Attentive Single-Tasking of Multiple Tasks

Kevis-Kokitsi Maninis[*]
Computer Vision Lab, ETH Zürich

Ilija Radosavovic
Facebook AI Research (FAIR)

Iasonas Kokkinos[*]
Ariel AI, UCL

Abstract

In this work we address task interference in universal networks by considering that a network is trained on multiple tasks, but performs one task at a time, an approach we refer to as "single-tasking multiple tasks". The network thus modifies its behaviour through task-dependent feature adaptation, or task attention. This gives the network the ability to accentuate the features that are adapted to a task, while shunning irrelevant ones. We further reduce task interference by forcing the task gradients to be statistically indistinguishable through adversarial training, ensuring that the common backbone architecture serving all tasks is not dominated by any of the task-specific gradients.

Results in three multi-task dense labelling problems consistently show: (i) a large reduction in the number of parameters while preserving, or even improving performance and (ii) a smooth trade-off between computation and multi-task accuracy. We provide our system's code and pre-trained models at `http://vision.ee.ethz.ch/~kmaninis/astmt/`.

1. Introduction

Real-world problems involve a multitude of visual tasks that call for multi-tasking, universal vision systems. For instance autonomous driving requires detecting pedestrians, estimating velocities and reading traffic signs, while identity recognition, pose, face and hand tracking are required for human-computer interaction.

A thread of works have introduced multi-task networks [57, 13, 22, 28] handling an increasingly large number of tasks. Still, it is common practice to train devoted networks for individual tasks when single-task performance is critical. This is supported by negative results from recent works that have aimed at addressing multiple problems with a single network [22, 28] - these have shown that there is a limit on performance imposed by the capacity of the network, manifested as a drop in performance when loading a single network with more tasks. Stronger backbones can uniformly improve multi-task performance, but still the per-task performance can be lower than the single-task performance with the same backbone.

[*]Work done while at Facebook AI Research.

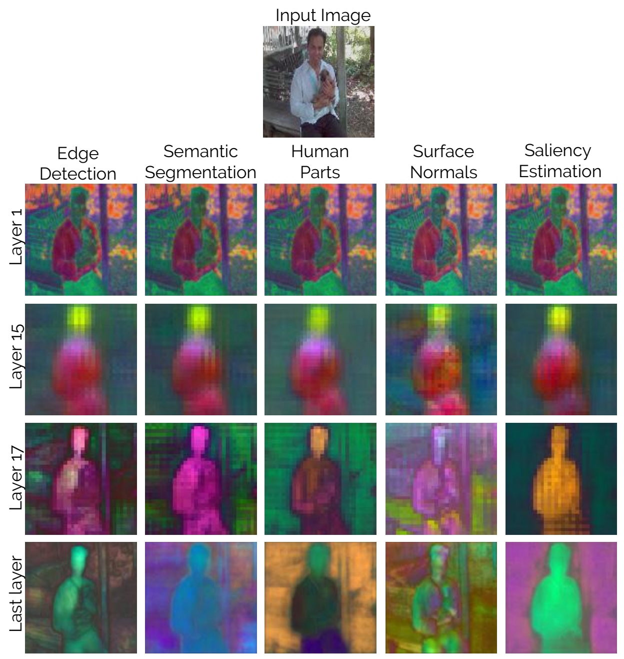


Figure 1. **Learned representations across tasks and layers:** We visualize how features change spatially in different depths of our multi-task network. For each layer (row) we compute a common PCA basis across tasks (column) and show the first three principal components as RGB values at each spatial location. We observe that the features are more similar in early layers and get more adapted to specific tasks as depth increases, leading to disentangled, task-specific representations in the later layers. We see for instance that the normal task features co-vary with surface properties, while the part features remain constant in each human part.

This problem, known as task interference, can be understood as facing a the dilemma of invariance versus sensitivity: the most crucial information for one task can be a nuisance parameter for another, which leads to potentially conflicting objectives when training multi-task networks. An example of such a task pair is pose estimation and object detection: when detecting or counting people the detailed pose information is a nuisance parameter that should be eliminated at some point from the representation of a network aiming at pose invariance [22]. At the same time, when watching a dance performance, one needs the detailed pose of the dancers, while ignoring the large majority of spectators. More generally this is observed when combining a task

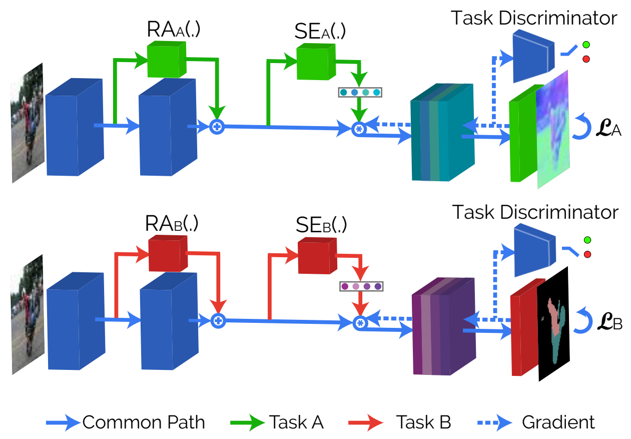


Figure 2. While using a shared backbone network, every task adapts its behavior in a separate, flexible, and lightweight manner, allowing us to customize computation for the task at hand. We refine features with a task-specific residual adapter branch (RA), and attend to particular channels with task-specific Squeeze-and-Excitation (SE) modulation. We also enforce the task gradients (dashed lines) to be statistically indistinguishable through adversarial training, further promoting the separation between task-specific and generic layers.

that is detail-oriented and requires high spatial acuity with a task that requires abstraction from spatial details, e.g. when one wants to jointly do low- and high-level vision. In other words, one task's noise is another one's signal.

We argue that this dilemma can be addressed by single-tasking, namely executing task a time, rather than getting all task responses in a single forward pass through the network. This reflects many practical setups, for instance when one sees the results of a single computational photography task at a time on the screen of a mobile phone, rather than all of them jointly. Operating in this setup allows us to implement an "attention to task" mechanism that changes the network's behaviour in a task-adapted manner, as shown in Fig. 1. We use the exact same network backbone in all cases, but we modify the network's behavior according to the executed task by relying on the most task-appropriate features. For instance when performing a low-level task such as boundary detection or normal estimation, the network can retain and elaborate on fine image structures, while shunning them for a high-level task that requires spatial abstraction.

We explore two different *task attention mechanisms*, as shown in Fig. 2. Firstly, we use data-dependent modulation signals [44] that enhance or suppress neuronal activity in a task-specific manner. Secondly, we use task-specific Residual Adapter [47] blocks that latch on to a larger architecture in order to extract task-specific information which is fused with the representations extracted by a generic backbone. This allows us to learn a shared backbone representation that serves all tasks but collaborates with task-specific processing to build more elaborate task-specific features.

These two extensions can be understood as promoting a disentanglement between the shared representation learned across all tasks and the task-specific parts of the network. Still, if the loss of a single task is substantially larger, its gradients will overwhelm those of others and disrupt the training of the shared representation. In order to make sure that no task abuses the shared resources we impose a *task-adversarial loss* to the network gradients, requiring that these are statistically indistinguishable across tasks. This loss is minimized during training through double back-propagation [10], and leads to an automatic balancing of loss terms, while promoting compartmentalization between task-specific and shared blocks.

2. Related Work

Our work draws ideas from several research threads. **Multiple Task Learning (MTL):** Several works have shown that jointly learning pairs of tasks yields fruitful results in computer vision. Successful pairs include detection and classification [19, 48], detection and segmentation [22, 12], or monocular depth and segmentation [13, 64]. Joint learning is beneficial for unsupervised learning [45], when tasks provide complementary information (eg. depth boundaries and motion boundaries [70]), in cases where task A acts as regularizer for task B due to limited data [32], or in order to learn more generic representations from synthetic data [49]. Xiao et al. [62] unify inhomogeneous datasets in order to train for multiple tasks, while [67] explore relationships among a large amount of tasks for transfer learning, reporting improvements when transferring across particular task pairs.

