We would like to thank all three reviewers for the close look given at the paper and the issues that were raised. You will find below a detailed answer to each comment.

About acceleration (REVIEWER #1) In the experimental section, we did not use acceleration for PPRS by referring to Theorem 4 of Sec. 4.2 (the non-convex case) that does not use acceleration (*i.e.*, $\mu = 0$). For the sake of completeness, we also tried the accelerated version of the algorithm (with $\mu = 0.99$), which did not improve the results. Hence, to improve the readability of the figures, we decided not to show them and keep the (non-convex) theoretical version of the algorithm. We will add a comment about this in the final version.

Time computation in Fig. 3 (REVIEWER #1) The term "Time (normalized)" in the x-axis was indeed misleading. We thus decided to replace it with "Parallel computation time" and state in the caption: "The parallel computation time of each algorithm is computed through the formula $T(K + \Delta - 1)$, where T is the number of iterations of an algorithm, K its number of samples per iteration and Δ the depth of the computation graph.". The time axis was indeed correctly scaled to take into account that PPRS with $K = 200$ and $\Delta = 20$ has a computation time per iteration approximately 10 times bigger than any other method.

Objectives of the experimental section (REVIEWER #2) The main focus of the paper is to provide a theoretical analysis of this optimization problem, and experiments are only provided to underline the analysis. More specifically, our aim is to show that some optimization problems are sufficiently difficult and non-smooth to require specialized optimization methods (*e.g.*, random smoothing and PPRS). The practical impact and efficiency of PPRS and the selection of its hyperparameters in practice is a very interesting line of research that we leave for future work.

Distribution mechanism and relation to related works (REVIEWER #2) While pipeline-parallel optimization is an abstraction that may be used for many different distribution setups, one of the main application is DL architectures distributed on multiple GPUs (with memory limitations and communication bandwidths) by partitioning the model. GPipe (with GD/AGD) may be seen as a special case of PPRS, with $\gamma, K = 0$ (*i.e.*, no randomized smoothing).

PPRS on smooth functions and relevance to ML/DL(REVIEWER #2 AND #3) Note that, technically, neural networks have non-smooth losses as soon as ReLU activation functions are used. However, the losses tend to be relatively smooth in practice. As is, we do not believe that PPRS would improve on GD/AGD for smooth problems (initial experiments seemed to support this intuition), and is more relevant for particularly hard optimization problems. However, this does not mean that it could not be useful for ML, as: 1) many practical aspects of PPRS can yet be improved. This should be the focus of follow-up work, as our work is of a more theoretical nature (see discussion in the paragraph on the objectives of the experiments). 2) One may argue that the lack of efficient optimization methods for hard optimization problems has drawn the ML community to more simple/smooth problems (e.g. using a quadratic loss instead of the hinge loss, or particular DL architectures that are easy to learn), and easily available optimizers for more difficult problems may enable researchers to discover novel architectures that might have not been learnable using standard GD/AGD (see also the last sentence of Sec. 6).

Robustness of the experimental results (REVIEWER #3) To assess the robustness of the approach with respect to different optimization problems, we ran the attack on 100 images from CIFAR10 with the same set of hyperparameters. Figure 1 shows the mean (and its standard deviation) of the minimized loss during training. The conclusions remain unchanged.

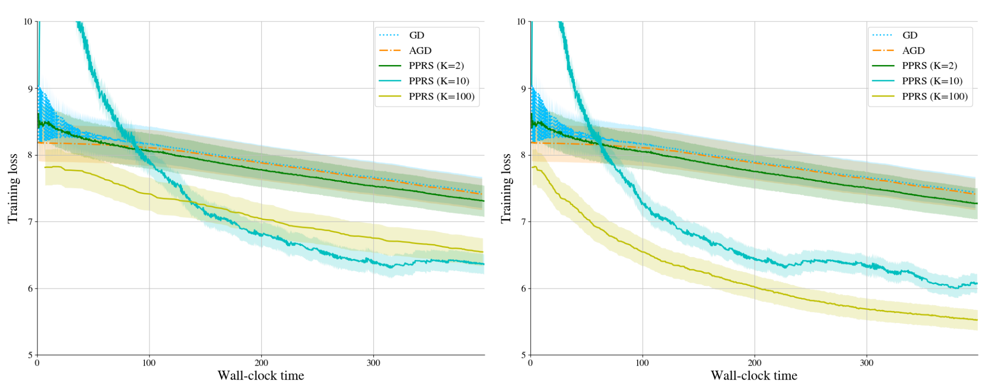

Figure 1: Figure 3 from the paper to be updated.