The RGB-D Triathlon: Towards Agile Visual Toolboxes for Robots

Fabio Cermelli[1], Massimiliano Mancini[2,3], Elisa Ricci[3,4] and Barbara Caputo[1,5]

***Abstract*— Deep networks have brought significant advances in robot perception, enabling to improve the capabilities of robots in several visual tasks, ranging from object detection and recognition to pose estimation, semantic scene segmentation and many others. Still, most approaches typically address visual tasks in isolation, resulting in overspecialized models which achieve strong performances in specific applications but work poorly in other (often related) tasks. This is clearly sub-optimal for a robot which is often required to perform simultaneously multiple visual recognition tasks in order to properly act and interact with the environment. This problem is exacerbated by the limited computational and memory resources typically available onboard to a robotic platform. The problem of learning flexible models which can handle multiple tasks in a lightweight manner has recently gained attention in the computer vision community and benchmarks supporting this research have been proposed. In this work we study this problem in the robot vision context, proposing a new benchmark, the RGB-D Triathlon, and evaluating state of the art algorithms in this novel challenging scenario. We also define a new evaluation protocol, better suited to the robot vision setting. Results shed light on the strengths and weaknesses of existing approaches and on open issues, suggesting directions for future research.**

I. INTRODUCTION

Recent years have witnessed great advances in computer and robot vision thanks to deep networks [1]. Deep models are used in many applications in robot vision, ranging from egomotion estimation [2] to depth prediction [3], [4], object grasping [5], [6] semantic segmentation [7], [8], etc.

A common procedure for addressing a specific visual recognition problem consists in fine-tuning an existing pre-trained model on a given, problem-specific dataset. While this strategy leads to excellent performance on the specific problem and setting, it ignores two key aspects of robot vision. The first is that robots need visual abilities for several tasks. For instance, to complete a simple carrying task a robot needs to localize itself, detect and recognize the objects in front of it and estimate their poses (see Fig. 1). Clearly, having an overspecialized network for each specific task would scale poorly. Second, the multiple tasks that a robot is required to solve are often closely related (see Fig. 1). Hence, addressing the tasks jointly would probably lead to an increased accuracy as well as to an improved computational efficiency with respect to training task-specific networks.

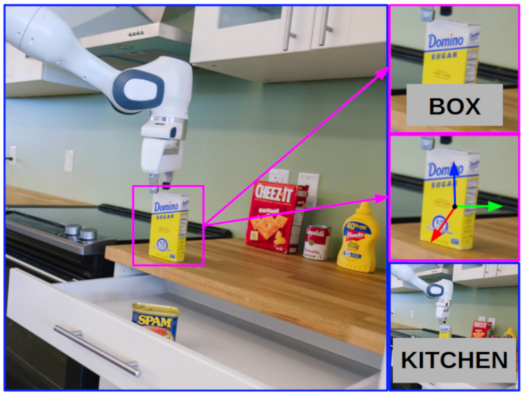


Fig. 1. The ability of a robot to perform multiple tasks at the same time is crucial. For example, to manipulate correctly the object in the scene above, a robot should understand the class of the object, its orientation and the overall context where it operates. Image courtesy of NVIDIA Robotics Research Lab, Seattle (news.developer.nvidia.com/nvidia-opens-robotics-research-lab-in-seattle/).

To deal with this issue, over the years Multi- Task Learning (MTL) [9] approaches have been developed. MTL refers to jointly learn a set of classification/regression models, each associated to a specific task, by leveraging information about task relatedness, *e.g.* sharing models' parameters. By reducing the number of parameters, MTL also decreases the time needed for model training and inference. MTL has been successfully applied in robotics: Teichmann *et al.* [10] proposed an architecture performing jointly object detection, classification, and semantic segmentation. Similarly, in [11] Rahmatizadeh *et al.* described a technique to train a controller to perform several complex picking and placing tasks.

MTL assumes that the data for all the tasks are available during training. In robotics this assumption is often unrealistic and the ability to add new tasks sequentially is crucial [12], [13]. However, a well-known problem of sequential learning is that, while learning how to perform a new task, an algorithm typically forgets about previous tasks. This *catastrophic forgetting* [14], [15] must be kept into account while developing visual recognition models.

Sequential Multi-Task Learning (SMTL) algorithms [16], [17], [18], [19], [20] automatically address the catastrophic forgetting problem by considering a common network backbone and introducing a small set of task-specific network parameters. In practice, during training, for each new task these methods instantiate and learn few task-specific parameters (see Fig. 2), while the other network's weights are kept fixed. While interesting and effective, these approaches

This work was partially supported by the ERC project RoboExNovo (B.C.).

F. Cermelli and B. Caputo are with Politecnico di Torino, Turin, Italy. s236363@studenti.polito.it, barbara.caputo@polito.it

[2] M. Mancini is with Sapienza University of Rome, Rome, Italy. mancini@diag.uniroma1.it

[3] M. Mancini and E. Ricci are with Fondazione Bruno Kessler, Trento, Italy. eliricci@fbk.eu

[4] E. Ricci is with University of Trento, Trento, Italy.

[5] B. Caputo is with Italian Institute of Technology, Milan, Italy.

