HIGHLY PARALLEL DEEP ENSEMBLE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we propose a novel highly parallel deep ensemble learning, which leads to highly compact and parallel deep neural networks. The main idea is to *split data into spectral subsets; train subnetworks separately; and ensemble the output results in the inference stage*. The proposed method has parallel branches with each branch being an independent neural network trained using one spectral subset of the training data. It ensembles the outputs of the parallel branches to produce an overall network with substantially stronger generalization capability. It can also scale up the model to the large scale dataset with limited memory. The joint data/model parallel method is amiable for GPU implementation. Due to the reduced size of inputs, the proposed spectral tensor network exhibits an inherent network compression, which leads to the acceleration of training process. We evaluate the proposed spectral tensor networks on the MNIST, CIFAR-10 and ImageNet data sets, to highlight that they simultaneously achieve network compression, reduction in computation and parallel speedup. Specifically, on both ImageNet-1K and ImageNet-21K dataset, our proposed AlexNet-spectral, VGG-16-spectral, ResNet-34-spectral, CycleMLP-spectral and MobileVit-spectral networks achieve a comparable performance with the vanila ones, and enjoy up to $4\times$ compression ratio and $1.5\times$ speedups.

1 INTRODUCTION

Deep neural networks (DNNs) [1] have made impressive successes in many applications, such as computer vision [2][3][4], online game [5][6][7], natural language processing [8][9][10], autonomous driving [11][12][13], and robotics [14][15][16]. However, DNNs are memory-intensive and computation-intensive, which are two major challenges for wider adoption, e.g., in Internet of Things (IoT) applications [17]. Modern DNNs may have billions of parameters that consume excessive amount of memory and usually require long training time. Besides, it has been a great challenge to train models on large datasets for netter performance, such as ImageNet-1K and ImageNet-22K for image classification, Apache software foundation public mail archives for natural language procecssing and Waymo Open for autopilot. All of them are more than 100 GB in storage which makes it difficult to load in memory at the same time. We have to consistently pull and push batches of dataset into memory from the storage to train the models, which leads to an unefficient training. For example, AlexNet [18] consists of three fully-connected layers and five convolutional layers, containing 60 million parameters and consuming about 250 MB of memory and about 40 hours for training on ImageNet data set [19].

Many existing works have been proposed to alleviate the challenge of training big models on large scale datasets. Wei *et al*. [20] propose to select a subset from the large scale training dataset to make a balance between the training time and the classification performance. He *et al*. [21] further propose a block-based sampling method for large scale dataset. However, the sampling techniques suffer from an issue of the drop of task performance, which hinders its application to real scenes. On the other hand, some works focus on the distribute training to handle the large scale data [22][23][24]. But the communication between different training nodes becomes the bottleneck. Although some work has researched on the communication efficient scheme [25][26][27], the inherent low transfer speed and bandwidth between computing nodes limit the efficiency of distributed learning. In our work, we adopts a different way that we reduce the dimension of training data by splitting the dataset, which leads to a compression of models and a communication-free parallel ensemble learning scheme for efficient training.

In this paper, we propose a unified approach that simultaneously achieves both model compression and parallel learning *without* communication overhead. The key technique is a novel spectral tensor layer that enables a joint *data/model-parallel* implementation of a DNN as follows: 1) The training data set is split into multiple orthogonal spectral sets; 2) The neural network is split into parallel branches with each branch being a conventional neural network, that are trained asynchronously and independently on the corresponding spectral sets; 3) The outputs of the parallel branches are finally ensembled to yield an overall neural network with substantially stronger generalization capability than that of those parallel branches.

We evaluate the proposed spectral tensor networks on the MNIST, CIFAR-10 and ImageNet data sets, to highlight that they simultaneously achieve network compression, reduction in computation and parallel speedup. Specifically, on both ImageNet-1K and ImageNet-21K dataset, our proposed AlexNet-spectral, VGG-16-spectral, ResNet-34-spectral, CycleMLP-spectral and MobileVit-spectral networks achieve a comparable performance with the vanila ones, and enjoy up to $4\times$ compression ratio and $1.5\times$ speedups.

The remainder of this paper is organized as follows. Section 2 presents an brief overview of the related work. Section 3 presents the scheme of highly parallel deep ensemble learning. Section 4 presents the experimental results and we conclude this paper in Section 5.

2 Related Works

Signal processing for deep learning: Many studies have focused on the deep learning using conventional signal processing techniques. Darestani *et al*. [28] proposes to use compressive sensing to measure the robustness of deep learning. Lu *et al*. [29] apply Kalman fliter to deep neural network for video compression. Tseng *et al*. [30] uses the Fourier transform of input-level attribution scores as attribution prior to improve the interpretability and stability of deep models for genomics. Among them, Spectral-based methods, which generalize deep learning models to non-Euclidean domains, have achieved great success. Levie *et al*. [31] proposes a spectral domain convolution architecture CayleyNet. Chang *et al*. [32] embed the spectral mechanism into attention-based graph neural network. Li *et al*. [33] propose a graph deconvolution networks with inverse filters in spectral domain. Yi *et al*. [34] propose SyncSpecCNN which enables weight sharing by parametrizing kernels in the spectral domain.

