Appendix

IMPLEMENTATION

The training framework we use is Keras [1] with TensorFlow [2] as backend. This work is built upon the Keras ResNet sample [1] which includes both ResNet v1 [3] and ResNet v2 [4]. The platform of experiments is Ubuntu 16.04 with CUDA 9 and TitanX Pascal GPU. In this paper, we use ResNet v2 models with 56 layers and 110 layers.

The pipeline is based on Han's three-step pruning method [5]. The following steps are used to train and prune CLG-L1 models:

1. Firstly train the model with CLG regularization on layers that connected by element wise operations, and then prune the model;
2. Secondly retrain the model with L1 regularization on all layers, and prune the model;
3. At the end, remove regularization and fine tune the model to boost the accuracy.

We use structured pruning strategy [6] for which the minimum unit of weight removal is one kernel, which is efficient for GPU computation.

MODEL TRAINING

1. Training a CLG-L1 regularized model

CIFAR10 [7] dataset is used to train and evaluate models. The training set has 50000 samples, and the test set has 10000 samples. In the first phase of training, we use CLG regularization with regularization weight as in table 1. Assume there are ***N*** layer subsets, each has multiple layers correlated one another with element-wise operation. This procedure enforces the sparsity alignment of layers in each layer subset. Layers not belonging to any layer subset are free from regularization. The learning rate scheduler we used is in the Keras ResNet sample, which is a combination of piecewise and plateau reduction [1]. We set the initial learning rate as 0.001. The model is trained for 200 epochs, with the batch size=128.

The second training phase uses the L1 regularization on all layers. The learning rate scheduler, learning rate initialization, and batch size are as same as the first phase, but the regularization weight is set as table 1. We train the model for another 200 epochs.

In the third training phase, we remove regularization to let the cost only represents the accuracy loss. The purpose is to boost the accuracy. All other training hyper-parameters are as same as the first and the second training phase. The train lasts for 200 epochs.

2. Training a L1 regularzized model

In the first training phase we use L1 regularization on all layers. Other hyper-parameters are as same as those in CLG-L1first phase training. The regularization weight is in table 1. The model is trained for 200 epochs.

In the second phase, we remove regularization, and keep other training hyper-parameters the same. Train the model for 200 epochs.

TABLE I. REGULARIZATION WEIGHTS

Layer #	*Model*	*CLG weight*	*L1 weight*
56	L1 v2	---	1e-5
	CLG-L1_A v2	1.5e-4	3e-5
	CLG-L1_B v2	1.5e-4	7e-5
110	L1 v2	---	1e-5
	CLG-L1_A v2	1.5e-4	2.5e-5
	CLG-L1_B v2	1.5e-4	5e-5

REFERENCES

[1] F. Chollet. Keras: Deep learning library for theano and tensorflow. https://keras.io, 2016.

[2] M. Abadi, A. Agarwal, P. Barham, et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[4] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In ECCV, 2016

[5] Han, Song, Pool, Jeff, Tran, John, and Dally, William J. Learning both weights and connections for efficient neural networks. In Advances in Neural Information Processing Systems, 2015.

[6] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf. Pruning filters for efficient convnets. In International Conference on Learning Representations (ICLR), 2017

[7] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto, 2009.