Adaptive Ensemble Prediction for Deep Neural Networks based on Confidence Level

Hiroshi Inoue
IBM Research - Tokyo

Abstract

Ensembling multiple predictions is a widely used technique for improving the accuracy of various machine learning tasks. One obvious drawback of ensembling is its higher execution cost during inference. In this paper, we first describe our insights on the relationship between the probability of prediction and the effect of ensembling with current deep neural networks; ensembling does not help mispredictions for inputs predicted with a high probability even when there is a non-negligible number of mispredicted inputs. This finding motivated us to develop a way to adaptively control the ensembling. If the prediction for an input reaches a high enough probability, i.e., the output from the softmax function, 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 cost significantly while achieving accuracy similar to that of static ensembling using a pre-defined number of local predictions. We also show that our statistically rigorous confidence-level-based early-exit condition reduces the burden of task-dependent threshold tuning better compared with naive early exit based on a pre-defined threshold in addition to yielding a better accuracy with the same cost.

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 by using tremendously large 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 devices or an embedded controller. Hence, various techniques (such as [Hinton et al., 2015] and [Han et al., 2016]) for achieving a high prediction accuracy without increasing computation time have been studied to enable more applications to be deployed in the real world.

Ensembling multiple predictions is a widely used technique for improving 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 image classification tasks, for example, accuracy is significantly improved by ensembling local predictions for multiple patches extracted from an input image to make a 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 of input images with and without horizontal flipping in their Alexnet paper. They also used up to seven networks and averaged the prediction to increase the accuracy. GoogLeNet by [Szegedy et al., 2015] averaged up to 1,008 local predictions by using 144 patches and 7 networks. In some ensemble methods, meta-learning during training to learn how to best mix multiple local predictions from the networks is used (e.g., [Tekin et al., 2015]). In Alexnet or GoogLeNet papers, however, significant improvements have been obtained by just averaging local predictions without 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 a 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,

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

especially on small devices, even though using it is almost the norm to win image classification competitions that emphasize prediction accuracy.

In this paper, we first describe our insights on the relationship between the probability of prediction and the effect of ensembling with current deep neural networks; ensembling does not help mispredictions for inputs predicted with a high probability, i.e., the output from the softmax, even when there is a non-negligible number of mispredicted inputs. To exploit this finding to speed up ensembling, we developed adaptive ensemble prediction that maintains the benefits of ensembling with much smaller additional costs. 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 our adaptive ensembling by using four image classification datasets: ILSVRC 2012, CIFAR-10, CIFAR-100, and SVHN. Our results showed that adaptive ensemble prediction reduces the computation cost significantly while achieving accuracy similar to that of static ensemble prediction with a fixed number of local predictions. We also showed that our statistically rigorous confidence-level-based early-exit condition yields a better accuracy with the same cost (or lower cost for the same accuracy) in addition to reducing the burden of task-dependent threshold tuning better compared with a naive early-exit condition based on a pre-defined threshold in the probability.

2 Ensembling and Probability of Prediction

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

2.1 Observations

To show the relationship between the probability of prediction and the effect of ensembling, we evaluated the prediction accuracy for the ILSVRC 2012 dataset with and without the 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 by using the first network without ensembling, 2) sorted the images in terms of the probability of prediction, and 3) evaluated the images with the second network and assessed the accuracy after ensembling two local predictions by 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), and as can be seen, the first local predictions became more (less) probable. The gray dashed line shows the average probability for each percentile class. Overall, ensembling improved accuracy well, although we only averaged two predictions. Interestingly, we can observe that the improvements only appear on the right side of the figure. There were almost no improvements made by ensembling two predictions on the left side, i.e., input images with highly probable local predictions, even when there was a non-negligible number of mispredicted inputs. For example, in the 50- to 60-percentile range with GoogLeNet, the top-1 error rate was 29.6% and was not improved by averaging two predictions from different networks.

