February 27, 2017

RT0978

Computer Science; Mathematics 8 pages

Research Report

Fast and Accurate Inference with Adaptive Ensemble Prediction in Image Classification with Neural Networks

Hiroshi Inoue

IBM Research - Tokyo
IBM Japan, Ltd.
19-21, Nihonbashi Hakozaki-cho
Chuo-ku, Tokyo 103-8510 Japan

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Fast and Accurate Inference with Adaptive Ensemble Prediction in Image Classification with Deep Neural Networks

Hiroshi Inoue
IBM Research - Tokyo
inouehrs@jp.ibm.com

Abstract

Ensembling multiple predictions is a widely used technique to improve the accuracy of various machine learning tasks. In image classification tasks, for example, averaging the predictions for multiple patches extracted from the input image significantly improves accuracy. Using multiple networks trained independently to make predictions improves accuracy further. One obvious drawback of the ensembling technique is its higher execution cost during inference. If we average 100 predictions, the execution cost will be 100 times as high as the cost without the ensemble. This higher cost limits the real-world use of ensembling, even though using it is almost the norm to win image classification competitions. In this paper, we describe a new technique called adaptive ensemble prediction, which achieves the benefits of ensembling with much smaller additional execution costs. Our observation behind this technique is that many easy-to-predict inputs do not require ensembling. Hence we calculate the confidence level of the prediction for each input on the basis of the probability of the predicted label, i.e. the outputs from the softmax, during the ensembling computation. If the prediction for an input reaches a high enough probability on the basis of the confidence level, we stop ensembling for this input to avoid wasting computation power. We evaluated the adaptive ensembling by using various datasets and showed that it reduces the computation time significantly while achieving similar accuracy to the naive ensembling.

1. Introduction

The huge computation power of today's computing systems, equipped with GPUs, special ASICs, FPGAs, or multi-core CPUs, makes it possible to train deep networks using tremendous datasets. Although such high-performance systems can be used for training, actual inference in the real world may be executed on small devices such as a handheld device or an embedded controller, which have much smaller computation power and energy supply than the large systems used for training the network. Hence, a method to achieve high prediction accuracy with limited computation resources is needed to enable more applications to be deployed in the real world. To reduce the computation costs in the inference phase, Hinton *et al*. (2015) created a smaller network for deployment by distilling the knowledge from an ensemble of multiple models. Han *et al.* (2016) also targeted deployment for small (mobile) devices and showed that large networks can be significantly compressed after training by pruning unimportant connections and by quantizing each connection.

Ensembling multiple predictions is a widely used technique to improve the accuracy of various machine learning tasks (e.g. Hansen and Salamon 1990, Zhou *et al.* 2002) at the cost of more computation power. In the image classification tasks, for example, accuracy is significantly improved by ensembling the local predictions for multiple patches extracted from the input image to make the final prediction. Moreover, accuracy is further improved by using multiple networks trained independently to make local predictions. Krizhevsky *et al.* (2012) averaged 10 local predictions using 10 patches extracted from the center and the 4 corners with and without horizontal flipping in their Alexnet paper. GoogLeNet by Szegedy *et al.* (2015) averaged up to 1,008 local predictions by using 144 patches and 7 networks. In their paper, they reported that averaging 1,008 predictions reduced the top-5 error of ImageNet classification task by 3.45% whereas averaging 10 predictions with one model reduced the error by 0.92% compared with the baseline prediction without ensembling. In some ensemble methods, meta-learning during the training to learn how to best mix the multiple local predictions from the networks is used (e.g. Tekin *et al.* 2016). In the Alexnet or GoogLeNet papers, however, the significant improvements were obtained by just averaging the predictions without the meta-learning. In this paper, we do not use meta-learning either.

Although the benefits of ensemble prediction are quite significant, one obvious drawback is its higher execution cost during inference. If we make the final prediction by ensembling 100 predictions, we need to make 100 local predictions, and hence the execution cost will be 100 times as high as that without ensembling. This higher execution cost limits the real-world use of ensembling especially on small devices, even through using it is almost the norm to win image classification competitions that emphasize prediction accuracy.

