Specialized DNN Extraction based on Social Network Community Detection

Duo Wang
Shanghai Jiao Tong University
and
Xiaosong Jia
Shanghai Jiao Tong University

1. INTRODUCTION

The emergence of cloud services and mobile internet have changed people's life style as well as made tons of successful corparations. With the evolution of artificial intelligence and deep learning technologies, a new interest has arised in how to make scalable deep learning models which can fit in very limited resources on mobile devices. Many work have been done on DNN compression and acceleration[3][4]. A new approach focusing on class-level oriented optimization have been proposed recently[5]. While many of the popular deep learning models have very sophisticated functionalities, studies show[7] that only a few of the functionalities are needed in mobile application scenarios. For instance, CNN models for classification tasks are generally trained to identify a vast amount of classes(like ImageNet with 1000 output classes). However only a small part of the classes may be needed for mobile applications(like written or printed characters for dictionary apps). Our work is intended to propose a new approach to extract light-weighted specialized networks from pre-trained general-purpose DNN models with the capability to function properly on mobile systems. Our method is based on Social Network Community Detecion algorithms. Community detection aims at finding subsets of nodes in a complex network where nodes inside one community should have strong connections with each other while nodes belonging to different communities should have weak connections (strong intra-connection and weak inter-connection). Deep Neural Networks are proved to have sparsity at the end For DNN based classifying models, our idea is to obtain specialized networks by finding subset of units 'strongly connected' to some of the output units. The method is intended to find the units relatively 'important' to the prediction of some of the output classes. We adapted the community detection algorithm proposed by Andrea Lancichinetti et al.(2011)[1] to work for neural networks. Experiments are conducted on MNIST[6] data set with LeNet. Our community detection based algorithm is proved to compress the network by 0.265% on LeNet-300-100 and 2.28% on LeNet-5 averagely, with a less than 1% accuracy loss.

2. RELATED WORK

Storage and memory cost has long been Many works have been done on Deep Neural Network compression and accelerating. Song Han et al.(2016)[3] proposed a combined approach of pruning, quantization and huffman coding which managed to compress LeNet-5 by 39 times, AlexNet by 35 times and VGG-16 by 49 times without accuracy loss in the experiment they conducted. A recent work has brought forth a class adaptive pruning framework named CAPTOR[5]. The authors discussed the prosperity of launching light-weighted specialized DNN models on mobile devices. They have pointed out that most mobile apps have a very focused subset of typical classification targets when using classifying CNN models for image recognition, indicating that instead of launching the versatile powerful yet heavy-weighted DNN models trained on huge datasets with over 1000 classification targets, launching specialized models extracted from the original network which only focus on a subset of the classification targets will be much more efficient and barely damage the performance. Our work is based on the idea of extracting specialized sub-network from a pretrained versatile DNN model for mobile applications.

3. PROBLEM FORMULATION

Our algorithm is expected to extract specialized sub-network from a pre-trained DNN model with some of the functionalities from the original network which is much smaller than the original one and can function properly under some specific application domain. Precisely speaking, given a pre-trained CNN based classifier with multiple output targets, our algorithm should be able to extract a sub-network to make predictions on one of the output classes with a tolerable precision loss and a considerable compress rate. The desired precision loss and compersss rate are both 1%.

4. PROPOSED METHODS

We adaptted the community detection algorithm proposed by Andrea Lancichinetti et al.[1] named OSLOM.

OSLOM Basics

Basically, OSLOM constructs a community by randomly selecting a node as the original community and repeatedly including nodes with 'strong connection' with the current community into the community. The criteria of 'strong connection' for weighted networks defined by OSLOM is based on statistic significance of the weight between a pair of nodes. The expected weight between any two nodes is:

$$\tau < w_{ij} >= \frac{2 < w_i >< w_j >}{< w_i > + < w_j >} \tag{1}$$

where $< w_i >= \frac{\sum_{w \in W_i} w}{|W_i|}$ reflects the average weight of all the edges connected to node i.

We can easily adapt the above equation to reflect the expected connectiveness between a single node and a community:

$$< w_{Cj} >= \frac{2 < w_C >< w_j >}{< w_C > + < w_j >} \tag{2}$$

where $< w_c >= \frac{\sum_{w \in W_C} w}{|W_C|}$ reflects the average weight of all the edges connected to community C.

The statistic significance of node j in regard to community C is therefore:

$$r_j(C) = p(w_{Cj} > x) = exp(-x/ < w_{Cj} >) \quad (3)$$

where x is the actual weight between j and C.

The next step involves selecting all the nodes with significant connection to community C. OSLOM employs Bonferroni correction method to model the selection procedure by calculating:

$$\Omega_q(r) = p(r_q < x) = \sum_{i=q}^{N-n_C} \binom{N-n_C}{i} r^i (1-r)^{N-n_C-i} \quad (4)$$

$\{r_j\}$ is viewed as a uniform random variable distributed on [0,1], and r_q is the q-st smallest value of $\{r_j\}$.

