Light Multi-Segment Activation for Model Compression

Zhenhui Xu,[1*] Guolin Ke,[2*] Jia Zhang,[2] Jiang Bian,[2] Tie-Yan Liu[2]
[1]Peking University, [2]Microsoft Research
zhenhui.xu@pku.edu.cn, {guolin.ke, jia.zhang, jiang.bian, tie-yan.liu}@microsoft.com

Abstract

Model compression has become necessary when applying neural networks (NN) into many real application tasks that can accept slightly-reduced model accuracy but with strict tolerance to model complexity. Recently, Knowledge Distillation, which distills the knowledge from well-trained and highly complex teacher model into a compact student model, has been widely used for model compression. However, under the strict requirement on the resource cost, it is quite challenging to make student model achieve comparable performance with the teacher one, essentially due to the drastically-reduced expressiveness ability of the compact student model. Inspired by the nature of the expressiveness ability in NN, we propose to use multi-segment activation, which can significantly improve the expressiveness ability with very little cost, in the compact student model. Specifically, we propose a highly efficient multi-segment activation, called Light Multi-segment Activation (LMA), which can rapidly produce multiple linear regions with very few parameters by leveraging the statistical information. With using LMA, the compact student model is capable of achieving much better performance effectively and efficiently, than the ReLU-equipped one with same model complexity. Furthermore, the proposed method is compatible with other model compression techniques, such as quantization, which means they can be used jointly for better compression performance. Experiments on state-of-the-art NN architectures over the real-world tasks demonstrate the effectiveness and extensibility of the LMA.

Introduction

Neural Network (NN) has become a widely-used model in many real-world tasks, such as image classification, translation, speech recognition, etc. In the meantime, the increasing size and complexity of the advanced NN models have raised a critical challenge (Wang et al. 2018) in applying them into many real application tasks, which can accept appropriate performance drop with very extremely-limited tolerance to high model complexity. Running NN models on mobile devices and embedded systems are emerging examples that make every effort to avoid expensive computation and storage cost but can endure slightly-reduced model[1] accuracy.

Consequently, many studies have been paying attention to producing compact and fast NN models with maintaining acceptable model performance. In detail, there are two active directions investigated model compression through pruning (LeCun, Denker, and Solla 1990; Hassibi and Stork 1993; Han et al. 2015; Li et al. 2016; Frankle and Carbin 2019) or quantizing (Courbariaux, Bengio, and David 2015; Rastegari et al. 2016; Mellempudi et al. 2017) the trained large NN models into squeezed ones with trimmed redundancy but preserved accuracy. More recently, increasing efforts explored Knowledge Distillation (Hinton, Vinyals, and Dean 2015) to obtain compact NN models by training them with the supervision from well-trained larger NN models (Polino, Pascanu, and Alistarh 2018; Wang et al. 2018; Mishra and Marr 2018; Hubara et al. 2017; Luo et al. 2016; Wu et al. 2016; Zhu et al. 2016; Sau and Balasubramanian 2016). Compared with directly training a compressed model from scratch merely using the ground truth, the supervision in terms of soft distributed representations on the output layer of the large teacher model can even significantly enhance the effectiveness of the resulting compact student model. In practice, nevertheless, it is quite difficult to produce the compressed student model that can yield similar effectiveness to the complex teacher model, essential due to the limited expressiveness ability of the compressed one in terms of the strictly-restricted parameter size.

Intuitively, to enhance the power of the compressed model, it is necessary to increase its expressiveness ability. However, traditional approaches to introduce more layers or hidden units into the model can easily violate the strict restrictions on the model size. Fortunately, besides the neuron number, the nonlinear transformation, in terms of the activation, within the NN model plays an equally-important role in reflecting the expressiveness ability. As pointed out by (Montufar et al. 2014), an NN model that uses multi-layer ReLU or other piecewise linear activation as the activation is still essentially a complex piecewise linear function. Moreover, the number of linear regions produced by an NN model depends on not only its model size but also its activations.

[*]Equal Contribution. This work was done while the first author was visiting Microsoft Research.

[1]Unless otherwise stated, the term "model" used in this paper refers to the Neural Network model.

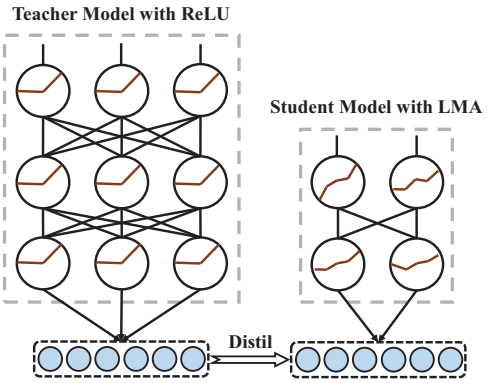

Figure 1: Depiction of knowledge distillation with LMA.

The more segments in piecewise linear activation, the more regions NN will produce, but there are only two segments in widely-used ReLU. Inspired by the few costs in adding more regions in the activation, it is more reasonable to improve the expressiveness ability of NN via multi-segment activations, instead of increasing the model size, either depth or width.