Ensemble learning: Ensemble learning [35] is a powerful technique in machine learning community, especially for AI competitions like Kaggle and ILSVRC [36]. It trains multiple weak learners for a same problem and combines all of them for inference. Many researchers have discovered that ensemble learning can be used to mitigate some challenges in machine learning methods, such as class imbalance [37], curse of dimensionality [38] and concept drift [39]. Generally used ensemble learning methods comprises of adaBoost [40], bagging [41], random forest [42], gradient boosting machine [43] *etc*. Some researchers also apply ensemble learning methods to deep learning. Chen *et al*. [44] propose to ensemble multiple convolution neural networks for hyperspectral image classification. Lee *et al*. [45] propose a simple unified framework which ensembles multiple actors to make final decision to make improvement in deep reinforcement learning. Lin *et al*. [46] propose to ensemble graph neural networks to alleviates the nonrobustness and oversmoothing issues of the models.

In our work, we leverage the signal processing techniques to transform the data into an independent and orthogonal representation in subspace, which provides convenience for ensemble learning. Besides, different from the sampling methods in conventional ensemble methods, our proposed method reduces the dimension of dataset, leading to a compressed model and faster training process.

3 Highly Parallel Deep Ensemble Learning

Notations: We use lowercase, boldface lowercase, boldface capital, and calligraphic letters to denote scalars, vectors, matrices, and tensors, e.g., $a \in \mathbb{R}$, $\boldsymbol{a} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{n_1 \times n_2}$, $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, respectively. We use $\mathcal{A}(:,:,k),\ \mathcal{A}(:,j,:),\ \mathcal{A}(i,:,:)$ to denote the frontal, lateral, and horizontal slices.

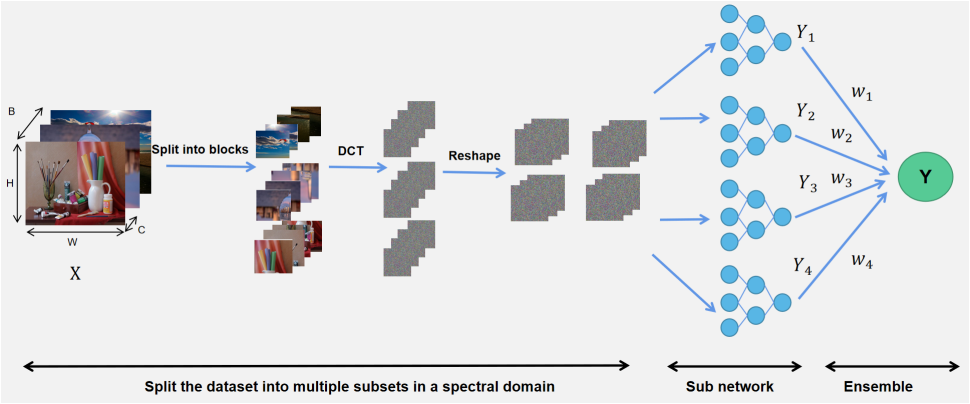

Figure 1: Overview of our highly parallel deep ensemble learning method.

3.1 Overview

Our highly parallel deep ensemble learning method is shown in Fig. 1. The process of our proposed method mainly consists of three steps, namely data preprocessing, parallel training of subnetworks, and ensemble results.

- First, the batched data is pre-processed to the spectral datasets by discrete cosine transform (DCT) along the last dimension and split into Q spectral sub-datasets.
- Second, Q sub-networks are training on Q different spectral sub-datasets in parallel.
- Finally, in the inference stage, a new sample is split into Q spectral data, and feed into these Q subnetworks. The final result is an ensemble of the Q outputs.

3.2 Methodology

Split the dataset into multiple subsets in a spectral domain: Gray or color images are represented as tensor $\mathcal{X} \in \mathbb{R}^{m\times H\times W}$ or $\mathcal{X}' \in \mathbb{R}^{m\times C\times H\times W}$, respectively. We split it into multiple subsets in a spectral domain as follows:

- First, we transform $\mathcal{X}$ to the spectral domain $\widetilde{\mathcal{X}} \in \mathbb{R}^{m\times H\times W}$ by using DCT along certain dimension, e.g., the third dimension. We parallel this process on GPU using the torch-dct library. Then $\widetilde{\mathcal{X}}$ be split into $Q = W$ frontal slices $\widetilde{\mathcal{X}}(:,:,k) \in \mathbb{R}^{m\times H}$ with $k = 1, ..., Q$.
- Then, we split $\mathcal{X}$ into Q parts and stack them as $\hat{\mathcal{X}} \in \mathbb{R}^{m\times C\times \frac{H}{q_1}\times \frac{W}{q_2}\times Q}$ with $Q = q_1 \times q_2$.
- Next, we transform $\hat{\mathcal{X}}$ to the spectral domain $\widetilde{\mathcal{X}} \in \mathbb{R}^{m\times C\times \frac{H}{q_1}\times \frac{W}{q_2}\times Q}$ by performing DCT on $\hat{\mathcal{X}}$ along its last dimension.

It should be noted that DCT transformed $\mathcal{X}$ to $\widetilde{\mathcal{X}}$ which is represented in a spectral space spanned by a series of orthogonal bases. With our proposed data splitting, each sub-dataset remains independent and orthogonal to each other. Using ensemble learning, different models trained on different sub-datasets learn an independent and orthogonal information, which leads to an improvement of generalization and robustness of models.

