Uncertainty-Aware Self-Supervised Learning of Spatial Perception Tasks

Mirko Nava, Antonio Paolillo, Jérôme Guzzi, Luca Maria Gambardella, and Alessandro Giusti

***Abstract*—We propose a general self-supervised learning approach for spatial perception tasks, such as estimating the pose of an object relative to the robot, from onboard sensor readings. The model is learned from training episodes, by relying on: A continuous state estimate, possibly inaccurate and affected by odometry drift; and a detector, that sporadically provides supervision about the target pose. We demonstrate the general approach in three different concrete scenarios: a simulated robot arm that visually estimates the pose of an object of interest; a small differential drive robot using 7 infrared sensors to localize a nearby wall; an omnidirectional mobile robot that localizes itself in an environment from camera images. Quantitative results show that the approach works well in all three scenarios, and that explicitly accounting for uncertainty yields statistically significant performance improvements. Videos, datasets, and code to reproduce our results are available at: https://github.com/idsia-robotics/uncertainty-aware-ssl-spatial-perception.**

***Index Terms*—Deep learning for visual perception, deep learning methods.**

I. Introduction

MANY robot perception tasks consist of interpreting sensor readings to extract high-level spatial information [1], such as the pose of an object of interest (OOI) with respect to the robot, or the pose of the robot itself in the environment. When sensors produce noisy, high-dimensional data that is difficult to interpret (e.g. cameras or lidars), a common solution is to rely on supervised learning [2]. In many real-world scenarios, collecting the necessary training sets is a fundamental problem; self-supervised learnings (SSLs) in robotics aims at equipping robots with the ability to acquire their own training data, e.g. by using additional sensors as a source of supervision, without any external assistance. In some cases, this allows robots to acquire training data directly in the deployment environment.

In this context, a state estimator, such as a robot's odometry, can be a rich source of information. Odometry allows a robot to estimate its own motion in the environment according to its kinematics (e.g. by integrating over time the motion of its wheels as measured by wheel encoders), often with some uncertainty and accumulating errors. Consider for example a robot capable of odometry, equipped with a camera and a collision detector [3]; after bumping into an object, the robot could reconsider the camera observations from the timesteps preceding the collision; assuming a static obstacle, the camera image acquired when the robot was (according to its odometry) 1m behind the place of collision, would depict an obstacle at a distance of 1m. This piece of information was acquired by the robot without any explicit external supervision, and can be used for training machine learning models that map acquired images to the spatial position of obstacles. The role of odometry is to leverage sparse information from a simple detector – which provides relevant information only for a few timesteps in a training sequence – to generate an informative labeled training set.

In this paper we generalize and extend this basic idea: we consider a generic robot that has a spatial perception task (e.g., estimating the pose of an OOI in the environment), is capable of state estimation, possibly affected by accumulating uncertainty due to errors (e.g. odometry), and is equipped with one or more sensors, whose outputs we want to use to estimate the target pose. Furthermore, the robot is equipped with a detector that, for at least a small fraction of timesteps, produces ground-truth information about the target pose (possibly uncertain). Given training sequences, we want to learn a stateless model that, given the sensor readings, estimates the target pose.

After reviewing the related work in Sec. II, we illustrate our main contribution in Sec. III: a formalization of this problem and a general solution based on deep learning, that: i) learns from sporadic supervision given by a detector and a possibly uncertain state estimator; ii) can explicitly account for uncertainty in the state estimates and in the supervision using a Monte Carlo approach; iii) integrates a recently proposed state-consistency loss [4] to further improve results, even with the hurdle of uncertainty. In Sec. IV we investigate the generality of our contribution by instantiating it in three different applications (more details in Table I):

- Estimating the relative 3D pose of an OOI from a camera mounted on a robotic arm manipulator.
- Estimating the heading of a differential drive robot in the vicinity of a straight wall, using data from 7 sensors that measure the amount of infrared light reflected from the environment.

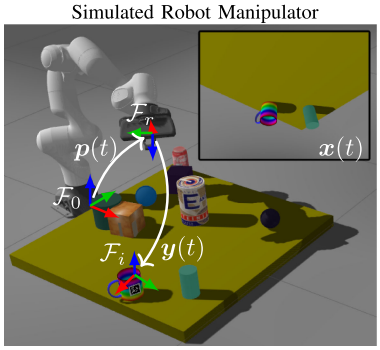
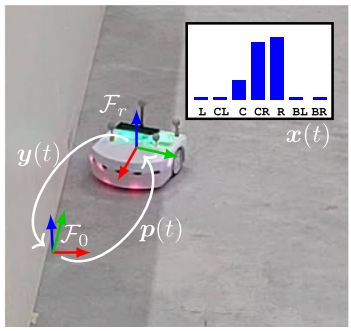
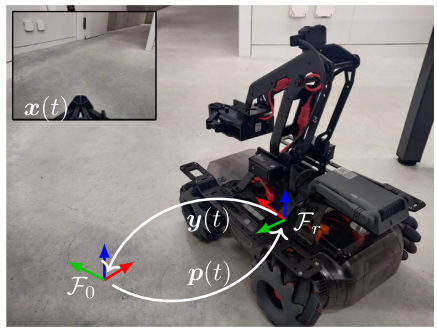
Manuscript received February 24, 2021; accepted June 22, 2021. Date of publication July 7, 2021; date of current version July 21, 2021. This letter was recommended for publication by Associate Editor J. McDonald and Editor C. Cadena Lerma upon evaluation of the reviewers' comments. This work was supported by the Swiss National Science Foundation (SNSF) through the NCCR Robotics. *(Corresponding author: Mirko Nava)*

The authors are with the Dalle Molle Institute for Artificial Intelligence (IDSIA), USI-SUPSI, 6928 Lugano, Switzerland (e-mail: mirko@idsia.ch; antonio.paolillo@supsi.ch; jerome@idsia.ch; luca@idsia.ch; alessandrog@idsia.ch).

