Strategies for Applying Low Rank Decomposition to Transformer-Based Models

Habib Hajimolahoseini
Ascend Team
Toronto Research Center
Huawei Technologies
habib.hajimolahoseini@huawei.com

Walid Ahmed
Ascend Team
Toronto Research Center
Huawei Technologies

Mehdi Rezagholizadeh
Huawei Noah's Ark Lab

Vahid Partovinia
Huawei Noah's Ark Lab

Yang Liu
Ascend Team
Toronto Research Center
Huawei Technologies

Abstract

Low rank decomposition decomposes each fully-connected layer of the transformer modules into two smaller layers using Singular Value Decomposition. The state-of-the-art techniques usually apply LRD in a single-shot, where all of the layers are decomposed simultaneously. In this paper, we propose and compare different strategies for applying low rank decomposition to compress pre-trained transformer based models. These strategies include: layer-by-layer and progressive decomposition. We observe that progressive low rank decomposition, in which the rank is decreased incrementally results in a higher accuracy after decomposition comparing to single-shot and layer-by-layer low rank decomposition. Furthermore, in contrast with many of state-of-the-art compression methods where intensive pre-training of the compressed model is necessary, we show that progressive LRD can provide promising performance by compressing the model in the fine-tuning stage.

1 Introduction

Deep learning models have become larger and larger over the past few years. In some of the Transformer-based models such as GPT, the number of trainable parameters can reach more than billions (Radford et al., 2018). Training or finetuning these huge models is a substantially expensive process with a huge memory consumption (Dean et al., 2012). With a widespread application of real-time AI models on smartphones and other embedding devices, deploying these over-sized models on device can be very challenging. Compressing deep learning models is a well-known potential solution with a set of diverse existing approaches to improve the efficiency of deep learning models at the cost of compromising their accuracy.

Current model compression algorithms may use different techniques for reducing the number of parameters or computational complexity including: Low Rank Decomposition (LRD), pruning, quantization and Knowledge Distillation (KD) (Cheng et al., 2017). In quantization-based techniques, model weights are approximated using a smaller number of bits e.g. 16-bits, 8-bits or even a single bit (binary networks) (Wu et al., 2016; Han et al., 2015; Courbariaux et al., 2015; Prato et al., 2019; Bie et al., 2019). Model pruning is another approach which removes non-important neurons or redundant connections between them (Lebedev and Lempitsky, 2016; Wen et al., 2016). On the other hand, KD approaches transfer the knowledge from a larger model (teacher) into a smaller one (student) by

36th Conference on Neural Information Processing Systems (NeurIPS 2022) Track on Datasets and Benchmarks.

adding and auxiliary loss to imitate softmax outputs or logits from the teacher as a representation of class distributions (Hinton et al., 2015). The goal is for the student model to mimic the behaviour of the teacher model and the knowledge could be distilled from the teachers output or from the intermediate layers (Mirzadeh et al., 2020).

In contrast, LRD techniques decompose each layer of DNNs using a tensor or matrix factorization method. LRD can be applied to both fully-connected and convolutional layers, using a matrix or tensor factorization technique, respectively. One advantage of LRD methods is that they do not need heavy pre-training because they start from a large pre-trained model to initialize the compressed model. This is in contrast to the layer truncation technique where we should train the compressed model from scratch. Therefore when using LRD, a few fine-tuning steps are enough to recover most of the accuracy drop.

However, it is worth mentioning that when using LRD for large compression factors, the approximation of the original weights of the compressed model becomes very poor and thus recovering the accuracy through fine-tuning becomes very difficult. Progressive or layer-by-layer LRD (Hajimolahoseini et al.) is a potential way of addressing this issue. In layer-by-layer LRD, only a portion of the layers are decomposed at each round, followed by a fine-tuning step Gusak et al. (2019). In progressive LRD, compression is applied in small steps iteratively, therefore in each step, the compressed model is an approximation of the previous model and thus has a better initialization point on the loss surface (Hajimolahoseini et al.).

Although prior arts (Mao et al., 2020) have used LRD for compressing the large transformer-based models, this technique has only been used and studied in a single-shot setup, in which all of the layers are decomposed at once and the model is then fine-tuned. To our knowledge, there is no study for comparing the advantages and disadvantages of different strategies including layer-by-layer and progressive decomposition when applied to transformer-based models. In this paper, we apply LRD with different strategies to investigate their performance on the extreme compression of Transformer-based models.