The main advantage of our proposed data splitting method is that we can train models on large scale datasets with limited memory. Because we can split the spectral dataset into multiple sub-datasets, which can be loaded in the memroy for model training. Moreover, the reduced dimension of input leads to a model compression. The scheme that trianing multiple sub-networks on muliple independent sub-datasets fits ensemble learning well. Thus, the data/model joint parallel scheme is amaible for GPU acceleration.

Train sub-networks: We consider supervised learning and assume that the training data set contains m samples with C categories, i.e., $\{(\mathcal{X}, \boldsymbol{L}\}$, where $\mathcal{X} \in \mathbb{R}^{m\times H\times W}$ is the training data with n

Table 1: Upper bounds for model compression and parallel speedup. For fully connected networks, n denotes the input size of each sub-network, r is the maximum number of neurons of each layer in sub-network, and there are Q branches. For convolutional networks, there are B branches.

-	Fully Connected	Convolutional (1D)
Compression ratio	$O(nQ/2r)$, $O(nQ^2/2r)$	$O(B)$, $O(B^2)$
Reduction in computation	$O(nQ/2r)$, $O(nQ^2/2r)$	$O(B)$, $O(B^2)$
Parallel speedup	$O(Q)$	$O(B)$

samples and $\boldsymbol{L} \in \mathbb{R}^{m\times C}$ is the corresponding categories matrix such that if the s-th sample $\mathcal{X}(s)$ belongs to class c then $\boldsymbol{L}(s,c) = 1$ and $\boldsymbol{L}(s,c') = 0$ for $c' \neq c$. By data splitting in spectral domain, we obtain $\{(\widetilde{\mathcal{X}}, \boldsymbol{L})\}$.

We distribute the training data into Q subsets, as $\{(\widetilde{\mathcal{X}}(:,:,:,q), \boldsymbol{L})\}$. We build Q independent neural networks f_q, $q = 1, 2, ..., Q$, for training on Q subsets in parallel. Specifically, $f_q(\cdot)$ takes $\widetilde{\mathcal{X}}(:,:,:,q)$ as the input and outputs $\boldsymbol{Y}^q \in \mathbb{R}^C$.

For the training process of each sub-network, its loss function is

$$\mathcal{L}^q = \Sigma_{s=1}^{m}\Sigma_{c=1}^{C}\boldsymbol{L}(s,c)\cdot \ln\left(\boldsymbol{Y}^q(s,c)\right). \tag{1}$$

Ensemble results: We ensemble the outputs of Q sub-networks $f_q(\cdot)$, $q = 1, 2, ..., Q$, and obtain the final result. Specifically, we computes

$$\boldsymbol{Y} = \Sigma_{q=1}^{Q} w_q \boldsymbol{Y}^q, \tag{2}$$

where $\boldsymbol{Y}^q = f_q(\widetilde{\mathcal{X}}^q)$ and the weights here are optionally from the following scheme,

$$w_s = \begin{cases} \frac{1}{Q}, & \text{equal weight} \\ \frac{1}{\mathcal{L}^q}, & \text{weights by loss} \\ p^{a(q)}, & \text{geometric weight} \\ \delta(a(q)=1), & \text{select the best} \end{cases}$$

where p is a user-defined parameter, $a(q)$ is the order of f_q according to the loss function value $\mathcal{L}^q$ in an ascending order, $1 \leq a(q) \leq Q$, and f_q is the best model when $a(q) = 1$. Besides, the "select-the-best" scheme retains only the best sub-network, i.e., the one with the lowest loss function value. Our experimental results indicate that the geometric weighting scheme yields the best inference performance. On the other hand, the "select-the-best" scheme achieves an additional compression, at the expense of some performance degradation.

Remark 1. *We summarize the compression ratio, the reduction of computation, and the parallel speedup in Table 1. They are theoretical upper bounds, while their actual values depend on data sets and implementations. For the compression ratio and reduction in computation, each fully connected network / convolutional network has two columns: the right one corresponds to select-the-best weighting, and the left one to other weighting methods.*

3.3 Mapping onto GPUs

The training process of each subnetwork is independent with each other and involves massive parallelism. We use the vamp function in Jax to vectorizing all the input data and subnetworks for batch computation for parallel in on GPU, while pmap is used for parallel computation in multiple GPUs.

Scheduling on one GPU: We use the vmap function to map the training process onto one GPU as follows,

- First, the dataset $\widetilde{\mathcal{X}}$ and ensemble models f are first vectorized as $\left[\widetilde{\mathcal{X}}(:,:,:,1), \widetilde{\mathcal{X}}(:,:,:,2), ..., \widetilde{\mathcal{X}}(:,:,:,Q)\right]$ and $\left[f_1, f_2, ..., f_Q\right]$, respectively.
- Then, the main process forks a total of Q threads scheduled to map $(\widetilde{\mathcal{X}}(:,:,:,q), f_q)$, $q = 1, 2, ..., Q$,onto the GPU in parallel.

- Next, Q treads compute $f_q(\widetilde{\mathcal{X}}(:,:,:,q))$ in parallel and send the result y_q to the main process, $q = 1, 2, ..., Q$. At the same time, each thread computes its own loss function value and leverages the backprogation to update the parameters of models in parallel.
- The main process merges Q results and computes the ensemble outputs as the final inference result.

Scheduling on multiple GPUs: We use the pmap function to map the training process onto multiple GPUs. The main difference between pmap and vmap is that pmap distributes the dataset $\widetilde{\mathcal{X}}$ and ensemble models f into multiple GPUs for the following computation.

