Knowledge Diversification in Ensembles of Identical Neural Networks

BMVC 2022 Submission # 0798

Abstract

It is well known that diversity in models is the key to enhancing the performance of an ensemble. In standard neural network ensemble techniques, two or more networks are trained independently and their logits or predictions are combined using a voting procedure or linear combination strategy. This procedure does not incorporate the exchange of information between the base networks of the ensemble. We propose a method for improving learnt representations in an ensemble by employing feature exchange between base models as a part of the training objective. Feature Difference Loss or FDL compels networks in an ensemble to learn diverse features in a Euclidean sense, thereby directly optimizing model diversity. Experiments with ensembles of two, three and four networks show significant performance boosts over competing ensemble techniques. The gains are larger for datasets with fewer examples per class, such as MNIST, CIFAR-10 and CIFAR-100. Positive gains can also be observed in large datasets such as ImageNet. The gains also generalize across several architectures from simple ConvNets to deeper networks such as VGG and ResNets.

1 Introduction

Creating an ensemble of neural networks is a common way to improve the performance of learning-based algorithms. The ILSVRC 2015 winning residual network [15] is an ensemble of six different base ResNet models. Training base models has mostly been done in a model-independent way. It is intuitive that an ensemble performs better if the base models learn different sets of features. Because each model contributes a unique set of features or logits to the ensemble, a broader range of information is captured, increasing generalization.

Random Initialization Ensemble is a method of taking ensembles that involves changing the initialization states of the base models in order to have different trajectories during optimization. One such technique is Deep Ensembles [25]. However such techniques do not optimize the diversity of learned representations during training. It involves performing multiple experiments with various initialization and then choosing a subset of the most diverse models as a part of the ensemble. The SnapShot Ensemble [18] [42] combines checkpoint collection with cyclic learning rate methods such as SGD with warm restarts [30] to generate ensembles. Fast Geometric Ensemble [11] traverses high accuracy paths between modes to generate ensembles by combining modes. Checkpoint ensembles [6] uses checkpoints during training as ensemble candidates, which are then combined near the end of training to generate an ensemble. Adaptive Ensemble [19] based on Confidence Intervals reduces computation overhead by discarding ensemble executions for input samples with high softmax outputs. Auto-Ensemble [44] creates an ensemble using checkpoint collection and a learning rate scheduling algorithm.

Loss functions and training routines that directly target model diversity has been explored in [20, 28, 29, 33, 40, 48]. In this paper, we try to directly optimize the diversity of features generated by many identical neural networks. We do this by introducing feature difference loss (FDL) into the training routine. In order to achieve a diverse set of features, the base models compete against each other in a minimax fashion with the injection of the FDL loss. In this paper, our motivation for this work are the following: to provide a trainable way of improving diversity amongst base models in an ensemble, and to have a stable training routine with minimal set of hyperparameters for wide applicablility and reproducibility. To this end, our contributions in this paper are: **(a)** we propose a method called FDL (Feature Difference Loss) which takes the Euclidean distance between feature representations into account during training, and with that we force the base networks towards diverse solution states at convergence. **(b)** We propose a training method that is completely devoid of hyperparameters. We split the training procedure in several phases that are independent of each other. **(c)** We provide extensive experimentation of various models and datasets ranging from simple ConvNets to deeper networks such as VGG and also residual architectures such as ResNet-18 and ResNet-50, which we train on MNIST, CIFAR-10, CIFAR-100 and ImageNet-1K for upto four networks. We show that FDL provides consistent improvement in all cases.

2 Method

2.1 Ensemble Architecture

We start with N identical base networks. Without loss of generality, we portray the $N = 2$ case. In Fig. 1(a), we show networks N_1 and N_2. The two networks produce two sets of prediction vectors independently, and then those two vectors are concatenated and fed to an ensemble head network, N_E. The ensemble head comprises a single layer of 1×1 convolution kernels and it learns the optimal linear combination to combine the prediction vectors at its input. All networks N_1, N_2 and N_E are trained using cross-entropy loss. However, we also have other losses during training which we portray in next section.

