Google AI Blog

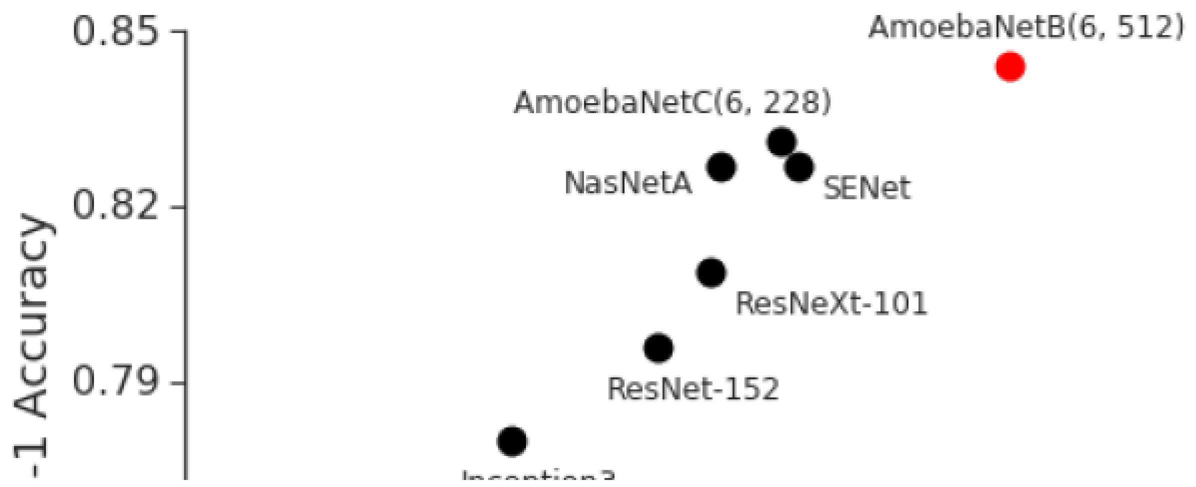
The latest from Google Research

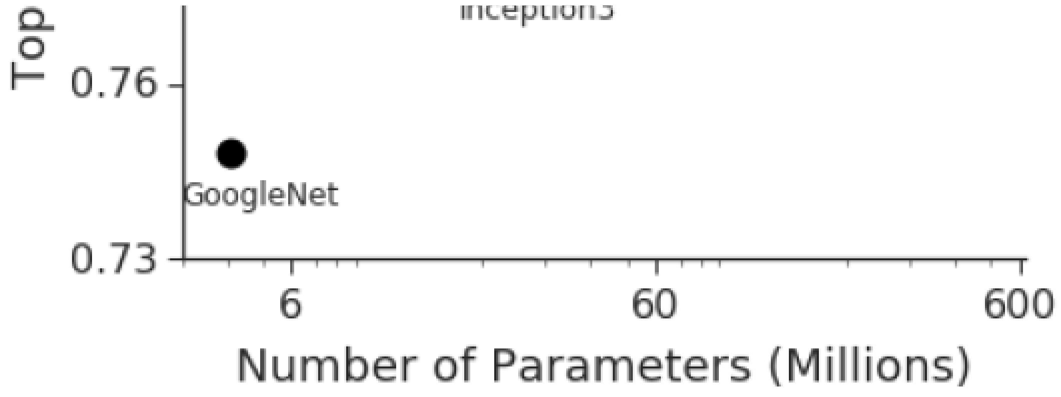
Introducing GPipe, an Open Source Library for Efficiently Training Large-scale Neural Network Models

Monday, March 4, 2019

Posted by Yanping Huang, Software Engineer, Google AI

Deep neural networks (DNNs) have advanced many machine learning tasks, including speech recognition, visual recognition, and language processing. Recent advances by BigGan, Bert, and GPT2.0 have shown that ever-larger DNN models lead to better task performance and past progress in visual recognition tasks has also shown a strong correlation between the model size and classification accuracy. For example, the winner of the 2014 ImageNet visual recognition challenge was GoogleNet, which achieved 74.8% top-1 accuracy with 4 million parameters, while just three years later, the winner of the 2017 ImageNet challenge went to Squeeze-and-Excitation Networks, which achieved 82.7% top-1 accuracy with 145.8 million (36x more) parameters. However, in the same period, GPU memory has only increased by a factor of ~3, and the current state-of-the-art image models have already reached the available memory found on Cloud TPUv2s. Hence, there is a strong and pressing need for an efficient, scalable infrastructure that enables large-scale deep learning and overcomes the memory limitation on current accelerators.