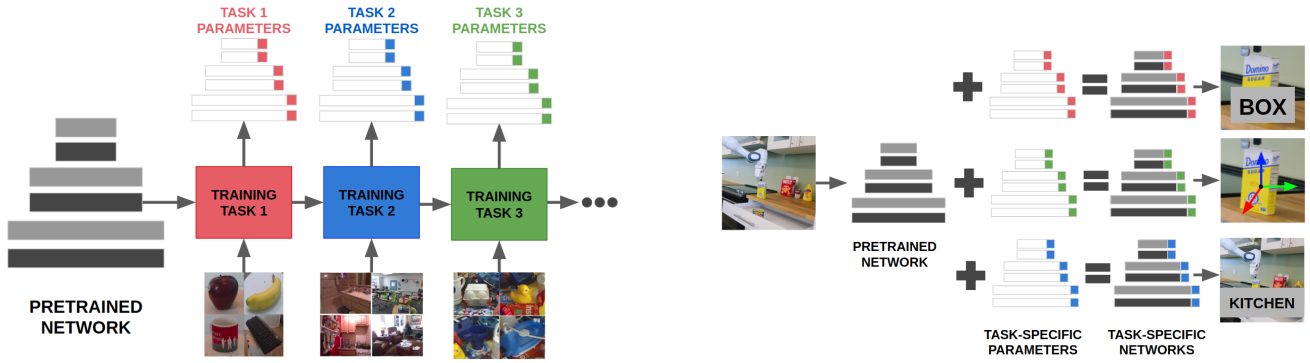


Fig. 2. The Sequential Multi-Task learning problem. During training (left) a sequence of tasks are presented to the network, one by one. For each task, a set of task-specific parameters are learned (colored blocks), freezing the shared ones (grey blocks). During inference (right) multiple tasks can be solved by combining the pretrained network with the task-specific parameters, thus keeping low the overhead in terms of memory requirements.

have been considered only in computer vision and it is not clear if they can also be used in a robotic setting where the computational/memory requirements are especially relevant.

This paper aims at studying SMTL in the robot vision context, presenting the first SMTL benchmark for robot vision while testing state of the art algorithms on this setting. Moreover, to take into account the peculiarities of the robotics scenario, we propose a new evaluation protocol which allows to compare different SMTL algorithms considering not only their recognition performances but also their memory requirements and their ability to deal with multiple/different input modalities (*i.e.* RGB-D).

Contributions. To summarize, the contributions of this work are three-fold.

- We propose the first benchmark for sequential multi-task learning in robotics, the RGB-D Triathlon. It considers different input modalities and three fundamental tasks: object recognition, pose estimation and scene classification. On the considered dataset, we evaluate several state of the art SMTL methods: the serial [16] and parallel [17] residual adapters, Piggyback [19] and Binarized Affine Transfom (BAT) [20].
- We propose a new metric for the SMTL task which considers both model accuracy and the memory/computational requirements.
- We release a toolbox enabling researchers to develop their own method for SMTL and compare it with baseline methods. Our code can be found at `https://github.com/fcdl94/RobotChallenge`.

II. RELATED WORKS

Our work is related to many previous studies in the area of visual learning, *e.g.* those addressing the problems of incremental and multi-task learning. In the following we first review recent benchmarks proposed in computer and robot vision involving multiple tasks. Then we describe the relation between our work and previous studies on incremental and multi-task learning. We consider only deep neural models, due to their clear advantages in performance compared to shallow models.

Learning from Multiple Tasks/Domains: Benchmarks. Benchmarks are fundamental tools to advance research in visual recognition and robot perception. For instance, in robot vision, datasets as SeqSLAM [21], RGB-D SLAM [22], allowed the development of methods for addressing loop-closing, SLAM [23] and semantic mapping [24]. In computer vision, datasets and challenges as ImageNet [25], Common Objects in Context (COCO) [26] and Places [27] have brought significant progresses, enabling to learn deep architectures which are not only effective for a single categorization problem but that can also serve as general purpose models to address other recognition tasks [1].

Recently, the idea of learning universal representations [28] and task-agnostic deep models, able to perform well over multiple tasks and/or domains, has attracted attention [16], [20], fostering the creation of new datasets and challenges. A notable example is the Robust Vision Challenge (http://www.robustvision.net), which aims to facilitate the development of robust models, *i.e.* models which must perform well on a specific problem (*e.g.* depth estimation, semantic segmentation) but on different settings (*e.g.* environments, sensors). Another interesting benchmark is the Visual Domain Decathlon [16], which promotes the developments of SMTL models considering ten different categorization problems. Drawing inspiration from these initiatives in the computer vision community, in this paper we propose a novel benchmark to stimulate research in robot perception. In particular, with respect to the aforementioned SMTL benchmarks, we consider a setting where i) the tasks to be addressed are different (namely, pose estimation, object recognition and scene classification); ii) the evaluation protocol is newly designed, in order to take into account not only the per-task performances but also the computational complexity of the model; iii) different input modalities are considered (*i.e.* RGB, Depth and RGB-D).

Sequential Multi-Task Learning. Given a pretrained model and a set of tasks whose data are available at *different* times, the goal of Sequential Multi-Task Learning [20] is to learn a network able to address all tasks while keeping low the overhead in terms of required parameters. In computer vision, this task has been tackled by methods addressing the aforementioned Visual Domain Decathlon challenge [16], [17], [19], [20]. These models differ on how they extend a pretrained model to new tasks, considering for instance adding task-specific network modules [16], [17] or changing the network parameters by means of binary masks [19], [20]. In robot perception, this task has not been addressed yet, thus we will take these models as baseline approaches.

Multi-Task and Incremental Learning. The formulation of our task is strictly related to MTL and incremental learning. Similar to ours, the goal of MTL is to train a model on multiple tasks. Different works addressed MTL in various contexts of robot perception, ranging from reinforcement learning [29], learning from demonstrations [11], grasping [30] and scene understanding [10]. Differently from standard MTL settings, we focus on the sequential setting, where we do not have access to the data of all the tasks beforehand but we receive them one after the other, sequentially.