Thus, in this paper, we introduce a novel highly efficient piecewise linear activation, in order to improve the expressiveness ability of the compressed models with little cost. In detail, as shown in Fig. 1, we leverage a generic knowledge distillation framework for model compression, in which, however, the compact student model is equipped with proposed multi-segment piecewise linear activations, named Light Multi-segment Activation (LMA). In LMA, we first cut the input range into multiple segments based on batch statistics information, and ensure it can adapt to any range of input lightly and efficiently. Then, we assign each input with customized slope and bias depending on the segment it belongs to, which thus empower NN models higher expressiveness ability due to the stronger non-linearity of the new activation. Owing to the above design, LMA-equipped compact student models yield two advantages: 1) It has much higher expressiveness ability, compared with one merely endowed with vanilla ReLU; 2) Its resource cost is still much smaller and even more controllable compared with the other type of multi-segment piecewise linear activation.

Extensive experiments of multi-size NN architectures on various real tasks, including image classification and machine translation, have demonstrated both the effectiveness and the efficiency of LMA, which implies the improved expressiveness ability, thus the performance, of the LMA-equipped model. Additional experiments further illustrate that our method can also improve the expressiveness ability of the models that have been compressed by some other popular techniques, such as quantization, so jointly using the others and ours can achieve even better compression results.

The main contributions of this paper are multi-fold:

- To the best of our knowledge, it is the first work that leverages multi-segment piecewise linear function as activation in model compression. It proposes a novel multi-segment activation, which improves the expressiveness ability of the compressed student model within the knowledge distillation framework.
- With using statistical information of each batch, the proposed activation can efficiently improve the performance of compressed models with preserving low resource cost.
- The proposed method is compatible with the other popular compression techniques, such that it is easy to combine them and further get better compression results.
- On various real challenging tasks, experimental results of multiple models with different sizes show our methods have good performance. And the effectiveness of joint usage, that combines our method with the others, is also shown in the experiments.

Related work

This work is mainly related to two research areas, model compression and piecewise linear activation. The representative work of the former includes model pruning, quantization and distillation, while the latter typically studies respective effects of ReLU, Maxout and APLU on the performance of NN.

Model Compression

In this area, (LeCun, Denker, and Solla 1990; Hassibi and Stork 1993) first explored pruning based on second derivations. More recently, (Han et al. 2015; 2016; Jin et al. 2016; Hu et al. 2016; Yang, Chen, and Sze 2017) pruned the weights of Neural Networks with different strategies and made some progress. Most recently, (Frankle and Carbin 2019) showed a dense Neural Network contains a sparse trainable subnetwork such that it can match the performance of the original network, named as the lottery ticket hypothesis. On the other hand, (Gupta et al. 2015) have done a comprehensive study on the effect of low precision fixed point computation for deep learning. Therefore, quantization is also an active research area, where various methods were proposed by many works (Mellempudi et al. 2017; Hubara et al. 2017; Mellempudi et al. 2017; Zhu et al. 2016; Rastegari et al. 2016; Wu et al. 2016).

Besides, using distillation for size reduction is mentioned by (Hinton, Vinyals, and Dean 2015), which gives a new direction for training compact student models. The weighted average of soft distributed representation from the teacher's output and ground truth is much useful when training a model, so some practices (Wang et al. 2018; Luo et al. 2016; Sau and Balasubramanian 2016) have been put for training compressed compact model. Moreover, recent works also proposed to combine the quantization with distillation, producing better compression results. Among these, (Mishra and Marr 2018) used knowledge distillation for low-precision models, which proposes distillation can also help training the quantized model. (Polino, Pascanu, and Alistarh 2018) proposed a more in-depth combination of these two methods, named Quantized Distillation. Besides, there are also some works (Han, Mao, and Dally 2015; Iandola et al. 2016; Wen et al. 2016; Gysel, Motamedi, and Ghiasi 2016; Mishra et al. 2017) further reduced the model size by combining multiple compression techniques like quantization,

weight sharing and weight coding. Similarly, the combination of our method with the other is also shown in this paper.

Piecewise Linear Activation

A piecewise linear function is composed of multiple linear segments. Some piecewise functions are continuous when the boundary value calculated by two adjacent intervals function is the same, whereas some may not be continuous. Benefit from its simplicity and the fitting ability to any function with enough segments, it is widely-used in machine learning models (Landwehr, Hall, and Frank 2005; Malash and El-Khaiary 2010), especially as activations in Neural Networks (LeCun, Bengio, and Hinton 2015). Theoretically, (Montufar et al. 2014; Pascanu, Montufar, and Bengio 2013) studied the number of linear regions in Neural Networks produced by piecewise linear activation functions (PLA), which can be used to measure the expressiveness ability of the networks.

Specifically, as a two-segment PLA, Rectified Linear Unit (ReLU) (Nair and Hinton 2010) and its parametric variants can be generally defined as $h_i(x) = \min(0, a_i x) + \max(0, x)$, where x is the input, a_i is a linear slope, and $h_i(x)$ is the activated output. For original ReLU, it fixes a_i to zero so the formula degenerates to $h_i(x) = \max(0, x)$; Parametric ReLU (PReLU) (He et al. 2015) makes a_i learnable and initializes it to 0.25. Besides, there are also some PLAs with multiple segments improved from ReLU. For example, Maxout (Goodfellow et al. 2013) is a typical multi-segment PLA, which is defined as $h_i(x) = \max(z_{ij})$ for all $j \in [1, k]$, where k can be treated as its segment number, and it transforms the input into the maximum of k-fold linear transformed candidates z_{ij}; Adaptive Piecewise Linear Units (APLU) (Agostinelli et al. 2014) is also a multi-segment one, which is defined as a sum of hinge-shaped functions,

$$h_i(x) = \max(0, x) + \sum\nolimits_{j=1}^{k} a_i^j \max(0, -x + b_i^j) \ , \quad (1)$$

where k is a hyper-parameter set in advance, while the variables a_i^j, b_i^j for $j \in \{1, ..., k\}$ are learnable. The a_i^j control the slope of the linear segments while the b_i^j determine the locations of the hinges similar to segments.

