Reproducing EL-Rec

Henry Sneed

Abstract

The size of the embedding tables Deep Learning Recommendation Models (DLRMs) rely on continues to expand exponentially [Z+20]. This motivates an exploration into methods to further compress embedding tables and to optimize the DLRM training process so that their resource consumption does not follow the same trend. EL-Rec, the Efficient Large-Scale Recommendation Model Training via Tensor-Train Embedding Table by Zheng Wang et al. proposes a training system that harnesses the Tensor-train (TT) technique to optimize TT decomposition based on key computation primitives of embedding tables [W+22]. EL-Rec introduces an index reordering technique and pipeline training paradigm to eliminate communication overhead, which they claim achieves a 3x speedup over state-of-the-art DLRM frameworks and handles the largest publicly available DLRM dataset with a single GPU. This paper outlines the effort to reproduce their results and assess its potential applicability for additional datasets. The results of this reproduction largely validate their original experimental findings and the techniques they propose. Based on these findings, further research into the use of Tensor Train embedding tables and the application of EL-Rec to other datasets is recommended.

1 Introduction

As streaming has surpassed all other forms of consumption of commercially produced music, video and streaming services have deployed deep learning recommendation models to make increasingly effective content recommendations to users based on their listening and viewing habits [IFP23]. Youtube, Tiktok, Spotify, and others attract and maintain subscribers by serving items to individual users from massive pools of potential content [Tob23]. The efficacy of the models directly impacts the platform's performance as better recommendations can lead to more satisfied users and less user churn. Discovery also impacts content creators, as the algorithms and parameters guiding the recommendation models dictates which content will surface for which users [Die14]. However, the scale required to service an effective model is massive and requires tremendous computing and financial resources. Deploying effective models with fewer resources will be critical to future DLRMs both to save resources of large incumbents and enable smaller projects without immense resources to deploy DLRMs. EL-Rec improves on the Tensor Train technique applied to TT-Rec and enables the embedding table to be compressed significantly [Y+21].

Conference'17, July 2017, Washington, DC, USA
© 2023 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 Methodology

I attempted to reproduce the results on a single A10 GPU with 24GB GPU memory, 200 GiB RAM, 1.4TiB SSD, and 600 GB/s HBM. This failed during the data processing step as the GPU ran out of memory. After refactoring some of the data processing scripts, the index bijection scripts failed for similar reasons. Ultimately, I used 4 V100 16GB NVLink GPUs running Ubuntu 20.04, containing 8 vCPUs, 24 GB RAM, and a shared SSD of 700GB on Fluidstack.io to replicate the same hardware used in the original paper. On the V100s, the data processing completed without error. However, the index bijection steps exceeded resource limits and the scripts were killed by the operating system. To resolve this for the Avazu and Kaggle datasets, I decreased batch size from 512 to 64. To resolve it for Terabyte, I decreased batch size from 65536 to 256, segmented the length of the element list so that it was processed in chunks, and added GPU detection so that a terabyte index bijection script could run on any available GPU instead of GPU 0. This allowed me to run 4 scripts simultaneously.

3 Results

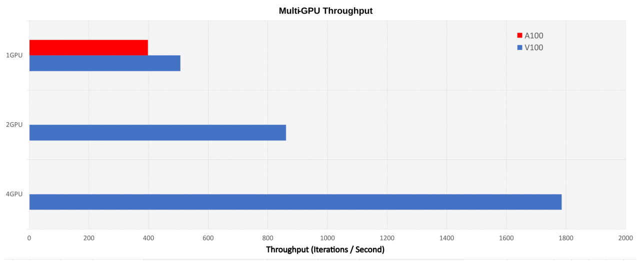

Figure 1. Large embedding table training throughput.

With some exceptions, the results were consistent with those reported in the original paper. As demonstrated in Figure 1, EL-Rec achieves significant increases in throughput with additional GPUs and an R squared value of .9379. Compared on each dataset in 2, EL-Rec's training throughput was even more pronounced than the original experimental results.