SMTL is also closely related to incremental learning [13]. Our aim is also to progressively add knowledge to a model without forgetting the previously learned concepts, overcoming the problem of catastrophic forgetting [31]. Still, differently from the incremental and the incremental class learning problems [32], [33], [34], [35], we want to add tasks to the network and not consider new categories, thus we need a separate output space for each new task.

III. An SMTL benchmark for Robot Perception

As stated above, the aim of this work is to foster research in developing robot systems which are able to sequentially learn to perform several visual tasks by applying SMTL methodologies. This is crucial for many reasons. First, the memory resources typically available in a robotic platform are limited, thus it is practically unfeasible to store a new deep network for each novel visual task a robot is asked to solve. Second, we may be interested in extending the visual capabilities of an existing robotic system, in order to solve new tasks not considered at the initial stage of deployment. In a nutshell, SMTL techniques aim to learn a set of classification/regression models for multiple tasks sequentially. The challenge is to learn new task-specific models while keeping the number of task-specific parameters as low as possible.

Formally, the SMTL task is defined as follows. Suppose we have a pretrained deep model, with parameters Θ_0 and a set $\mathcal{T} = \{\mathcal{T}_1, \cdots, \mathcal{T}_N\}$ of N tasks. For each task we have an input space $\mathcal{X} \subset \Re^{H\times W\times C}$, with C depending on the input modality (*i.e.* 1 for depth maps, 3 for RGB images and 4 for RGB-D), a feature space $\mathcal{Z}_t$ and an output space $\mathcal{Y}_t$. For simplicity, let us define as $\phi_{\Theta_t} : \mathcal{X} \to \mathcal{Z}_t$ a task-specific mapping from the input space to the feature space $\mathcal{Z}_t$. The mapping is parametrized by the set $\Theta_t = \{\theta_0,\ \theta_t\}$ which comprises the shared ($\theta_0 \in \Theta_0$) and task-specific (θ_t) parameters. Moreover, let us denote as $\psi_{\Omega_t} : \mathcal{Z}_t \to \mathcal{Y}_t$ the function mapping the feature space to the task-specific output space, parametrized by Ω_t. In our case, Ω_t are just the weights of the output layer of our neural network. Finally, let us define the mapping from the input to the output space $\Phi_t : \mathcal{X} \to \mathcal{Y}_t$, where $\Phi_t = \psi_{\Omega_t} \circ \phi_{\Theta_t}$. We also denote as $\rho_t = |\theta_t|$, the memory size (in bits) required to store the task-specific parameters $\theta_t \in \Theta_t$ and as α_t a task-specific performance measure (*e.g.* classification accuracy). The goal of an SMTL algorithm is to learn a set of task-specific mappings Φ_t while i) maximizing α_t and ii) keeping ρ_t as low as possible for any $\mathcal{T}_t \in \mathcal{T}$.

To stimulate the research community in robot vision on SMTL, in this paper we propose a dataset consisting of three different tasks and introduce an evaluation metric suitable to the STML setting. The dataset is a collection of three datasets commonly used in the robotics community, but in this paper we propose their joint adoption for studying the novel problem of SMTL.

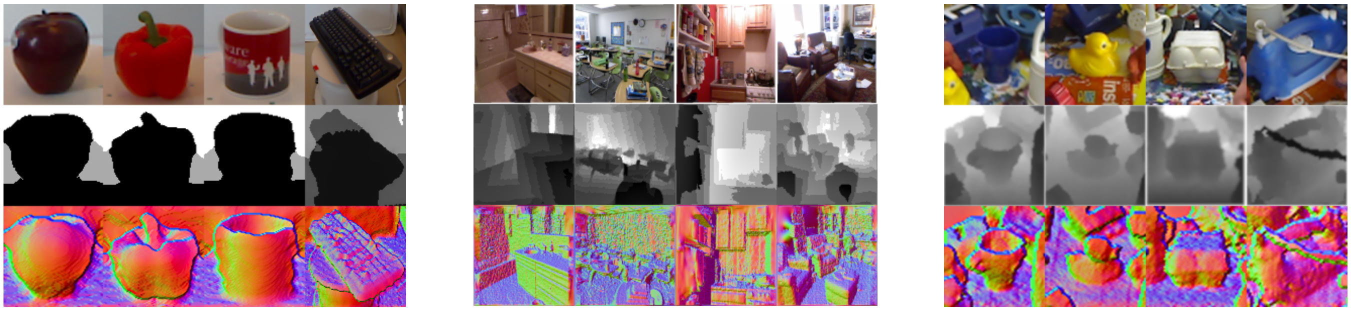
In the following subsections, we present the dataset and the tasks we proposed. Then, we describe the novel evaluation protocol we introduced for SMTL in robotics.

A. *The RGB-D Triathlon Dataset*

The proposed dataset considers three fundamental tasks: object recognition, pose estimation and scene classification. In choosing the tasks and benchmarks we follow three principles, namely i) the importance of the perception task for robotics; ii) the impact that the dataset has had in the past within the community; iii) the possibility of defining a standard training/testing protocol. Differently from challenges in computer vision [16], we propose three different settings: i) use only RGB images, ii) use only depth images, iii) use both RGB and depth images. Each setting is independent from the others, such as to permit the researchers to work only on the setting which is more relevant to their application.

Task 1. Object Recognition. Object recognition is of utmost importance in robot perception. This task consists in assigning to an input image depicting an object the corresponding semantic class label. Object recognition algorithms are fundamental tools for robots because i) knowing the category of an object allows reasoning on how the object can be manipulated and ii) they enable more complex tasks such as object detection.