If $\vec{v}_1$ and $\vec{v}_2$ are the prediction vectors for an input mini-batch I, then for networks N_1 and N_2 respectively, we have:

$$\vec{v}_1 = (v_{11}, \cdots, v_{1p}) = \mathcal{F}_1 * I \tag{1}$$

$$\vec{v}_2 = (v_{21}, \cdots, v_{2p}) = \mathcal{F}_2 * I \tag{2}$$

where $\mathcal{F}_1$ and $\mathcal{F}_2$ represents the parameters of N_1 and N_2 respectively, and $*$ is the convolution operator.

The output of the ensemble head can then be represented as,

$$\vec{v}_e = (v_1, \cdots, v_p) = K * (v_{11}, \cdots, v_{1p}, v_{21}, \cdots, v_{2p}) \tag{3}$$

where K is a set of 1×1 convolution kernels.

During the training of the ensemble head, no gradient is passed back to the base networks. The training routine of the base networks and the ensemble head is performed in five phases, which we describe in section 2.3.

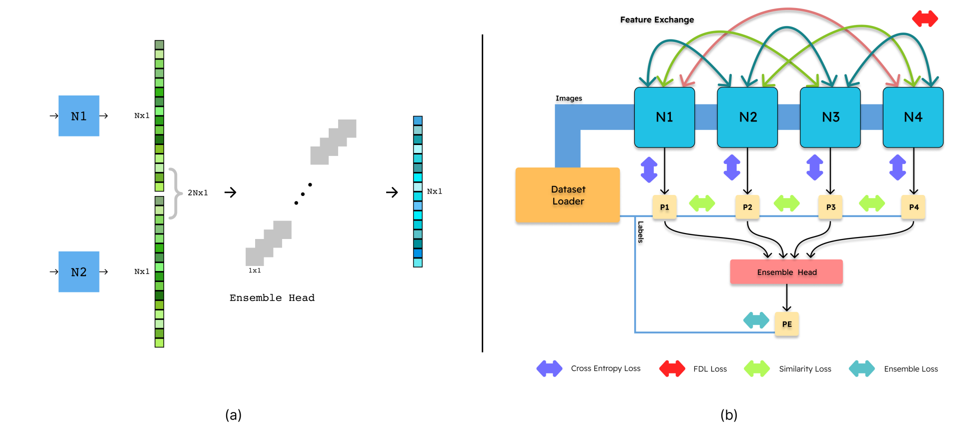

Figure 1: (a) The architecture of the ensemble head network. Predictions of two identical networks are combined into a single prediction vector using $2N$ linear units or $2N$ (1×1) convolution. (b) The architecture of four identical neural networks trained with FDL. All networks share a common minibatch. Features are exchanged and the networks weights are updated so as to maximize feature differences between them. The double sided arrows represent the different loss functions and their position indicates the location where they are invoked. Prediction vectors (P1, P2, P3, P4) are combined using an ensemble head to create PE.

2.2 Feature Difference Loss (FDL)

Given a convolutional base neural network N, we define W_i as the i^{th} convolutional layer of the network. A feature vector F_{i-1} is fed into W_i, and the output F_i is generated.

$$F_i = W_i * F_{i-1} \tag{4}$$

We denote the shape of the tensor F_i as (B, C, H, W) where B is the number of images in a batch, C is the number of channels of each image and $H \times W$ is the resolution of each image. In Fig. 1(b), we show the architecture and the computation graph of four neural networks trained using our proposed method. First we portray the two network case, following which we extend our method for ensembles of more than two networks in section 2.3. Given, two identical networks N_1 and N_2, we compute the feature difference loss for the i^{th} layer as:

$$L_i^{N_1,N_2} = \frac{1}{BCHW} \sum_{b=0}^{B-1} \sum_{c=0}^{C-1} \sum_{h=0}^{H-1} \sum_{w=0}^{W-1} (F_i^{N_1}(b,c,h,w) - F_i^{N_2}(b,c,h,w))^2 \tag{5}$$

The value of the feature loss is then summed up and averaged over the depth (D) of the network as:

$$L_{FDL}^{N_1,N_2} = \frac{1}{D} \sum_{d=0}^{D-1} L_d^{N_1,N_2} \tag{6}$$

Given the cross-entropy loss of network N_1 and N_2 as L_1 and L_2 respectively, we define the similarity loss function as:

$$S^{N_1,N_2} = (L_1 - L_2)^2 \tag{7}$$

Ideally, we want to perform maximization of the feature difference loss, Eq. 6. However, as is commonly followed in the training routines of generative adversarial networks (GANs), the discriminator's loss functions are not maximized, rather the negative of it is minimized. Here too, we minimize the negative of the FDL loss function, Eq. 6.