To make the ensemble prediction more feasible in a wider range of applications, we have developed adaptive ensemble prediction, which achieves the benefits of ensembling with much smaller additional costs. Our observation behind this technique is that many easy-to-predict inputs do not require ensembling. We use the output produced by the softmax, which is at the end of the neural network, for the predicted class label as the probability of the prediction. During the ensembling process, we calculate the confidence level of the probability obtained from local predictions for each input. If an input reaches a high enough confidence level, we stop ensembling and making more local predictions for this input to avoid wasting computation power. We evaluated the adaptive ensembling by using four image classification datasets: ILSVRC 2012, CIFAR-10, CIFAR-100, and SVHN. Our results showed that the adaptive ensemble prediction reduces the computation time significantly while achieving similar accuracy to the naive ensemble prediction.

2. Ensembling and Probability of Prediction

This section describes the observations that have motivated us to develop our proposed technique: how the ensemble prediction improves the accuracy of predictions with different probabilities.

To show the relationship between the probability of the prediction and the effect of ensembling, we evaluate the prediction accuracy for the ILSVRC 2012 dataset with and without ensembling of two predictions made by two independently trained networks. Figure 1(a) shows the results of this experiment with GoogLeNet; the two networks follow the design of GoogLeNet and use exactly the same configurations (hence the differences come only from the random number generator). In the experiment, we 1) evaluated the 50,000 images from the validation set of the ILSVRC 2012 dataset using the first network, 2) sorted the images by the probability of the prediction, and 3) evaluated the images with the second network and assessed the accuracy after ensembling two local predictions using the arithmetic mean. The x-axis of Figure 1(a) shows the percentile of the probability from high to low, i.e. going left (right), input images become easier (harder) to predict. The gray dashed line shows the average probability for each percentile class. On average for all images, the ensemble improves accuracy well, although we only averaged two predictions. Interestingly, we can observe that the improvements only come in the right of the figure. There are almost no improvements by ensembling two predictions on the left side, i.e. easy-to-predict images with highly probable predictions, even when there is a non-negligible number of mispredicted samples. For example, in the 50- to 60-percentile range, the error rate is 29.6% and is not improved by averaging two predictions from different networks. In this range, the average probability of prediction is 79.4%.

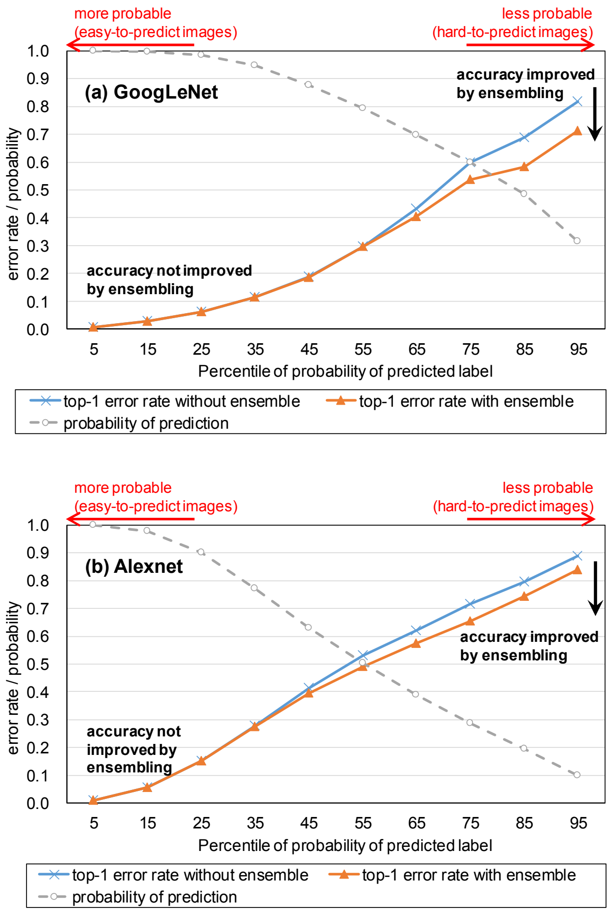

Figure 1. Improvements by ensemble and probabilities of predictions in ILSVRC 2012 validation set. X-axis shows percentile of probability of first local predictions from high (left) to low (right). Ensemble reduces error rates for samples with low probabilities but does not affect samples with high probabilities.

