint arXiv:1704.01664*, 2017.

[2] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger, "Snapshot ensembles: Train 1, get m for free," *arXiv preprint arXiv:1704.00109*, 2017.

[3] Lars Kai Hansen and Peter Salamon, "Neural network ensembles," *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, no. 10, pp. 993–1001, 1990.

[4] Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling, "Bayesian dark knowledge," in *Advances in Neural Information Processing Systems*, 2015, pp. 3438–3446.

[5] Zhe Gan, Chunyuan Li, Changyou Chen, Yunchen Pu, Qinliang Su, and Lawrence Carin, "Scalable bayesian learning of recurrent neural networks for language modeling," *arXiv preprint arXiv:1611.08034*, 2016.

[6] Ilya Loshchilov and Frank Hutter, "Sgdr: stochastic gradient descent with restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[7] Li Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[8] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini, "Building a large annotated corpus of english: The penn treebank," *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[10] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[11] Hakan Inan, Khashayar Khosravi, and Richard Socher, "Tying word vectors and word classifiers: A loss framework for language modeling," *arXiv preprint arXiv:1611.01462*, 2016.

[12] Max Welling and Yee W Teh, "Bayesian learning via stochastic gradient langevin dynamics," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, 2011, pp. 681–688.

[13] Tianqi Chen, Emily Fox, and Carlos Guestrin, "Stochastic gradient hamiltonian monte carlo," in *International Conference on Machine Learning*, 2014, pp. 1683–1691.

[14] Ming Yuan and Yi Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 68, no. 1, pp. 49–67, 2006.

[15] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li, "Learning structured sparsity in deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 2074–2082.

[16] Wei Wen, Yuxiong He, Samyam Rajbhandari, Wenhan Wang, Fang Liu, Bin Hu, Yiran Chen, and Hai Li, "Learning intrinsic sparse structures within long short-term memory," *arXiv preprint arXiv:1709.05027*, 2017.

[17] S Derin Babacan, Shinichi Nakajima, and Minh N Do, "Bayesian group-sparse modeling and variational inference," *IEEE transactions on signal processing*, vol. 62, no. 11, pp. 2906–2921, 2014.

[18] Benjamin M Marlin, Mark Schmidt, and Kevin P Murphy, "Group sparse priors for covariance estimation," in *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2009, pp. 383–392.

[19] Song Han, Jeff Pool, John Tran, and William Dally, "Learning both weights and connections for efficient neural network," in *Advances in Neural Information Processing Systems*, 2015, pp. 1135–1143.

[20] Song Han, Junlong Kang, Huizi Mao, Yiming Hu, Xin Li, Yubin Li, Dongliang Xie, Hong Luo, Song Yao, Yu Wang, et al., "Ese: Efficient speech recognition engine with sparse lstm on fpga.," in *FPGA*, 2017, pp. 75–84.

[21] Yee Whye Teh, Alexandre H Thiery, and Sebastian J Vollmer, "Consistency and fluctuations for stochastic gradient langevin dynamics," *Journal of Machine Learning Research*, vol. 17, pp. 1–33, 2016.

[22] Issei Sato and Hiroshi Nakagawa, "Approximation analysis of stochastic gradient langevin dynamics by using fokker-planck equation and ito process," in *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014, pp. 982–990.

[23] Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini, "Group sparse regularization for deep neural networks," *Neurocomputing*, vol. 241, pp. 81–89, 2017.

[24] Yarin Gal and Zoubin Ghahramani, "A theoretically grounded application of dropout in recurrent neural networks," in *Advances in neural information processing systems*, 2016, pp. 1019–1027.

[25] Stephen Merity, Nitish Shirish Keskar, and Richard Socher, "Regularizing and optimizing lstm language models," *arXiv preprint arXiv:1708.02182*, 2017.