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

Trevor Standley [1] **Amir Zamir** [2] **Dawn Chen** [3] **Leonidas Guibas** [1] **Jitendra Malik** [4] **Silvio Savarese** [1]
http://taskgrouping.stanford.edu/

Abstract

Many computer vision applications require solving multiple tasks in real-time. A neural network can be trained to solve multiple tasks simultaneously using *multi-task learning*. This can save computation at inference time as only a single network needs to be evaluated. Unfortunately, this often leads to inferior overall performance as task objectives can compete, which consequently poses the question: **which tasks should and should not be learned together in one network when employing multi-task learning?** We study task cooperation and competition in several different learning settings and propose a framework for assigning tasks to a few neural networks such that cooperating tasks are computed by the same neural network, while competing tasks are computed by different networks. Our framework offers a time-accuracy trade-off and can produce better accuracy using less inference time than not only a single large multi-task neural network but also many single-task networks.

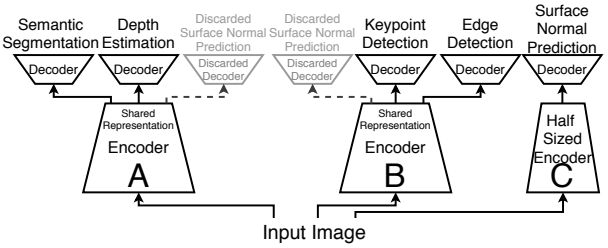

Figure 1. **Given five example tasks to solve, there are many ways that they can be split into task groups for multi-task learning. How do we find the best one?** We propose a computational framework that, for instance, suggests the following grouping to achieve the lowest total loss, using a computational budget of 2.5 units: train network A to solve Semantic Segmentation, Depth Estimation, and Surface Normal Prediction; train network B to solve Keypoint Detection, Edge Detection, and Surface Normal Prediction; train network C with a less computationally expensive encoder to solve Surface Normal Prediction alone; including Surface Normals as an output in the first two networks were found advantageous for improving the other outputs, while the best Normals were predicted by the third network. This task grouping outperforms all other feasible ones, including learning *all five tasks in one large network* or using *five dedicated smaller networks*.

1. Introduction

Many applications, especially robotics and autonomous vehicles, are chiefly interested in using multi-task learning to reduce the inference time required to estimate many characteristics of visual input. For example, an autonomous vehicle may need to detect the location of pedestrians, determine a per-pixel depth, and predict objects' trajectories, all within tens of milliseconds. In multi-task learning, multiple tasks are solved at the same time, typically with a single neural network. In addition to reduced inference time, solving a set of tasks jointly rather than independently can, in theory, have other benefits such as improved prediction accuracy, increased data efficiency, and reduced training time.

Unfortunately, the quality of predictions are often observed to suffer when a network is tasked with making multiple predictions due to a phenomenon called *negative transfer*. In fact, multi-task performance can suffer so much that smaller independent networks are often superior. This may be because the tasks must be learned at different rates. Or because one task may dominate the learning leading to poor performance on other tasks. Furthermore, task gradients may interfere and multiple summed losses may make the optimization landscape more difficult.

On the other hand, when task objectives do not interfere much with each other, performance on both tasks can be maintained or even improved when jointly trained. Intuitively, this loss or gain of quality seems to depend on *the relationship between the jointly trained tasks*. We empirically study these relationships in depth.

Prior work has studied the relationship between tasks for transfer learning (Zamir et al., 2018). However, we find that transfer relationships are not highly predictive of multi-task relationships. In addition to studying multi-task relation-

[1]Stanford University [2]Swiss Federal Institute of Technology (EPFL) [3]Google Inc. [4]The University of California, Berkeley. Correspondence to: Trevor Standley <tstand@cs.stanford.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

ships, we attempt to determine how to produce good prediction accuracy under a limited inference time budget by assigning competing tasks to separate networks and cooperating tasks to the same network.

More concretely, this leads to the following problem: Given a set of tasks, $\mathcal{T}$, and a computational budget b (e.g., maximum allowable inference time), what is the optimal way to assign tasks to networks with combined cost $\leq b$ such that a combined measure of task performances is maximized?

To this end, we develop a computational framework for choosing the best tasks to group together in order to have a small number of separate neural networks that completely cover the task set and that maximize task performance under a given computational budget. We make the intriguing observation that the inclusion of an additional task in a network can potentially improve the accuracy of the other tasks, even though the performance of the added task might be poor. This can be viewed as *guiding* the loss of one task by adding an additional loss, for instance similar to curriculum learning (Bengio et al., 2009). Achieving this, of course, depends on picking the proper auxiliary task to be added – our system can take advantage of this phenomenon, as schematically shown in Figure 1.

This paper has two main contributions. First, in Section 4, we provide an empirical study of a number of factors that influence multi-task learning including network size, dataset size, and how tasks influence one another when learned together. Then, in Section 5, we outline a framework for assigning tasks to networks in order to achieve the best total prediction accuracy with a limited inference-time budget. We analyze the results and show that selecting the best assignment of tasks to groups is critical for good performance.

2. Prior Work

Multi-Task Learning: See (Zhang & Yang, 2017) for a good overview of multi-task learning. The authors of (Ruder, 2017) identify two clusters of contemporary techniques hard parameter sharing and soft parameter sharing.

In hard parameter sharing, most or all of the parameters in the network are shared among all tasks. A known contemporary example of hard parameter sharing in computer vision is UberNet (Kokkinos, 2017) which tackles 7 computer vision problems using hard parameter sharing. The author focuses on reducing the computational cost of training for hard parameter sharing, but experiences a rapid degradation in performance as more tasks are added to the network. Hard parameter sharing is also used in many other works such as (Thrun, 1996; Caruana, 1997; Nekrasov et al., 2018; Dvornik et al., 2017; Bilen & Vedaldi, 2016; Pentina & Lampert, 2017; Doersch & Zisserman, 2017; Zamir et al., 2016; Long et al., 2017; d. Miranda et al., 2012; Zhou et al., 2018; Rudd et al., 2016; Leang et al., 2020; Chennupati et al., 2019; Suteu & Guo, 2019).

Other works, such as (Sener & Koltun, 2018) and (Chen et al., 2018b), aim to dynamically re-weight each task's loss during training. The former work finds weights that provably lead to a Pareto-optimal solution, while the latter attempts to find weights that balance the influence of each task on network weights. (Kendall et al., 2018) uses uncertainty to weight tasks. (Gong et al., 2019) and (Leang et al., 2020) compare loss weighting strategies. They both find no clear winner and similar performance among strategies, including a uniform weighting strategy. Similarly to us, they find that multi-task learning is often inferior to single task learning with multiple networks.

In soft or partial parameter sharing, either there is a separate set of parameters per task, or a significant fraction of the parameters are unshared. The models are tied together either by information sharing or by requiring parameters to be similar. Examples include (Dai et al., 2016; Duong et al., 2015; Misra et al., 2016; Tessler et al., 2017; Yang & Hospedales, 2017; Lu et al., 2017).