To determine whether or not this characteristic of ensembling is unique to GoogLeNet architecture, we conducted the same experiment using Alexnet as another network architecture and show the results in Figure 1(b). Although the prediction error rate is higher for Alexnet than for GoogLeNet, we observe similar characteristics of improvements by ensembling. Again, the improvements by ensembling are only observed in the right of the figure, i.e. for hard-to-predict images. These characteristics of the improvements by the ensemble are not unique to an ILSVRC dataset; we have observed similar trends in other datasets.

These results motivate us to make our adaptive ensemble prediction for reducing the additional cost of ensembling while keeping the benefit of improved accuracy. Once we obtain high enough prediction probability for an input image, doing further local predictions and ensembling will waste computation power without improving accuracy. The challenge is how to identify the condition in which to terminate ensembling. As described later, we identify the termination condition on the basis of the confidence level of the probability.

Also, we show the results when mixing GoogLeNet and Alexnet in the appendix.

3. Related Work

Various prediction methods that ensemble the outputs from many classifiers (e.g. neural networks) have been widely studied to achieve higher accuracy in machine learning tasks. Boosting (Freund and Schapire 1996) and Bagging (Breiman 1996) are famous examples of ensemble methods. Boosting and Bagging produce enough variances in classifiers included in an ensemble by changing the training set for each classifier. In recent studies on image classifications with deep neural networks, however, random numbers (e.g. for initialization or for ordering input images) used in the training phase can give sufficient variances in networks even using the same training set for all classifiers (networks). Hence, we use networks trained using the same training set and network architecture in this study.

The higher execution cost of the ensembling is a known problem, so we are not the first to attack it. For example, Hinton *et al.* (2015) also tackled the high execution cost of the ensembling. Unlike us, they trained a new smaller network by distilling the knowledge from an ensemble of networks by following Buciluă *et al.* (2006).

In our technique, we use the probability of the predictions to control the ensembling during the inference. Typically, the probability of the prediction generated by the softmax is used during the training of the network; the cross entropy of the probabilities is often used as the objective function of the optimization. However, using the probability for purposes other than the target of the optimization is not unique to us. For example, Hinton *et al.* (2015) used the probabilities from the softmax while distilling the knowledge from the ensemble. As far as we know, ours is the first study focusing on the relationship between the probability of the prediction and the effect of ensembling with current deep neural networks. Opitz and Maclin (1999) showed an important observation related to ours. They showed that the large part of the gain of ensembling came from the ensembling of the first few local predictions. Our observation discussed in the previous section enhances Opitz's observation from a different perspective: most gain of the ensembling comes from hard-to-predict samples.

4. Adaptive Ensemble Prediction

4.1 Basic Idea

This section details our proposed adaptive ensemble prediction method. As shown in Figure 1, the ensemble typically does not improve the accuracy of predictions if a local prediction is highly probable. Hence, we terminate ensembling without processing all N local predictions on the basis of the probabilities of the predictions. We execute the following steps:

1) start from $i = 1$
2) obtain i-th local prediction, i.e. the probability for each class label. We denote the probability for label L of i-th local prediction $p_{L,i}$
3) calculate the average probabilities for each class label
$$\langle p_L \rangle_i = \frac{\sum_{j=1}^{i} p_{L,j}}{i}$$
4) if $i < N$ and the termination condition is not satisfied, increment i and repeat from step 2
5) output the class label that has the highest average probability $\arg\max_L(\langle p_L \rangle_i)$ as the final prediction.

4.2 Termination Conditions

For the termination condition in Step 3, we test two conditions: one based on a simple static threshold, and the other on a confidence level.

4.2.1 Static-threshold-based termination condition

We can use a simple condition on the basis of a pre-determined threshold T to terminate the ensembling. In this condition, we just compare the highest average probability $\max_L(\langle p_L \rangle_i)$ against the threshold T. If the average probability exceeds the threshold, i.e. $\max_L(\langle p_L \rangle_i) > T$, we do not execute further local predictions for ensembling.

4.2.2 Confidence-Level-based termination condition

Instead of the pre-defined threshold, we can use the confidence intervals (CIs) as a termination condition. We first find the label that has the highest average probability (*predicted label*). Then, we calculate the CI of the probabilities using i local predictions. If the calculated CI of the predicted label does not overlap with the CIs for other labels, i.e. the predicated label is the best prediction with a certain confidence level, we terminate the ensembling and output the predicted label as the final prediction.