We pose the optimization criterion as:

$$N_E^*(W) = \underset{W}{\operatorname{argmin}}(-L_{FDL}^{N_1,N_2} + k\ S^{N_1,N_2} + k_1\ L_X(\hat{y}_{N_1}, y) + k_2\ L_X(\hat{y}_{N_2}, y)) \tag{8}$$

L_X is the cross-entropy loss function that acts on the the prediction vector $\hat{y}$ and the ground truth y. The constants k, k_1, k_2 controls the weights ('importance') of the individual losses. However, directly optimizing Eq. 8 requires careful tuning of the constants. To avoid this, we train the entire system in several different phases with each phase targeting a single loss function and it entirely eliminates the task of tuning the constants of Eq. 8.

2.3 Training Phases

To train the ensemble network with the different losses of Eq. 8, we develop a training algorithm that is akin to the way generative adversarial networks are trained, i.e. in phases. We have five different phases of training as portrayed in Fig. 2. Without loss of generality we discuss the $N = 2$ case, i.e. two base models and a single ensemble head as the ensemble network. During training, we train N_1 and N_2 independently from a different seed for a few epochs. We denote this pretraining phase as the zero'th phase as it occurs only once in the training cycle. After the pretraining phase is completed, phase 1 through 4 repeats in a cycle occuring once every iteration until convergence. If we invoke the FDL loss function from the very first iteration (i.e. without invoking phase zero), the training progresses haphazardly, leading to instability. So we pretrain N_1 and N_2 for a few epochs depending on the dataset. Smaller datasets such as the CIFAR datasets, the number of pretraining epochs are often just 1. Larger datasets require more pretraining. We mention all training details and hyperparameters for complete reproducibility in the supplementary section of this article. It is important to note that we stop training much before convergence. At this state, the networks have not reached their convergence point yet. If we invoke the FDL loss from this point onwards, the two neural networks diverge from their paths and arrive at different and diverse optimal points. The FDL loss maximizes the gap between the two optimal points, by making the networks compete against each other. The farther the final optimal points of each base network, the better it is for the ensemble as a whole.

In the first phase, we train the base models with a single minibatch from the training dataset and immediately move on to the next phase. In the second phase of training, we invoke the similarity loss. In a two network setting, we take the loss values of the two networks and compute the mean squared error between them. Since the two loss 'tensors' are a part of the computation graph, the effect of the mean squared error computation backpropagates through the entire network. The aim of the similarity loss function is to stabilize training. With the similarity loss, the cross-entropy loss values are kept within a reasonable range of one another. The reason why we choose to compare the loss values instead of the prediction vectors is the following. The loss value (a scalar) represents the height of a point on the loss landscape whose value is determined by the parameters of the network and the input vector

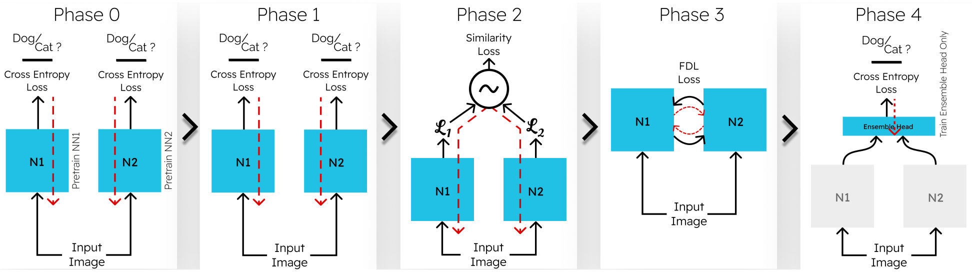

Figure 2: The training pipeline of two networks N1, N2 along with the ensemble head network trained with FDL loss. The red dotted line indicates the flow of gradients during back-propagation.

