ArchEnsemble: Architectural Approach for More Efficient Ensemble

STAT433 Final Project

Hyogon Ryu
Department of Cyber Defense, Korea University
ryuhyogon@korea.ac.kr

Abstract

The ensemble technique is used to achieve state-of-the-art in various tasks. In particular, in order to implement a service, it is recommended to use a model with the best possible performance, so many company and task challengers use the various models and ensemble it. However, ensemble has a disadvantage in that it requires a lot of computational cost to train several models. If you need to train deep learning models, much more computational cost is involved. This is one of the main concern of service provider. In this paper, we propose ArchEnsemble, an efficient ensemble technique that can produce the effect of ensemble while minimizing the increase in computational cost. In my experiments, I measure the accuracy on CIFAR-10 and CIFAR-100 dataset. Significant performance improvement was confirmed in SimpleCNN and ResNet20 model, and slight performance improvement was also confirmed in ResNet32 model.

1 Introduction

Model ensemble is used to maximize the performance of the model.[1] To apply a model to a service, it is necessary to make the performance the best, and for this, various ensemble methods are used. In particular, these ensemble methods are used more in services and challenges than in the research area.[2, 3, 4] Using ensembles can benefit from accuracy and robustness. However, there are also problems that arise from using the ensemble. There are two major problems when ensemble models. The first is the train cost and the second is the inference cost.

To use multiple models, you need to train different models. A lot of computation cost is used in this process. When you don't use the ensemble, only one model could be used, but since several models need to be trained, the train cost is doubled. As the number of ensemble models increases, the train cost increases. In particular, train cost is an important issue in deep learning. Train cost is an important issue as many studies to lighten the deep learning model are in progress.[5, 6] From this point of view, Model Ensemble can be effective in terms of performance, but has the disadvantage of increasing computation cost.

In addition, when Model Ensemble is performed, the cost of inference time increases. Inference cost also increases because of multiple models inference. This is fatal in that it continues to require a lot of computation power even after the train is completed. Compared to train model, inference requires less computational power at a time, but inference continues while providing service, so this is also an important problem in the existing ensemble method.

In this paper, we propose a new efficient ensemble method to solve the above problem of Model Ensemble. Unlike the existing method of training multiple models, we propose ArchEnsemble, a method to obtain the effect of Ensemble by tiny modifying of the Model Architecture. ArchEnsemble solves

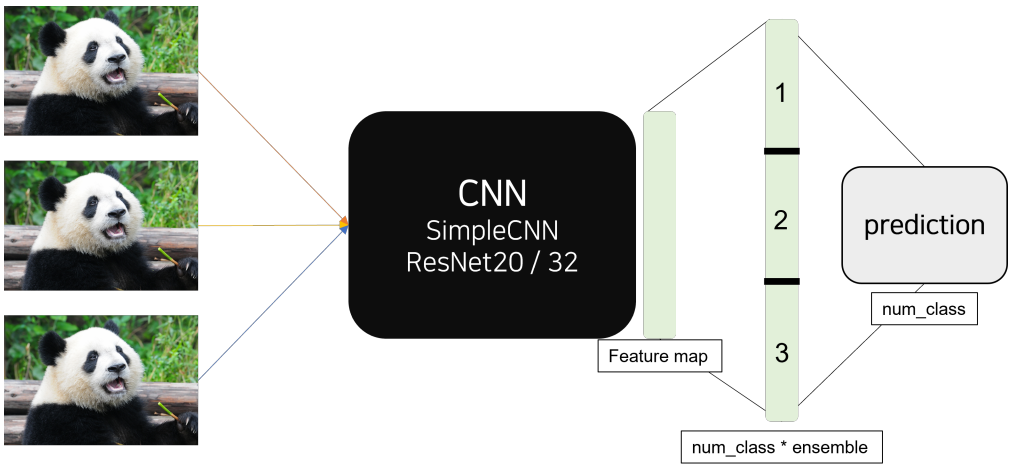

Figure 1: overall architecture of ArchEnsemble

both the computation cost problem of the existing Model Ensemble method that occurs in train time and inference time. When ArchEnsemble was applied to resnet, increasing accuracy while increasing only 0.3% of parameters was confirmed. I conducted an experiment on the CIFAR10/CIFAR100 dataset, and it was confirmed that the performance improved compared to when no ensemble was performed.