We calculate the confidence interval for the probability of label L using i local predictions by

$$\langle p_L \rangle_i \pm z \frac{1}{\sqrt{i}} \sqrt{\frac{\Sigma_{j=1}^{i} \left(p_{L,j} - \langle p_L \rangle_i\right)^2}{i-1}} \quad (1)$$

Here, z means the student-t distribution for the confidence level and the number of samples i.

Preferably, we want to do pair-wise comparisons between the predicted label and all other labels. However, computing CIs for all labels is costly, especially when there are many labels. To avoid excess costs of computing CIs, we compare the probability of the predicted label against the total of the probabilities of other labels. Since the total of the probabilities of all labels (including the predicted label) is 1.0 by definition, the total probabilities for the labels other than the predicted label are $1 - \langle p_L \rangle_i$ and the CI is the same size as that of the probability of the predicted label. Hence, our termination condition is

$$\langle p_L \rangle_i - (1 - \langle p_L \rangle_i) > 2z \frac{1}{\sqrt{i}} \sqrt{\frac{\Sigma_{j=1}^{i} \left(p_{L,j} - \langle p_L \rangle_i\right)^2}{i-1}} \quad (2)$$

We avoid computing CI if $\langle p_L \rangle_i < 0.5$ to avoid wasteful computation because the termination condition of equation 2 cannot be met in such cases.

Since the CI cannot be calculated with only one local prediction as is obvious from equation 1 to avoid zero divisions, we can use a hybrid of the two termination conditions. We use the static-threshold-based condition only for the first local prediction (i.e. $i = 1$) with a quite conservative threshold, and after the second local prediction is calculated, the confidence-level-based condition is used.

5. Experiments

5.1 Implementation

In this section, we investigate the effects of adaptive ensemble prediction on the prediction accuracy and the execution cost using various image classification tasks: ILSVRC 2012, Street View House Numbers (SVHN), CIFAR-10, and CIFAR-100 (with fine and course labels) datasets.

For the ILSVRC 2012 dataset, we use GoogLeNet as the network architecture and train the network using the stochastic gradient descent with momentum as the optimization method. For other datasets, we use a network that has six convolutional layers with batch normalization (Ioffe and Szegedy 2015) followed by two fully connected layers. We used the same network architecture except for the number of neurons in the output layer. We train the network using Adam (Kingma and Ba 2015) as the optimizer. For each task, we trained two networks independently. During the training, we used data augmentations by extracting a patch from a random position of the input image and using random horizontal flipping. Since adaptive ensemble prediction is an inference-time technique, it does not affect the network training. We executed the training and the inference on a Tesla K40 GPU for the ILSVRC 2012 dataset and a Tesla K20 GPU for other datasets.

We averaged up to 20 local predictions using ensembling. We created 10 patches from each input image by extracting from the center and the four corners with and without horizontal flipping by following Alexnet. For each patch, we made two local predictions using two networks. The patch size is 224×224 for the ILSVRC 2012 dataset and 28×28 for the other datasets. For adaptive ensemble prediction, we made local predictions in the following order: (center, no flip, network1), (center, no flip, network2), (center, flipped, network1), (center, flipped, network2), (top-left, no flip, network1), …, (bottom-right, flipped, network2). For the inference, we use a batch of 200 samples. As we repeated local predictions, the batch became smaller as computation for parts of samples terminated.

5.2 Results

Tables 1, 2, and 3 show how adaptive ensemble prediction affected the accuracy of predictions and the execution costs. Here, for our adaptive ensemble, we use the confidence-level-based termination condition with a 95% confidence level combined with the static threshold of 99.99% at the first local prediction.