This letter has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/LRA.2021.3095269

TABLE I
PLATFORMS AND SENSORY EQUIPMENT USED IN THE PERCEPTION TASKS

Robot	Simulated Robot Manipulator	Real Differential Drive Robot	Real Mecanum Robot
			
Sensor	RGB Camera	Infrared Sensors	RGB Camera
Target	Object of Interest 3D Pose	Robot Heading	Docking Station 2D Pose
Detector	Fiducial Marker	Initial Robot Pose	Initial Robot Pose
State Estimator	Forward Kinematics	Odometry (inaccurate)	Odometry (inaccurate)
Section	IV-A, V-A	IV-B, V-B	IV-C, V-C

- Estimating the 2D pose of a docking station, using images from a camera mounted on a mecanum robot.

Sec. V experimentally evaluates the approach on the three applications and quantifies the improvements due to explicitly modeling uncertainty and enforcing state-consistency; while conclusions are drawn in Sec. VI.

II. RELATED WORK

A. Self-Supervised Learning

SSL refers to approaches that utilize a subset of the available features, or an elaboration of those, as supervisory information [1]. In robotics, the term is also used to denote an autonomous, unattended robot that collects data, from which both input features and supervisory labels are extracted. While many SSL approaches focus on specific systems to extract the target variable [3], [5]–[12], no explicit notion of a general detector is present. In navigation, supervisory information is extracted from the knowledge that at time $t = 0$ a vehicle is on the road [5], from letting a drone roam the environment until a crash happens [3], or by continuously measuring the distance from the surroundings [6]. Similar to our approach, Lieb *et al.* [5] and Gandhi *et al.* [3] reconstruct the ground-truth from a small subset of data, respectively at the beginning and at the end of each episode; while Kouris *et al.* [6] utilize a dense ground-truth estimate generated from three laser sensors. In grasping, supervision is derived from measuring the force perceived on the end effector before and after a grasp attempt [7] or by iteratively refining the 3D pose prediction of known objects [9]. Pinto *et al.* [7] predict the probability of grasping an object from 18 possible angles using images. Ground-truth information is generated once per each grasping attempt; while Deng *et al.* [9] utilize a pre-trained convolutional neural network (CNN) to serve as a continuous source of ground-truth information. The model learns online, and by grasping and moving objects produces more data, repeating the cycle. In traversability estimation, a dense detector is derived from future or past sensors readings [10]–[12]. Nava *et al.* [10] learn a robot-centric obstacle map composed of traversable and obstacle cells. Labels are generated for each timestep considering proximity sensors' readings collected over a sliding time window. Both sensors' readings and odometry are assumed to be ideal, while in our work we deal with uncertain (noisy) information. Wellhausen *et al.* [11] predict terrain class and a traversability score by relating the future torque measured on the legs of the ANYmal quadruped, to image space footholds perceived currently. Similarly, Zürn *et al.* [12] learn to classify the terrain from images, using as supervision the Fourier transform of sound captured near the wheels of a ground robot. In the visual odometry field, self-supervised approaches extract ground-truth information directly from ego motion, learning the relative transformation between two images collected by a camera [13], [14], or from inverse warping images collected at time $t + 1$ and $t - 1$ with respect to the current image [15]. Iyer *et al.* [14] propose a geometric consistency term, aimed at improving the performance of the visual odometry module: they enforce the concatenation of relative motions from time t to $t + n$ to be close to the motion of $t + n$ with respect to t. In contrast, the state-consistency loss [4] adopted in this work, is not limited to ego-motion information and enforces consistency on a general model's prediction (i.e., can be applied to pose estimation of the robot itself or other objects in the environment). Additionally, [14] require the use of ad-hoc SE(3) Layers in the neural network (NN), which represent the pose as a 3D rigid body transformation matrix, while the proposed work does not require any specific NN architecture, nor a particular pose representation and associated distance function.

B. Noisy Data in Regression

In real world scenarios, especially when data is collected directly by robots, noise present in the labels can severely degrade the performance of learned models [2]. Most approaches that do regression from noisy data focus on estimating the uncertainty associated with predictions. These approaches fall under

the category of Bayesian learning: some apply an uncertainty estimation model on an existing regressor [16], while others design ad-hoc architectures, casting the problem as learning the distribution of the model weights [17]–[19].

Loquercio *et al.* [16] estimate the uncertainty associated with model predictions by fitting a Bayesian belief network. Dropout is used during multiple forward passes of the network to approximate the uncertainty of the prediction in a Monte Carlo sampling fashion. Similarly, Gal *et al.* [18] propose to approximate the true uncertainty by utilizing dropout Monte Carlo samples. In contrast to [16], they require to alter the network architecture by placing dropout layers after each non-linearity. By keeping the dropout functionality active during inference, they compute the uncertainty as the variance of the produced samples. Blundell *et al.* [17] propose to model NN weights with a zero-centered Gaussian distribution. Learning is then casted as a variational inference problem, where the true target distribution is approximated by the learned model conditioned on input data. By noting that commonly used weight distributions are symmetric and independent, Yeming *et al.* [19] improve the ideas of [17] by decomposing the forward pass into the multiplication of input and weight means by independently sampled sign matrices [19, eq. (4)], resulting in faster computation and less variance in the gradients.

In contrast, our approach tackles the different problem of learning from noisy data, without delving into the estimation of the uncertainty associated with predictions.

C. Related Case Studies

We demonstrate the generality of our approach by solving common tasks in the robotics field: OOI pose estimation with a robotic arm, and localization of mobile ground robots.

Zeng *et al.* [20] use SSL to collect RGB-D images of single objects from different points of view, then segmented by a background removal algorithm. Labels are then extracted by fitting the 3D model of the object in the corresponding image using an iterative closest point (ICP) algorithm. Deng *et al.* [9] iteratively refine the performance of a NN tasked to predict the pose of known objects from RGB data with online self-supervision: initially the network is trained on simulated data; it is then deployed in a real-world scenario where a camera-equipped robotic arm moves by pushing or grasping the objects around, generating new data used to improve the pose estimation. Both approaches rely on 3D models of the objects, used during the learning process to refine the prediction, overlaying the 3D object on top of the image and measuring the difference. Instead, our approach does not require prior knowledge of the object and relies solely on a detector capable of recognizing the object, or a fiducial marker attached to it.

