TAuto: An Automated ML Pipeline Profiling and Optimization Framework

High Performance Machine Learning Final Project Proposal

Jonathan S. Kent
Fu Foundation School of Engineering and Applied Science
Columbia University
jsk@cs.columbia.edu

Simay Cural
Fu Foundation School of Engineering and Applied Science
Columbia University
sc5559@columbia.edu

***Abstract*—This project proposes the development of TAuto, an automated machine learning optimization framework that performs systematic profiling and parameter tuning to maximize both training efficiency and model performance. Unlike traditional AutoML systems that focus primarily on architecture search, TAuto targets the optimization of training hyperparameters including data loading configurations, batch sizes, optimizer selection and parameters, learning rate schedules, and PyTorch compilation options. The framework will conduct intelligent search across these parameter spaces while generating comprehensive performance reports and visualizations to identify bottlenecks and document improvements. By automating this typically manual tuning process, TAuto aims to reduce the time data scientists spend on optimization while providing deeper insights into training performance dynamics.**

***Index Terms*—AutoML, hyperparameter optimization, performance profiling, PyTorch, training efficiency**

I. Goal/Objective

The primary goal of this project is to develop a comprehensive AutoML framework that automatically optimizes machine learning training pipelines for both efficiency and effectiveness. Specifically, TAuto will:

- Systematically profile PyTorch model training to identify performance bottlenecks across data loading, preprocessing, model computation, and backward passes
- Implement intelligent search strategies to optimize training parameters including number of DataLoader workers, batch size, optimizer algorithms, learning rate schedules, and torch.compile configurations
- Automatically tune hardware-specific parameters to maximize GPU utilization and minimize memory bottlenecks
- Generate detailed performance reports with comparative metrics and visualizations that illustrate before/after improvements
- Provide practitioners with deeper insights into performance tradeoffs that affect training time and model quality

Unlike many existing AutoML solutions that focus on neural architecture search [1], TAuto targets the often overlooked but critically important training process optimizations that can significantly reduce training time and improve model convergence. The specific capabilities and tests described in this proposal are aspirational, rather than hard commitments; functionality will be developed as time allows.

II. Challenges

Several technical challenges must be addressed to successfully implement the proposed framework:

- **Complex parameter interdependencies**: Many optimization parameters have coupled effects (e.g., batch size affects both memory usage and learning dynamics), requiring careful coordination of multiple parameters simultaneously.
- **Efficient search space exploration**: The high-dimensional parameter space makes exhaustive search prohibitively expensive, necessitating intelligent sampling and search strategies.
- **Hardware-specific optimizations**: Different GPU architectures respond differently to optimization techniques, requiring adaptable strategies that work across hardware configurations.
- **Balancing training speed vs. model quality**: Optimizing solely for throughput may negatively impact model convergence or final accuracy, requiring multi-objective optimization approaches.
- **Accurate and lightweight profiling**: Collecting detailed performance metrics without significantly impacting the training process itself presents a methodological challenge.
- **Optimization stability**: Ensuring that optimizations remain stable across different runs and don't introduce training instability or convergence issues.

III. Approach and Performance Optimization Techniques

TAuto will implement a systematic approach to optimization with the following components:

A. Profiling and Analysis

The framework will integrate PyTorch Profiler [2] and Nvidia Nsight [3] to collect detailed performance metrics at each stage of the training pipeline, including:

- Data loading and preprocessing time
- Forward pass computation time by layer

- Backward pass and gradient computation overhead
- GPU memory usage patterns
- CPU-GPU transfer bottlenecks
- GPU utilization and kernel efficiency

B. Optimization Techniques

Based on profiling results, TAuto will apply appropriate optimizations from a comprehensive toolkit:

- **Data pipeline optimization**:
  - Automatic tuning of num_workers in DataLoader
  - Prefetch factor optimization
  - Pin memory and non-blocking transfer configuration
  - Dataset sharding and caching strategies
- **Training optimization**:
  - Batch size optimization accounting for memory constraints
  - Dynamic batch sizing based on gradient accumulation
  - Automatic mixed precision (AMP) configuration
  - Gradient checkpointing for memory-intensive models
- **Optimizer selection and tuning**:
  - Comparative analysis of optimizer algorithms (SGD, Adam, Adadelta, etc.)
  - Optimizer-specific parameter tuning (momentum, weight decay, etc.)
  - Learning rate schedule optimization (linear, cosine, one-cycle, etc.)
- **Compiler optimizations**:
  - torch.compile mode selection (default, reduce-overhead, max-autotune)
  - Just-in-time (JIT) compilation configuration
  - Operator fusion opportunities
  - Backend selection (inductor, nnc, aot_eager)
- **Memory optimizations**:
  - Activation checkpointing strategies
  - Optimal use of CPU offloading for large models
  - CUDA memory allocation tuning

C. Search Strategy

Rather than random or grid search, TAuto will employ:

- Bayesian optimization with Gaussian Processes [4] for global parameter search
- Tree-structured Parzen Estimator (TPE) approaches for conditional parameter spaces
- Multi-objective optimization to balance training speed and model quality

IV. Implementation Details

A. Hardware

The development and evaluation of TAuto will utilize:

- Primary development: Nvidia RTX 3090 GPU (local)
- Cloud evaluation: Google Cloud with appropriate GPU instances
- Additional testing on Columbia's Terremoto cluster (if available)

B. Software Stack

- PyTorch 2.0+ for model implementation and training
- PyTorch Profiler for detailed performance analysis
- Nvidia Nsight Systems for GPU kernel-level profiling
- Weights & Biases for experiment tracking and visualization
- Custom profiling and optimization modules built on top of existing tools

C. Datasets and Models

The framework will be evaluated on standard vision and language tasks:

- Vision models: ResNet-50, EfficientNet, Vision Transformer on ImageNet or CIFAR-100
- Language models: BERT, RoBERTa, or smaller transformer models on GLUE tasks

V. Demo Planned

The final demonstration will showcase:

- A live demonstration of the TAuto framework optimizing a model training pipeline
- Interactive dashboard visualizing performance metrics before and after optimization
- Comparative analysis of different optimization strategies across hardware configurations
- Case studies showing significant speedups achieved on specific models
- Documentation of optimal configurations discovered for reference architectures

The demo will highlight both the performance improvements achieved and the insights gained about model training dynamics through the profiling process.

VI. Expected Outcomes

By the end of this project, I hope to deliver:

- A functional TAuto framework capable of optimizing PyTorch training pipelines
- Comprehensive documentation of optimization techniques and their effectiveness
- Quantitative performance improvements demonstrated across multiple model architectures
- Insights into hardware-specific optimization strategies and their relative impact
- A GitHub repository with the complete framework, usage examples, and benchmark results

Additionally, the project aims to advance understanding of how different optimization techniques interact and their relative importance in modern deep learning training pipelines.

References

[1] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.

[2] PyTorch Team, "Pytorch profiler," https://pytorch.org/tutorials/recipes/recipes/profiler_recipe.html, 2023, accessed: 2025-03-07.

[3] NVIDIA Corporation, "Nvidia nsight systems," https://developer.nvidia.com/nsight-systems, 2023, accessed: 2025-03-07.

[4] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, vol. 25, 2012.