In this paper, after studying the connection of above two areas, we are the first to leverage the properties of PLA for model compression, that to improve the expressiveness ability of compact model via multi-segment activation, thereby improving its performance.

Methodology

We start by studying the connection between PLA and the expressiveness ability of Neural Networks, followed by introducing the Light Multi-segment Activation (LMA) that is used to further improve the performance of the compact model in model compression.

Preliminaries

Expressiveness Ability Study Practically, increasing complexity of the neural networks, in terms of either width (Zagoruyko and Komodakis 2016) or depth (He et al. 2016), can result in swelling performance, essentially due to the higher expressiveness ability of the NN. However, when applying the NN into some resource-exhausted environments, its size cannot be inflated without limit. Fortunately, the non-linear transformation within the NN, in terms of the activation, provides another vital channel to enhance the expressiveness ability. Yet, the widely-used ReLU in NN is just a simple PLA with only two segments, where the slope on the positive segment is fixed to one while the other is zero. Therefore, other than enlarging the size of the NN model, another effective alternative method to enhance the expressiveness ability of the NN model is to leverage more powerful activation functions. In this paper, we propose to increase the segment number in activation to enhance its expressiveness ability, and further empower the compact NN to yield good performance.

Theoretically, there are also some related analysis (Montufar et al. 2014) that can justify our motivation. As pointed out by them, the capacity, i.e. the expressiveness ability, of a PLA-activated Neural Network can be measured by the number of linear regions of this model. And for an NN, in the l-th hidden layer with n_l units, the number of separate input-space neighbourhoods that are mapped to a common neighborhood $R \subseteq S_l \subseteq \mathbb{R}^{n_l}$ can be decided recursively as

$$\mathcal{N}_R^l = \sum\nolimits_{R' \in P_R^l} \mathcal{N}_{R'}^{l-1}, \ \mathcal{N}_R^0 = 1, \quad \text{for each } R \subseteq \mathbb{R}^{n_0}, \ (2)$$

where S_l denotes the set of (vector valued) activations reachable by the l-th layer for all possible input; P_R^l denotes the set of subsets $\bar{R}_1, ..., \bar{R}_k \subseteq S_{l-1}$ that are mapped by the activation onto R. Based on the above result, the following lemma (see (Montufar et al. 2014); Lemma 2) is given.

Lemma 1 *The maximal number of linear regions of the functions computed by an L-layer Neural Network with piecewise linear activations is at least* $\mathcal{N} = \sum_{R \in P^L} \mathcal{N}_R^{L-1}$ *, where* $\mathcal{N}_R^{L-1}$ *is defined by Eqn. (2), and* P^L *is a set of neighborhoods in distinct linear regions of the function computed by the last hidden layer.*

Given the above lemma, the number of linear regions of a Neural Network is in effect influenced by the layer number, the hidden unit size, and the region number in PLA. From ReLU to Maxout, the significant improvement is on the P^l in the lemma, which is also the nature of our approach. Taking Maxout as an example of detailed analysis, it can lead to an important corollary that a Maxout network with L layers of width n and rank k can compute functions with at least $k^{L-1}k^n$ linear regions (see Montufar et al. (2014); Theorem 8). Meanwhile, ReLU can be treated as a special rank-2 case of Maxout, whose bound is obtained similarly by (Pascanu, Montufar, and Bengio 2013). Obviously, the number of linear regions can be improved by increasing either L, n or k. However, in a compressed model, neither the layers L nor hidden units n can be increased too much. Thus, we propose to construct a highly efficient multi-segment activation function with its linear regions k becomes larger.

Analysis on Existing Multi-segment PLAs As mentioned in Related Work, some previous studies have already

proposed some multi-segment PLAs. In the following of this subsection, we will analyze whether they are suitable for being applied in model compression.

Considering Maxout first, its regions are produced by k-fold weights and only the maximum of its k-fold outputs is picked to feed forward, which obviously causes the redundancy within Maxout. On the contrary, to construct a PLA with multiple segments and ensure limited parameters increment in the meantime, a more intuitive inspiration, from the definition of piecewise linear function, lies in that it first cuts the input range into multiple segments, and then transforms the input linearly by individual coefficients (i.e. slopes and biases) on different segments. In this way, the parameter number of the network based on this scheme can be controlled as $L * (k + n^2)$, compared with $L * kn^2$ in the above assumed Maxout NN.

In fact, APLU is a hinge-based implementation of this scheme, with few additive parameters. Specifically, in Eqn. 1, $\boldsymbol{b}$ are the cut points of the input range, and $\boldsymbol{a}$ can be grouped accumulatively into coefficients. However, APLU can increase the memory cost due to its accumulation operation. In details, APLU requires k times intermediate variables to compute the items parallel and then accumulates all of them one-time. Although we also accumulate them recursively to avoid this, it will be k times slower and is unacceptable. Besides, with the k becomes larger, the memory cost will growth linearly.