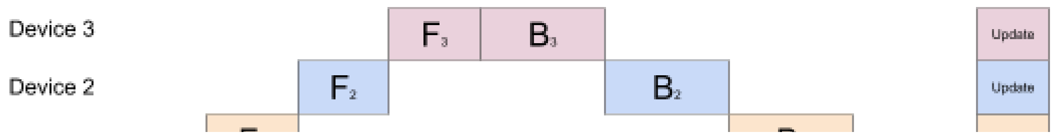


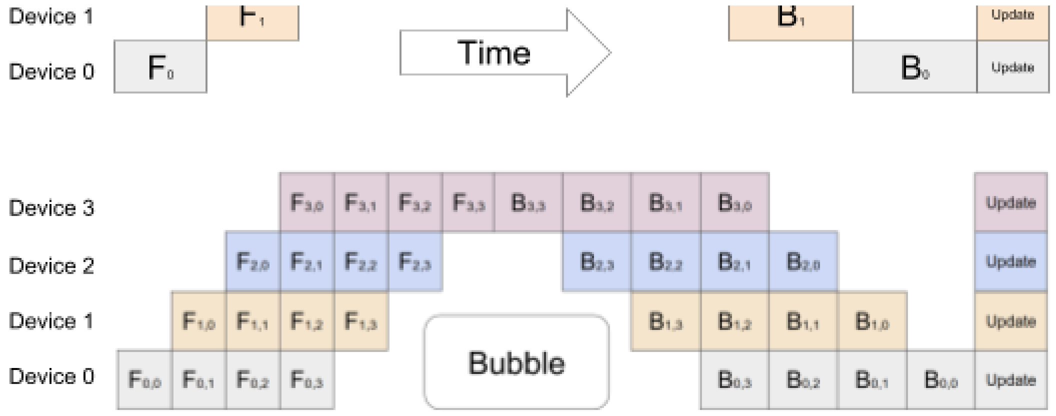
Strong correlation between ImageNet accuracy and model size for recently developed representative image classification models

In "GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism", we demonstrate the use of pipeline parallelism to scale up DNN training to overcome this limitation. GPipe is a distributed machine learning library that uses synchronous stochastic gradient descent and pipeline parallelism for training, applicable to any DNN that consists of multiple sequential layers. Importantly, GPipe allows researchers to easily deploy more accelerators to train larger models and to scale the performance without tuning hyperparameters. To demonstrate the effectiveness of GPipe, we trained an AmoebaNet-B with 557 million model parameters and input image size of 480 x 480 on Google Cloud TPUv2s. This model performed well on multiple popular datasets, including pushing the single-crop ImageNet accuracy to 84.3%, the CIFAR-10 accuracy to 99%, and the CIFAR-100 accuracy to 91.3%. The core GPipe library has been open sourced under the Lingvo framework.

From Mini- to Micro-Batches
There are two standard ways to speed up moderate-size DNN models. The data parallelism approach employs more machines and splits the input data across them. Another way is to move the model to accelerators, such as GPUs or TPUs, which have special hardware to accelerate model training. However, accelerators have limited memory and limited communication bandwidth with the host machine. Thus, model parallelism is needed for training a bigger DNN model on accelerators by dividing the model into partitions and assigning different partitions to different accelerators. But due to the sequential nature of DNNs, this naive strategy may result in only one accelerator being active during computation, significantly underutilizing accelerator compute capacity. On the other hand, a standard data parallelism approach allows concurrent training of the same model with different input data on multiple accelerators, but cannot increase the maximum model size an accelerator can support.

To enable efficient training across multiple accelerators, GPipe partitions a model across different accelerators and automatically splits a *mini-batch* of training examples into smaller *micro-batches*. By pipelining the execution across micro-batches, accelerators can operate in parallel. In addition, gradients are consistently accumulated across micro-batches, so that the number of partitions does not affect the model quality.



