Reparameterizing Convolutions for Incremental Multi-Task Learning without Task Interference

Menelaos Kanakis[1], David Bruggemann[1], Suman Saha[1],
Stamatios Georgoulis[1], Anton Obukhov[1], and Luc Van Gool[1,2]

[1] ETH Zurich [2] KU Leuven

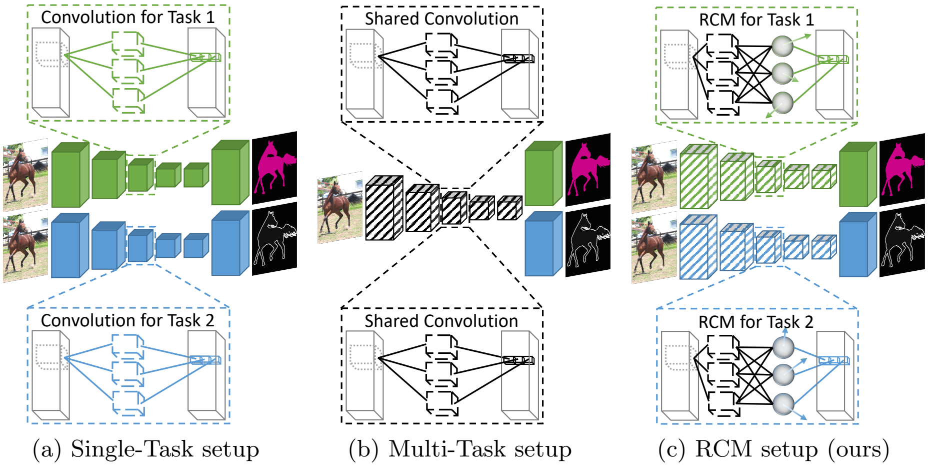
Abstract. Multi-task networks are commonly utilized to alleviate the need for a large number of highly specialized single-task networks. However, two common challenges in developing multi-task models are often overlooked in literature. First, enabling the model to be inherently incremental, continuously incorporating information from new tasks without forgetting the previously learned ones (incremental learning). Second, eliminating adverse interactions amongst tasks, which has been shown to significantly degrade the single-task performance in a multi-task setup (task interference). In this paper, we show that both can be achieved simply by reparameterizing the convolutions of standard neural network architectures into a non-trainable shared part (filter bank) and task-specific parts (modulators), where each modulator has a fraction of the filter bank parameters. Thus, our reparameterization enables the model to learn new tasks without adversely affecting the performance of existing ones. The results of our ablation study attest the efficacy of the proposed reparameterization. Moreover, our method achieves state-of-the-art on two challenging multi-task learning benchmarks, PASCAL-Context and NYUD, and also demonstrates superior incremental learning capability as compared to its close competitors. The code and models are made publicly available[1].

Keywords: Multi-Task Learning, Incremental Learning, Task Interference

1 Introduction

Over the last decade, convolutional neural networks (CNNs) have been established as the standard approach for many computer vision tasks, like image classification [25, 54, 17], object detection [15, 48, 32], semantic segmentation [33, 3, 63], and monocular depth estimation [12, 26]. Typically, these tasks are handled by CNNs independently, i.e., a separate model is optimized for each task, resulting in several task-specific models (Fig. 1a). However, real-world problems are more complex and require models to perform multiple tasks on-demand without significantly compromising each task's performance. For example, an interactive advertisement system tasked with displaying targeted content to its audience

[1] https://github.com/menelaoskanakis/RCM



(a) Single-Task setup (b) Multi-Task setup (c) RCM setup (ours)

Fig. 1. (a) Optimizing independent models per task allows for the easy addition of new tasks, at the expense of a multiplicative increase in the total number of parameters with respect to a single model (green and blue denote task-specific parameters). **(b)** A single backbone for multiple tasks must be meaningful to all, thus, all tasks interact with the said backbone (black indicates common parameters). **(c)** Our proposed setup, RCM (Reparameterized Convolutions for Multi-task learning), uses a pre-trained filter bank (denoted in black) and independently optimized task-specific modulators(denoted in colour) to adapt the filter bank on a per-task basis. New task addition is accomplished by training the task-specific modulators, thus explicitly addressing task interference while parameters scale at a slower rate than having independent models per task.

should be able to detect the presence of humans in its viewpoint effectively, estimate their gender and age group, recognize their head pose, etc. At the same time, there is a need for flexible models able to gradually add more tasks to their knowledge, without forgetting previously known tasks or having to re-train the whole model from scratch. For instance, a car originally deployed with lane and pedestrian detection functionalities can be extended with depth estimation capabilities post-production.

When it comes to learning multiple tasks under a single model, multi-task learning (MTL) techniques [2, 50] have been employed in the literature. On the one hand, encoder-focused approaches [38, 24, 34, 10, 40, 31, 1, 57] emphasize learning feature representations from multi-task supervisory signals by employing architectures that encode shared and task-specific information. On the other hand, decoder-focused approaches [59, 61, 62, 58] utilize the multi-task feature representations learned at the encoding stage to distill cross-task information at the decoding stage, thus refining the original feature representations. In both cases, however, the joint learning from multiple supervisory signals (i.e., tasks) can hinder the individual task performance if the associated tasks point to conflicting gradient directions during the update step of the shared feature representations (Fig. 1b). Formally this is known as *task interference* or *negative transfer* and has been well documented in the literature [24, 36, 65]. To suppress negative transfer, several approaches [6, 21, 55, 16, 65, 52, 36] dynamically re-weight each task's loss function or re-order the task learning, to find a 'sweet spot' where individual task performance does not degrade significantly. Arguably, such approaches mainly focus on mitigating the negative transfer problem in the MTL architectures above, rather than eliminating it (see Section 3.2). At the same time, existing works seem to disregard the fact that MTL models are commonly desired to be incremental, i.e., information from new tasks should be continuously incorporated while existing task knowledge is preserved. In existing works, the MTL model has to be re-trained from scratch if the task dictionary changes; this is arguably sub-optimal.