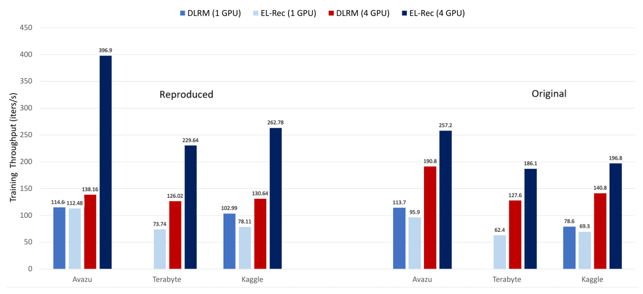

Figure 2. EL-Rec training throughput under multi-GPU setting.

Table 1 corresponds to Table IV in EL-Rec. It demonstrates that EL-Rec achieves the same results as the original paper within a 1% margin of error. I was unable to complete the training of FAE within 8 hours and without exhausting memory so the results are not included. Similarly, Terabyte could not fit on Facebook DLRM and the process was repeatedly killed by the operating system. One significant difference from the original findings is the use of Pipeline training.

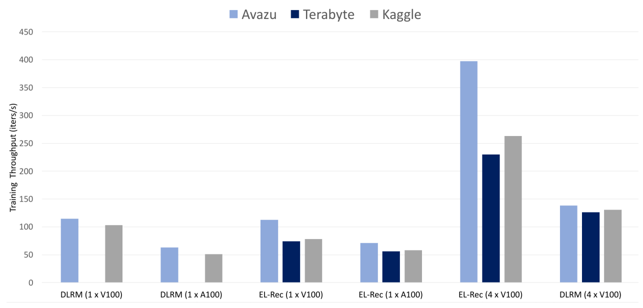

Figure 3. EL-Rec training throughput A100 vs V100.

When tested against the A100 80GB SXM4 NVLink as shown in Figure 3, the throughput of the V100 consistently outperforms and demonstrates a significant increase in throughput as more V100 GPUs are introduced.

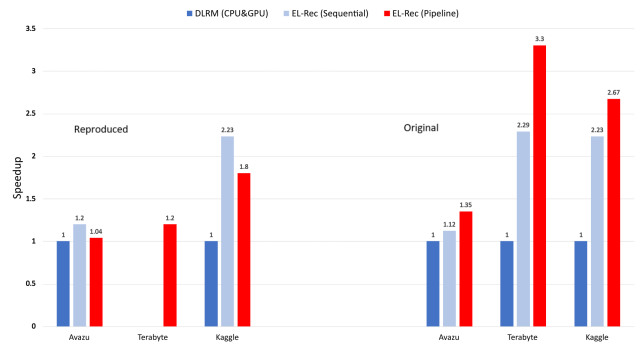

Figure 4. Pipeline training speedup (×) over sequential training.

As shown in Figure 4, pipeline training does not result in speedup greater than sequential training. On both Avazu and Kaggle datasets, EL-Rec sequential outperforms pipeline training. However, results on terabyte could not be completed as the training script was killed by the operating system on DLRM and EL-Rec sequential. The total speedup achieved by both pipeline and sequential EL-Rec during training is also smaller than the original results, with EL-Rec pipeline achieving an average 1.35 speedup with Terabyte compared to an average speedup of 2.11 in the original experimental results.

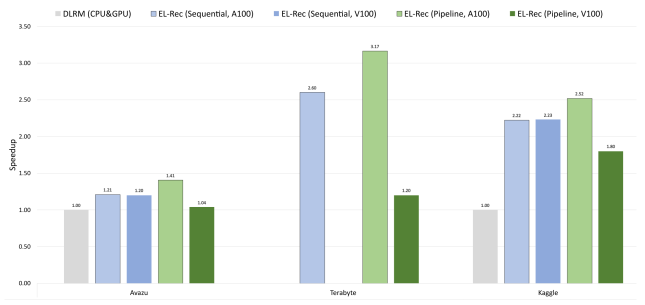

Figure 5. EL-Rec training throughput under multi-GPU setting.

On the A100, the use of pipeline parallelism shown in Figure 5 significantly improves speedup over the V100 results. This could be due to the greater high bandwidth memory of the A100, the A100's Ampere architecture, and the improved efficiency of its tensor cores. It is also notable that Sequential EL-REC ran to completion on the A100 for the Terabyte dataset while it failed on the V100.

