We thank the reviewers for their valuable comments.

Reviewer #1:

Q: Optimization of objective (22) with the proximal map (21) is not discussed in detail. MATLAB code is not available.

A: Thank you for bringing it to our attention. The MATLAB code has been uploaded to the GitHub repository. It simply invokes fminunc to optimize (22), without providing a gradient subroutine. This leaves fminunc to choose its own solver which typically utilizes its own finite difference routine. The result looks good and efficient for this dataset.

Q: the paper does not explore more straightforward applications to per-layer regularization in deep neural networks. Mentioning this seemingly fundamental application, the authors do not study it just saying that it is straightforward to implement. This makes the reader wonder whether this method can actually show promising results in this setting.

A: ProxNet for multi-view learning used proximal mapping for only one layer, and so did Figure 1. In contrast, ProxLSTM in Section 4 embeded a proximal mapping into *each* step/layer of an LSTM. It will be very interesting to study the use of multiple proximal mappings at different layers for diverse purposes, e.g., to enforce equivariance in each layer of feature extractor by using the violation as the regularier R, disentanglement at a certain layer, and fairness in prediction. This requires more space than a NeurIPS paper, and we are exploring some of these combinations.

Reviewer #2:

Thank you for pointing out Meinhardt et al. (2017). It is relevant and we will cite it in the future versions.

Q: Proximal operators may hurt the performance if they are not instantiated correctly for the problem and model that are being considered. I do not have much intuition on how the ProxNet improvements upon other regularization approaches.

A: ProxNet is a means of enforcing deep regularization, while the regularizer R itself is to be designed by the practitioners for the specific application, e.g., CCA for multi-view learning. So ProxNet by itself does not introduce any new regularizer. Implementing it is straightforward as shown in Eq (2) for any generic R, though some parameters need to be chosen (see the next question). The purpose of the paper is to show that for regularizers defined in terms of hidden layer outputs, it is more effective to regularize **in-place** through a proximal mapping at that layer, compared with adding the regularizer to the overall objective and relying on backpropagation for optimization. By "more effective", we have compared by using the test performance instead of the training objective value, because unlike comparing two different nonconvex solvers, ProxNet results in a different objective than regularized risk minimization. A rigorous analysis beyond the intuition of in-place regularization (to quote Reviewer 3: "the kind of idea that seems obvious in retrospect") will be interesting in the deep context, and we plan to investigate it in the future.

Q: Does care need to be taken to properly setting the ϵ term to ensure the derivative approximation is well-behaved? Will the finite-difference derivative approximation in L178 causes instability during learning?

A: [16] provided several heuristics for setting ϵ. Our experiment set $\epsilon = \delta(1 + ||(X, Y)||_\infty)||(\frac{\partial L}{\partial P}, \frac{\partial L}{\partial Q})||_\infty^{-1}$ so as to be adaptive to the magnitude of the gradient, and δ is a small constant whose value was chosen by checking a few gradients at the beginning of training. To estimate the "true" gradient needed for the check, we used a small ϵ and solved the proximal mapping in Eq (10) to high accuracy. With this heuristic, the approximate gradient did not cause instability in training. It is also noteworthy that the stochastic gradient computed from a mini-batch introduces noise in the first place.

Reviewer #3:

Q: Would it be helpful to anneal the RRM term over time as the ProxNet term is annealed out?

A: We conducted some additional experiments to show how an annealed λ influences the performance of RRM. We not only tried the annealing schedule used for ProxNet, but also other schedules. The results for multi-view learning on the Sketchy dataset are shown in the right table, where, overall, annealing has mixed influence on the vanilla RRM, but it remains inferior to ProxNet.

#class	20	50	100	125
RRM new	17.4 $\pm$ 0.8	22.5 $\pm$ 0.5	24.3 $\pm$ 0.9	26.1 $\pm$ 0.7
RRM old	15.2 $\pm$ 0.6	20.1 $\pm$ 0.4	26.8 $\pm$ 0.5	28.1 $\pm$ 0.4
ProxNet	**13.7** $\pm$ 0.3	**17.9** $\pm$ 0.5	**20.2** $\pm$ 0.3	**22.0** $\pm$ 0.4

Reviewer #4:

Q: I don't see any aspect of the robustness method for RNNs that actually relies on the recurrent structure. Am I missing something, or could it be placed on any type of neural network hidden layer?

A: The ProxLSTM in Section 4 made RNN robust by applying a proximal mapping at each layer/step for invariantization. This is innately synergistic with the recurrent structure. In other words, we achieved robustness not because of using the recurrent structure itself, but by properly invariantizing each step via embedding a proximal mapping. So this technique can be generically deployed in any type of neural network.