Ratz *et al.* [21] localize the robot within an environment from LiDAR scans and camera frames. A NN fuses together the two sensors and produces a multi-modal descriptor. Localization is done in a one-shot fashion by matching the descriptor with a pre-computed database. In our case studies, the approach utilizes only camera images and directly learns to localize, internally learning a discriminative feature space for the environment where data are collected.

III. Model

A. Definitions

We aim to train a model that, given readings $\boldsymbol{x}(t)$ collected by onboard sensors, predicts a target variable $\boldsymbol{y}(t)$, which is a pose in SE(3) of the frame $\mathcal{F}_i$ of an OOI, relative to the moving robot frame $\mathcal{F}_r$. The sensor readings $\boldsymbol{x}(t)$ do not need to have an explicit geometric interpretation, might be high-dimensional (e.g. an uncalibrated image), and could potentially represent the concatenated outputs from multiple heterogeneous sensors.

To train the model, we use data collected in one or more training episodes. During each episode, the OOI is static (i.e. $\boldsymbol{y}$ does not change when expressed in a fixed reference frame) while the robot moves in the environment. Let $\mathcal{T}$ be the set of all timesteps in a given training episode.

We assume that a black-box *detector* provides temporally sparse estimates of $\boldsymbol{y}(t)$:

$$\boldsymbol{d}(t) = \begin{cases} \tilde{\boldsymbol{y}}(t) & \text{if } t \in \mathcal{T}_d, \\ \text{undefined} & \text{otherwise} \end{cases} \tag{1}$$

where $\mathcal{T}_d \subseteq \mathcal{T}$ denotes the set of timesteps in which the detector module provides an output, and the tilde over $\boldsymbol{y}$ denotes the fact that this is a potentially inaccurate estimate of the true value of the target variable.

Finally, we assume that an odometry module outputs for every $t \in \mathcal{T}$ a (potentially inaccurate) estimate $\tilde{\boldsymbol{p}}(t)$ of the robot pose $\boldsymbol{p}(t)$ in a fixed inertial frame $\mathcal{F}_0$. Given two timesteps $t, u \in \mathcal{T}$, we denote with $\tilde{\boldsymbol{p}}(t, u)$ the odometry's estimate of the pose of the robot at u with respect to its pose at t. In particular, $\tilde{\boldsymbol{p}}(t, u) = \ominus \boldsymbol{p}(t) \oplus \boldsymbol{p}(u)$, where $\oplus$ denotes the pose composition operator, and the unary operator $\ominus$ denotes pose inversion [22].

For each training episode, we collect the set of samples

$$\{\boldsymbol{x}(t), \boldsymbol{p}(t), \boldsymbol{d}(t) | t \in \mathcal{T}\}; \tag{2}$$

and use data from all training episodes to learn a mapping from $\boldsymbol{x}$ to $\boldsymbol{y}$. The mapping is implemented by a NN model $\boldsymbol{m}(\boldsymbol{x}|\boldsymbol{\theta})$ parametrized by $\boldsymbol{\theta}$. Training is performed by minimizing a loss function

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_{\text{sc}} \mathcal{L}_{\text{sc}}, \tag{3}$$

composed by a task loss $\mathcal{L}_{\text{task}}$ and a state-consistency loss $\mathcal{L}_{\text{sc}}$; the latter is scaled with a factor λ_{sc}.

We first describe these two terms in case the detector and odometry return point-wise estimates; then, we extend the discussion to the case in which their uncertainty can be modeled with a probability distribution, see Fig. 1 and Fig. 2.

B. Task Loss

Consider two timesteps t, u in the same episode, such that $t \notin \mathcal{T}_d$ and $u \in \mathcal{T}_d$. The task loss enforces that the model, when fed with $\boldsymbol{x}(t)$ returns a pose that is consistent with $\boldsymbol{d}(u)$, accounting for the pose transform measured between t and u by the robot odometry, i.e. $\tilde{\boldsymbol{p}}(t, u)$.

More specifically, $\boldsymbol{d}(u)$ is an estimate of the target variable with respect to the robot frame at time u; it follows that $\tilde{\boldsymbol{y}}(t) = \tilde{\boldsymbol{p}}(t, u) \oplus \boldsymbol{d}(u)$ is an estimate of the target variable at t. Thus,

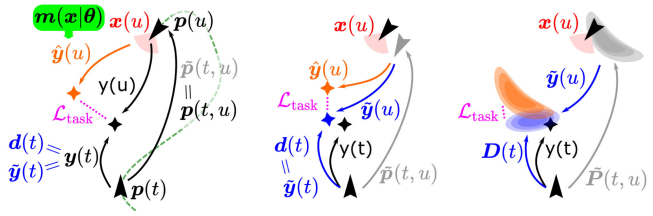


Fig. 1. Illustration of the task loss in case the detector and odometry are ideal (left), inaccurate (center), or with a known uncertainty model (right). Black color denotes the true poses for the robot (arrowhead) and OOI (diamond); blue color denotes the OOI pose as returned by the detector; orange denotes model predictions, obtained from data (red) sensed at time u. Gray denotes robot odometry. See text for details.