The algorithm selects the smallest possible value of q where $\Omega_q < t, \Omega_{q+1} \geq t$ and includes the nodes corresponding to $r_1, r_2, , r_q$.

Weight Representation

Since Network Community Detection algorithms require positive weights on edges, we desiged a weight representation method, where:

(1) $w' = |w|$ for numerical weights(fully-connected linear layers)

(2) $w' = mean(abs(w))$ for convolution kernels

Specialized DNN Extraction

Algorithm 1: Specialized DNN Extraction

Data: Weights[m][n],t
Result: selected[q]

```
Weights←mean(abs(Weights))
R[m]← exp(−(m/2)(Σ_{i∈C} W_ij / Σ_{i∈C,j∈J} W_ij) − 1/2)
Rq[m]←sorted(R) Omega[m]← Σ_{i=q}^{m} C(m,i) r^i (1−r)^{m−i}
for q=0;q¡m-1;q++ do
    if Omega[q]¡t,Omega[q+1]≥t then
        select units 1,2,...,q and return
    end
end
return all units
```

We therefore propose our community detection based specialized DNN extraction algorithm(Algorithm.1).

In order to obtain sub-network with specialized functionalities, we perform community detection starting from one of the output units. Since neural networks are fully-connected level graphs, the only reasonable approach is to run community detection in a layer-by-layer manner. Also, because of the special structure of neural networks, the criteria of strong connection can be rewritten as:

(1) $r_j(C) = exp(-\frac{m}{2}\frac{\sum_{i\in C} W_{ij}}{\sum_{i\in C, j\in J} W_{ij}} - \frac{1}{2})$, where $\{W_{ij}\}$ is the weight matrix, m is the number of units in the current layer, C is all the units included in the last layer and J is all the units in the current layer.

(2) $\Omega_q(r) = p(r_q < r) = \sum_{i=q}^{m} \binom{m}{i} r^i (1-r)^{m-i}$

5. EXPERIMENTS

We conducted our experiment on MNIST[6] dataset using LetNet-300-100 and LeNet-5. Table.I&II shows the results, with the compress rate and test accuracy listed for each model we have extracted. We managed to extract specialized models with an average compress rate of 0.265% for LeNet-300-100 and 2.51% for LeNet-5 in regard of the number of parameters.

REFERENCES

Lancichinetti A, Radicchi F, Ramasco JJ, Fortunato S (2011) *Finding Statistically Significant Communities in Networks*. *PLoS ONE*,6(4):e18961.

John Palowitch, Shankar Bhamidi, Andrew B. Nobel (2017) *Significance-based community detection in weighted networks*. *Journal of Machine Learning Research*18(2018):1-48.

Song Han, Huizi Mao, William J. Dally (2016) *Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding*. *arXiv*:1510.00149 [cs.CV].

Max Jaderberg, Andrea Vedaldi, Andrew Zisserman (2014) *Speeding up convolutional neural networks with low rank expansions*. *arXiv*:1405.3866 [cs.CV].

CAPTOR: A Class Adaptive Filter Pruning Framework for Convolutional Neural Networks in Mobile Applications

Y. LeCun, L. Bottou, Y. Bengio, P. Haffner (1998) *Gradient-based learning applied to document recognition*. *Proceedings of the IEEE*, 86(11):2278-2324.

Best image recongnition apps for android(Top 100)-AppCrawlr. *https://appcrawlr.com/android-apps/best-apps-image-recognition*

Table I. LeNet-300-100 on MNIST

	0-spec	1-spec	2-spec	3-spec	4-spec	5-spec	6-spec	7-spec	8-spec	9-spec	spec avg	original
Compress Rate	0.5015%	0.0770%	0.2014%	0.1262%	0.3388%	0.2618%	0.1349%	0.4343%	0.1315%	0.4467%	0.26541%	100%
Accuracy	98.9%	98.66%	98.47%	97.23%	99.01%	98.46%	97.29%	98.18%	96.88%	98.70%	98.12%	98.28%

This is an example of table footnote. This is an example of table footnote. This is an example of table footnote. This is an example of table footnote. This is an example of table footnote.

Table II. LeNet-5 on MNIST

	0-spec	1-spec	2-spec	3-spec	4-spec	5-spec	6-spec	7-spec	8-spec	9-spec	spec avg	original
Compress Rate	0.7689%	2.2764%	2.5366%	2.6829%	1.2892%	2.1580%	4.0790%	2.2787%	2.5668%	4.5134%	2.515%	100%
Accuracy	99.21%	99.79%	99.37%	99.63%	99.34%	98.53%	99.73%	99.56%	99.56%	99.05%	99.377	99.41%

This is an example of table footnote. This is an example of table footnote. This is an example of table footnote. This is an example of table footnote. This is an example of table footnote.