AdaShare: Learning What To Share For Efficient Deep Multi-Task Learning

Ximeng Sun[1], Rameswar Panda[2], Rogerio Feris[2], Kate Saenko[1,2]

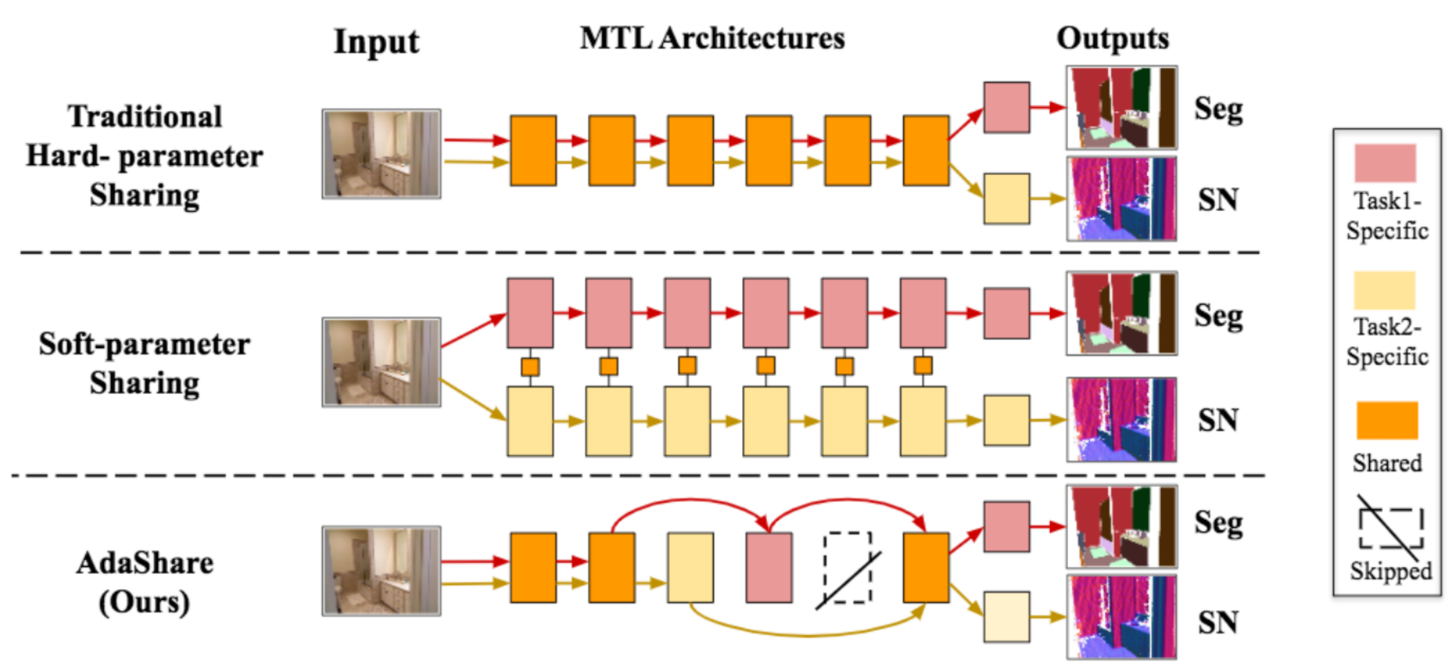
[1]Boston University, [2]MIT-IBM Watson AI Lab, IBM Research

Project Page: https://cs-people.bu.edu/sunxm/AdaShare/project.html

Introduction

Multi-task learning: one machine learning system to solve multiple tasks
Challenge: exploit **commonalities** and **differences** across tasks.

Theoretically, MTL improves generalization by leveraging the domain-specific information contained in the training signals of related tasks



Hard-parameter Sharing:
Advantages: Scalable
Disadvantages: Pre-assumed tree structures, negative transfer, sensitive to task weights