A canonical example of soft parameter sharing can be seen in (Duong et al., 2015). The authors are interested in designing a deep dependency parser for languages such as Irish that do not have much treebank data available. They leverage the abundant data from other languages and tie the weights of two networks together by adding an L2 distance penalty between corresponding weights and show substantial improvement for the target language.

Another example of soft parameter sharing is Cross-stitch Networks (Misra et al., 2016). Starting with separate networks for two tasks, the authors add 'cross-stitch units' between them, which allow each network to peek at the other network's hidden layers. This approach reduces but does not eliminate task interference, and the overall performance is less sensitive to the relative loss weights.

(Liu et al., 2019) adds a per-task attention layer after each layer, and (Maninis et al., 2019) adds task specific residual paths and adversarial training to match gradients. Both run the network separately for each task.

Finally, (Bingel & Søgaard, 2017) studies task interaction for NLP, and (Zamir et al., 2020) makes use of task relationships to improve the predictions for related tasks.

Unlike our method, none of the aforementioned works attempt to discover good groups of tasks to train together. Also, soft parameter sharing does not reduce inference time, a major goal of ours.

Hybrid approaches such as PathNet (Fernando et al., 2017), and AdaShare (Sun et al., 2019) attempt to learn what to share and what not to. These works do not attempt to dis-

cover good pairs of source and target tasks nor to meet an inference time budget. Most importantly, they cannot achieve much inference-time speedup from module reuse.

Transfer Learning: Transfer learning (Pratt, 1993; Helleputte & Dupont, 2009; Silver & Bennett, 2008; Finn et al., 2016; Mihalkova et al., 2007; Niculescu-Mizil & Caruana, 2007; Luo et al., 2017; Razavian et al., 2014; Pan & Yang, 2010; Mallya & Lazebnik, 2018; Fernando et al., 2017; Rusu et al., 2016) is similar to multi-task learning in that solutions are learned for multiple tasks. Unlike multi-task learning, however, transfer learning methods often assume that a model for a source task is given and then adapt that model to a target task. Transfer learning methods generally neither seek any benefit for source tasks nor a reduction in inference time as their main objective.

Neural Architecture Search (NAS): Many recent works search the space of network architectures to find ones that perform well (Zoph et al., 2018; Zoph & Le, 2017; Liu et al., 2018; Pham et al., 2018; Xie et al., 2019; Elsken et al., 2019; Zhou et al., 2019; Baker et al., 2017; Real et al., 2019). This is related to our work as we search the space of task groupings. Just as with NAS, the computationally found groupings often perform better than human-engineered ones.

Task Relationships: Our work is related to *Taskonomy* (Zamir et al., 2018) which studied the relationships between visual tasks for *transfer learning* and introduced a dataset with over 4 million images and corresponding labels for 26 tasks. A number of recent works further analyzed task relationships (Pal & Balasubramanian, 2019; Dwivedi & Roig., 2019; Achille et al., 2019; Wang et al., 2019) for transfer learning. While they extract relationships between these tasks for *transfer learning*, we are interested in the *multi-task learning* setting. Interestingly, we find notable differences between *transfer task affinity* and *multi-task affinity*. Their method also differs in that they are interested in labeled-data efficiency and not inference-time efficiency. Finally, the transfer quantification approach taken by Taskonomy (readout functions) is only capable of finding relationships between the high-level bottleneck representations developed for each task, whereas structural similarities between tasks at all levels are potentially relevant for multi-task learning.

3. Experimental Setup

Dataset: We perform our study using the Taskonomy dataset (Zamir et al., 2018), which is currently the largest multi-task dataset for computer vision with diverse tasks. The data was obtained from 3D scans of about 600 buildings. The dataset has about 4 million examples, which we divided into about 3.9 million training instances (200k for Setting 3), about 50k validation instances, and about 50k test instances. There was no overlap in the buildings that appeared in the training and test sets.

Task Sets: We have selected two sets of five tasks each from this dataset. **Task Set 1** includes *Semantic Segmentation, Depth Estimation, Surface Normal Prediction, SURF Keypoint Detection*, and *Canny Edge Detection*. One semantic task, two 3D tasks, and two 2D tasks are included. These tasks were chosen to be representative of major task categories, but also to have enough overlap in order to test the hypothesis that similar tasks will train well together. **Task Set 2** includes *Auto Encoder, Surface Normal Prediction* again, *Occlusion Edges, Reshading*, and *Principal Curvature*. For a detailed definition of each of these tasks, see the Supplemental Materials for the Taskonomy paper. Cross-entropy loss was used for Semantic Segmentation, while an $L1$ loss was used for all other tasks.

Architectures: In all experiments, we used a standard encoder-decoder architecture with a modified Xception (Chollet, 2017)) encoder. Our choice of architecture is not critical and was chosen for its reasonably low inference time. All max-pooling layers were replaced by 2×2 convolution layers with a stride of 2, similar to (Chen et al., 2018a). This network has 16.5 million parameters and requires 6.4 billion multiply-adds per image. The input image size for all networks is 256×256.

In order to study the effect of network size on task relationships, we also define a smaller **Xception17** network. For Xception17, the Xception network encoder was simplified to have 17 layers and the middle flow layers were reduced to having 512 rather than 728 channels, resulting in about 4 million parameters. This corresponds to 2.28 billion multiply-adds.

Our decoders were designed to be lightweight, with four transposed convolutional layers (Noh et al., 2015) and four convolutional layers.

Settings: We run our experiments under four settings.

	Network	Training Set Size	Task Set	Purpose
Setting 1	**Xception17**	3.9 million	Task Set 1	Network Size
Setting 2	Xception	3.9 million	Task Set 1	Control
Setting 3	Xception	**200 thousand**	Task Set 1	Dataset Size
Setting 4	Xception	3.9 million	**Task Set 2**	Task Set

We test the effect of network size on multi-task learning with **Setting 1**. It uses a smaller and less deep network than the other settings. **Setting 2** is the control. It uses a large network, the full dataset, and Task Set 1. We test the effect of dataset size on multi-task learning with **Setting 3**. Here we limit ourselves to only 200 thousand training instances. Finally, we test other task relationships with **Setting 4**, which uses Task Set 2.

Training Details: All training was done using PyTorch (Paszke et al., 2017) with Apex for fp16 acceleration (Mi-

cikevicius et al., 2017).

The training loss we used was the unweighted mean of the losses for the included tasks. Networks were trained with an initial learning rate of 0.1, which was reduced by half every time the training loss stopped decreasing. Networks were trained until their validation loss stopped improving. The network with the highest validation loss (checked after each epoch of 20% of our data) was saved. No hyper-parameter search was conducted.

Comparison: We define a computational network cost unit, the Standard Network Time (SNT). In Setting 1, an SNT is the number of multiply-adds in Xception17. In the other settings, an SNT is the number of multiply-adds in the normal Xception network. In order to produce our smaller/larger models to compare within each setting, we shrunk/grew the number of channels in every layer of the encoder such that it had the appropriate number of multiply-adds.

