Review 1: We thank the reviewer for the comments and suggestions. We will fix all typos pointed out in the review. Below we address the two main issues raised by the reviewer .

How is "sum" harder than "unique sum"? Indeed accuracy on the *unique-sum* task is better than those of *sum*. The reason is that for unique-sum the input domain is $\{0, \ldots, 9\}$, and for sum it is $\{0, \ldots, 99\}$ (we followed the setup in the Janossy paper). This makes it possible to learn unique-sum by simply having a 10 dimensional state where dimension i is "on" when digit i appears. For the rebuttal, we trained unique-sum with $\{0, \ldots, 99\}$ and accuracy was much lower than sum.

How can DeepSets fail for $n = 15$? DeepSets failure here is due to convergence to a bad local minimum. We used the standard optimization setup for DeepSets (in the Janossy code). We believe that one advantage of SIRE is improved robustness to optimization failures, because the regularizer also makes optimization easier by using shorter sequences.

Baseline results for "half range". For the rebuttal, we ran the *half-range* experiment with DeepSets and it failed to converge to a good solution, resulting in a model which outputs predictions based on the statistics of the data. The resulting accuracy results numbers are (std in brackets): $0.050(4 \cdot 10^{-6}), 0.077(1.4 \cdot 10^{-4}), 0.0895(2.5 \cdot 10^{-7})$ for $n = 10, 15, 20$. These are much worse than what SIRE obtains.

Review 2: We thank the reviewer for the comments. We will clarify definitions 2 and 3 according to the suggestions.

Scalability of the proposed method. The main focus of our work is to highlight the representational advantages of RNNs, which makes them a very relevant part of the toolbox in invariant modeling. While this comes at a certain computational cost, we believe there are many cases where the benefit in accuracy will make this attractive in practice as demonstrated theoretically and empirically throughout the paper. A similar trade-off is evident in natural language processing where recurrent methods such as GPT are widely used though there exists competing non-recurrent language models such as BERT.

Review 3: We thank the reviewer for the through review and insights. We address each point raised below.

How is SIRE sampled? We estimate R_{SIRE} by applying a random permutation over input sequences $x_1, \ldots, x_n$ followed by truncating the permuted input sequence to result in two sequences: $x'_1, \ldots, x'_{k-1}, x'_k$ and $x'_1, \ldots, x'_k, x'_{k-1}$ (i.e., the two last elements are switched). This simple scheme worked well, but we agree that other distributions can be explored. Also, note that when the data is generated from S_n the above scheme is uniform over $\mathcal{S}_{\mathcal{D},\Theta}$.

What is the complexity of sampling s in estimating SIRE? As mentioned above, SIRE is estimated by sampling sequence pairs. The sampling operation is linear in sequence length, and then each one of the RNN is applied to the two sequences. Thus, cost is similar to regular training.

Number of SIRE Samples. For each original input sequence, we sample just one pair of sequences for the SIRE regularizer. Of course, for a given sequence, different pairs may be sampled each time this sequence is used.

Def. 3, variables i and t. Indeed, the definition requires $2 \leq t \in \mathbb{N}$ and $i < t$. We will clarify this.

In Theorem 5, it should be "if and only if"? The condition in the theorem is slightly stronger than subset-permutation-invariance, because the sequence that results in adding x_1, x_2 may not correspond to a permutation of an actual input sequence. The theorem can be easily adapted to make it "iff", but notation would be more cumbersome.

What kind of "semi"-invariance is compatible with SIRE? In problems that are not permutation invariance, SIRE will seek RNNs that are "as close as possible" to permutation invariant, where closeness is measured by SIRE. It will indeed be interesting to further analyze this tradeoff, and we leave this for future work.

How far can one push the sequence length until it no longer works? The answer to this question depends heavily on the task at hand and is mostly a matter of optimization. For the sum task, performance begins to deteriorate on relatively short sequences - for $n = 15$ the average accuracy of an RNN using SIRE regularization is 0.957 (0.022). For the purpose of the rebuttal we ran the same experiment with $n = 17$ and got an accuracy of 0.913 (0.003).

Review 4: The authors wish to thank the reviewer for the comments and points raised. We address the concerns below.

Can the proposed method be used for permutation equivariance? Yes. In a multi-output setting, we can add a regularizer similar to SIRE, but that requires that adding x_1, x_2 to state s will generate outputs y_1, y_2 whereas adding x_2, x_1 will generate outputs y_2, y_1. This will result in similar guarantees to what we have, but for equivariance.

Alternative aggregation methods. This is indeed a main motivation for our work. In DeepSets and related methods, the choice of aggregation method is key to obtaining good performance. While in principle one can check $\{min, max, sum\}$ there are many other possibilities. On the other hand, in our case, the aggregation function is learned automatically by the RNN, and as we show, the RNN works for cases corresponding to different aggregations.

Sequential computation. Recurrent architectures are used extensively in many machine learning applications, from generative models in NLP (e.g., the GPT model is state of the art in language generation, and arguably outperforms BERT which is non-recurrent) to video processing (where 3D convolutions and RNN over 2D convolutions are both commonly used). The reviewer is correct that their sequential nature makes them less amenable to GPU speedups. However, their representational power compensates for this limitation in many cases. Our argument in this paper is that RNNs are very relevant for permutation invariant modeling, and we demonstrate this empirically in several cases.