(a minibatch of training images). Several different points on this loss landscape can have the same loss value. By comparing the loss values of the base models, we force the network to arrive at a 'similar' quality optima at convergence. Due to the fact that stability issues are typical when training neural networks with negative loss landscapes (FDL or GAN), the presence of the similarity loss function is essential. If in case we train the base models to only minimize the negative of the FDL loss without having the similarity loss in the training pipeline, we observe that in some cases the weights of one of the base networks goes to zero, while the other network arrives at a perfectly good optimal convergence point. This is intuitively plausible, as it is optimal to 'sacrifice' the accuracy of one of the networks to have the FDL loss at a maximum value since the feature vector differences will then be at their greatest. The presence of the similarity loss prohibits this phenomena. Also, in our formulation of the similarity loss we do not compare the prediction vectors and instead choose to just compare the loss values. This is because it is possible for the networks to have the same loss value for different prediction vectors. Comparing the prediction vectors would instead lead the networks to arrive at the exact same optimal point, which will lead to a loss in ensemble diversity. A group of networks that output the same prediction vectors for any input images does not provide any additional information to the ensemble.

In case of N networks, we find that invoking the similarity loss between every pair of ${}^{N}C_2$ networks is computationally intensive and so we randomly choose any two networks per iteration and invoke the similarity loss objective only on those two networks. A uniform random sampler ensures that all pairs of networks are choosen with equal likelihood.

In the third phase, we invoke the FDL loss. In this step, we collect all output activations from only the convolutional layers. We broadcast the activation tensors to all other networks in the pool, and compare the mean squared difference between them in a pairwise fashion as per Eq. 5. This means that for N networks, the total number of transfers is ${}^{N}C_2$. This is crucially important as it ensures information exchange between all networks in the pool, which is the crux of our algorithm. The final FDL loss for N networks is the average over all ${}^{N}C_2$ transfers. Instead of maximizing the FDL loss, we minimize the negative of it.

$$L^{*}_{FDL} = \frac{1}{{}^{N}C_2} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} L^{i,j}_{FDL} \tag{9}$$

Finally in the fourth phase of training, we train the ensemble head network N_E. The input to N_E is a concatenated array of all logits accumulated from all base networks. While

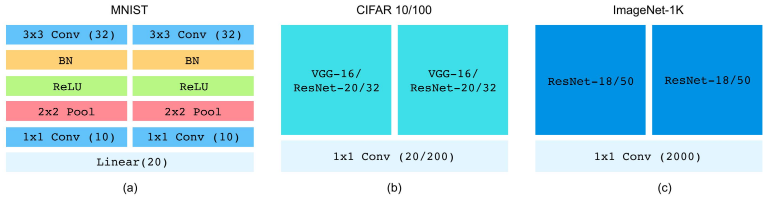

Figure 3: We perform experiments on the following models. **(a)**: A simple two layer network on MNIST. **(b)**: Deeper networks, VGG-16 and ResNet-20/32 on CIFAR-10/100. **(c)**: Larger models: ResNet variants on ImageNet-1K.

training the head network, we do not pass the gradients back to the base networks. This is an important step required to measure the efficacy of the FDL loss. If the gradient that is computed at the ensemble end is used to update the weights of the base networks, then there is a possibility that the base networks improve the ensemble performance through this gradient update and not through FDL. Hence we choose to keep the weights of the base networks completely disjoint from the weights of the N_E network.

In none of the training phases, we optimize the ensemble's performance by the loss computed at the ensemble head. Instead, we use feature exchange and FDL to allow the network to arrive at better optimas, by nudging it towards different solutions where the learnt feature representations are diverse. Each iteration comprises phase 1 through 4 exactly once. The process continues until convergence.

3 Experiments

Our first experiment consists of a simple network (Fig. 3(a)) that has a single convolution layer with M units of 3×3 filters. We train all base networks on the MNIST dataset. We vary M from 1 to 256, and for each experiment and we record the performance of the single full-width network (of M filters) and the ensemble performance of two half-width networks ($M/2$ filters each) (Fig. 4(a)). We observe that in almost every case, the two network FDL ensemble learns better representations and outperform the single full-width network.

We also experiment with deeper networks, such as VGG-16 [38], ResNet-20 and ResNet-32 [15] and with the CIFAR-10 and CIFAR-100 [22] datasets. We use a single layer of 1×1 convolution units as the ensemble head, Fig. 3(b). We portray the observed FDL ensemble accuracies Fig. 4(b). From table 1, we observe that ensembles trained with FDL outperform other methods. FDL scores higher than SSE (SnapShot Ensemble), FGE (Fast Geometric Ensemble), and AE (Auto-Ensemble). In CIFAR-10 experiments, FDL outperforms AE Full by 1% and FGE by 0.59%. In CIFAR-100 experiments, it outperforms FGE by 0.56%.