Trained Networks: In each setting, we train a 1-SNT network for each of the $2^n - 1$ feasible subsets of our setting's 5 tasks. Thus, we train 5 single-task networks, 10 two-task networks, 10 three-task networks, 5 four-task networks, and a single five-task network. Another five single-task networks were trained, each having a half-size ($1/2$-SNT) encoder and a standard decoder. Finally, we trained a number of fractional-SNT single-task networks as comparisons and baselines.

4. Study of Task Relationships

We study a number of factors that influence a multi-task network's performance. We look at the relationships between tasks in each setting, and compare them to the relationships from our other settings.

Setting 1: The smaller network, Xception17.

First, we analyze how the number of tasks included in a multi-task network affects performance. Table 1 shows the performance of multi-task learning relative to independent training.

	1 task	2 tasks	3 tasks	4 tasks	5 tasks
1-SNT each	0.00%	-7.56%	-7.23%	-10.69%	-19.00%
1-SNT total	0.00%	-1.15%	4.23%	4.86%	0.34%

Table 1. **Performance of multi-task learning *relative* to independent (i.e. single-task) training.** For example, when a 4-task network is compared with four 1-SNT single-task networks, the 4-task network sees a 10.69% worse total loss on average. When the same network is compared to four $1/4$-SNT networks, the total loss is 4.86% better on average.

In Setting 1, multi-task networks do not compare favorably to multiple single-task networks that are each allowed the same computational budget as the single multi-task network.

		Relative Performance On					
		SemSeg	Depth	Normals	Keypoints	Edges	Average
Trained With	SemSeg	–	-5.41%	-11.29%	-4.32%	-34.64%	-13.92%
	Depth	4.17%	–	-3.55%	3.49%	3.76%	1.97%
	Normals	8.50%	2.48%	–	1.37%	12.33%	6.17%
	Keypoints	4.82%	1.38%	-0.02%	–	-5.26%	0.23%
	Edges	3.07%	-0.92%	-4.42%	1.37%	–	-0.23%
	Average	5.14%	-0.62%	-4.82%	0.48%	-5.95%	-1.15%

Table 2. **Pairwise multi-task relationships in Setting 1.** The table lists the performance of every task when trained as a pair with every other task. For instance, when SemSeg (Semantic Segmentation) is trained with Depth, SemSeg performs 4.17% better than when SemSeg is trained alone on a $1/2$-SNT network.

However, when the single-task networks are shrunk so that they have the same total budget as the multi-task network, multi-task networks with 3, 4, or 5 tasks outperform the single-task networks on average. Nevertheless, two-task networks still do not compare favorably. Table 2 gives a more detailed view of these ten two-task networks, showing the performance on each task when it is trained with each other task relative to when it is trained alone using a $1/2$-SNT network. We see that the Normals task helps the performance of every other task with which it is trained.

	Depth	Normals	Keypoints	Edges
SemSeg	-0.62%	-1.39%	0.25%	-15.78%
Depth		-0.54%	2.43%	1.42%
Normals			0.67%	3.95%
Keypoints				-1.95%

Table 3. **The *multi-task* learning affinity between pairs of tasks for Setting 1.** These values show the average change in the performance of two tasks when trained as a pair, relative to when they are trained separately.

	Depth	Normals	Keypoints	Edges
SemSeg	1.740%	1.828%	0.723%	0.700%
Depth		1.915%	0.406%	0.468%
Normals			0.089%	0.118%
Keypoints				0.232%

Table 4. **The *transfer* learning affinities** between pairs of tasks according to the authors of Taskonomy (Zamir et al., 2018). Forward and backward transfer affinities are averaged.

In order to determine the between-task affinity for multi-task learning, we took the average of our first-order relationships matrix (Table 2) and its transpose. The result is shown in Table 3. The pair with the highest affinity by this metric are Surface Normal Prediction and 2D Edge Detection. Quite surprisingly, our two 3D tasks, Depth Estimation and Surface Normal Prediction, do not score highly on this similarity metric. This contrasts with the findings for transfer learning in Taskonomy (Table 4), for which they have the highest affinity. In fact, there seems to be no correlation between multi-task and transfer learning affinities (tables 3 and 4) in this setting (Pearson's r is -0.12, $p = 0.74$) .

Setting 2: The control setting.

We test the effect of network capacity on multi-task affinity, by retraining all of our networks using a higher-capacity encoder. We used the full Xception network (Chollet, 2017) this time, which uses 6.4 billion multiply-adds. The resulting data allows us to generate a new version of Table 2 for Setting 2 as Table 5.

		Relative Performance On					
		SemSeg	Depth	Normals	Keypoints	Edges	Average
Trained With	SemSeg	–	3.00%	-2.79%	-5.20%	27.80%	5.70%
	Depth	1.72%	–	1.18%	-3.52%	25.73%	6.28%
	Normals	10.81%	7.12%	–	88.98%	71.59%	44.62%
	Keypoints	3.12%	-0.41%	-10.12%	–	61.07%	13.42%
	Edges	0.03%	-1.40%	-4.78%	-3.05%	–	-2.30%
		3.92%	2.08%	-4.13%	19.30%	46.54%	13.54%

Table 5. **Pairwise multi-task relationships in Setting 2.**

We can see by comparing Table 5 with Table 2 that tasks are much more likely to benefit from being trained together when using the larger network. However, there are still tasks that both suffer when trained together. Furthermore, the values in Table 5 do not seem to correlate with the values in Table 2 (Pearson's $r = 0.08$). The affinities also do not seem to correlate with Taskonomy's transfer affinities (Pearson's $r = -0.14$). These results stress the necessity of using a automatic framework each particular setup to determine which tasks to train together.

Setting 3: Using only 5% of the available training data.

We study the effect of training set size by using only 199,498 training instances (420 from each training building). This amount of data is more similar to other multi-task datasets in the literature. One common assumption is that multi-task learning is likely to be better in low-data scenarios, as MTL effectively allows you to pool supervision. However, we see this assumption violated in Table 6, as most tasks suffer when trained with another task in this setting, especially when compared to Setting 2. Furthermore, were it not for the huge gains in the *Edges* task in a couple of cases, MTL would be deleterious on average.

		Relative Performance On					
		SemSeg	Depth	Normals	Keypoints	Edges	Average
Trained With	SemSeg	–	1.91%	-6.00%	-9.91%	-21.93%	-8.98%
	Depth	-12.63%	–	2.95%	1.44%	-9.70%	-4.48%
	Normals	8.32%	15.38%	–	-1.35%	52.08%	18.61%
	Keypoints	-5.84%	-7.21%	-2.26%	–	55.63%	10.08%
	Edges	-5.62%	6.02%	-4.16%	-5.02%	–	-2.20%
		-3.95%	4.03%	-2.37%	-3.71%	19.02%	2.6%

Table 6. **Pairwise multi-task relationships in Setting 3.**

Although we did not find correlations between the task relationships for the low-capacity model and those for the high-capacity model, the low-data relationships show a positive correlation with both. The small-data relationships correlate with the low-capacity relationships (Pearson's $r = +0.375, p = 0.10$) as well as the high-capacity relationships (Pearson's $r = +0.558, p = 0.01$). However, these affinities still do not correlate highly with Taskonomy's transfer-learning affinities (Pearson's $r = -0.235, p = 0.51$).