We tested two different configurations: with one network (i.e. up to 10 local predictions) and two networks (up to 20 local predictions). In all datasets, the ensemble improved the accuracy in a tradeoff for the increased execution costs as expected. Using two networks doubled the number of local predictions on average (from 10 to 20) and increased both the benefit and drawback. If we use further local predictions (e.g. original GoogLeNet averaged up to 1,008 predictions), the benefit and the cost will become much more significant. Comparing our adaptive ensemble with the naive ensemble, our adaptive ensemble similarly improved accuracy (from 92% to 99% when we use two networks and from 83% to 99% when we use one network) while reducing the execution time by 2.1x to 2.8x and by 2.3x to 3.5x for the one-network and two-network configurations, respectively. These performance boosts came from the reduced number of local predictions used in the ensembles. The reductions were up to 6.9x and 12.7x for the one-network and two-network configurations. The reductions in the execution time over the naive ensemble were smaller than the reduction in the number of averaged predictions because of the additional overhead due to the confidence interval calculation, which was written in Python in the current implementation. Also, mini batches gradually became

Table 1. Prediction accuracy with and without adaptive ensemble

dataset / network		# class labels	**classification error rate (lower is better)** with one network			**classification error rate** with two networks	
			no ensemble	naive ensemble	our adaptive ensemble	naive ensemble	our adaptive ensemble
CIFAR-10		10	8.39%	6.97% (-1.41%)	7.00% (-1.39%)	6.23% (-2.16%)	6.34% (-2.04%)
SVHN		10	4.40%	3.44% (-0.96%)	3.50% (-0.90%)	3.19% (-1.21%)	3.29% (-1.11%)
CIFAR-100 (course label)		20	20.63%	17.84% (-2.79%)	18.04% (-2.59%)	16.56% (-4.07%)	16.78% (-4.06%)
CIFAR-100 (fine label)		100	30.28%	27.04% (-3.24%)	27.34% (-2.94%)	25.04% (-5.24%)	25.15% (-5.13%)
ILSVRC 2012	top-1 error	1000	32.36%	30.21% (-2.15%)	30.26% (-2.10%)	28.11% (-4.25%)	28.12% (-4.24%)
	top-5 error		12.67%	11.11% (-1.37%)	11.35% (-1.14%)	9.99% (-2.50%)	10.21% (-2.28%)

Ratios in parenthesis show improvements in error rate over baseline (no ensemble).

Table 2. Execution time with and without adaptive ensemble

dataset	**execution time per sample (lower is better)** with one network			**execution time per sample** with two networks	
	no ensemble	naive ensemble	our adaptive ensemble	naive ensemble	our adaptive ensemble
CIFAR-10	0.30 msec (1.0x)	2.55 msec (8.37x)	0.98 msec (3.20x)	4.98 msec (16.34x)	1.61 msec (5.28x)
SVHN	0.28 msec (1.0x)	2.52 msec (9.09x)	0.89 msec (3.20x)	4.96 msec (17.83x)	1.43 msec (5.13x)
CIFAR-100 (course label)	0.31 msec (1.0x)	2.55 msec (8.28x)	1.04 msec (3.58x)	4.99 msec (16.16x)	1.87 msec (6.01x)
CIFAR-100 (fine label)	0.31 msec (1.0x)	2.56 msec (8.36x)	1.25 msec (4.07x)	4.99 msec (16.28x)	2.15 msec (7.03x)
ILSVRC 2012	3.75 msec (1.0x)	35.84 msec (9.56x)	16.67 msec (4.45x)	70.74 msec (18.86x)	30.10 msec (8.03x)

Ratios in parenthesis show relative slowdown over baseline (no ensemble).

Table 3. Number of local predictions ensembled with and without adaptive ensemble

dataset	**# local predictions ensembled (lower is better)** with one network			**# local predictions ensembled** with two networks	
	no ensemble	naive ensemble	our adaptive ensemble	naive ensemble	our adaptive ensemble
CIFAR-10	1	10	1.66	20	1.92
SVHN			1.44		1.57
CIFAR-100 c			2.74		4.09
CIFAR-100 f			3.59		5.93
ILSVRC 2012			3.94		7.40

small as ensembling for parts of samples terminated. The smaller batch sizes reduced the efficiency of execution on current GPUs. Since the speedup by our adaptive technique over the naive ensemble became larger as the number of max predictions to ensemble increased, the benefit of our adaptive technique will become more impressive if we use larger ensemble configurations.

