ACTIVE COVARIATE SHIFT: A NEW WAY FOR DEEP ENSEMBLE

HAOXIANG YOU [YOUHAOX@SEAS.UPENN.EDU], HIEN BUI [XUANHIEN@SEAS.UPENN.EDU],

ABSTRACT. Deep neural ensemble is a technique to improve the test performance and robustness of neural networks. Particularly, a set of distinct models are trained separately and their outputs are subsequently combined using a decision fusion method. In this project, we proposed a new method to train a set of different models by putting a fixed transformation layer. In addition, we introduce a decision fusion technique inspired by Kalman filter. We conduct experiments on CIFAR10 dataset and show the effectiveness of our method to improve generalization. Furthermore, the result demonstrates that our proposed method is more robust to adversarial attacks compared to a single model.

1. INTRODUCTION

Generalization, a major challenge in deep learning, refers to the ability of a model to perform well on new data. This is an important property of a good machine-learning model, as in the real world, the model will encounter data that are not seen during training.

A common technique to improve the generalization is to increase the amount and diversity of the datasets. This can be done either by conducting real experiments to gather additional data or by augmenting existing datasets. However, collecting real-world data can be expensive, and data augmentation is often not enough to obtain a good model. An alternative way to improve the generalization is to combine several individual models, a method called ensemble learning. Dropout[1]is an example of ensemble learning, where it randomly turns off neurons during the training process and weights all neurons by a factor during testing. Yet, a common practice in model ensemble training is to assign a specific portion of the training datasets or set of hyper-parameters (e.g different hidden layer sizes, epochs) to each model. However, that trick might not be sufficient to produce a variety of models, hence restricting the performance.

In this project, we propose a new way to train a set of different deep networks by active shift the input covariate. More specifically, we place a distinct transformation layer before each network. These transformation layers can be sampled from typical image augmentation policy[2]. Our objective is to force different networks to use different features to make predictions. For example, if one model relies on color information to make a prediction(e.g. classify an image as an apple if it is red), we can train an distinct model by putting a gray transformation, which removes the color information.

We conduct experiments on the CIFAR10 dataset and show the ensemble deep networks have better generalization performance than a single model. We also find our ensemble networks are more robust to adversarial attacks.

2. BACKGROUND

2.1. **Covariate shift.**

Covariate shift is a phenomenon to describe the inputs to the neural network are sampled from different distributions. Here, we actively seek covariate shifts among models to obtain distinct models. These covariate shifts are achieved by distinct transformation layers (e.g. rotating the image by 10 degrees). It is important to note that the covariate shift only happened across different models. The input distributions remain the same for each model during the training and testing time since the transformation layers are fixed after initialization.

2.2. **Uncertainty in neural works.**

There are two major types of uncertainty: aleatoric uncertainty and epistemic uncertainty [3]. Aleatoric uncertainty indicates the inherent randomness of data (e.g. stochasticity of nature), while epistemic uncertainty accounts for the deficiency of the model (e.g. lack of data). Typical methods for estimating uncertainty in neural models[4] include Monte Carlo dropout, Bayesian network, and test-time data augmentation.

2.3. **Deep ensemble.**

Ensemble learning is the process of combining multi-models to achieve better performance. Deep ensemble learning combines the advantage of both deep learning models and ensemble learning, in which the final result have better generalization performance and is more robust to adversarial attacks [5]

3. Approach

3.1. **Model architecture and training.** The model architecture is shown in figure 1. The overall model consists of three parts: **N** transformation layers for shifting images, **N** deep neural networks for making predictions, and an ensemble layer for fusing the results.

The transformation layers are sampled from typical data augmentation policy. After initialization, they are fixed and remain the same at both training and testing times.