Setting 4: Using Task Set 2.

Aside from the Auto Encoder task, the other four tasks in Task Set 2 nearly all have a moderate positive influence on one another. These four tasks are all very similar 3D tasks, so it seems that similar tasks can work well with one another in some situations. Unfortunately, the Auto Encoder task hurts the performance of the other tasks on average. Once again, we find no correlation with Taskonomy's transfer affinity (Pearson's $r = -0.153, p = 0.674$).

		Relative Performance On					
		AutoEnc	Normals	Occ Edges	Reshading	Curvature	Average
Trained With	AutoEnc	–	-3.23%	-2.66%	0.10%	-1.39%	-1.79%
	Normals	19.31%	–	3.16%	4.60%	1.95%	7.25%
	Occ Edges	35.83%	-0.25%	–	1.15%	0.84%	9.39%
	Reshading	-24.46%	3.71%	3.16%	–	1.88%	-3.93%
	Princ Curv	10.69%	2.61%	2.46%	3.15%	–	4.73%
		10.34%	0.71%	1.53%	2.25%	0.82%	3.13%

Table 7. **Pairwise multi-task relationships in Setting 4.**

Key Takeaways: Ideally, the relationships between tasks would be independent of the learning setup. We find that this is not the case, therefore using a computational approach like ours for finding task affinities and groupings seems necessary. We saw that both network capacity and the amount of training data influence multi-task task affinity. We also found that more similar tasks don't necessarily train better together. We also find no correlation between multi-task task affinity and transfer learning task affinity in any setting.

Finally, the Normals task seems to improve the performance of the tasks it is trained with. In 15 out of 16 models that were trained with Normals, the other task improved. Furthermore, this improvement was better than the effect of co-training with any different task 13 out of 15 times. This may be because Normals have uniform values across surfaces and preserve 3D edges. However, the Normals task itself tends to perform worse when trained with another task.

We see that choosing which tasks to learn together is critical for achieving good performance. Now, we turn to the study of how to find the best tasks to train together.

5. Task Grouping Framework

Overview: Our goal is to find a set of networks, each of which is trained on a subset of the tasks, that results in the best overall loss within a given computational budget. We do this by considering the space of all possible task subsets, training a network for each subset, and then using each network's performance to choose the best networks that fit within the budget. Because fully training a network for each subset may be prohibitively expensive, we outline two

strategies for predicting training outcomes as well (Sec. 5.3).

Formal Problem Definition: We want to minimize the overall loss on a set of tasks $\mathcal{T} = \{t_1, t_2, ..., t_k\}$ given a limited inference time budget, b, which is the total amount of time we have to complete all tasks. Each neural network that solves some subset of $\mathcal{T}$ and that could potentially be a part of the final solution is denoted by n. It has an associated inference time cost, c_n, and a loss for each task, $\mathcal{L}(n, t_i)$ (which is ∞ for each task the network does not attempt to solve). A solution $\boldsymbol{S}$ is a set of networks that together solve all tasks. The computational cost of a solution is $\text{cost}(\boldsymbol{S}) = \sum_{n \in \boldsymbol{S}} c_n$. The loss of a solution on a task, $\mathcal{L}(\boldsymbol{S}, t_i)$, is the lowest loss on that task among the solution's networks[1], $\mathcal{L}(\boldsymbol{S}, t_i) = \min_{n \in \boldsymbol{S}} \mathcal{L}(n, t_i)$. The overall performance for a solution is $\mathcal{L}(\boldsymbol{S}) = \sum_{t_i \in \mathcal{T}} \mathcal{L}(\boldsymbol{S}, t_i)$.

We want to find the solution with the lowest overall loss and a cost that is under our budget, $\boldsymbol{S}_b = \text{argmin}_{\boldsymbol{S}:\text{cost}(\boldsymbol{S}) \leq b} \mathcal{L}(\boldsymbol{S})$.

5.1. Which Candidate Networks to Consider?

For a given task set $\mathcal{T}$, we wish to determine not just how well each *pair* of tasks performs when trained together, but also how well each *combination* of tasks performs together so that we can capture higher-order task relationships. To that end, our candidate set of networks contains all $2^{|\mathcal{T}|} - 1$ possible groupings: $\binom{|\mathcal{T}|}{1}$ networks with one task, $\binom{|\mathcal{T}|}{2}$ networks with two tasks, $\binom{|\mathcal{T}|}{3}$ networks with three tasks, etc. For the five tasks we use in our experiments, this is 31 networks, of which five are single-task networks.

The size of the networks is another design choice, and to somewhat explore its effects we also include 5 single task networks each with half of the computational cost of a standard network. This brings our total up to 36 networks.

In principle, one could include additional candidate networks of arbitrarily varying computational cost, and let the framework below decide which are worth using. It may even be advantageous to throw in networks trained with different architectures, task weights or training strategies. We do not explore this.

5.2. Network Selection

Consider the situation in which we have an initial candidate set $\boldsymbol{C}_0 = \{n_1, n_2, ..., n_m\}$ of **fully-trained** networks that each solve some subset of our task set $\mathcal{T}$. Our goal is to choose a subset of $\boldsymbol{C}_0$ that solve all the tasks with total inference time under budget b and the lowest overall loss. More formally, we want to find a solution $\boldsymbol{S}_b = \text{argmin}_{\boldsymbol{S} \subseteq \boldsymbol{C}_0 : \text{cost}(\boldsymbol{S}) \leq b} \mathcal{L}(\boldsymbol{S})$.

It can be shown that solving this problem is NP-hard in general (reduction from SET-COVER). However, many techniques exist that can optimally solve *most* reasonably-sized instances of problems like these in acceptable amounts of time. All of these techniques produce the same solutions. We chose to use a branch-and-bound-like algorithm for finding our optimal solutions (pseudo code for the algorithm is in the supplemental material, and our implementation is on GitHub), but in principle the exact same solutions could be achieved by other optimization methods, such as encoding the problem as a binary integer program (BIP) and solving it in a way similar to Taskonomy (Zamir et al., 2018).

Most contemporary MTL works use fewer than 4 unique task types, however, using synthetic inputs, we found that our branch-and-bound like approach requires less time than network training for all $2^{|\mathcal{T}|} - 1 + |\mathcal{T}|$ candidates for fewer than ten tasks.

