We thank the reviewers for their thoughtful comments. We will address their comments in the revised version of the paper. We believe our contributions will be interesting to the ML community for several reasons:

- General, ***automatic*** methods to reduce memory consumption for large deep learning models are critical. To our knowledge, techniques up to this point have been mostly focused on special case models. In practice, getting extremely large models to run on GPUs or TPUs is a time-consuming task, requiring expert knowledge of the problem.
- As Reviewer 1 pointed out, it is not always clear that the increase in running time is worth the savings in memory. But consider the case of training BERT on a TPU pod, which takes around 4 days. If that model doubled in size, it could take weeks of engineering work to get it to run. Our algorithm offers the ability to greatly reduce the memory with no human intervention, at the cost of an extra week of computation; our personal experience suggests this is a tradeoff that many would happily make. Of course, there are cases on both sides of this tradeoff, but having a reasonable starting point is valuable in itself.
- We provide a formalization of the problem with rigorous guarantees. As several reviewers noted, this is a useful contribution. In fact, while using treewidth to improve rematerialization has been suggested in an informal setting before (the tensorflow gradient checkpointing blog post), we show later in this rebuttal that the primary theoretical arguments therein are incorrect. In addition, we feel that our formalization will make the problem more attractive to the algorithmic community within NeurIPS, which would help garner additional insights. (For example, the fact that low pathwidth graphs have better theoretical guarantees is quite surprising, even if it doesn't immediately lend itself to practical algorithms.)

We now address a few of the specific reviewer concerns.

Minimizing schedule length with memory constraint. We have heuristics based on the TwRemat algorithm that, for a given memory limit, work by stopping the recursion early when the current set of nodes to compute can be scheduled without rematerialization under the memory limit. While minimizing the execution time of the schedule given a hard constraint on memory is an ideal version of our problem, we felt that including those heuristics would muddy the clarity of our results. However, in the revised version of this paper we will include a more thorough discussion of this. From a theoretical viewpoint, no algorithm with runtime sub-exponential in treewidth can yield a provable approximation guarantee on the schedule length (under common complexity assumptions). We will also add a discussion regarding this hardness result in the revised version.

Comparisons with model parallelism. While it is true that model parallelism is also a way to reduce the memory requirements of training, rematerialization and model parallelism are orthogonal techniques. One can simultaneously apply the two for even greater memory savings; hence our work is complementary to model parallelism. In addition, our work makes rematerialization easy to automate, whereas most model parallelism is achieved by careful hand-crafting. Since our rematerialization algorithms operate automatically on all networks, the neural network designer can obtain the benefits of memory reduction for "free" without any specialized optimization.

Addressing the blog post. Unfortunately the algorithm suggested by the TensorFlow checkpointing blog post is incorrect. That post draws on Courcelle's theorem (namely, every graph property definable in the monadic second-order logic of graphs is linear-time decidable on bounded treewidth graphs) to characterize the problems that become tractable for graphs of small treewidth. However, rematerialization is known to be PSPACE-complete [1]. Writing down a monadic second-order logic formula for rematerialization would imply that the problem is in PH (the polynomial hierarchy), and hence we would arrive at the surprising result that PSPACE collapses to PH. Thus, it is unlikely that we could apply Courcelle's theorem to rematerialization.

Comparisons with previous work. We thank the reviewers for pointing out the references that we missed and will be sure to include a comparison with those approaches in the revised version. We feel that it's more accurate to avoid the term "gradient checkpointing" here, because our technique can be applied even to inference graphs, but we will definitely include a discussion of terms.

References

[1] John R Gilbert, Thomas Lengauer, and Robert Endre Tarjan. The pebbling problem is complete in polynomial space. In *Proceedings of the 11th annual ACM Symposium on Theory of Computing*, pages 237–248. ACM, 1979.