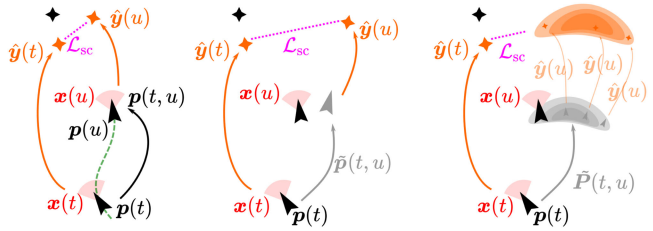


Fig. 2. Illustration of the state-consistency loss in case the odometry is ideal (left), inaccurate (center), or with a known uncertainty model (right); black color denotes the true poses for the robot (arrowhead) and OOI (diamond). Orange denotes the outputs of the model at times t and u, which depends on sensed data (red). The state consistency loss (violet) forces the model to output consistent estimates, accounting also for odometry (gray) and its uncertainty, if known.

the task loss is defined as

$$\mathcal{L}_{\text{task}} = \sum_{t\in\mathcal{T}_d, u\in\mathcal{T}} \Delta_{\text{SE3}}(\tilde{\boldsymbol{p}}(t,u) \oplus \boldsymbol{d}(u), \boldsymbol{m}(\boldsymbol{x}(t)|\boldsymbol{\theta})), \tag{4}$$

where the function $\Delta_{\text{SE3}}(\cdot,\cdot)$ is a measure of the distance between two poses in SE(3) and defined as

$$\Delta_{\text{SE3}}(\boldsymbol{p}_a, \boldsymbol{p}_b) := \lambda_{\text{o}} \left\|\boldsymbol{o}_a - \boldsymbol{o}_b\right\| + \frac{1}{\pi}\, \Delta_{\text{quat}}\left(\boldsymbol{q}_a, \boldsymbol{q}_b\right), \tag{5}$$

where $\boldsymbol{p}_a$ and $\boldsymbol{p}_b$ are two generic poses, composed of the position components $\boldsymbol{o}_a, \boldsymbol{o}_b \in \mathbb{R}^3$, and the rotation components represented as quaternions $\boldsymbol{q}_a, \boldsymbol{q}_b \in \mathbb{H}$, being $\mathbb{H}$ the non-commutative ring of the quaternions. In (5), $\Delta_{\text{quat}}(\cdot,\cdot)$ denotes the quaternionic distance [23, eq. (4)]. Note that the rotational term of the distance is bound in $[0, 1]$ while the positional term has no upper bound. The parameter λ_{o} is introduced as a scaling factor to weigh the two terms.[1]

This definition of the task loss uses odometry to propagate the estimate of $\boldsymbol{y}$ (produced by the detector in a timestep $u \in \mathcal{T}_d$), to any other timestep t in the same episode. If both the detector and the odometry are ideal (i.e. error-free), using any $u \in \mathcal{T}_d$ yields the same value of $\tilde{\boldsymbol{y}}(t) = \tilde{\boldsymbol{p}}(t,u) \oplus \boldsymbol{d}(u)$. Otherwise, if the detector and/or the odometry are not ideal, every different choice of u yields a different estimate of $\tilde{\boldsymbol{y}}(t)$. In practice, this is expected to mitigate inaccuracies as errors are averaged out during training.

[1]In principle, other options for representing poses and their distance [24], [25] might be adopted, as long as the distance function is continuous and derivable.

C. State-Consistency Loss

Consider two timesteps t, u in the same sequence, and assume that $t, u \notin \mathcal{T}_d$. The state-consistency loss enforces that the predictions of the model at t and u are consistent with each other [4], accounting for the robot's odometry between t and u. More specifically,

$$\mathcal{L}_{\text{sc}} = \sum_{t,u\in\mathcal{T}} \Delta_{\text{SE3}}(\tilde{\boldsymbol{p}}(t,u) \oplus \boldsymbol{m}(\boldsymbol{x}(u)|\boldsymbol{\theta}), \boldsymbol{m}(\boldsymbol{x}(t)|\boldsymbol{\theta})). \tag{6}$$

Consider the following example; given $\boldsymbol{x}(t)$, the model returns an estimated pose for an OOI 1.5m in front of the robot; after the robot advances 1m, at time u, the model given $\boldsymbol{x}(u)$ should return a pose that is 0.5 m in front of the robot. The state-consistency loss ensures that predictions $\hat{\boldsymbol{y}}(t) = \boldsymbol{m}(\boldsymbol{x}(t)|\boldsymbol{\theta})$ and $\hat{\boldsymbol{y}}(u) = \boldsymbol{m}(\boldsymbol{x}(u)|\boldsymbol{\theta})$ match this expectation.

D. Dealing With Uncertainty

Our approach relies on two sources of information, namely the detector $\tilde{\boldsymbol{d}}(t)$ and the odometry $\tilde{\boldsymbol{p}}(t,u)$, both of which are possibly affected by measurement errors (instantaneous for the detector, accumulated over time for the odometry). If such errors can be modeled, we can explicitly account for them in our approach.

In particular, we represent the uncertainty of $\boldsymbol{d}(t)$ by considering that the detector's output is, instead of a pointwise estimate of the target pose, a probability distribution over poses, defined in SE(3). We denote such probability distribution as $\boldsymbol{D}(t)$.

Similarly, for odometry we define $\boldsymbol{P}(t,u)$ as the probability distribution of the relative pose of $\boldsymbol{p}(t,u)$; this also accounts for the fact that odometry errors accumulate over time, and therefore are not independent for different times.

This representation allows to reformulate the task loss as

$$\mathcal{L}_{\text{task}} = \sum_{\substack{t\in\mathcal{T}_d \\ u\in\mathcal{T}}} \mathbb{E}\Big[\, \Delta_{\text{SE3}}\left(\boldsymbol{p}(t,u) \oplus \boldsymbol{d}(t), \boldsymbol{m}(\boldsymbol{x}(u)|\boldsymbol{\theta})\right)\Big], \tag{7}$$

and similarly, the state-consistency loss can be rewritten as

$$\mathcal{L}_{\text{sc}} = \sum_{t,u\in\mathcal{T}} \mathbb{E}\Big[\, \Delta_{\text{SE3}}\left(\boldsymbol{p}(t,u) \oplus \boldsymbol{m}(\boldsymbol{x}(u)|\boldsymbol{\theta}), \boldsymbol{m}(\boldsymbol{x}(t)|\boldsymbol{\theta})\right)\Big] \tag{8}$$

where $\boldsymbol{p}(t,u) \sim \boldsymbol{P}(t,u)$ and $\boldsymbol{d}(t) \sim \boldsymbol{D}(t)$.

