We thank all the reviewers for their time. In what follows, reviewer comments are italicized and proceeded by our response in blue.

Reviewer #3

We thank the reviewer for the helpful references. Importantly, we note that the SVM GPU-speedup paper by Catanzaro et al. is for nonlinear SVMs, which are outside the class of fast, intricate algorithms considered in the paper (see lines 30-51), like TRON.

Does that mean there is a trade-off between memory/computation and communication. If z is huge or extremely dense in some future applications, is this a potential issue? The reviewer is correct, there is a trade-off between memory/computation and communication for intricate algorithms like TRON–which contain highly sequentially dependendent variables–stated on lines 42-44 and 173-177 (and discussed at length in Sections 4.1-4.2, and lines 240-246). z is a 1D array of length equal to the number of instances, so it will fit in GPU memory even for extremely large problems (as demonstrated in Figure 2). *Probably not appropriate to just report the speedup given the comparison is based on different platforms.* All experiments were conducted on the same platform (stated and detailed on lines 248-249) to ensure a fair comparison. Furthermore, we note that wall clock time (as used in the paper) is the standard metric for speed in machine learning papers, e.g., PyTorch, LIBLINEAR, Cocoa, etc.

The huge speedup of GPU over CPU is not new. In general, and in the context of deep learning speedups, we agree. However, GPU speedups for many of the fastest machine learning classification and regression algorithms (such as those in scikit-learn, as well as LIBLINEAR, commonly considered the fastest ML package for such problems) are nonexistent, as discussed on lines 30-39 of the main paper. As stated in Section 1.2.13 of the current scikit-learn documentation:

> "Outside of neural networks, GPUs don't play a large role in machine learning today, and much larger gains in speed can often be achieved by a careful choice of algorithms."

The goal of the paper is to show that, contrary to this common conception, GPUs may effectively speedup extremely intricate, fast machine learning algorithms, such as TRON.

Reviewer #2

We thank the reviewer for the helpful suggestions on how to improve the readability of Sections 4 and 5, which we will add to the paper.

The claim for tenfold speedup on sparse representation dataset training is not well justified compared to state-of-the-art multi-threaded TRON solver. The tenfold speedup is relative to the most widely-adapted, standard LIBLINEAR package (which is optimized for single-threaded use), as detailed in the description of Figure 1. The average reduction in overall runtime compared to the multithreaded CPU optimized TRON is 65.2%, as stated in the bold text on lines 60-61.

Disabling parallelization of Percolator's outer most cross-validation might have worsened the training efficiency of TRON-SVM-CPU. The opposite was observed; as noted on lines 268-269, as the number of threads grows large, thread scheduling overhead diminished overall multithreaded performance. With parallelization of Percolator's outer most cross-validation enabled, the overhead of scheduling threads with two nested thread-pools significantly diminished TRON-SVM-CPU and L2-SVM-MFN performance for modest-to-large numbers of threads. In order to fairly measure the performance of these multithread-optimized solvers for large numbers of threads, Percolator's outer-most cross-validation was disabled.

With reasonably larger number of threads, the speedup claims should be more modest... there exist other platforms with even higher number of threads... It would be interesting to see at what thread count the benefits over multi-threaded TRON start to become negligible. In practice, solely multithreaded speedups often do not scale linearly in the number of increasing threads, largely due to thread-scheduling overhead (see the above discussion). Thus, even higher thread counts do not necessarily provide more multithreaded performance, as can be seen in Figure 1, where purely multithreaded performance peaks before utilizing the maximum number of threads used in the paper (48) on four of the six datasets. Note that while TRON-LR-MIX similarly displays diminished performance for increased thread counts (as noted on lines 268-720), the relative GPU-speedup gains never become negligible.

Reviewer #1

We thank the reviewer for the detailed comments not discussed herein, which we will incorporate into the paper. In addition to lines 30-39, we will include further discussion of previous GPU-offloading work (which, other than non-linear kernels and deep learning, are lacking for fast, intricate classification and regression learning algorithms).

How generalizable are the proposed optimizations for GPU offloading to algorithms other than LR and SVM? The general optimization principles described are applicable to any highly specialized, CPU-centric algorithm, e.g., any of the extremely fast algorithms listed on lines 35-36. The specific GPU-optimizations for TRON are generally applicable to arbitrary non-linear losses–such as those widely used in deep learning–through popular automatic differentation packages such as PyTorch and Tensorflow, as stated on lines 150-153 and 313-317.

Reviewer #4

It is unclear how the proposed method could benefit the deep learning community. As stated on lines 150-153 and 313-317, the detailed TRON GPU-optimizations are readily applicable to arbitrary deep learning losses through popular automatic differentation packages, such as PyTorch and Tensorflow. *The time complexity and memory complexity are not specified.* TRON is a quasi-Newton method, and thus enjoys quadratic convergence while using only linear memory (detailed in the supplementary). Further discussion of complexity and tradeoffs will be included in the paper.