For more insight into these characteristics, Figure 2(a) shows the breakdown of 5,000 samples in each 10-percentile range into 4 categories based on 1) whether the first prediction was correct or not and 2) whether the two networks made the same prediction or different predictions. When a prediction with a high probability was made first, we can observe that another local prediction tended to be the same regardless of its correctness. In the highest 10-percentile range, for instance, the two independently trained networks made the same misprediction for all 43 mispredicted samples. The two networks made different predictions only for 2 out of the 5,000 samples even when we included the correct predictions. In the 10- to 20-percentile range, the two networks generated different predictions only for 3 out of 139 mispredicted samples. Ensembling does not work well when local predictions tend to make the same mispredictions.

To determine whether or not this characteristic of ensembling is unique to the GoogLeNet architecture, we conducted the same experiment using Alexnet as another network architecture and show the results in Figure 1(b) and 2(b). Although the prediction error rate was higher for Alexnet than for GoogLeNet, we observed similar characteristics of improvements made by ensembling.

When we combined Alexnet (for the first prediction) and GoogLeNet (the second prediction), ensembling the local prediction from GoogLeNet, which yielded much higher accuracy than the first prediction by Alexnet, did not produce a significant gain in the 0- to 20-percentile range. We discuss this deeper in the supplementary material. Also, we show the results with ResNet50 [He et al., 2016] in the supplementary material. Even with ResNet, which has higher accuracy than GoogLeNet, the improvements made by ensem-

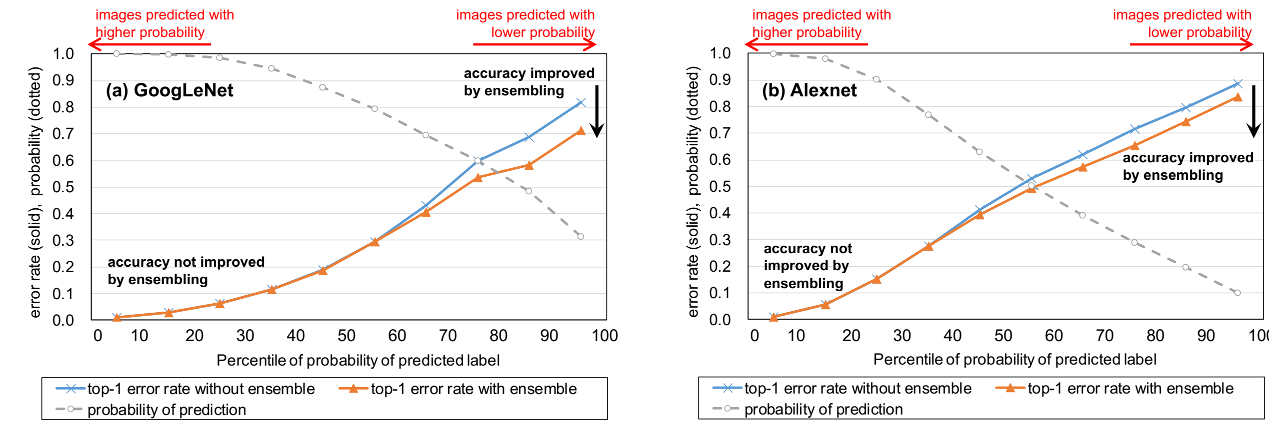

Figure 1: Improvements made by ensembling and probabilities of predictions for ILSVRC 2012 validation set. X-axis shows percentile of probability of first local predictions from high (left) to low (right). Ensembling reduces error rates for inputs with low probabilities but does not affect inputs with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