2 Low Rank Decomposition

Transformer modules of the NLP models include feed forward and multi-head attention modules, in which the trainable blocks are fully-connected (FC) layers (Vaswani et al., 2017). The FC layers have a 2D weight matrix $\mathbf{W}$ of size (C, S), where C and S represent the number of input and output features, respectively. Suppose that the input to the FC layer is a two dimensional tensor $\mathbf{X}$ of size (B, C), where B and C are the batch size and the number of input features, respectively. The FC layer will multiply the input tensor $\mathbf{X}$ with its weight matrix and deliver the output $\mathbf{Y}$ of shape (B, S) through $\mathbf{Y} = \mathbf{X}\mathbf{W}$. In most NLP models, the dimensions of FC layers could be so large that their weight matrix includes millions of parameters only in a single FC layer.

The rank of matrix $\mathbf{W}$ can be interpreted in different ways, as the number of its linearly independent columns or rows sometimes even the condition index, condition number, normalized trace or determinants are used as proxy to the rank. The low rank approximation of matrix $\mathbf{W}$ is an optimization problem which minimizes the difference between $\mathbf{W}$ and its low rank version. In terms of Singular Value Decomposition (SVD), this optimization problem has an analytical solution as follows (Van Loan, 1987): Assume that the weight matrix $\mathbf{W} \in \mathbb{R}^{\mathrm{C}\times\mathrm{S}}$ is decomposed using SVD as follows

$$W = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\top} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^{\top}, \tag{1}$$

where $\mathbf{U} \in \mathbb{R}^{\mathrm{C}\times\mathrm{C}}$ and $\mathbf{V} \in \mathbb{R}^{\mathrm{S}\times\mathrm{S}}$ are the orthogonal matrices and $\mathbf{\Sigma} \in \mathbb{R}^{\mathrm{C}\times\mathrm{S}}$ is a diagonal rectangular matrix containing the singular values $\sigma_i >$ of $\mathbf{W}$ of rank r. In (1), if we only use the first R terms of the summation, the resulted matrix $\mathbf{W}'$ would be an approximation of $\mathbf{W}$ with a lower rank $R < r$:

$$\mathbf{W}' = \sum_{i=1}^{R} \sigma_i \mathbf{u}_i \mathbf{v}_i^{\top} = \mathbf{U}'\mathbf{\Sigma}'\mathbf{V}'^{\top} \tag{2}$$

where $\mathbf{U}' \in \mathbb{R}^{\mathrm{C}\times\mathrm{R}}$ and $\mathbf{V}' \in \mathbb{R}^{\mathrm{S}\times\mathrm{R}}$ are the new orthogonal matrices and $\mathbf{\Sigma}' \in \mathbb{R}^{\mathrm{R}\times\mathrm{R}}$ is the new diagonal rectangular matrix. Therefore, the low rank estimation problem could be explained as the

following minimazation

$$\min_{\text{rank}(\mathbf{X})\leq R} \|\mathbf{W}-\mathbf{X}\|_2 = \|\mathbf{W}-\mathbf{W}'\|_2 \,. \tag{3}$$

According to (2), a QR-type reconstruction of $\mathbf{W}'$ is possible multiplication the following two matrices

$$\mathbf{W}' = \mathbf{W}_0\mathbf{W}_1, \quad \mathbf{W}_0 = \mathbf{U}'\sqrt{\mathbf{\Sigma}'}, \quad \mathbf{W}_1 = \sqrt{\mathbf{\Sigma}'}\mathbf{V}'^{\top} \tag{4}$$

where $\mathbf{W}_0 \in \mathbb{R}^{\mathrm{C}\times\mathrm{R}}$ and $\mathbf{W}_1 \in \mathbb{R}^{\mathrm{R}\times\mathrm{S}}$, and $\sqrt{\mathbf{\Sigma}}$ is a diagonal of square root of singular values $\sqrt{\sigma_i}$. It is clear that if we replace the FC layer $\mathbf{W}$ with two consecutive FC layers $\mathbf{W}_0$ and $\mathbf{W}_1$, the number of parameters could shrink significantly depending on the rank R.

3 Layer-by-Layer Decomposition

Different strategies can be used for applying LRD to a deep learning model. In contrast with applying a single shot LRD to all layers of the model at the same time, one alternative is to do it layer by layer. In this strategy, we apply LRD only to a subset of FC layers in the model. The new model is then finetuned for some number of epochs, before the LRD is applied to the next subset of layers. This process is repeated until all of the layers are decomposed. For transformer-based models, we decompose all of the FC layers inside each of the transformer modules together. For example, if the model consists of 12 transformer module, the LRD could be applied in at most 12 steps using layer-by-layer method.

