We thank the reviewers for their time and helpful feedback. Below we respond to their comments in turn.

Reviewer 1 ‘permutation invariance in write values’ Yes, we process the values in parallel and then take the sum over the batch dimension. We will make this clearer in the updated manuscript.

‘parameterization of the backwards function’ Thanks for this suggestion. We agree this discussion would be useful. In the Appendix, we briefly describe the different parameterization strategies for the backward function that we initially attempted. We’ll move this to the main text and add more discussion.

‘Error bars for the plots in Figure 2.’ We will conduct multiple runs and include error bars in Figure 2.

‘Add titles to each subgraph in Figure 5.’ Indeed, we should have done this. We will add titles to each subgraph.

‘MNM-p results for the Maze exploration task.’ At the time of submission, we were still running the MNM-p model on the maze task. The results show that MNM-p performs on par with MNM-g on the maze task. We will include this finding in the paper.

‘cite synthetic gradients’ Thanks for pointing this out. We’ll add that citation.

Reviewer 2 ‘Is there any intuition that proposed model has better performance than NTM ’ Our hypothesis is that a memory module implemented as a neural network, whose weights can change over the course of an episode, will offer greater expressivity than attention-based tabular memory modules like the NTM, as well as constant time and space overhead. The main goal for the new dictionary inference task introduced in the paper is to validate this hypothesis empirically. Note that the MNM uses similar network components as the NTM: we use an LSTM controller combined with an MLP memory function (more complex architectures for the memory would be interesting to explore in future work). On the other hand, the meta-training process for writing to memory is indeed based on recent techniques (more on techniques from [1,2] than MAML), as you point out, and our experiments suggest this kind of meta-training can be adopted for information storage over an episode.

‘The experiments and their results are not intuitive to understand’ The double copy and sort tasks are standard algorithmic benchmarks from the literature on memory-augmented neural networks [3,4]. We apologize if this was not stated clearly and will fix this in the updated manuscript. The LSTM+SALU baseline that we compare against replaces the metalearned MLP memory function with the soft-attention look-up table used in several memory augmented models, eg RNNSearch[5], MemN2N[6] Transformers[7]. Because this is the same overall architecture as MNM with a different memory module, we believe it’s a fair, minimally distinct model to compare against to validate the hypothesis stated above. The dictionary inference task we introduced is a toy proxy for few-shot machine translation and we believe it can be used more broadly to study models with adaptive/memory behavior. We will make the small plots larger to improve their readability.

Reviewer 3 ‘Figure 1: notations should be avoided’ That makes sense. We will use the name “interaction vectors” in Figure 1 or describe the notations in the caption or move the figure.

‘Section 3: the definition of the meta loss’ That’s correct: during training, we store H values as part of the computation at each time step and use them later for backprop.

‘the exact training set-up is unclear’ During training, the model sees many instances of a given task. These instances are used for meta-training. For example, in the “sort” algorithmic task it sees many randomly generated sequences to be sorted. The sequence length is 20 and each element in sequence is 8 bits. The model sees 1M randomly generated sequences during training. We plotted the training curve for show convergence. See NTM [3] for a more detailed description. We will clarify the training setup in the updated manuscript.

‘Line 134: Why are biases missing?’ For simplicity, we didn’t use biases in the neural memory module.

‘Line 137: Notation appears incorrect’ Thanks for pointing this out. We’ll fix the notation.

‘why the authors referred to a function approximator as a memory module’ We cast memory as a family of adaptive functions – neural nets – that are suitable for the storage of information. These functions can be updated rapidly over the course of an episode (not just at training time) to “write in” new information, and they can be read from over an episode to recall information. This is indeed a somewhat novel take on memory, but note that a flexible function approximator could, in theory, learn to approximate more standard “data structure” types of memory functions, like hash maps, etc. Note also that typically memory-augmented neural networks do not store “raw” input information in their memories, but rather vector encodings derived from the inputs.

‘the authors could try their method on a harder QA task’ Our focus in this work was to validate the model and understand its memory operation by running on bAbI task. Since the results were very encouraging, we plan to put more effort and try the model on other benchmarks in future work.

Ref: [1] Andrychowicz et al. "Learning to learn by gradient descent by gradient descent." Advances in neural information processing systems. 2016. [2] Ravi et al. "Optimization as a model for few-shot learning." (2016). [3] Graves et al. "Neural turing machines." arXiv preprint arXiv:1410.5401 (2014). [4] Santoro et al. " Advances in Neural Information Processing Systems. 2018. [5] Bahdanau et al. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014). [6] Sukhbaatar et al. "End-to-end memory networks." Advances in neural information processing systems. 2015. [7] Vaswani et al. "Attention is all you need." Advances in neural information processing systems. 2017.