We consider the popular RGB-D Object Dataset (ROD) [36]. It contains 300 common household objects organized in 51 categories. The dataset was recorded using a Kinect style 3D camera that gathers 30 synchronized and aligned 640 × 480 RGB and depth images per second. Each object was placed on a turntable during the recording and it was captured during a whole rotation. For each object, there are three video sequences, corresponding to the camera mounted at a different height (so that the object is seen from different angles, *i.e.* approximately 30, 45, and 60 degrees relative to the horizon). The performance of a model on this task is evaluated as the accuracy on the test samples. We use the same evaluation protocol of [36] subsampling the dataset by taking every fifth frame, resulting in 41,877 RGB-D images for training and evaluation. They defined 10 pre-defined



(a) RGB-D Object dataset [36]

(b) NYU Depth V2 [37]

(c) LineMOD [38]

Fig. 3. Images taken from the datasets included in the benchmark. For each image, the top row shows the RGB version, the middle row the raw depth, and the bottom row the depth images colored with the Surface method [39].

training and test splits for cross-validation, and in each split, one random object instance from each class is left out from the training set and used for testing. This results in roughly 35,000 training images and 7,000 test images in each split. At test time, the classifier has to assign the correct label to a previously unseen object instance from each of the 51 classes. For training and test we use the first split among the ten available.

Task 2. 3D Pose Estimation. Pose estimation refers to the task of predicting the pose of an object with respect to the viewer's camera. This task is important in robotics, *e.g.* due to the information it provides for grasping and manipulating an object [5]. The pose estimation task can be decomposed in two sub-tasks: the localization of the object in the image and the estimation of the rotation matrix between the object and the camera. Even though many works attempt to perform both sub-tasks at the same time [40], [41], following [42], [43] in this paper we assume that the object is always centered in the input image, reducing the pose estimation task only to the latter sub-task, *i.e.* 3D pose estimation.

We consider the LineMOD dataset [38]. We choose this dataset since it is a standard benchmark in robotics and it also provides depth information, as opposed to other benchmarks [44]. It contains 18,000 RGB-D images of 15 different objects classes. We adopt the cropped version of the dataset that was proposed in [43]. In this version all the images are squared with size 64×64 pixels and they contain the objects centered in the scene. The ground truth of the pose is given in the form of 3×3 rotation matrix that maps the camera world coordinate into the camera coordinates.

Some methods tested on this dataset consider the use of both synthetic and real images [45], [46]. In this paper, in order to have a simpler experimental setup, we follow [47] and we consider a setting where only real images are used for training and testing the models. In particular, we use 80% of the images for training and 20% for testing.

For evaluating the performances of a model in the pose estimation task we consider both the prediction of the pose of the object and its semantic class. In particular, we consider a pose to be correctly estimated if the predicted object class is correct and the geodesic error of the predicted rotation is less then 20 degrees.

Task 3. Scene Classification. This task consists of assigning to an image a label that indicates the place where the picture was taken. Semantically localizing a robot provides relevant information on how the robot should interact with the environment.

We use the popular NYU-Depth V2 dataset [37], employed by many previous works [48], [49], [50]. The dataset contains 1,449 pairs of synchronized RGB and depth images, gathered from a wide range of commercial and residential buildings in three different US cities, comprising 464 different indoor scenes across 27 scene classes. Anyway, most of these scene classes are not well represented, thus, following [51], we reorganized the original 27 categories into 10 classes (*i.e.* the 9 most represented categories and the rest). The dataset is split in training and test data. Following [37], 795 images belong to training data and 654 to the test set. The performance of the model is evaluated as the classification accuracy on the test set.

Finally, we show sample images (RGB, depth and colorized depth [39]) for each dataset included in our benchmark in Figure 3.

B. The Evaluation Metrics

The RGB-D Triathlon, as we defined it in section III, requires a new evaluation metric, going beyond the mere comparison of standard accuracies. To this extent, in the following we define a metric which takes into account the two properties a good SMTL algorithm should have: i) single-task performances as close as possible to those of task-specific architectures ii) number of task-specific parameters as low as possible.

To define the metrics, we will consider two standard transfer learning methods: *fine-tuning* and *feature extractor*. *Fine-tuning* (FT) replicates the pretrained backbone architecture for every task and fine-tunes it independently: this produces a different network for each task. Since the full architecture is replicated and fine-tuned, $\rho_t = \rho_0$, where $\rho_0 = |\Theta_0|$ denotes the memory size required to store the parameters of the pretrained backbone architecture, excluding the final output layer. *Feature extractor* (FE) keeps a single backbone architecture but for each task it instantiates a new output layer. The weights of the network are frozen (*i.e.* they are not optimized) and only the task-specific output layers are learned. In this model, $\rho_t = 0$ since $\theta_t = \emptyset$ and only the classifier ψ_{Ω_t} is learned. In the text we will denote as α_t^{FE} and α_t^{FT} the performance on the task $\mathcal{T}_t$ of the feature extractor and fine-tuning baseline respectively.

In the following, we review standard and previous evaluation metrics (*i.e.* [16]), highlighting their drawbacks and proposing a new metric, the Locally Linear Score, overcoming them.

Average Accuracy (AvgA). Obviously, the most straightforward metric one can adopt is the average accuracy per task. Assuming the task-specific performance measures (*i.e.* accuracies) to be directly comparable (*i.e.* normalized on the same range) we can compute the performance score as:

$$S_{\text{AvgA}} = \frac{1}{N}\sum_{t=1}^{N} \alpha_t. \tag{1}$$

Obviously, this choice has the drawbacks of not considering i) the complexity of each task and ii) the amount of memory required to store the task-specific parameters.

