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

Ximeng Sun\textsuperscript{1}, Rameswar Panda\textsuperscript{2}, Rogerio Feris\textsuperscript{2}, Kate Saenko\textsuperscript{1,2}

\textsuperscript{1}Boston University, 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.

Our Contributions

- We learn the feature sharing pattern jointly with the network weights across which tasks
- 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 address 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 $u_{	ext{task}}$ (a.k.a policy) for each layer $l$ and task $T$ that determines whether the $l$-th layer in a deep neural network is selected to execute or skipped when solving $T$ to obtain the optimal sharing pattern, yielding the best overall performance over the task set $T$.

Optimization

Gumbel Softmax Sampling:

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

$$
\alpha_l(j) = \frac{\exp\left[\log(\pi_{l}(j)) + \theta_l(j) / \tau\right]}{\sum_{j'} \exp\left[\log(\pi_{l}(j')) + \theta_l(j') / \tau\right]}
$$

Loss Terms:

- $L_{\text{total}} = \sum_k \lambda_k L_k + \lambda_{\text{sparsity}} + \lambda_{\text{sharing}}$
- $L_k$: Task-specific loss (e.g., Cross-Entropy for Semantic Segmentation)
- $L_{\text{sharing}}$: encourages residual block sharing across tasks to avoid the whole network being split up by tasks with little knowledge shared among them

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: $T_1$: Semantic Segmentation, $T_2$: Surface Normal Prediction, $T_3$: Depth Prediction, $T_4$: Keypoint Estimation, $T_5$: Edge Estimation.

<table>
<thead>
<tr>
<th>Model</th>
<th>$P_{\text{Precision}}$</th>
<th>$P_{\text{Segmentation}}$</th>
<th>$P_{\text{Normal}}$</th>
<th>$P_{\text{Surface}}$</th>
<th>$P_{\text{Depth}}$</th>
<th>$P_{\text{Keypoint}}$</th>
<th>$P_{\text{Edge}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdaShare</td>
<td>1.00</td>
<td>0.90</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
<td>0.95</td>
</tr>
<tr>
<td>Cross-Share</td>
<td>0.95</td>
<td>0.85</td>
<td>0.90</td>
<td>0.90</td>
<td>0.90</td>
<td>0.90</td>
<td>0.90</td>
</tr>
<tr>
<td>AdaShare</td>
<td>0.90</td>
<td>0.80</td>
<td>0.85</td>
<td>0.85</td>
<td>0.85</td>
<td>0.85</td>
<td>0.85</td>
</tr>
</tbody>
</table>

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

Policy Visualization:

Observations:

- Not all blocks contribute to the task equally
- More blocks shared 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 $\text{NYUDv2}$, $\text{NYUDv2}$, $\text{Cross-Slit/Suicide}$, $\text{NNDR}$, $\text{MTAN}$, $\text{DEN}$, and $\text{AdaShare}$ use 37.06G, 38.32G, 44.31G, 39.18G and 33.35G FLOPS and in NYU v2 3 task, they use55.9G, 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:


Acknowledgments: DARPA Contract No. F33615-19-1-8031, NSF and IBM

Paper Link