On the ImageNet-1K [35] (Fig. 3(c)), we see a significant improvement of 1.58% in ResNet-18, and an improvement of 0.67% in ResNet-50 2. From table 2, we observe that FDL enables ensmebles of two base ResNet models to achieve higher accuracies over the standard non-FDL methods. This indicates that FDL is a diversity enoucouraging loss function. Two ResNet-50 models trained with FDL outperforms SSE (SnapShot Ensemble) by 0.39% and FGE by 0.37% (table 1).

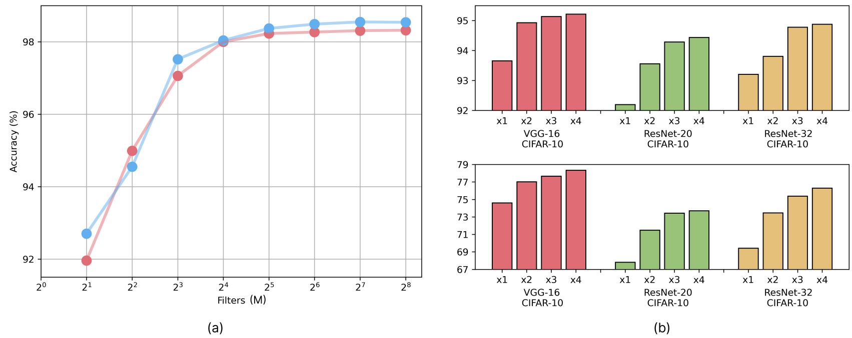

Figure 4: (a) Number of Filters (M) vs Accuracy plot of a one layer ConvNet. Red line = single network of M filters. Blue line = FDL ensemble of 2x networks (M/2 filters each). (b) Multi-network FDL ensemble accuracies for various models (VGG-16, ResNet-20, ResNet-32) on (CIFAR-10 and CIFAR-100).

Table 1: Left: Comparisons of ensemble methods in image classification task, performed on the CIFAR-10 and CIFAR-100 datasets.

Method	Accuracy (%)	
	CIFAR-10	CIFAR-100
VGG-16 (1x) baseline	93.66	74.61
VGG-16 RIE (2x)	93.7	76.95
VGG-16 SSE [18]	94.05	75.31
VGG-16 FGE [11]	94.34	76.46
VGG-16 AE Full [44]	93.93	72.16
VGG-16 FDL (2x) [ours]	**94.93**	**77.02**

Method	Accuracy (%)
	ImageNet-1K
ResNet-50 (1x) baseline	76.38
ResNet-50 RIE (2x)	76.96
ResNet-50 SSE [18]	76.67
ResNet-50 FGE [11]	76.69
ResNet-50 FDL (2x) [ours]	**77.06**

4 Ablation Studies

We discuss the efficacy of FDL loss by performing the same training routine of Fig. 2, with the same hyperparameters, but as two separate experiments: 'with' FDL and 'without' FDL. In table 2 (right), column **[C]** denotes the accuracies of ensembles of two identical base networks trained 'with' and 'without' FDL. Column **[A]** denotes the performance of each base network for which the ensemble in **[C]** is obtained. Column **[B]** denotes the individual best accuracy obtained during the entire training process. We observe that with FDL, not only the ensemble performance is better than without it, but also the base models trained with FDL outperform base models trained without FDL.

We also provide a visualization of feature difference maps between two VGG-16 models in Fig. 6 trained with and without FDL. We observe that FDL forces higher feature differences among base models, which we hypothesize as the primary contributor to increase in diversity, leading to the observed higher accuracies in FDL ensembles.