The **N** deep networks are the only learnable layers in these architectures. They can be any standard image classifier architecture such as ResNet.[6]

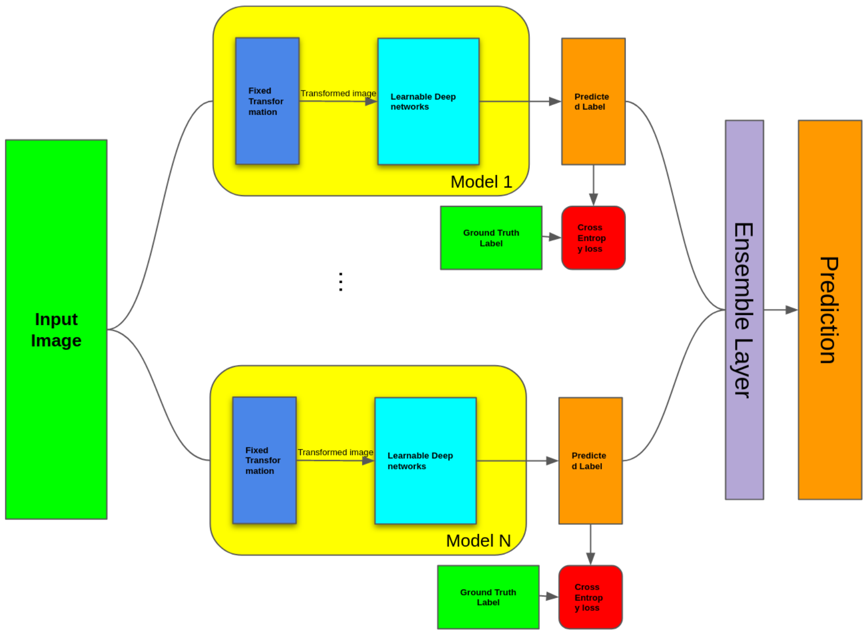

FIGURE 1. Model architecture

3.2. **Deep ensemble.**

3.2.1. *Uncertainty estimation.* In order to ensemble models optimally, we need to estimate the uncertainty of each model's predictions. Specifically, we use Monte Carlo dropout [7] as a measurement of the model uncertainties. For each model, we set the mode as "train" to enable the dropout layer randomly deactivate nodes and run that model **M** times to obtain different predictions. We would then use these **M** results to calculate the mean and covariances/variances. Thus, we obtained **N** different predictions $o_1, \dots o_n$ and $\Sigma_1, \dots, \Sigma_n$

3.2.2. *Decision fusing.* Kalman filter is a common technique to fuse different observations in robotics. Given a previous estimation of hidden state $\hat{s}_{t-1}$, a new observation about hidden state o_t and their covariance $\Sigma_{\hat{s}_{t-1}}$ and Σ_{o_t}, Kalman filter calculate the best estimation of hidden state and the uncertainty by following:

$$K = \Sigma_o(\Sigma_o + \Sigma_1)^{-1} \tag{1a}$$

$$\hat{s}_t = K\hat{s}_{t-1} + (1-K)o_t \tag{1b}$$

$$\Sigma_{\hat{s}_t} = K\Sigma_{\hat{s}_{t-1}}K^T + (1-K)\Sigma_{o_t}(1-K)^T \tag{1c}$$

We apply the idea of the Kalman filter to fuse **N** predictions. We first set $\hat{s}_1 = o_1$, and $\Sigma_{\hat{s}_1} = \Sigma_1$. Then, we use the equations (1) to recursively update $\hat{s}$ and $\Sigma_{\hat{s}}$. The $\hat{s}_N$ is our final predictions

Note that the Kalman filter treats the $\hat{s}$ and o as vector space. However, in our problem, the observations are probability. The final results $\hat{s}_N$ would not be a probability after **N**-1 Kalman updates. We address this issue by replacing the covariance matrix Σ with variance σ^2. The matrix K now becomes a scalar which turns the Kalman filter into a weighted sum algorithm where each model is weighted by the inverse of its variances.

