Device Placement for Neural Networks

Tianze Wang
EECS, SCS
KTH Royal Institute of Technology
Stockholm, Sweden
tianzew@kth.se

I. Introduction

Now at the start of 2021, it is hard to deny that Machine Learning (ML) is everywhere. This is perhaps even more true when it comes to Deep Neural Networks (DNNs). Thanks to the availability of large training datasets, DNNs have driven advances in many practical problems, e.g., image classification, speech recognition, machine translation, image generation, and autonomous driving.

However, together with the boost in performance of Deep Learning models comes the growth in the computational requirements for training DNNs. Training complex DNN model requires an increasing amount of computation and it becomes typical to speed up the training process through parallelization by utilizing a mixture of devices (e.g., CPUs, GPUs). One common way to perform parallelization is data parallelism where a dataset is partitioned and different partitions are distributed across multiple devices that each holds a complete copy of the model. In model parallelism, on the other hand, we partition the model itself. However, the efficient placement of computational graphs (DNN models) onto hardware devices is not a trivial problem. On the one hand, certain constraints must be satisfied, e.g., the parts given to a device should not exceed its memory capacity. On the other hand, a balance between computation and communication among the given devices needs to be achieved to minimize runtime.

Device placement studies how to automatically partition a DNN for a given hardware topology so that the training and inference time of the DNN is minimized. To achieve a good performance, device placement methods need to learn which parts of the DNN model should be placed on which device and how to arrange the computations so that the communication is also optimized. Early effort [1] solves device placement using graph partitioning and scheduling techniques and proposes several heuristics, e.g, critical path, to partition the DNN computation graph. State-of-the-art methods turn to reinforcement learning for device placement. For example, **ColocRL** [2] uses a sequence-to-sequence Recurrent Neural Network (RNN) that takes a list of operator groups as inputs and outputs a device placement for each group. The model is trained using reinforcement learning to minimize the total execution time.

Review report for FID3018 (HT20-VT21) : Advanced Course in Data Mining and Analytics.

In the rest of the review, we will discuss some of the efforts that researchers have made to automate the process of device placement to speed up the training process. We will start with a classical hierarchical approach for device placement using reinforcement learning and then see how cross-entropy minimization and proximal policy optimization can be integrated to improve efficiency. We will end the session with some short discussions of the pros and cons as well as possible alternatives for approaching the device problem.

Papers

1) Mirhoseini, Azalia, et al. ”A hierarchical model for device placement.” International Conference on Learning Representations. 2018;
2) Gao, Yuanxiang, Li Chen, and Baochun Li. ”Spotlight: Optimizing device placement for training deep neural networks.” International Conference on Machine Learning. 2018;
3) Gao, Yuanxiang, Li Chen, and Baochun Li. ”Post: Device placement with crossentropy minimization and proximal policy optimization.” Advances in Neural Information Processing Systems. 2018..

Hereafter, we are going to refer to these three papers by the names of the frameworks they propose which are **HDP** for the first paper, **Spotlight** for the second paper, and **Post** for the third paper.

II. Device Placement for Neural Networks

A Hierarchical Model for Device Placement [3] (HDP)

Motivation: HDP is a follow-up of ColocRL by the same research group. Its goal is to overcome one of the limitations of the previous work highlighted above: the need for expert-based or heuristic-based grouping of operations.

Contribution: The authors propose HDP, a Reinforcement Learning-based approach for device placement, very similar to ColocRC, but with an additional layer that learns how to assign operations to groups, without the need for any heuristics.

Solution: HDP is, as the name suggests, a hierarchical model, composed of stacking two separate networks. The top network is identical to the one presented in ColocRL. The novelty is in the bottom network, which generates the group embeddings that are input to the top network. This bottom network is a simple feed-forward neural network with a final softmax layer, which maps each operation in the graph to one of a predefined number of groups.

The rest of the system is identical to ColocRL: the Reinforcement Learning training is done using the REINFORCE equations, using the square root of the running time as the reward, and sampling K placements in each iteration. The distributed architecture for RL training is also the same, with multiple controllers and workers.

Strong Points: This paper successfully addresses one of the limitations of ColocRL, removing one of the non-trainable heuristic-based aspects that could hold back the quality of the placement. While the total number of groups still needs to be manually hard-coded in the neural network, the authors noted that the system learns to use a smaller number of groups than the maximum specified, when this provides better results.

For the rest, this paper shares most of the strong and weak points of the previous work. This includes the convincing results section, with several real-world models and datasets and various baselines for comparison.

One weak point of the previous work that was resolved in this paper is the lack of estimates regarding the overhead of RL training compared to actual model training. While the authors do not provide a comprehensive analysis of the issue, they provide an example, in which the amount of GPU-hours saved by the placement is one order of magnitude larger than the number of GPU-hours spent training HDP.

Weak Points: As clearly detailed in the review of the previous paper, the lack of explicit coding of device and operator properties makes the RL model very specific for each computational graph and runtime environment. This is probably the most important limitation of this line of work.

However, one additional weak point is the lack of a proper comparison with the previous work from the same authors, ColocRL, which provides a very similar solution to the problem. The authors say that it is not possible to provide comparable numbers, because the previous work was using different GPUs and an older version of TensorFlow. However, as the authors of HDP and ColocRL are the same, it would seem easy to adapt their previous code to provide meaningful, comparable numbers. Instead, the authors provide an example and a brief intuitive explanation of why HDP should perform better than ColocRL, due to its higher flexibility.