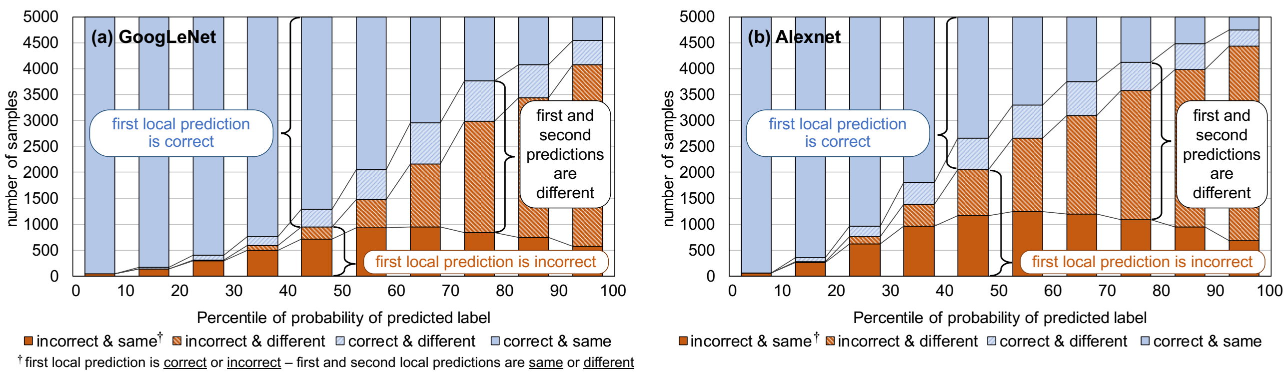

Figure 2: Breakdown of samples into four categories based on 1) whether first local prediction is correct or incorrect and 2) whether first and second predictions are same or different. X-axis shows percentile of probability of first local predictions from high (left) to low (right). Two networks tend to make same (mis)predictions for samples predicted with high probability (left), and, hence, ensembling does not work well for them.

bling were only observed on the right side of the figure, i.e., for images with low probabilities.

These characteristics of the improvements made by ensembling are not unique to an ILSVRC dataset; we have observed similar trends in other datasets.

2.2 Why This Happens

To understand why ensembling does not work for inputs predicted with high probabilities, we investigated a simplified classification task, which we detail in the supplementary material, and found a common type of misclassification that ensembling did not help.

Trained neural networks (or any classifiers in general) often make a prediction for a sample near the decision boundary of class A and class B with a low probability; the classifier assigns similar probabilities for both classes, and a class with a higher probability (but with a small margin) is selected as the result. For such samples, ensembling works efficiently by reducing the effects of random perturbations.

While ensembling works near a decision boundary that is properly learned by a classifier, some decision boundaries can be totally missed by a trained classifier due to insufficient expressiveness in the model that is used or a lack of appropriate training data. Figure 3 shows an example of true decision boundaries in 2-D feature space and classification results with a classifier that is not capable of capturing all of these boundaries. In this case, the small region of class A in the top-left was totally missed in the classification results obtained with a trained network, i.e., the samples in this region were mispredicted with high probabilities. Typically, such mispredictions cannot be avoided by ensembling predictions from another classifier trained with different random numbers since these mispredictions are caused by the poor expressiveness of the model rather than the perturbations that come from random numbers.

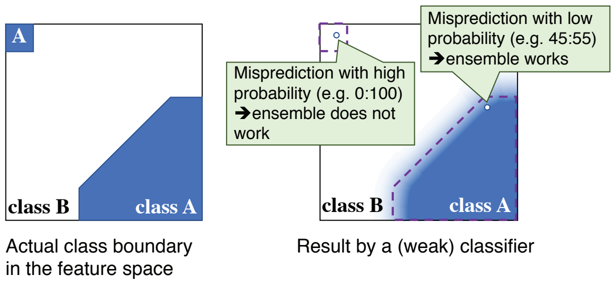

Figure 3: Schematic examples of mispredictions with high and low probabilities. Classifier may fail to learn some decision boundaries, and this leads to mispredictions with high probability; ensembling cannot help them.

These results motivated us to make our adaptive ensemble prediction for reducing the additional cost of ensembling while keeping the benefit of improved accuracy. Once we obtain a high enough prediction probability for an input image, further local prediction and ensembling will waste computation power without improving accuracy. The challenge is how to identify the condition under which ensembling is terminated early. As described later, we determine that an early-exit condition based on the confidence level of probability works well for all tested datasets.