My main contributions can be summarized as follows:

- I propose the ArchEnsemble, which is the new approach of the ensemble method. The method is very simple and need tiny modifying of architecture.
- The method doesn't have to train the other model. And also doesn't have to inference multiple times. It solves the main cost problem of the model ensemble method and is greatly benefited in computation cost.
- The method can be applied to all kind of CNNs. In this paper, I confirmed that it is work on simplecnn, resnet20, resnet32. In addition, model accuracy was significantly increased on CIFAR10/CIFAR100 dataset by applying the method.

2 Approach

This section describes the details of the ArchEnsemble Method. Section 2.1. describes the main idea of ArchEnsemble, Section 2.2 describes the Architecture, and Section 2.3 describes the train and inference method.

2.1 Main Idea

Since I had to solve the computation cost problem of Model Ensemble, I had to obtain the effect of ensemble with one train and one inference. At the same time, it had to be applicable to existing models. This is one of the most important conditions because it is difficult to call ensemble if the method cannot be applied to the existing model. So, like the existing models that output multiple outputs(e.g. google inception[7]) or models that receive multiple inputs[8, 9] were referred to. In addition, methods such as dropout[10] and BatchEnsemble[11] to obtain the effect of ensemble while minimizing changes in architecture were also referred to. Through this, I devised a model that receives multiple input images and generates multiple predictions.

2.2 Architecture

Figure 1 shows the overall architecture of ArchEnsemble. First, process multiple images (not an image) as input. In this case, the number of images is the same as the number of ensembles. And it goes through the process of extracting features through Backbone. A CNN model is used as the backbone. In this paper, SimpleCNN, ResNet20, and ResNet32 were used as backbones, but any

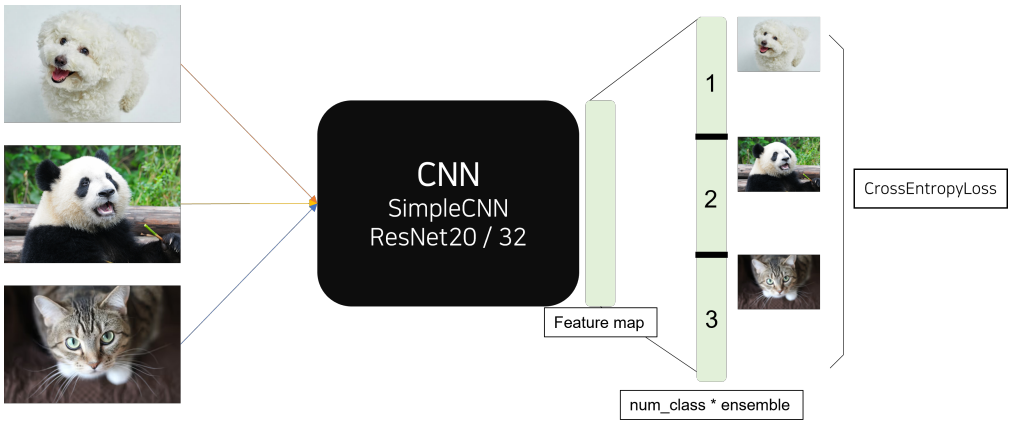

Figure 2: Train Method of ArchEnsemble

CNN model capable of feature extraction can be used. At this time, the existing CNN models output a tensor which have the same dimension as the number of class labels through the fc-layer at the end. However, ArchEnsemble sets the output dimension of this fc-layer as (# of classes in label) $\times$ (# of ensembles). After that, when making the final prediction, the previous output vector is divided into vectors which have a dimension of (# of classes in label), and then averaged.