4 Progressive Decomposition

In progressive LRD, we use a rank selection approach in which, we start from a lower compression ratio and apply low rank decomposition to the entire model altogether (just like the single-shot method). The model is then fine-tuned for some epochs in order to recover the accuracy. The rank R is decreased incrementally in the next steps so that we have a higher compression ratio. This process is repeated until the desired compression ratio is achieved for the model. Starting from the second compression step, we only decompose the second decomposed layer of previously decomposed layers and multiply them so that it results in two decomposed layers again. This way, the number of layers will not increase again in the next stages of decomposition as we use matrix multiplication for merging the newly generated weight matrices in order to keep the number of layers the same as the first stage of LRD. This process is shown in Fig. 1.

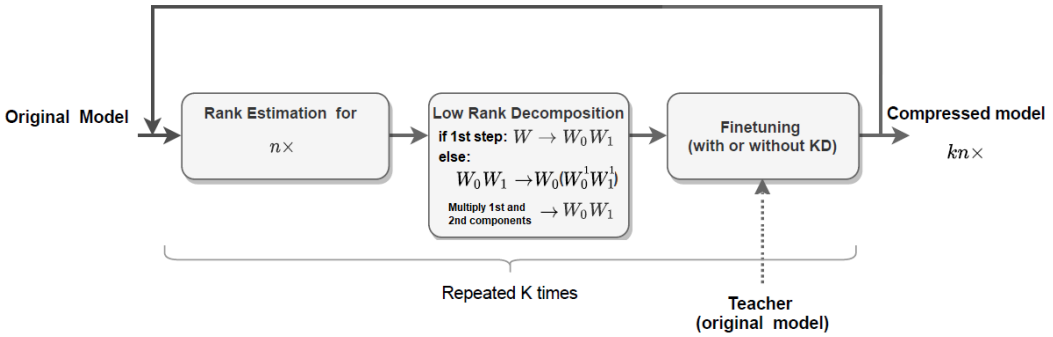

Figure 1: Progressive decomposition scheme

5 Experimental Results

In order to compare different strategies for applying LRD, we use the vision transformers model known as ViT as our baseline Dosovitskiy et al. (2020). We use the pre-trained weights of ImageNet-21k dataset for initialization and then fine-tune on CIFAR-10 and CIFAR-100 for 12 epochs. In the single-shot cases, we apply LRD with $16\times$ compression ratio to all of the feed forward layers, resulting in total compression ratio of $2.63\times$ for the ViT model. For layer-by-layer case,

we decompose 25% of the feed forward layers of the ViT model at each compression step with a compression rate of $2\times$. The resulting model is then fine-tuned for 3 epochs. This process is repeated for the next 25% of the model (4 times) until the entire model is decomposed. In the progressive case, all of the feed forward layers are decomposed with $2\times$ compression ratio and then finetuned for 3 epochs. This process is repeated 4 times untill the feed forward layers are compressed by $16\times$, resulting in the overall compression of $2.63\times$ for the ViT model. The results are reported in Table 1.

Table 1: Experimental results for ViT model decomposed LRD using different strategies including single-shot, layer-by-layer and progressive LRD. The model is pretrained on ImageNet-21k and finetuned on CIFAR-10 and CIFAR-100 datasets.

Method	**# LRD Steps**	**Comp Ratio**	**CIFAR-10 Top-1 (%)**	**CIFAR-100 Top-1 (%)**
ViT (Org)	0	$1\times$	98.76	92.95
Single-Shot	1	$2.63\times$	94.53	87.09
Layer-by-Layer	4	$2.63\times$	95.36	87.51
Progressive	4	$2.63\times$	96.01	88.62

As seen in Table 1, the progressive strategy shows the highest accuracy comparing to the other methods for both datasets. The layer-by-layer method is also slightly better than single-shot technique. However, it is worth mentioning that the single-shot technique requires to compile the LRD model only once whil the other techniques compile the model 4 times in this experiment. Therefore, in the cases where the total training time has a higher priority comparing to accuracy drop, the single-shot method could be considered.

Note that we used only 12 epochs for fine-tuning in both CIFAR-10 and CIFAR-100 cases just to compare different LRD strategies in the same condition. That is the reason for higher accuracy gap between original and LRD models for CIFAR-100 dataset.