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. Another technique for improving binary classification using multiple classifiers is soft-cascade (e.g., [Bourdev and Brandt, 2005], [Zhang and Viola, 2008]). With soft-cascade, multiple weak sub-classifiers are trained to reject a part of negative inputs. Hence, when these classifiers are combined to make one strong classifier, many easy-to-reject inputs are rejected in the early stages without consuming a huge amount of computation time. Compared with boosting, bagging, or soft-cascade, ours is an inference-time technique and does not affect the training phase.

In recent studies on image classification with deep neural networks, random numbers (e.g., for initialization or for ordering input images) used in the training phase can give sufficient variances in networks even with the same training set for all classifiers (networks). Hence, we use networks trained by using the same training set and network architecture in this study, and we assume that the capabilities of local classifiers are not that different. If a classifier is way too much weaker than the later classifiers as in soft-cascade, the ensembling goes differently compared with our observations as discussed in Section 2; mispredicted inputs in a weak classifier may be predicted correctly by later powerful classifiers even if they are predicted with high probabilities in the local prediction made by the weak classifier. For example, a later classifier that is capable of capturing the top-left region of class A in Figure 3 may predict the sample in the top-left correctly.

Another series of studies on accelerating classification tasks with two or few classes is based on the dynamic pruning of majority voting (e.g., [Hernández-Lobato et al., 2009], [Soto et al., 2016]). Like our technique, dynamic pruning uses a certain confidence level to prune ensembling with a sequential voting process to avoid wasting computation time. We show that the confidence-level-based approach is quite effective at accelerating ensembling by averaging local predictions in many-class classification tasks with deep neural networks when we use the output of the softmax as the probability of the local predictions. COMET by [Basilico et al., 2011] stops ensembling for random forest classifiers on the basis of the confidence interval. We also use the confidence interval but in a different way; COMET stops ensembling for binary classification tasks when an unobserved proportion of positive votes falls on the same side of 0.5 as the current observed mean with a certain confidence level. We use the confidence interval to confirm that a predicted label has a higher probability than other labels with confidence. We cannot naively follow GLEE since we do not target binary classification tasks. Also, COMET cannot take our approach because the random forest does not provide probability information for each local prediction unlike neural network classifiers. [Wang et al., 2018] decided the order of local predictions and also the thresholds for early stopping by solving a combinatorial optimization problem for binary classification tasks. In our study, we fixed the order of local predictions since our local predictions are based on the same network architecture and ordering is not that important. The basic idea of using the confidence level for the early-exit condition does not depend on this specific order of local predictions.

Some existing classifiers with a deep neural network (e.g., [Bolukbasi et al., 2017], [Teerapittayanon et al., 2017], [Huang et al., 2018]) take an early exit approach in ensembling similar to ours or take an early exit from one neural network.

In our study, we study how the early-exit condition affects the execution time and the accuracy in detail and show that our confidence-level-based condition works better than naive threshold-based conditions.

The higher execution cost of 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 ensembling. Unlike us, they trained a new smaller network by distilling the knowledge from an ensemble of networks by following [Buciluǎ et al., 2006].

To improve the performance of the inference of deep neural networks by making small and efficient executables suitable for handheld devices from trained networks, graph compilers and optimizers, such as NNVM/TVM ([Chen et al., 2018]) and Glow ([Rotem et al., 2018]), were recently developed.

In our technique, we use the probability of predictions to control ensembling during inference. Typically, the probability of predictions generated by the softmax is used during the training of a network; the cross entropy of the probabilities is often used as the objective function of optimization. However, using probability for purposes other than the target of optimization is not unique to us. For example, [Hinton et al., 2015] used probabilities from the softmax while distilling knowledge from an ensemble of multiple models to create a smaller network for deployment. As far as we know, ours is the first study focusing on the relationship between the probability of prediction and the effect of ensembling with deep neural networks.

[Opitz and Maclin, 1999] showed an important observation related to ours. They showed that a large part of the gain of ensembling came from the ensembling of the first few local predictions. Our observation discussed in the previous sections enhances Opitz's observation from a different perspective: most of the gain of ensembling comes from inputs with low probabilities in prediction.