In Fig. 5(a) we plot different loss plots and test accuracy plots for 2x VGG-16 ensemble on CIFAR-100. The FDL loss decreases initially (red line). After 200 epochs it starts to increase, even though the cross-entropy loss of N_1 keeps on decreasing (blue line). The yellow circle marks the epoch where maximum ensemble accuracy is achieved. In Fig. 5(b, c) we perform additional ablation studies, where we train the networks with FDL and

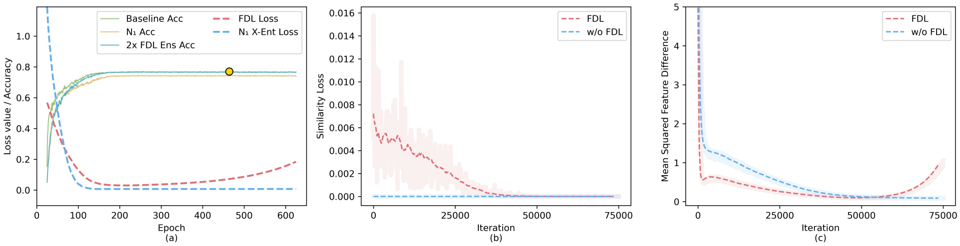

Figure 5: 2x VGG-16 ensemble training on CIFAR-100. (a) Loss and accuracy plots. (b) Similarity loss plots. (c) Plot of Mean Squared Feature Differences during the training.

Table 2: Left: Test accuracies on the test set of ImageNet-1K. Right: Results of ablation study performed on the ResNet-18 network on ImageNet. We observe that FDL ensemble performs better than the non-FDL emsemble.

Model	ImageNet Accuracy (%)		
	1x Network	2x Ensemble	2x FDL Ensemble
ResNet-18	70.012	71.014	**71.594**
ResNet-50	76.386	76.964	**77.06**

Model	Network 1		Network 2		Ensemble
	[A]	**[B]**	**[A]**	**[B]**	**[C]**
ResNet-18	69.902	70.012	69.632	69.748	71.014
+ FDL	69.892	69.998	69.666	69.73	**71.594**
ResNet-50	76.042	76.386	75.97	76.306	76.964
+ FDL	76.186	76.246	76.052	76.108	**77.06**

without FDL. In Fig. 5(b), we plot the similarity loss during training. We observe that in both cases the networks final attain close to zero similarity loss. This indicates that both networks achieve similar optimal points at convergence. However, in case of FDL, the inital few iterations the similarity loss is quite high and fluctuates rapidly. This indicates that with FDL the networks' states initially moves away from each other jumping across different local optimas as it explores the entire loss landscape. Whereas in the without-FDL scenario, the similarity loss is very close to zero right from the beginning of training. In Fig. 5(c) we plot the Mean Squared Feature Differences during training for both 'with' and 'without' FDL cases. We observe that FDL loss steadily decreases to zero if the networks are not trained with it (blue line), but increases later down during training (red line).

5 Related Works

Several methods exist for learning ensembles such as [14], [23] and [8]. Negative correlation learning [1] and error independent ensembling [13] has also been utilized to create ensembles. In many cases, the dataset is often divided into many overlapping or non-overlapping subsets and fed to the base models and with enough hardware, the models can be simultaneously trained [12] [47] and their predictions aggregated. Often, the base models are trained with different hyperparameters to reach different solution points. The base models are combined using model averaging, various types of voting (majority voting, soft voting or plurality voting) or weighted consensus (boosting voting) [43]. In collaborative learning [39], many ensemble head networks are used on the same network to improve robustness and generalization. A similar idea is explored in [49] and from the perspective of pruning in [26].