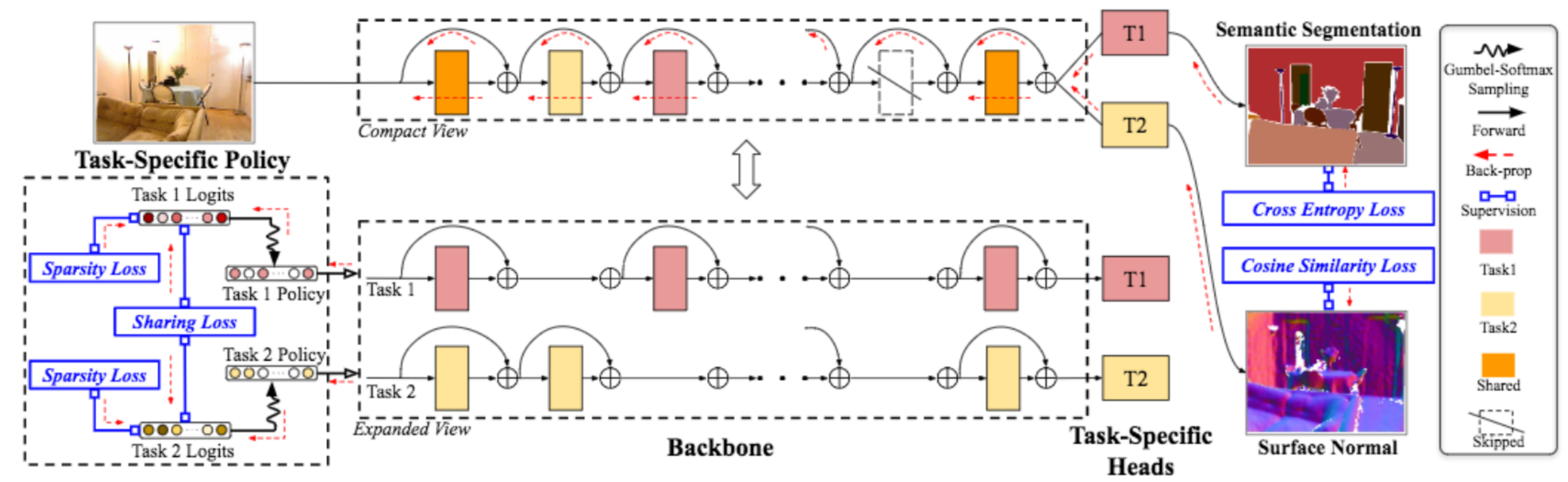
Soft-parameter Sharing:
Advantages: Less negative interference (yet existed), better performance
Disadvantages: Not Scalable

Ours: **Adaptively** learn the sharing scheme, instead of tree structure, in hard-parameter sharing setting

Our Contributions

- We propose a novel and differentiable approach **for adaptively determining the feature sharing pattern** across multiple tasks (what layers to share across which tasks)
- We learn the feature sharing pattern jointly with the network weights **using standard back-propagation**. We also introduce **two new loss terms** for learning a compact multi-task network and **a curriculum learning strategy**.
- We **conduct extensive experiments on several MTL benchmarks** (NYU v2, CityScapes,Tiny-Taskonomy, DomainNet, and text classification datasets) with variable number of tasks to demonstrate the superiority of our proposed approach over state-of-the-art methods.

Our Approach



Generally, we **seek a binary random variable** $u_{l,k}$ (a.k.a policy) **for each layer** l **and task** $\mathcal{T}_k$ that determines whether the l-th layer in a deep neural network is selected to execute or skipped when solving $\mathcal{T}_k$ to obtain the optimal sharing pattern, yielding the best overall performance over the task set $\mathcal{T}$.

Optimization

Gumbel Softmax Sampling:

We use Gumbel Softmax Sampling to generate the select-or-skip decision for the l-th block in $\mathcal{T}_k$. It substitutes the original non-differentiable sample from a discrete distribution with a differentiable sample with the reparameterization trick:

$$v_{l,k}(j) = \frac{\exp\left((\log \pi_{l,k}(j) + G_{l,k}(j))/\tau\right)}{\sum_{i\in\{0,1\}} \exp\left((\log \pi_{l,k}(i) + G_{l,k}(i))/\tau\right)},$$

Loss Terms:

$$\mathcal{L}_{total} = \sum_k \lambda_k \mathcal{L}_k + \lambda_{sp}\mathcal{L}_{sparsity} + \lambda_{sh}\mathcal{L}_{sharing},$$

$\mathcal{L}_k$ Task-specific loss (e.g. Cross-Entropy for Semantic Segmentation)

$\mathcal{L}_{sharing}$ encourages residual block sharing across tasks to avoid the whole network being split up by tasks with little knowledge shared among them

$$\mathcal{L}_{sharing} = \sum_{k_1,k_2\le K}\sum_{l\le L}\frac{L-l}{L}|\alpha_{l,k_1} - \alpha_{l,k_2}|.$$

$\mathcal{L}_{sparsity}$ enhances the model's compactness by minimizing the log-likelihood of the probability of a block being executed

$$\mathcal{L}_{sparsity} = \sum_{l\le L,k\le K} \log \alpha_{l,k}.$$

Curriculum Learning:

Gradually enlarge the decision space and form a set of learning tasks from easy to hard. Specifically, for the l-th ($l < L$) epoch, we only learn the policy distribution of last l blocks. We then gradually learn the distribution parameters of additional blocks as l increases and learn the joint distribution for all blocks after L epochs.

