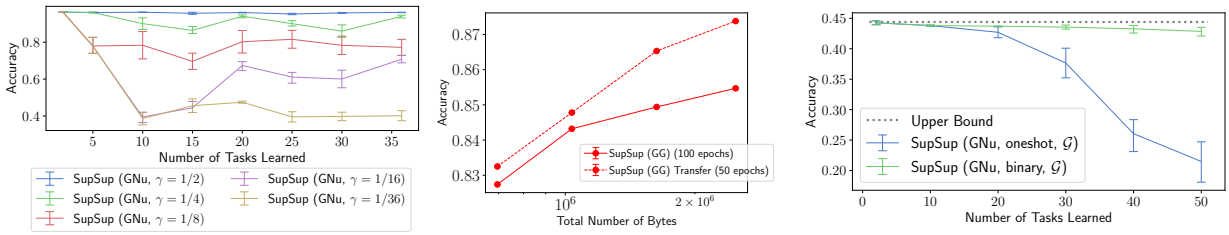


Figure 1: **(left)** Interpolating between the binary and one-shot algorithm with $\gamma$. **(center)** Transfer enables faster learning on SplitCIFAR. **(right)** One-shot vs. binary for permutations of CIFAR. Figures viewed best with zoom.

• **R1**: Thank you for the great suggestions and thorough review, we look forward to fully incorporating your detailed recommendations to improve the paper.

**Empirical comparison between binary and one-shot.** An empirical comparison between the binary and one-shot algorithms is a fantastic addition: In Figure 1 **(left)** we directly interpolate between these two algorithms. We replace line 6 of the binary algorithm, $g_i \leq \textbf{median}(g)$, with $g_i \leq \textbf{top-}\gamma\%\textbf{-element}(g)$. Then when $\gamma = 1/2$ we recover the binary algorithm (as $\textbf{median}(g) = \textbf{top-}50\%\textbf{-element}(g)$) and when $\gamma = 1/k$ we recover the one-shot algorithm. A performance drop is observed from binary to one-shot for the difficult task of MNISTRotate—sequentially learning 36 rotations of MNIST (each new rotation differing by only 10 degrees).

**Further comparison in GNu.** We believe the comparison of SupSup (in GNu) with recent methods (PSP [1], BatchE [2]) in the GG scenario is fair since GG is strictly easier than GNu. However we agree that this is a weakness and will update the paper to compare SupSup with methods *e.g.* from [3]. The initial reason for comparison of SupSup in GNu with recent methods in the strictly easier GG scenario is because they were more competitive. For instance [3] considers sequential learning problems with only 5-10 tasks. SupSup, after sequentially learning 250 permutations of MNIST, outperforms all non-replay methods from [3] in the GNu scenario after they have learned only 10 permutations of MNIST with a similar network: In GNu, Online EWC achieves 33.88% & SI achieves 29.31% on 10 permutations of MNIST [3] while SupSup achieves 94.91% accuracy after 250 permutations (see Table 5 in [3] vs. Table 7 in our work).

**Additional comments.** We have updated the appendix to explicitly detail the supermask training algorithm, which improves clarity. The method provided for NNs will not work with data that is common between tasks. We say that 16 bit integers are used instead of single bits because they store the index of the nonzero elements of the mask with the CSC sparse matrix format. PSP on SplitCIFAR achieves worse performance than BatchE (GG) with similar bytes. We will definitely think about extensions of SupSup to the continuous case, but do not currently have a solution.

• **R2**: We appreciate the suggestions, in particular to enhance the clarity of the figures which will improve the paper. For further comparisons please see the **Further Comparison in** GNu section in **R1** above. In reference to the comments concerning lack of novelty and lack of coverage of previous approaches we highlight **R3**'s comments: 1) *Applying supermasks to a continual learning scenario is definitely novel, and of interest to the community* and 2) *The paper has a lot of prior work to cover (CL, supermasks, batch ensembles) and is done correctly. Prior work is concise and clear.* The most similar approach to SupSup is [4] and they are limited to scenario GG while requiring more storage.

• **R3**: We are grateful for a comprehensive and thoughtful review. We complete variants of the two very useful experiments that you have suggested, and detail the results below.

**Forward transfer.** Thank you for highlighting the importance of transfer. We illustrate in Figure 1 **(center)** that our method for transfer (initializing each new mask with a running mean of previous masks) does enable faster learning (less epochs are required to reach a given accuracy). Training all tasks for 50 epochs with our transfer method provides a significant accuracy boost over training individually on all tasks for 100 epochs for SplitCIFAR.

**Towards more complex tasks.** We illustrate in Figure 1 **(right)** that SupSup with the binary algorithm can sequentially learn 50 permutations of CIFAR pixels with minimal forgetting. As in the paper we use the FC-1024-1024 network and train on each task for 1000 iterations, and accordingly the upper bound accuracy is low. However, we are most interested in the deviation from the upper bound. We will update the introduction to better reflect the scope of the paper.

[1] Brian Cheung *et al.*. Superposition of many models into one. NeurIPS 2019.

[2] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. ICLR 2020.

[3] Gido M van de Ven and Andreas S Tolias. Three scenarios for continual learning. arXiv, 2019.

[4] Arun Mallya, *et al.*: Adapting a single network to multiple tasks by learning to mask weights. ECCV, 2018.