4 Adaptive Ensemble Prediction

4.1 Basic Idea

This section details our proposed adaptive ensemble prediction method. As shown in Figure 1, ensembling 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 the i-th local prediction, i.e., the probability for each class label. We denote the probability for label L of the i-th local prediction $p_{L,i}$
3. calculate the average probabilities for each class label
$$\langle p_L \rangle_i = \frac{\sum_{j=1}^{i} p_{L,j}}{i} \tag{1}$$
4. if $i < N$, and the early-exit condition is not satisfied, increment i, and repeat from step 2
5. output the class label that has the highest average probability $\arg\max_{L} \left(\langle p_L \rangle_i \right)$ as the final prediction.

4.2 Confidence-level-based Early Exit

For the early-exit condition in step 4, we propose a condition based on confidence level.

We can use a naive condition on the basis of a predetermined static threshold T to terminate the ensembling, i.e., we just compare the highest average probability $\max_L \left(\langle p_L \rangle_i \right)$ against the threshold T. If the average probability exceeds the threshold, we do not execute further local predictions for ensembling. As we empirically show later, the best threshold T heavily depends on the task. To avoid this difficult tuning of the threshold T, we propose a dynamic and more statistically rigorous condition in this paper.

Instead of a pre-defined threshold, we can use confidence intervals (CIs) as an early-exit 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 predicted 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 as follows.

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

Here, z is defined such that a random variable Z that follows the Student's-t distribution with $i - 1$ degrees of freedom satisfies the condition $Pr\left[Z \leq z\right] = 1 - \alpha/2$. α is the significance level, and $(1 - \alpha)$ is the confidence level. We can read the value z from a precomputed table at runtime. To compute the confidence interval with a small number of samples (i.e., local predictions), it is known that the Student's-t distribution is more suitable than the normal distribution. When the number of local predictions increases, the Student's-t distribution approximates the normal distribution.