Decathlon Score (DS). In order to jointly consider all the tasks and measuring the performance of SMTL methods, Rebuffi *et al.* proposed the Decathlon Score [16]. This metric favors methods that perform well on all tasks at the same time over methods which have mixed performances (*i.e.* high accuracy on some tasks and low on the others). The metric is based on the definition of a minimal accuracy per task $\alpha_t^{\min}$ under which the score obtained for the task $\mathcal{T}_t$ is zero. The minimal accuracy is set in [16] to the accuracy obtained by doubling the error on the task $\epsilon_t^{\text{FT}} = 1 - \alpha_t^{\text{FT}}$ of the task-specific fine-tuned model, namely:

$$\alpha_t^{\min} = \max(0, 1 - 2\cdot\epsilon_t^{\text{FT}}) = \max(0,\, 2\cdot\alpha_t^{\text{FT}} - 1) \tag{2}$$

The overall score is computed as follows:

$$S_{\text{DS}} = \sum_{t=1}^{N} \eta_t \, \max(0, \alpha_t - \alpha_t^{\min})^{\gamma_t}, \tag{3}$$

$$\eta_t = 1000 \cdot (1 - \alpha_i^{\min})^{-\gamma_i} \tag{4}$$

where $\gamma_t \geq 1$ is a coefficient which rewards accuracy improvements and it has been set to 2 in [16]. The parameter η_t normalizes the score of each task in order to constrain it in the range $[0, 1000]$. The advantage of this score is that it emphasizes the consistency of the performances across tasks, penalizing models which achieve very good performances but just on few tasks.

While this metric takes into account the complexity of each task by defining $\alpha_t^{\min}$, it does not contain any term reflecting the amount of memory required for storing the task-specific parameters. A possibility to include this in the score is by revisiting DS as follows (RevDS):

$$S_{\text{RevDS}} = \sum_{t=1}^{N} \eta_t \cdot \lambda^{-\frac{\rho_t}{\rho_0}} \cdot \, \max(0, \alpha_t - \alpha_t^{\min})^{\gamma_t} \tag{5}$$

where $\lambda > 1$ is a coefficient weighting the impact of the memory size required to store the task-specific parameters ρ_t.

TABLE I

A COMPARISON OF THE PROPERTIES OF SMTL METRICS

	considers complexity of tasks	penalizes additional parameters	compares to feature extractor	compares to fine tuning
Avg.A	✗	✗	✗	✗
DS	✓	✗	✗	✓
RevDS	✓	✓	✗	✓
LL	✓	✓	✓	✓

The higher is λ, the higher is the impact of the parameters. We set $\lambda = 10$ in our experiments.

This metric preserves the positive aspects of the DS while considering also the memory consumption. However, it requires to set several parameters (*e.g.* λ, γ) and it does not take into account the actual benefits that the task-specific parameters may bring with respect to a baseline where just the final output layer is learned.

Locally Linear Score (LL). In this work we propose a novel metric which considers both the memory requirement and the accuracy, while getting rid of the coefficients required by DS and RevDS. It is computed as follows:

$$S_{\text{LL}} = \frac{1000}{N}\sum_{t=1}^{N} R_t \cdot A_t \tag{6}$$

where:

$$R_t = \max(0, 1 - \frac{\rho_t}{\rho_0}) \quad \text{and} \quad A_t = \frac{\max(0, \alpha_t - \alpha_t^{\text{FE}})}{\alpha_t^{\text{FT}} - \alpha_t^{\text{FE}}} \tag{7}$$

This metric contains two terms. The first element, R_t, penalizes the increase of the memory size required to store the task-specific parameters. The second element, A_t, normalizes the single task performance by considering i) the gain obtained by introducing task-specific parameters α_t with respect to not introducing them (*i.e.* α_t^{FE}) and ii) the ratio between this gain and the one obtained by fine-tuning the full architecture (*i.e.* α_t^{FT}).

From Eq. (6) and (7), if an SMTL model does not require any task-specific parameter (*i.e.* $\rho_t = 0$), $R_t = 1$ and only the second term A_t will be considered for computing the score. In the case $\rho_t > 0$, A_t will be linearly scaled by the ratio ρ_t/ρ_0. $S_{\text{LL}} = 0$ if the size of the task-specific parameters is equal (or greater) to the size of the shared ones, as in the full fine-tuning case where $\rho_t = \rho_0$.

For the accuracy component A_t the rationale is similar. Since we require the single-task performance α_t of an SMTL model to be at least better than the performance obtained by not adding any parameter (*i.e.* the feature extractor baseline, α_t^{FE}), we set the score to zero if $\alpha_t \leq \alpha_t^{\text{FE}}$. At the same time, we use the difference among α_t^{FE} and α_t^{FT} as a normalization factor to check how well a model is able to fill the performance gap existing between the task-agnostic baseline (*i.e.* α_t^{FE}) and the full task-specific counterpart (*i.e.* α_t^{FT}). The properties of this new metric and a comparison with the metrics previously defined is reported in Table I.

Finally, we highlight that while in the current version of the challenge $N = 3$, the evaluation metrics we have defined are applicable to any number of tasks and to any backbone architecture, given the corresponding α_t^{FE} and α_t^{FT}. This will allow to easily extend the benchmark in the future. Moreover, we *do not* specify any fixed order for the tasks: this choice allows researchers to experiment with different sequences, with the possibility of exploring the relations among the tasks.

IV. EXPERIMENTS

In this section, we test state of the art SMTL algorithms [16], [17], [19], [20] on our RGB-D Triathlon. We first present the baseline methods and describe the implementation details, then we discuss the results of our evaluation.

A. SMTL methods

Together with the FT and FE baselines, introduced in Section III, we evaluated four state of the art SMTL methods in our experiments: *series* [16] and *parallel residual adapters* [17], *Piggyback* [19], and *binarized affine transformation (BAT)* [20].

