Thank you (R1, R2, R3, R4) for your insightful comments. We will release the code and fix the minor issues.

Cost of Task Similarity Detection (R1,R2). Yes, it incurs additional computation, but it is reasonable. MTCL's training time (h) ranges from 0.03 to 0.15, while HAT ranges from 0.0035 to 0.0039. MTCL's number of parameters (M) ranges from 43.4 to 50.4, while HAT ranges from 7 to 10.5. See Sec. 2.3 in *Appendix* for more details. As MTCL makes large gains compared to strong baselines (at least 3%, up to 30%), we believe the performance gain is worth the additional computation. Also importantly, the proposed problem is more general and has not been attempted before.

Use of Validation Set (R1). Validation set (part of the training set) has been used in many continual learning techniques, e.g., HAT, EWC and ProgressiveNet, for parameter tuning. In our case, we also use it for similarity detection.

Necessity of Similarity Detection (R1). It may be possible to design an algorithm without explicit similarity detection. In our current model, it is needed; otherwise, we cannot generate TSV (Task Similarity Vector). Ablation study (Table 3) shows that MTCL degrades severely without TSV (*-TSV*) as TSV allows only the units of similar tasks to be updated; otherwise, severe forgetting occurs. HAT does not detect task similarity. It only detects which units are associated with which tasks and block them when learning future tasks.

Regression v.s. Classification (R1). We regard task similarity detection as a binary classification problem. The detection results (TSV) are used to determine what parameters to update (i.e. only those units associated with similar tasks can be updated). It is also interesting to consider this as a regression problem. We can study that in the future.

Overall	NCL	ONE	HAT	**MTCL**
M(CIFAR100-10,F-CelebA)	0.6155	0.6764	0.6178	**0.6831**
M(CIFAR100-20,F-CelebA)	0.6931	0.7428	0.6946	**0.7468**

Table 1

Simple Architecture - does CNN work for CIFAR and CelebA? (R2). As we want to make our experiments uniform, we adopt the same backbone MLP architecture for experiments. Replacing MLP with CNN is straightforward for MTCL. We did that (see CNN's overall results in Table 1, better than the MLP results) - MTCL again outperforms the baselines. ONE is one-task learning, learning each task independently.

Overall	PathNet	RPSNet	**MTCL**
M(EMNIST-10,F-EMNIST)	0.5901	0.7044	**0.7710**
M(CIFAR100-10,F-CelebA)	0.5504	0.4801	**0.6194**
M(EMNIST-20,F-EMNIST)	0.7049	0.7620	**0.8439**
M(CIFAR100-20,F-CelebA)	0.6169	0.5410	**0.6843**

Table 2

Additional Baselines, Dataset Choice and How Similar Tasks Different from Existing Systems' Setting (R3). Our work is different from existing systems: (1) We consider a *mixed sequence of tasks*. Forgetting mainly affects learning *dissimilar* tasks. For similar tasks, forgetting is not a major issue (see lines 274-277). Knowledge transfer is important. Regarding *dissimilar* tasks, we adopt datasets CIFAR100 and EMNIST as they are commonly used benchmarks. They form *dissimilar* tasks as their classes have little knowledge sharing. For *similar* tasks, we adopt the datasets in federated learning (see lines 4-14 in Appendix). Federated learning is to train through model aggregation rather than the normal data aggregation and keep local data on the local device. These datasets naturally consist of *similar* tasks. How to leverage the shared knowledge is important for similar tasks, (2) Existing models mainly focus on addressing forgetting because they don't consider *mixed* datasets, but only dissimilar datasets/tasks. Although some models do limited forward transfer, they still mainly deal with forgetting. They cannot improve the results of similar tasks as we do. For example RPSNet (Rajasegaran et al., 2019) and PathNet (Fernando et al., 2017) detect which path to reuse but do not allow the used parameters to be updated. These are insufficient because similar tasks' parameters should be allowed to update for both forward and backward knowledge transfer. We believe this is an important contribution of ours. LwF and GEM also use the previous model in learning the new task, but it is also for dealing with forgetting only and it cannot improve similar task performance through forward and backward transfer. **Additional comments**: (1) We have conducted new experiments on the two suggested baselines: PathNet and RPSNet. Table 2 shows their overall accuracy results (average over 5 random sequences). Detailed results will be added to the paper. MTCL outperforms PathNet and RPSNet considerably. (2) About motivation of our work, please refer to lines 15-36. We will explain more in the revised paper. (3) Our dataset choice: (a) datasets in federated learning give us similar tasks, and (b) the similar datasets should be paired with our dissimilar datasets EMNIST and CIFAR in terms of the image size to copmose the mixed task sequences. Other public datasets (Caldas et al., 2018) in federated learning are either text or with different image/input sizes, which are unsuitable for the data pairing.

Other Comments (R1,R2,R3,R4). *(1) How to make the output of the sigmoid function binary.* It is binaried by adding the hyper-parameter s in Eq. 1. s is annealed (see lines 139-144). When s is very large ($s \to \infty$), the output of sigmoid $m_l^{(T)} \to \{0, 1\}$, approximating to a unit step function. *(2) Are the readout functions trained after applying the masks?* Yes. We first use the saved mask $\{m_l^{(k)}\}$ to get the representation/features for $x^{(t)}$ produced by the task k model in the KB and then train the readout function. *(3) The annealing strategy is somewhat similar to controller proposed in iTAML.* Yes, but one notable difference is that the controller in iTAML is to balance between plasticity and stability, while ours is to train a binary mask. We will cite and compare with iTAML in the revised paper. *(4) Task ID has to be known.* Task ID is used in all task-based continual learning algorithms. This is so because the tasks can be completely different or overlapping and thus need separate sub-models in the neural network. For example, one task is to classify fish and non-fish, and another task is to classify different kinds of fishes. Without knowing the user's specific task at hand, the system will not know which is the right answer.