A Diversity-Penalizing Ensemble Training Method for Deep Learning

Xiaohui Zhang[1], *Daniel Povey*[1,2], *Sanjeev Khudanpur*[1,2]

[1]Center for Language and Speech Processing
[2]Human Language Technology Center of Excellence
The Johns Hopkins University, Baltimore, MD 21218, USA
xiaohui@jhu.edu, danielpovey@gmail.com, khudanpur@jhu.edu

Abstract

A common way to improve the performance of deep learning is to train an ensemble of neural networks and combine them during decoding. However, this is computationally expensive in test time. In this paper, we propose an diversity-penalizing ensemble training (DPET) procedure, which trains an ensemble of DNNs, whose parameters were differently initialized, and penalizes differences between each individual DNN's output and their average output. This way each model learns to emulate the average of the whole ensemble of models, and in test time we can use one arbitrarily chosen member of the ensemble. Experimental results on a variety of speech recognition tasks show that this technique is effective, and gives us most of the WER improvement of the ensemble method while being no more expensive in test time than using a single model.

Index Terms: deep learning, ensemble learning, model compression, speech recognition.

1. Introduction

A simple way to improve the performance of deep learning is to train an ensemble of neural networks which are randomly initialized with different parameters on the same data, and then average their predictions during decoding [1]. More powerful ensemble methods aim to produce an accurate yet diverse ensemble of models, by encouraging diversity among models in the ensemble during training, and combine them during decoding [2]. Since using an ensemble at test time is expensive, people have applied model compression techniques to compress the ensemble into a single model. Various model compression techniques [3] [4] have shown that it is possible to compress the knowledge in an ensemble into a single model which is much cheaper to deploy. In this paper, rather than using ensemble methods to produce a diverse ensemble of models and then explicitly compress them into one model, we enforce interaction among the models during training to drive them towards their average in the models' output space, so that at the end, each single model's performace gets close to the ensemble's performance. In other words, we want to achieve model compression as part of the process of training the individual models. Note that we already briefly mentioned an earlier version of this method in [5]. In this paper we describe the idea in more detail and provide experimental results to support its effectiveness.

The rest of the paper is organized as follows. In Section 2 we provide details of our training procedure. In Section 3.1 we state that the proposed ensemble method can improve models' generalization performance since it acts as a good regularizer. In Section 3.2 we analyze the relationship between our method and other ensemble methods. In Section 4 we discuss our experimental conditions; we present the results in Section 5.

2. Diversity-penalizing ensemble training

2.1. Objective function

We initialize each neural network in an ensemble of size N with different random parameters, then we train each network using mini-batch SGD on the same sequence of training examples (pre-randomized). On each training example, let $\mathbf{p^i}$ denote the posterior of the i'th model in the ensemble, and $\mathbf{q}$ denote the zero-one training targets. For the i'th model, instead of minimizing the cross entropy between $\mathbf{p^i}$ and $\mathbf{q}$:

$$\mathbf{CrossEnt}(\mathbf{q}, \mathbf{p^i}) \tag{1}$$

as we usually do, our DPET method adds a penalty term to the objective function, which is the KL divergence between the individual posterior $\mathbf{p^i}$ and the averaged posterior:

$$\bar{\mathbf{p}} = \frac{1}{N}\sum_{j=1}^{N}\mathbf{p^j} \tag{2}$$

and we jointly optimize the sum of the objective functions for each individual model in the ensemble:

$$F = \sum_{i=1}^{N}\Big(\mathbf{CrossEnt}(\mathbf{q}, \mathbf{p^i}) + \lambda\mathbf{D}(\bar{\mathbf{p}}||\mathbf{p^i})\Big) \tag{3}$$

The concept is that in addition to learning the original training labels $\mathbf{q}$ we are dynamically training the models toward each other, by penalizing the KL divergence between each individual model's output and the average output. This is how DPET penalizes diversity in the ensemble. In [4], a number of models are first trained independently, and then another model is trained using an interpolation of the hard labels and the averaged predictions of the previously mentioned models. In DPET we combine these two stages into a single procedure.