In our next experiment, we use the pre-trained language model GPT2 as our baseline and apply different compression techniques to it (Radford et al., 2019). The baseline GPT2 model includes 12 Transformer layers. In our experiments, we also include three truncated versions of GPT2 including DistilGPT2, GPT2-6L and GPT2-4L which are the truncated models with 6, 6 and 4 Transformer layers, respectively. We use the standard Wikitext-103 benchmark dataset (Merity et al., 2016) in which the perplexity is the measure of performance. We also apply two versions of LRD method, one with a single-shot LRD and one with progressive LRD with 2 rounds of $2\times$ compression to the FC layers inside each Transformer module. The result is reported in Table 2.

Table 2: Experimental results for GPT2 using low rank decomposition v.s. layer truncation on Wikitext-103 dataset. GPT2-6L and GPT2-4L are the truncated versions of GPT2 with 6 and 4 Transformer layers, respectively. The LRD result is after applying 2 rounds of $2\times$ compression (total of $4\times$).

Method	**# Layers**	**# Params (M)**	**Perplexity**
GPT2 (Baseline)	12	124	16.3
Distiltgpt2	6	82	21.1
GPT2-6L	6	82	22.7
GPT2-4L	4	68	30.0
Single Step LRD	12	47	28.2
Progressive LRD	12	31	22.0

As seen in Table 2, the progressive LRD model is able to reach a perplexity close to DistilGPT2, with only 0.9 unit gap in perplexity, while providing a much higher compression ratio of $4\times$ comparing to that of truncated versions e.g. DistilGPT2 which is $1.5\times$. The results for GPT2-4L also shows that if we truncate 2 more layers of the GPT2 to get a higher compression ratio, we will start to lose the perplexity significantly (around 9 perplexity units) while the compression ratio will only increase from $1.5\times$ to $1.84\times$.

6 Conclusion

In this work, different strategies for applying LRD was proposed and studied, including single-shot, layer-by-layer and progressive. In contrast to many of state-of-the-art compression methods where

intensive pre-training of the compressed model is necessary, progressive LRD can provide promising performance by compressing the model in the fine-tuning stage. This leads to reduction in the computation resources needed for obtaining a compressed model for a given task. We show that using an iterative compression where the deep learning models becomes compressed progressively can achieve higher compression ratios with less drop in accuracy comparing to single-shot or layer-by-layer LRD. Future direction could be to apply progressive low rank decomposition to other architectures based on transformers.

References

Alex Bie, Bharat Venkitesh, Joao Monteiro, Md Haidar, Mehdi Rezagholizadeh, et al. 2019. A simplified fully quantized transformer for end-to-end speech recognition. arXiv preprint arXiv:1911.03604.

Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2017. A survey of model compression and acceleration for deep neural networks. arXiv preprint arXiv:1710.09282.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In Advances in neural information processing systems, pages 3123–3131.

Jeffrey Dean, Greg S Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V Le, Mark Z Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, et al. 2012. Large scale distributed deep networks.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Julia Gusak, Maksym Kholiavchenko, Evgeny Ponomarev, Larisa Markeeva, Ivan Oseledets, and Andrzej Cichocki. 2019. Musco: Multi-stage compression of neural networks. arXiv preprint arXiv:1903.09973.

Habib Hajimolahoseini, Mehdi Rezagholizadeh, Vahid Partovinia, Marzieh Tahaei, Omar Mohamed Awad, and Yang Liu. Compressing pre-trained language models using progressive low rank decomposition.

Song Han, Huizi Mao, and William J Dally. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

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

Vadim Lebedev and Victor Lempitsky. 2016. Fast convnets using group-wise brain damage. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2554–2564.

Yihuan Mao, Yujing Wang, Chufan Wu, Chen Zhang, Yang Wang, Yaming Yang, Quanlu Zhang, Yunhai Tong, and Jing Bai. 2020. Ladabert: Lightweight adaptation of bert through hybrid model compression. arXiv preprint arXiv:2004.04124.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. 2020. Improved knowledge distillation via teacher assistant. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 34, pages 5191–5198.

Gabriele Prato, Ella Charlaix, and Mehdi Rezagholizadeh. 2019. Fully quantized transformer for machine translation. arXiv preprint arXiv:1910.10485.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9.

Charles Van Loan. 1987. Matrix computations and signal processing. Technical report, Cornell University.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in neural information processing systems, pages 5998–6008.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. 2016. Learning structured sparsity in deep neural networks. Advances in neural information processing systems, 29:2074–2082.

Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. 2016. Quantized convolutional neural networks for mobile devices. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4820–4828.