Spotlight: Optimizing Device Placement for Training Deep Neural Networks [4] (Spotlight)

Motivation: Previous RL-based approaches for device placement use the policy gradient method and suffers from sub-optimal training times.

Contribution: Spotlight paper proposes a new reinforcement learning algorithm, i.e., Spotlight, based on Proximal Policy Optimization (PPO).

Solution: For Spotlight to work, the device placement problem is modeled as a Markov Decision Process (MDP) with multiple stages. At each stage, the system occupies a state about the placements of the previous operations on GPU and CPU devices. We then select the next operation and sample a probability distribution to obtain a placement recommendation on one of the available devices. After the operation is placed at a recommended device, the system transitions to the next stage with a new placement state where the previous operation has been placed.

By repeating these transitions, the entire neural network is completely placed. The training time of the neural network with the final placement state is the reward of the MDP.

Strong Points: Spotlight proposes a more efficient way for training the policy of device placement problem while also introducing the MDP model for the device placement problem. There is a theoretical guarantee on the performance improvements.

Weak Points: The grouping (colocation) methods used for Spotlight is rather simple. Also, it would be better if the empirical evaluations were performed on more complex CNN models.

Post: Device placement with crossentropy minimization and proximal policy optimization [5] (Post)

Motivation: State-of-the-art solves device placement problems with a reinforcement learning approach based on the policy gradient method. This method is inefficient because it relies on the Monte Carlo method to generate samples (treat each data sample equally without emphasizing important ones. Cross-entropy Method, on the other hand, is an important sampling technique. There are also theoretical guarantees on the optimal efficiency when generating samples.

Contribution: Post integrates cross-entropy minimization (a batch learning algorithm) and PPO (an online reinforcement learning algorithm) to achieve faster learning with theoretical guarantees on optimal efficiency. Empirical evaluation has demonstrated that Post has better learning performance than the policy gradient method, proximal policy optimization, and the cross-entropy method alone.

Solution: Post represents device placement as a high-dimensional softmax probability distribution, which translates the problem of finding the best placement to one of estimating the optimal density. The authors also proposed an algorithm that integrates the proximal policy optimization method, an online reinforcement learning algorithm, and the cross-entropy method, a batch learning algorithm.

While PPO achieves a higher sample efficiency than policy gradient, both methods perform only one step or several steps of stochastic gradient descent/ascent, which might be considered slow as they do not achieve the optimum. Cross-entropy minimization methods, on the other hand, directly work towards the optimal solution which improves efficiency.

Post proposed a joint learning algorithm to combine the proximal policy optimization method and cross-entropy method. During training, for every K sampled placements, Post performs several stochastic gradient ascent steps with regard to the objective of PPO (incremental policy improvements). For every N sampled placement (N is several or tens of times larger than K), Post solves the cross-entropy minimization problem (a global and an aggressive policy improvement).

Strong Points: Post uses softmax distributions instead of an RNN for the policy representation. The PPO method used offers better efficiency than the policy gradient method.

Post proposes a customized cross-entropy method to estimate the optimal placement, which theoretically guarantees the best possible efficiency.

Post uses a joint learning algorithm to combine PPO with the cross-entropy method to achieve better training efficiency.

Empirical evaluation demonstrates that Post achieved a significantly better learning performance than the policy gradient method, proximal policy optimization, and the cross-entropy method alone.

Weak Points: While the joint learning process does demonstrate better performance in terms of placement runtime, the joint learning process might require extra work on tuning the frequency of performing PPO and cross-entropy method. Also, the joint training process requires a bit of extra coding when performing the experiment.

In the related work section, the term "This work ..." sometimes causes confusion on whether it refers to the related work mentioned or Post itself.

III. Discussion & Conclusion

To summarize, in the study of device placement for neural networks, deep reinforcement learning approaches serve as a common solution in state-of-the-art methods. While the problem formulation of device placement has not changed much, efforts have been made to improve the training efficiency of device placement. On the one hand, more efficient reinforcement learning methods, e.g., proximal policy optimization, have been adapted. On the other hand, other optimization techniques like the cross-entropy method have also been used. Apart from that, we can also observe the performance improvement brought by the use of more advanced policy representations, e.g., the replacement of RNN with LSTM, the introduction of graph embeddings.

Generalization of device placement has also gained its research interest as it is infeasible to retrain the model for device placement every time when facing a slightly different neural network. Thus, generalization will be an interesting topic to work on in the future.

References

[1] R. Mayer, C. Mayer, and L. Laich, "The tensorflow partitioning and scheduling problem: it's the critical path!" in *Proceedings of the 1st Workshop on Distributed Infrastructures for Deep Learning*, 2017, pp. 1–6.

[2] A. Mirhoseini, H. Pham, Q. V. Le, B. Steiner, R. Larsen, Y. Zhou, N. Kumar, M. Norouzi, S. Bengio, and J. Dean, "Device placement optimization with reinforcement learning," *arXiv preprint arXiv:1706.04972*, 2017.

[3] A. Mirhoseini, A. Goldie, H. Pham, B. Steiner, Q. V. Le, and J. Dean, "A hierarchical model for device placement," in *International Conference on Learning Representations*, 2018.

[4] Y. Gao, L. Chen, and B. Li, "Spotlight: Optimizing device placement for training deep neural networks," in *International Conference on Machine Learning*, 2018, pp. 1676–1684.

[5] ——, "Post: Device placement with cross-entropy minimization and proximal policy optimization," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018.