To study the differences due to the termination condition in our adaptive ensemble, we show the relationship between the prediction accuracy and the computation cost for ILSVRC 2012 and CIFAR-10 datasets in Figure 2. We used 2 networks in this experiment, i.e. up to 20 predictions were ensembled. In the figure, the x-axis is the number of ensembled predictions, so smaller means faster. The y-axis is the improvements in classification error rate over the baseline (no ensemble), so higher means better. We tested the static-threshold-based conditions by changing the threshold *T* and drew lines in

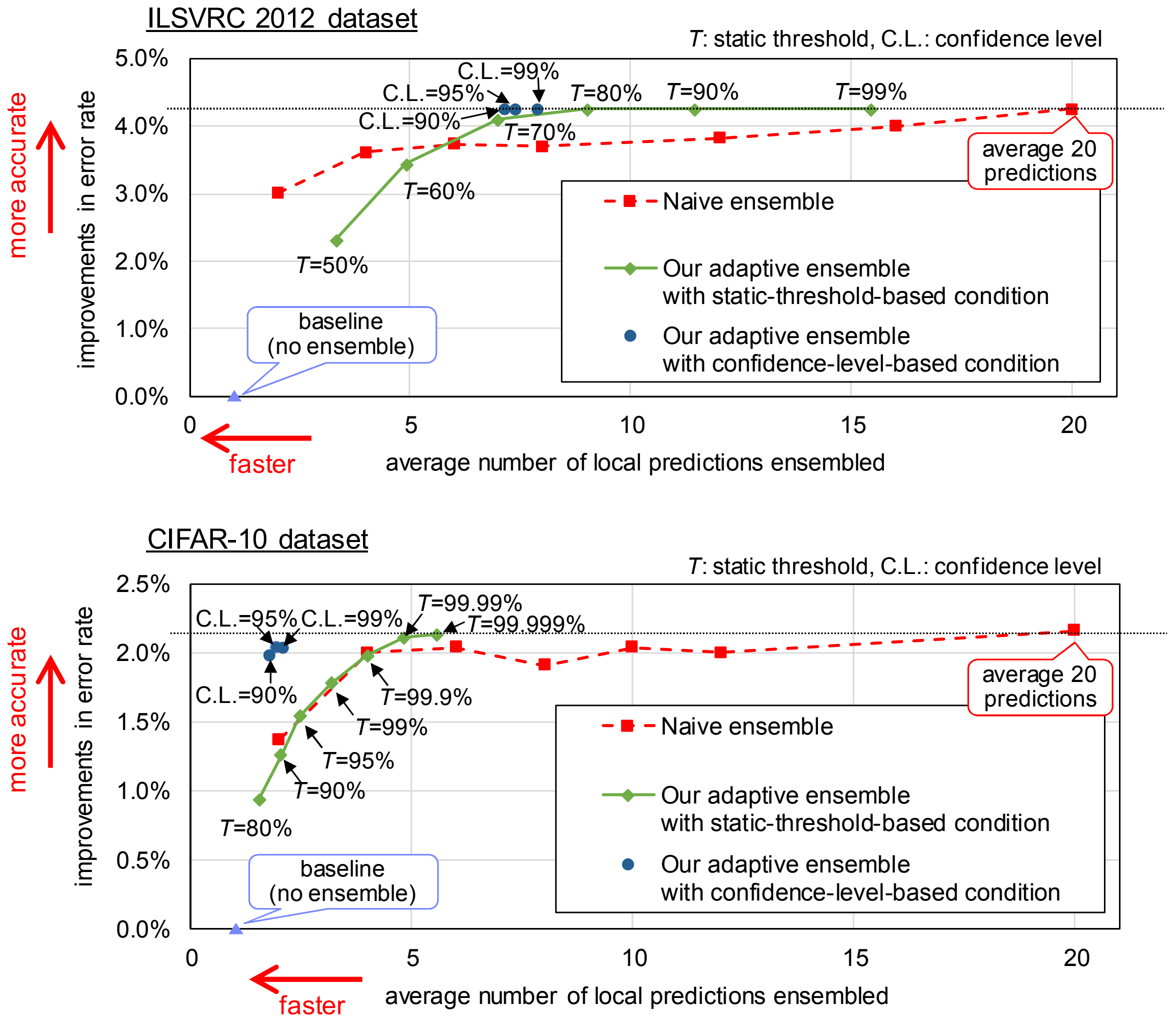

Figure 2. Prediction accuracy and computation cost with naive ensemble and our adaptive ensemble using different termination conditions. Static threshold *T* can be used to control tradeoff between accuracy and computation cost. Naive ensemble with all 20 predictions and no ensemble (0.0% in figure) are two tradeoff extremes. Confidence-level-based condition achieved better accuracy than static-threshold-based conditions with same computation cost especially for CIFAR-10. Tuning of confidence level (CL) is less sensitive than that of static threshold.