In practice, when implementing the losses we approximate the expectation as the average over a finite number of realizations, in which $\boldsymbol{d}(t)$ and $\boldsymbol{p}(t,u)$ are Monte Carlo samples of the respective distributions.[2]

IV. Experimental Setup

This section validates our approach with three applications that differ in complexity and input dimensionality: (i) pose estimation of an OOI with a robotic arm; (ii) robot heading estimation using infrared sensors; and (iii) indoor localization of a ground robot.

[2]In a straightforward implementation, this multiplies the number of training samples by a factor equal to N_{mc}, and yields a correspondingly longer training time; in contrast, no additional computation is needed during inference.

A. Object of Interest Pose Estimation With a Robotic Arm

We consider the robotic manipulator Panda by Franka Emika, equipped with an Intel RealSense D435 sensor [26] at its end-effector, simulated with Gazebo [27], see Table I. The task is to estimate the 3D pose of an OOI, i.e. a colored mug, which is equipped with a small visual fiducial marker that is visible only when the cup is observed from a specific viewpoint. The frames $\mathcal{F}_0$ and $\mathcal{F}_r$ are placed at the base of the robot and at its end-effector respectively; while the frame $\mathcal{F}_i$ of the OOI is in the center of the mug. The end-effector pose $\boldsymbol{p}(t)$ is given by the robot forward kinematics. The input $\boldsymbol{x}(t)$ is a 160×120 pixel RGB image acquired from the RealSense camera. The estimates $\boldsymbol{d}(t)$ are generated by the AprilTag [28] off-the-shelf fiducial marker detector operating on $\boldsymbol{x}(t)$; only when the marker is clearly visible in the frame, the detector returns a noisy estimate of the 3D pose of the marker w.r.t. $\mathcal{F}_r$[3].

As shown in Table I, the environment consists of a flat table of 90×90 cm with different objects, some textured and some others having a solid color, besides the mug. In each training episode, the table and objects color, the position of the objects, and the direction of the light illuminating the scene are randomized to generate different environments. For each environment, the robot moves the end-effector in order to reach a total of 32 goal poses using the ROS MoveIt [29] implementation of the RRT planner. Each goal position is sampled from a semi-sphere having a radius of 55 cm placed at a height of 35 cm from the table; the goal orientation is set to make the camera look towards a random point lying 5 cm above the table. For each environment, the end-effector pose is initialized at the center of the semi-sphere.

The collected data amounts to 237 k tuples (of which only 78 k have the mug visible), corresponding to 157 environments and 5 simulated hours. The data is finally split into a training set (119 environments), a validation set (18 environments), and a testing set (20 environments). Training is performed with a $\lambda_{\mathrm{o}} = 10$, striking a balance between the positional and rotational errors.

B. Robot Heading Estimation Using Infrared Sensors

For this application, we use a Thymio [30], a small differential drive robot equipped with 7 infrared sensors: 5 mounted at the front and 2 at the rear of the robot body. Each sensor measures the amount of infrared light reflected from the environment, which is related in some unknown way to the distance and orientation of the sensor with respect to an obstacle. The input of the model $\boldsymbol{x}(t)$ consists in the uncalibrated readings of the 7 sensors at time t, while $\boldsymbol{y}(t)$ is the angle of the wall w.r.t the robot. Note that we can still adopt Δ_{SE3}, setting $\lambda_{\mathrm{o}} = 0$, thus considering only the heading. The robot odometry $\tilde{\boldsymbol{p}}(t)$, derived from the wheel encoders, provides the 2D transformation of the robot frame $\mathcal{F}_r$ w.r.t. the inertial frame $\mathcal{F}_0$. The scenario, along with an illustrative schematic of the sensor readings, is shown in Table I.

[3]The rigid transformation between the end-effector and the camera frame is assumed to be known, e.g., through a calibration procedure. Similarly, the offset between the origin of $\mathcal{F}_i$ and the center of the marker is taken into account in our computations.

Episodes are collected by teleoperating the robot along trajectories in the proximity of the wall. During each episode, the robot true pose is tracked by a fixed tracking infrastructure (12 Optitrack cameras), which is used as a comparison for experiments. At the beginning of each episode, the robot touches the wall with its rear side, thus the inertial frame $\mathcal{F}_0$ coincides with the robot frame $\mathcal{F}_r$ at this instant in time. This piece of information also acts as a virtual detector, whose output $\tilde{\boldsymbol{d}}(t)$ is available only in the first timestep of each episode.

Information about the rotation of each wheel is computed by the robot's firmware by measuring the current flowing through each motor – an inaccurate approach whose errors we model to compute the robot's uncertain odometry.

A total of 16 distinct episodes are recorded, each lasting on average 34 seconds, during which samples are collected at 10 Hz (a total of 5453 samples). Episodes are split into training and validation sets for experiments using a leave-one-episode-out cross-validation scheme.

C. Indoor Localization of a Ground Robot

We consider a wheeled ground robot, the DJI RoboMaster EP, equipped with an onboard camera and omnidirectional motion capabilities using Swedish wheels, see Table I. We consider a situation in which the robot navigates a given indoor environment and, when required, needs to come back to a fixed docking station (e.g. to recharge the batteries). In our setting, the docking station is represented by a mark on the floor. The input $\boldsymbol{x}(t)$ is the camera stream, downsampled to a resolution of 160×120 pixels; the perception task consists in predicting the pose of the docking station relative to the robot frame $\mathcal{F}_r$, given an image $\boldsymbol{x}(t)$ acquired at a generic pose. Note that, since the docking station is fixed in the environment, this problem is equivalent to robot localization; it differs from the object localization task presented in Sec. IV-A because the docking station does not need to be visible in the input image for successful estimation.