Recently, task-conditional networks [36] emerged as an alternative for MTL, inspired by work in multi-domain learning [45, 46]. That is, performing separate forward passes within an MTL model, one for each task, every time activating a set of task-specific residual responses on top of the shared responses. Note that, this is useful for many real-world setups (e.g., an MTL model deployed in a mobile phone with limited resources that adapts its responses according to the task at hand), and particularly for incremental learning (e.g., a scenario where the low-level tasks should be learned before the high-level ones). However, the proposed architecture in [36] is prone to task interference due to the inherent presence of shared modules, which is why the authors introduced an adversarial learning scheme on the gradients to minimize the performance degradation. Moreover, the model needs to be trained from scratch if the task dictionary changes.

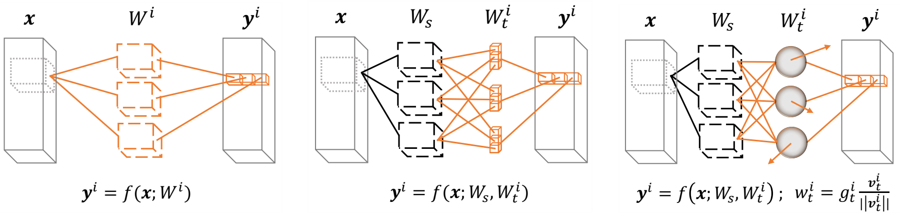
All given, existing works primarily focus on either improving the multi-task performance or reducing the number of parameters and computations in the MTL model. In this paper, we take a different route and explicitly tackle the problems of incremental learning and task interference in MTL. We show that both problems can be addressed simply by reparameterizing the convolutional operations of a neural network. In particular, building upon the task-conditional MTL direction, we propose to decompose each convolution into a shared part that acts as a filter bank encoding common knowledge, and task-specific modulators that adapt this common knowledge uniquely for each task. Fig. 1c illustrates our approach, RCM (Reparameterized Convolutions for Multi-task learning). Unlike existing works, the shared part in our case is not trainable to explicitly avoid negative transfer. Most notably, as any number of task-specific modulators can be introduced in each convolution, our model can incrementally solve more tasks without interfering with the previously learned ones. Our results demonstrate that the proposed RCM can outperform state-of-the-art methods in multi-task (Section 4.6) and incremental learning (Section 4.7) experiments. At the same time, we address the common multi-task challenge of task interference by construction, by ensuring tasks can only update task-specific components and cannot interact with each other.

2 Related Work

Multi-task learning (MTL) aims at developing models that can solve a multitude of tasks [2, 50]. In computer vision, MTL approaches can roughly be divided into encoder-focused and decoder-focused ones. Encoder-focused approaches primarily emphasize on architectures that can encode multi-purpose feature representations through supervision from multiple tasks. Such encoding is typically achieved, for example, via feature fusion [38], branching [24, 40, 34, 57], self-supervision [10], attention [31], or filter grouping [1]. Decoder-focused approaches start from the feature representations learned at the encoding stage, and further refine them at the decoding stage by distilling information across tasks in a one-off [59], sequential [61], recursive [62], or even multi-scale [58] manner. Due to the inherent layer sharing, the approaches above typically suffer from task interference. Several works proposed to dynamically re-weight the loss function of each task [6, 21, 55, 52], sort the order of task learning [16], or adapt the feature sharing between 'related' and 'unrelated' tasks [65], to mitigate the effect of negative transfer. In general, existing MTL approaches have primarily focused on improving multi-task performance or reducing the network parameters and computations. Instead, in this paper, we look at the largely unexplored problems of incremental learning and negative transfer in MTL models and propose a principled way to tackle them.

Incremental learning (IL) is a paradigm that attempts to augment the existing knowledge by learning from new data. IL is often used, for example, when aiming to add new classes [47] to an existing model, or learn new domains [30]. It aims to mitigate 'catastrophic forgetting' [14], the phenomenon of forgetting old tasks as new ones are learned. To minimize the loss of existing knowledge, Li and Hoiem [30] optimized the new task while preserving the old task's responses. Other works [23, 28] constrained the optimization process to minimize the effect learning has on weights important for older tasks. Rebuffi et al. [47] utilized exemplars that best approximate the mean of the learned classes in the feature space to preserve performance. Note that the performance of such techniques is commonly upper bounded by the joint training of all tasks. More relevant to our work, in a multi-domain setting, a few approaches [45, 46, 49, 35] utilize a pre-trained network that remains untouched and instead learn domain-specific components that adapt the behavior of the network to address the performance drop common in IL techniques. Inspired by this research direction, we investigate the training of parts of the network, while keeping the remaining components constant from initialization amongst all tasks. This technique not only addresses catastrophic forgetting but also task interference, which is crucial in MTL.

Decomposition of filters and tensors within CNNs has been explored in the literature. In particular, filter-wise decomposition into a product of low-rank filters [20], filter groups [44], a basis of filter groups [29], etc. have been utilized. In contrast, tensor-wise examples include SVD decomposition [9, 60], CP-decomposition [27], Tucker decomposition [22], Tensor-Train decomposition [42],



(a) Standard Conv.

(b) RC without NFF

(c) RC with NFF

Fig. 2. (a) A standard convolutional module for a given task i, with task-specific weights W^i in orange. **(b)** A reparameterized convolution (RC) consisting of a shared filter bank W_s in black, and task-specific modulator W_t^i in orange. **(c)** An RC with Normalized Feature Fusion (NFF), consisting of a shared filter bank W_s in black, and task-specific modulator W_t^i in orange. Each row $\boldsymbol{w}_t^i$ of W_t^i is reparameterized as $g_t^i \cdot \boldsymbol{v_t^i} / \parallel \boldsymbol{v_t^i} \parallel$.

Tensor-Ring decomposition [64], T-Basis [41], etc. These techniques have been successfully used for compressing neural networks or reducing their inference time. Instead, in this paper, we utilize decomposition differently. We decompose each convolutional operation into two components: a shared and a task-specific part. Note that although we utilize the SVD decomposition for simplicity, the same principles hold for other decomposition types too.