2.3 Train and Inference Method

Figure 2 shows ArchEnsemble's Train method. During training, different images are used as input images. Prepare different images as many as the number of ensembles, and use them as input to the model. After that, the output vector is divided by the number of classes, and CrossEntorpyLoss is calculated for each divided part as the label of the input image. After that, the sum of these becomes the total loss. Training is performed on the train dataset through the below loss.

$$Loss_{ArchEnsemble} = CE_{img1} + CE_{img2} + CE_{img3} \tag{1}$$

Inference time, unlike train, all input images are used as the same image. This is the same as you can see in the input part of Figure 1.

3 Experiments

Section 3.1 describes the experiment details, and Section 3.2 describes the experimental results of the method for two image classification tasks. Section 3.3 describes the change of number of model parameters.

3.1 Experiments Detail

The experimental datasets were CIFAR-10 and CIFAR-100. It was used because it is the most frequently used image classification dataset. The CNN models that used as backbone were the SimpleCNN(2 conv-layer) model[1] and the ResNet20 and ResNet32 models[2]. The number of ensembles of ArchEnsemble was compared by conducting an experiment for 1, 2, and 4 cases. As the optimizer, Adam was used. The learning rate schedular, StepLR, was used, and the initial learning rate started with 1.0 and gamma was used as 0.7. batch_size is set to 10. The epoch was set differently for each experiment because it was set at a value that seems to have sufficiently converges when looking at the model's accuracy log. Detailed setting for epoch is described in section 3.2. All experiments were conducted on Google Colaboratory.

[1]SimpleCNN Model is from `https://github.com/pytorch/examples/blob/main/mnist/main.py`
[2]ResNet Model is from `https://github.com/akamaster/pytorch_resnet_cifar10`

Figure 3: Test Accuracy log models. **Left side** are accuracy log of CIFAR-10 and **right side** are accuracy log of CIFAR-100. **a)** The first row is log of SimpleCNN, **b)** second is ResNet20, **c)** third is ResNet32.

Since this is not the application project, I didn't work hard to optimize the hyperparameter. Therefore, if the hyperparameter is optimized, better performance can be confirmed than the experimental results reported in section 3.2.

3.2 CIFAR-10/100 Classification

Figure 3 shows the overall experiment result. At this time, the horizontal axis is the epoch and the vertical axis is the test accuracy(%). First, Looking at the experimental results of a)SimpleCNN, in CIFAR-10, the Cyan color log is not ensembled. Comparing with the other two lines, it can be seen that the other two lines have higher accuracy in most epochs. If you check the result of CIFAR-100 in the graph on the right, you can see it better. It can be seen that the green line is the log without ensemble, and the brown line continues to achieve higher accuracy. This tendency can also be confirmed in the experimental result of b)ResNet20 in the second row. In the CIFAR-10 graph of ResNet20 (left), the orange line shows a value that is not ensembled, and in the CIFAR-100 graph (right), the peach line shows a value that is not ensembled. In all, it can be seen that the ensembled lines tend to have higher accuracy than the non-ensembled lines.

On the other hand, if you check the experimental result of c)ResNet32 corresponding to the 3rd row, you can see a slightly different result. In the CIFAR-10 graph of ResNet32, the orange line is a non-ensemble line, but there are many epochs showing better performance than the ensemble line. This can be confirmed more clearly by looking at the CIFAR-100 graph. In CIFAR-100, the purple

	# of ensemble	CIFAR-10	CIFAR-100
SimpleCNN	1	68.63	35.13
	2	74.35	**40.63**
	4	**76.15**	
ResNet20	1	80.92	52.03
	2	84.88	**56.46**
	4	**85.97**	
ResNet32	1	82.69	56.53
	2	83.07	**59.68**
	4	**87.02**	

