Which Tasks Should Be Learned Together in Multi-task Learning? (Supplemental)

Abstract

The following items are provided in the supplemental material:

1. A video showing various models' and baselines' predictions for each task in each frame of a YouTube video.
2. Validation and test set results for the single-task and multi-task networks that we trained.

1. Network Selection Algorithm

Algorithm 1 Get Best Networks

Input: $\boldsymbol{C}_r$, a running set of candidate networks, each with an associated cost $c \in \mathbb{R}$ and a performance score for each task the network solves. Initially, $\boldsymbol{C}_r = \boldsymbol{C}_0$
Input: $\boldsymbol{S}_r \subseteq \boldsymbol{C}_0$, a running solution, initially $\emptyset$
Input: $b_r \in \mathbb{R}$, the remaining time budget, initially b

1: **function** GETBESTNETWORKS($\boldsymbol{C}_r, \boldsymbol{S}_r, b_r$)
2: $\quad\boldsymbol{C}_r \leftarrow$ FILTER($\boldsymbol{C}_r, \boldsymbol{S}_r, b_r$)
3: $\quad\boldsymbol{C}_r \leftarrow$ SORT($\boldsymbol{C}_r$) ▷ Most promising networks first
4: $\quad\boldsymbol{Best} \leftarrow \boldsymbol{S}_r$
5: $\quad$**for** $n \in \boldsymbol{C}_r$ **do**
6: $\quad\quad\boldsymbol{C}_r \leftarrow \boldsymbol{C}_r \setminus n$ ▷ $\setminus$ is set subtraction.
7: $\quad\quad\boldsymbol{S}_i \leftarrow \boldsymbol{S}_r \cup \{n\}$
8: $\quad\quad b_i \leftarrow b_r - c_n$
9: $\quad\quad\boldsymbol{Child} \leftarrow$ GETBESTNETWORKS($\boldsymbol{C}_r, \boldsymbol{S}_i, b_i$)
10: $\quad\quad\boldsymbol{Best} \leftarrow$ BETTER($\boldsymbol{Best}, \boldsymbol{Child}$)
11: $\quad$**return** $\boldsymbol{Best}$

12: **function** FILTER($\boldsymbol{C}_r, \boldsymbol{S}_r, b_r$)
13: $\quad$Remove networks from $\boldsymbol{C}_r$ with $c_n > b_r$.
14: $\quad$Remove networks from $\boldsymbol{C}_r$ that cannot improve $\boldsymbol{S}_r$'s performance on any task.
15: $\quad$**return** $\boldsymbol{C}_r$

16: **function** BETTER($\boldsymbol{S}_1, \boldsymbol{S}_2$)
17: $\quad$**if** $C(\boldsymbol{S}_1) < C(\boldsymbol{S}_2)$ **then**
18: $\quad\quad$**return** $\boldsymbol{S}_1$
19: $\quad$**else**
20: $\quad\quad$**return** $\boldsymbol{S}_2$

Algorithm 1 chooses the best subset of networks in our collection, subject to the inference time budget constraint. The algorithm recursively explores the space of solutions and prunes branches that cannot lead to optimal solutions. The recursion terminates when the budget is exhausted, at which point $\boldsymbol{C}_r$ becomes empty and the loop body does not execute.

The sorting step on line 3 requires a heuristic upon which to sort. We found that ranking models based on how much they improve the current solution, $\boldsymbol{S}$, works well. It should be noted that this algorithm always produces an optimal solution, regardless of which sorting heuristic is used. However, better sorting heuristics reduce the running time because subsequent

iterations will more readily detect and prune portions of the search space that cannot contain an optimal solution. In our setup, we tried variants of problems with 5 tasks and 36 networks, and all of them took less than a second to solve.

The definition of the BETTER() function is application-specific. For our experiments, we prefer networks that have the lowest total loss across all five tasks. Other applications may have hard performance requirements for some of the tasks, and performance on one of these tasks cannot be sacrificed in order to achieve better performance on another task. Such application-specific constraints can be encoded in BETTER().

2. Tabular Data

	Ours Optimal	Single 20% pass 5.3.1	Higher Order 5.3.2
1	SDNKE	SDNKE	SDNKE
1.5	DNKE, S	SDNK, E	DNKE, S
2	nKE, SDN	SDke, NKE	DNK, E, S
2.5	nKE, SDn, N	SDke, nKE, N	DNK, E, Sn
3	nKE, SDn, N	SDne, sdke, NKE	DNK, E, Sn
3.5	nKE, Snk, Dnk, N	SDne, sdke, nKE, N	DnK, E, Sn, N
4	nKE, Snk, Dnk, N	SDne, sdke, nKE, N	Sn, DK, E, N
4.5	nKE, Snk, Dnk, N	sDne, sdke, nKE, N, Snk	Sn, E, K, Dn, N
5	nkE, Snk, Dnk, N, K	sDne, sdke, nKE, N, Snk	Sn, E, K, Dn, N

Table 1. **Setting 1: The task groups picked by each of our techniques for every budget choice between 1 and 5.** Networks are shown as a list of letters corresponding to each task the network contains. *S: Semantic Segmentation, D: Depth Estimation, N: Surface Normal Prediction, K: Keypoint Detection, E: Edge Detection*. Capital letters denote that a solution used that network's prediction for that task. Half-sized networks are shown in red.

