We thank all the reviewers for their time and for their thoughtful comments. We agree with all that was said and will do our best to address it in the final version. Most importantly, we will incorporate the discussion below that aims to comment on a concern raised by reviewer #3 and to answer a question asked by reviewer #1. Moreover, we will add the suggested references, and add a "broader impact" section[1] (We apologize for not realizing that we are required to explicitly say the impact even for a theoretical work like ours.)

Is grammar-compression useful for vectors and matrices encountered in ML? (a concern raised by reviewer #3)

We prove that grammar compressions are harder to analyze (without decompression) than simpler ones like RLE. It is natural to wonder: "*Are grammar compressions really more effective than simpler methods like RLE for the data encountered in ML?*" In principle, of course they could be, but are they *actually*? Because, if not, it is not clear why this community should care about the complexity of grammar-compressed linear algebra. Indeed, in many applications, the vectors are very sparse, and RLE is sufficient to get a very strong compression; zip or another, more specialized, grammar-compression could reduce the size further, but the benefit might be negligible.

In the submission, we have only commented on this question in passing. We mentioned that the source file of our paper compresses from 10KB to 4KB with zip, while RLE has negligible effect, and then moved on, supposing that this anecdote extends to many other scenarios that are popular in ML. The reviewer rightfully objects that sequences behave differently from vectors, and therefore this anecdote is not fully convincing.

As the reviewer suggests, it would have been much better to give empirical evidence: showing real-world datasets of vectors and matrices that are of interest to the ML community where grammar compressions make a difference. Fortunately, such a test was already performed in the "Compressed Linear Algebra" paper [3]. In Table 1 the authors present the compression rates achieved with Gzip for five data sets of vectors that are not very sparse (Higgs, Census, Covtype, ImageNet, Mnist8m). For example, the Census [2] data set has sparsity of 0.43 but a much higher compression rate of 0.0584 is achieved with Gzip. We will highlight this reference in the paper, and we will also perform our own experiments with more datasets and include an explicit comparison with RLE. We plan to pick data sets with large vectors that are not sparse containing Boolean, integer or real values, e.g. ISOLET [2]. From a quick test that we performed we see compression rates of about 0.2 even when the sparsity is 0.9.

(True, even in these cases, there may be another scheme that is simpler than zip that achieves strong compression and is easier to compute over; but this is exactly the message of our paper: the "easy solution" of just using any grammar-compression has serious limitations, and the research should be directed towards task-oriented compression schemes. To quote reviewer #4: "*In this sense, proving a limitation of those models will greatly influence future research programmes.*")

On the weaker result that follows from [1] (an answer for reviewer #1, we will elaborate on this in the paper)

First, let us remark that the grammar-compressed vector inner product problem is NP-Hard, but we do not think this result is of much practical interest because it is only meaningful when n is logarithmic in N, meaning that the vectors are exponentially compressible. On the other hand, our results address the regime where the vectors are only polynomially compressible, and in fact, our result for inner product holds when $n = N^{1/3}$. The previous work [1] had a different, less efficient reduction showing a lower bound for Disjointness. The lower bound that follows is $N^{1/2}$ instead of our $N^{2/3}$, and it only holds for more compressible vectors where $n = N^{1/4}$ rather than our $n = N^{1/3}$. Technically, the novelty is that we manage to encode two sets (A and B) into one vector of length mU rather than m^2U. This new construction is crucial for the extensions we show in the paper. In particular, we do not see how to prove any lower bound for matrix-vector inner product without building on this new construction.

We wish to sincerely thank the reviewers and the PC again for their time and help in improving the quality of this work.

References

[1] A. Abboud, A. Backurs, K. Bringmann, and M. Künnemann. Fine-grained complexity of analyzing compressed data: Quantifying improvements over decompress-and-solve. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 192–203, 2017.

[2] D. Dua and C. Graff. UCI machine learning repository, 2017.

[3] A. Elgohary, M. Boehm, P. J. Haas, F. R. Reiss, and B. Reinwald. Compressed linear algebra for declarative large-scale machine learning. *Commun. ACM*, 62(5):83–91, 2019.

[1]The section will express that the broader impact is to inform algorithm design for compressed linear algebra, which can lead to faster algorithms for a variety of tasks on large data sets. The ethical consequences depend on the specific application. We do not see any inherently new concerns raised by our results, beyond those that follow from faster algorithms.