Table 1: top-1 Accuracy of each model(in percent). Best results are bolded

	# of ensemble	of parameter	increase(%)
SimpleCNN	1	1,626,442	0.000
	2	1,627,732	0.079
	4	1,630,312	0.238
ResNet20	1	278,682	0.000
	2	279,972	0.463
	4	282,552	1.389
ResNet32	1	473,114	0.000
	2	474,404	0.273
	4	476,984	0.818

Table 2: number of parameter and parameter increase

line is a non-ensemble line, and it is difficult to say which one is better than the other lines. This part has room for analysis through additional ablation studies.

Table 1 shows the top-1 accuracy of each model. In the table, it can be seen that when all models were ensembled, there was a significant performance improvement compared to that which did not ensemble.

3.3 Model Parameters

As ArchEnsemble progresses, the model parameters will increase. Since the number of model parameters directly affects the train cost, we checked how many parameters were increased during ensemble. Table 2 shows the number of parameters of each model and how many parameters have increased compared to when ensemble is not performed. When looking at the parameter increase, all are around 1%. This is an extremely small number compared to the fact that it takes twice the train cost for training when two models are ensembled by standard ensemble method.

4 limitation

Several experiments and analyzes have been conducted on ArchEnsemble, but stil the following limitations exist.

- It is a pity that we were not able to conduct experiments on imagenet dataset due to insufficient computing cost.
- I have conducted various experiments on model Architecture and suggested a structure like ArchEnsemble, but since all my experiments have been conducted only on simple settings, there may be a better structure than the one suggested.

- An ablation study on model complexity is required. Comparing the structure of SimpleCNN and ResNet, ResNet is structurally more complicated, but SimpleCNN has more model parameters. Therefore, additional ablation studies on the influence of architecture and the influence of model parameters are needed.
- It has been confirmed that the accuracy increases with very little cost, but comparison with the general ensemble method is insufficient.
- Since the optimization of hyperparameters has hardly been carried out, there is room for further performance improvement by optimizing it.

5 Conclusion

In this paper, I propose ArchEnsemble, the new novel approach for efficient ensemble, to solve the computation cost problem of the existing model ensemble method. This is a method to obtain the effect of ensemble by giving a slight change to the model architecture, and solves both the computation cost problem of the existing model ensemble method that occurs in train time and inference time. As a result of conducting the experiment on CIFAR-10, it was possible to obtain an accuracy improvement of 5% or more while increasing the number of model parameters within 1%. The strength of this method is that it is easy to apply to the existing methods as well as those mentioned above. This method is very simple and easy to implement. In addition, it can be applied to all types of CNNs where feature extraction is possible.

Through this study, I proposed an architectural approach for an efficient ensemble. Based on this, it is hoped that the architectural approach will be one way to solve the computation cost problem of model ensemble through more studies in the future.

References

[1] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[2] Claudia Tebaldi and Reto Knutti. The use of the multi-model ensemble in probabilistic climate projections. *Philosophical transactions of the royal society A: mathematical, physical and engineering sciences*, 365(1857):2053–2075, 2007.

[3] Yawen Xiao, Jun Wu, Zongli Lin, and Xiaodong Zhao. A deep learning-based multi-model ensemble method for cancer prediction. *Computer methods and programs in biomedicine*, 153:1–9, 2018.

[4] Christopher Clark, Mark Yatskar, and Luke Zettlemoyer. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. *arXiv preprint arXiv:1909.03683*, 2019.

[5] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[6] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

[7] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[8] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew M Dai, and Dustin Tran. Training independent subnetworks for robust prediction. *arXiv preprint arXiv:2010.06610*, 2020.

[9] Tim Whitaker and Darrell Whitley. Prune and tune ensembles: Low-cost ensemble learning with sparse independent subnetworks. *arXiv preprint arXiv:2202.11782*, 2022.

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[11] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*, 2020.