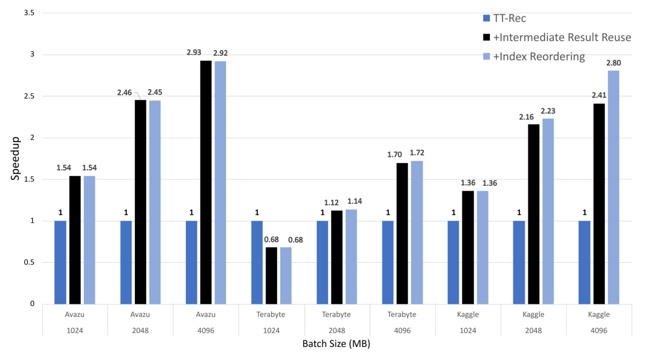

Figure 6. . Speedup (×) breakdown of the TT table lookup optimization.

The addition of Intermediate Result Reuse and Index Reordering (6) show similar speedups from the introduction of both. This differs slightly from their results, which showed the addition of index reordering improving speedup slightly more than intermediate result reuse on batch sizes of 4096MB. However, these results still demonstrate the efficacy of both techniques on all datasets as both achieve greater than 2.9 speedup on Avazu, 1.70 on Terabyte, and 2.2 on Kaggle with batch size 4096MB. The test of removing optimizations demonstrates in Figure 9 nearly identical results to the original findings, as the removal of gradient aggregation has an out-sized impact compared to the removal of the other optimizations. When instead the technique is tested by inclusion, the addition of gradient aggregation and index reordering shown in Figure 10 re-affirm the results demonstrated in Figure 9.

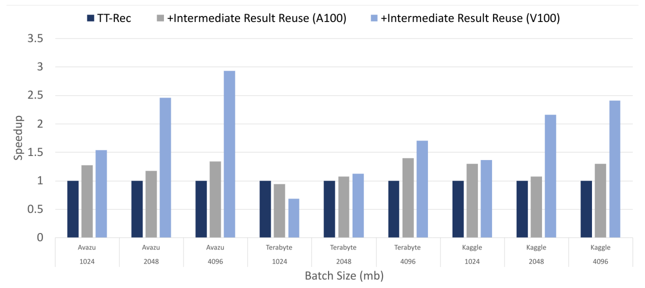

Figure 7. . Speedup (×) breakdown of the TT table lookup optimization.

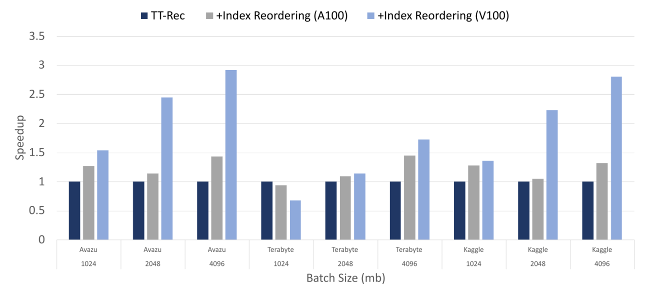

Figure 8. . Speedup (×) breakdown of the TT table lookup optimization.

Figures 8 and 7 compare the addition of intermediate result reuse and index reordering on V100 and A100 GPUs. The results demonstrate the the V100 achieves superior speedup to the A100 for both the introduction of index reordering and intermediate result reuse for all tests except for Terabyte at a 1024 MB batch size.

When completing the multi-gpu training, the program entered into a state of deadlock when run with more than 1 GPU. To resolve this, I added a barrier to synchronize all processes before logging the output and I altered the logging of results to force the contents of the buffer to write out immediately.

4 Analysis

The significant speedup and throughput achieved by EL-Rec demonstrates the efficacy of the approach. By compressing the embedding table using tensor train decomposition,

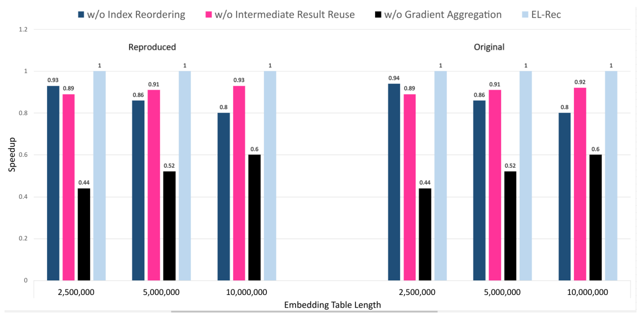

Figure 9. Eff-TT table optimization breakdown.

