Name: Patrick Watters

Degree: PhD, Computer Science and Engineering

Email: pwatters@unr.edu

Institution: University of Nevada, Reno

Mentors: Lei Yang, Feng Yan

Research Title: BatchFlow: Coordinated Sharing of Mini-Batches Across Heterogeneous Deep Learning Training Jobs

Abstract:

Training deep learning (DL) models requires enormous computational throughput, which is typically provided by specialized hardware accelerators such as GPUs and TPUs. To keep these accelerators fully utilized, input pipelines must deliver training data at least as fast as the accelerators can consume it. When the pipeline cannot sustain this rate, accelerators become idle, increasing training time and cost. As accelerator performance continues to advance, providing high-throughput and reliable data delivery has become increasingly critical. Traditional DL systems run input data pipelines and model training on the same machine, tying training performance to the node's compute, memory, and storage bandwidth. To overcome these limitations, recent systems execute data pipeline operations on remote resources, allowing them to scale independently of training. However, when many training jobs run concurrently, providing sufficient pipeline capacity to satisfy their aggregate demand becomes difficult, often resulting in overprovisioning for some jobs and bottlenecks for others. In this work, we present BatchFlow, a disaggregated data pipeline service that coordinates the preparation, caching, and delivery of training data across heterogeneous DL jobs in a shared environment. Unlike conventional approaches, in which each job independently fetches and preprocesses data, BatchFlow serves preprocessed data to all training jobs. It controls when and which data items are delivered to each job based on their consumption rate and latency requirements, while enforcing a training data order that preserves correct model convergence and accuracy. The system leverages real-time feedback from active training jobs to make informed decisions about cache allocation, eviction policies, and data scheduling. This coordination mechanism enables BatchFlow to minimize system-wide data stalls while optimizing overall training throughput across concurrent jobs.