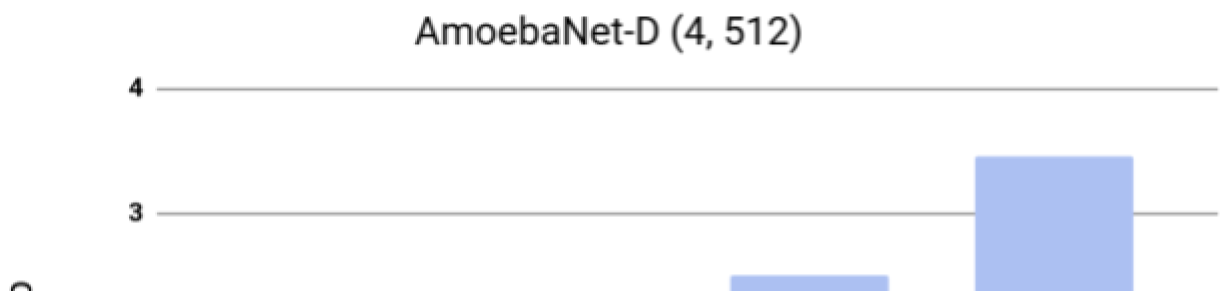


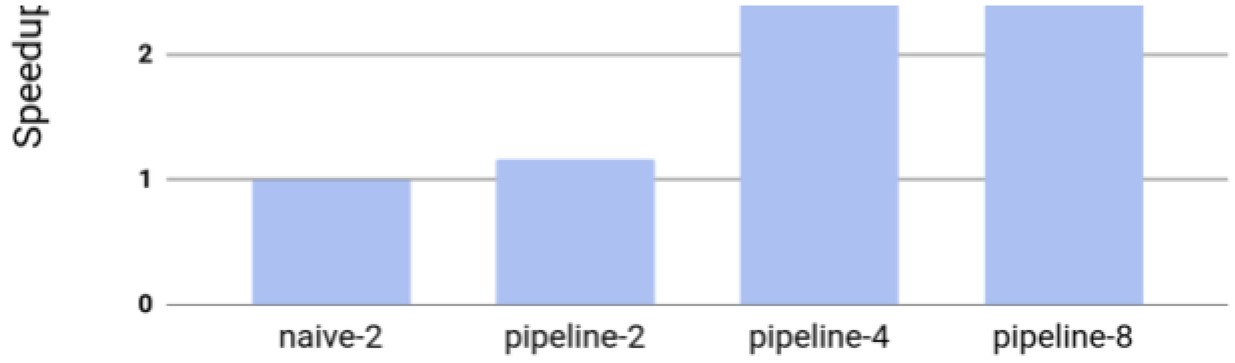
Top: *The naive model parallelism strategy leads to severe underutilization due to the sequential nature of the network. Only one accelerator is active at a time.* ***Bottom:*** *GPipe divides the input mini-batch into smaller micro-batches, enabling different accelerators to work on separate micro-batches at the same time.*

Maximizing Memory and Efficiency

GPipe maximizes memory allocation for model parameters. We ran the experiments on Cloud TPUv2s, each of which has 8 accelerator cores and 64 GB memory (8 GB per accelerator). Without GPipe, a single accelerator can train up to 82 million model parameters due to memory limits. Thanks to recomputation in backpropagation and batch splitting, GPipe reduced intermediate activation memory from 6.26 GB to 3.46GB, enabling 318 million parameters on a single accelerator. We also saw that with pipeline parallelism the maximum model size was proportional to the number of partitions, as expected. With GPipe, AmoebaNet was able to incorporate 1.8 billion parameters on the 8 accelerators of a Cloud TPUv2, 25x times more than is possible without GPipe.

To test efficiency, we measured the effects of GPipe on the model throughput of AmoebaNet-D. Since training required at least two accelerators to fit the model size, we measured the speedup with respect to the naive case with two partitions but no pipeline parallelization. We observed an almost linear speedup in training. Compared to the naive approach with two partitions, distributing the model across four times the accelerators achieved a speedup of 3.5x. While all experiments in our paper used Cloud TPUv2, we see even better performance with the currently available Cloud TPUv3s, each of which has 16 accelerator cores and 256 GB (16 GB per accelerator). GPipe enabled 8 billion parameter Transformer language models on 1024-token sentences with a speedup of 11x when distributing the model across all sixteen accelerators.





Speedup of AmoebaNet-D using GPipe. This model could not fit into one accelerator. The baseline naive-2 is the performance of the native partition approach when the model is split into two partitions. Pipeline-k refers to the performance of GPipe that splits the model into k partitions with k accelerators.

GPipe can also scale training by employing even more accelerators without changes in the hyperparameters. Therefore, it can be combined with data parallelism to scale neural network training using even more accelerators in a complementary way.

Testing Accuracy
We used GPipe to verify the hypothesis that scaling up existing neural networks can achieve even better model quality. We trained an AmoebaNet-B with 557 million model parameters and input image size of 480 x 480 on the ImageNet ILSVRC-2012 dataset. The network was divided into 4 partitions and applied parallel training processes to both model and data. This giant model reached the state-of-the-art 84.3% top-1 / 97% top-5 single-crop validation accuracy without any external data. Large neural networks are not only applicable to datasets like ImageNet, but also relevant for other datasets through transfer learning. It has been shown that better ImageNet models transfer better. We ran transfer learning experiments on the CIFAR10 and CIFAR100 datasets. Our giant models increased the best published CIFAR-10 accuracy to 99% and CIFAR-100 accuracy to 91.3%.

Conclusion
The ongoing development and success of many practical machine learning applications, such as autonomous driving and medical imaging, depend on achieving the highest accuracy possible. As this often requires building larger and even more complex models, we are happy to provide GPipe to the broader research community, and hope it is a useful infrastructure for efficient training of large-scale DNNs.

Acknowledgments
Special thanks to the co-authors of the paper: Youlong Cheng, Dehao Che, HyoukJoong Lee, Jiquan Ngiam, Quoc V. Le, and Zhifeng Chen. We wish to thank Esteban Real, Alok Aggarwal, Xiaodan Song, Naveen Kumar, Mark Heffernan, Rajat Monga, Megan Kacholia, Samy Bengio, and Jeff Dean for their support and valuable input; Noam Shazeer, Patrick Nguyen, Xiaoqiang Zheng, Yonghui Wu, Barret Zoph, Ekin Cubuk, Jonathan Shen, Tianqi Chen, and Vijay Vasudevan for helpful discussions and inspirations; and the larger Google Brain team.