the figure. Similarly, we evaluated the confidence-level-based condition with three confidence levels with the static threshold of 99.99% only at the first local prediction. We also evaluated the naive ensemble (averaging the pre-defined number of predictions) by changing the number of predictions to average. From the figure, the threshold *T* can be used as a knob to control the tradeoff between the accuracy and the computation cost as well as the number of predictions on average in the naive ensemble. For both our adaptive ensemble with the static threshold and the naive ensemble, the naive ensemble with 20 predictions is at one end of the tradeoff because it corresponds with *T*=100%. The baseline, which does not execute ensemble, is at the other end, which always terminates at the first prediction regardless of the probability. By comparing the two lines, the adaptive ensemble with the static threshold achieved accuracy better than or comparable to the naive ensemble using the same number of predictions on average unless the threshold *T* was too small. This means that the probability of prediction is an effective criterion to control the number of predictions to ensemble. From Figure 1, it is reasonable that an excessively small threshold *T,* e.g. less than 80%*,* decreases the accuracy since it will significantly miss the opportunity that we can gain from ensembling.

Obviously, how to decide the best threshold *T* is the most important problem for the static threshold based condition. The confidence-level-based condition resolves this problem. The differences in the number of samples and the improvements in the error rate due to the choice of the confidence level were much less significant than the differences due to the static threshold in the static-threshold-based condition. Hence task-dependent fine tuning of the confidence level is not as important as the tuning of the static threshold. As is obvious from Figure 2, the static threshold tuning is highly dependent on the dataset and task. The easier tuning of the parameter is an important advantage of the confidence-level-based condition.

In addition to the benefit of the easier parameter tuning, the confidence-level-based condition further reduced the computation cost while maintaining the accuracy. The gain with the confidence-level-based condition over the static-threshold-based was significant especially for CIFAR-10 whereas it was marginal for ILSVRC 2012. These two datasets show the largest and smallest gain with the confidence-level-based condition over the static-threshold-based condition; other datasets showed improvements between those of the two datasets shown in Figure 2. By using the confidence-level-based condition, the adaptive ensemble largely outperformed the naïve ensemble for both data sets.

6. Conclusion

In this paper, we described our adaptive ensemble prediction to reduce the computation cost of ensembling many predictions. We were motivated to develop this technique by our observation that ensembling does not improve the prediction accuracy if the samples are easy to predict. Our experiments using various image classification tasks showed that our adaptive ensemble makes it possible to avoid wasting computing power without significantly sacrificing the prediction accuracy by terminating ensembles on the basis of the probabilities of the local predictions. The benefit of our technique will become larger if we use more predictions in an ensemble. Hence, we expect our technique to make the ensemble techniques more valuable for real-world systems by reducing the total computation power required while maintaining good accuracies and throughputs.

References

Leo Breiman, 1996, Bagging Predictors, *Machine Learning*, 24(2), pp 123-140.

Cristian Buciluă, Rich Caruana and Alexandru Niculescu-Mizil, 2006, Model compression, In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data <ining*, pp 535-541.

Yoav Freund and Robert E. Schapire, 1996, Experiments with a New Boosting Algorithm, In *Proceedings of International Conference on Machine Learning*, pp 148-156.

Song Han, Huizi Mao and William J. Dally, 2016, Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding, In *Proceedings of International Conference on Learning Representations*.

Lars Kai Hansen and Peter Salamon, 1990, Neural Network Ensembles, In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10), pp 993-1001.

Geoffrey Hinton, Oriol Vinyals and Jeff Dean, 2015, Distilling the Knowledge in a Neural Network, In arXiv:1503.02531 [stat.ML].

Sergey Ioffe and Christian Szegedy, 2015, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, In arXiv:1502.03167 [cs.LG].

Diederik Kingma and Jimmy Ba, Adam: A Method for Stochastic Optimization, In arXiv:1412.6980 [cs.LG].

Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton, 2012, ImageNet Classification with Deep Convolutional Neural Networks, In *Proceedings of Advances in Neural Information Processing Systems* 25, pp 1106-1114.

