Research | Open Source

DLRM: An advanced, open source deep learning recommendation

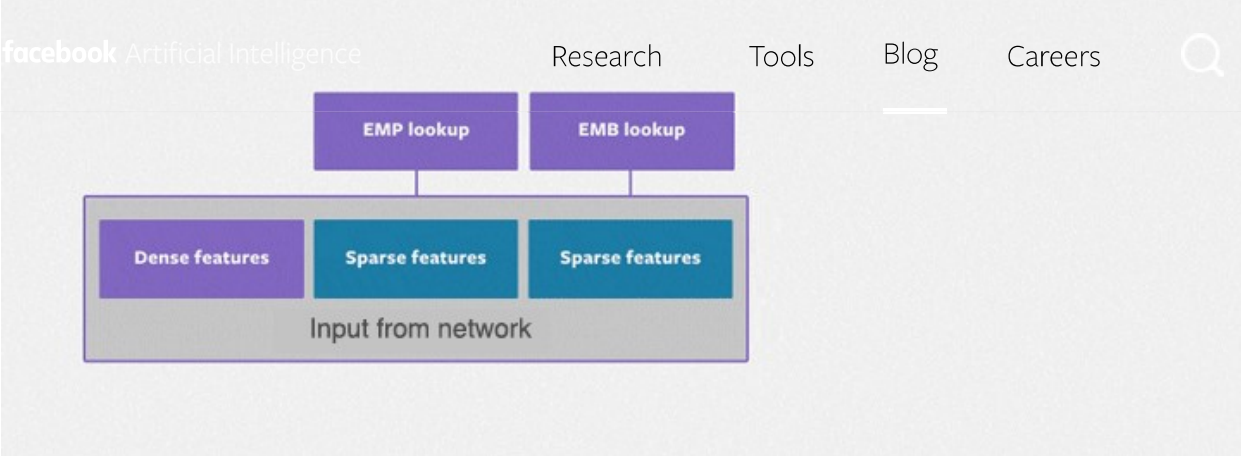
With the advent of deep learning, neural network-based personalization and recommendation models have emerged as an important tool for building recommendation systems in production environments, including here at Facebook. However, these models differ significantly from other deep learning models because they must be able to work with categorical data, which is used to describe higher-level attributes. It can be challenging for a neural network to work efficiently with this kind of sparse data, and the lack of publicly available details of representative models and data sets has slowed the research community's progress.

To help advance understanding in this subfield, we are open-sourcing a state-of-the-art deep learning recommendation model (DLRM) that was implemented using Facebook's open source PyTorch and Caffe2 platforms. DLRM advances on other models by combining principles from both collaborative filtering and predictive analytics-based approaches, which enables it to work efficiently with production-scale data and provide state-of-art results.

By releasing this model and also detailing its functionality here and in this accompanying , we hope to help the community find new ways to address to the unique challenges presented by this class of models. We also hope to encourage further algorithmic experimentation, modeling, system co-design, and benchmarking. This in turn will lead to new models and more efficient systems that can provide more relevant content to people using a wide range of digital services.

Understanding the DLRM model

In the DLRM model, categorical features are processed using embeddings, while continuous features are processed with a bottom multilayer perceptron (MLP). Then, second-order interactions of different features are computed explicitly. Finally, the results are processed with a top MLP and fed into a sigmoid function in order to give a probability of a click. (The schematic representation of this model was discussed in detail in this



The DLRM model handles continuous (dense) and categorical (sparse) features that describe users and products, as shown here. It exercises a wide range of hardware and system components, such as memory capacity and bandwidth, as well as commmunication and compute resources.

Benchmarking and system co-design

The open source implementation of DLRM can be used as a benchmark to measure:

- The speed at which the model (and associated operators) performs.
- How various numerical techniques affect its accuracy.

This can be done on different hardware platforms, such as the BigBasin AI platform.

The DLRM benchmark provides two versions of the code, one using PyTorch and another using Caffe2 operators. Further, a variation of this implementation is also provided using Glow C++ operators. (The code for each varies slightly to adapt to the specifics of each framework, but the overall structure is similar.) The implementations allow us to contrast the Caffe2 framework with the PyTorch framework, as well as with the Glow implementation currently focused on accelerators. Perhaps most important, we can then highlight the best features from each, which could be incorporated into a single framework in the future.