The robot onboard odometry module provides $\tilde{\boldsymbol{p}}(t)$ w.r.t. the inertial frame $\mathcal{F}_0$, which coincides with $\mathcal{F}_r$ at the beginning of the acquisition. Figure 3 visualizes the inaccurate robot odometry as measured, and 50 realizations accounting for uncertainty, for the first minute of a training episode.

Furthermore, at time $t = 0$ the robot undocks from the docking station; similarly to the previous case, we use this information as a detector. In this case study, the OOI is the docking station and its frame $\mathcal{F}_i$ coincides with frame $\mathcal{F}_0$, as in Sec. IV-B.

We collect 20 episodes recording data at 15 Hz, for a total of 70 k samples in 80 minutes (4 minutes per episode). Each episode begins with the robot attached to the docking station; the robot is then teleoperated to explore the environment. Data from different episodes are split into a training set (40 k samples, 10 episodes), a validation set (15 k samples, 5 episodes), and a testing set (15 k samples, 5 episodes). Training is performed with $\lambda_{\mathrm{o}} = 10$, similarly to Sec. IV-A.

D. Model Training

In all case studies, a NN is trained using Adam [31] as optimizer with a learning rate of $1\mathrm{e}^{-3}$; early stopping is used to

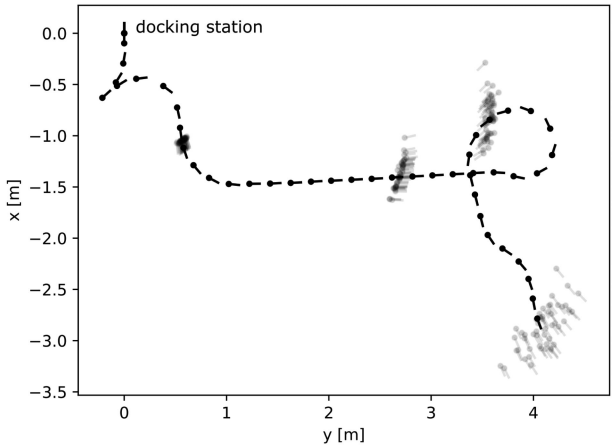


Fig. 3. Visualization of measured inaccurate odometry (black), and 50 realizations of the uncertain odometry, for the first minute of a training episode of the RoboMaster robot.

determine when to conclude the training process. An architecture based on MobileNet-V2 [32] with a total of 1 million parameters is used for the case studies in Sec. IV-A and Sec. IV-C; it maps a 160×120 RGB image to an output vector representing a 3D pose (composed of 7 elements, 3 for the position and 4 for the quaternion). In these two use cases, we artificially increase the amount of data used for the training, by adopting the following data augmentation techniques on the input images: blurring, multiplicative Gaussian noise, random brightness and contrast, random resized cropping.

In the case study detailed in Sec. IV-B, we use a simpler NN architecture composed of four linear layers with a total of 1000 parameters, mapping the 7 infrared sensors' readings to a 3D pose.

V. Results and Discussion

A. Object of Interest Pose Estimation With a Robotic Arm

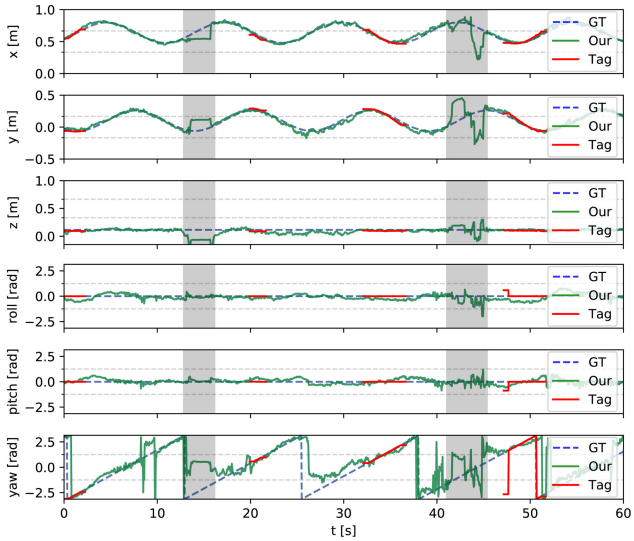
For the scenario described in Sec. IV-A, we first evaluate the trained model on the testing set, by comparing the predicted poses $\hat{\boldsymbol{y}}$ to the corresponding ground-truth $\boldsymbol{y}$. Predictions occur independently for every frame; no information from state estimation or the detector is used in the process.

For the position component, the model achieves a Root Mean Squared Error of 39.9mm, and a coefficient of determination R^2 of 0.962, 0.960 and 0.866 on the x, y and z components, respectively. The coefficient of determination is an adimensional measure of the quality of a regressor, which quantifies the amount of variance in the target variable that is explained by the model; an ideal regressor yields $R^2 = 1$, whereas a dummy regressor estimating the mean of the target variable yields $R^2 = 0$; we observe that, while all components are estimated well, the z coordinate, i.e. the distance of the OOI w.r.t. the camera, is estimated with lower accuracy; this is expected, since estimating distances is hard from low-resolution monocular images. The rotational component is estimated with an average rotational error $\Delta_{\text{quat}} = 0.1417$, corresponding to an angle of about $25°$.

As a comparison, we also trained a supervised model using ground-truth poses of the OOI instead of the detector outputs:

(a) Predictions (magenta-lime-cyan colored frame) and reconstructed ground-truth (red-green-blue frame) on testing set data.

(b) Input images from the additional testing scenario.



(c) Pose of the OOI in the inertial frame on the additional testing scenario.

Fig. 4. Prediction of the OOI pose with a robotic arm on the testing set (a); input images (b), ground-truth and predictions (c) relative to the additional testing scenario. In (b-c) a mug is placed on top of a rotating disk support. Grayed-out areas indicate time intervals in which the mug is not visible.

the resulting prediction performance is the same as with the self-supervised approach. In fact, in the considered simulated environment the detector is very accurate and state estimation is ideal, which makes the self-supervised labels almost identical to ground-truth labels. This is not the case for the experiment in Section V-B.