5.3. Approximations for Reducing Training Time

This section describes two techniques for reducing the training time required to obtain a collection of networks as input to the network selection algorithm. Our goal is to produce task groupings with results similar to the ones produced by the complete search, but with less training time burden. Both techniques involve predicting the performance of a network without actually training it to convergence. The first technique involves training each of the networks for a short amount of time, and the second involves inferring how networks trained on more than two tasks will perform based on how networks trained on two tasks perform.

5.3.1. EARLY STOPPING PRIOR TO CONVERGENCE

We found a moderately high correlation (Pearson's $r = 0.49$) between the validation loss of our 36 candidate neural networks in Setting 1 after a pass through just 20% of our data and the final validation loss of the fully trained networks. This implies that the task relationship trends stabilize early. We find that we can get decent results by running network selection on the lightly trained networks, and then simply training the chosen networks to convergence. This is a common technique in hyperparameter optimization, (Domhan et al., 2015; Li et al., 2017).

For our setup, this technique reduces the training time burden by about **20x** over fully training all candiate networks. Obviously, this technique does come with a prediction accuracy penalty. Because the correlation between early network performance and final network performance is not perfect, the decisions made by network selection are no longer guaranteed to be optimal once networks are trained to convergence. We call this approximation the Early Stop-

[1] In principle, it may be possible to create an even better-performing ensemble when multiple networks solve the same task, though we do not explore this.

ping Approximation (ESA) and present the results of using this technique in Section 6.

5.3.2. Predict Higher-Order From Lower-Order

Do the performances of a network trained with tasks A and B, another trained with tasks A and C, and a third trained with tasks B and C tell us anything about the performance of a network trained on tasks A, B, and C? As it turns out, the answer is yes. Although this ignores complex task interactions and nonlinearities, a simple average of the first-order networks' accuracies was a good indicator of the accuracy of a higher-order network. For example, if you have networks, a&b with losses 0.1&0.2, b&c with 0.3&0.4, and a&c with 0.5&0.6, the per-task loss estimate for a network with a&b&c would be $a = (0.1 + 0.6)/2 = 0.35$, $b = (0.2 + 0.3)/2 = 0.25$ and $c = (0.4 + 0.6)/2 = 0.5$.

Using this strategy, we can predict the performance of all networks with three or more tasks using the performance of all of the fully trained two task networks. First, simply train all networks with two or fewer tasks to convergence. Then predict the performance of higher-order networks, run network selection on both the trained and the predicted networks, then train the higher order networks from scratch.

With our setup, this strategy saves only about 45% of the training time, compared with 95% for the early stopping approximation, and it still comes with a prediction quality penalty. However, this technique requires only a quadratic number of networks to be trained rather than an exponential number, and would therefore theoretically win out when the number of tasks is large.

We call this strategy the Higher Order Approximation (HOA), and present its results in Section 6.

6. Task Grouping Evaluation

We applied our framework and approximations in all four settings. We computed solutions for inference time budgets from 1 SNT to 5 SNT at increments of $1/2$ SNT. The performance scores used for network selection were calculated on the validation set. Each solution chosen was evaluated on the test set. Note that the total loss numbers reported cannot be properly compared between settings because of small differences in label normalization and loss definitions.

Baselines: We compared our results with conventional methods, such as five single-task networks and a single network with all tasks trained jointly.

For Setting 1, we also compared with two multi-task methods in the literature. The first one is (Sener & Koltun, 2018). We found that their algorithm under-weighted the Semantic Segmentation task too aggressively, leading to poor performance on the task and poor performance overall compared to a simple sum of task losses. We speculate that this is because the loss for Semantic Segmentation behaves differently from the other losses. Next we compared to GradNorm (Chen et al., 2018b). GradNorm's results were also slightly worse than classical MTL with uniform task weights, though that may be because we evaluated with uniform task weights, while GranNorm optimized for task weights that it computed. In any event, these techniques are orthogonal to ours and can be used in conjunction for situations in which they lead to better solutions than simply summing losses.

Each baseline was evaluated with multiple encoder sizes (more or fewer channels per layer) so that the results could be compared at many inference time budgets.

Finally, we compared our results to two control baselines illustrative of the importance of making good choices about which tasks to train together, 'Random' and 'Pessimal.' 'Random' is a solution consisting of valid random task groupings that solve our five tasks. The reported values are the average of one million random trials. 'Pessimal' is a solution in which we choose the networks that lead to the worst overall performance, though the solution's performance on each task is still the best among the networks that solve that task.

Setting 1: Figure 3 shows the task groups that were chosen for each technique, and Figure 2 shows the performance of these groups along with those of our baselines. We can see that each of our methods outperforms the traditional baselines for every computational budget.

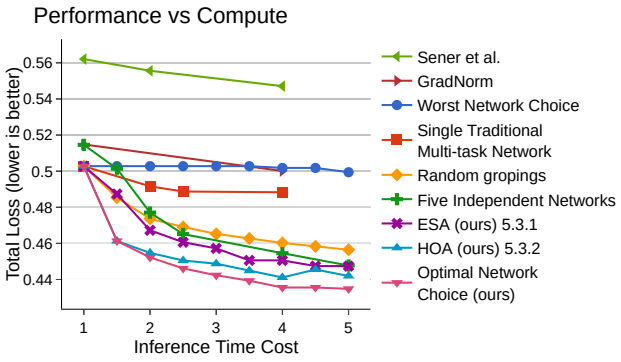

Figure 2. **Performance/inference time trade-off for various methods in Setting 1.** We do not report error bars because the test set is large enough that standard errors are too small to be shown.

When the computational budget is only 1 SNT, all of our methods must select the same model—a traditional multi-task network with a 1 SNT encoder and five decoders. This strategy outperforms GradNorm (Sener & Koltun, 2018), as well as independent networks. However, solutions that utilize multiple networks outperform this traditional strategy for every budget > 1.5—better performance can always be achieved by grouping tasks according to their compatibility.

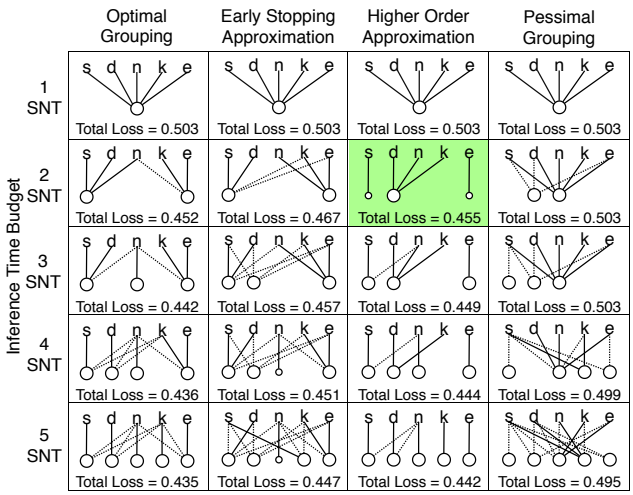

Figure 3. **The task groups picked by each of our techniques for integer budgets between 1 and 5.** Networks are shown as ○ (full-size) or ∘ (half-size). Networks are connected to the tasks for which they compute predictions. *s: Semantic Segmentation, d: Depth Estimation, n: Surface Normal Prediction, k: Keypoint Detection, e: Edge Detection*. Dotted edges represent unused decoders. For example, the green highlighted solution consists of two half-size networks and a full-size network. The full-size network solves Depth Estimation, Surface Normal Prediction, and Keypoint Detection. One half-size network solves Semantic Segmentation and the other solves Edge Detection. The total loss for all five tasks is 0.455. The groupings for fractional budgets are shown in the supplemental material.