It should be noted that our proposed method is communication-free for the sub-network training, which is extremely suitable for parallel training. Besides, the reduced dimension of spectral sub-dataset leads to a compressed model, which provides a further acceleration for the training process.

4 Performance Evaluation

We first describe the experimental settings, then present the results on the MNIST, CIFAR-10 and ImageNet data sets.

4.1 Data Sets and Performance Metrics

We verify the performance of the proposed spectral tensor networks on the following three widely used data sets: 1) MNIST [47] contains grayscale images of handwritten digits. Each image has 28×28 pixels. The training set has $60,000$ images and the testing set has $10,000$ images. 2) CIFAR-10 [48] contains $60,000$ color images in 10 classes, where each image has size $32 \times 32 \times 3$.3) ImageNet-1K [19]: It contains $12,000,000$ training images and $50,000$ testing images with size of $224 \times 224 \times 3$, labeled with the presence or absence of 1000 object categories that do not overlap with each other.

We are interested in the following performance metrics:1) *Compression ratio*: the ratio of the conventional network size to the spectral tensor network size, which is the also the total reduction in computation due to the reduced number of non-zero network weights; 2) *Parallel speedup*: the ratio of the training time of a conventional network to that of the spectral tensor network, due to the fully parallel training of all sub-networks; 3) *Convergence*: the loss value versus the training iterations: 4) *Accuracy*: the percentage of correctly estimated labels. Both the training and testing processes are executed on a DGX-2 server [49] that has two 64 core AMD CPUs, 8 NVIDIA A100 GPUs and 2 TB of memory. The operating system is Ubuntu 20.04 with CUDA 10.1. We use PyTorch [50] to implement neural networks.

We summarize the compression ratio, the reduction of computation, and the parallel speedup in Table 1. They are theoretical upper bounds, while their actual values depend on data sets and implementations. For the compression ratio and reduction in computation, each fully connected network / convolutional network has two columns: the right one corresponds to select-the-best weighting, and the left one to other weighting methods.

4.2 Verification on MNIST and CIFAR Data Sets

For comparison, we consider a conventional fully connected network (FC) [1], the tNN [51], and the fully connected spectral tensor network (FC-tensor) in Section **??**. All three methods use the ReLU activation function as $\sigma(\cdot)$ in the hidden layers, the softmax function as the output function $f(\cdot)$ in the last layer, and the cross-entropy loss function in (**??**). We use $N = 8$ layers in each method and the DCT transform in tNN and the proposed FC-tensor method. The learning rate was set to be 0.01, the batch size was set to be 64, and we used the Adam optimizer [52]. We split the MNIST dataset into $Q = 28$ spectral subsets and the CIFAR dataset into $Q = 32$ spectral subsets. For the combination weights, we report the results for the four weighting schemes.

For the MNIST data set, the conventional FC method has $n = 28$, $\ell'_0 = ... = \ell'_7 = 784$, and $L = 10$. Both the tNN method and our FC-tensor method have $n = 28, Q = 28$, $\ell_0 = ... = \ell_7 = 28$, and $L = 10$. Our FC-tensor method has $r = 8$. For the CIFAR-10 data set, the following parameters are different: $n = 32, Q = 32$, $\ell'_0 = ... = \ell'_7 = 1,024$, and $\ell_0 = ... = \ell_7 = 32$. Therefore, our methods achieve a compression ratio of $49\times$ and $64\times$ for the two data sets, respectively.

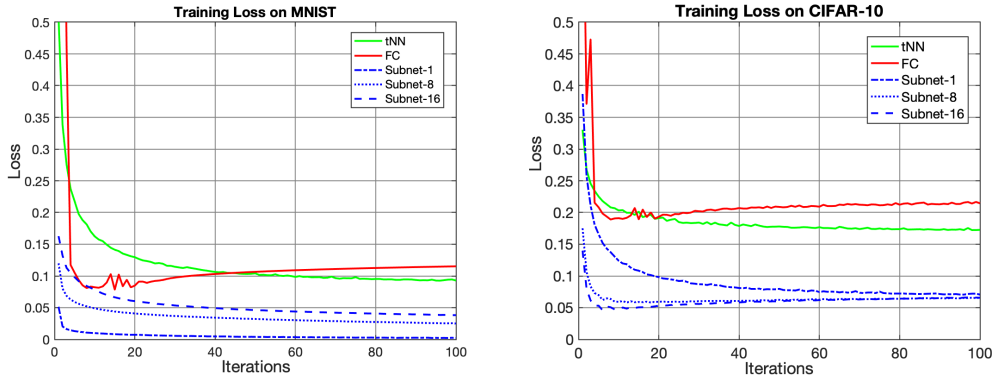

Figure 2: Training loss of fully connected networks on the MNIST data set (left) and CIFAR-10 data set (right).

The training loss over iterations is shown in Fig. 2, with the left one for the MNIST datas set and the right one for the CIFAR-10 data set. Our scheme converges faster than tNN and FC, while the training process is more stable than FC. The possible reason is that the FC-tensor has much less parameters so that a more stable model can be learned from the same amount of data samples[1]. The loss values of our sub-networks are lower than both tNN and FC.