Despite these positive results, joint learning can be harmful in the absence of a direct relationship between task pairs. This was reported clearly in [28] where the joint learning of low-, mid- and high-level tasks was explored, reporting that the improvement of one task (e.g. normal detection) was to the detriment of another (e.g. object detection). Similarly, when jointly training for human pose estimation on top of detection and segmentation, Mask R-CNN performs worse than its two-task counterpart [22].

This negative result first requires carefully calibrating the relative losses of the different tasks, so that none of them deteriorates excessively. To address this problem, GradNorm [8] provides a method to adapt the weights such that each task contributes in a balanced way to the loss, by normalizing the gradients of their losses; a more recent work [60] extends this approach to homogenize the task gradients based on adversarial training. Following a probabilistic treatment [25] re-weigh the losses according to each task's uncertainty, while Sener and Koltun [56] estimate an adaptive weighting of the different task losses based on a pareto-optimal formulation of MTL. Similarly, [20] provide a MTL framework where tasks are dynamically sorted by difficulty and the hardest are learned first.

A second approach to mitigate task interference consists in avoiding the 'spillover' of gradients from one task's loss to the common features serving all tasks. One way of doing this is explicitly constructing complementary task-specific feature representations [53, 51], but results in an increase of complexity that is linear in the number of tasks. An alternative, adopted in the related problem of lifelong learning consists in removing from the gradient of a task's loss those components that would incur an increase in the loss of previous tasks [26, 33]. For domain adaptation [4] disentangle the representations learned by shared/task-specific parts of networks by enforcing similarity/orthogonality constraints. Adversarial Training has been used in the context of domain adaptation [17, 32] to the feature space in order to fool the discriminator about the source domain of the features.

In our understanding these losses promote a compartmental operation of a network, achieved for instance when a block-structured weight matrix prevents the interference of features for tasks that should not be connected. A deep single-task implementation of this would be the gating mechanism of [1]. For multi-tasking, Cross Stitch Networks [39] automatically learn to split/fuse two independent networks in different depths according to their learned tasks, while [41] estimate a block-structured weight matrix during CNN training.

As we show in the next section, a soft "compartmentalization" effect that does not interfere with the network's weight matrix can be achieved in the case of multi-task learning through a soft, learnable form of task-specific feature masking. This shares several similarities with attention mechanisms, briefly described below.

Attention mechanisms: Attention has often been used in deep learning to visualize and interpret the inner workings of CNNs [58, 68, 55], but has also been employed to improve the learned representations of convolutional networks for scale-aware semantic segmentation [5], fine-grained image recognition [16] or caption generation [65, 34, 2]. Squeeze and Excitation Networks [24] and their variants [61, 23] modulate the information of intermediate spatial features according to a global representation and be understood as implementing attention to different channels. Deep Residual Adapters [46, 47] modulate learned representations depending on their source domain. Several works study modulation for image retrieval [69] or classification tasks [44, 40], and embeddings for different artistic styles [11]. [66] learns object-specific modulation signals for video object segmentation, and [50] modulates features according to given priors for detection and segmentation. In our case we learn task-specific modulation functions that allow us to drastically change the network's behaviour while using identical backbone weights.

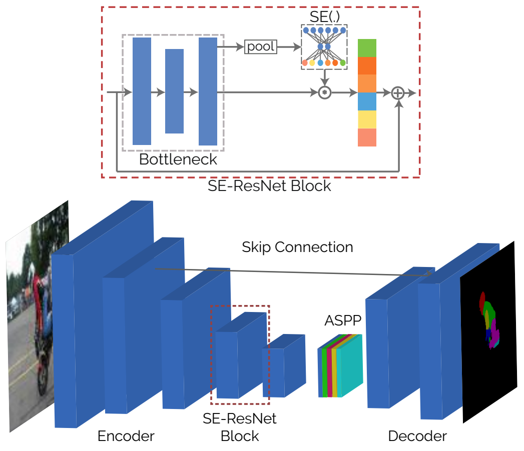


Figure 3. **Single-task network architecture:** We use Deeplab-v3+ with a Squeeze-and-Excitation (SE)-ResNet backbone. SE modules are present in all bottleneck blocks of the encoder and the decoder. Attentive multi-tasking uses different SE layers per task to modulate the network features in a task-specific manner.

3. Attentive Single-Tasking Mechanisms

Having a shared representation for multiple tasks can be efficient from the standpoint of memory- and sample-complexity, but can result in worse performance if the same resources are serving tasks with unrelated, or even conflicting objectives, as described above. Our proposed remedy to this problem consists in learning a shared representation for all tasks, while allowing each task to use this shared representation differently for the construction of its own features.

3.1. Task-specific feature modulation

In order to justify our approach we start with a minimal example. We consider that we have two tasks A and B that share a common feature tensor $\mathrm{F}(x, y, c)$ at a given network layer, where x, y are spatial coordinates and $c = 1, \ldots, C$ are the tensor channels. We further assume that a subset $\mathcal{S}_A$ of the channels is better suited for task A, while $\mathcal{S}_B$ is better for B. For instance if A is invariant to deformations (detection) while B is sensitive (pose estimation), $\mathcal{S}_A$ could be features obtained by taking more context into account, while $\mathcal{S}_B$ would be more localized.

One simple way of ensuring that tasks A and B do not interfere while using a shared feature tensor is to hide the features of task B when training for task A:

$$\mathrm{F}_A(x, y, c) = m_A[c] \cdot \mathrm{F}(x, y, c) \tag{1}$$

where $m_A[c] \in \{0, 1\}$ is the indicator function of set $\mathcal{S}_A$. If $c \notin \mathcal{S}_A$ then $\mathrm{F}_A(x, y, c) = 0$, which means that the gradient $\frac{\partial \mathcal{L}_A}{\partial \mathrm{F}(x,y,c)}$ sent by the loss $\mathcal{L}_A$ of task A to $c \in \mathcal{S}_A$ will be zero. We thereby avoid task interference since Task A does not influence nor use features that it does not need.

Instead of this hard choice of features per task we opt for a soft, differentiable membership function that is learned in tandem with the network and allows the different tasks to discover during training which features to use. Instead of a constant membership function per channel we opt for an image-adaptive term that allows one to exploit the power of the squeeze-and-excitation block [24].

In particular we adopt the squeeze-and-excitation (SE) block (also shown in Fig. 2), combining a global average pooling operation of the previous layer with a fully-connected layer that feeds into a sigmoid function, yielding a differentiable, image-dependent channel gating function. We set the parameters of this layer to be task-dependent, allowing every task to modulate the available channels differently. As shown in Section 5, this can result in substantial improvements when compared to a baseline that uses the same SE block for all tasks.

3.2. Residual Adapters

The feature modulation described above can be understood as shunning those features that do not contribute to the task while focusing on the more relevant ones. Intuitively, this does not add capacity to the network but rather cleans the signal that flows through it from information that the task should be invariant to. We propose to complement this by appending task-specific sub-networks that adapt and refine the shared features in terms of residual operations of the following form:

$$\mathrm{L}_A(x) = x + \mathrm{L}(x) + \mathrm{RA}_A(x), \tag{2}$$

where $\mathrm{L}(x)$ denotes the default behaviour of a residual layer, RA_A is the task-specific residual adapter of task A, and $\mathrm{L}_A(x)$ is the modified layer. We note that if $\mathrm{L}(x)$ and $\mathrm{RA}_A(x)$ were linear layers this would amount to the classical regularized multi-task learning of [15].

These adapters introduce a task-specific parameter and computation budget that is used in tandem with that of the shared backbone network. We show in Section 5 that this is typically a small fraction of the budget used for the shared network, but improves accuracy substantially.

When employing disentangled computation graphs with feature modulation through SE modules and/or residual adapters, we also use task-specific batch-normalization layers, that come with a trivial increase in parameters (while the computational cost remains the same).

4. Adversarial Task Disentanglement

The idea behind the task-specific adaptation mechanisms described above is that even though a shared representation has better memory/computation complexity, every task can profit by having its own 'space', i.e. separate modelling capacity to make the best use of the representation - by modulating the features or adapting them with residual blocks.