In the *series residual adapter* (RS) [16] task-specific parameters correspond to residual adapter modules added in series after the convolutional layer of each residual unit. In *parallel residual adapter* (RP) [17] the task-specific modules are added in parallel to the convolutional layers of each residual block and not serially. In both cases, the residual adapter is implemented as a convolutional layer with kernel size 1×1 (plus a batch-normalization layer in [16]).

Piggyback (PB) [19] makes use of task-specific binary masks that are added to each convolutional layer of a backbone network. These binary masks multiply point-wise the original network weights, de facto producing new convolutional filters for the task of interest.

Similarly to [19], *BAT* [20] uses task-specific binary masks that are paired with each convolutional layer of a backbone network. Differently from [19] the masks are not applied through a point-wise multiplication but are used to perform an affine transformation of the convolutional filters, creating a new set of task-specific weights.

B. Implementation

All the methods we evaluated require a pretrained architecture. We use the *ResNet-18* [52] pretrained on ImageNet [25]. This network has been chosen as it guarantees a good trade-off between accuracy and speed.

All the methods have been tested on three settings, corresponding to different inputs: RGB only, depth only and RGB-D. To handle the depth images we took inspiration from [39], processing the depth images using the Surface++ approach. For the single modality case, we simply consider as backbone network the ResNet-18. For the RGB-D setting we fuse by concatenation the features of two ResNet-18, one processing only RGB images and the other only depth maps. The fused features are passed through a fully connected layer that we use as the output layer, similarly to [53].

To fairly compare all the methods, the same set of hyperparameters is used for all the methods in each task. The networks are trained using Stochastic Gradient Descent (SGD) [54] with momentum except for Piggyback and BAT where SGD is used for the classification layer and Adam [55] is used for the rest of the network, as suggested in [19], [20]. The networks are trained for 30 epochs in each setting, with a batch-size equal to 32 and weight decay $5 \cdot 10^{-5}$. The learning rate of the SGD optimizer is set to 0.005 for every task and method, while the Adam learning rate is adapted for each task, using $1 \cdot 10^{-5}$ for ROD, $5 \cdot 10^{-5}$ for LineMOD and $1 \cdot 10^{-4}$ for NYU. The learning rates are decayed by a factor of 10 after 20 epochs.

TABLE II

COMPARISON BETWEEN RESNET-18 [52] FINE-TUNED ON THE TASK AND STATE OF THE ART METHODS IN THE RGB-D SETTING.

		ROD	**LineMOD**	**NYU**
ROD	CNN+Fisher [56]	93.8	-	-
	(DE)^{2}CO [57]	93.6	-	-
	FusionNet enhanced [58]	93.5	-	-
LineMOD	Zakharov et al. [45]	-	93.2[2]	-
	Wohlhart et al. [43]	-	96.2[2]	-
NYU	Du et al. [50]	-	-	67.5
	Song et al. [48]	-	-	66.7
Ours	ResNet-18	93.9	96.9	68.4

To implement all the baselines we used the PyTorch framework. The code of the networks and the training procedure are publicly available[1].

C. Results

Comparison with state of the art. Before analyzing the performance of SMTL methods, we conduct a preliminary experiment to evaluate the performance of ResNet-18 with FT on the considered tasks. In Table II we report our results on the RGB-D setting.

For ROD [36] we only report the three best performing methods in the literature [56], [57], [58]. In [58] an ensemble of deep models is used, extracting depth information through different colorization techniques. In [57] a deep architectures is considered to learn how to map depth data to three channel images. In [56] depth data are encoded with Fisher vectors without adopting colorization approaches.

For the LineMOD dataset [38] we consider two state of the art approaches [45], [43]. In [43] a convolutional network is used to map the image space to a descriptor space where the pose and object classes are predicted through a nearest neighbour classifier. The method in [45] builds upon [43], introducing a triplet loss function with a dynamic margin. These works employ a slightly different settings than ours since they use synthetic images. We report as accuracy measure a metric that consider correct a test image only if the angular error is below 20 degree.

For the NYU Depth V2 dataset [37] we report two state of the art methods [50], [48]. Du *et al.* [50] introduce a two-step training strategy that translates RGB images to depth ones. In [48] Song *et al.* propose to learn from scratch the depth features with the help of depth patches.

[1] `https://github.com/fcdl94/RobotChallenge`

[2] The metric does not consider classification

TABLE III

RESULTS IN TERM OF ACCURACY FOR EACH TASK IN THE RGB-D TRIATHLON. THE BEST METHOD IS BOLD.

Setting	Method	ROD	LineMOD	NYU	AvgA$_{\%}$
RGB	FT	90.8	**96.7**	67.5	85.0
	FE	87.6	13.9	57.9	53.1
	PB [19]	89.4	95.9	**68.4**	84.6
	BAT [20]	**92.9**	95.0	68.1	**85.3**
	RS [16]	90.9	89.7	67.1	82.6
	RP [17]	91.3	89.9	68.3	83.2
Depth	FT	83.3	**87.0**	59.2	**76.5**
	FE	78.3	7.6	43.1	43.0
	PB [19]	83.5	83.2	56.6	74.4
	BAT [20]	**83.7**	**87.0**	57.3	76.0
	RS [16]	83.6	79.8	**61.2**	74.8
	RP [17]	**83.7**	76.9	57.3	72.6
RGB-D	FT	**93.9**	**96.9**	68.4	**86.4**
	FE	91.6	14.8	60.2	55.5
	PB [19]	89.6	96.6	66.7	84.2
	BAT [20]	93.5	95.2	68.1	85.6
	RS [16]	93.7	93.8	**70.9**	86.1
	RP [17]	93.8	92.1	67.5	84.5

TABLE IV

BASELINES SCORE FOR THE RGB-D TRIATHLON. THE BEST METHOD IS BOLD, THE SECOND BEST IS UNDERLINED.

