We thank the reviewers for their insightful comments and encouraging feedback. We hope that the comments raised are addressed adequately below.

**Relation to PowerSGD and GradZip (R3, R4)**

PowerSGD (Vogels et al., 2019) and GradZip (Cho et al., 2019) are two similar algorithms for *centralized* distributed optimization that are also based on low-rank approximation. These methods approximate the average gradient update across workers. This global averaging operation requires a fully-connected network and prevents straightforward application of these methods in a decentralized setting.

The key difference in the proposed PowerGossip algorithm is that it instead approximates *model differences* between connected workers. PowerGossip effectively instantiates multiple independent copies of PowerSGD; one for each pair of connected workers. In the special case of a fully connected network, PowerGossip would use a different projection vector for each pair of workers, rather than a global one as in PowerSGD.

**Relation to other algorithms for decentralized learning (R1)**

We compare our work to other compression algorithms for decentralized learning (Koloskova et al. 2020, Tang et al., 2019). While those algorithms also support low-rank compression, PowerGossip especially leverages the linearity and contractivity of the operation by directly compressing model differences. This avoids the introduction of additional hyperparameters that plagues prior work.

**Bounded variance assumption (R2)**

The relaxation of the bounded variance assumption follows easily using standard techniques (using e.g. (Koloskova et al. 2020) as pointed out by the reviewer). We chose to use a stronger assumption to ease presentation since we believed that such a relaxation yields no new insights. We will be happy to extend our analysis to the relaxed assumption setting.

**Varying the compression rank (R4)**

Similarly to PowerSGD, PowerGossip supports ranks larger than 1. A Rank-n compression step requires the same data transfer as n rank-1 steps, and those alternatives work equally well (see Appendix F). We opt for multiple rank-1 steps as it avoids an expensive orthogonalization operation (Vogels et al., 2019). There could be a benefit of larger ranks in latency-bound settings. We can highlight this trade-off in the manuscript.