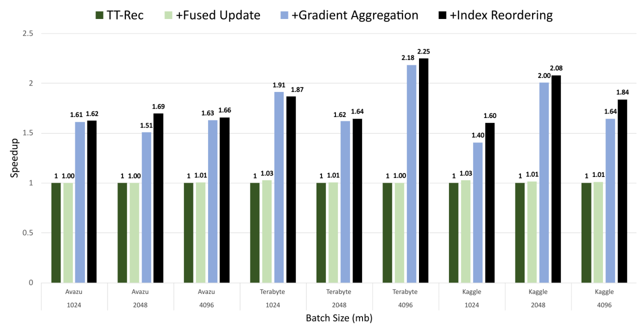

Figure 10. Speedup (×) breakdown of the TT table backward optimization.

Table 1. COMPARISON OF PREDICTION ACCURACY.

Model / Dataset	Avazu	Criteo Terabyte	Criteo Kaggle
DLRM	83.53	81.96	78.53
TT-Rec	83.51	81.86	78.51
FAE	83.53	81.94	78.52
EL-Rec	**83.51**	**81.90**	**78.50**
DLRM	83.54	—	78.45
TT-Rec	83.47	82.02	78.71
FAE	—	—	—
EL-Rec	**83.52**	**82.03**	**78.651**

each GPU can hold the embedding table in memory. This decreases overhead from data movement and enables data parallel training instead of model parallel training. Additionally, the techniques do not demonstrate any meaningful reduction in prediction accuracy relative to the other models. One practical disadvantage of the current project is that adapting it requires additional time, money, and debugging to correctly pre-process the data on the GPUs. In total, it requires the refactoring or creating at least 9 scripts to perform the data processing and index bijection steps required to fully process the data and perform index reordering. The results also demonstrate that the deployment of EL-REC to a more powerful A100 GPU does not improve throughput or any other measured metric. This implies a sub-optimal utilization of the A100 architecture as no modifications were made to adapt it specifically for the A100. However, in the original proposal of an efficient computing framework using the Tensor-train technique, the authors did not indicate that EL-REC is specifically optimized for the V100 and the findings are proposed for GPUs generally. For this reason, a custom GPU adaptations should not necessarily be required to see performance improvements, and the EL-REC results obtained on the A100 do still outperform those of the other DLRMs tested.

5 Conclusion

The results demonstrate the effectiveness of EL-Rec and justify future implementation of the model and its techniques to minimize resource overhead from massive embedding tables and enable data parallel training. Although replicating the results was not trivial and encountered many bugs, memory problems, and resource utilization errors, the experiments prove that the task was worth the effort and warrants future work adapting the model and training it on new datasets such as the Spotify Sequential Skip dataset. [S.And].

References

[Die14] Sander Dieleman. Recommending music on spotify with deep learning. sander.ai/2014/08/05/spotify-cnns.html, 2014.

[IFP23] IFPI. Global music report 2023 $\rightarrow$ state of the industry, 2023. Accessed on: April 28, 2023.

[S.And] Spotify Technology S.A. Datasets - spotify research. https://research.atspotify.com/datasets/, n.d.

[Tob23] Josh Tobin. Monolith: The recommendation system behind tiktok. gantry.io/blog/papers-to-know-20230110/, 2023.

[W+22] Zheng Wang et al. El-rec: Efficient large-scale recommendation model training via tensor-train embedding table. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, 2022.

[Y+21] Chunxing Yin et al. Tt-rec: Tensor train compression for deep learning recommendation models. *ArXiv:2101.11714 [Cs]*, 2021.

[Z+20] Weijie Zhao et al. Distributed hierarchical gpu parameter server for massive scale deep learning ads systems. *ArXiv:2003.05622 [Cs, Stat]*, 2020.

6 Appendix

Source Code: https://github.com/henryjsneed/reproducing-el-rec