Title: Fast, safe and cost efficient model development to increase consistent experimentation throughput

Authors: Ziv Gome ziv.gome@taoola.com, Charles Sitbon charles.s@taboola.com

Abstract:
Taboola is a world leading content recommendation system matching each user in a given context with the best items out of millions. The algorithms group grew rapidly (x5) in the last few years to support the increasing demand for new types of models and model complexity. Currently, just in the personalisation stage, we have multiple different model types (Organic, CTR, CVR, Calibration, RTB, e-Commerce and more) that are optimizing towards different KPIs and competing over the same resource pool. Model complexity has grown as well, both in neural network architecture and also in operations flow, which is required to support knowledge sharing between models and multitask learning which are expected to drive further improvements in our models. Model improvement is measured both by offline metrics such as accuracy, and by online performance which has to be verified through extensive experimentation.

Experimentation requires web-traffic and time, which are the only two hard limitations we have. To utilize our traffic and time efficiently we have built a platform team focused on increasing experimentation throughput. In Taboola we identified four principles for a good platform which increases experimentation throughput: fast, safe, cost efficient and consistent. 1. Fast - reducing time-to-experiment by standardizing our machine learning pipeline to accelerate model design and training flow by making them both modular and configurable. 2. Safe - as models and training pipelines become modular and more complex, possible points of failure are multiplying as well. To make sure that the increased velocity will not increase downtime occurrences, we developed configurable verifications to make sure data integrity and the model's offline performance. 3. If fast and cost efficiency is achieved our system will need to support more models running in parallel which increase load on resources which requires resource management. Specifically, GPU usage which is handled completely internally. 4. Consistent - As mentioned, different models are aimed to optimize different KPIs which might create deadlocks in models rollout and increased agent costs, To reduce the risk of agents costs and facilitate efficient conclusion of experiments we standardized experiments evaluation across multiple KPIs.

Summary:
Taboola is a world leading content recommendation system matching each user in a given context with the best items out of millions. The algorithms group has experienced rapid growth in the last few years. While growing quickly, our models increased in variety, in architecture complexity, and in training flow requirements. We found that our models benefit from warm starting weights and teacher-student training paradigm. Furthermore, we believe that value can be gained by knowledge sharing between different model types which can be

achieved by weight sharing and multitask learning. The above significantly increases the need for more complex and flexible model architecture and training pipelines. For example to support multitask learning we need to allow learning from different datasets (of each task) and very flexible architecture as we need to propagate multiple losses to different parts of the network.

Naturally, we have accumulated technical debt while growing since often the quickest path to production was tailor-made code adjustments that were not easily transferable between different model types. As models grew in complexity and use cases multiplied, the technical debt increased our time-to-experiment (TTE), i.e. the time required to develop and train a new model.

Furthermore, in Taboola we rely heavily on online experimentation to make sure that a new model is benefiting different business KPIs before a new model is rolled out and replaces a current running model. Offline improvements guide the research and help identify issues in the models, but they are often inconsistent with online performance. This makes experimentation crucial and the more we can experiment the faster we can improve and bring value. To utilize our traffic and time efficiently we have built a platform team focused on increasing experimentation throughput via investment in infrastructure. The infrastructure is designed to make model development fast, experimentation safe, resources efficient and evaluation consistent across the group.

Increasing experimentation throughput can be achieved by reducing time-to-experiment, identifying problematic models and converging experiments quickly. To reduce TTE and support present and future initiatives, such as multi-task learning and self-supervised learning, we enforced uniformity and modularity in our ML code. The main goal of the standardization and modularization was to move the entire model and flow design out from the code and into configuration files.

While our model addresses different business problems and we experiment with very different network architectures - there is still much in common in the features available, and in surrounding utilities. We found that configuration-based modeling ends up with reduced TTE as compared to coding-based modeling. We therefore built a modeling framework which supports the superset of the modeling operations and handles common use cases such as reporting metrics, but leaves the specific architecture for each engineer via configuration. By doing that, we can plug in a new experiment in minutes and know it is safe and all the basics are properly handled. It is important to notice that we enforce uniformity by requiring new possible model architectures that require coding to be available via configuration for all other users. This avoids the fallback to inefficient coding practice of working with similar but tailored made functions per model use case instead of developing a completely shared infrastructure.

In addition, as our flows became more sophisticated - with the introduction of transfer learning techniques, multi-task learning and more - we faced new challenges in the machine learning flows. These new challenges, increased TTE and harmed stability. We needed a new approach, a modular approach to building pipelines - one that acknowledges Model Lineage as first class citizen and treats datasets, configurations, output models and code as versioned elements and traces the workflow end-to-end. Here two, we chose a configuration-based approach to speed up our experimentation.

Since models become ever more dependent on other models and data preprocessing to be completed correctly and independently, it was critical to add verifications post each step. Specifically, we added verifications after the preprocessing of each data subset to make sure that the data distribution didn't change significantly compared to previous days/weeks. Similarly, we verified, after each model training, that the model characteristics are similar to previously trained models. Failure in the verifications can either alert or fail the step, which would fail dependent steps until it's resolved. An added benefit of adding the verifications was direct. Bad data or poorly trained models are costly for Taboola. Even small inefficiencies can cause significant losses due to the high scale, which naturally pushed the group to invest in and embrace these protections.

The above reduced our TTE while allowing ever more flexible architectures and workflows. On the flip side, the demand on our resources, such as GPU, grew with the reduced TTE and team growth. Our infrastructure was already built for scale, but the economics of just buying more hardware changed. We therefore took a fresh look at our GPU utilizations and were able to increase utilization by 400%. First, we applied ML optimizations - such as batch size and learning rate changes. Then, through careful profiling - which led us to share GPUs by multiple training processes. Next, we added software around the different GPU consumers such that we can push it to the limit - separating priorities for example between daily production-oriented training, and heavy offline grid-search HPT process. At the end, we were able to meet both production needs, while encouraging the teams to experiment with the GPU resources as much as they needed safely.

Finally, to ensure consistency between experiments and teams we standardized the experiments evaluation process. To achieve this, we are maintaining python notebooks for experiment evaluation so that different tests could be evaluated under the same methodology. Using these standard notebooks solved an unexpected issue as well as we learned that different models that optimize for different KPIs are affecting other KPIs as well. The solution was straightforward by consolidating different notebooks into one that considers all relevant KPI simultaneously. The decision to roll out a model will be made by the team weighing in the different effects measured in the notebook. Another important side benefit is that these notebooks are a tool for knowledge preservation, as it always implements the state-of-the-art evaluation method that is used by someone in the group.