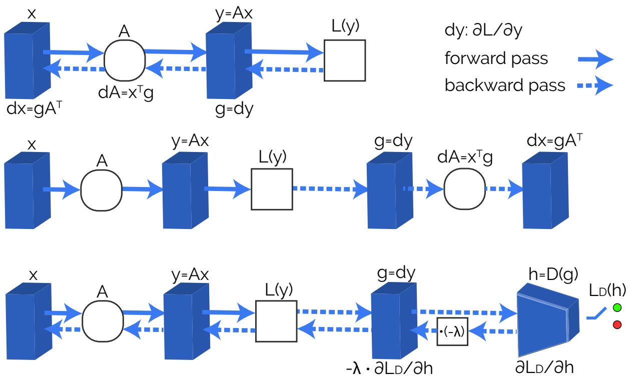


Figure 4. Double backprop [10] exposes the gradients computed during backprop (row 1) by unfolding the computation graph of gradient computation (row 2). Exposing the gradients allows us to train them in an adversarial setting by using a discriminator, forcing them to be statistically indistinguishable across tasks (row 3). The shared network features x then receive gradients that have the same distribution irrespective of the task, ensuring that no task abuses the shared network, e.g. due to higher loss magnitude. The gradient of the discriminator is reversed (negated) during adversarial training, and the parameter $\lambda \in [0, 1]$ controls the amount of negative gradient that flows back to the network [17].

Database	Type	# Train Im.	# Test Im.	Edge	S.Seg	H. Parts	Normals	Saliency	Depth	Albedo
PASCAL	Real	4,998	5,105	✓	✓	✓	✓*	✓*		
NYUD	Real	795	654	✓	✓		✓		✓	
FSV	Synthetic	223,197	50,080		✓				✓	✓

Table 1. **Multi-task benchmark statistics**: We conduct the main volume of experiments on PASCAL for 5 tasks (* labels obtained via distillation). We also use the fully labelled subsets of NYUD, and the synthetic FSV dataset.

Pushing this idea further we enforce a strict separation of the shared and task-specific processing, by requiring that the gradients used to train the shared parameters are statistically indistinguishable across tasks. This ensures that the shared backbone serves all tasks equally well, and is not disrupted e.g. by one task that has larger gradients.

We enforce this constraint through adversarial learning. Several methods, starting from Adversarial Domain Adaptation [18], use adversarial learning to remove any trace of a given domain from the learned mid-level features in a network; a technique called Adversarial multi-task training [32] falls in the same category.

Instead of removing domain-specific information from the features of a network (which serves domain adaptation), we remove any task-specific trace from the gradients sent from different tasks to the shared backbone (which serves a division between shared and task-specific processing). A concurrent work [60] has independently proposed this idea.

As shown in Fig. 4 we use double back-propagation [10] to 'expose' the gradient sent from a task t to a shared layer l, say $g_t(l)$. By exposing the variable we mean that we unfold its computation graph, which in turn allows us to back-propagate through its computation. By back-propagating on

Task	Dataset	Metric	R-101	strong baseline
Edge	BSDS500	odsF ↑	82.5	81.3 [27]
S.Seg	VOC	mIoU ↑	78.9	79.4 [6]
H. Parts	P. Context	mIoU ↑	64.3	64.9* [5]
Normals	NYUD	mErr ↓	20.1	19.0 [3]
Saliency	PASCAL-S	maxF ↑	84.0	83.5 [28]
Depth	NYUD	RMSE ↓	0.56	0.58 [64]

Table 2. **Architecture capacity**: We report upper-bounds of performance that can be reached on various competitive (but inhomogeneous) datasets by our architecture, and compare to strong task-specific baselines. All experiments are initialized from ImageNet pre-trained weights (* means that COCO pre-training is included). The arrow indicates better performance for each metric.

the gradients we can force them to be statistically indistinguishable across tasks through adversarial training.

In particular we train a task classifier on top of the gradients lying at the interface of the task-specific and shared networks and use sign negation to make the task classifier fail [17]. This amounts to solving the following optimization problem in terms of the discriminator weights, w_D and the network weights, w_N:

$$\min_{w_D} \max_{w_N} L(D(g_t(w_N), w_D), t), \tag{3}$$

where $g_t(w_N)$ is the gradient of task t computed with w_N, $D(\cdot, w_D)$ is the discriminator's output for input $\cdot$, and $L(\cdot, t)$ is the cross-entropy loss for label t that indicates the source task of the gradient.

Intuitively this forces every task to do its own processing within its own blocks, so that it does not need from the shared network anything different from the other tasks. This results in a separation of the network into disentangled task-specific and shared compartments.

5. Experimental Evaluation

Datasets We validate our approach on different datasets and tasks. We focus on dense prediction tasks that can be approached with fully convolutional architectures. Most of the experiments are carried out on the PASCAL [14] benchmark, which is popular for dense prediction tasks. We also conduct experiments on the smaller NYUD [42] dataset of indoor scenes, and the recent, large scale FSV [29] dataset of synthetic images. Statistics, as well as the different tasks used for each dataset are presented in Table 1.

Base architecture: We use our re-implementation of Deeplab-v3+ [6] as the base architecture of our method, due to its success on dense semantic tasks. Its architecture is based on a strong ResNet encoder, with a-trous convolutions to preserve reasonable spatial dimensions for dense prediction. We use the latest version that is enhanced with a parallel a-trous pyramid classifier (ASPP) and a powerful decoder. We refer the reader to [6] for more details. The ResNet-101 backbone used in the original work is replaced with its Squeeze-and-Excitation counterpart (Fig. 3),

SE-bb	#T	Edge ↑	Seg ↑	Parts ↑	Norm ↓	Sal ↑
	1	70.3	63.98	55.85	15.11	63.92
✓	1	**71.3**	**64.93**	**57.12**	**14.90**	**64.17**
	5	68.0	58.59	53.80	**16.68**	60.71
✓	5	**69.2**	**60.20**	**54.10**	17.04	**62.10**

(a) **Baselines.** Using SE blocks in ResNet backbones (SE-bb) improves results. In all our experiments we use SE-bb baselines for fair comparison.

SE	RA	#T	Edge ↑	Seg ↑	Parts ↑	Norm ↓	Sal ↑	$\Delta_m\%$ ↓
		1	71.3	64.93	57.12	14.90	64.17	
		5	69.2	60.20	54.10	17.04	62.10	6.62
	✓	5	70.5	62.80	56.41	15.27	64.84	**1.42**
✓		5	71.1	64.00	56.84	15.05	64.35	**0.59**

(b) **Modulation.** Both SE and RA are effective modulation methods.

enc	dec	#T	Edge ↑	Seg ↑	Parts ↑	Norm ↓	Sal ↑	$\Delta_m\%$ ↓
		1	71.3	64.93	57.12	14.90	64.17	
		5	69.2	60.20	54.10	17.04	62.1	6.62
	✓	5	70.6	63.33	56.73	15.14	63.23	**1.44**
✓	✓	5	71.1	64.00	56.84	15.05	64.35	**0.59**

(c) **SE modulation.** Modulating varying portions of the network (e.g. encoder or decoder) allows trading off performance and computation.

mod	A	#T	Edge ↑	Seg ↑	Parts ↑	Norm ↓	Sal ↑	$\Delta_m\%$ ↓
		1	71.3	64.93	57.12	14.90	64.17	
		5	69.2	60.20	54.10	17.04	62.10	6.62
	✓	5	69.7	62.20	55.04	16.17	62.19	**4.34**
✓		5	71.1	64.00	56.84	15.05	64.35	**0.59**
✓	✓	5	71.0	64.61	57.25	15.00	64.70	**0.11**

(d) **Adversarial training** is beneficial both w/ and w/o SE modulation.