In a word, neither Maxout nor APLU can be directly employed for model compression in that Maxout produces much more parameters and APLU is memory-consuming. In the following subsection, we will introduce a new activation process that is both effective and efficient for model compression.

Light Multi-segment Activation

Method LMA mainly contains two steps. The first is batch segmentation, which is proposed to find the segment cut-points based on the batch statistical information. Then the inputs are transformed with the corresponding linear slopes and biases according to their belonging segments.

Firstly, to construct a multi-segment piecewise activation, it needs to cut the continuous inputs to multiple segments. There are two straightforward solutions: 1) pre-defined like the vanilla ReLU; 2) training cut-points like APLU. For the former, as the input ranges of hidden layers are dramatically changed during training, it is hard to define the appropriate cut points in advance. For the latter, the cut points are unstable due to the random initialization and stochastic update by back-propagation. As the naive solutions cannot work well, inspired by the success of Batch Normalization (Ioffe and Szegedy 2015), we propose Batch Segmentation, to determine the segment boundaries by the statistical information.

There are two statistical schemes (Dougherty, Kohavi, and Sahami 1995) to find appropriate segments. One is based on frequency, and the other one is based on numerical values. Concretely, after using the frequency-based method, each segment has the same number of inputs, while if using numerical value-based one, the numerical width of each segment is equal. Indeed, the frequency-based method is more robust since it is not sensitive to numerical values. However, it is not efficient, especially running on GPU and applied for model compression. Thus, the numerical value-based solution is used in LMA for efficiency purpose. Specifically, we assume the input is a normal distribution and cut the segments by equal value width. So, here each segment cut-point is defined as,

Table 1: Cost comparison between multi-segment activation functions.

	Maxout	APLU	LMA
Param. Size	$O(k * n^2)$	$O(k * n + n^2)$	$O(k + n^2)$
Mem. Cost	$O(k * n)$	$O(k * n)$	$O(n)$

$$b_0 = \mu - 3\sigma, \qquad b_j = b_{j-1} + \frac{6\sigma}{k}, \text{ for } j = 1, 2, \ldots, k \ , \quad (3)$$

where k is the segment number, a hyper-parameter, μ and σ are the mean and standard deviation of the batch input $\boldsymbol{x}$, respectively. To reduce the effect of outliers and make use of the property of normal distribution, we assume $\mu \pm 3\sigma$ are the range endpoints and assign cut points according to this assumption. Like Batch Normalization, the moving average of $\boldsymbol{b}$ is used in the test phase. To further improve the efficiency, as well as more stable statistical information, the $\boldsymbol{b}$ could be calculated and shared in the same layer.

After determining segment boundaries, it needs to assign the coefficient, i.e. slope and bias, to each input according to its belonging segment. To avoid the memory-consuming problem in APLU, we use the independent slopes and biases in LMA. Formally, the activation process can be defined as,

$$h_i(x) = \alpha_j^i \cdot x + \beta_j^i, \quad x \in (b_j, b_{j+1}] \quad (4)$$

where $\boldsymbol{\alpha}$ denotes the slope coefficient, $\boldsymbol{\beta}$ denotes the bias, and j denotes segment indices. Especially, considering there still may be few extreme inputs out of the normal distribution assumption, the first and last segment are set to $(-\infty, b_1]$ and $(b_{k-1}, +\infty)$ respectively, instead of determining by b_0 and b_k. Finally, after the above steps, the linear transformed values $h_i(x)$ feed-forward to the next layer.

Analysis and Discussion In the following, we will take more detailed discussions on LMA from the perspective of complexity analysis and initialization. Obviously, in LMA, there is only two additional trainable variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ for each layer, whose total size is $2 * k * n$, where k is segment number and n is hidden unit number. Furthermore, to reduce the parameter size extremely, the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are shared in the layer-level, which means that all the units or feature maps are activated by the same LMA in one specific layer. Therefore, the parameters brought by LMA in one layer is only $2 * k$, even reduced by n times compared with APLU. Moreover, about the running memory cost in inference phase, LMA only produces the belonging segment indices for inputs, whose space cost is $O(n)$, while APLU needs $O(k * n)$ hinges and Maxout needs $O(k * n)$ activation candidates. To conclude, the cost comparisons between each multi-segment

PLAs are shown in Table 1, where the parameter size and the running space cost at activation in one layer are listed. It shows LMA is more suitable for model compression because of its less storage and running space cost.

Besides, the slopes and biases on all segments need to be initialized in LMA. The initialization methods always can be categorized into two classes: 1) random initialization like the other parameters in NN; 2) initializing it as a known activation, such as vanilla Relu or PReLU. Though the random initialization does not impose any assumptions and may achieve a better performance (Mishkin and Matas 2015), it usually introduces uncertainty and leads to unstable training too. With this in mind, we choose the second initialization method for LMA. Specifically, we initialize the LMA to be the vanilla ReLU, which means that all biases are initialized to zero, the slopes of the half left segments are initialized to zero while the rest slopes are initialized to one.