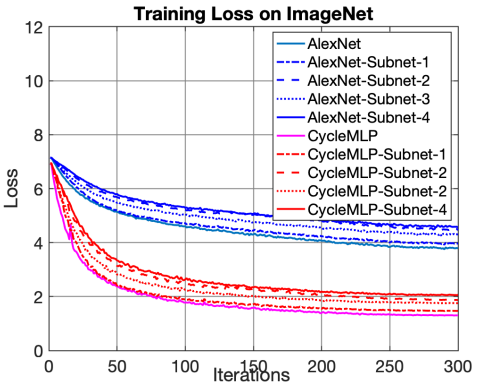

Figure 3: Training loss on ImageNet-1K data set.

Table 3: MNIST and CIAFR-10 data sets.

Methods	MNIST	CIFAR-10
FC [1]	98.71%	**59.19%**
tNN [51]	97.59%	44.50%
FC-spectral (**average**)	97.43%	47.24%
FC-spectral (**weighted sum**)	98.02%	48.13%
FC-spectral (**geometric**)	**99.01%**	48.33%

In Table 3, we report accuracy results on both MNIST and CIAFR-10 data sets. Among the four schemes for weighting the sub-networks , the geometric weighting gives the best performance. For the MNIST data set, all three methods achieve a relative high accuracy, i.e., over 97%, while our FC-tensor method reaches 98.36%. For the CIFAR-10 data set, all three methods achieve a relative low accuracy, i.e., below 60%. This is consistent with the known fact that fully connected layers are not enough for the classification task on CIFAR-10. Note that both tNN and FC-tensor achieve lower accuracy than the FC method.

Convolutional Networks

For comparison, we consider a convolutional neural network (CNN) [1], the convolutional spectral tensor network (Conv-spectral). Both methods use the ReLU funtion as the activation function, and the cross-entropy loss function. The learning rate was 0.01 and the batch size was 64. We used the Adam optimizer [52] and $N = 8$ layers. The results can be shown in Fig. 4 and Tab. 2.

The training loss and testing accuracy are shown in Fig. 4. Both methods achieve a relative high accuracy, i.e., $\geq 92\%$. Our proposed Conv-tensor method converges much faster, using less than 60 iterations to reach the same accuracy.

[1]Note that we use the same number of layers and the same batch size.

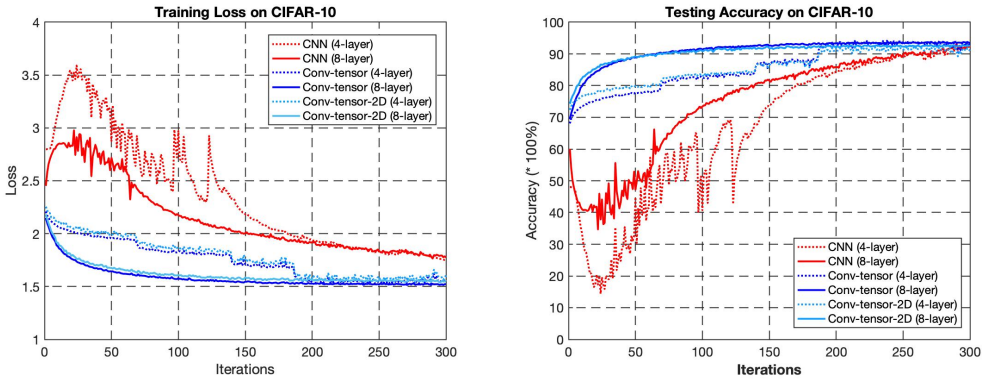

Figure 4: Training loss and testing accuracy of convolutional networks on the CIFAR-10 dataset.

Table 2: Results on the CIFAR-10 data set.

Methods	Accuracy	Size	Training Time
CNN [1]	92.07%	203 MB	6.2 h
Conv-spectral (**average**)	97.25%	125 MB	2.7 h
Conv-spectral (**weighted sum**)	98.81%	125 MB	2.7 h
Conv-spectral (**geometric**)	**99.95**%	125 MB	2.7 h
Conv-spectral (**select-the-best**)	97.25%	33 MB	2.7 h

For CNN, we assume the input and output channels to be $C = 16$. Since the input image is $32 \times 32 \times 3$ with 3 RGB channels, we divide it into $B = 4$ images with a downsampling factor $4\times$ and obtain a $16 \times 16 \times 12$ tensor. Compared with the CNN scheme, Conv-spectral achieve a compression ratio $B = B_1 B_2 = 4\times$.

4.3 Performance on ImageNet-1K and ImageNet-22K Data Sets

ImageNet-1K

The ImageNet data set [19] is split into $B = 4$ spectral subsets, where each image is organized into a tensor of size $56 \times 56 \times 3 \times 16$ and then processed into a spectrum tensor using DCT transform. Note that the three RGB channels are processed independently.

Our proposed spectral tensor methods have the same structure in Fig. **??**, where each branch is replaced by either AlexNet [18] or CycleMLP. We use the DCT transform in our spectral methods. We set the learning rate 0.01 and the batch size 128. We follow the standard practice in the community by reporting the top-1 accuracy on the testing set.