2.2. Derivative computation

Here we show, with the new objective function F, the derivatives could still be computed efficiently. Using n to specify a certain dimension of a posterior, the objective F can be writen as:

$$F = \sum_{i=1}^{N}\left(-\sum_{n}\mathbf{q_n}log(\mathbf{p_n^i}) + \lambda\sum_{n}\bar{\mathbf{p}}_\mathbf{n}log\frac{\bar{\mathbf{p}}_\mathbf{n}}{\mathbf{p_n^i}}\right) \tag{4}$$

So, the derivative of F w.r.t the nth dimension of jth network's posterior is:

$$\frac{\partial F}{\partial \mathbf{p_n^j}} = \left(-\frac{\mathbf{q_n}}{\mathbf{p_n^j}}\right) + (\lambda(1 + log(\bar{\mathbf{p}}_\mathbf{n}))) \quad (5)$$

$$- \left(\frac{\lambda}{N} log(\mathbf{p_n^j}) + \lambda \frac{\bar{\mathbf{p}}_\mathbf{n}}{\mathbf{p_n^j}} + \frac{\lambda}{N} \sum_{i!=j} log(\mathbf{p_n^i})\right) \quad (6)$$

$$= - \frac{\mathbf{q_n} + \lambda \bar{\mathbf{p}}_\mathbf{n}}{\mathbf{p_n^j}} + \lambda(1 + log(\bar{\mathbf{p}}_\mathbf{n})) - \lambda \overline{log(\mathbf{p_n})}) \quad (7)$$

We can ignoring the constant term, which will disappear after backpropgating through softmax. And considering we are maximizing $-F$, we get the vector form derivative:

$$\frac{\partial(-F)}{\partial \mathbf{p^j}} = \frac{\mathbf{q} + \lambda \bar{\mathbf{p}}}{\mathbf{p^j}} + \lambda(\overline{log(\mathbf{p})} - log(\bar{\mathbf{p}})) \quad (8)$$

where

$$\overline{log(\mathbf{p})} = \frac{1}{N} \sum_{j=1}^{N} log(\mathbf{p^j}) \quad (9)$$

Computationally, with the standard cross entropy objective, DPET is equivalent to replacing the target $\mathbf{q}$ with the modified target:

$$\tilde{\mathbf{q}} = \mathbf{q} + \lambda \bar{\mathbf{p}} + \mathbf{p^j} \circ (\lambda(\overline{\mathbf{log(p)}} - \mathbf{log}(\bar{\mathbf{p}}))) \quad (10)$$

where $\circ$ means entry-wise product.

2.3. Schedule for λ

In our method, the λ parameter is empirically determined. We found that increasing λ as training progresses always gave more improvements than keeping λ constant. This makes sense because during early iterations of training, we should not put emphasis on letting individual models learn from each other, since they perform too poorly to be worthwhile learning from. It also makes sense from the regularization point of view, since overffiting becomes a more serious issue at later stage of training, which our method aims to relieve. We also found that the performance is not sensitive to the specific schedule of λ, e.g. whether we increase it linearly or exponentially. For simplicity we just adopted a linear λ schedule determined by the initial/final lambda values, λ_{init} and λ_{final}, and tuned these two values. It turns out that a small λ_{init} like 0.1 or 0.01 and a λ_{final} between 3 and 5 tends to perform well.

2.4. Ensemble size

Another parameter of our recipe is the ensemble size N. The larger the ensemble size is, the more significant DPET's effect will be. This is a tradeoff between performance and computational costs. In practice, we find that setting the ensemble size as 4 to 8 is usually enough, and making N larger than this does not bring noticeable improvements. When we finish training, unless stated otherwise, we arbitrarily choose one model from the ensemble for decoding, so in test time we only need to evaluate one model.

3. Related work

3.1. Diversity-penalizing ensemble training as a good regularizer

We argue that the proposed diversity-penalizing ensemble training method is a sensible regularizer during training, since it says: "We believe a priori that the predictions should be independent of the initialization". The same principle is also used by conventional ensemble methods where we just train differently initialized models separately and then average their outputs during decoding. In our method, individual models in the ensemble were initialized with different random parameters and were trained on the same data simultaneously. Thus, during each training iteration, the only source of diversity in the models' outputs on each training example is the difference in initialization. By penalizing the diversity of models' outputs, we are hoping that, as training goes on, models in the ensemble will behave more and more similarly, and the variance caused by initialization can be reduced. Note that our method improves generalization performance by reducing the variance brought by the initialization of a model, not by reducing the variance brought by training data as in bagging [6].

From another perspective, we can also say our regularizer does implicit model averaging by driving individual models towards their average in the models' output (prediction) space. In this sense, it has a similar motivation as dropout [7], which does sub-model averaging by removing a random subset of the model parameters during each training iteration. We don't provide an experimental comparision with dropout here, since previous experiments with dropout on speech recognition tasks in our setup did not show any improvements.