Model Compression As an effective method to improve the expressiveness ability of the compressed model, LMA can be applied with distillation and other compression techniques. Under the distillation framework, we first train a state-of-the-art model and get as much good performance. Then given it as the teacher model, a more compact architecture is employed to as the student to learn the knowledge from the teacher. Because of the parameter reduction in the student, it always underperforms much lower than the teacher despite using knowledge distillation. Here, we replace all the original ReLUs with our LMA for the student model, improving its expressiveness ability, and further improving the performance much. The replacement is very convenient that it only needs to change one line of code in the implementation. After that, according to (Hinton, Vinyals, and Dean 2015; Polino, Pascanu, and Alistarh 2018), the distillation loss for training the student is also a normal weighted average of the loss from ground truth and the one from teacher's output, which is formally defined as,

$$\mathcal{L} = (1-\alpha)\mathcal{L}_{CE/NLL}(\hat{y}(\boldsymbol{x}),\, y_{GT}) + \alpha\mathcal{L}_{KL}(\hat{y}(\boldsymbol{x}),\, y_T)\,, \quad (5)$$

where α is a hyper-parametric factor, which is always set to 0.7, to adjust the weight of two losses; $\hat{y}(\boldsymbol{x})$ is the student's output logits; the first loss $\mathcal{L}_{CE/NLL}$ is a Cross Entropy Loss or Negative Log Likelihood Loss with the ground truth labels y_{GT}, depending on the tasks (CE is for image classification and NNL is for machine translation in our experiments); the latter loss $\mathcal{L}_{KL}$ is a Kullback–Leibler Divergence Loss with the teacher's output logits y_T. Additionally, when calculating $\mathcal{L}_{KL}$, we also use a temperature factor τ to soften the y_T and the $\hat{y}$, whose specific settings will be shown in the experiments.

Besides, LMA is well compatible with the other compression techniques, since it is convenient to replace the activations from ReLU with LMA. For example, based on a recent representative method, Quantized Distillation (Polino, Pascanu, and Alistarh 2018), after replacing the ReLU with LMA in student model, though it is quantified to low-precision model during training, our method still empowers it to achieve higher performance than origin one, which will be shown in the experiments.

Experiment

In this section, we will conduct thorough evaluations on the effectiveness of LMA for model compression under two popular scenarios, image classification and machine translation. Besides, under the model compression scenarios, we will compare the performance of LMA with that of several widely-used baseline activations.[2] The baselines adopted includes ReLU(Nair and Hinton 2010), PReLU (He et al. 2016), APLU (Agostinelli et al. 2014) and Swish (Ramachandran, Zoph, and Le 2017). For Swish is a well-known SOTA ReLU-like activation, we also adopt it to further show our effectiveness. Specifically, we will start with our experimental setup, including the data and models employed in the experiments. After that, we will analyze the performance of our method applied singly or jointly with some others to demonstrate its effectiveness and advantages for model compression.

General Settings To ensure credible results, we run all the experiments 5 times with different random seeds, and report the average and standard deviation of them. Besides, for fair comparisons, we set all the common parameters, including learning rate, batch size, hyper-parameters in distillation loss, etc., the same for all the baselines. Note that, the settings for parametric baseline activations (PReLU, APLU and Swish), are all consistent with the original authors' demonstration. For multi-segment activations (APLU and LMA), the segment numbers are set as the same to each other, which is 8 in our main experiments. Moreover, to measure the resource cost by the models, we report their parameter size and inference memory cost (*Mem.*), in which the latter is recorded when predicting the testing samples one by one. The model size hardly changes after replacing the activation function, since the additional parameters in all these activations are relatively very few. However, for another activation, Maxout, it will yield much more parameters if replaced. Due to the poor performance compared with other baselines under the setting of the same model size, we only report one little result (see Table 2).

Image Classification

Settings Following what (Polino, Pascanu, and Alistarh 2018) does in its code[3], we first evaluate our method on CIFAR-10 and CIFAR-100, both of which are well-known image classification datasets. For experiments on CIFAR-10, some relatively small CNN architectures are employed, including one teacher model and three student models with different sizes. Widen Residual Networks (WRN) (Zagoruyko and Komodakis 2016) are employed for experiments on CIFAR-100, where WRN-16 is used as teacher model while two WRN-10 are used as students. In the first phase, we train the teacher models and save them for the next distilled training. Then, we compare the performance of the student models with different activations under the supervision from both the teacher models and the ground truth. Accuracy (*Acc.*) is used as the evaluation metric on this task.

[2] We released the code at: https://github.com/motefly/LMA

[3] https://github.com/antspy/quantized_distillation

Table 2: Image Classification Results. The metrics of Teacher models on each dataset are shown in the left-most cells. The accuracy (%) is shown in "*mean* ± *std* " pattern and the inference memory cost (in MB) is shown in "*A (+D)*" pattern, where *A* denotes absolute memory cost and *D* is additional part compared with ReLU-equipped model. The last column shows the improvement of LMA (comparing performance with ReLU and memory with APLU). The results show the PLA with more segments (APLU and LMA) outperforms the fewer ones, especially widely-used ReLU, meanwhile LMA maintains much lower memory cost than APLU. *For a **Maxout-8**-based Student Model, whose size is the same as Student 1, only get 87.76±0.55% accuracy, even lower than ReLU-based one.