3 Reparameterizing CNNs for Multi-Task Learning

In this section, we present techniques to adapt a CNN architecture, such that it can increasingly learn new tasks in an MTL setting while scaling more efficiently than simply adding single-task models. Section 3.1 introduces the problem formulation. Section 3.2 demonstrates the effect of task interference in MTL and motivates the importance of CNN reparameterization. Section 3.3 presents techniques to reparameterize CNNs and limit the parameter increase with respect to task-specific models.

3.1 Problem Formulation

Given P tasks and input tensor $\boldsymbol{x}$, we aim to learn a function $f(\boldsymbol{x}; W_s, W_t^i) = \boldsymbol{y}^i$ that holds for task $i = 1, 2, \dots P$, where W_s and W_t^i are the shared and task-specific parameters respectively. Unlike existing approaches [34, 38] which learn such functions $f(\cdot)$ on the layer level of the network, i.e., explicitly designing shared and task-specific layers, we aim to learn f on a block-level by *reparameterizing* the convolutional operation, and adapting its behaviour conditioned on the task i, as depicted in Fig. 2b and Fig. 2c. By doing so, we can explicitly address the task interference and catastrophic forgetting problems within an MTL setting.

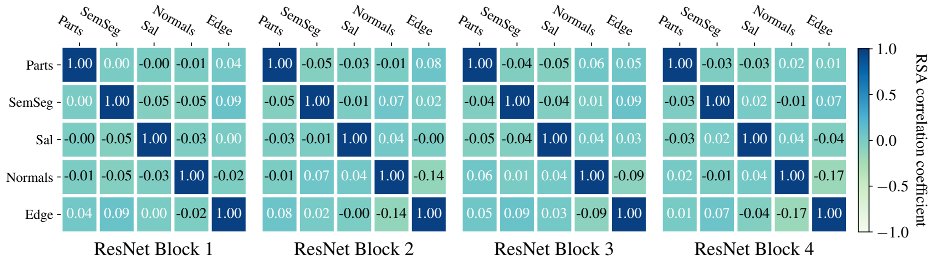


Fig. 3. Visualization of the Representation Similarity Analysis (RSA) on the task-specific gradients at different depths of a ResNet-26 model [36]. The analysis was conducted on: human parts segmentation (Parts), semantic segmentation (SemSeg), saliency estimation (Sal), normals estimation (Normals), and edge detection (Edge).

3.2 Task Interference

To motivate the importance of addressing task interference by construction, we analyze the task-specific gradient directions on the shared modules of a state-of-the-art MTL model. Specifically, we utilize the work of [36], who used a discriminator to enforce indistinguishable gradients amongst tasks.

We acquire the gradients from the training dataset of PASCAL-Context [39] for each task, using minibatches of size 128, yielding 40 minibatches. We then use the Representation Similarity Analysis (RSA), proposed in [11] for transfer learning, as a means to quantify the correlation of the gradients amongst the different tasks. Fig. 3 depicts the task gradient correlations at different depths of a ResNet-26 model [17], trained to have indistinguishable gradients in the output layer [36]. It can be seen that there is a limited gradient correlation amongst the tasks, demonstrating that addressing task interference indirectly (here with the use of adversarial learning on the gradients) is a very challenging problem. We instead follow a different direction and propose to utilize reparameterizations with shared components amongst different tasks that are untouched during the training process, and each task being able to optimize only its parameters. As such, task interference is eliminated by construction.

3.3 Reparameterizing Convolutions

We define a convolutional operation $f(\boldsymbol{x};\boldsymbol{w}) = y$ for the single-task learning setup, Fig. 2a. $\boldsymbol{w} \in \mathbb{R}^{k^2 c_{in}}$ denotes the parameters of a single convolutional layer (we omit the bias to simplify notation) for a kernel size k and c_{in} channels. $\boldsymbol{x} \in \mathbb{R}^{k^2 c_{in}}$ is the input tensor volume at a given spatial location ($\boldsymbol{x}$ and $\boldsymbol{w}$ are expressed in vector notation), and y is the scalar response. Assuming c_{out} such filters, the convolutional operator can be rewritten in matrix notation as $f(\boldsymbol{x};W) = \boldsymbol{y}$, where $\boldsymbol{y} \in \mathbb{R}^{c_{out}}$ provides c_{out} responses, and $W \in \mathbb{R}^{c_{out} \times k^2 c_{in}}$. In a single-task setup:

$$f(\boldsymbol{x};W^1) = \boldsymbol{y}^1, \ \ldots\ , \ f(\boldsymbol{x};W^P) = \boldsymbol{y}^P \tag{1}$$

where W^i and $\boldsymbol{y}^i$ are the task-specific parameters and responses for a given convolutional layer, respectively. The total number of parameters for this setup is $\mathcal{O}(Pk^2c_{in}c_{out})$. Our goal is to reparameterize $f(\cdot)$ in Eqn. 1 as:

$$f(\boldsymbol{x}; W^i) = h(\boldsymbol{x}; W_s, W_t^i), \quad \forall i = 1, \ldots, P \tag{2}$$

using a set of shared ($W_s \in \mathbb{R}^{c_{out} \times k^2 c_{in}}$) and task-specific ($W_t^i \in \mathbb{R}^{c_{out} \times c_{out}}$) parameters for each convolutional layer of the backbone. Our formulation aims to retain the prediction performance of the original convolutional layer (Eq. 1), while simultaneously reducing the rate in which the total number of parameters grows. The complexity now becomes $\mathcal{O}((k^2c_{in} + Pc_{out})c_{out})$, which is less than $\mathcal{O}(Pk^2c_{in}c_{out})$ for standard layers. We argue that this reparameterization is necessary for coping with task interference and incremental learning in an MTL setup, in which we only optimize for task-specific parameters W_t^i, while keeping the shared parameters W_s intact. Note that, when adding a new task $i = \omega$, we do not need to train the entire network from scratch as in [36]. We only optimize W_t^ω for each layer of the reparameterized CNN.