For the ImageNet data set, the training loss over training iterations is shown in Fig. 3. Our spectral sub-networks have similar loss curve to their original networks. In Table 3, we report the accuracy, model size, and training time. For the AlextNet structure, our spectral network achieves $1.88\times$ model compression and $1.28\times$ speedup in training time, at the cost of an accuracy drop of 1.18%. For the VGG-16 structure, our spectral network achieves $1.88\times$ model compression and $1.28\times$ speedup in training time, with an improvement of the accuracy of 1.37%. For the ResNet-34 structure, our spectral network achieves $1.88\times$ model compression and $1.28\times$ speedup in training time, at the cost of an accuracy drop of 1.18%. For the CycleMLP structure, our spectral network achieves $1.36\times$ model compression and $1.55\times$ speedup in training time, at the cost of an accuracy drop of 0.03%. For the MobileVit structure, our spectral network achieves $1.88\times$ model compression and $1.28\times$ speedup in training time, at the cost of an accuracy drop of 1.18%.

ImageNet-21K

The ImageNet-21K dataset is similar with ImageNet-1K dataset, but is more challenging. It has the number of 21K categories and $10\times$ total number of images than that of ImageNet-1K dataset. Because the differences of ImageNet-1K and ImageNet-22K are only the number of images and

Table 3: Results on the ImageNet-1K data set.

Methods	Accuracy	Size	Training Time
AlexNet [18]	**63.44**%	244 MB	40.8 h
AlexNet-spectral (**average**)	61.26%	130 MB	31.9 h
AlexNet-spectral (**weighted sum**)	58.01%	130 MB	31.9 h
AlexNet-spectral (**geometric**)	62.26%	130 MB	31.9 h
AlexNet-spectral (**select-the-best**)	56.45%	32 MB	31.9 h
VGG-16 [53]	**71.25**%	527 MB	81.7 h
VGG-16-spectral (**average**)	67.39%	436 MB	64.0 h
VGG-16-spectral (**weighted sum**)	69.57%	436 MB	64.0 h
VGG-16-spectral (**geometric**)	72.62%	436 MB	64.0 h
VGG-16-spectral (**select-the-best**)	68.66%	112 MB	64.0 h
ResNet-34 [54]	**74.90**%	83 MB	76.4 h
ResNet-34-spectral (**average**)	70.68%	69 MB	50.3 h
ResNet-34-spectral (**weighted sum**)	73.66%	69 MB	50.3 h
ResNet-34-spectral (**geometric**)	74.11%	69 MB	50.3 h
ResNet-34-spectral (**select-the-best**)	72.19%	15 MB	50.3 h
CycleMLP [55]	**83.23**%	103 MB	93.6 h
CycleMLP-spectral (**average**)	78.80%	76 MB	60.4 h
CycleMLP-spectral (**weighted sum**)	77.54%	76 MB	60.4 h
CycleMLP-spectral (**geometric**)	83.20%	76 MB	60.4 h
CycleMLP-spectral (**select-the-best**)	72.45%	22 MB	60.4 h
MobileVit [56]	**72.90**%	9 MB	192.4 h
MobileVit-spectral (**average**)	70.99%	7 MB	137.2 h
MobileVit-spectral (**weighted sum**)	71.22%	7 MB	137.2 h
MobileVit-spectral (**geometric**)	72.06%	7 MB	137.2 h
MobileVit-spectral (**select-the-best**)	72.10%	2 MB	137.2 h

the number of categories, we adopt the same setting for training neural networks on ImageNet-1K dataset. The result can be shown in Table 4. For the AlextNet structure, our spectral network achieves $1.85\times$ model compression and $1.34\times$ speedup in training time, at the cost of an accuracy drop of 1.07%. For the VGG-16 structure, our spectral network achieves $1.25\times$ model compression and $1.25\times$ speedup in training time. It has an improvement of the accuracy of 1.33%. For the ResNet-34 structure, our spectral network achieves $1.36\times$ model compression and $1.42\times$ speedup in training time, at the cost of an accuracy drop of 0.25%. For the CycleMLP structure, our spectral network achieves $1.64\times$ model compression and $1.52\times$ speedup in training time, at the cost of an accuracy drop of 2.11%. For the MobileVit structure, our spectral network achieves $1.44\times$ model compression and $1.31\times$ speedup in training time, at the cost of an accuracy drop of 1.44%.

5 Conclusions

In this paper, we have proposed a spectral tensor form of deep neural networks that is inherently compressive and allows communication-free parallel/distributed implementations. The data is organized into tensors and a linear transform is applied along certain dimension, resulting in different spectral subsets. The overall network consists of parallel branches of networks, each independently performs training and inference on a spectral data subset. We tested the proposed spectral networks, including fully connected, convolutional, AlexNet, VGG-16, ResNet-34, CycleMLP and MobileVit, on the MNIST, CIFAR-10 and ImageNet data sets, and results show that they can achieve relatively high accuracy with substantial network compression, computation reduction, and parallel speedup, compared with conventional networks.

For future works, we would like to explore an ensemble-style approach *model soup* [57] that takes average over multiple trained models and achieves state-of-the-art performance on the ImageNet data set.

Table 4: Results on the ImageNet-21K data set.