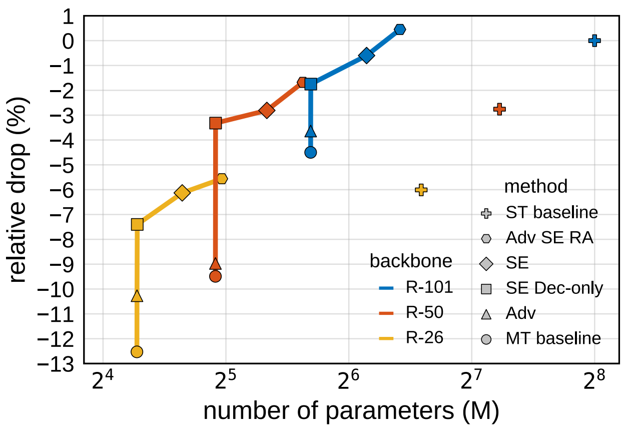
backbone	SEA	#T	Edge ↑	Seg ↑	Parts ↑	Norm ↓	Sal ↑	$\Delta_m\%$ ↓
R-26		1	71.3	64.93	57.12	14.90	64.17	
R-26		5	69.2	60.20	54.10	17.04	62.10	6.62
R-26	✓	5	71.0	64.61	57.25	15.00	64.70	**0.11**
R-50		1	72.7	68.30	60.70	14.61	65.40	
R-50		5	69.2	63.20	55.10	16.04	63.60	6.81
R-50	✓	5	72.4	68.00	61.12	14.68	65.71	**0.04**
R-101		1	73.5	69.76	63.48	14.15	67.41	
R-101		5	70.5	66.45	61.54	15.44	66.39	4.50
R-101	✓	5	73.5	68.51	63.41	14.37	67.72	**0.60**

(e) **Backbones.** Improvements from SE modulation with adversarial training (SEA) are observed regardless of the capacity/depth of the backbones.

Table 3. **Ablations on PASCAL**. We report average relative performance drop ($\Delta_m\%$) with respect to single task baselines. Backbone is R-26 unless otherwise noted.

pre-trained on ImageNet [52]. The pre-trained SE modules serve as an initialization point for the task-specific modulators for multi-tasking experiments.

The architecture is tested for a single task in various competitive benchmarks for dense prediction: edge detection, semantic segmentation, human part segmentation, surface normal estimation, saliency, and monocular depth estimation. We compare the results obtained with various competitive architectures. For edge detection we use the BSDS500 [38] benchmark and its optimal dataset F-measure (`odsF`) [37]. For semantic segmentation we train



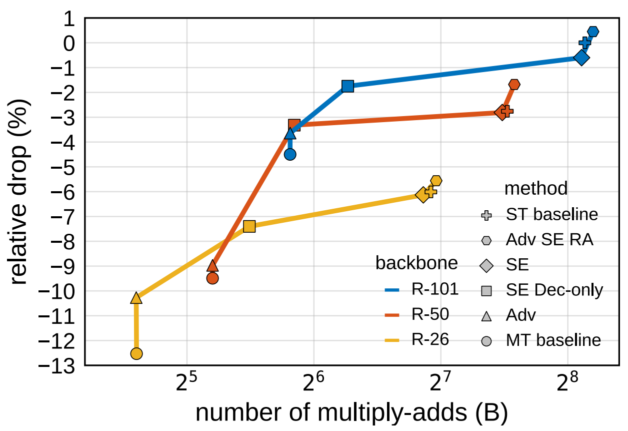


Figure 5. **Performance vs. Resources:** Average relative drop ($\Delta_m\%$) as a function of the number of parameters (left), and multiply-adds (right), for various points of operation of our method. We compare 3 different backbone architectures, indicated with different colors. We compare against single-tasking baseline (ST baseline), and multi-tasking baseline (MT baseline). Performance is measured relative to the best single-tasking model (R-101 backbone). An increase in performance comes for free with adversarial training (Adv). Modulation per task (SE) results in large improvements in performance, thanks to the disentangled graph representations, albeit with an increase in computational cost if used throughout the network, instead of only on the decoder (SE Dec-only vs. SE). We observe a drastic drop in number of parameters needed for our model in order to reach the performance of the baseline (SE, Adv). By using both modulation and adversarial training (Adv SE RA), we are able to reach single-task performance, with far fewer parameters.

SEA	#T	Edge ↑	Seg ↑	Norm ↓	Depth ↓	$\Delta_m\%$ ↓
	1	74.4	32.82	23.30	0.61	
	4	73.2	30.95	23.34	0.70	5.44
✓	4	74.5	32.16	23.18	0.57	**-1.22**

(a) Results on **NYUD-v2**.

SEA	#T	Seg ↑	Albedo ↓	Disp ↓	$\Delta_m\%$ ↓
	1	71.2	0.086	0.063	
	3	66.9	0.093	0.078	7.04
✓	3	70.7	0.085	0.063	**-0.02**

(b) Results on **FSV**.

Table 4. Improvements from SE with modulation (SEA) transfer to **NYUD-v2 and FSV** datasets. We report average performance drop with respect to single task baselines. We use R-50 backbone.

on PASCAL VOC `trainaug` [14, 21] (10582 images), and evaluate on the validation set of PASCAL using mean intersection over union (`mIoU`). For human part segmentation we use PASCAL-Context [7] and `mIoU`. For surface normals we train on the raw data of NYUD [42] and evaluate on the test set using mean error (`mErr`) in the predicted angles as the evaluation metric. For saliency we follow [28] by training on MSRA-10K [9], testing on PASCAL-S [31] and using the maximal F-measure (`maxF`) metric. Finally, for depth estimation we train and test on the fully annotated training set of NYUD using root mean squared error (`RMSE`) as the evaluation metric. For implementation details, and hyper-parameters, please refer to the Appendix.

Table 2 benchmarks our architecture against popular state-of-the-art methods. We obtain competitive results, for all tasks. We emphasize that these benchmarks are inhomogeneous, i.e. their images are not annotated with all tasks, while including domain shifts when training for multi-tasking (eg. NYUD contains only indoor images). In order to isolate performance gains/drops as a result of multi-task learning (and not domain adaptation, or catastrophic forgetting), in the experiments that follow, we use homogeneous datasets.

Multi-task learning setup: We proceed to multi-tasking experiments on PASCAL. We keep the splits of PASCAL-Context, which provides labels for edge detection, semantic segmentation, and human part segmentation. In order to keep the dataset homogeneous and the architecture identical for all tasks, we did not use instance level tasks (detection, pose estimation) that are provided with the dataset. To increase the number of tasks we automatically obtained ground-truth for surface normals and saliency through label-distillation using pre-trained state-of-the-art models ([3] and [6], respectively), since PASCAL is not annotated with those tasks. For surface normals, we masked out predictions from unknown and/or invalid classes (eg. sky) during both training and testing. In short, our benchmark consists of 5 diverse tasks, ranging from low-level (edge detection, surface normals), to mid-level (saliency) and high-level (semantic segmentation, human part segmentation) tasks.

Evaluation metric: We compute multi-tasking performance of method m as the average per-task drop with respect to the single-tasking baseline b (i.e different networks trained for a single task each):

$$\Delta_m = \frac{1}{T}\sum_{i=1}^{T}(-1)^{l_i}\left(M_{m,i} - M_{b,i}\right)/M_{b,i} \quad (4)$$

where $l_i = 1$ if a lower value means better for measure M_i

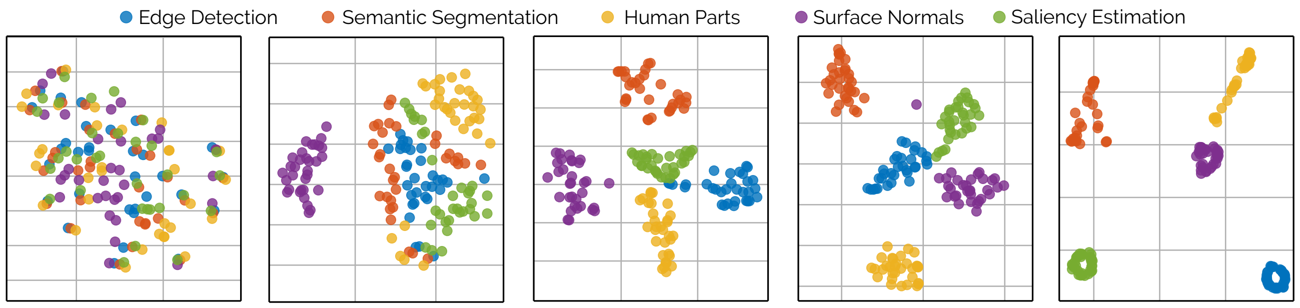


Figure 6. **t-SNE visualization of task-dependent feature activations of a single image** at increasing depths of the network (from left to right). Features in early layers are more similar across tasks and progressively get more adapted to specific tasks in later layers.

of task i, and 0 otherwise. Average relative drop is computed against the baseline that uses *the same backbone*.