Method			ReLU	PReLU	Swish	APLU-8	LMA-8	
CIFAR-10 21.4 MB Acc. 92.83 Mem. 29.28	Student 1 4.04 MB	Acc.	88.74 ±0.25	89.31 ±0.35	89.03 ±0.11	89.92 ±0.21	**90.57 ±0.20**	2.1% ↑
		Mem.	14.24	15.95 (+1.7)	15.10 (+0.9)	25.80 (+11.6)	16.81 (+2.6)	78% ↓
	Student 2 1.28 MB	Acc.	82.67 ±0.46	84.35 ±0.37	84.06 ±0.36	85.31 ±0.60	**85.66 ±0.34**	3.6% ↑
		Mem.	3.40	4.72 (+1.3)	4.06 (+0.7)	11.69 (+8.3)	5.57 (+2.2)	73% ↓
	Student 3 0.44 MB	Acc.	73.33 ±0.79	75.30 ±0.17	75.45 ±0.34	77.54 ±0.97	**77.66 ±0.47**	5.9% ↑
		Mem.	1.45	2.07 (+0.6)	1.76 (+0.3)	5.15 (+3.7)	2.55 (+1.1)	70% ↓
CIFAR-100 68.7 MB Acc. 77.56 Mem. 140.2	Student 1 4.88 MB	Acc.	69.11 ±0.80	70.03 ±0.21	69.67 ±0.40	**70.99 ±0.42**	70.92 ±0.42	2.6% ↑
		Mem.	16.27	16.40 (+0.13)	16.33 (+0.06)	17.03 (+0.76)	16.46 (+0.19)	75% ↓
	Student 2 1.28 MB	Acc.	63.12 ±1.00	64.52 ±0.67	63.82 ±0.78	66.28 ±0.49	**66.31 ±0.68**	5.1% ↑
		Mem.	6.37	6.44 (+0.07)	6.41 (+0.04)	6.91 (+0.54)	6.47 (+0.10)	81% ↓

Result Table 2 summarizes the image classification results by various methods. From this table, we can find that the multi-segment activations (APLU and LMA) outperform the other baselines, on both two datasets with all the models of various sizes, where LMA outperforms ReLU by 2% to 6% on accuracy. Meanwhile, we can find that smaller compact model can imply more obvious improvement caused by multi-segment activations. In detail, on CIFAR-10, the LMA outperforms ReLU by 2% on Student 1 while that is 6% on Student 3. Besides, comparing APLU with LMA, we can easily find though their accuracy is sometimes close, the additional inference memory cost brought by equipping APLU is much larger than that by LMA, about 3 to 4 times more.

Machine Translation

Setting To further evaluate the effectiveness of LMA, we also conduct experiments on machine translation using the OpenNMT integration test dataset (Ope) consisting of 200K train sentences and 10K test sentences and WMT13 (Koehn 2005) dataset for a German-English translation task. The translational models we employed are based on the seq2seq models from OpenNMT [4], where the encoder and decoder are both Transformers (Vaswani et al. 2017) instead of LSTM used in (Polino, Pascanu, and Alistarh 2018). LSTM is not selected because its activations are usually Sigmoid and Tanh, both of which are saturated and much different from PLA. Besides one teacher model for each data, we also employ three student models with different sizes on Ope, and two student models on WMT13. We use the perplexity (*Ppl.*, lower is better) and the BLEU score (*BLEU*), computed by the Moses project (mos), as two evaluation metrics.

Result Table 3 shows the results on machine translation. From this table, we can find that our method outperforms all the baseline activations. Specifically, the BLEU scores of LMA increase by 3% to 8% over ReLU on Ope and 1% to 4% on larger WMT13. Moreover, we can observe the similar advantages of LMA in terms of the multi-segment effectiveness and memory cost comparison as in image classification tasks. It is worth to note that using APLU may cause memory overflow due to its huge cost (Out of Memory, OOM), as shown by APLU-equipped Student-1 on WMT13.

[4]https://github.com/OpenNMT/OpenNMT-py

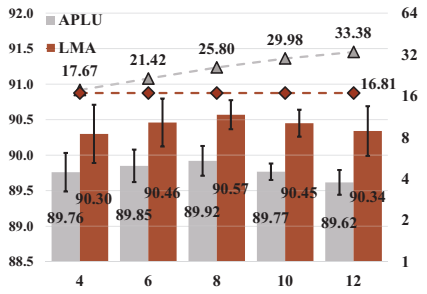

Figure 2: Segment Study for APLU and LMA on CIFAR-10. The bars and left axis show accuracy (%) while the lines and right axis show memory cost (MB). The cost of APLU grows linearly but that of LMA remains much lower.

Additional Experiment

Segment Study To verify if the expressiveness ability can be enhanced via increasing the segment number, we conduct additional experiments on CIFAR-10 to study the effect of segment number k in LMA. As shown in Fig. 2, with the segment number increasing from 4 to 8, both APLU and LMA yield soaring performance. Despite a slight decline beyond 10, LMA is still much better than ReLU. Besides, the memory cost of APLU grows linearly with the segment number while that of LMA remains stable and much lower.