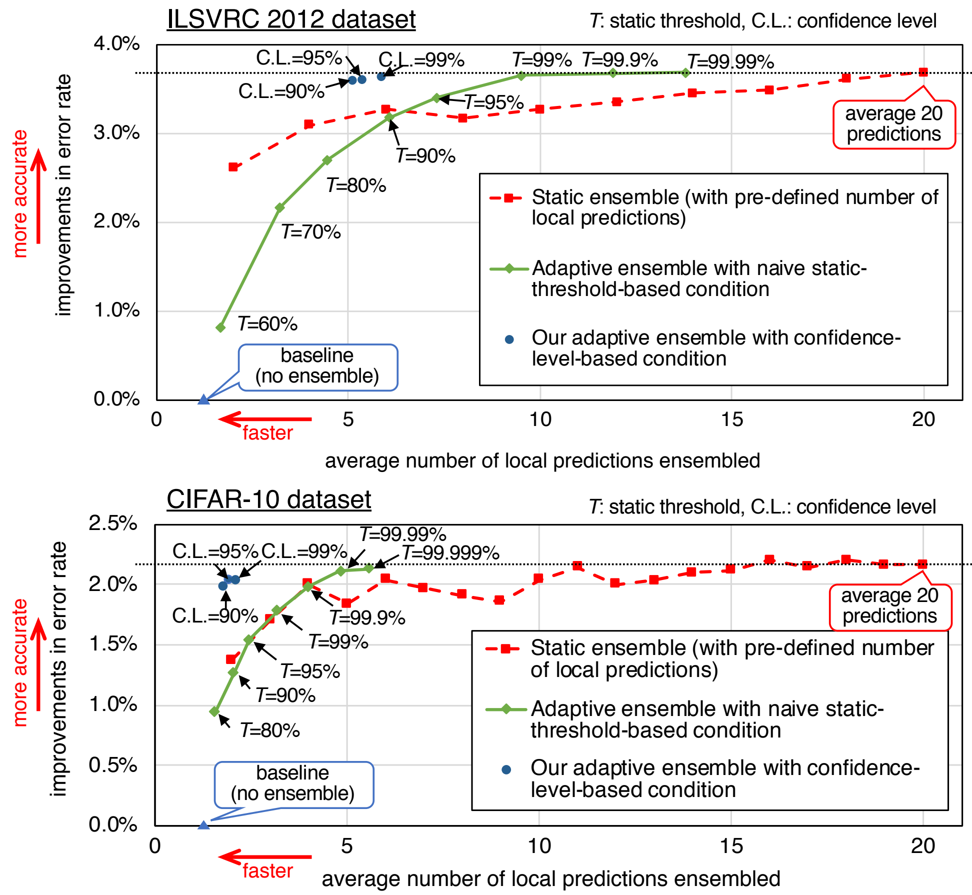

Figure 4: Prediction accuracy and computation cost with static ensemble and our adaptive ensemble using different early-exit conditions.Confidence-level-based condition achieved better accuracy than static-threshold-based conditions with same computation cost, especially for CIFAR-10. Tuning of confidence level (CL) is less sensitive than that of static threshold.

We can 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 the 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 of the probabilities for the labels other than the predicted label is $1 - \langle p_L \rangle_i$, and the CI is the same size as that of the probability of the predicted label. Hence, our early-exit condition is as follows.

$$\langle p_L \rangle_i - (1 - \langle p_L \rangle_i) > 2z \frac{1}{\sqrt{i}} \sqrt{\frac{\sum_{j=1}^{i} \left(p_{L,j} - \langle p_L \rangle_i\right)^2}{i-1}} \tag{3}$$

We avoid computing CI if $\langle p_L \rangle_i < 0.5$ to avoid wasteful computation because the early-exit 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 (3) to avoid zero divisions, we can use a hybrid of the two early-exit conditions. We use the static-threshold-based condition only for the first local prediction with a quite conservative threshold (99.99% in the current implementation) to terminate ensembling only for trivial inputs as early as possible, and after the second local prediction is calculated, the confidence-level-based condition of equation (3) is used. We also performed evaluation by doing pair-wise comparisons of CIs with all other labels or with the label having the second-highest probability. However, the differences in the results obtained by doing pair-wise comparisons were mostly negligible. There can be many other criteria for the early-exit conditions, but our approach with the confidence level is a reasonable choice for balancing multiple objectives, including accuracy, computation cost, and ease of tuning.

Table 1: Prediction accuracy with and without adaptive ensemble

dataset		# class labels	classification error rate (lower is better) with one network			classification error rate with two networks	
			no ensemble	static (10) ensemble	our adaptive ensemble	static (20) 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	31.72%	30.10% (-1.62%)	30.26% (-1.46%)	28.04% (-3.68%)	28.12% (-3.60%)
	top-5 error		11.55%	10.74% (-0.81%)	10.87% (-0.68%)	9.71% (-1.84%)	9.94% (-1.61%)

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

Table 2: 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	static ensemble	our adaptive ensemble	static ensemble	our adaptive ensemble
CIFAR-10			1.66		1.92
SVHN			1.44		1.57
CIFAR-100 c	1	10	2.74	20	4.09
CIFAR-100 f			3.59		5.93
ILSVRC 2012			2.99		5.36

5 Experiments

5.1 Implementation

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

For the ILSVRC 2012 dataset, we used GoogLeNet as the network architecture and trained the network by using stochastic gradient descent with momentum as the optimization method. For other datasets, we used a network that has six convolutional layers with batch normalization ([Ioffe and Szegedy, 2015]) followed by two fully connected layers with dropout. We used the same network architecture except for the number of neurons in the output layer. We trained the network by using Adam ([Kingma and Ba, 2014]) 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 an input image and using random horizontal flipping. Since adaptive ensemble prediction is an inference-time technique, network training is not affected.