Methods	Accuracy	Size	Training Time
AlexNet [18]	**54.52**%	289 MB	42.9 h
AlexNet-spectral (**average**)	50.44%	156 MB	32.7 h
AlexNet-spectral (**weighted sum**)	49.54%	156 MB	32.7 h
AlexNet-spectral (**geometric**)	53.35%	156 MB	32.7 h
AlexNet-spectral (**select-the-best**)	48.99%	41 MB	32.7 h
VGG-16 [53]	**60.66**%	592 MB	82.8 h
VGG-16-spectral (**average**)	56.88%	472 MB	65.9 h
VGG-16-spectral (**weighted sum**)	58.62%	472 MB	65.9 h
VGG-16-spectral (**geometric**)	61.99%	472 MB	65.9 h
VGG-16-spectral (**select-the-best**)	59.31%	132 MB	65.9 h
ResNet-34 [54]	**63.81**%	103 MB	80.1 h
ResNet-34-spectral (**average**)	61.98%	76 MB	56.2 h
ResNet-34-spectral (**weighted sum**)	62.33%	76 MB	56.2 h
ResNet-34-spectral (**geometric**)	63.56%	76 MB	56.2 h
ResNet-34-spectral (**select-the-best**)	61.25%	19 MB	56.2 h
CycleMLP [55]	**74.56**%	135 MB	97.2 h
CycleMLP-spectral (**average**)	69.23%	82 MB	63.9 h
CycleMLP-spectral (**weighted sum**)	67.88%	82 MB	63.9 h
CycleMLP-spectral (**geometric**)	72.45%	82 MB	63.9 h
CycleMLP-spectral (**select-the-best**)	63.11%	26 MB	63.9 h
MobileVit [56]	**63.79**%	13 MB	199.5 h
MobileVit-spectral (**average**)	58.11%	9 MB	143.9 h
MobileVit-spectral (**weighted sum**)	60.67%	9 MB	143.9 h
MobileVit-spectral (**geometric**)	62.35%	9 MB	143.9 h
MobileVit-spectral (**select-the-best**)	61.56%	2 MB	143.9 h

REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*, MIT Press, 2016.

[2] Yang Liu, Peng Sun, Nickolas Wergeles, and Yi Shang, "A survey and performance evaluation of deep learning methods for small object detection," *Expert Systems with Applications*, vol. 172, pp. 114602, 2021.

[3] Xiyang Dai, Yinpeng Chen, Bin Xiao, Dongdong Chen, Mengchen Liu, Lu Yuan, and Lei Zhang, "Dynamic head: Unifying object detection heads with attentions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7373–7382.

[4] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al., "Sparse r-cnn: End-to-end object detection with learnable proposals," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14454–14463.

[5] Xiangjun Wang, Junxiao Song, Penghui Qi, Peng Peng, Zhenkun Tang, Wei Zhang, Weimin Li, Xiongjun Pi, Jujie He, Chao Gao, et al., "Scc: an efficient deep reinforcement learning agent mastering the game of starcraft ii," in *International Conference on Machine Learning*. PMLR, 2021, pp. 10905–10915.

[6] Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu, "Fop: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 12491–12500.

[7] Bo Liu, Qiang Liu, Peter Stone, Animesh Garg, Yuke Zhu, and Anima Anandkumar, "Coach-player multi-agent reinforcement learning for dynamic team composition," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6860–6870.

[8] Zhiqi Huang, Fenglin Liu, Xian Wu, Shen Ge, Helin Wang, Wei Fan, and Yuexian Zou, "Audio-oriented multimodal machine comprehension via dynamic inter-and intra-modality attention," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, vol. 35, pp. 13098–13106.

[9] Tianyang Zhao, Zhao Yan, Yunbo Cao, and Zhoujun Li, "Asking effective and diverse questions: a machine reading comprehension based framework for joint entity-relation extraction," in *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*, 2021, pp. 3948–3954.

[10] Cong Sun, Zhihao Yang, Lei Wang, Yin Zhang, Hongfei Lin, and Jian Wang, "Biomedical named entity recognition using bert in the machine reading comprehension framework," *Journal of Biomedical Informatics*, vol. 118, pp. 103799, 2021.

[11] Aditya Prakash, Kashyap Chitta, and Andreas Geiger, "Multi-modal fusion transformer for end-to-end autonomous driving," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 7077–7087.

[12] Yingfeng Cai, Tianyu Luan, Hongbo Gao, Hai Wang, Long Chen, Yicheng Li, Miguel Angel Sotelo, and Zhixiong Li, "Yolov4-5d: An effective and efficient object detector for autonomous driving," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–13, 2021.

[13] Sudeep Fadadu, Shreyash Pandey, Darshan Hegde, Yi Shi, Fang-Chieh Chou, Nemanja Djuric, and Carlos Vallespi-Gonzalez, "Multi-view fusion of sensor data for improved perception and prediction in autonomous driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2022, pp. 2349–2357.

[14] Huu-Thiet Nguyen, Chien Chern Cheah, and Kar-Ann Toh, "An analytic layer-wise deep learning framework with applications to robotics," *Automatica*, vol. 135, pp. 110007, 2022.

[15] Radouan Ait Mouha et al., "Deep learning for robotics," *Journal of Data Analysis and Information Processing*, vol. 9, no. 02, pp. 63, 2021.

[16] Yinong Chen and Gennaro De Luca, "Technologies supporting artificial intelligence and robotics application development," *Journal of Artificial Intelligence and Technology*, vol. 1, no. 1, pp. 1–8, 2021.

[17] G. Menghani, "Efficient deep learning: A survey on making deep learning models smaller, faster, and better," *arXiv preprint arXiv:2106.08962*, 2021.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012.

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, "ImageNet: A large-scale hierarchical image database," in *IEEE CVPR*. Ieee, 2009, pp. 248–255.

[20] Kai Wei, Yuzong Liu, Katrin Kirchhoff, Chris Bartels, and Jeff Bilmes, "Submodular subset selection for large-scale speech training data," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 3311–3315.

