TorchRec: a PyTorch domain library for recommendation systems

DMYTRO IVCHENKO, Meta AI, USA
DENNIS VAN DER STAAY, Meta AI, USA
COLIN TAYLOR, Meta AI, USA
XING LIU, Meta AI, USA
WILL FENG, Meta AI, USA
RAHUL KINDI, Meta AI, USA
ANIRUDH SUDARSHAN, Meta AI, USA
SHAHIN SEFATI, Meta AI, USA

Recommendation Systems (RecSys) comprise a large footprint of production-deployed AI today. The neural network-based recommender systems differ from deep learning models in other domains in using high-cardinality categorical sparse features that require large embedding tables to be trained. In this talk we introduce TorchRec, a PyTorch domain library for Recommendation Systems. This new library provides common sparsity and parallelism primitives, enabling researchers to build state-of-the-art personalization models and deploy them in production. In this talk we cover the building blocks of the TorchRec library including modeling primitives such as embedding bags and jagged tensors, optimized recommender system kernels powered by FBGEMM, a flexible sharder that supports a veriety of strategies for partitioning embedding tables, a planner that automatically generates optimized and performant sharding plans, support for GPU inference and common modeling modules for building recommender system models. TorchRec library is currently used to train large-scale recommender models at Meta. We will present how TorchRec helped Meta's recommender system platform to transition from CPU asynchronous training to accelerator-based full-sync training.

CCS Concepts: • **Information systems** → **Recommender Systems**; **Open source software**; • **Computing methodologies** → *Massively parallel algorithms.*

Additional Key Words and Phrases: recommender systems, information retrieval

ACM Reference Format:
Dmytro Ivchenko, Dennis van der Staay, Colin Taylor, Xing Liu, Will Feng, Rahul Kindi, Anirudh Sudarshan, and Shahin Sefati. 2022. TorchRec: a PyTorch domain library for recommendation systems. In . ACM, New York, NY, USA, 3 pages. https://doi.org/XX.XX

1 TORCHREC

By mid-2020, the PyTorch team received a lot of feedback that there has not been a large-scale production-quality recommender systems package in the open-source PyTorch ecosystem. Starting from Meta's stack, we began modularizing and designing a fully-scalable codebase that is adaptable for diverse recommendation use-cases. Our goal was to extract parts of Meta's software stack to simultaneously enable creative exploration and scale. We hope this package opens a dialogue and collaboration across the RecSys community, starting with Meta as the first sizable contributor.

1.1 Scaling Performance at Meta

PyTorch introduced support for accelerators, namely GPUs, as well as improved developer experience and productivity by providing the eager Python-based API. The PyTorch ecosystem still lacked the model parallelism to fully push full-sync training limits. The support for large-scale production-grade recommender systems motivated the development of the TorchRec library.

TorchRec is now used to train large recommender models at Meta with models exceeding 3×10^{12} parameters. TorchRec helped Meta to transition from CPU asynchronous training to accelerator-based full-sync training, dramatically increasing FLOPs and model sizes resulting in significant model quality gains. Because TorchRec is PyTorch-based, it provides better modeling flexibility and machine learning engineers are now able to use a free-form Python code to write training programs.

In addition to training, TorchRec supports inference model preparation and optimization, e.g. quantization and pruning as these operations can be applied to a sharded version of a model. TorchRec also supports GPU inference. It provides low-latency communications primitives and optimized quantized compute kernels. TorchRec is run on `torch::deploy` (https://pytorch.org/docs/stable/deploy.html) to ensure low latency and high throughput while preserving Python model authoring flexibility.

The ability to use Python throughout the whole production life-cycle i.e. training, inference model preparation and inference serving, enables fast modeling explorations and streamlines research exploration to production workflows at Meta.

1.2 TorchRec Library building blocks

TorchRec includes a scalable low-level modeling foundation alongside rich batteries-included modules. We initially target "two-tower" ([1], [2]) architectures that have separate submodules to learn representations of candidate items and the query or context. Input signals can be a mix of floating point “dense” features or high-cardinality categorical “sparse” features that require large embedding tables to be trained. Efficient training of such architectures involves combining data parallelism that replicates the "dense" part of computation and model parallelism that partitions large embedding tables across many nodes. The TorchRec library includes:

- Modeling primitives, such as embedding bags and jagged tensors, that enable easy authoring of large, performant multi-device/multi-node models using hybrid data-parallelism and model-parallelism.
- Optimized RecSys kernels powered by FBGEMM , including support for sparse and quantized operations, fused sparse optimizers, GPU memory caching and other advance performance optimizations.
- A sharder which can partition embedding tables with a variety of different strategies including data-parallel, table-wise, row-wise, table-wise-row-wise, and column-wise sharding, leveraging intra and inter-host hardware topologies.
- A planner which can automatically generate optimized sharding plans for models evaluating numerous plans and choosing the most performant one based model and hardware characteristics.
- Pipelining to overlap dataloading device transfer (copy to GPU), inter-device communications (input dist), and computation (forward, backward) for increased performance.
- GPU inference support including low-latency high-performant quantized kernels and communications.
- Common modules for RecSys, such as models and public datasets (Criteo and Movielens).

The code is available at https://github.com/pytorch/torchrec.

ACKNOWLEDGMENTS

We would like to thank our collaborators who contributed to developing the TorchRec library: Renfei Chen, Shabab Ayub, Joshua Deng, Zheng Yan, Ning Wang, Ying Liu, Leon Gao, Liang Luo, Hongbo Zhang, Xing Wang, Donny Greenberg, Alex Morgan Bell, Bernard Nguyen and many more.

2 SPEAKER BIO

Dmytro Ivchenko is a software engineer at Meta. Dmytro has joined Meta over 6 years ago. He is a part of PyTorch team focusing on recommender systems. Prior to Meta Dmytro worked at LinkedIn where he wrote personalized search engine used to search for people, jobs and other site content.

Dennis van der Staay is a software engineer at Meta. Dennis joined Meta 2 years ago. He is part of the PyTorch team focusing on recommender systems. Prior to Meta, Dennis worked in ML engineering at Branch International, an Andreessen Horowitz backed startup.

REFERENCES

[1] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. 2020. DLRM: An advanced, open source deep learning recommendation model.

[2] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Ajit Kumthekar, Zhe Zhao, Li Wei, and Ed Chi (Eds.). 2019. *Sampling-Bias-Corrected Neural Modeling for Large Corpus Item Recommendations.*