Setting	Method	Par	AvgA$_{\%}$	DS	RevDS	LL
RGB	FT	1.00	85.0	**750**	8	0
	FE	0.00	53.1	229	229	0
	PB [19]	0.03	84.6	577	463	853
	BAT [20]	0.03	**85.3**	693	**556**	**1196**
	RS [16]	0.16	82.6	500	169	818
	RP [17]	0.13	83.2	542	228	922
Depth	FT	1.00	**76.5**	**750**	8	0
	FE	0.00	43.0	213	213	0
	PB [19]	0.03	74.4	598	480	909
	BAT [20]	0.03	76.0	741	**595**	**957**
	RS [16]	0.16	74.8	582	197	865
	RP [17]	0.13	72.6	501	211	824
RGB-D	FT	1.00	**86.4**	**750**	8	0
	FE	0.00	55.5	237	237	0
	PB [19]	0.03	84.2	451	362	560
	BAT [20]	0.03	85.6	514	**413**	890
	RS [16]	0.16	86.1	524	177	**891**
	RP [17]	0.13	84.5	481	203	818

The results in Table II clearly show a fine-tuned ResNet-18 is a competitive baseline by itself on each task, confirming the appropriateness of our choice.

Results on the RGB-D Triathlon. In this subsection, we compare SMTL methods on our RGB-D Triathlon. We first analyze the performance of all SMTL techniques and compare it with FT and FE. Table III reports the accuracy for each method, setting and task. AvgA$_{\%}$ denotes the average accuracy in percentage. Looking at the results for different modalities, in two out of three settings FT guarantees the best accuracy. This is somehow expected as this corresponds to use a separate network for each task. Moreover, FE corresponds to the worst performance, as considering the backbone network purely as a feature extractor is a sub-optimal strategy. Interestingly, the SMTL methods are very competitive with FT, despite they use few task-specific parameters (see also Table IV). In particular, in the RGB setting BAT outperforms FT. We ascribe this behavior to the regularization effect introduced by the use of binary masks. Comparing different SMTL techniques, in the Depth only and RGB only settings BAT outperforms RS, RP and PB.

Table IV provides a better comparison of all the methods considering different evaluation metrics, settings and tasks. In the table, the *Par* column reports the average per task of the ratio between the memory required by task-specific parameters ρ_t and by parameters of the backbone network ρ_0, in formulas $\frac{1}{N}\sum_{t=1}^{N}\rho_t/\rho_0$. For instance, FT corresponds to the value 1 since it implies learning all the weights of a ResNet-18 for all the three tasks and FE corresponds to 0 as only the classifier is learned.

It is interesting to discuss the different metrics. We focus on the comparison between BAT and RS in the RGB-D setting, which are the best performing methods among SMTL techniques. In term of the average accuracy AvgA$_{\%}$, RS considerably outperforms BAT. However, looking at Table III we can see that RS is not always the best performer. Even if it achieves a very high accuracy in NYU, BAT outperforms RS in the LineMOD dataset and they are comparable in ROD. This result underlines the need for a metric that jointly consider all tasks. Indeed, the two methods have similar performance considering the Decathlon Score metric (DS) because DS ranks higher methods that are accurate in all the tasks. However, DS does not penalize methods which add several parameters. Oppositely, the Revised Decathlon Score (RevDS) strongly penalizes the addition of parameters. Thus, under RevD, we note a remarkable change in the ranking and BAT significantly outperforms RS. Actually, RS obtain a worse score than all the SMTL methods. Finally, considering the proposed Locally Linear score (LL), RS and BAT are comparable, as this metric is the only one which reflects the best trade-off between accuracy and use of extra parameters.

V. CONCLUSIONS

We presented a novel benchmark for robot visual systems, namely the RGB-D Triathlon. This dataset allows to compare model in the sequential multi-task learning scenario where, starting from a pretrained deep architecture, the goal is to learn models for diverse visual tasks while i) obtaining performances as close as possible to fine-tuned task-specific architectures; ii) keeping as low as possible the number of additional parameters required per-task; and iii) not changing the performances on old tasks. We introduce a novel metric for the SMTL problem which takes into accounts both the performances and the number of parameters. We then evaluated state of the art SMTL methods on the new benchmark and released our source code for promoting further research on the topic. We hope that the RGB-D Triathlon will help to stimulate research in robot perception. While in the RGB-D Triathlon we considered a specific setting, our definition of tasks, metrics and baselines is modular and can be easily extended in other applications.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS-12*.

[2] G. Costante, M. Mancini, P. Valigi, and T. A. Ciarfuglia, "Exploring representation learning with cnns for frame-to-frame ego-motion estimation," *RA-L-16*, vol. 1, no. 1.

[3] O. H. Jafari, O. Groth, A. Kirillov, M. Y. Yang, and C. Rother, "Analyzing modular cnn architectures for joint depth prediction and semantic segmentation," in *ICRA-17*.

[4] M. Mancini, G. Costante, P. Valigi, T. A. Ciarfuglia, J. Delmerico, and D. Scaramuzza, "Toward domain independence for learning-based monocular depth estimation," *RA-L-17*, vol. 2, no. 3.

[5] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," in *IROS-16*.

[6] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *IJRR-18*, vol. 37, no. 4-5.

[7] M. Schwarz, A. Milan, A. S. Periyasamy, and S. Behnke, "Rgb-d object detection and semantic segmentation for autonomous manipulation in clutter," *IJRR-18*, vol. 37, no. 4-5.