To better understand the effect of different aspects of our method, we conduct a number of ablation studies and present the results in Table 3.

We construct a second baseline, which tries to learn all tasks simultaneously with a single network, by connecting T task-specific convolutional classifiers (1×1 conv layers) at the end of the network. As also reported by [28], a non-negligible average performance drop can be observed (-6.6% per task for R-26 with SE). We argue that this drop is mainly triggered by conflicting gradients during learning.

Effects of modulation and adversarial training: Next, we introduce the modulation layers described in Section 3. We compare parallel residual adapters to SE (Table 3b) when used for task modulation. Performance per task recovers immediately by separating the computation used by each task during learning (-1.4 and -0.6 vs. -6.6 for adapters and SE, respectively). SE modulation results in better performance, while using slightly fewer parameters per task. We train a second variant where we keep the computation graph identical for all tasks in the encoder, while using SE modulation only in the decoder (Table 3c). Interestingly, this variant reaches the performance of residual adapters (-1.4), while being much more efficient in terms of number of parameters and computation, as only one forward pass of the encoder is necessary for all tasks.

In a separate experiment, we study the effects of adversarial training described in Section 4. We use a simple, fully convolutional discriminator to classify the source of the gradients. Results in Table 3d show that adversarial training is beneficial for multi-tasking, increasing performance compared to standard multi-tasking (-4.4 vs -6.6). Even though the improvements are less significant compared to modulation, they come without extra parameters or computational cost, since the discriminator is used only during training.

The combination of SE modulation with adversarial training (Table 3d) leads to additional improvements (-0.1% worse than the single-task baseline), while further adding residual adapters surpasses single-tasking (+0.45%), at the cost of 12.3% more parameters per task (Fig. 5).

Deeper Architectures: Table 3e shows how modulation and adversarial training perform when coupled with deeper architectures (R-50 and R-101). The results show that our method is invariant to the depth of the backbone, consistently improving the standard multi-tasking results.

Resource Analysis: Figure 5 illustrates the performance of each variant as a function of the number of parameters, as well as the FLOPS (multiply-adds) used during inference. We plot the relative average per-task performance compared to the single-tasking R-101 variant (blue cross), for the 5 tasks of PASCAL. Different colors indicate different backbone architectures. We see a clear drop in performance by standard multi-tasking (crosses vs. circles), but with fewer parameters and FLOPS. Improvements due to adversarial training come free of cost (triangles) with only a small overhead for the discriminator during training.

Including modulation comes with significant improvements, but also with a very slight increase of parameters and a slight increase of computational cost when including the modules on the decoder (rectangles). The increase becomes more apparent when including those modules in the encoder as well (diamonds). Our most accurate variant using all of the above (hexagons) outperforms the single-tasking baselines by using only a fraction of their parameters.

We note that the memory and computational complexities of the SE blocks and the adapters are negligible, but since it affects the outputs of the layer it means that we cannot share the computation of the ensuing layers across all tasks, and thus the increased number of multiply-adds.

Learned Disentangled Representations: In order to highlight the importance of task modulation, we plot the learned representations for different tasks in various depths of our network. Figure 6 shows the t-SNE representations [35] of the SE activations in equally spaced levels of the network. The activations are averaged for the first 32 samples of the validation set, following [24], and they are sorted per task. The resulting plots show that in the early stages of the network the learned representations are almost identical. They gradually become increasingly different as

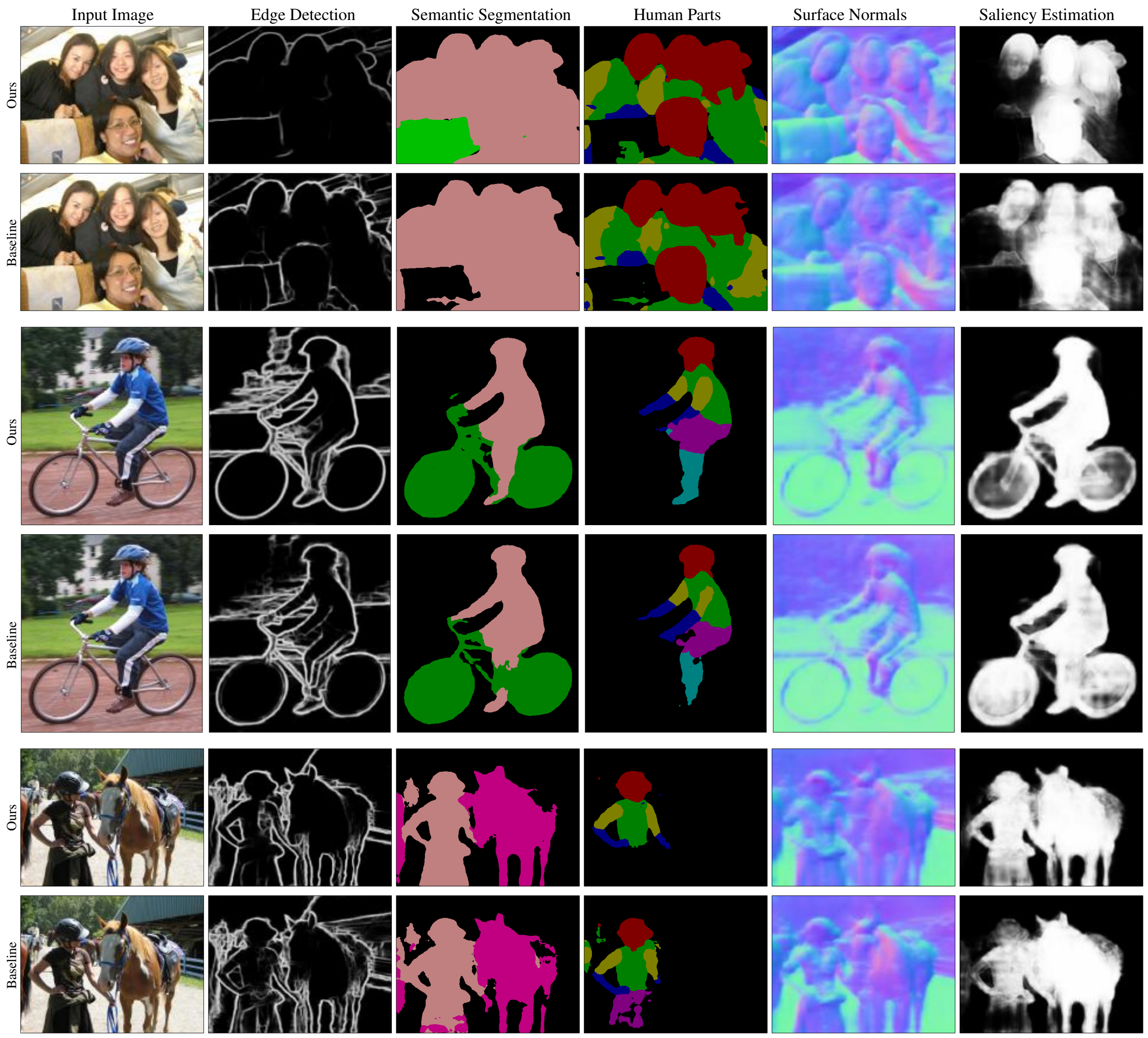


Figure 7. **Qualitative Results on PASCAL**: We compare our model against standard multi-tasking. For the baseline, features from edge detection appear in saliency estimation results (second row), and surface normals are noisy, indicating the need to disentangle the learned representations.

depth grows, until they are completely different for different tasks at the level of the classifier. We argue that this disentanglement of learned representations also translates to performance gains, as shown in Table 3.

Validation on additional datasets: We validate our approach in two additional datasets, NYUD [42] and FSV [29]. NYUD is an indoor dataset, annotated with labels for instance edge detection, semantic segmentation into 41 classes, surface normals, and depth. FSV is a large-scale synthetic dataset, labelled with semantic segmentation (3 classes), albedo, and depth (disparity).

Table 4 presents our findings for both datasets. As in PASCAL, when we try to learn all tasks together, we observe a non-negligible drop compared to the single-tasking baseline. Performance recovers when we plug in modulation and adversarial training. Interestingly, in NYUD and FSV we observe larger improvements compared to PASCAL. Our findings are consistent with related works [64, 13] which report improved results for multi-tasking when using depth and semantics.

