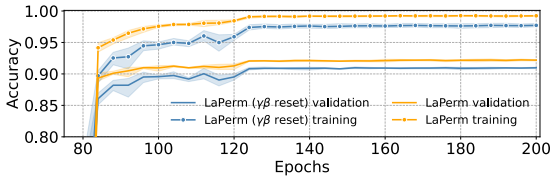
We greatly appreciate the reviewers for the time and expertise they have invested in the reviews. This essay focuses on addressing the main concerns raised by each reviewer. Due to the page limit, we will not be able to mention and answer all the questions. However, we would like to thank the reviewers for every observation and suggestions for revision and clarification and will make sure to improve on every aspect of our paper.

1. The meaningfulness of the analysis of "weight profiles". (Reviewer 1) In Section 2, our focus is on the statistical similarity between weight vectors within a trained weight matrix and it is crucial for later developing the insight: reordering of neuron connections can result in learning. A majority of the main results in our paper are obtained using *uniform random distribution* as opposed to the gaussian-like distribution seen in well trained neural networks. Moreover, we have conducted experiments (not mentioned in the paper) using different initializations, e.g., uniform, with different scales and observed that the resulting weights to be still roughly gaussian with SoWP.

2. Training BatchNorm (BN) alone yields decent performance [1]. (Reviewer 1) To isolate the contribution of BN's scaling (γ) and shifting (β) terms from LaPerm reconnected networks, we propose to add experiments using the following "$\gamma\beta$ *reset*" training scheme into the appendix. Take ResNet50 as an example: we use BN as usual and update γ and β using the inner optimizer of LaPerm. At each synchronization, we reset γ and β to 1 and 0. We compare its performance with LaPerm (never reset γ and β) both using $k = 800$ (other experiment settings are the same as Section 5.4). We repeat the experiment three times and show their results on the right. We observe only roughly 1% decrease in the final accuracies when γ and β do not hold information. In contrast, [1] focuses on the expressive power of γ and β.



3. Section 5.2 lacks sufficient detail. (Reviewer 1) We would like to formalize our claim as following: We hypothesize that there are 2+1 dimensions of differences between initial and learned weights: D_0: Common distribution of weights, D_1: Locations of weights, and D_2: Deviations of exact learned weights from the best weights representable by well-chosen distribution (D_0) and locations (D_1). Each weight vector w_j is represented as $w_j = \sigma_j(\theta) + \delta_j$, where θ is a vector drawn from the common distribution (D_0), σ_j is a permutation operation (D_1), and δ_j is the remainder (D_2). This decomposition, however, is not unique, unless we put more restrictions on the choice of θ and σ_j. — The rest of the subsection will be revised accordingly.

4. LaPerm can be understood as a regularizer and is sensitive to k. (Reviewer 1) We agree that LAPerm has inherent regularization effects. But our claim — DNNs can be trained by only reconnecting random weights — would imply more than just regularization. Sections 5.4 and 5.5 provide partial results supporting our view. As for sensitivity of k, if we define being sensitive as "responding unpredictably to slight changes", then LaPerm is not quite sensitive. For example, we showed in Section 5.2 that the performance of LaPerm responded monotonically w.r.t. k and changing it from 500 to 2000 effects only 2% of the accuracies.

5. The paper does not seem focused enough. (Reviewer 1) Our focus is dual: (1) most of the information of DNNs is stored in the orderings of weights, (2) we can actually construct algorithms to find good orderings. We think both are equally important and cannot defocus one of them. We would like to improve the first paragraph of sections 2 to 5 in the final version of the paper to further emphasize this structure for clarity.

6. What is the reason to have multiple synchronizations in LaPerm? (Reviewer 2) LaPerm appears to need repeated synchronizations to find the optimal reordering. We conducted experiments on Conv4 under the same setting as in Section 5.2, but choose to synchronize only once at the end of the training, we obtained on average 13.8% (both validation and training) accuracies. We will merge this experiment into Figure 5 to stress this point.

7. Performance under a higher sparsity. (Reviewer 3) Results for higher sparsity for Conv4 on line 222 are reported in Figure 8 (2). Instead of only showing their final performance w.r.t. "% of weights" as we did in the paper, we will add their detailed training trend in the appendix. In Figure 7, we will show two more trends for $p = 5\%$ and 1%.

8. The costs of lookahead. (Reviewer 4) Except when k is extremely small, LaPerm, on average, has few extra computational overheads in addition to the cost of its inner optimizer. For example, running the scripts provided in the supplementary material on a Google Colab GPU runtime, synchronizing Conv13 (14.9M parameters) once takes around 200ms. For Conv13 in Figure 6, we needed to synchronize totally 125 times which only added 25s to the overall training time. Moreover, a larger k is observed to work well (e.g. Figure 5 in the paper) and thus should often be used. We will improve the main text in Section 4 to stress this point.

9. Response to details and clarity. (All Reviewers) We thank all the reviewers for the careful observations. We will improve the clarity of the figures. We will revise the main text and expand the paper's references and appendix according to the reviewers' suggestions in their "Correctness", "Clarity", and "Prior work" sections, especially focus on constraining our claims to only classification problems, stressing the architecture choices and their purposes, and fixing errors such as changing L2 weight decay to L2 regularization, fixing ambiguity and mistakes in notations and formulas. Due to limited time and space, we are unable to answer the comments about: trying other inner optimizers, conducting large-scale experiments, analyzing decision boundaries, theoretical justifications. We will pursue them in future works.

[1] Frankle et al. "Training BatchNorm and Only BatchNorm..." arXiv 2003.00152, 2020.