We averaged up to 20 local predictions by using ensembling. We created 10 patches from each input image by extracting from the center and four corners of images with and without horizontal flipping by following Alexnet. For each patch, we made two local predictions using two networks. The patch size was 224 x 224 for the ILSVRC 2012 dataset and 28 x 28 for the other datasets. We made local predictions in the following order: (center, no flip, network 1), (center, no flip, network 2), (center, flipped, network 1), (center, flipped, network 2), (top-left, no flip, network 1), ..., (bottom-right, flipped, network 2). Since averaging local predictions from different networks typically yields better accuracy, we used this order for both our adaptive ensembling and fixed-number static ensembling. As far as we tested, the order of local predictions slightly affected the error rates, but it did not change the overall comparisons shown in the evaluations.

5.2 Results

To study the effects of our adaptive ensembling on the computation cost and accuracy, we show the relationship between them for the ILSVRC 2012 and CIFAR-10 datasets in Figure 4. We used two 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 evaluated the static ensembling (averaging the fixed number of predictions) by changing the number of predictions and our adaptive ensembling. For the adaptive ensembling, we also performed evaluation with two early-exit conditions: with naive static threshold and with confidence interval. We tested the static-threshold-based condition by changing the threshold T and drew lines in the figure. Similarly, we evaluated the confidence-level-based condition with three confidence levels frequently used in statistical testing: 90%, 95%, and 99%.

From the figure, there is an obvious trade-off between the accuracy and the computation cost. The static ensemble with 20 predictions was at one end of the trade-off because it never exited early. The baseline, at which ensembling was not executed, was at the other end, and it always terminated at the first prediction regardless of the probability. Our adaptive ensembling with confidence-level-based condition achieved better accuracy with the same computation cost (or smaller cost for the same accuracy) compared with the static or naive adaptive ensembling with a static threshold. The gain with the confidence-level-based condition over the static-threshold-based was significant for CIFAR-10, whereas it was marginal for ILSVRC 2012. These two datasets showed the largest and smallest gain with the confidence-level-based condition over the static-threshold-based condition; the other datasets showed improvements between those of the two datasets shown in Figure 4.

When comparing the two early-exit conditions in adaptive ensembling, the confidence-level-based condition eliminated the burden of parameter tuning better compared with the naive threshold-based condition in addition to the benefit of the reduced computation cost. Obviously, how to decide the best threshold T is the most important problem for the static-threshold-based condition. The threshold T can be used as a knob to control the trade-off between accuracy and computation cost, but the static threshold tuning is highly dependent on the dataset and task. From Figure 4, for example, $T = 90\%$ or $T = 95\%$ seem to have been a reasonable choice for ILSVRC 2012, but it was a problematic choice for CIFAR-10. For the confidence-level-based condition, the confidence level also controlled the trade-off. However, the differences in the computation cost and the improvements in accuracy due to the choice of the confidence level were much less significant and less sensitive to the current task than the differences due to the static threshold. Hence, task-dependent fine tuning of the confidence level is not as important as the tuning of the static threshold. The easier (or no) tuning of the parameter is an important advantage of the confidence-level-based condition.

Tables 1 and 2 show how adaptive ensemble prediction affected the accuracy of predictions and the execution costs in more detail for five datasets. Here, for our adaptive ensembling, we used the confidence-level-based early-exit condition with a 95% confidence level for all datasets on the basis of the results of Figure 4. We tested two different configurations: with one network (i.e., up to 10 local predictions) and with two networks (up to 20 local predictions). In all datasets, the ensembling improved the accuracy in the trade-off 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 were to use further local predictions (e.g., original GoogLeNet averaged up to 1,008 predictions), the benefit and cost would become much more significant. Comparing our adaptive ensembling with static ensembling, our adaptive ensembling similarly improved accuracy while reducing the number of local predictions used in the ensembles; the reductions were up to 6.9 and 12.7 times for the one-network and two-network configurations. Since the speed-up with our adaptive technique over static ensembling becomes larger as the number of max predictions to ensemble increases, the benefit of our adaptive technique will become more impressive if we use larger ensemble configurations.