Figures 7 and 8 illustrate some qualitative examples, obtained by our method on PASCAL and NYUD, respectively. Results in each row are obtained with a single network. We compare our best model to the baseline architecture for multi-tasking (without per-task modulation, or adversarial training). We observe a quality degradation in the results of

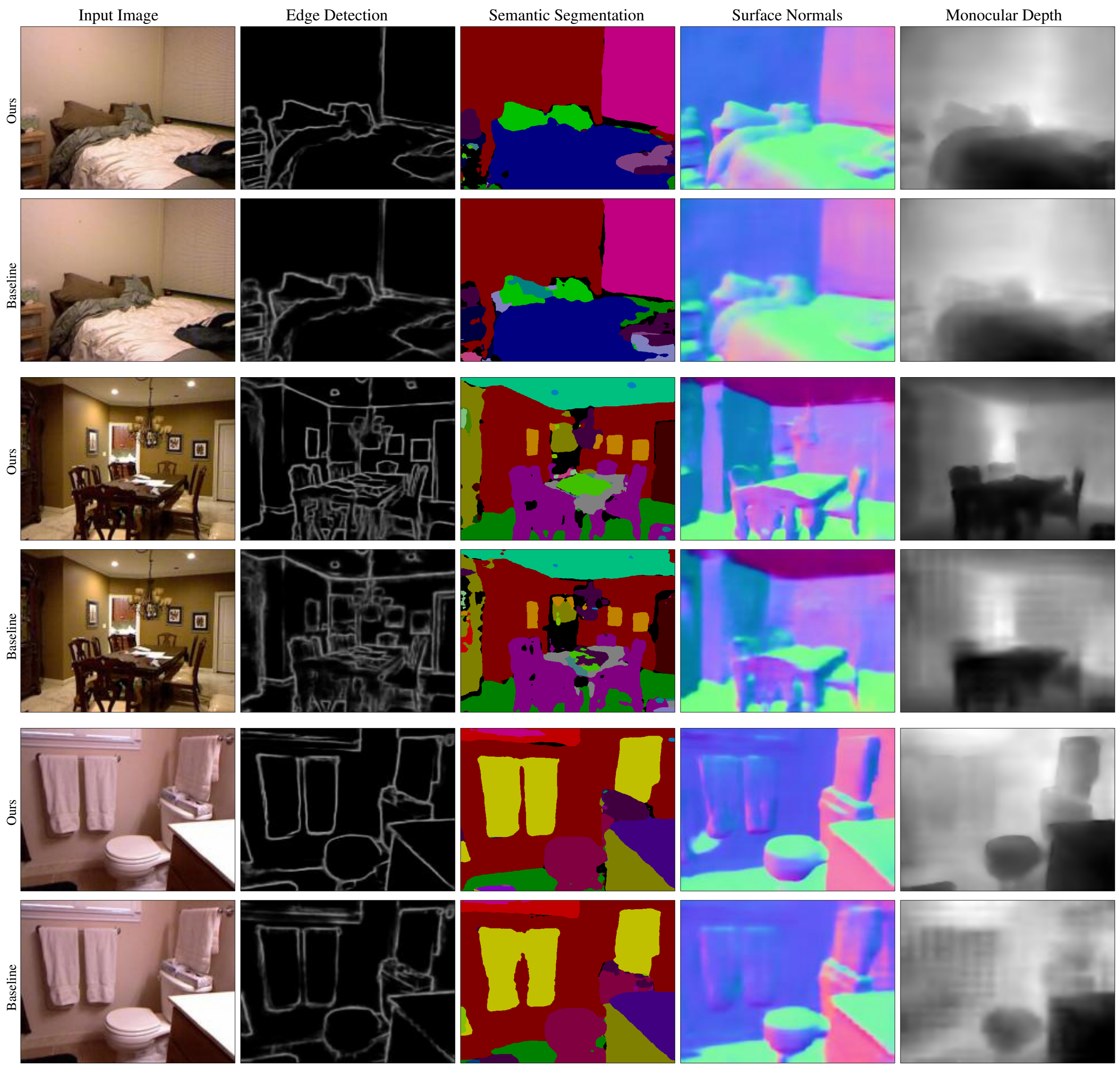


Figure 8. **Qualitative Results on NYUD**: We compare our model against standard multi-tasking. The baseline predicts blurry edges and depth, as well as inconsistent labels on the pillow (where surface normals change). Our method is able to recover from these issues.

the baseline. Interestingly, some errors are obtained clearly as a result of standard multi-tasking. Edge features appear during saliency estimation in Fig 7, and predicted semantic labels change on the pillows, in areas where the surface normals change, in Fig 8. In contrast, our method provides disentangled predictions that are able to recover from such issues, reach, and even surpass the single-tasking baselines.

6. Conclusions

In this work we have shown that we can attain, and even surpass single-task performance through multi-task networks, provided we execute one task at a time. We have achieved this by introducing a method that allows a network to 'focus' on the task at hand in terms of task-specific feature modulation and adaptation.

In a general vision architecture one can think of task attention as being determined based on the operation currently being performed - e.g. using object detection to find an object, normal estimation and segmentation to grasp it. Tasks can also be executed in an interleaved manner, with low-level tasks interacting with high-level ones in a bottom-up/top-down cascade [30]. We intend to explore these directions in future research.

backbone	SEA	#T	Edge ↑	Seg ↑	Norm ↓	Depth ↓	Δ_m% ↓
R-26		1	72.9	29.87	24.34	0.650	0
R-26		4	72.4	27.74	24.83	0.729	5.50
R-26	✓	4	73.5	30.07	24.316	0.625	**-1.36**
R-50		1	74.4	32.82	23.3	0.610	
R-50		4	73.2	30.95	23.34	0.700	5.44
R-50	✓	4	74.5	32.16	23.18	0.570	**-1.22**
R-101		1	74.9	35.90	22.90	0.580	
R-101		4	73.8	31.20	23.07	0.650	6.63
R-101	✓	4	75.6	35.60	22.73	0.560	**-1.07**

(a) Our method on NYUD [42], for different backbones.

backbone	SEA	#T	Seg ↑	Albedo ↓	Depth ↓	Δ_m% ↓
R-26		1	69.77	0.087	0.065	0
R-26		3	66.71	0.090	0.073	6.41
R-26	✓	3	71.36	0.085	0.065	**-1.80**
R-50		1	71.14	0.086	0.063	0
R-50		3	66.90	0.093	0.078	7.04
R-50	✓	3	70.69	0.085	0.063	**-0.02**
R-101		1	72.10	0.086	0.063	0
R-101		3	68.12	0.091	0.072	8.75
R-101	✓	3	72.24	0.083	0.062	**-1.57**

(b) Our method on FSV [29], for different backbones.

Table 5. **Our method for NYUD, and FSV**: Results with different backbones: R-26, R-50, and R-101. Negative drop indicates improved performance with respect to the single-tasking baseline. Arrows indicate desired behaviour of each metric.

Appendix

The following additional information is provided in the appendix:

- Ablated results using different backbone architectures for NYUD and FSV datasets in Appendix A.
- Baselines using UberNet-type networks with and without the proposed modulation and adversarial training in Appendix B.
- Results using MobileNet as the backbone architecture, in order to highlight potential mobile phone applications in Appendix C.
- Technical details for training and testing in Appendix D. Code will also be published.