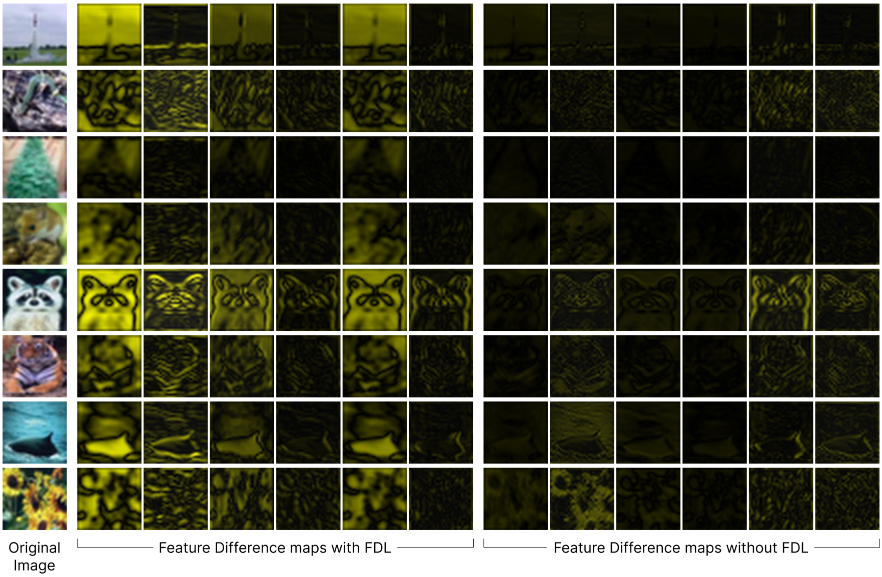

Figure 6: Differences between feature maps of two VGG-16 base models trained on the CIFAR-100 with and without FDL. The intensity of yellow on each feature map indicates the strength of the mean squared difference between two feature maps from the same layer and the same channel.

Some well-known techniques for obtaining ensembles from base models include Random Forest [5] [17], Bagging [4], Boosting [36], and AdaBoost [37], Random Layer Sampling [27]. EnsembleBench [43] tries to minimize the number of possible ensemble candidates required for evaluation while maximizing overall ensemble performance. It is widely accepted that maximizing diversity is critical to maximizing ensemble benefits. It is obvious that if all the base models are exact replicas of each other, the ensemble's performance is no better than any of its base models. Correlation between diversity and ensemble performance is explored in [24] and [32].

Co-training losses ensure that all base models eventually perform similarly with the ensemble output being an average of all predictions. Student-teacher based ensembles has been explored in [31, 47] using distillation [16]. Co-training has been used to achieve diversification [2, 3, 12, 34, 45]. Neural Architecture Search (NAS) based methods have also been used to create ensembles [7, 9, 10, 21, 41, 46, 50].

6 Conclusion

We present a method of optimizing ensemble performance of identical networks by introducing feature difference losses and a custom training routine. FDL maximizes feature differences by pushing the networks towards more diverse solutions. From experiments on shallow networks and smaller datasets, to larger models and larger datasets show positive performance gains in all scenarios. FDL encourages high diversity in base model by directly optimizing Euclidean difference between pairs of feature sets across all ${}^{N}C_2$ combinations of N networks.

References

[1] Monther Alhamdoosh and Dianhui Wang. Fast decorrelated neural network ensembles with random weights. *Information Sciences*, 264:104–117, 2014.

[2] Tanmay Batra and Devi Parikh. Cooperative learning with visual attributes. *arXiv preprint arXiv:1705.05512*, 2017.

[3] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100, 1998.

[4] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[5] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[6] Hugh Chen, Scott Lundberg, and Su-In Lee. Checkpoint ensembles: Ensemble methods from a single training process. *arXiv preprint arXiv:1710.03282*, 2017.

[7] Corinna Cortes, Xavier Gonzalvo, Vitaly Kuznetsov, Mehryar Mohri, and Scott Yang. Adanet: Adaptive structural learning of artificial neural networks. In *International conference on machine learning*, pages 874–883. PMLR, 2017.

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

[9] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *The Journal of Machine Learning Research*, 20(1):1997–2017, 2019.

[10] Jiemin Fang, Yukang Chen, Xinbang Zhang, Qian Zhang, Chang Huang, Gaofeng Meng, Wenyu Liu, and Xinggang Wang. Eat-nas: elastic architecture transfer for accelerating large-scale neural architecture search. *arXiv preprint arXiv:1901.05884*, 2019.

[11] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8803–8812, 2018.

[12] Yixiao Ge, Dapeng Chen, and Hongsheng Li. Mutual mean-teaching: Pseudo label refinery for unsupervised domain adaptation on person re-identification. *ICLR*, 2020.

[13] Giorgio Giacinto and Fabio Roli. Design of effective neural network ensembles for image classification purposes. *Image and Vision Computing*, 19(9-10):699–707, 2001.

[14] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[17] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[18] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[19] Hiroshi Inoue. Adaptive ensemble prediction for deep neural networks based on confidence level. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1284–1293. PMLR, 2019.

[20] Siddhartha Jain, Ge Liu, Jonas Mueller, and David Gifford. Maximizing overall diversity for improved uncertainty estimates in deep ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4264–4271, 2020.

[21] Haifeng Jin, Qingquan Song, and Xia Hu. Auto-keras: An efficient neural architecture search system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1946–1956, 2019.