Time Budget	1	1.5	2	2.5	3	3.5	4	4.5	5
Sener et al.	0.562		0.556	0.551			0.547		
GradNorm	0.515						0.500		
Pessimal Grouping	**0.503**	0.503	0.503	0.503	0.503	0.502	0.499	0.496	0.495
Traditional MTL	**0.503**		0.492	0.487			0.488		
Random Groupings	**0.503**	0.483	0.475	0.471	0.467	0.464	0.462	0.460	0.459
Independent	0.515	0.501	0.477	0.465			0.454		0.448
Ours (ESA) 5.3.1	**0.503**	0.487	0.467	0.461	0.457	0.451	0.451	0.447	0.447
Ours (HOA) 5.3.2	**0.503**	**0.461**	0.455	0.451	0.449	0.445	0.444	0.445	0.442
Ours Optimal	**0.503**	**0.461**	**0.452**	**0.446**	**0.442**	**0.439**	**0.436**	**0.436**	**0.435**

Table 2. Setting 1: The total test set loss on all five tasks for each method under each inference time budget. Lower is better. The data is the same as in Figures 2 and 3.

	SemSeg	Depth	Normals	Keypoints	Edges
S	0.08039	–	–	–	–
D	–	0.1695	–	–	–
N	–	–	0.08591	–	–
K	–	–	–	0.0895	–
E	–	–	–	–	0.02783
SD	0.07858	0.1833	–	–	–
SN	0.074	–	0.0997	–	–
SK	0.07722	–	–	0.09718	–
SE	0.07897	–	–	–	0.04462
DN	–	0.1695	0.09275	–	–
DK	–	0.1706	–	0.09318	–
DE	–	0.1748	–	–	0.03192
NK	–	–	0.08968	0.09181	–
NE	–	–	0.09358	–	0.02908
KE	–	–	–	0.09185	0.03488
SDN	0.07498	0.1698	0.09575	–	–
SDK	0.07699	0.1782	–	0.09704	–
SDE	0.07893	0.1863	–	–	0.04559
SNK	0.0722	–	0.09919	0.0961	–
SNE	0.07222	–	0.0982	–	0.03689
SKE	0.0766	–	–	0.09342	0.03508
DNK	–	0.1654	0.09358	0.09253	–
DNE	–	0.1708	0.09396	–	0.03286
DKE	–	0.1793	–	0.09073	0.02937
NKE	–	–	0.09626	0.09024	0.02609
SDNK	0.07762	0.1822	0.09869	0.1015	–
SDNE	0.07576	0.1735	0.09718	–	0.04513
SDKE	0.0795	0.1797	–	0.09272	0.04141
SNKE	0.07369	–	0.09944	0.09697	0.03312
DNKE	–	0.1708	0.09392	0.09334	0.02803
SDNKE	0.07854	0.1864	0.1	0.09814	0.04453

Table 3. **Setting 1: The validation set performance of our 31 networks on each task that they solve.** Tasks are named to contain a letter for each task that they solve. *S: Semantic Segmentation, D: Depth Estimation, N: Surface Normal Prediction, K: Keypoint Detection, E: Edge Detection*.

	SemSeg	Depth	Normals	Keypoints	Edges
S	0.07662	–	–	–	–
D	–	0.1696	–	–	–
N	–	–	0.08555	–	–
K	–	–	–	0.08847	–
E	–	–	–	–	0.0275
SD	0.07419	0.1831	–	–	–
SN	0.07084	–	0.0994	–	–
SK	0.07369	–	–	0.09601	–
SE	0.07504	–	–	–	0.044
DN	–	0.1694	0.09249	–	–
DK	–	0.1713	–	0.08882	–
DE	–	0.1753	–	–	0.03145
NK	–	–	0.08934	0.09077	–
NE	–	–	0.09327	–	0.02865
KE	–	–	–	0.09077	0.0344
SDN	0.07193	0.17	0.09544	–	–
SDK	0.07311	0.1785	–	0.09591	–
SDE	0.07617	0.1865	–	–	0.04474
SNK	0.06933	–	0.09966	0.09302	–
SNE	0.06859	–	0.09796	–	0.03625
SKE	0.07323	–	–	0.09232	0.03463
DNK	–	0.1658	0.09318	0.09143	–
DNE	–	0.1706	0.09362	–	0.03239
DKE	–	0.1795	–	0.08968	0.02887
NKE	–	–	0.09596	0.08921	0.02566
SDNK	0.07338	0.1826	0.09836	0.1003	–
SDNE	0.07249	0.1739	0.09689	–	0.04441
SDKE	0.07634	0.1801	–	0.09157	0.04097
SNKE	0.07111	–	0.09941	0.09464	0.03328
DNKE	–	0.1704	0.09356	0.09226	0.02768
SDNKE	0.07603	0.186	0.09976	0.09704	0.04395

Table 4. **Setting 1: The test set performance of our 31 networks on each task that they solve.**