Appendix A: More results on NYUD and FSV

Table 5 illustrates the quantitative results obtained by our method on NYUD [42] and FSV [29], by changing the backbone architecture. Results are consistent among backbones, and by including modulation and adversarial training to the pipeline, we get improved results with respect to the multi-task and single-task baselines, irrespective of the network depth.

backbone	SEA	#T	Edge ↑	Seg ↑	Parts ↑	Norm ↓	Sal ↑	Δ_m% ↓
R-50-Uber		1	71.7	66.90	59.80	15.00	64.56	
R-50-Uber		5	70.3	60.90	57.00	16.65	62.15	7.10
R-50-Uber	✓	5	70.5	65.50	60.15	14.94	64.98	**0.43**
R-50	✓	5	72.4	68.00	61.10	14.80	65.70	n.a

(a) UberNet results in PASCAL.

backbone	SEA	#T	Edge ↑	Seg ↑	Norm ↓	Depth ↓	Δ_m% ↓
R-50-Uber		1	73.9	32.50	22.90	0.669	0
R-50-Uber		4	72.3	29.48	24.16	0.716	6.00
R-50-Uber	✓	4	73.7	31.19	23.46	0.632	**0.30**
R-50	✓	4	74.5	32.20	23.20	0.570	n.a

(b) UberNet-type results in NYUD.

backbone	SEA	#T	Seg ↑	Albedo ↓	Depth ↓	Δ_m% ↓
R-50-Uber		1	70.26	0.092	0.101	0
R-50-Uber		3	67.02	0.093	0.124	9.50
R-50-Uber	✓	3	69.45	0.091	0.111	**3.32**
R-50	✓	3	70.70	0.085	0.063	n.a

(c) UberNet results in FSV.

Table 6. **UberNet for PASCAL, NYUD, and FSV**: Standard multi-task learning results in a significant drop in performance, that is recovered with modulation and adversarial training. Last rows of each table present results obtained by our architecture.

Appendix B: UberNet-type baseline

The architecture of [28] learns multiple tasks by using a common backbone and a light-weight decoder (skip connections and 1×1 convolutions) per task. We re-implement the UberNet architecture, and we substitute the VGG [59] backbone of the original work with the SE-ResNet [24] used in our work. The main difference with our architecture are the skip connections and 1×1 convolutions that comprise each task-specific head, instead of the powerful Deeplab-v3+ ASPP decoder [6] used throughout our work. Similarly to our work, and similarly to the observation of the original author, we observe a non-trivial drop in performance when learning to multi-task with a common, entirely shared backbone (Table 6). We plug-in SE and adversarial training and we recover most of the drop. We provide results for all 3 datasets that have been used throughout this work. Results obtained by our architecture are presented in the last row of each table. The relatively lower performance especially for semantic tasks compared to our method is due to the absence of a strong decoder. The observations regarding modulation and adversarial training are, nevertheless, consistent.

Appendix C: MobileNet-v2 backbone for multiple tasks

Our multi-tasking framework could find application in light-weight CNNs designed for mobile phone applications. For example, by using our framework, many different tasks

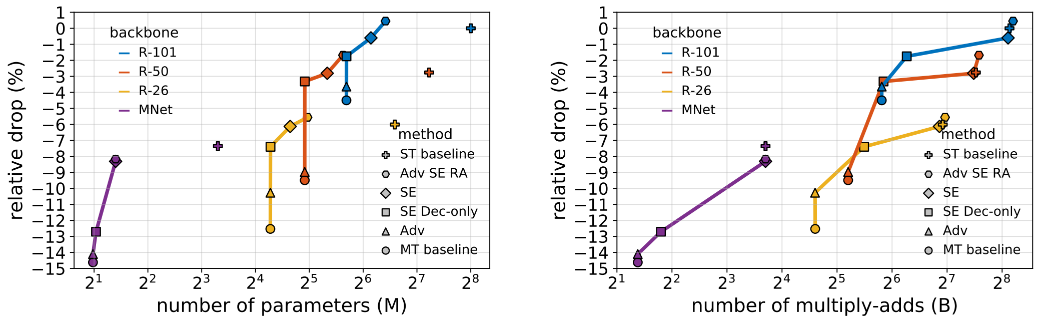


Figure 9. **Performance vs. Resources for MobileNet:** Average relative drop as a function of the number of parameters (left), and multiply-adds (right), for various points of operation of our method. Different backbones are indicated with different colors. MobileNet with our modulation is able to reach R-50 results for standard multi-tasking, by using much less parameters and computation.

backbone	enc	dec	#T	Edge ↑	Seg ↑	Parts ↑	Norm ↓	Sal ↑	$\Delta_m\%$ ↓
MNet			1	69.5	62.10	54.88	14.88	66.30	0
MNet			5	67.2	54.10	53.00	16.76	62.70	7.57
MNet		✓	5	67.5	57.40	54.50	16.55	63.00	5.47
MNet	✓	✓	5	69.2	61.60	55.17	15.21	65.60	**0.97**

Table 7. **Results using MobileNet in PASCAL**: Modulation with SE is able to recover the performance that is lost using standard multi-task learning.

can be executed with only a single and small set of parameters being shipped to the end user. To test this idea, we change our backbone to the light-weight MobileNet-v2 [54], an architecture specifically designed for mobile phones. We change the decoder accordingly: convolutions are changed to depth-wise convolutions, and ReLU activations are changed to ReLU6. We pre-train a variant that uses Squeeze and Excitation modules on ImageNet (SE-MobileNet), and fine-tune for multi-task learning. We test standard multi-tasking against the variants of our method that use SE modulation. Table 7 summarizes our findings. Similarly to the experiments using SE-ResNet, disentangling the representations for each task also helps for MobileNet. By using SE per task both on the encoder and decoder, our method outperforms the single-tasking baseline. Figure 9 puts these results in perspective, comparing them to results obtained by SE-ResNet architecture. It is remarkable that by using modulation, MobileNet is no worse than the R-50 standard multi-tasking baseline using much less computational cost, and only 8% of its parameters.

Appendix D: Implementation Details

In this section we provide the technical details for our implementation.

Generic hyper-parameters: The entire hyper-parameter search was performed on the single-task baselines. For all tasks, we use synchronized SGD with momentum of 0.9 and weight decay of 1e-4. We set the initial learning rate to 0.005 and use the poly learning rate [5]. All our models are trained on a single GPU with batch size 8, and spatial input of 512×512. We used multi-GPU training with batch size 16 and synchronous batchnorm layers only for the sanity check experiments (Table 2 of main paper), for a fair comparison with competing methods. Standard flipping, rotations, and scaling was used for data augmentation. The number of total epochs is set to 60 for PASCAL [14], 200 for NYUD [42], and 3 for the large-scale FSV [29].

Weighting of the losses: Related work deals with automatically weighting of the losses for multi-task learning [8, 56]. We compared these methods to selecting the optimal weights by grid-search. In our setup, we found out that grid-search works better, probably because of the very imbalanced optimal parameters (optimal weighting of loss for edge detection is 50 times higher than semantic segmentation, for example). In particular, we re-implemented [56] that uses multi-objective optimization to re-weight the gradients from each task, in order to optimize the shared parts of the network towards a direction useful for all tasks. This lead to better results than uniform weights, however we obtained better results by simple grid-search.

For all our experiments, when training for multiple tasks, we divide the learning rate of the shared layers by the number of tasks (T), since T updates are happening for the same mini-batch on the shared part of the network.

During training, we used the following formula for the weights of the losses:

$$L = (1 - w_d) \cdot \sum_{t=1}^{T} w_t \cdot L_t + w_d \cdot L_d, \qquad (5)$$

where w_t weights the loss L_t of task t, and w_d the loss L_d of the discriminator. All losses are averaged to the number of samples that the prediction contains ($W \times H \times C \times N$).

Edge Detection: For edge detection, we use $w_t = 50$ and binary cross-entropy loss. As is common practice [63, 27, 36], the positive pixels are weighted more (0.95) than the negative ones (0.05), to account for the class imbalance. When training for a single task in BSDS (Table 2 of main paper), where there are more than a single annotators, we use the multi-instance learning (MIL) loss of [27]. No MIL is used when training in PASCAL or NYUD. We follow the common evaluation on those two datasets [36] by setting the maximum allowed mis-localization of edges (`maxDist` parameter) to 0.0075 and 0.011 for PASCAL and NYUD, respectively.

Semantic Segmentation: For semantic segmentation, we used $w_t = 1$, and cross-entropy loss. When training on VOC trainaug [6] (Table 2 of main paper), we did not finetune separately on VOC val.

Human Part Segmentation: For human part segmentation, we used $w_t = 2.$ and cross-entropy loss. Samples that do not contain any humans did not contribute to the loss.

Surface Normals: For surface normal estimation, we used $w_t = 10$ and $\mathcal{L}_1$ loss with unit-vector normalization. During rotation augmentation, we carefully rotate the unit vector of the surface normals accordingly, to point to a consistent direction.