Fig. 4(a) shows a sequence of camera frames from the testing set with an overlay of the model's prediction and the ground-truth. We observe that the model correctly identifies the object and accurately estimates its pose, even when the OOI is only partially visible or occluded, and independently on the visibility of the fiducial marker used during training. When the object is not visible at all on the image, the model tends to confuse it with other objects that are visually similar; this highlights a limitation of our approach: we do not explicitly handle aliasing, i.e. the case in which $\boldsymbol{x}$ does not contain enough information to estimate $\boldsymbol{y}$.

To demonstrate the fact that the model, once trained, can be used in dynamic environments, we consider an additional testing scenario: the mug is placed on a rotating disk support

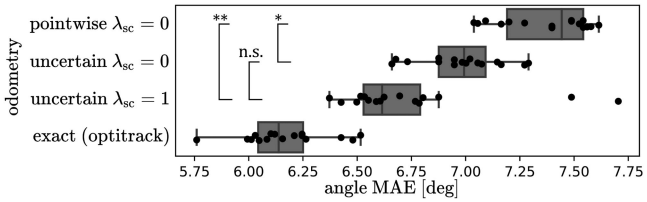


Fig. 5. Mean absolute angle error (lower is better) in the heading estimation task. * $p = 0.0003$; ** $p = 0.0008$; n.s. means not significant ($p = 0.029$).

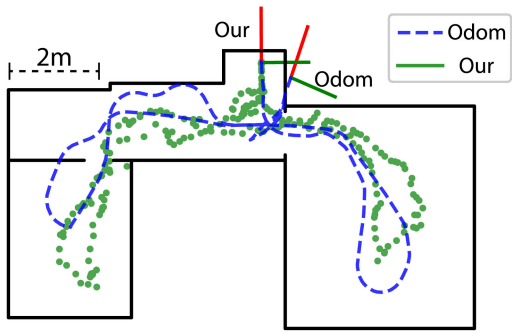


Fig. 6. Robot localization task on testing data.

and observed while the robot also moves. Fig. 4(b) shows a sequence of camera frames captured during the experiment. Fig. 4(c) compares the model prediction to AprilTag detections, and to ground-truth. For the sake of clarity, the target has been transformed in the frame $\mathcal{F}_0$, along with the model predictions. The model (green in the plots) manages to predict a consistent pose (it well overlaps the blue dashed line of the ground-truth), outperforming the AprilTag detection (red line), which provides a measure for a small fraction of frames in which the tag is visible, while our model runs at 25 Hz on a GPU Nvidia Quadro P2000. The model produces good estimates of the mug pose from most points of view, even when it appears upside down. Occasional failures occur when the mug is seen from a point of view that does not provide any reference to infer its actual rotation around the vertical axis – i.e., when both the marker and the handle of the mug are not visible. While it is robust to partial occlusions of the mug, the model fails to estimate the mug's pose when a very small portion of it is visible, or when it is totally invisible; time intervals in which the mug is not visible are depicted with a gray shadow in Fig. 4(c).

B. Robot Heading Estimation Using Infrared Sensors

For the scenario described in Sec. IV-B we consider three possible sources of odometry: *exact odometry*, where $\tilde{\boldsymbol{p}}(t,u) = \boldsymbol{p}(t,u)$, as measured by the optitrack system, acting as an upper bound of the achievable performance; *pointwise odometry*, where the relative pose $\tilde{\boldsymbol{p}}(t,u)$ is computed according to the known robot kinematics and the readings of the wheel rotation sensors between timesteps t and u; *uncertain odometry*, where the probability distribution $\tilde{\boldsymbol{P}}(t,u)$ is approximated by $N_{\text{mc}} = 50$ realizations of the odometry between timesteps t and u, obtained by corrupting the readings of the wheel rotation sensors with white Gaussian noise, whose variance matches the known uncertainty of the sensor.

Fig. 5 reports the Mean Absolute Error between the predicted angle of the robot pose with respect to the wall, against the ground-truth angle as measured by the optitrack system; this metric is reported for four models trained with different odometry sources, with ($\lambda_{\text{sc}} = 1$) or without ($\lambda_{\text{sc}} = 0$) the state-consistency loss. Each of the four models is trained and evaluated 16 times according to the leave-one-episode-out cross-validation scheme. We observe that: i) using uncertain odometry instead of pointwise odometry improves the prediction performance of the resulting model;[4] ii) additionally enforcing the state-consistency loss ($\lambda_{\text{sc}} = 1$) further improves performance, even though the incremental improvement over the uncertain model with $\lambda_{\text{sc}} = 0$ is not statistically significant; iii) compared to the model trained with exact odometry, serving as a supervised learning upper-bound, the performance gap of our best model is less than half of the performance gap of the baseline model (pointwise $\lambda_{\text{sc}} = 0$), i.e. 0.57° vs 1.20°.

C. Indoor Localization of a Ground Robot

For the scenario described in Sec. IV-C, we report one qualitative and one quantitative experiment.

In the qualitative experiment, we train a model using uncertain odometry ($N_{\text{mc}} = 50$) and $\lambda_{\text{sc}} = 1$, then apply the trained model independently on each frame of a testing episode. Fig. 6 shows the position component of the predicted poses (green), compared to the poses returned by the robot's odometry (blue) for the same trajectory; despite the short duration of the testing episode (about 4 minutes), the drift of the odometry trajectory is apparent (e.g. the blue trajectory passes through a wall on the left); in comparison, the poses predicted by our approach are locally noisy but not affected by accumulating error. The figure also depicts the predicted pose at the end of the trajectory for each of the two methods; our model correctly predicts that the robot is back at the docking station, whereas the odometry has drifted by about 0.5m and 20°.