3.2. Relationship with other ensemble methods

Ensemble methods have been widely used to improve the generalization performance of a single model. In a practical ensemble learning scenario, people either train an ensemble of models on the same data separately and then combine them during decoding like [8], or encourage some kind of diversity among models in the ensemble during training. For example, bagging [6] generates diversity by bootstrap sampling of training data, boosting-type [9] algorithms encourage diversity by incrementally building an ensemble, training each new model to emphasize the accuracy on subsets of the training examples which previous models have mis-classified. Also there have been efforts on explicitly encouraging diversity in the ensemble during training. Among them the most notable one is Negative Correlation Learning (NCL) [2] algorithm and related methods, whose aim was to obtain highly correct models that disagree as much as possible. They emphasize interaction among the individual models in the ensemble to increase diversity among them, by introducing unsupervised penalty terms in the objective function during training. Although positive experimental results were shown, the difficulty of defining diversity, and the basic assmuption that explicitly encouraging diversity in an ensemble benefits learning, have been problematic [10]. Also it has been pointed out that these methods suffer from over-fitting the training data [11].

Similar to NCL, our proposed ensemble method also emphasizes interaction among individual models during training. However the effect of the interaction has the opposite sign: we penalize diversity among models in the ensemble, rather than encouraging it (here, diversity refers to the diversity in models' predictions/outputs). The reason is that our motivation is different: rather than aiming to produce an accurate yet diverse ensemble of models and combine them during decoding, we are aiming to deploy an interative ensemble training procedure to achieve model compression/knowledge distillation [4] while training, or more specifically, to gradually enforce each single model to behave as powerful as the ensemble.

4. Experimental conditions

All experiments were done using the Kaldi open source speech recognition toolkit [12]. We did some preliminary experiments on IARPA Babel tasks to show the effectiveness of a vanilla version DPET on limited resource ASR tasks and did more experiments on TED-LIUM [13] task to show the generalization performace of DPET on various conditions in detail. The experiments on Babel limited resource ASR tasks using a similar ensemble training approach were already briefly mentioned in [5]. We build HMM-DNN Hybrid ASR systems on each language: Tamil, Assamese, Bengali, and Zulu, which are trained on an IARPA-provided limited language pack (*LimitedLP*) containing 10 hours of transcribed speech and a dictionary that covered words in the transcripts, and the WER performance was measured on IARPA-provided 10 hours of transcribed speech as the development set[1].

We used DNN with p-norm activation functions [14] with $p = 2$ and natural-gradient SGD [15] for both the baseline and DPET recipes (also for all TED-LIUM experiments). We separately tuned the networks' number of hidden layers and total number of parameters (per individual DNN) for both the baseline and ensemble training recipes, and gave the best for each; this results in more parameters per model for the ensemble systems. After tuning, all ensemble trained DNNs have 3 layers and 5.3 million parameters, and all the baseline DNNs have 2 layers and 2.53 million parameters. All DPET runs on Babel data involved 4 models in each ensemble, with $\lambda_{\text{init}} = 0.1$ and $\lambda_{\text{final}} = 5$.

For TED-LIUM experiments, all the Hybrid HMM-DNN ASR systems are trained on the standard 118 hour training set, and the performance is measured on the 1.6 hour Dev set and the 2.6 hour Test set separately. We used the time-delay neural network architecture described in [16] with 40 dimensional MFCC features and 100 dimensional i-vectors [17] computed for each speaker as inputs, which allows for very rapid speaker adaptation. For DNNs used in both baseline and DPET recipes, we tuned and then fixed the number of layers (6 layers, which gives the best results for both). However we did experiments on settings with different number of parameters, by adjusting the width of the hidden layers. By doing this we can thoroughly inspect how DPET, as a regularizer, can relieve the over-fitting problem and help to achieve good generalization performance in settings with a large number of parameters. Also we measured the final models' frame accuracy on a held-out validation set, which is a 300 utterance subset of the TED-LIUM training set. At the end, in order to inspect how DPET affects the performance gap between the ensemble and individual models, we decode using the entire DPET trained ensemble (averaging the log-pseudo-likelihoods), as well as the individual models constituting the ensemble. We do the same in the baseline setting: we perform decoding using an ensemble of separately trained DNNs (with different initial parameters), and the individual models constituting the ensemble. For this experiment DNNs are trained on the best number of parameters setting for each recipe. All DPET runs on TED-LIUM data involved 4 models in each ensemble, with $\lambda_{\text{init}} = 0.1$ and $\lambda_{\text{final}} = 4$.