[21] Yulin He, Joshua Zhexue Huang, Hao Long, Qiang Wang, and Chenghao Wei, "I-sampling: A new block-based sampling method for large-scale dataset," in *2017 IEEE International Congress on Big Data (BigData Congress)*. IEEE, 2017, pp. 360–367.

[22] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Marc'aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, et al., "Large scale distributed deep networks," *Advances in neural information processing systems*, vol. 25, 2012.

[23] Alexander Sergeev and Mike Del Balso, "Horovod: fast and easy distributed deep learning in tensorflow," *arXiv preprint arXiv:1802.05799*, 2018.

[24] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton, "Large scale distributed neural network training through online distillation," *arXiv preprint arXiv:1804.03235*, 2018.

[25] Zhenheng Tang, Shaohuai Shi, Xiaowen Chu, Wei Wang, and Bo Li, "Communication-efficient distributed deep learning: A comprehensive survey," *arXiv preprint arXiv:2003.06307*, 2020.

[26] Nikita Ivkin, Daniel Rothchild, Enayat Ullah, Ion Stoica, Raman Arora, et al., "Communication-efficient distributed sgd with sketching," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[27] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan, "Communication-efficient distributed dual coordinate ascent," *Advances in neural information processing systems*, vol. 27, 2014.

[28] Mohammad Zalbagi Darestani, Akshay S Chaudhari, and Reinhard Heckel, "Measuring robustness in deep learning based compressive sensing," in *International Conference on Machine Learning*. PMLR, 2021, pp. 2433–2444.

[29] Guo Lu, Wanli Ouyang, Dong Xu, Xiaoyun Zhang, Zhiyong Gao, and Ming-Ting Sun, "Deep kalman filtering network for video compression artifact reduction," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[30] Alex Tseng, Avanti Shrikumar, and Anshul Kundaje, "Fourier-transform-based attribution priors improve the interpretability and stability of deep learning models for genomics," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, Eds. 2020, vol. 33, pp. 1913–1923, Curran Associates, Inc.

[31] Ron Levie, Federico Monti, Xavier Bresson, and Michael M Bronstein, "Cayleynets: Graph convolutional neural networks with complex rational spectral filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2018.

[32] Heng Chang, Yu Rong, Tingyang Xu, Wenbing Huang, Somayeh Sojoudi, Junzhou Huang, and Wenwu Zhu, "Spectral graph attention network," 2020.

[33] Jia Li, Jiajin Li, Yang Liu, Jianwei Yu, Yueting Li, and Hong Cheng, "Deconvolutional networks on graph data," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, Eds. 2021, vol. 34, pp. 21019–21030, Curran Associates, Inc.

[34] Li Yi, Hao Su, Xingwen Guo, and Leonidas J. Guibas, "Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[35] Omer Sagi and Lior Rokach, "Ensemble learning: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 4, pp. e1249, 2018.

[36] 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*, vol. 115, no. 3, pp. 211–252, 2015.

[37] Shuo Wang, Leandro L Minku, and Xin Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1356–1368, 2014.

[38] Vipin Kumar and Sonajharia Minz, "Multi-view ensemble learning: an optimal feature set partitioning for high-dimensional data classification," *Knowledge and Information Systems*, vol. 49, no. 1, pp. 1–59, 2016.

[39] Leandro L Minku and Xin Yao, "Ddd: A new ensemble approach for dealing with concept drift," *IEEE transactions on knowledge and data engineering*, vol. 24, no. 4, pp. 619–633, 2011.

[40] Robert E Schapire, "Explaining adaboost," in *Empirical inference*, pp. 37–52. Springer, 2013.

[41] Leo Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[42] Leo Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[43] Jerome H Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[44] Yushi Chen, Ying Wang, Yanfeng Gu, Xin He, Pedram Ghamisi, and Xiuping Jia, "Deep learning ensemble for hyperspectral image classification," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 12, no. 6, pp. 1882–1897, 2019.

[45] Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel, "Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 6131–6141.

[46] Qi Lin, Shuo Yu, Ke Sun, Wenhong Zhao, Osama Alfarraj, Amr Tolba, and Feng Xia, "Robust graph neural networks via ensemble learning," *Mathematics*, vol. 10, no. 8, pp. 1300, 2022.

[47] L. Deng, "The MNIST database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[48] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Master's thesis, University of Tront*, 2009.

[49] J. Choquette et al., "NVIDIA A100 tensor core GPU: Performance and innovation," *IEEE Micro*, vol. 41, no. 2, pp. 29–35, 2021.

[50] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, and et al, "PyTorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8026–8037.

[51] E. Newman, L. Horesh, H. Avron, and M. Kilmer, "Stable tensor neural networks for rapid deep learning," *arXiv preprint arXiv:1811.06569*, 2018.

[52] D.P Kingma and J. Ba, "Adam: A method for stochastic optimization," *ICLR*, 2015.

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

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[55] S. Chen, E. Xie, C. Ge, D. Liang, and P. Luo, "CycleMLP: A MLP-like architecture for dense prediction," *ICLR*, 2022.

[56] Sachin Mehta and Mohammad Rastegari, "Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer," *arXiv preprint arXiv:2110.02178*, 2021.

[57] Mitchell Wortsman, Gabriel Ilharco, and et al, "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time," *preprint arXiv:2203.05482*, 2022.