We thank the reviewers for their detailed comments and suggestions. We will address the minor presentation comments in the paper, and will address the remaining comments below.

Reviewer 1: For the first reviewer, we would like to emphasize that while the case of $p = 2$ is central in practice, the case of $p = 1$ is also very important, and in fact is often more desirable than $p = 2$ for its robustness properties. For instance, in the rank regression estimator also considered in this paper, it is necessary that regression is carried out with respect to absolute deviation ($p = 1$), and not $p = 2$. For $p < 2$, our results demonstrate an even larger improvement over the prior work, which had runtime that grew super-linearly in $\sum_i \text{nnz}(A_i)$. Moreover, in several interesting cases where the vector b itself is a Kronecker product (such as all-pairs regression), our $p < 2$ algorithms do not even require the nnz(b) runtime, thus matching our results for $p = 2$. Additionally, regarding the remark made by reviewers 1 and 2 about a comparison of our work with SGD, we note that getting provable relative-error regression guarantees with first order methods such as SGD would indeed require a condition number assumption on the matrices $A_1, \ldots, A_q$ (note that the condition number of $\otimes_{i=1}^q A_i$ is the product of the condition numbers of A_i). Our algorithms, interestingly, do not depend at all on the condition number of the A_i's, and make no assumptions on these matrices.

Reviewer 2 In response to the reviewer's worries about practicality and applicability of our results, we would like to reiterate that Kronecker products do arise naturally in many applications. Significantly, Kronecker product regression comes up very frequently in tensor related-problems, which are relevant whenever the data has more than two associated dimensions. For instance, tensor regression is used centrally in [1] as well as in [2]. Kronecker products naturally arise when solving matrix equations, which show up in many different settings (see e.g. [4]). For instance, solving $AX - XB = C$, which is the Sylvester equation, can be written as $(A \otimes B)\text{vec}(x)$ =vec(c). Additionally, Kronecker products of more than two matrices arise when looking at partial differential equations such as a Poisson equation. Another important motivation to study Kronecker product regression is that a common way of solving low rank approximation (LRA) is via alternating minimization. Namely, if you want a LRA to a matrix $A \in \mathbb{R}^{n \times n}$, you can first fix some $U \in \mathbb{R}^{n \times k}$ and solve for the $V \in \mathbb{R}^{k \times n}$ that minimizes $\|A - UV\|$ via regression. Once you have a V, then you solve for the best U given the V, and so on. The same procedure is used for third-order tensor low rank approximation, where you have $A \approx U \otimes V \otimes W$ and use regression to solve for one of the factors at a time with the other two fixed, which is now a Kronecker product regression problem.

With regards to dependence on d in Theorem 3.1, we are indeed missing a factor of d in the definition of m – we will fix this typo. With respect to the reviewer's comment on the run-time of this theorem, we first note that the main computation taking place is the leverage score computation from Proposition A.3. For a constant number of input matrices q (as is generally the case in applications), the term $d^{O(1)}$ in Proposition A.3 to approximate leverage scores is $O(d^3)$. The remaining computation is to compute the pseudo-inverse of a $d/\epsilon^2 \times d$ matrix, which requires $O(d^3/\epsilon^2)$ time, so the additive term in Theorem 3.1 can be replaced with $O(d^3/\epsilon^2)$. This implies that even for $q = 2$ matrices, if $n > d^2$, the size of the Kronecker product will be $\Omega(d^4) \times d$, which is larger than the runtime of our algorithm.

Additionally, the regime where $n \gg d$, known as the over-constrained regime, is motivated by many common situations in practice, and as a result has been the focus of a large amount of algorithmic research over the past decade (as cited in our submission). The special case of Kronecker products is no different, and also frequently arises in the $n \gg d$ setting, such as [3] which considers very rectangular matrices (see their experiments in Table 5.1). This setting addresses another comment of the second reviewer regarding the relative sizes of nnz(b), and $\sum_i \text{nnz}(A_i)$. Even in the above example with n only slightly larger than d, we would have nnz$(b) = \Omega(d^4)$ but our runtime would be $O(d^3)$. The above shows that our algorithms become much more practical than both traditional linear algebraic algorithms and SGD (which can have a bad dependence on the condition number) even in a mildly over-constrained setting.

Reviewer 3 With regards to the comment on the impact of our algorithms in applications, we first point to the above paragraphs which discuss settings in practice where our algorithms have been proven to perform substantially better than prior techniques. In addition, we remark that our experiments section demonstrates strong improvements on the run-time of our algorithms when compared to the prior work of Diao et al. This suggests that our algorithms are even faster than the provable guarantee of $O(d^3)$, which may be pessimistic because it is a worst-case theoretical bound.

[1] O. A. Malik and S. Becker. Low-rank tucker decomposition of large tensors using tensorsketch. In *Advances in Neural Information Processing Systems*, pages 10096–10106, 2018.

[2] O. A. Malik and S. Becker. Fast randomized matrix and tensor interpolative decomposition using countsketch. *arXiv preprint arXiv:1901.10559*, 2019.

[3] G. Pisinger and A. Zimmermann. Linear least squares problems with data over incomplete grids. *BIT Numerical Mathematics*, 47(4):809–824, 2007.

[4] V. Simoncini. Computational methods for linear matrix equations. *SIAM Review*, 58(3):377–441, 2016.