We thank reviewers for their feedback. As stated by reviewers, the work is novel (R4), very interesting (R1, R2, R4), backed with extensive experiments (R2, R3), detailed ablation studies (R1, R2, R4) and qualitative analysis (R2, R4).

R2, R3, R4: **Experiments on Additional Datasets.** We experiment on more diverse DomainNet [a] with 6 dissimilar image classification tasks using ResNet34 and text recognition with larger number of NLP tasks (10 different publicly available datasets from [b]) using VD-CNN [c]. AdaShare improves average accuracy over 'Multi-Task' by **4.6%** (max. **16%** in *quickdraw*) for DomainNet, and **7.2%** (max. **27.8%** in *sogou_news*) for text recognition. Similar to Fig.3.b, we visualize task relationship on DomainNet, which shows similar tasks are more correlated, such as *real* is closer to *painting* than *quickdraw* (Fig. 1).

Models	$\Delta_{\mathcal{T}_1} \uparrow$	$\Delta_{\mathcal{T}_2} \uparrow$	$\Delta_{T} \uparrow$
WRN			
Multi-Task	-0.35	9.63	4.64
AdaShare	9.36	11.53	10.44
MobileNet-v2			
Multi-Task	0.18	8.02	4.10
AdaShare	4.16	10.61	7.39

Table 1: NYU v2 2-Task. $\mathcal{T}_1$: Semantic Seg., $\mathcal{T}_2$: Surface Normal Pred.

R1, R4: **Extension to other Architectures.** We implemented AdaShare using Wide ResNets (WRN) and MobileNet-v2 in addition to ResNets. AdaShare outperforms 'Multi-Task' by **5.8%** and **3.2%** using WRN and MobileNet respectively in NYU v2 2-Task (Tab. 1). We observe a similar trend on CityScapes.

R1, R2: **Computation Cost (FLOPs).** AdaShare requires much less computation (FLOPs) as compared to existing MTL methods. E.g., in Cityscapes 2-task, Cross-stitch/Sluice, NDDR, MTAN, DEN, and AdaShare use 37.06G, 38.32G, 44.31G, 39.18G and **33.35G** FLOPs and in NYU v2 3-task, they use 55.59G, 57.21G, 58.43G, 57.71G and **50.13G** FLOPs, respectively. Overall, AdaShare offers on average about **7.67%-18.71%** computational savings compared to SOTA methods over all the tasks while achieving better recognition accuracy with about 50%-80% less parameters.

R1: **Sparsity Loss.** Sparsity loss enhances compactness and also helps learning task-specific layers (i.e., skipped layers in one task form the task-specific layers of other tasks) which potentially reduces negative transfer, leading to performance improvement in MTL.

R1: **RL-based Methods.** Table 5 shows that AdaShare is better than AdaShare-RL, in line with comparison in [57]. This is due to RL policy gradients are often complex, unwieldy to train and require techniques to reduce variance during training. In contrast, Gumbel Softmax sampling (used in this work) makes the framework fully differentiable with more efficient gradient feedback from the training loss and also easier to optimize, as shown in [29,55,59].

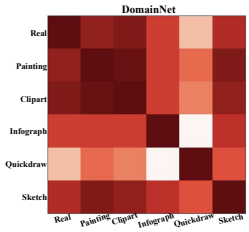

Figure 1: Task Correlations in DomainNet.

R2: **Applications.** Our approach is easy and straightforward to apply: during training, we learn an feature sharing pattern and then at testing, the learned pattern is followed, selectively choosing what layers to compute for each task. Our source code will be publicly available (also included in supp).

R3: **Difference from Prior Works.** While methods in [1-4] enhance efficiency of a single classification task via training regularization, AdaShare **jointly** learns feature sharing patterns among multiple tasks via adaptive computation. Compared to *task-specific residual adapters*, AdaShare requires **36.2%** less parameters and **23.4%** less FLOPs, with an overall improvement of **5.6%** on NYU-v2 3-Task learning. As suggested, we also compare with *task-specific stochastic depth* and find that AdaShare outperforms it by **5.7%** on NYU-v2 3-Task. Our approach is effective as it not only encourages positive sharing among tasks via shared blocks but also minimizes negative interference by using task-specific blocks when necessary. Thanks for the references–we will cite them in our final version.

R3: **Dropped Blocks vs Performance.** The average probability of a block to be dropped depends on the real task difficulty and hence more blocks can be dropped for an easier task without affecting the performance. AdaShare mediates among tasks and automatically decides shared and task-specific blocks adaptive to given task set.

R3: **Higher Task-to-Layer Ratio.** We believe using a much higher task-to-layer ratio may require increase in network capacity to superimpose all the tasks into a single multi-task network. AdaShare can be extended to dynamically grow the network capacity in addition to feature sharing, which is an interesting topic for future work.

R3: **Effect of Pre-Training and Extension to Channel Sharing.** Thanks! Effectiveness of pre-training depends on tasks but we observe that it improves our performance by 11.3% in NYUv2 3-Task. We started from scratch for a fair comparison among different methods. AdaShare can be easily extended for finding a channel sharing pattern and our preliminary experiments on DomainNet shows encouraging results; we leave this as an interesting future work.

R4: **Stage-wise Training and Curriculum Learning.** We follow [55,58] and adopt a two stage training approach to ensure the feature sharing pattern generalize to the validation dataset. We observe that the network weights learned using one stage training is not fully optimized resulting in a drop of performance by about 15% in NYUv2 2-Task. Both Tab. 5-main and Tab. 7-supplementary shows effectiveness of curriculum learning (improvement of 3.3% in both cases).

R4: **Task Relationships**–See Fig.1 and analysis at top for diverse task correlations in DomainNet. **NYU v2 Surface Normals**–We use publicly available surface normals provided by [15]. **Clarity Issues**–We will fix them in final version.

References: [a] Peng et al, Moment Matching for Multi-Source Domain Adaptation, ICCV'19. [b] Chen et al, Exploring Shared Structures and Hierarchies for Multiple NLP Tasks, arXiv'18. [c] Conneau et al, Very Deep Convolutional Networks for Text Classification, EACL'17.