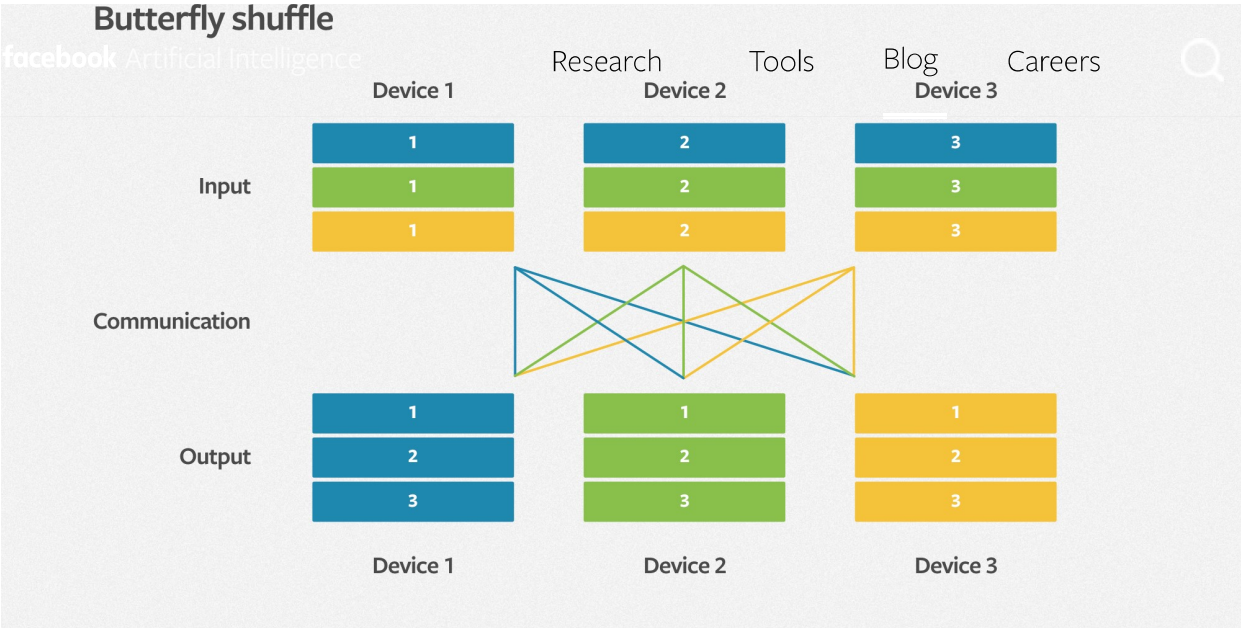
Big Basin features a modular, scalable architecture. The open source design is available

The DLRM benchmark supports generation of both random and synthetic inputs. There are many reasons to support custom generation of indices corresponding to categorical features. For instance, if our application uses a particular data set but we would not like to share it for privacy reasons, then we may choose to express the categorical features through distributions. Also, if we would like to exercise system components, such as studying memory behavior, we may want to capture the fundamental locality of accesses of original trace within synthetic trace.

Moreover, services across Facebook use a variety of personalized recommendation models depending on the use case. For example, to achieve high performance at scale, services could parallelize inferences across different platforms by batching inputs and colocating multiple models on a single machine. In addition, the variety of servers present in Facebook's data center introduces architectural heterogeneity, ranging from varying SIMD width to different implementations of the cache hierarchy. The architectural heterogeneity exposes additional hardware-software co-design and optimization opportunities. (An in-depth analysis on the system architectural implications for Facebook's neural recommendation systems is available in this paper .)

Parallelism

As shown in the first diagram above, the DLRM benchmark is composed of compute-dominated MLPs as well as memory capacity limited embeddings. It is therefore natural to rely on data parallelism to improve performance of MLPs and model parallelism to address the memory capacity requirements of embeddings. The DLRM benchmark provides a parallel implementation that follows this approach. (It is functionally correct but not currently optimized for performance.) We note that during interactions it requires an efficient all-to-all communication primitive, which we refer to as a butterfly shuffle. It shuffles the results of an embedding lookup of an entire minibatch on each device into parts of a minibatch of embedding lookups on all devices. This is shown in the graphic below, where each color denotes a different element of the minibatch and each number denotes the device and the embeddings allocated to it. (We plan to optimize the system and share a detailed performance study in a future blog post.)