When the computational budget is effectively unlimited (5 SNT), our optimal method picks five networks, each of which is used to make predictions for a separate task. However, three of the networks are trained with three tasks each, while only two are trained with one task each. This shows that the networks learned through multi-task learning were found to be best for three of our tasks (s, d, and e), whereas two of our tasks (n and k) are best solved individually.

We also see that our optimal technique using 2.5 SNT and our Higher Order Approximation using 3.5 SNT can both outperform individual networks using 5 SNT total.

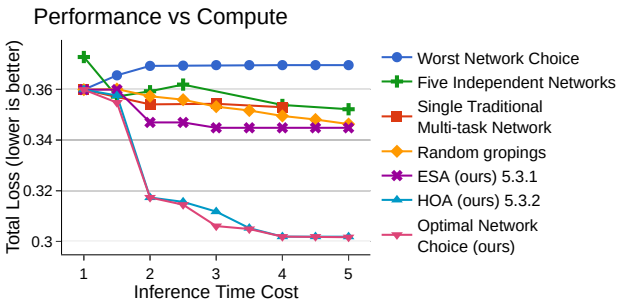

Figure 4. **Performance/inference time trade-off in Setting 2.**

Setting 2: When we apply our network selection framework on the performance data from this high-capacity network (Figure 4) we again see that our method outperforms both training an individual network for each task, as well as training all tasks together. This is true even though pairs of tasks tend to cooperate better in this setting. In fact, the performance of our groupings is superior by an even wider margin here. It should be noted that although ESA outperforms the baselines, there is a significant gap between ESA and the optimal solution. Perhaps stopping after training on more of the data would improve ESA's results.

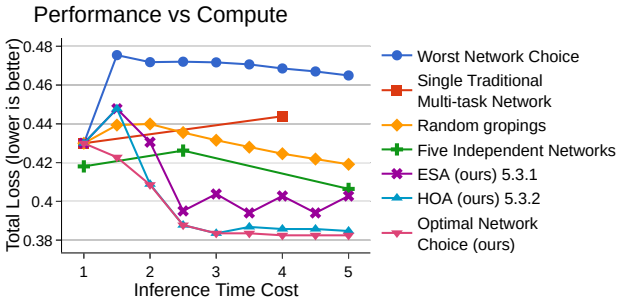

Figure 5. **Performance/inference time trade-off in Setting 3.**

Setting 3: We can see in Figure 5 that our selection framework on the networks trained with only 200k examples is again superior to the baselines. For ESA in this setting, we ran through the entire 200k examples four times, rather than through only 20% once. This represents the same amount of training as in the other settings.

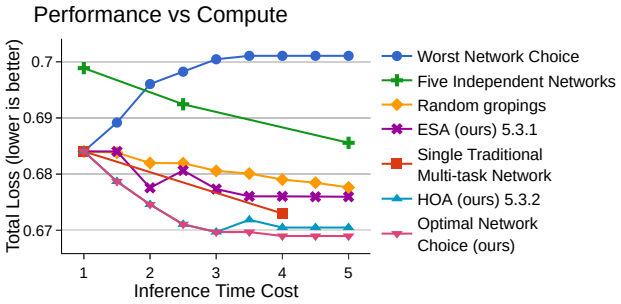

Figure 6. **Performance/inference time trade-off in Setting 4.**

Setting 4: In this setting, the performances of the various networks we trained were quite similar. This shows up in Figure 6 as a very small difference between the 1-SNT all-in-one, and the 5-SNT optimal solution. Since the tasks in this setting tend to cooperate well, it's not surprising that the independent training baseline is not very competitive. In fact, we see that the all-in-one network trained with 4 SNT actually outperforms the Early Stopping Approximation. Yet even in this highly cooperative scenario, our optimal solution outperforms every baseline, as does the HOA.

Qualitative Results: Figure 7 allows qualitative comparison between our methods and the baselines. We can see clear visual issues with each of the baselines that are not