We denote our reparameterized convolutional layer as a matrix multiplication between the two sets of parameters: $W_t^iW_s$. In order to find a set of parameters $W_t^iW_s$ that approximates the single-task weights W^i a natural choice is to minimize the Frobenius norm $\|W_t^iW_s - W^i\|_F$ directly. Even though direct minimization of this metric is appealing due to its simplicity, it poses some major caveats. (i) It assumes that all directions in the parameter space affect the final performance for task i in the same way and are thus penalized uniformly. However, two different solutions for W_t^i with the same Frobenius norm can yield drastically different losses. (ii) This approximation is performed independently for each convolutional layer, neglecting the chain effect an inaccurate prediction in one layer can have in the succeeding layers. In the remainder of this section, we propose different techniques to address these limitations.

Reparameterized Convolution. We implement the Reparameterized Convolution (RC) $W_t^iW_s$ as a stack of two 2D convolutional layers without non-linearity in between, with W_s having a spatial filter size k and W_t^i being a 1×1 convolution (Fig. 2b)[2]. We optimize only W_t^i directly on the task-specific loss function using stochastic gradient descent while keeping the shared weights W_s constant. This ensures that training for one task is independent of other tasks, ruling out interference amongst tasks while optimizing the metric of interest.

Normalized Feature Fusion. One can view $\boldsymbol{w}_t^i$, a row in matrix W_t^i, as a soft filter adaptation mechanism, i.e., a modulator which generates new task-specific filters from a given filter bank W_s, depicted in Fig. 2b. However, instead of training the vector $\boldsymbol{w}_t^i$ directly, we propose its reparameterization into two

[2] To ensure compliance with ImageNet [8] initialization, the new architecture is first pre-trained on ImageNet using the publicly available training script from PyTorch [43].

terms, a vector term $\boldsymbol{v}_t^i \in \mathbb{R}^{c_{out}}$, and a scalar term g_t^i as:

$$\boldsymbol{w}_t^i = g_t^i \frac{\boldsymbol{v}_t^i}{\parallel \boldsymbol{v}_t^i \parallel}, \tag{3}$$

where $\parallel \cdot \parallel$ denotes the Euclidean norm. We refer to this reparameterization as Normalized Feature Fusion (NFF), depicted in Fig. 2c. NFF provides an easier optimization process in comparison to an unconstrained $\boldsymbol{w}_t^i$. This reparametrization enforces $\boldsymbol{v}_t^i / \parallel \boldsymbol{v}_t^i \parallel$ to be unit length and point in the direction which best merges the filter bank. The vector norm $\parallel \boldsymbol{w}_t^i \parallel = g_t^i$ learns independently the appropriate scale of the newly generated filters, and thus the scale of the activation. Directly optimizing $\boldsymbol{w}_t^i$ attempts to learn both jointly, which is a harder optimization problem. Normalizing weight tensors has been generally explored for speeding up the convergence of the optimization process [7, 51, 56]. In our work, we use it differently and demonstrate empirically (see Section 4.5) that such a reparameterization in series with a filter bank also improves performance in the MTL setting. As seen in Eq. 3, additional learnable parameters are introduced in the training process (g_t^i and $\boldsymbol{v}_t^i$), however, $\boldsymbol{w}_t^i$ can be computed after training and used directly for deployment, eliminating additional overhead.

Response Initialization. We build upon the findings of matrix/tensor decomposition literature [9, 60] that network weights/responses lie on a low dimensional subspace. We further assume that such a subspace can be beneficial for multiple tasks, and thus good for network initialization under a MTL setup. To this end, we identify a meaningful subspace of the responses for the generation of a better filter bank W_s when compared to that directly learned by pre-training W_s on ImageNet. More formally, let $\boldsymbol{y} = f(\boldsymbol{x}; W^m)$ be the responses for input tensor $\boldsymbol{x}$, where $W^m \in \mathbb{R}^{c_{out} \times k^2 c_{in}}$ are the pre-trained ImageNet weights. We define $Y \in \mathbb{R}^{c_{out} \times n}$ as a matrix containing n responses of $\boldsymbol{y}$ with the mean vector $\overline{\boldsymbol{y}}$ subtracted. We compute the eigen-decomposition of the covariance matrix $YY^T = USU^T$ (using Singular Value Decomposition, SVD), where $U \in \mathbb{R}^{c_{out} \times c_{out}}$ is an orthogonal matrix with the eigenvectors on the columns, and S is a diagonal matrix of the corresponding eigenvalues. We can now initialize the shared convolution parameters W_s with $U^T W^m$, and the task-specific W_t^i with U. We refer to this initialization methodology as Response Initialization (RI). We point the reader to the supplementary material for more details.

4 Experiments

4.1 Datasets

We focus our evaluation on dense prediction tasks, making use of two datasets. We conduct the majority of the experiments on PASCAL [13], and more specifically, PASCAL-Context [39]. We address edge detection (Edge), semantic segmentation (SemSeg), human parts segmentation (Parts), surface normals estimation (Normals), and saliency (Sal). We evaluate single-task performance using optimal dataset F-measure (odsF) [37] for edge detection, mean intersection over union (mIoU) for semantic segmentation, human parts and saliency, and finally mean error (mErr) for surface normals. Labels for human parts segmentation are acquired from [5], while for saliency and surface normals from [36].

We further evaluate the proposed method on the smaller NYUD dataset [53], comprised of indoor scenes, on edge detection (Edge), semantic segmentation (SemSeg), surface normals estimation (Normals), and depth (Depth). The evaluation metrics for edge detection, semantic segmentation, and surface normals estimation are identical to those for PASCAL-Context, while for depth we use root mean squared error (RMSE).

4.2 Architecture

All of our experiments make use of the DeepLabv3+ architecture [4], originally designed for semantic segmentation, which performs competitively for all tasks of interest as demonstrated in [36]. DeepLabv3+ encodes multi-scale contextual information by utilizing a ResNet [17] encoder with a-trous convolutions [3] and an a-trous spatial pyramid pooling (ASPP) module, while a decoder with a skip connection refines the predictions. Unless otherwise stated, we use a ResNet-18 (R-18) based DeepLabv3+, and report the mean performance of five runs for each experiment[3].