6 Conclusion

In this paper, we described our adaptive ensemble prediction with statistically rigorous confidence-level-based early-exit condition 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 predictions are highly probable. Our experiments using various image classification tasks showed that our adaptive ensembling makes it possible to avoid wasting computing power without significantly sacrificing 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 ensemble techniques more valuable for real-world systems by reducing the total computation power required while maintaining good accuracy and throughput.

References

[Basilico et al., 2011] Basilico, J. D., Munson, M. A., Kolda, T. G., Dixon, K. R., and Kegelmeyer, W. P. (2011). Comet: A recipe for learning and using large ensembles on massive data. In *International Conference on Data Mining (ICDM)*, pages 41–50.

[Bolukbasi et al., 2017] Bolukbasi, T., Wang, J., Dekel, O., and Saligrama, V. (2017). Adaptive neural networks for efficient inference. *arXiv:1702.07811*.

[Bourdev and Brandt, 2005] Bourdev, L. and Brandt, J. (2005). Robust object detection via soft cascade. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 236–243.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. In *Machine Learning*, pages 123–140.

[Buciluǎ et al., 2006] Buciluǎ, C., Caruana, R., and Niculescu-Mizil, A. (2006). Model compression. In *KDD*, pages 535–541.

[Chen et al., 2018] Chen, T., Moreau, T., Jiang, Z., Shen, H., Yan, E., Wang, L., Hu, Y., Ceze, L., Guestrin, C., and Krishnamurthy, A. (2018). Tvm: End-to-end optimization stack for deep learning. *arXiv:1802.04799*.

[Han et al., 2016] Han, S., Mao, H., and Dally, W. J. (2016). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *International Conference on Learning Representation (ICLR)*.

[Hansen and Salamon, 1990] Hansen, L. K. and Salamon, P. (1990). Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001.

[He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

[Hernández-Lobato et al., 2009] Hernández-Lobato, D., Martínez-Muñoz, G., and Suárez, A. (2009). Statistical instance-based pruning in ensembles of independent classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):364–369.

[Hinton et al., 2015] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv:1503.02531*.

[Huang et al., 2018] Huang, G., Chen, D., Li, T., Wu, F., van der Maaten, L., and Weinberger, K. Q. (2018). Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations (ICLR)*.

[Ioffe and Szegedy, 2015] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv:1412.6980*.

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 1106–1114.

[Opitz and Maclin, 1999] Opitz, D. and Maclin, R. (1999). Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198.

[Rotem et al., 2018] Rotem, N., Fix, J., Abdulrasool, Saleem amd Deng, S., Dzhabarov, R., Hegeman, J., Levenstein, R., Maher, B., Nadathur, S., Olesen, J., Park, J., Rakhov, A., and Smelyanskiy, M. (2018). Glow: Graph lowering compiler techniques for neural networks. *arXiv:1805.00907*.

[Soto et al., 2016] Soto, V., Suárez, A., and Martínez-Muñoz, G. (2016). An urn model for majority voting in classification ensembles. In *Annual Conference on Neural Information Processing Systems (NIPS)*, pages 4430–4438.

[Szegedy et al., 2015] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[Teerapittayanon et al., 2017] Teerapittayanon, S., McDanel, B., and Kung, H. (2017). Branchynet: Fast inference via early exiting from deep neural networks. *arXiv:1709.01686*.

[Tekin et al., 2015] Tekin, C., Yoon, J., and van der Schaar, M. (2015). Adaptive ensemble learning with confidence bounds. *arXiv:1512.07446*.

[Wang et al., 2018] Wang, S., Gupta, M., and You, S. (2018). Quit when you can: Efficient evaluation of ensembles with ordering optimization. *arXiv:1806.11202*.

[Zhang and Viola, 2008] Zhang, C. and Viola, P. (2008). Multiple-instance pruning for learning efficient cascade detectors. In *Annual Conference* *on Neural Information Processing Systems (NIPS)*, pages 1681–1688.

[Zhou et al., 2002] Zhou, Z.-H., Wu, J., and Tang, W. (2002). Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1-2):239–263.