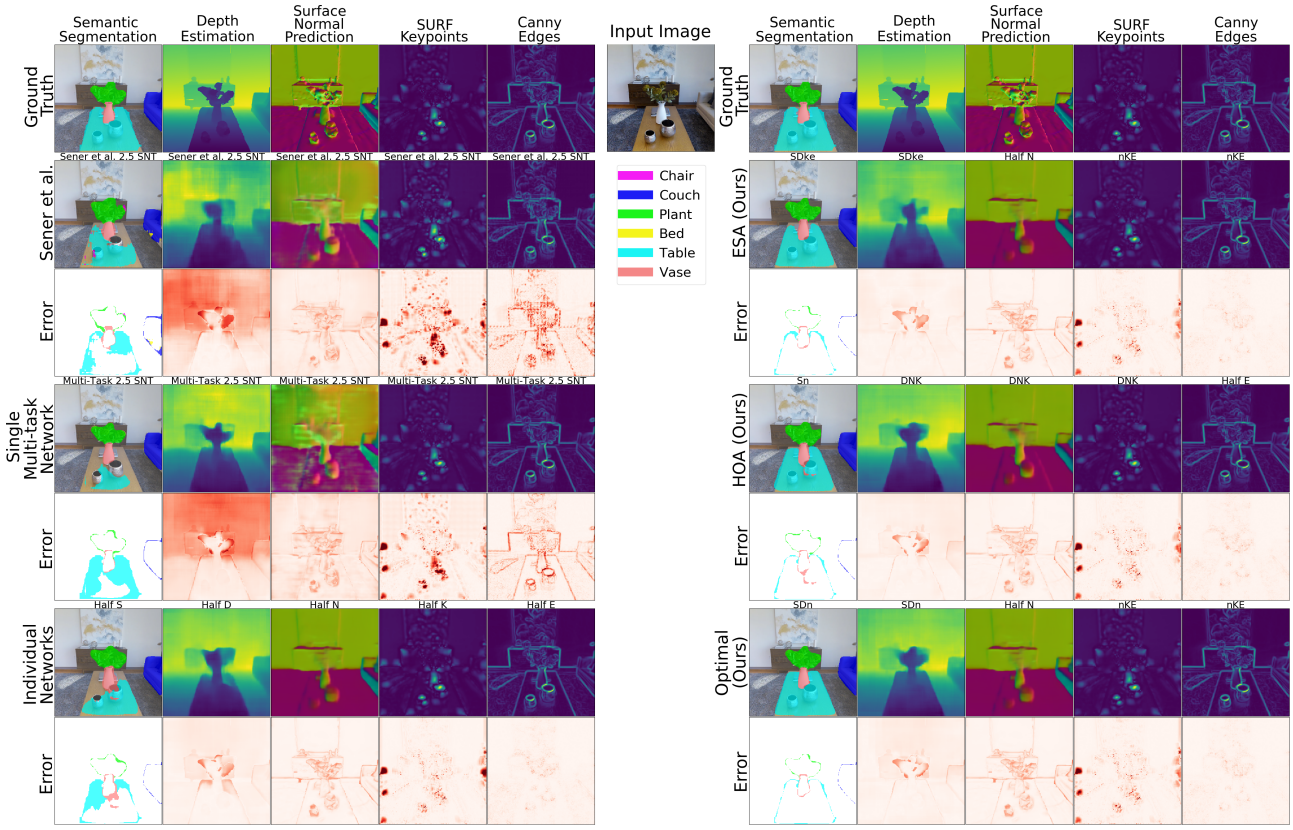

Figure 7. **Setting 1:** Qualitative results for our baselines (left) and our techniques (right). All solutions are allowed 2.5 SNT.

present in our methods. Both of our approximate methods produce predictions similar to the optimal task grouping.

Discussion: In all four settings, our optimal grouping outperformed every baseline. Furthermore, the Higher Order Approximation always performed very similarly to the optimal solution, and often equally well. However, because we had five tasks, more than half of the candidate set was fully trained for this approximation. The Early Stopping Approximation was usually competitive, but the solutions tended to be worse than optimal. In principle, one could stop after training on more of the data, likely resulting in better results. We could not test ESA with other data amounts because our adaptive learning rates resulted in networks that were no longer comparable after being trained on more data.

7. Conclusion

We describe the problem of task compatibility as it pertains to multi-task learning. We provide a computational framework for determining which tasks should be trained jointly and which tasks should be trained separately in a given setting. Our solution can take advantage of situations in which joint training is beneficial to some tasks but not others in the same group. For many use cases, this framework is sufficient, but it can be costly at training time. Hence, we offer two strategies for coping with this issue and evaluate their performance. Our methods outperform single-task networks, a multi-task network with all tasks trained jointly, as well as other baselines. We also use this opportunity to analyze how particular tasks interact in a multi-task setting and compare that with previous results on transfer learning task interactions. We find that unlike transfer task affinities, multi-task affinities depend highly on a number of factors such as dataset size and network capacity. This is another reason why a computational framework like ours is necessary for discovering which tasks should be learned together in multi-task learning.

Acknowledgements We gratefully acknowledge the support of TRI (S-2018-29), NSF grant DMS-1546206, a Vannevar Bush Faculty Fellowship, and ONR MURI (W911NF-150100479). Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

Achille, A., Lam, M., Tewari, R., Ravichandran, A., Maji, S., Fowlkes, C., Soatto, S., and Perona, P. Task2Vec: Task Embedding for Meta-Learning. *arXiv e-prints*, art.

arXiv:1902.03545, February 2019.

Baker, B., Gupta, O., Naik, N., and Raskar, R. Designing neural network architectures using reinforcement learning. *International Conference on Learning Representations*, 2017.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Bilen, H. and Vedaldi, A. Integrated perception with recurrent multi-task neural networks. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 235–243. Curran Associates, Inc., 2016.

Bingel, J. and Søgaard, A. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 164–169, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/E17-2026.

Caruana, R. Multitask learning. *Machine Learning*, 28 (1):41–75, Jul 1997. ISSN 1573-0565. doi: 10.1023/A:1007379606734. URL https://doi.org/10.1023/A:1007379606734.

Chen, L., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, pp. 833–851, 2018a. doi: 10.1007/978-3-030-01234-2_49. URL https://doi.org/10.1007/978-3-030-01234-2_49.

Chen, Z., Badrinarayanan, V., Lee, C., and Rabinovich, A. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pp. 793–802, 2018b. URL http://proceedings.mlr.press/v80/chen18a.html.

Chennupati, S., Sistu, G., Yogamani, S., and Rawashdeh, S. Multinet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

Chollet, F. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

d. Miranda, P. B. C., Prudêncio, R. B. C., d. Carvalho, A. C. P. L. F., and Soares, C. Combining a multi-objective optimization approach with meta-learning for svm parameter selection. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2909–2914, Oct 2012. doi: 10.1109/ICSMC.2012.6378235.

Dai, J., He, K., and Sun, J. Instance-aware semantic segmentation via multi-task network cascades. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3150–3158, 2016.

Doersch, C. and Zisserman, A. Multi-task self-supervised visual learning. In *International Conference on Computer Vision*, 2017.

Domhan, T., Springenberg, J. T., and Hutter, F. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pp. 3460–3468. AAAI Press, 2015. ISBN 9781577357384.

Duong, L., Cohn, T., Bird, S., and Cook, P. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pp. 845–850, July 2015.

Dvornik, N., Shmelkov, K., Mairal, J., and Schmid, C. BlitzNet: A real-time deep network for scene understanding. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.

Dwivedi, K. and Roig., G. Representation similarity analysis for efficient task taxonomy and transfer learning. In *CVPR*. IEEE Computer Society, 2019.

Elsken, T., Metzen, J. H., and Hutter, F. Efficient multi-objective neural architecture search via lamarckian evolution. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=ByME42AqK7.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. Pathnet: Evolution channels gradient descent in super neural networks. *CoRR*, abs/1701.08734, 2017. URL http://arxiv.org/abs/1701.08734.

Finn, C., Tan, X. Y., Duan, Y., Darrell, T., Levine, S., and Abbeel, P. Deep spatial autoencoders for visuomotor learning. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 512–519. IEEE, 2016.

Gong, T., Lee, T., Stephenson, C., Renduchintala, V., Padhy, S., Ndirango, A., Keskin, G., and Elibol, O. H. A comparison of loss weighting strategies for multi task learning in deep neural networks. *IEEE Access*, 7:141627–141632, 2019. ISSN 2169-3536. doi: 10.1109/ACCESS.2019.2943604.

Helleputte, T. and Dupont, P. Feature selection by transfer learning with linear regularized models. In Buntine, W., Grobelnik, M., Mladenić, D., and Shawe-Taylor, J. (eds.), *Machine Learning and Knowledge Discovery in Databases*, pp. 533–547, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. ISBN 978-3-642-04180-8.

Kendall, A., Gal, Y., and Cipolla, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Kokkinos, I. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 5454–5463, 2017. doi: 10.1109/CVPR.2017.579. URL https://doi.org/10.1109/CVPR.2017.579.

Leang, I., Sistu, G., Burger, F., Bursuc, A., and Yogamani, S. Dynamic task weighting methods for multi-task networks in autonomous driving systems. *arXiv preprint arXiv:2001.02223*, 2020.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., and Talwalkar, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *J. Mach. Learn. Res.*, 18 (1):6765–6816, January 2017. ISSN 1532-4435.

Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 19–34, 2018.