4.3 Evaluation Metric

We follow standard practice [36, 58] and quantify the performance of a model m as the average per-task performance drop with respect to the corresponding single-task baseline b:

$$\Delta_m = \frac{1}{P} \sum_{i=1}^{P} (-1)^{l_i} \frac{M_{m,i} - M_{b,i}}{M_{b,i}} \tag{4}$$

where l_i is either 1 or 0 if a lower or a greater value indicates better performance, respectively, for a performance measure M. P indicates the total number of tasks.

4.4 Analysis of network module sharing

We investigate the level of task-specific adaptation required for a common backbone to perform competitively to single-task models, while additionally eliminating negative transfer. In other words, the necessity for task-specific modules, i.e., convolutions (Convs) and batch normalizations (BNs) [19]. Specifically, we optimize for task-specific Convs, BNs, or both along the network's depth. Modules that are not being optimized maintain their ImageNet pre-trained parameters. Table 1 presents the effect on performance, while Fig. 4 depicts the total

[3] Baseline comparisons to competing methods, as well as additional backbone experiments, can be found in the supplementary material.

Table 1. Performance analysis of task-specific modules. We report the effect network modules (Convs and BNs) have on the performance of PASCAL-Context.

Method	Convs	BNs	Edge ↑	SemSeg ↑	Parts ↑	Normals ↓	Sal ↑	$\Delta_m\%$ ↓
Freeze encoder			67.32	60.37	47.86	17.40	58.39	14.98
Task-specific BNs		✓	69.80	63.93	53.22	14.78	64.44	5.76
Task-specific Convs	✓		71.72	66.00	59.05	13.78	66.31	0.62
Single-task	✓	✓	71.88	66.22	59.69	13.64	66.62	-

number of parameters with respect to the number of tasks. Experiments vary from common Convs and BNs (Freeze encoder) to task-specific Convs and BNs (Single-task), and anything in-between.

The model utilizing a common backbone pre-trained on ImageNet (Freeze encoder), as expected, is unable to perform competitively to the single-task counterpart, with a performance drop of 14.98%. Task-specific BNs significantly improve performance with a percentage drop of 5.76%, at a minimal increase in parameters (Fig. 4). The optimization of Convs is essential for competitive performance to single-task, with a percentage drop of 0.62%. However, the increase in parameters is comparable to single-task, which is undesirable (Fig. 4).

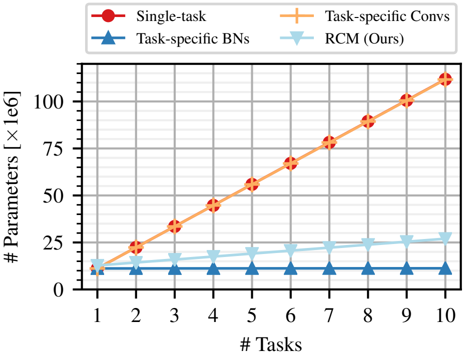


Fig. 4. Backbone parameter scaling. Total number of parameters with respect to the number of tasks for R-18 backbone.

4.5 Ablation study

To validate the proposed methodology from Section 3, we conduct an ablation study, presented in Table 2. We additionally report the performance of a model trained jointly on all tasks, consisting of a fully shared encoder and task-specific decoders (Multi-task). This multi-task model is not trained in an IL setup but merely serves as a reference to the traditional multi-tasking techniques. We report a performance drop of 3.32% with respect to the single-task setup.

Reparameterized Convolution. We first develop a new baseline for our proposed reparameterization, where we replace every convolution with the RC (Section 3.3) counterpart. As seen in Table 2, RC achieves a performance drop of 2.13%, outperforming the 3.32% drop of the multi-task baseline, as well as the Task-specific BNs (Table 1) that achieved a performance drop of 5.76%. This observation corroborates the claim made in Section 4.4 that task-specific adaptation of the convolutions is essential for a model to perform competitively for

Table 2. Ablation study of the proposed RCM. We present ablation experiments for the proposed Reparameterized Convolution (RC), Response Initialization (RI), Normalized Feature Fusion (NFF) on PASCAL-Context dataset.

Method	NFF	RI	Edge ↑	SemSeg ↑	Parts ↑	Normals ↓	Sal ↑	Δ_m% ↓
Single-task			71.88	66.22	59.69	13.64	66.62	-
Multi-task			70.74	62.43	57.89	14.43	66.31	3.32
RC			71.10	64.56	56.87	13.91	66.37	2.13
RC+NFF	✓		71.12	64.71	56.91	13.90	66.33	2.07
RC+RI		✓	71.36	65.58	57.99	13.70	66.21	1.12
RC+RI+NFF	✓	✓	71.34	65.70	58.12	13.70	66.38	0.99

all tasks. Additionally, we demonstrate that even without training entirely task-specific convolutions, as in Table 1 (Task-specific Convs), a performance boost can still be observed at a smaller magnitude, while the total number of parameters scales at a slower rate (Fig. 4). RCM in Fig. 4 depicts the parameter scaling of all the RC-based methods introduced in Table 2, described in this section. As such, improvements in performance from this baseline do not stem from an increase in network capacity.

Response Initialization. We investigate the effect on the performance of a more meaningful filter bank, RI (Section 3.3), against the filter bank learned by directly pre-training the RC architecture on ImageNet. In Table 2 we report the performance of our proposed model when directly pre-trained on ImageNet (Table 2-RC), and with the RI based filter bank (Table 2-RC+RI). Compared to the RC model, the performance significantly improves from a 2.13% drop to a 1.12% drop with the RC+RI model. This observation clearly demonstrates that the filter bank generated using our proposed RI approach is beneficial for better weight initialization.

