We thank the reviewers for the time and effort they put into the reviews and address their questions and comments below. In particular R3 and R4 asked for more information about baselines and we point them below to the relevant baselines in the paper (we compare to 6 different methods) and try our best to address their concerns about the method.

re CIFAR/other architectures and code (R1/R2): We agree that more datasets/architectures would always be useful however in practice CIFAR is easier to sparsify and train sparse than ImageNet, thus we believe that ImageNet is a better demonstration of the capabilities of our algorithm. We would argue that CIFAR should not be used for comparisons of these techniques as findings do not always translate to larger datasets. We provide results from two completely different/popular families of models (resnet and transformer) and two different tasks (language modelling and classification) on large scale datasets and believe this demonstrates the efficacy of the method. We will add pseudocode for how to implement this method. In general this can be implemented with a custom getter in TensorFlow, such that the weight matrix is sparsified before any computation occurs.

re tuning backwards sparsity (R2): In practice the correlation between backward sparsity and performance is straightforward: lower backwards sparsity generally allows us to discover better sparsity masks. A benefit of this aspect of our method's design is that backwards sparsity can thus be chosen based on the FLOP budget available and the amount of memory available – in general the recommendation would be to use as dense a backward pass as the budget allows.

Re baselines (R3/R4): Both reviewers ask for a comparison to dense-to-sparse methods. We point the reviewers to Figure 2 in the paper **where we compare with six different baselines, of which two (Pruning and SNIP) are dense-to-sparse methods**. Moreover we outperform most of them on a per FLOP basis while staying entirely sparse. Reviewer 4 asks for comparisons to SET and DSR. We compare to RigL which consistently outperforms both SET and DSR so we did not see the value in including this baseline, although we do also compare to SET in Figure 2 and are happy to add DSR.

Re Novelty and Tuning (R3): We disagree that the model is a simple tweak of pruning given we maintain constant sparsity throughout training. Moreover we match state-of-the-art performance on Imagenet and sparsify TransformerXLs – something which has thus far not been done in a sparse-to-sparse manner. On the point about tuning hyperparameters: we in fact reduce the hyperparamtere compared to pruning (which requires a hand-tuned schedule). Additionally our results outperform or are similar to those published by other sparse-to-sparse methods and are also the first method that allows differing backward and forward sparsities.

Re Wiki103 (R3): There is no published method that has been applied to Wiki103 in a completely sparse-to-sparse manner. We also disagree that there are huge performance drops - as we show **we match the performance of a 97M parameter dense model with 57M and 60$\%$ backward sparsity**. On the point about comparing to pruning, we do this on a smaller model (as we could not fit the pruning masks for bigger models in memory) and compare in Table 5 in the appendix.

Re theoretical flops (R4): Every single published method we compare to also relies on theoretical FLOP reduction. Current industry trends strongly suggest that future advancements in hardware/software kernels will allow us to implement sparse methods with native-sparse support. While valid, the above criticism can be equally levied against every previously published sparse-to-sparse method and would devalue every single published paper in the field.

re theoretical proofs/limited practical value(R4): We do not have a proof that using top-k is a best/better choice and is unclear whether this is tractable theoretically. We do however point the reviewer to section 2.1 where we explain with a taylor approximation why this approximation holds and that the algorithm will converge to some local minima. On structured sparsity: our method is equally applicable to block sparsity given a function such as max-or sum-of-absolute-values that reduces a block to a scalar value.

re regularization loss (R4): this is used in all experiments and is an integral part of Top-KAST (not a potential add-on). We will make this clearer in the paper.

re correctness and other experiments (R2): We thank the reviewer for their comments and suggestions – on random vs topk, we will explore this further. One added benefit of top-k over random is also the stability of the mask and not having to constantly resample it. On corner cases, bwd sparsity=fwd would be similar to SET with a difference in how we sample new parameters and bwd=0 would be like DNW but with the special regularization. On the comparison with RigL and some of the subjective words, we will further clarify our language in the paper as our intention was not to overclaim. In general for lower values of sparsity, RigL outperforms Top-KAST, and Top-KAST outperforms for higher. We view the overall results as comparable (specifically both methods significantly outperform previous baselines like SET) and note in the paper the main advantages of Top-KAST over RigL in section 4.

Typos and minor comments and ablations: We thank all reviewers for their notes on these and will correct this in future versions. We thank R1,R2 and R4 for their suggestions for ablations. While unfortunately out of scope for this document, we will strive to include this in the paper.