CL in Low-Rank Orthogonal Subspaces (Reviewer points are color coded **R1**, **R3**, **R4**, **R5**)

R1: Computation during training/ Wall-clock time: Our method requires one additional matrix multiplication in the second-last layer of the network in the forward pass and three additional matrix multiplications in the backward pass. On modern GPUs, the forward pass (inference time computation) is as efficient as standard training and backward pass adds a very small overhead to the overall training wall-clock time (seconds) which we record in the table below.

Dataset	Finetune	ER	AGEM	Ours	Dataset	Finetune	ER	AGEM	Ours
MNIST	280	309	522	317	CIFAR	529	850	1300	1117

Forgetting Results: Our forgetting results of EWC are compatible with that of Chaudhry et al, 2019 that the reviewer referred to (c.f. their Tab. 4 in the appendix). In fact, we used their codebase to develop our method and didn't modify the EWC routine. The only difference is that in our experiments, for fair comparisons, we initialize all the methods with orthogonal weights.

Yes 'k' is randomly picked.

The practicality of the multi-head approach: We believe that the jury is still out on what is the most practical setting for continual learning (single-/ multi-head, single-/ multiple-epochs) see `https://arxiv.org/abs/1909.08383` for a comprehensive survey. We don't claim a universal efficacy of our algorithm in all settings. We pick one setting that many recent works (especially the ones that the reviewer pointed out) adopted and provide our algorithmic contribution in that setting.

R3: Defining g_L: Yes, we are defining $g_L^t = P_t \frac{dl}{dh_L}$. The notation follows from lines 139-140 in the paper.

Figure 2: The reviewer made a very good observation. The inner products are not exactly zero because the weight matrices are not square and hence perfect orthogonalization is not observable. Regarding the ReLU's activation being in the linear region, see our response (Identity Jacobian) to R4.

Decreasing size of layers: Yes, the architectures we use follow the assumption that the layers are of decreasing size and for almost all the modern deep networks this assumption holds.

R4: Changing number of tasks: The way our method is presented in the paper, we do assume to know the total number of tasks 'T' beforehand but we don't consider this to be a critical limitation of our approach. One could dynamically resize the $m \times m$ orthogonal matrix to $2m \times 2m$ with zero padding, and backup the original matrix (similar to dynamic resizing of a hash table). This would entail dynamically expanding the second last layer of the network.

Identity Jacobian We somewhat agree with the reviewer about the non-linearity of ReLUs. We assume Jacobian to be identity and our principal motivation for this assumption comes from the work of Arora et al. (`https://arxiv.org/pdf/1901.08584.pdf`) – please refer to the text above their Eq. 7. However, we note that the authors identify the convergence to the linear update rule only for small networks. We will clarify this assumption more carefully in the paper.

Section 2: We don't claim *any* theoretical contributions in this work. Ours is only algorithmic contribution using well-known concepts from optimization and linear algebra. We apologize if the current presentation suggested otherwise. We will update/ rearrange the draft to avoid any pretense.

Memory size and generative replay: Performing well with the tiniest of memories is the focus of many recent continual learning works (`https://arxiv.org/pdf/1902.10486.pdf`, `https://arxiv.org/abs/2002.08165`, `https://arxiv.org/abs/1908.04742`) and it is the main motivation of our study. When the memory size is large, simple experience replay (multi-task training) performs the best and many recent works agree on that. Generative replay-based methods are problematic because of 1) they rely on learning a generative model in a continual setup which is as much, if not more, difficult than learning a discriminative model, 2) the memory requirement of storing a generative model is orders of magnitude higher than tiny/ small memories.

Ablation: Already provided in the paper – when P_t's are not orthogonal (Table 2, row 1 and 3). Effect of the replay buffer size (appendix Tab. 3)

Subset selection: Already provided in the paper – Alg. 1, line 21, we use ring buffers storing the last 'k' examples from each class. We used a fixed seed thereby storing the same examples in the replay buffer across different methods.

The number of runs for average and std: Already provided in the paper – Line 208, we use 5 runs.

Baseline numbers are computed by us: Already provided in the paper – Line 198, we generate the numbers for *all* the baselines (except VCL) from the same codebase that we made available in the supplementary.

R5: Using episodic memory in a method cannot be described as its weakness. In fact, there is a class of continual learning method that relies on the replay buffer of past tasks. Our method falls in that category. Six of the eight methods that we compare against make use of replay buffers. EWC does not use episodic memory but the other six methods do. Comparing non-memory and memory-based methods is a common practice in the continual learning community. We do not think the reviewer's criticism of our work is grounded in the continual learning literature. We respectfully ask the reviewer to reconsider their evaluation of our work.