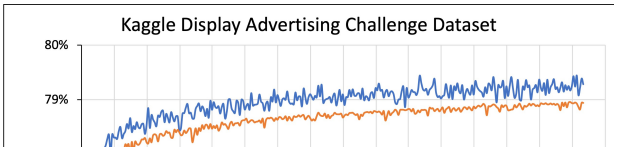
This graphic shows DLRM's butterfly shuffle.

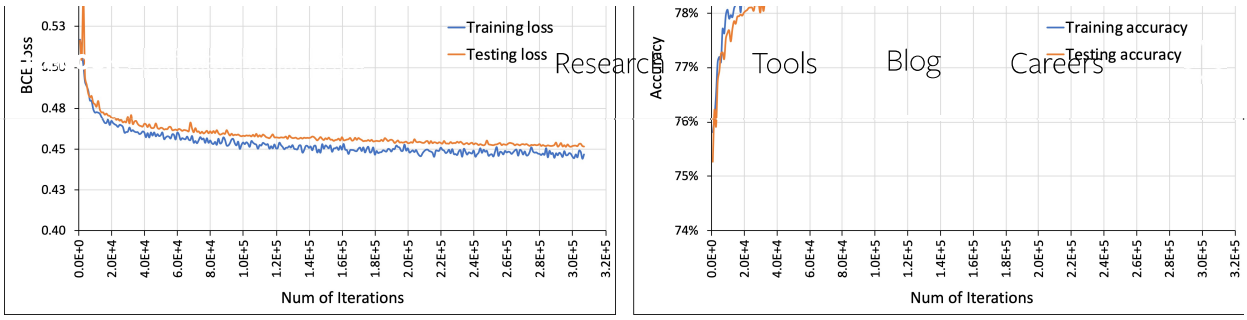
Modeling and algorithmic experimentation

The DLRM benchmark is written in Python to allow for a flexible implementation, where the model architecture, data set, and other parameters are defined by the command line arguments. DLRM can be used for both inference and training. In the latter case, the backward-pass operators are added to the computational graph to allow for parameter updates.

The code is self-contained and can interface with public data sets, including the Kaggle Display Advertising Challenge Dataset. This particular data set contains 13 continuous and 26 categorical features, which define the size of the MLP input layer as well as the number of embeddings used in the model, while other parameters can be defined on the command line. For example, the results of DLRM run with the following command line arguments are shown in the charts below.

```
python dlrm_s_pytorch.py --arch-sparse-feature-size=16 --arch-mlp-bot="13-512-256-64-16" --arch-mlp-top="512-256-1" --data-generation=dataset --data-set=kaggle --processed-data-file=./input/kaggle_processed.npz --loss-function=bce --round-targets=True --learning-rate=0.1 --mini-batch-size=128 --print-freq=1024 --print-time
```





The chart on the left shows training and testing binary cross entropy loss. The chart on the right shows training and testing accuracy.

The model runs on a realistic data set that allows us to measure the accuracy of the model, which is especially useful when experimenting with different numerical techniques and other models. We plan to provide additional in-depth analysis of effects of quantization and algorithmic experiments on this model in forthcoming work.

By providing a detailed description of the DLRM state-of-the-art personalized recommendation system and its open source implementation, we hope to draw attention to the unique challenges that this class of models presents. We look forward to working with others in the AI community to make advances in algorithmic experimentation, modeling, system co-design, and benchmarking. In the long term, developing new and better methods to use deep learning for recommendation and personalization tools (and to improve model efficiency and performance) will lead to new ways to connect people to the content that is most relevant to them.

Written by

Maxim Naumov

Research Scientist

Dheevatsa Mudigere

Research Scientist

Related posts

Open-sourcing Ax and BoTorch: New AI tools for adaptive experimentation

May 01, 2019

Recap of first-ever Glow Summit

April 04, 2019

Aroma: Using machine learning for code recommendation

April 04, 2019

Related Tags

Research ML Applications Open Source

Research

Research

Areas

Developers

Developer

Tools

Blog

Read Blog

Careers

Join Our

Team