Joint Use To show the effectiveness of the jointly using our method with other compression techniques, we conduct further experiment to combine Quantized Distillation (Polino, Pascanu, and Alistarh 2018) with our method on CIFAR-10. From Table 4, we can find that the accuracy of

Table 3: Machine Translation Results (*Mem.* in MB). The metrics of Teacher models are shown in the left-most cells. Note that on WMT13, the memory needed for training APLU-equipped Student-1 exceeds the maximum memory of our GPU (24GB), thus there is no result of APLU. Besides some similar observations on images, it also shows APLU on translations may also cost so much memory that the task failed, but LMA still works well.

Method			ReLU	PReLU	Swish	APLU-8	LMA-8	
Ope 443.4 MB BLEU 14.92 Ppl. 29.71 Mem.1014.8	Student 1 177.6 MB	Ppl.	31.84 ±0.31	31.89 ±0.64	30.91 ±0.43	30.80 ±0.39	**30.21 ±0.25**	5.1% ↓
		BLEU	13.73 ±0.19	13.67 ±0.27	13.89 ±0.26	13.98 ±0.21	**14.11 ±0.12**	2.8% ↑
		Mem.	407.39	458.98 (+52)	430.77 (+23)	719.73 (+312)	487.23 (+80)	74% ↓
	Student 2 87.2 MB	Ppl.	44.51 ±0.52	44.23 ±0.56	43.44 ±0.39	42.97 ±0.62	**41.21 ±0.35**	7.4% ↓
		BLEU	10.46 ±0.18	10.51 ±0.24	10.78 ±0.23	10.87 ±0.30	**10.94 ±0.18**	4.6% ↑
		Mem.	282.05	335.34 (+53)	305.43 (+23)	596.10 (+314)	363.60 (+82)	74% ↓
	Student 3 43.3 MB	Ppl.	71.69 ±0.51	72.56 ±1.03	70.45 ±0.69	70.31 ±0.61	**67.62 ±0.31**	5.7% ↓
		BLEU	6.12 ±0.12	6.06 ±0.15	6.26 ±0.25	6.40 ±0.29	**6.64 ±0.04**	8.5% ↑
		Mem.	220.49	274.63 (+54)	243.87 (+23)	535.39 (+315)	302.89 (+82)	74% ↓
WMT13 443.4 MB BLEU 28.56 Ppl. 5.31 Mem. 1040.8	Student 1 177.6 MB	Ppl.	6.44 ±0.02	6.47 ±0.03	6.34 ±0.03	OOM	**6.29 ±0.04**	2.3% ↓
		BLEU	26.89 ±0.05	26.81 ±0.06	26.98 ±0.08		**27.12 ±0.07**	0.9% ↑
		Mem.	419.40	470.99 (+52)	442.78 (+23)		499.24 (+81)	N/A
	Student 2 43.3 MB	Ppl.	12.61 ±0.05	12.72 ±0.04	12.51 ±0.03	12.35 ±0.06	**12.25 ±0.05**	2.9% ↓
		BLEU	20.39 ±0.09	19.96 ±0.07	20.82 ±0.08	21.02 ±0.10	**21.19 ±0.08**	3.9% ↑
		Mem.	230.83	284.97 (+54)	254.21 (+23)	545.73 (+315)	313.23 (+82)	74% ↓

Table 4: Joint Use Results with Quantized Distillation on CIFAR-10. The Teacher model employed is the same as the one in the above experiments on CIFAR-10. It shows that LMA works well with Quantization and Distillation, at the same time.

Method	Student 1		Student 2		Student 3	
	ReLU	LMA-8	ReLU	LMA-8	ReLU	LMA-8
4 bits	85.74 ±0.15	**86.31 ±0.41**	77.04 ±0.51	**79.48 ±0.79**	65.33 ±0.17	**68.85 ±0.99**
8 bits	87.02 ±0.23	**88.56 ±0.52**	80.53 ±0.75	**83.37 ±0.51**	70.23 ±0.98	**74.47 ±0.74**

LMA-equipped model is much higher than that of ReLU-equipped one, also by about 2% to 6%, with all different settings of the number of bits in the quantized model.

Overall, all experiments above have implied that the multi-segment activation, including APLU and LMA, can achieve better performance than the two-segment ones, and the improvement brought by multi-segment design becomes increasingly apparent against reducing model size. Therefore, it is quite useful to leverage the segment number of PLA to improve the performance of the compact model in model compression. Furthermore, LMA outperforms APLU mostly and maintain more efficient memory usage simultaneously even in the only one case LMA not beating APLU. It indicates that the high efficiency of LMA makes it quite suitable in resources-exhausted environments. More than this, LMA can also be used conveniently and effectively together with the other techniques. To conclude, LMA can play the most critical role in model compression due to its highly competitive effectiveness, efficiency and compatibility.

Conclusion and Outlook

In model compression, especially knowledge distillation, to fill the expressiveness ability gap between the compact NN and complex NN, we propose a novel highly efficient Light Multi-segment Activation (LMA) in this paper, which empowers the compact NN to yield comparable performance with the complex one. Specifically, to produce more segments but preserving low resource cost, LMA uses statistical information of the batch input to determine multiple segment cut points. Then, it transforms the inputs linearly over different segments. Experimental results on the real-world tasks with multi-size NN have demonstrated the effectiveness and efficiency of LMA. Besides, LMA is well compatible with the other techniques like quantization, also helping the performance of other approaches improved.