Liu, S., Johns, E., and Davison, A. J. End-to-end multi-task learning with attention. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1871–1880. IEEE, 2019.

Long, M., CAO, Z., Wang, J., and Yu, P. S. Learning multiple tasks with multilinear relationship networks. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 1594–1603. Curran Associates, Inc., 2017.

Lu, Y., Kumar, A., Zhai, S., Cheng, Y., Javidi, T., and Feris, R. S. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *CVPR*, pp. 1131–1140. IEEE Computer Society, 2017.

Luo, Z., Zou, Y., Hoffman, J., and Fei-Fei, L. F. Label efficient learning of transferable representations acrosss domains and tasks. In *Advances in Neural Information Processing Systems*, pp. 164–176, 2017.

Mallya, A. and Lazebnik, S. Piggyback: Adding multiple tasks to a single, fixed network by learning to mask. *CoRR*, abs/1801.06519, 2018. URL http://arxiv.org/abs/1801.06519.

Maninis, K., Radosavovic, I., and Kokkinos, I. Attentive single-tasking of multiple tasks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pp. 1851–1860. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00195. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Maninis_Attentive_Single-Tasking_of_Multiple_Tasks_CVPR_2019_paper.html.

Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

Mihalkova, L., Huynh, T., and Mooney, R. J. Mapping and revising markov logic networks for transfer learning. In *AAAI*, volume 7, pp. 608–614, 2007.

Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. Cross-stitch networks for multi-task learning. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3994–4003, 2016.

Nekrasov, V., Dharmasiri, T., Spek, A., Drummond, T., Shen, C., and Reid, I. D. Real-time joint semantic segmentation and depth estimation using asymmetric annotations. *CoRR*, abs/1809.04766, 2018. URL http://arxiv.org/abs/1809.04766.

Niculescu-Mizil, A. and Caruana, R. Inductive transfer for bayesian network structure learning. In *Artificial Intelligence and Statistics*, pp. 339–346, 2007.

Noh, H., Hong, S., and Han, B. Learning deconvolution network for semantic segmentation. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1520–1528, Dec 2015. doi: 10.1109/ICCV.2015.178.

Pal, A. and Balasubramanian, V. N. Zero-shot task transfer, 2019.

Pan, S. J. and Yang, Q. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, Oct 2010. ISSN 1041-4347. doi: 10.1109/TKDE.2009.191.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Pentina, A. and Lampert, C. H. Multi-task learning with labeled and unlabeled tasks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 2807–2816, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/pentina17a.html.

Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. Efficient neural architecture search via parameters sharing. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 4095–4104, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL http://proceedings.mlr.press/v80/pham18a.html.

Pratt, L. Y. Discriminability-based transfer between neural networks. In Hanson, S. J., Cowan, J. D., and Giles, C. L. (eds.), *Advances in Neural Information Processing Systems 5*, pp. 204–211. Morgan-Kaufmann, 1993.

Razavian, A. S., Azizpour, H., Sullivan, J., and Carlsson, S. Cnn features off-the-shelf: An astounding baseline for recognition. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, CVPRW '14, pp. 512–519, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-4308-1. doi: 10.1109/CVPRW.2014.131. URL http://dx.doi.org.stanford.idm.oclc.org/10.1109/CVPRW.2014.131.

Real, E., Aggarwal, A., Huang, Y., and Le, Q. V. Regularized evolution for image classifier architecture search. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pp. 4780–4789. AAAI Press, 2019. doi: 10.1609/aaai.v33i01.33014780. URL https://doi.org/10.1609/aaai.v33i01.33014780.

Rudd, E. M., Günther, M., and Boult, T. E. MOON: A mixed objective optimization network for the recognition of facial attributes. In *ECCV (5)*, volume 9909 of *Lecture Notes in Computer Science*, pp. 19–35. Springer, 2016.

Ruder, S. An overview of multi-task learning in deep neural networks. *CoRR*, abs/1706.05098, 2017. URL http://arxiv.org/abs/1706.05098.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. Progressive neural networks. *CoRR*, abs/1606.04671, 2016. URL http://arxiv.org/abs/1606.04671.

Sener, O. and Koltun, V. Multi-task learning as multi-objective optimization. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 525–536. Curran Associates, Inc., 2018.

Silver, D. L. and Bennett, K. P. Guest editor's introduction: special issue on inductive transfer learning. *Machine Learning*, 73(3):215–220, 2008.

Sun, X., Panda, R., and Feris, R. Adashare: Learning what to share for efficient deep multi-task learning. *arXiv preprint arXiv:1911.12423*, 2019.

Suteu, M. and Guo, Y. Regularizing deep multi-task networks using orthogonal gradients, 2019.

Tessler, C., Givony, S., Zahavy, T., Mankowitz, D. J., and Mannor, S. A deep hierarchical approach to lifelong learning in minecraft. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pp. 1553–1561. AAAI Press, 2017. URL http://dl.acm.org.stanford.idm.oclc.org/citation.cfm?id=3298239.3298465.

Thrun, S. Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, pp. 640–646. The MIT Press, 1996.

Wang, A. Y., Wehbe, L., and Tarr, M. J. Neural taskonomy: Inferring the similarity of task-derived representations from brain activity. *BioRxiv*, pp. 708016, 2019.

Xie, S., Zheng, H., Liu, C., and Lin, L. SNAS: stochastic neural architecture search. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rylqooRqK7.

Yang, Y. and Hospedales, T. Trace norm regularised deep multi-task learning. In *5th International Conference on Learning Representations Workshop*, 2017.

Zamir, A., Sax, A., Yeo, T., Kar, O., Cheerla, N., Suri, R., Cao, Z., Malik, J., and Guibas, L. Robust learning through cross-task consistency. *arXiv*, 2020.

Zamir, A. R., Wekel, T., Agrawal, P., Wei, C., Malik, J., and Savarese, S. Generic 3d representation via pose estimation and matching. In Leibe, B., Matas, J., Sebe, N., and Welling, M. (eds.), *Computer Vision – ECCV 2016*, pp. 535–553, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46487-9.

Zamir, A. R., Sax, A., Shen, W. B., Guibas, L. J., Malik, J., and Savarese, S. Taskonomy: Disentangling task transfer learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2018.

Zhang, Y. and Yang, Q. A survey on multi-task learning. *CoRR*, abs/1707.08114, 2017. URL http://arxiv.org/abs/1707.08114.

Zhou, D., Wang, J., Jiang, B., Guo, H., and Li, Y. Multi-task multi-view learning based on cooperative multi-objective optimization. *IEEE Access*, 6:19465–19477, 2018. ISSN 2169-3536. doi: 10.1109/ACCESS.2017.2777888.

Zhou, H., Yang, M., Wang, J., and Pan, W. BayesNAS: A Bayesian approach for neural architecture search. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 7603–7613, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/zhou19e.html.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=r1Ue8Hcxg.

Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 8697–8710. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00907. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Zoph_Learning_Transferable_Architectures_CVPR_2018_paper.html.