David Opitz and Richard Maclin, 1999, Popular ensemble methods: An empirical study, In *Journal of Artificial Intelligence Research*, 11, pp 169-198.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan and Vincent Vanhoucke and Andrew Rabinovich, 2015, Going Deeper with Convolutions, In *Proceedings of Computer Vision and Pattern Recognition*.

Cem Tekin, Jinsung Yoon and Mihaela van der Schaar, 2016, Adaptive Ensemble Learning with Confidence Bounds. In arXiv:1512.07446 [cs.LG].

Zhi-Hua Zhou, Jianxin Wu and Wei Tang, 2002, Ensembling neural networks: Many could be better than all, In *Artificial Intelligence*, 137(1-2), pp 239-263.

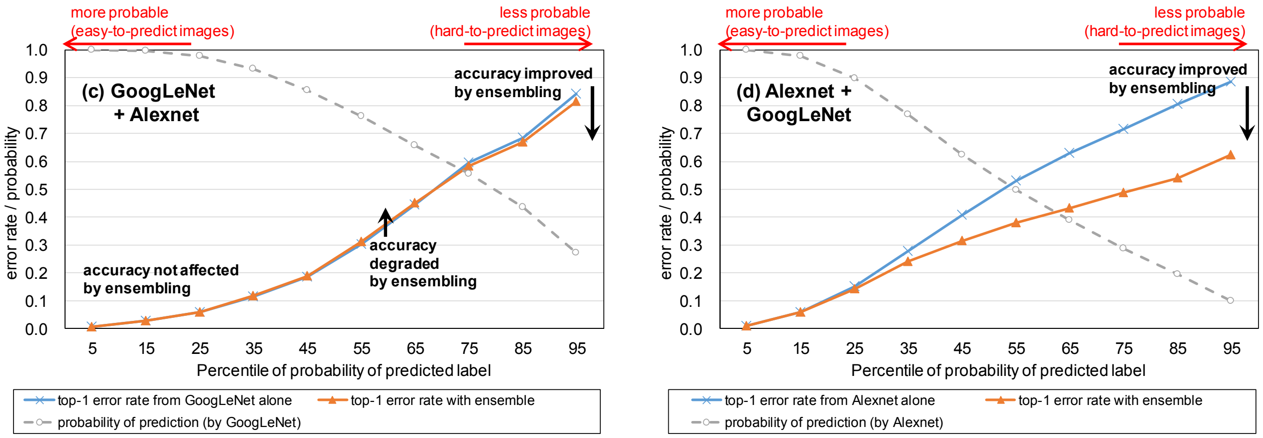

Figure 3. Improvements by ensemble and probabilities of predictions in ILSVRC 2012 validation set when we mix predictions from GoogLeNet and Alexnet. X-axis shows percentile of probability of first local predictions from high (left) to low (right).

Appendix

In Chapter 2, we showed that ensembling two predictions from two networks of the same network architecture can improve the prediction accuracy only for hard-to-predict samples, which have low probabilities of prediction. In this appendix, we show the results when we mix the predictions from GoogLeNet and Alexnet. Figure 3(c) shows the result when we use GoogLeNet as the first prediction and Alexnet as the second. As in Figure 1, the x-axis shows the percentile of the probability of the prediction by GoogLeNet from high to low, i.e. going left (right), input images become easier (harder) to predict.

For the leftmost region, i.e. 0- to 40-percentile samples, the ensemble from the two different networks predicts as accurately as GoogLeNet, which has higher prediction accuracy than Alexnet. For the rightmost region, i.e. 70- to 100-percentile samples, the ensemble improves the error rate over that of GoogLeNet alone. These characteristics are consistent with the cases using two identical networks shown in Figure 1. The interesting results are in the middle of these two regions, from 40- to 70-percentile samples. In this region, the ensemble slightly worsens the prediction accuracy below that from GoogLeNet alone. This is because GoogLeNet has better prediction performance than Alexnet, so the samples in this region are easy to predict for GoogLeNet but not for Alexnet. Although we study the ensemble of predictions from the same network in this paper, we need to take this behavior into account when extending this work to use predictions from different networks. When we use the predictions from Alexnet as the first prediction, the ensemble improves the accuracy for much wider regions as shown in Figure 3(d).