Normalized Feature Fusion. We replace the unconstrained task-specific components of RC with the proposed NFF (Section 3.3). We demonstrate in Table 2 that NFF improves the performance no matter the initialization of the filter bank. RC improves from a 2.13% drop to a 2.07% in RC+NFF, while RC+RI improved from a 1.12% drop to 0.99% for RC+RI+NFF.

The architecture used for the remaining experiments is the Reparameterized Convolution (RC) with Normalized Feature Fusion (NFF), initialized using the Response Initialization (RI) methodology. This architecture is denoted as RCM.

4.6 Comparison to state-of-the-art

In this work, we focus on comparing to task-conditional methods that can address MTL. We compare the performance of our method to Series Residual Adapter (Series RA) [45] and Parallel RA [46]. Series and Parallel RAs learn multiple visual domains by optimizing domain-specific residual adaptation modules (rather than using RCM as in our work, Fig. 2c) on an ImageNet pre-trained backbone.

Table 3. Comparison with state-of-the-art methods on PASCAL-Context.

Method	Edge ↑	SemSeg ↑	Parts ↑	Normals ↓	Sal ↑	$\Delta_m\%$ ↓
Single-task	71.88	66.22	59.69	13.64	66.62	-
ASTMT (R-18 w/o SE) [36]	71.20	64.31	57.79	15.06	66.59	3.49
ASTMT (R-26 w SE) [36]	71.00	64.61	57.25	15.00	64.70	4.12
Series RA [45]	70.62	65.99	55.32	14.27	66.08	2.97
Parallel RA [46]	70.84	66.51	56.56	14.16	66.36	2.09
RCM (ours)	71.34	65.70	58.12	13.70	66.38	**0.99**

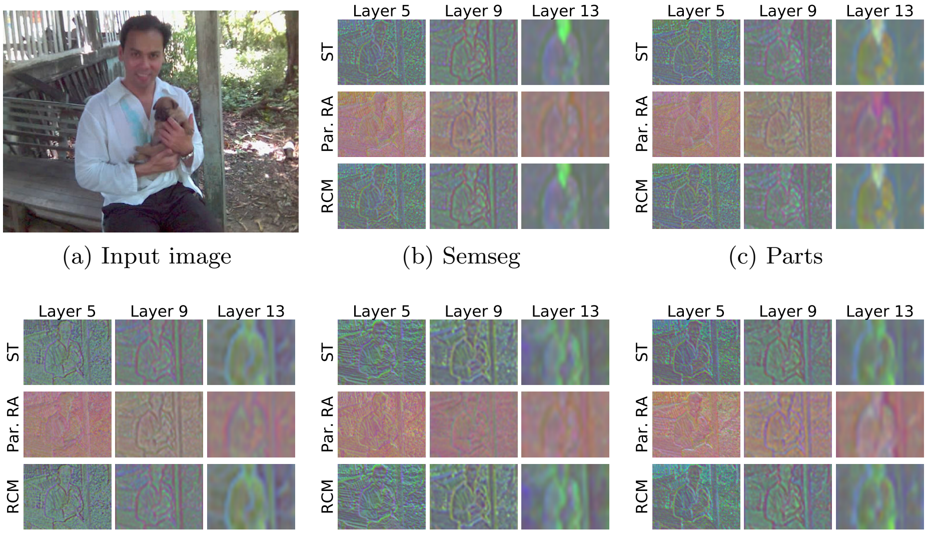
Table 4. Comparison with state-of-the-art methods on NYUD.

Method	Edge ↑	SemSeg ↑	Normals ↓	Depth ↓	$\Delta_m\%$ ↓
Single-task	68.83	35.45	22.20	0.56	-
ASTMT (R-18 w/o SE) [36]	68.60	30.69	23.94	0.60	6.96
ASTMT (R-26 w SE) [36]	73.50	30.07	24.32	0.63	7.56
Series RA [45]	67.56	31.87	23.35	0.60	5.88
Parallel RA [46]	68.02	32.13	23.20	0.59	5.02
RCM (ours)	68.44	34.20	22.41	0.57	**1.48**

Since both methods were developed for multi-domain settings, we optimize them using our own pipeline, ensuring a fair comparison amongst the methods while additionally benchmarking the capabilities of multi-domain methods in a multi-task setup. We further report the performance of ASTMT [36], which utilizes an architecture resembling that of Parallel RA [46] with Squeeze-and-Excitation (SE) blocks [18] and adversarial task disentanglement of gradients. Specifically, we report the performance of the models using a ResNet-26 (R-26) DeepLab-V3+ with SE as reported in [36], and also optimize with the use of their codebase a ResNet-18 model without SE. The latter model uses an architecture resembling more closely that of the other methods since SE can be additionally incorporated in the others as well. We report the average performance drop with respect to our single-task baseline.

The results for PASCAL-Context (Table 3) and NYUD (Table 4) demonstrate that our method achieves the best performance, outperforming the other methods that make use of RA modules. This demonstrates that although the RA can perform competitively in multi-domain settings, placing the convolution in series without non-linearity is a more promising direction for the drastic adaptations required for different tasks in a multi-task learning setup.

We visualize in Fig. 5 the learned representations of single-task, Parallel RA [46], and RCM across tasks and network depths. For each task and layer combination, we compute a common PCA basis for the methods above and depict the first three principal components as RGB values. For all tasks and layers of the network, the representations of RCM closely resemble those of the single-task models. Simultaneously, Parallel RA is unable to adapt the convolution behavior to the extent required to be comparable to single-task models.



(a) Input image (b) Semseg (c) Parts

(d) Edge (e) Normals (f) Sal

Fig. 5. Feature visualizations. We visualize the features of the input image (a) for the tasks of PASCAL-Context. The first row of each sub-figure corresponds to the responses of the single-task model (ST), the second row those of Parallel RA (Par. RA) [46] and the final row of our proposed method (RCM). For all tasks and depths of the network, the responses of RCM closely resemble those of ST, in contrast to the responses of Par. RA. This is made apparent from the colours utilized by the different methods. The RGB values were identified from a common PCA basis across the three methods in order to highlight similarities and differences between them.