Experiments

Datasets: NYU v2 (2 or 3 tasks), CityScapes (2 tasks), Tiny-Taskonomy (5 tasks), DomainNet (6 tasks), Text Classification (10 tasks)

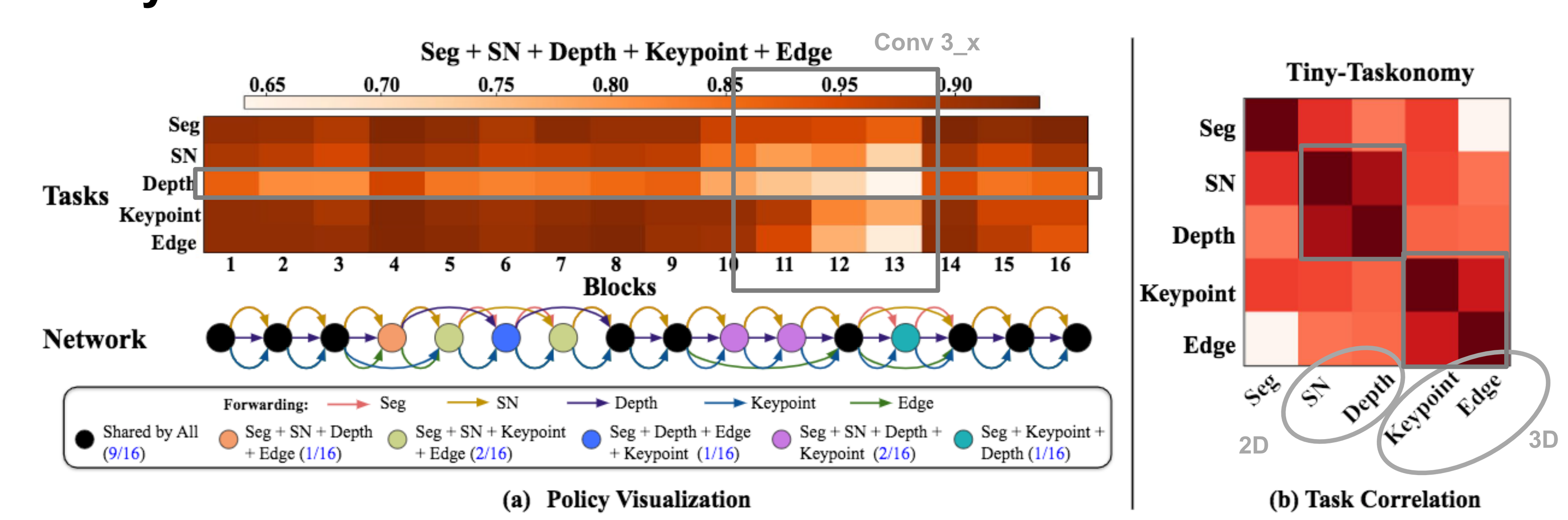
Table 4: **Tiny-Taskonomy 5-Task Learning**. $\mathcal{T}_1$: Semantic Segmentation, $\mathcal{T}_2$: Surface Normal Prediction, $\mathcal{T}_3$: Depth Prediction, $\mathcal{T}_4$: Keypoint Estimation, $\mathcal{T}_5$: Edge Estimation.

Models	# Params ↓	$\Delta_{\mathcal{T}_1}$ ↑	$\Delta_{\mathcal{T}_2}$ ↑	$\Delta_{\mathcal{T}_3}$ ↑	$\Delta_{\mathcal{T}_4}$ ↑	$\Delta_{\mathcal{T}_5}$ ↑	$\Delta_{\mathcal{T}}$ ↑
Multi-Task	**-80.0**	- 3.7	- 1.6	- 4.5	0.0	+ 4.2	- 1.1
Cross-Stitch	0.0	+ 0.9	- 4.0	**0.0**	- 1.0	- 2.4	- 1.3
Sluice	0.0	-3.7	-1.7	-9.1	+ 0.5	+ 2.4	- 2.3
NDDR-CNN	+8.2	-4.2	-1.0	- 4.5	+ 2.0	+ 4.2	- 0.7
MTAN	-9.8	-8.0	- 2.8	- 4.5	0.0	+ 2.8	- 2.5
DEN	-77.6	-28.2	- 3.0	-22.7	+ 2.5	+ 4.2	- 9.4
AdaShare	**-80.0**	**+ 2.3**	**- 0.7**	- 4.5	**+ 3.0**	**+ 5.7**	**+ 1.1**

Single-Task Learning: Seg: 0.575; SN: 0.707; Depth: 0.022; Keypoint: 0.197; Edge: 0.212

AdaShare outperforms SOTA methods with about 50%-80% fewer parameters

Policy Visualization:



(a) Policy Visualization (b) Task Correlation

Observations:

- Not all blocks contribute to the task equally
- More blocks shared only among a sub-group of tasks in ResNet's conv3_x layers, where middle/high-level features (more task-specific) are starting to get captured
- Similar tasks should have similar execution distribution to share knowledge

Computational 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 use55.59G, 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 state-of-the-art methods over all the tasks while achieving better recognition accuracy with about 50%-80% less parameters

References:

- [Cross-Stitch]: Misra et al. Cross-stitch networks for multi-task learning, CVPR 2016.
- [Sluice]: Ruder et al. Latent multi-task architecture learning, AAAI 2019.
- [NDDR-CNN]: Gao et al. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction, CVPR 2019.
- [MTAN]: Liu et al. End-to-end multi-task learning with attention, CVPR 2019.
- [DEN]: Ahn et al. Deep elastic networks with model selection for multi-task learning, ICCV 2019.

Paper Link

Acknowledgements: DARPA Contract No. FA8750-19-C-1001, NSF and IBM