To the best of our knowledge, it is the first work that leverages multi-segment piecewise linear activation for model compression, which provides a good insight on designing efficient and powerful compact models. In the future, on the one hand, we will further reduce the time and space costs of LMA computing from the bottom as much as possible, by hardware-level or specialized computation. On the other hand, improving the capacity of activation is also a novel and significant direction to simplify complex architectures and apply Neural Networks more efficiently.

References

Agostinelli, F.; Hoffman, M.; Sadowski, P.; and Baldi, P. 2014. Learning activation functions to improve deep neural networks. *arXiv:1412.6830*.

Courbariaux, M.; Bengio, Y.; and David, J.-P. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, 3123–3131.

Dougherty, J.; Kohavi, R.; and Sahami, M. 1995. Supervised and unsupervised discretization of continuous features. In *Machine Learning Proceedings 1995*. Elsevier. 194–202.

Frankle, J., and Carbin, M. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Goodfellow, I. J.; Warde-Farley, D.; Mirza, M.; Courville, A.; and Bengio, Y. 2013. Maxout networks. *arXiv preprint arXiv:1302.4389*.

Gupta, S.; Agrawal, A.; Gopalakrishnan, K.; and Narayanan, P. 2015. Deep learning with limited numerical precision. In *International Conference on Machine Learning*, 1737–1746.

Gysel, P.; Motamedi, M.; and Ghiasi, S. 2016. Hardware-oriented approximation of convolutional neural networks. *arXiv preprint arXiv:1604.03168*.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, 1135–1143.

Han, S.; Pool, J.; Narang, S.; Mao, H.; Tang, S.; Elsen, E.; Catanzaro, B.; Tran, J.; and Dally, W. J. 2016. Dsd: regularizing deep neural networks with dense-sparse-dense training flow. *arXiv preprint arXiv:1607.04381* 3(6).

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Hassibi, B., and Stork, D. G. 1993. Second order derivatives for network pruning: Optimal brain surgeon. In *Advances in neural information processing systems*, 164–171.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Hu, H.; Peng, R.; Tai, Y.-W.; and Tang, C.-K. 2016. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research* 18(1):6869–6898.

Iandola, F. N.; Han, S.; Moskewicz, M. W.; Ashraf, K.; Dally, W. J.; and Keutzer, K. 2016. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

Ioffe, S., and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv:1502.03167*.

Jin, X.; Yuan, X.; Feng, J.; and Yan, S. 2016. Training skinny deep neural networks with iterative hard thresholding methods. *arXiv preprint arXiv:1607.05423*.

Koehn, P. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, 79–86.

Landwehr, N.; Hall, M.; and Frank, E. 2005. Logistic model trees. *Machine learning* 59(1-2):161–205.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature* 521(7553):436.

LeCun, Y.; Denker, J. S.; and Solla, S. A. 1990. Optimal brain damage. In *Advances in neural information processing systems*, 598–605.

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Graf, H. P. 2016. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*.

Luo, P.; Zhu, Z.; Liu, Z.; Wang, X.; and Tang, X. 2016. Face model compression by distilling knowledge from neurons. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Malash, G. F., and El-Khaiary, M. I. 2010. Piecewise linear regression: A statistical method for the analysis of experimental adsorption data by the intraparticle-diffusion models. *Chemical Engineering Journal* 163(3):256–263.

Mellempudi, N.; Kundu, A.; Mudigere, D.; Das, D.; Kaul, B.; and Dubey, P. 2017. Ternary neural networks with fine-grained quantization. *arXiv preprint arXiv:1705.01462*.

Mishkin, D., and Matas, J. 2015. All you need is a good init. *arXiv:1511.06422*.

Mishra, A., and Marr, D. 2018. Apprentice: Using knowledge distillation techniques to improve low-precision network accuracy. In *International Conference on Learning Representations*.

Mishra, A.; Nurvitadhi, E.; Cook, J. J.; and Marr, D. 2017. Wrpn: wide reduced-precision networks. *arXiv preprint arXiv:1709.01134*.

Montufar, G. F.; Pascanu, R.; Cho, K.; and Bengio, Y. 2014. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, 2924–2932.

Nair, V., and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of international conference on machine learning*, 807–814.

Pascanu, R.; Montufar, G.; and Bengio, Y. 2013. On the number of response regions of deep feed forward networks with piece-wise linear activations. *arXiv preprint arXiv:1312.6098*.

Polino, A.; Pascanu, R.; and Alistarh, D. 2018. Model compression via distillation and quantization. In *International Conference on Learning Representations*.

Ramachandran, P.; Zoph, B.; and Le, Q. V. 2017. Swish: a self-gated activation function. *arXiv:1710.05941*.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, 525–542. Springer.

Sau, B. B., and Balasubramanian, V. N. 2016. Deep model compression: Distilling knowledge from noisy teachers. *arXiv preprint arXiv:1610.09650*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, J.; Bao, W.; Sun, L.; Zhu, X.; Cao, B.; and Yu, P. S. 2018. Private model compression via knowledge distillation.

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning structured sparsity in deep neural networks. In *Advances in neural information processing systems*, 2074–2082.

Wu, J.; Leng, C.; Wang, Y.; Hu, Q.; and Cheng, J. 2016. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4820–4828.

Yang, T.-J.; Chen, Y.-H.; and Sze, V. 2017. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5687–5695.

Zagoruyko, S., and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhu, C.; Han, S.; Mao, H.; and Dally, W. J. 2016. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*.