[1] The exact corpus identifiers are

Assamese, `IARPA-babel102b-v0.4`;

Bengali, `IARPA-babel103b-v0.3`;

Tamil, `IARPA-babel204b-v1.1b`;

Zulu, `IARPA-babel206b-v0.1e`.

5. Experimental results

5.1. Results on BABEL limited resource speech recognition tasks

We can see from Table 1 that across four BABEL languages, DPET brings $0.9 - 1.6\%$ absolute improvement ($1.2 - 2.5\%$ relative) in WER. DPET seems to help a lot here, since for these experiments we only have 10 hours training data for each language, which means over-fitting might be a severe problem.

Language	WER (Babel Dev set)		
	Baseline	DPET	Absolute improvement
Tamil	76.6%	75.7%	0.9%
Assamese	64.5%	62.9%	1.6%
Bengali	65.9%	65.0%	0.9%
Zulu	68.7%	67.8%	0.9%

Table 1: ASR Performance (WER%) of DPET (with an ensemble size of 4) on Babel.

5.2. Results on TED-LIUM experiments

Table 2 shows how DPET affects the ASR performance on TED-LIUM Dev and Test sets as we change the number of parameters in the individual DNNs. (Note that we didn't the tune the optimal ensemble size or λ_{init} and λ_{final} for each setting.) We can see when the number of parameters is small, like 1.2 or 1.5 million, DPET seems to have little effect. This may be because for small models, there is little over-training and regularization is not needed. However, as the number of parameters increases from 1.5 million to 2.3 million, the baseline performances start to decrease, while DPET performances keep improving, and are consistently better than the baseline. Comparing the best model sizes from the baseline and DPET runs, we conclude that DPET brings 0.5% absolute improvement (2.5% relative) on the Dev set and 0.6% absolute improvement (3.3% relative) on the Test set.

# Param. (million)	WER (TED-LIUM Dev set)		
	Baseline	DPET	Absolute improvement
1.2	20.2%	20.0%	0.2%
1.5	**19.9%**	20.0 %	-0.1
1.9	19.9%	19.7%	0.2%
2.3	20.3%	19.5%	0.8%
2.8	20.0%	**19.4%**	0.6%

# Param. (million)	WER (TED-LIUM Test set)		
	Baseline	DPET	Absolute improvement
1.2	18.0%	17.7%	0.3%
1.5	**17.9%**	17.7%	0.2%
1.9	18.1%	17.6%	0.5%
2.3	18.2%	**17.3%**	0.9%
2.8	18.1%	17.4%	0.7%

Table 2: ASR performance (WER%) of DPET (with an ensemble size of 4) on TED-LIUM Dev and Test sets, on settings with different number of paremeters.

For the same sets of experiments, Tabel 3 shows the baseline and DPET models' final frame accuracy on a held-out vali-

dation set. It further verifies DPET can help prevent over-fitting when the number of parameters gets large.

# Param. (million)	Frame accuracy		
	Baseline	DPET	Absolute improvement
1.2	59.28%	59.23%	-0.05%
1.5	58.95%	60.53%	0.58%
1.9	59.43%	60.80%	1.37%
2.3	59.65%	60.42%	0.77%
2.8	59.75%	60.60%	0.85%

Table 3: Generalization performance of DPET (with an ensemble size of 4), measured by an individual final model's frame accuracy on the validation set.

For the 1.9 million parameter setting, where the difference in frame accuracy is most significant, we plot the validation frame accuracy versus iterations in Figure 1, where we can see that during later training iterations, the improvements brought by DPET start to be noticeable.

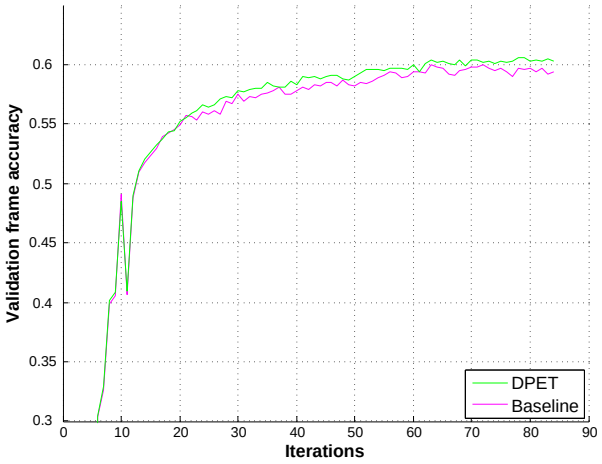

Figure 1: Frame accuracy (on validation set) vs. iterations, where # param. is 1.9 million.