Albedo: For albedo, we use $w_t = 10$, and standard $\mathcal{L}_1$ loss. We emphasize that our architecture is not optimal for this task, since the output is 4 times smaller than the input, and tiny details are not captured. Using an architecture suited for albedo is out of the scope of this work.

Monocular Depth: For monocular depth estimation we use $w_t = 1$, and the loss of [13], which is a combination of $\mathcal{L}_1$ loss and a smoothness term that enforces the spatial gradients of the prediction to be consistent with the ones of the ground truth. Our experiments showed that including the smoothness term to the loss leads to better results, quantitatively and qualitatively.

Discriminator: We use a fully-convolutional discriminator, which consists of two 1×1 conv layers and a ReLU activation. We did not observe improvements when using discriminators of larger depth. We normalize the gradient of the losses of the tasks by their norm before passing them through the discriminator. This practice makes training more stable because the norm of the gradients becomes smaller as training progresses. We use $w_d = 0.1$.

Further Technical Details: Our work is implemented in PyTorch [43]. During weight update PyTorch applies momentum and weight decay to all modules in the definition of a network. This behaviour is not desired When using generic and task-specific weights, since the task-specific ones are only used in the forward pass of the particular task, which leads to $T - 1$ unwanted updates. This behaviour is avoided by tracing the graph of computation and updating only the weights that were used, which also translated into quantitative improvements.

References

[1] K. Ahmed and L. Torresani. Maskconnect: Connectivity learning by gradient descent. In *ECCV*, 2018. 3

[2] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *CVPR*, 2018. 3

[3] A. Bansal, X. Chen, B. Russell, A. Gupta, and D. Ramanan. Pixelnet: Representation of the pixels, by the pixels, and for the pixels. *arXiv:1702.06506*, 2017. 5, 6

[4] K. Bousmalis, G. Trigeorgis, N. Silberman, D. Krishnan, and D. Erhan. Domain separation networks. In *NIPS*, 2016. 3

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *T-PAMI*, 2017. 3, 5, 11

[6] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 5, 6, 10, 12

[7] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *CVPR*, 2014. 6

[8] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *arXiv:1711.02257*, 2018. 2, 11

[9] M.-M. Cheng, N. J. Mitra, X. Huang, P. H. Torr, and S.-M. Hu. Global contrast based salient region detection. *T-PAMI*, 37(3):569–582, 2015. 6

[10] H. Drucker and Y. Le Cun. Double backpropagation increasing generalization performance. In *IJCNN*, 1991. 2, 4

[11] V. Dumoulin, J. Shlens, and M. Kudlur. A learned representation for artistic style. In *ICLR*, 2017. 3

[12] N. Dvornik, K. Shmelkov, J. Mairal, and C. Schmid. Blitznet: A real-time deep network for scene understanding. In *ICCV*, 2017. 2

[13] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015. 1, 2, 8, 12

[14] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. 5, 6, 11

[15] T. Evgeniou and M. Pontil. Regularized multi–task learning. In *KDD*, 2004. 4

[16] J. Fu, H. Zheng, and T. Mei. Look closer to see better: Recurrent attention convolutional neural network for fine-grained image recognition. In *CVPR*, 2017. 3

[17] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015. 3, 4, 5

[18] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. In *JMLR*, 2016. 4

[19] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 2

[20] M. Guo, A. Haque, D.-A. Huang, S. Yeung, and L. Fei-Fei. Dynamic task prioritization for multitask learning. In *ECCV*, 2018. 2

[21] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011. 6

[22] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2

[23] J. Hu, L. Shen, S. Albanie, G. Sun, and A. Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. *arXiv:1810.12348*, 2018. 3

[24] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 3, 4, 7, 10

[25] A. Kendall, Y. Gal, and R. Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *CVPR*, 2018. 2

[26] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 3

[27] I. Kokkinos. Pushing the boundaries of boundary detection using deep learning. In *ICLR*, 2016. 5, 11, 12

[28] I. Kokkinos. UberNet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, 2017. 1, 2, 5, 6, 7, 10

[29] P. Krähenbühl. Free supervision from video games. In *CVPR*, 2018. 5, 8, 10, 11

[30] T. S. Lee and D. Mumford. Hierarchical bayesian inference in the visual cortex. *J. Opt. Soc. Am. A*, 20(7):1434–1448, Jul 2003. 10

[31] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille. The secrets of salient object segmentation. In *CVPR*, 2014. 6

[32] P. Liu, X. Qiu, and X. Huang. Adversarial multi-task learning for text classification. In *ACL*, 2017. 2, 3, 4

[33] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *NIPS*, 2017. 3

[34] J. Lu, J. Yang, D. Batra, and D. Parikh. Hierarchical question-image co-attention for visual question answering. In *NIPS*, 2016. 3

[35] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *JMLR*, 9(Nov):2579–2605, 2008. 7

[36] K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. V. Gool. Convolutional oriented boundaries: From image segmentation to high-level tasks. *T-PAMI*, 40(4):819 – 833, 2018. 11, 12

[37] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *T-PAMI*, 26(5):530–549, 2004. 6

[38] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001. 5

[39] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *CVPR*, 2016. 3

[40] P. K. Mudrakarta, M. Sandler, A. Zhmoginov, and A. Howard. K for the price of 1: Parameter efficient multi-task and transfer learning. In *ICLR*, 2019. 3

[41] C. Murdock, Z. Li, H. Zhou, and T. Duerig. Blockout: Dynamic model selection for hierarchical deep networks. In *CVPR*, 2016. 3

[42] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012. 5, 6, 8, 10, 11

[43] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017. 12

[44] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. FiLM: Visual reasoning with a general conditioning layer. *arXiv:1709.07871*, 2017. 2, 3

[45] A. Ranjan, V. Jampani, K. Kim, D. Sun, J. Wulff, and M. J. Black. Adversarial collaboration: Joint unsupervised learning of depth, camera motion, optical flow and motion segmentation. *arXiv:1805.09806*, 2018. 2

[46] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Learning multiple visual domains with residual adapters. In *NIPS*, 2017. 3

[47] S.-A. Rebuffi, H. Bilen, and A. Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *CVPR*, 2018. 2, 3

[48] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2

[49] Z. Ren and Y. J. Lee. Cross-domain self-supervised multi-task feature learning using synthetic imagery. In *CVPR*, 2018. 2

[50] A. Rosenfeld, M. Biparva, and J. K. Tsotsos. Priming neural networks. *arXiv*, 2018. 3

[51] A. Rosenfeld and J. K. Tsotsos. Incremental learning through deep adaptation. *T-PAMI*, 2018. 3

[52] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 5

[53] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. Progressive neural networks. *arXiv:1606.04671*, 2016. 3

[54] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. 11

[55] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, et al. Grad-CAM: Visual explanations from deep networks via gradient-based localization. In *ICCV*, 2017. 3

[56] O. Sener and V. Koltun. Multi-task learning as multi-objective optimization. In *NIPS*, 2018. 2, 11

[57] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 1

[58] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv:1312.6034*, 2013. 3

[59] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 10

[60] A. Sinha, Z. Chen, V. Badrinarayanan, and A. Rabinovich. Gradient adversarial training of neural networks. *arXiv:1806.08028*, 2018. 2, 4

[61] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. CBAM: Convolutional block attention module. In *ECCV*, 2018. 3

[62] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018. 2

[63] S. Xie and Z. Tu. Holistically-nested edge detection. *IJCV*, pages 1–16, 2017. 11

[64] D. Xu, W. Ouyang, X. Wang, and N. Sebe. PAD-Net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *CVPR*, 2018. 2, 5, 8

[65] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2015. 3

[66] L. Yang, Y. Wang, X. Xiong, J. Yang, and A. K. Katsaggelos. Efficient video object segmentation via network modulation. In *CVPR*, 2018. 3

[67] A. R. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese. Taskonomy: Disentangling task transfer learning. In *CVPR*, 2018. 2

[68] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014. 3

[69] X. Zhao, H. Li, X. Shen, X. Liang, and Y. Wu. A modulation module for multi-task learning with applications in image retrieval. In *ECCV*, 2018. 3

[70] Y. Zou, Z. Luo, and J.-B. Huang. DF-Net: Unsupervised joint learning of depth and flow using cross-task consistency. In *ECCV*, 2018. 2