4.7 Incremental learning for multi-tasking

We further evaluate the methods from Section 4.6 in the incremental learning (IL) setup. In other words, we investigate the capabilities of the models to learn new tasks without the need to be completely retrained on the entire task dictionary. We divide the tasks of PASCAL-Context into three groups, **(i)** edge detection and surface normals (low-level tasks), **(ii)** saliency (mid-level task) and **(iii)** semantic segmentation and human parts segmentation (high-level tasks). IL experiments are conducted by allowing the base network to initially use knowledge from either (i) or (iii), and reporting the capability for the optimized model to learn additional tasks without affecting the performance of the already learned tasks (the performance drop is calculated over the new tasks that were not used in the initial training). In the IL setup, ASTMT [36] is initially trained using an R-18 backbone without SE (a comparable backbone to the competing methods for a fair comparison) on the subset of the tasks (either i or iii). New tasks can be incorporated by training their task-specific modules independently. On the other hand, Series RA, Parallel RA, and RCM, were designed to be inherently incremental due to directly optimizing only the task-specific modules. Consequently,

Table 5. Incremental learning experiments on a network originally trained on the low-level tasks (Edge and Normals) of PASCAL-Context.

Method	Edge ↑	Normals ↓	SemSeg ↑	Parts ↑	Sal ↑	$\Delta_m\%$ ↓
Single-task	71.88	13.64	66.22	59.69	66.62	-
ASTMT (R-18 w/o SE) [36]	70.70	14.84	55.32	50.49	64.34	11.77
Series RA [45]	70.62	14.27	65.99	55.32	66.08	2.83
Parallel RA [46]	70.84	14.16	66.51	56.56	66.36	1.73
RCM (ours)	71.34	13.70	65.70	58.12	66.38	**1.26**

Table 6. Incremental learning experiments on a network originally trained on the high-level tasks (SemSeg and Parts) of PASCAL-Context.

Method	SemSeg ↑	Parts ↑	Edge ↑	Normals ↓	Sal ↑	$\Delta_m\%$ ↓
Single-task	66.22	59.69	71.88	13.64	66.62	-
ASTMT (R-18 w/o SE) [36]	63.91	57.33	68.67	14.12	64.43	3.76
Series RA [45]	65.99	55.32	70.62	14.27	66.08	2.39
Parallel RA [46]	66.51	56.56	70.84	14.16	66.36	1.88
RCM (ours)	65.70	58.12	71.34	13.70	66.38	**0.52**

their task-specific performance in the IL setup is identical to that reported in Section 4.6.

In Tables 5 and 6 we report the performance of tasks that are utilized to generate the initial knowledge of the model in grey (important for ASTMT [36]), while in black the performance of the incrementally learned tasks. As shown in both tables, and in particular Table 5, ASTMT does not perform competitively in the IL experiments. This observation further demonstrates the importance of utilizing generic filter banks that can be adapted based on the task-specific needs, in particular for IL setups. We consider research in generic multi-task filter banks to be a promising direction.

5 Conclusion

We have presented a novel method of a convolutional operation reparameterization and its application to training multi-task learning architectures. These reparameterized architectures can be applied on a multitude of different tasks, and allow the CNN to be inherently incremental, while additionally eliminating task interference, all by construction. We evaluate our model on two datasets and multiple tasks, and show experimentally that it outperforms competing baselines that address similar challenges. We further demonstrate its efficacy when compared to the state-of-the-art task-conditional multi-task method, which is unable to tackle incremental learning.

Acknowledgments. This work was sponsored by Advertima AG and co-financed by Innosuisse. We thank the anonymous reviewers for their valuable feedback.

References

1. Bragman, F.J., Tanno, R., Ourselin, S., Alexander, D.C., Cardoso, J.: Stochastic filter groups for multi-task cnns: Learning specialist and generalist convolution kernels. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1385–1394 (2019)
2. Caruana, R.: Multitask learning. Machine learning **28**(1), 41–75 (1997)
3. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence **40**(4), 834–848 (2017)
4. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: Proceedings of the European conference on computer vision (ECCV). pp. 801–818 (2018)
5. Chen, X., Mottaghi, R., Liu, X., Fidler, S., Urtasun, R., Yuille, A.: Detect what you can: Detecting and representing objects using holistic models and body parts. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1971–1978 (2014)
6. Chen, Z., Badrinarayanan, V., Lee, C.Y., Rabinovich, A.: Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. arXiv preprint arXiv:1711.02257 (2017)
7. Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 933–941. JMLR. org (2017)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition. pp. 248–255. Ieee (2009)
9. Denton, E.L., Zaremba, W., Bruna, J., LeCun, Y., Fergus, R.: Exploiting linear structure within convolutional networks for efficient evaluation. In: Advances in neural information processing systems. pp. 1269–1277 (2014)
10. Doersch, C., Zisserman, A.: Multi-task self-supervised visual learning. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2051–2060 (2017)
11. Dwivedi, K., Roig, G.: Representation similarity analysis for efficient task taxonomy & transfer learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 12387–12396 (2019)
12. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: Advances in neural information processing systems. pp. 2366–2374 (2014)
13. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision **88**(2), 303–338 (2010)
14. French, R.M.: Catastrophic forgetting in connectionist networks. Trends in cognitive sciences **3**(4), 128–135 (1999)
15. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014)
16. Guo, M., Haque, A., Huang, D.A., Yeung, S., Fei-Fei, L.: Dynamic task prioritization for multitask learning. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 270–287 (2018)

17. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
18. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
19. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. pp. 448–456 (2015)
20. Jaderberg, M., Vedaldi, A., Zisserman, A.: Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.3866 (2014)
21. Kendall, A., Gal, Y., 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. pp. 7482–7491 (2018)
22. Kim, Y.D., Park, E., Yoo, S., Choi, T., Yang, L., Shin, D.: Compression of deep convolutional neural networks for fast and low power mobile applications. arXiv preprint arXiv:1511.06530 (2015)
23. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al.: Overcoming catastrophic forgetting in neural networks. Proceedings of the national academy of sciences **114**(13), 3521–3526 (2017)
24. Kokkinos, I.: Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6129–6138 (2017)
25. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
26. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., Navab, N.: Deeper depth prediction with fully convolutional residual networks. In: 2016 Fourth international conference on 3D vision (3DV). pp. 239–248. IEEE (2016)
27. Lebedev, V., Ganin, Y., Rakhuba, M., Oseledets, I., Lempitsky, V.: Speeding-up convolutional neural networks using fine-tuned cp-decomposition. arXiv preprint arXiv:1412.6553 (2014)
28. Lee, S.W., Kim, J.H., Jun, J., Ha, J.W., Zhang, B.T.: Overcoming catastrophic forgetting by incremental moment matching. In: Advances in neural information processing systems. pp. 4652–4662 (2017)
29. Li, Y., Gu, S., Gool, L.V., Timofte, R.: Learning filter basis for convolutional neural network compression. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 5623–5632 (2019)
30. Li, Z., Hoiem, D.: Learning without forgetting. IEEE transactions on pattern analysis and machine intelligence **40**(12), 2935–2947 (2017)
31. Liu, S., Johns, E., Davison, A.J.: End-to-end multi-task learning with attention. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1871–1880 (2019)
32. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision. pp. 21–37. Springer (2016)
33. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3431–3440 (2015)

34. Lu, Y., Kumar, A., Zhai, S., Cheng, Y., Javidi, T., Feris, R.: Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5334–5343 (2017)
35. Mallya, A., Davis, D., Lazebnik, S.: Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 67–82 (2018)
36. Maninis, K.K., Radosavovic, I., Kokkinos, I.: Attentive single-tasking of multiple tasks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1851–1860 (2019)
37. Martin, D.R., Fowlkes, C.C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE transactions on pattern analysis and machine intelligence **26**(5), 530–549 (2004)
38. Misra, I., Shrivastava, A., Gupta, A., Hebert, M.: Cross-stitch networks for multitask learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3994–4003 (2016)
39. Mottaghi, R., Chen, X., Liu, X., Cho, N.G., Lee, S.W., Fidler, S., Urtasun, R., Yuille, A.: The role of context for object detection and semantic segmentation in the wild. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 891–898 (2014)
40. Neven, D., De Brabandere, B., Georgoulis, S., Proesmans, M., Van Gool, L.: Fast scene understanding for autonomous driving. arXiv preprint arXiv:1708.02550 (2017)
41. Obukhov, A., Rakhuba, M., Georgoulis, S., Kanakis, M., Dai, D., Van Gool, L.: T-basis: a compact representation for neural networks. In: Proceedings of Machine Learning and Systems 2020, pp. 8889–8901 (2020)
42. Oseledets, I.V.: Tensor-train decomposition. SIAM Journal on Scientific Computing **33**(5), 2295–2317 (2011)
43. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems. pp. 8024–8035 (2019)
44. Peng, B., Tan, W., Li, Z., Zhang, S., Xie, D., Pu, S.: Extreme network compression via filter group approximation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 300–316 (2018)
45. Rebuffi, S.A., Bilen, H., Vedaldi, A.: Learning multiple visual domains with residual adapters. In: Advances in Neural Information Processing Systems. pp. 506–516 (2017)
46. Rebuffi, S.A., Bilen, H., Vedaldi, A.: Efficient parametrization of multi-domain deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8119–8127 (2018)
47. Rebuffi, S.A., Kolesnikov, A., Sperl, G., Lampert, C.H.: icarl: Incremental classifier and representation learning. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 2001–2010 (2017)
48. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 779–788 (2016)
49. Rosenfeld, A., Tsotsos, J.K.: Incremental learning through deep adaptation. IEEE transactions on pattern analysis and machine intelligence (2018)
50. Ruder, S.: An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098 (2017)

51. Salimans, T., Kingma, D.P.: Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In: Advances in neural information processing systems. pp. 901–909 (2016)
52. Sener, O., Koltun, V.: Multi-task learning as multi-objective optimization. In: Advances in Neural Information Processing Systems. pp. 527–538 (2018)
53. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from rgbd images. In: European conference on computer vision. pp. 746–760. Springer (2012)
54. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
55. Sinha, A., Chen, Z., Badrinarayanan, V., Rabinovich, A.: Gradient adversarial training of neural networks. arXiv preprint arXiv:1806.08028 (2018)
56. Srebro, N., Shraibman, A.: Rank, trace-norm and max-norm. In: International Conference on Computational Learning Theory. pp. 545–560. Springer (2005)
57. Vandenhende, S., Georgoulis, S., De Brabandere, B., Van Gool, L.: Branched multi-task networks: Deciding what layers to share. arXiv preprint arXiv:1904.02920 (2019)
58. Vandenhende, S., Georgoulis, S., Van Gool, L.: Mti-net: Multi-scale task interaction networks for multi-task learning. arXiv preprint arXiv:2001.06902 (2020)
59. Xu, D., Ouyang, W., Wang, X., Sebe, N.: Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 675–684 (2018)
60. Zhang, X., Zou, J., He, K., Sun, J.: Accelerating very deep convolutional networks for classification and detection. IEEE transactions on pattern analysis and machine intelligence **38**(10), 1943–1955 (2015)
61. Zhang, Z., Cui, Z., Xu, C., Jie, Z., Li, X., Yang, J.: Joint task-recursive learning for semantic segmentation and depth estimation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 235–251 (2018)
62. Zhang, Z., Cui, Z., Xu, C., Yan, Y., Sebe, N., Yang, J.: Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 4106–4115 (2019)
63. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2881–2890 (2017)
64. Zhao, Q., Zhou, G., Xie, S., Zhang, L., Cichocki, A.: Tensor ring decomposition. arXiv preprint arXiv:1606.05535 (2016)
65. Zhao, X., Li, H., Shen, X., Liang, X., Wu, Y.: A modulation module for multi-task learning with applications in image retrieval. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 401–416 (2018)