The overall ensemble algorithm is shown in algorithm (1)

Algorithm 1 Deep ensemble

```
for $n = 1, \dots, N$ do
    Set model n to train mode
    for $m = 1, \dots, M$ do
        Use model n make predictions $o_{n,m}$
    end for
    Calculate the mean and variance of $M$ predictions: $o_n$, $\sigma_n^2$
end for
Set $\hat{s} = o_1$ and $\sigma_{\hat{s}} = \sigma_1^2$
for $n = 2, \dots, N$ do
    $k = \frac{\sigma_n^2}{\sigma_{\hat{s}}^2 + \sigma_n^1}$
    $\hat{s} = k\hat{s} + (1-k)o_n$
    $\sigma_{\hat{s}} = k\sigma_{\hat{s}}k + (1-k)\sigma_n^2(1-k)$
end for
Output $\hat{s}$
```

4. Experimental Results

4.1. Training setting. We use the CIFAR10 dataset[8] to test the effectiveness of our methods. We employ 10%, 50%, 75%, and 100% of training data to show the data efficiency of our methods.

For the learnable deep networks architecture, we used the allcnn[9] structures and remain all hyperparameters related to networks as default values. We initialized 16 models and sampled 15 random transformation layers from PyTorch's AutoAugment [2] while keeping the first model with an identity transformation layer. For each model, we trained 50 epochs with data augmentation.

4.2. Generalization performance. In this section, we show the generalization ability of the ensemble network. We also compare our proposed decision fusion methods with other common techniques such as simple average and majority voting methods.

The testing accuracies of our method are shown in figure 2. The blue bar showed the performance of the best single model on the test dataset. Other colored bars show the results of fusing 16 models with different strategies. The testing accuracy for every single model using all training data are also shown in figure 3

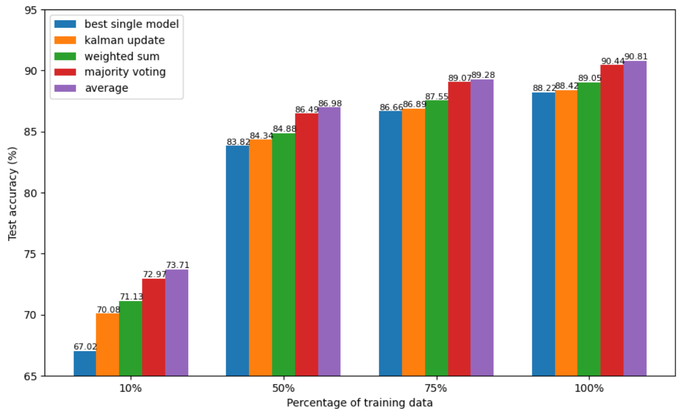

FIGURE 2. Generalization Performance

Overall the ensemble model has better generalization performance than a single model. When using all data to train, the best single model obtain 88.22% accuracy on test data, the ensemble model can still improve the performance by 2.5% by using the simple average strategy.

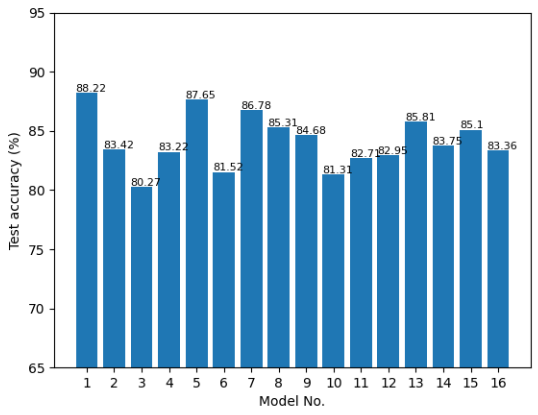

FIGURE 3. Single model testing accuracy

4.3. **Robustness adversarially attack.** In this section, we would add adversarially noises to the input image to test the robustness of the ensemble model.

