We sincerely thank all reviewers for their helpful and constructive feedback.

Response to Reviewer 1:

-**Pareto solutions for less relevant tasks:** 1) Less relevant tasks often compete with one another, and could lead to worse performance as noticed in [decaNLP, McCann2018]; 2) Our experiments on Multi-Fashion+MNIST (Fig.4(c), two less relevant tasks) show that Pareto MTL can still provide a set of widely-distributed Pareto solutions; 3) The solution with a strong preference on one task (e.g., (1,0)) can achieve the best performance on it, and the other one can be treated as an auxiliary task; 4) Some recently proposed works on learning tasks relation (e.g., [Ma2018, KDD]) would be useful for Pareto MTL to deal with less relevant tasks. We will discuss it in the revision.

-**Unclear sentences:** 1) L87-88: What we claim is that all those methods try to balance different tasks by adapting the weights and do not have a systematic approach to incorporate preference information (more details in the supplement: adaptive weight vectors); 2) L173-175: As discussed in [30], the projection approach (8) is an n-D constrained optimization problem. It is inefficient to solve it directly, especially for a DNN with millions of parameters. Our approach reformulates it as an unconstrained problem (to reduce the value of all activated constraints) which can then be efficiently solved by a gradient-based method; 3) We will rewrite these sentences to make them clear in the revision.

-**Preference vector:** They are evenly distributed on the first quadrant of a unit circle. Five preference vectors for two tasks are: $\{(cos(\frac{k\pi}{8}), sin(\frac{k\pi}{8}))|k = 0, 1, ..., 4\}$. See response to R4 for random vectors. We will add it in the revision.

-**Better performance:** Short analysis: 1) Using the same argument in [7, 13], ParetoMTL as an adaptive weight method can outperform fixed weight method; 2) Using the same argument in [12], treating MTL as MOO can obtain better performance than heuristic-based adaptive weight approach; 3) Compared with MOO-MTL[12], Pareto MTL can find Pareto solutions with different trade-offs by decomposing the MTL; 4) We will add a detailed analysis in the revision.

-**Advanced networks:** 1) Following the setting in [12], we used LeNet as the basic network for MNIST-like datasets; 2) Our preliminary results show that a more powerful network can provide better results for all experiments while the conclusions still hold. We will report these results in the revision.

Response to Reviewer 2: We have fixed the typos and carefully proofread the whole paper. We agree that testing Pareto MTL on tasks from different modalities is important. We will add a comparison and discussion in the revision.

Response to Reviewer 4:

-**Why other methods fail:** 1) As shown in [ConvexOptBook, Chapter 4.7.4 Scalarization, Boyd2004], linear scalarization cannot find the concave part of the Pareto front. We will discuss it in the revision; 2) MOO-MTL tries to balance different tasks during the optimization process (see the adaptive weight analysis in the supplement), so its solutions are most likely in the middle of the Pareto front; 3) Our proposed Pareto MTL decomposes a given MTL and guides search to different sub-regions for obtaining well-distributed solutions; 4) We will move the introduction of the toy example from the supplement to the main paper and add analysis on the concentrated behaviors.

-**Preference vector:** See response to R1 for vector generation. The final distribution of the solutions depends on both the preference vectors and the shape of the Pareto front. Fig.1 shows the effect of different random prefs. Utilizing prior information/learning-based method is an important extension to find better-distributed solutions.

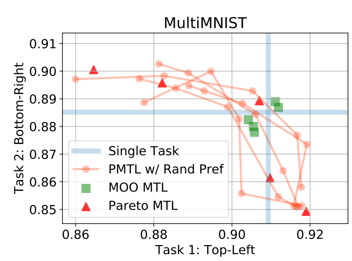

Figure 1: PMTL w/ different sets of rand prefs can consistently generate well-distributed sols, but too close prefs might lead to worse performance. Discussion will be added in the revision.

-**Many tasks:** 1) Combining many tasks is an important problem in MTL. As discussed in [decaNLP, McCann2018], simply combining many (and less relevant) tasks might deteriorate their performance; 2) Pareto MTL needs to solve an (n**_tasks** + n**_preferences**)-dimensional constrained opt problem to find the descent direction at each iteration, and solve n**_preferences** subproblems in total, which would be very time-consuming; 3) However, generating a large number of Pareto solutions would be useless (think about an MTL practitioner needs to choose 1 among 50 Pareto solutions balancing 15 tasks); 4) For many tasks case, we can still apply Pareto MTL by a) finding representative solutions with preferred trade-offs and b) tasks/objectives reduction method from MTL/MOO community.

-**More accurate equations:** 1) Equation 3 intuition: for finding a descent direction , we either have: a) $\nabla \mathcal{L}_i(\theta_t)^T d_t \leq 0, i = 1, ..., m$, which means d_t is a valid descent direction for all tasks or b) no valid descent direction can be found, the current solution is a Pareto critical point. We will add a paragraph to explain this intuition; 2) We have also fixed the equations you pointed out and carefully proofread the whole paper to make all equation clear to be understood.