[22] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[23] Anders Krogh and Jesper Vedelsby. Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, 7, 1994.

[24] Ludmila I Kuncheva and Christopher J Whitaker. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, 51(2): 181–207, 2003.

[25] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

[26] Duong H Le, Trung-Nhan Vo, and Nam Thoai. Paying more attention to snapshots of iterative pruning: Improving model compression via ensemble distillation. *arXiv preprint arXiv:2006.11487*, 2020.

[27] Hakmin Lee, Hong Joo Lee, Seong Tae Kim, and Yong Man Ro. Robust ensemble model training via random layer sampling against adversarial attack. *arXiv preprint arXiv:2005.10757*, 2020.

[28] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks. *arXiv preprint arXiv:1511.06314*, 2015.

[29] Yong Liu and Xin Yao. Ensemble learning via negative correlation. *Neural networks*, 12(10):1399–1404, 1999.

[30] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[31] Shuchang Lyu, Qi Zhao, Yujing Ma, and Lijiang Chen. Make baseline model stronger: Embedded knowledge distillation in weight-sharing based ensemble network. 2021.

[32] Derek Partridge and Wojtek Krzanowski. Software diversity: practical statistics for its measurement and exploitation. *Information and software technology*, 39(10):707–717, 1997.

[33] Tim Pearce, Felix Leibfried, and Alexandra Brintrup. Uncertainty in neural networks: Approximately bayesian ensembling. In *International conference on artificial intelligence and statistics*, pages 234–244. PMLR, 2020.

[34] Siyuan Qiao, Wei Shen, Zhishuai Zhang, Bo Wang, and Alan Yuille. Deep co-training for semi-supervised image recognition. In *Proceedings of the european conference on computer vision (eccv)*, pages 135–152, 2018.

[35] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115 (3):211–252, 2015.

[36] Robert E Schapire. A brief introduction to boosting. In *Ijcai*, volume 99, pages 1401–1406. Citeseer, 1999.

[37] Robert E Schapire. Explaining adaboost. In *Empirical inference*, pages 37–52. Springer, 2013.

[38] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[39] Guocong Song and Wei Chai. Collaborative learning for deep neural networks. *ICLR*, 2018.

[40] Andrew M Webb, Charles Reynolds, Dan-Andrei Iliescu, Henry Reeve, Mikel Luján, and Gavin Brown. Joint training of neural network ensembles. *stat*, 1050:12, 2019.

[41] Charles Weill, Javier Gonzalvo, Vitaly Kuznetsov, Scott Yang, Scott Yak, Hanna Mazzawi, Eugen Hotaj, Ghassen Jerfel, Vladimir Macko, Ben Adlam, et al. Adanet: A scalable and flexible framework for automatically learning ensembles. *arXiv preprint arXiv:1905.00080*, 2019.

[42] Long Wen, Liang Gao, and Xinyu Li. A new snapshot ensemble convolutional neural network for fault diagnosis. *Ieee Access*, 7:32037–32047, 2019.

[43] Yanzhao Wu, Ling Liu, Zhongwei Xie, Juhyun Bae, Ka-Ho Chow, and Wenqi Wei. Promoting high diversity ensemble learning with ensemblebench. *arXiv preprint arXiv:2010.10623*, 2020.

[44] Jun Yang and Fei Wang. Auto-ensemble: An adaptive learning rate scheduling based deep learning model ensembling. *IEEE Access*, 8:217499–217509, 2020.

[45] Taojiannan Yang, Sijie Zhu, Chen Chen, Shen Yan, Mi Zhang, and Andrew Willis. Mutualnet: Adaptive convnet via mutual learning from network width and resolution. In *European Conference on Computer Vision*, pages 299–315. Springer, 2020.

[46] Sheheryar Zaidi, Arber Zela, Thomas Elsken, Chris C Holmes, Frank Hutter, and Yee Teh. Neural ensemble search for uncertainty estimation and dataset shift. *Advances in Neural Information Processing Systems*, 34:7898–7911, 2021.

[47] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4320–4328, 2018.

[48] Tianyi Zhou, Shengjie Wang, and Jeff A Bilmes. Diverse ensemble evolution: Curriculum data-model marriage. *Advances in Neural Information Processing Systems*, 31, 2018.

[49] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.

[50] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.