Table 4 shows the performance of the ensemble and individual models constituting the ensemble, with and without DPET. We can see that DPET gives us most of the WER improvement of the ensemble method: among the four DPET trained individual models, three of them perform no worse than the baseline ensemble, which is 17.4%. Also, we notice that DPET shrinks the performance gap between the ensemble and the individual models: The gap is $0.1\% - 0.3\%$ for DPET and is $0.4\% - 0.5\%$ for the baseline.

Training procedure	WER (on TED-LIUM Test set)				
	Individual models				4xEnsemble
Baseline	17.8%	17.9%	17.9%	17.9%	17.4%
DPET	17.3%	17.4%	17.5%	17.4%	17.2%

Table 4: ASR performance of the ensemble and four individual models constituing it, trained normally or with DPET, on the tuned #parameters for each (1.5m for baseline and 2.3m for DPET).

6. Conclusions

In this paper, we proposed a Diversity-Penalizing Ensemble Training (DPET) technique for training deep neural networks. By penalizing the diversity of the outputs of individual DNNs in an ensemble, whose parameters were differently initialized, we force each model to emulate the average of the whole ensemble of models. And therefore, in test time we can use one arbitrarily chosen member of the ensemble, whose performance is close to the ensemble. We argued that DPET works as a regularizer which reduces the variance caused by model initialization, thereby improving the performance of individual models in the ensemble. This is supported by the experimental results on various speech recognition tasks. The results have also shown that this technique reduces the performance gap between the ensemble and the individual models. The performance of an individual model trained using this technique almost matches the performance of a baseline ensemble, while being no more expensive in test time than using a single model.

In future we plan to investigate more efficient ways to train ensembles of this kind.

7. Acknowledgements

Thanks to Prof. Yoshua Bengio for useful discussions. This work was partially supported by DARPA BOLT Contract No HR0011-12-C-0015, NSF Grant No IIS 0963898, and IARPA BABEL contract No W911NF-12-C-0015. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA, IARPA, DoD/ARL or the U.S. Government.

8. References

[1] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple classifier systems*. Springer, 2000, pp. 1–15.

[2] Y. Liu and X. Yao, "Ensemble learning via negative correlation," *Neural Networks*, vol. 12, no. 10, pp. 1399–1404, 1999.

[3] C. Bucilu, R. Caruana, and A. Niculescu-Mizil, "Model compression," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 535–541.

[4] O. Hinton, G. Vinyals and J. Dean, "Distilling knowledge in a neural network," in *In Deep Learning and Representation Learning Workshop, NIPS.*, 2014.

[5] J. Trmal, G. Chen, D. Povey, S. Khudanpur, P. Ghahremani, X. Zhang, V. Manohar, C. Liu, A. Jansen, D. Klakow *et al.*, "A keyword search system using open source software," in *Proc. SLT*, 2014.

[6] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[7] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[8] L. Deng and J. C. Platt, "Ensemble deep learning for speech recognition," in *Proc. Interspeech*, 2014.

[9] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.

[10] N. Garcia-Pedrajas, C. Hervas-Martinez, and D. Ortiz-Boyer, "Cooperative coevolution of artificial neural network ensembles for pattern classification," *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 3, pp. 271–302, 2005.

[11] H. Chen and X. Yao, "Regularized negative correlation learning for neural network ensembles," *Neural Networks, IEEE Transactions on*, vol. 20, no. 12, pp. 1962–1979, 2009.

[12] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlíček, Y. Qian, P. Schwarz *et al.*, "The kaldi speech recognition toolkit," *Proc. ASRU*, 2011.

[13] A. Rousseau, P. Deléglise, and Y. Estève, "Ted-lium: an automatic speech recognition dedicated corpus." in *LREC*, 2012, pp. 125–129.

[14] X. Zhang, J. Trmal, D. Povey, and S. Khudanpur, "Improving deep neural network acoustic models using generalized maxout networks," in *Proc. ICASSP*, 2014.

[15] D. Povey, X. Zhang, and S. Khudanpur, "Parallel training of deep neural networks with natural gradient and parameter averaging," *ICLR: Workshop track*, 2015.

[16] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," *Proc. Interspeech*, 2015.

[17] M. Karafiat, L. Burget, P. Matejka, O. Glembek, and J. Cernocky, "ivector-based discriminative adaptation for automatic speech recognition," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 152–157.