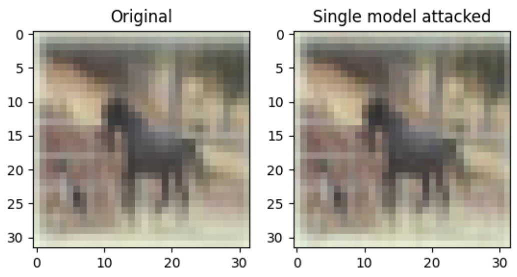

FIGURE 4. An example of attacked image

To generate the attack noises, we first calculate the loss gradients with respect to input images for each model. Then we would use different methods to ensemble **N** gradients. The ensemble methods include: ”single model”, which simply picks the gradient of a best single model (e.g. the model has the best test performance), ”average”, which average the n gradients, and ”weighted sum”, which weights each gradient with the coefficient calculated by variance version Kalman filter. Finally, we scale the ensemble gradient within an ϵ-ball using equation (2):

$$d_k = \epsilon \frac{\nabla l_k}{|\nabla l_k|} \tag{2}$$

Here, l_k is each element in the ensemble gradient and d_k is the elements in the final disturbance. For this experiment, we choose $\epsilon = 0.01$, by contrast, the values in every input channel are in [0,1]. A typical image with a ”single model” attack and its original one is shown in figure 4:

Figure 5 shows the model's performance under adversarial attack. The ensemble method is much more robust to adversarial attacks compared to just using a single model, in which the test performance drops by more than half. Our proposed Kalman filter weighted sum is slightly more robust to the attack than the simple average of the 16 models.

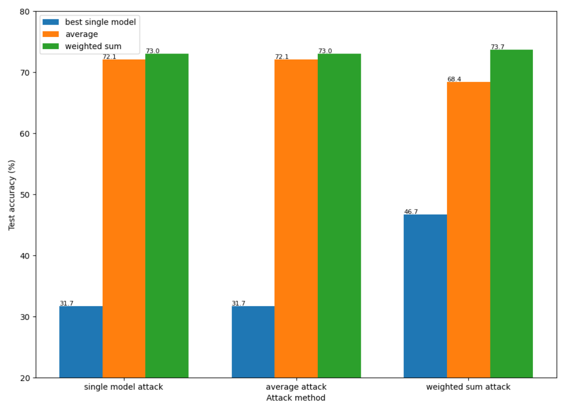

FIGURE 5. Performance under adversarial attack

5. DISCUSSION

In this paper, we proposed a new way to train a set of different neural models for deep ensembles. We tested the effectiveness of our method in terms of generalization and robustness in the CIFAR10 dataset.

We also proposed a weighted sum method to fuse the results of different models and showed it's more robust to adversarial attacks compared to the simple averages. However, we notice simple average and majority voting have better generalization performance than our proposed decision-fused method. It might be because Monte Carlo dropout is not a property way to estimate model uncertainty in our case.

In the future, we will compare our method with other deep ensemble methods and investigate how to properly estimate model uncertainty for our framework.

REFERENCE

[1] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56 (2014), pp. 1929–1958.
[2] Ekin Dogus Cubuk et al. "AutoAugment: Learning Augmentation Policies from Data". In: *CoRR* abs/1805.09501 (2018). arXiv: 1805.09501.
[3] Alex Kendall and Yarin Gal. "What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision?" In: *CoRR* abs/1703.04977 (2017). arXiv: 1703.04977.
[4] Jakob Gawlikowski et al. "A Survey of Uncertainty in Deep Neural Networks". In: *CoRR* abs/2107.03342 (2021). arXiv: 2107.03342.
[5] M.A. Ganaie et al. "Ensemble deep learning: A review". In: *Engineering Applications of Artificial Intelligence* 115 (Oct. 2022), p. 105151. DOI: 10.1016/j.engappai.2022.105151.
[6] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. DOI: 10.48550/ARXIV.1512.03385.
[7] Yarin Gal and Zoubin Ghahramani. *Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning*. 2015. DOI: 10.48550/ARXIV.1506.02142.
[8] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. *CIFAR-10 (Canadian Institute for Advanced Research).*
[9] Pratik Chaudhari. *Allcnn.py*.