We thank all reviewers for the constructive feedback. We are encouraged by the acknowledgement of novelties of our formulation [R1, R2, R3, R4], the comprehensiveness of our studies [R2, R3], and potential industry impact [R1, R2, R4]. We address specific comments below and will incorporate them to the updated version.

[R1] **Performance analysis of 3 models; Modeling z_i as a sequence is unnatural.** Many DL models contain only few non-sequential *variable-to-variable* connections, e.g. only in embedding variables of BERT and NCF. Feeding z_i into RNN following the data flows presented in $\mathcal{G}$ suffices for the RNN to capture the variable-to-variable structure. For GAT, we observe large discrepancy between training and validation accuracy; it achieves lower-than-expected performance due to the scarcity of training data and overfitting. Besides, the linear model cannot leverage raw features.

[R1, R2, R4] **Experiment w/ more clusters and baselines.** Throughout our study, we have tested on 14 different clusters (Table.3 of the supplementary). *We managed to find better-than-hand-optimized strategies on 13 configurations*, for the 3 models used in paper, except on B4 where we matched `Horovod`. A fair comparison to winning entries in MLPerf is challenging due to the lack of access to similar setups (e.g., TPUs, P100-equipped clusters, their code). We compare to SoTA PS implementation BytePS [27] (despite of a different distributed runtime) on (BERT-large, B): AutoSync finds *1.3x faster* strategies in 200 trials. We'd be happy to submit an entry to MLPerf using our clusters.

[R1, R3] **Test set of Table.1 & details on simulator training.** For the end-to-end results (Fig.2), we train simulators using runtime data collected on-the-fly during trials, strictly following Algorithm.1 in the supplementary. For the ablation studies (Table.1-3), we split pre-collected data at auto-optimization of a specific domain (e.g., (NCF-dense, A)) into train/val/test at 70%/15%/15%, respectively, and report the ranking accuracy on the test split (averaged over 3 runs). More information about the dataset is in §11.2 of the supplementary. Training RNN simulators in all settings use Adam with 1e-3 lr, decayed by 0.3 at the 80th/160th epoch, for 200 epochs. We clipped the gradient norm to 0.25.

[R1, R3, R4] **Using ranking accuracy in Table.1; no measurement of real runtime estimation**: We train simulators with an augmented ranking loss (see L187-194) instead of regression loss (to estimate runtime), as (1) auto-optimization only requires predicting whether a strategy is better than others; (2) we internally tried predicting runtime and found it resulting in worse end-to-end optimization performance; (3) ranking loss can transfer better across domains.

[R1] **Connection to EASGD and [Chen et al.]**. EASGD can be thought of as an optimizer (equivalents are Adam/Adagrad). So far, AutoSync optimizes system throughput without interfering with optimization results, and lets users decide which optimizer. In future work, we can add an "optimizer" aspect (including EASGD) to the strategy space, and extend Eq.1 to optimize model convergence. Similar reasoning applies to the technique in [Chen et al.].

[R2] **Considering Bayesian optimization (BO)**. Applying BO for strategy optimization is theoretically feasible, but faces two main obstacles: the strategy feature ($z^{pre} \oplus z^{raw}$) is very high-dimensional; Encoding variable-specific choices of multiple synchronization aspects, it has variable-length. These prevent an easy adoption of BO in our case.

[R2] **Improvement on presentations, and standard deviation (std) in results**. Thanks for the suggestions! We reinspect the results: the std of 3 runs for (1) ranking accuracy results (Table.1-3) are under $\pm 2\%$; (2) hit rates (Fig.2) are in $\pm 3\% - 7\%$; (3) improvements over baseline at trial 200 (Fig.2) are in $\pm .04 - .1$. We'll extend results from 3 to 5 runs, add std in tables and plots, add hit rates in Fig.3, and fix presentation issues raised by R2.

[R2] **Computation of hit rates; decreased hit rates in some plots**. For each run, the hit rate at trial k is calculated as percentage of strategies explored up to trial k that are above hand-optimized baselines. We calculate the hit rates independently for each run and report the average percentage at k. The hit rates plateau or decrease majorly due to randomness at exploration. 42%/28% hit rates for random exploration without a simulator (`AutoSync(-s)`) is higher than our expectation; we'll adjust the tone to be more objective.

[R2] **Context of $2200 AWS credits saving.** Training BERT-large on Wikipedia+BookCorpus needs ≈2M iterations to report results w/ bs $= 128$ [7, 23]. Suppose we rent cluster B ($3.912/h/node) and use the (maximally doable) per-GPU bs $= 8$, training using the hand-tuned `PS` (Fig.2 bottom-right plot, 1.56s/iter) will spend $2M * 1.56 * 4 * 3.912/3600 =$ $13561.6. `AutoSync` takes 50 trials (2.5K iterations) to find a 1.2x faster strategy than `PS`. Hence, the net saving is approximated as: $13561.6 - 2M * 1.56/1.2 * 4 * 3.912/3600 - 2.5K * 1.56 * 3.912 * 4/3600 \approx$ $2243, per job.

[R1, R2] **Measures and implication of accuracy results in Table.3**. Sorry for the confusion. To clarify, the test accuracy in Table.3 is obtained by transferring a simulator trained on source domain data to infer target domain data *without any additional training*. Simulators with closer accuracy to those in Table.1 and 2, which are trained using target domain data, are more competitive to guide the auto-optimization (normally >70% suffices to guide the exploration).

[R3] **Transferability between different clusters**. Due to the feature design (see §3.3), the simulators do transfer between cluster platforms (see §5.3). Table.3 and Fig.3 show proven results (in both ranking accuracy and end-to-end results) that the simulator can transfer between cluster A and B with different setups *without additional training*.

[R4] **Visualizing strategies**. We inspected strategies over 20 $(\mathcal{G}, \mathcal{D})$ settings and found: (1) PS is dominant in dense BERT-large; other settings have mixed PS/collective. (2) Inconsistent with [18], sparse variables on cluster B prefer being synchronized via AllGather than PS on many models. (3) "Partition-then-allreduce" patterns (contrast to "merge-then-allreduce" pattern in existing systems) are observed in BERT-large and NCF-dense on cluster B. We'll add a section covering qualitative analysis of these strategies.