[8] G. L. Oliveira, C. Bollen, W. Burgard, and T. Brox, "Efficient and robust deep networks for semantic segmentation," *IJRR-18*, vol. 37.

[9] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, 1997.

[10] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," in *IVS-18*.

[11] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration," in *ICRA-18*.

[12] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, *et al.*, "Never-ending learning," *Communications of the ACM*, vol. 61, no. 5, 2018.

[13] S. Thrun, "Lifelong learning algorithms," in *Learning to learn*. Springer, 1998.

[14] M. McCloskey and N. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," *Psychology of Learning and Motivation - Advances in Research and Theory*, vol. 24, no. C, 1989.

[15] I. J. Goodfellow, M. Mirza, D. Xiao, A. Courville, and Y. Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," *arXiv:1312.6211*, 2013.

[16] S.-A. Rebuffi, H. Bilen, and A. Vedaldi, "Learning multiple visual domains with residual adapters," in *NIPS-17*.

[17] ——, "Efficient parametrization of multi-domain deep neural networks," in *CVPR-18*.

[18] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *CVPR-18*.

[19] A. Mallya, D. Davis, and S. Lazebnik, "Piggyback: adapting a single network to multiple tasks by learning to mask weights," in *ECCV-18*.

[20] M. Mancini, E. Ricci, B. Caputo, and S. Rota Buló, "Adding new tasks to a single network with weight transformations using binary masks," in *ECCV-WS-18*.

[21] M. J. Milford and G. F. Wyeth, "Seqslam: Visual route-based navigation for sunny summer days and stormy winter nights," in *ICRA-12*.

[22] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *IROS-12*.

[23] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *T-RO-15*, vol. 31.

[24] N. Sünderhauf, T. Pham, Y. Latif, M. Milford, and I. Reid, "Meaningful maps with object-oriented semantic mapping," in *IROS-17*.

[25] 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-15*, vol. 115, no. 3.

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV-14*.

[27] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: A 10 million image database for scene recognition," *T-PAMI-18*, vol. 40, no. 6.

[28] H. Bilen and A. Vedaldi, "Universal representations: The missing link between faces, text, planktons, and cat breeds," *arXiv:1701.07275*, 2017.

[29] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," in *ICRA-17*.

[30] K. Fang, Y. Bai, S. Hinterstoisser, S. Savarese, and M. Kalakrishnan, "Multi-task domain adaptation for deep learning of instance grasping from simulation," in *ICRA-18*.

[31] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, 1999.

[32] V. Lomonaco and D. Maltoni, "Core50: a new dataset and benchmark for continuous object recognition," in *CoRL-17*.

[33] S. Valipour, C. Perez, and M. Jagersand, "Incremental learning for robot perception through hri," in *IROS-17*.

[34] R. Camoriano, G. Pasquale, C. Ciliberto, L. Natale, L. Rosasco, and G. Metta, "Incremental robot learning of new objects with fixed update time," in *ICRA-17*.

[35] M. O. Turkoglu, F. B. Ter Haar, and N. van der Stap, "Incremental learning-based adaptive object recognition for mobile robots," in *IROS-18*.

[36] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *ICRA-11*.

[37] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV-12*.

[38] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *ACCV-12*.

[39] A. Aakerberg, K. Nasrollahi, and T. B. Moeslund, "Depth value preprocessing for accurate transfer learning based rgb-d object recognition," in *IJCAI-17*.

[40] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *ICCV-17*, vol. 1, no. 4.

[41] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv:1711.00199*, 2017.

[42] S. Mahendran, H. Ali, and R. Vidal, "3d pose regression using convolutional neural networks," in *ICCV-17*, vol. 1, no. 2.

[43] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *CVPR-15*.

[44] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond pascal: A benchmark for 3d object detection in the wild," in *WCACV-14*.

[45] S. Zakharov, W. Kehl, B. Planche, A. Hutter, and S. Ilic, "3d object instance recognition and pose estimation using triplet loss with dynamic margin," in *IROS-17*.

[46] P. Wohlhart and V. Lepetit, "Learning descriptors for object recognition and 3d pose estimation," in *CVPR-15*.

[47] L. Porzi, A. Penate-Sanchez, E. Ricci, and F. Moreno-Noguer, "Depth-aware convolutional neural networks for accurate 3d pose estimation in rgb-d images," in *IROS-17*.

[48] X. Song, L. Herranz, and S. Jiang, "Depth cnns for rgb-d scene recognition: Learning from scratch better than transferring from rgb-cnns." in *AAAI-17*.

[49] X. Song, S. Jiang, and L. Herranz, "Combining models from multiple sources for rgb-d scene recognition," *IJCAI-17*.

[50] D. Du, X. Xu, T. Ren, and G. Wu, "Depth images could tell us more: Enhancing depth discriminability for rgb-d scene recognition," in *ICME-18*.

[51] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from rgb-d images," in *CVPR-13*.

[52] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR-16*.

[53] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust rgb-d object recognition," in *IROS-15*.

[54] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *COMPSTAT-2010*. Springer.

[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[56] W. Li, Z. Cao, Y. Xiao, and Z. Fang, "Hybrid rgb-d object recognition using convolutional neural network and fisher vector," in *CAC-15*.

[57] F. M. Carlucci, P. Russo, and B. Caputo, "(de^2)co: Deep depth colorization," *RAL-18*, vol. 3, no. 3.

[58] A. Aakerberg, K. Nasrollahi, and T. Heder, "Improving a deep learning based rgb-d object recognition model by ensemble learning," in *IPTA-17*.