In the quantitative experiment, we consider four timesteps $t \in t_1, t_2, t_3, t_4$ of the testing episode, whose ground-truth poses $\boldsymbol{p}(t)$ have been manually measured with respect to the docking station. For each of the four timesteps, we measure the positional and rotation components of the error against such ground-truth, for: i) the pose $\tilde{\boldsymbol{p}}(t)$ estimated by the robot odometry; ii) the poses estimated by each of four models. In particular, we consider models trained with pointwise or uncertain odometry, with or without using the state-consistency loss. Table II shows that: using the state-consistency loss improves both positional and rotational errors; using uncertain odometry during training consistently outperforms pointwise odometry.

[4]We use the non-parametric Wilcoxon signed-rank test between matched samples, i.e. the performance of two models on the same cross-validation fold ($p = 0.0003$).

TABLE II
QUANTITATIVE PERFORMANCE ON THE INDOOR LOCALIZATION TASK

Method	Position Error [mm]	Rotation Error [deg]
Odometry	84.9	11.9
Pointwise, $\lambda_{sc} = 0$	62.8	23.8
Pointwise, $\lambda_{sc} = 1$	49.1	7.5
Uncertain, $\lambda_{sc} = 0$	73.5	9.1
Uncertain, $\lambda_{sc} = 1$	**35.2**	**3.8**

VI. CONCLUSION

We presented a general self-supervised learning approach for spatial perception tasks, and instantiated it on three different case studies. The approach is general enough to be applicable to different robots and sensor apparatus, requiring only a possibly uncertain odometry and a detector that sparsely produces a ground-truth estimate. A novel loss allows us to evaluate the model outputs also for timesteps in which no supervision is available, by propagating such supervision from different timesteps using uncertain state estimates. Furthermore, the loss formulation enforces consistency among predictions at different timesteps, which further improves performance. Results show consistent and statistically significant improvements of models learned with the uncertainty-aware version of the loss compared to the baseline.

REFERENCES

[1] L. Jing and Y. Tian, "Self-supervised visual feature learning with deep neural networks: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published, 2020.

[2] H. Song, M. Kim, D. Park, Y. Shin, and J.-G. Lee, "Learning from noisy labels with deep neural networks: A survey," 2020, *arXiv:2007.08199*.

[3] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 3948–3955.

[4] M. Nava, L. Gambardella, and A. Giusti, "State-consistency loss for learning spatial perception tasks from partial labels," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 1112–1119, Apr. 2021.

[5] D. Lieb, A. Lookingbill, and S. Thrun, "Adaptive road following using self-supervised learning and reverse optical flow," in *Robot.: Sci. Syst.*, vol. 1, pp. 273–280, 2005.

[6] A. Kouris and C.-S. Bouganis, "Learning to fly by myself: A self-supervised cnn-based approach for autonomous navigation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 1–9.

[7] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50 k tries and 700 robot hours," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 3406–3413.

[8] E. Jang, C. Devin, V. Vanhoucke, and S. Levine, "Grasp2vec: Learning object representations from self-supervised grasping," in *Proc. Conf. Robot Learn.*, 2018, pp. 99–112.

[9] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox, "Self-supervised 6d object pose estimation for robot manipulation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 3665–3671.

[10] M. Nava, J. Guzzi, R. O. Chavez-Garcia, L. M. Gambardella, and A. Giusti, "Learning long-range perception using self-supervision from short-range sensors and odometry," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1279–1286, Apr. 2019.

[11] L. Wellhausen, A. Dosovitskiy, R. Ranftl, K. Walas, C. Cadena, and M. Hutter, "Where should i walk? predicting terrain properties from images via self-supervised learning," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1509–1516, Apr. 2019.

[12] J. Zürn, W. Burgard, and A. Valada, "Self-supervised visual terrain classification from unsupervised acoustic feature learning," *IEEE Trans. Robot.*, vol. 37, no. 2, pp. 466–481, Apr. 2021.

[13] P. Agrawal, J. Carreira, and J. Malik, "Learning to see by moving," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 37–45.

[14] G. Iyer, J. Krishna Murthy, G. Gupta, M. Krishna, and L. Paull, "Geometric consistency for self-supervised end-to-end visual odometry," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2018, pp. 267–275.

[15] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1851–1858.

[16] A. Loquercio, M. Segu, and D. Scaramuzza, "A general framework for uncertainty estimation in deep learning," *IEEE Robot. Automat. Lett.*, vol. 5, no. 2, pp. 3153–3160, Apr. 2020.

[17] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *Proc. Int. Conf. Mach. Learn.* , 2015, pp. 1613–1622.

[18] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1050–1059.

[19] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. B. Grosse, "Flipout: Efficient pseudo-independent weight perturbations on mini-batches," in *Proc. Int. Conf. Learn. Representations*, 2018.

[20] A. Zeng *et al.*, "Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1386–1383.

[21] S. Ratz, M. Dymczyk, R. Siegwart, and R. Dubé, "Oneshot global localization: Instant lidar-visual pose estimation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 5415–5421.

[22] P. Corke, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB Second, Completely Revised*. Springer, 2017, vol. 118.

[23] S. Mahendran, H. Ali, and R. Vidal," 3D pose regression using convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 2174–2182.

[24] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 5745–5753.

[25] V. Peretroukhin, M. Giamou, W. N. Greene, D. Rosen, J. Kelly, and N. Roy, "A smooth representation of belief over SO(3) for deep rotation learning with uncertainty," in *Proc. Robotics: Sci. Syst.*, Corvalis, Oregon, USA, 2020.

[26] L. Keselman, J. Iselin Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel realsense stereoscopic depth cameras," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops*, 2017, pp. 1–10.

[27] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, vol. 3, 2004, pp. 2149–2154 .

[28] J. Wang and E. Olson, "Apriltag 2: Efficient and robust fiducial detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 4193–4198.

[29] D. Coleman, I. A. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: A moveit! case study," *J. Softw. Eng. Robot.*, 2014, *arXiv:1404.3785*.

[30] F. Mondada *et al.*, "Bringing robotics to formal education: The thymio open-source hardware robot," *IEEE Robot. Automat. Mag.*, vol. 24, no. 1, pp. 77–85, Mar. 2017.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic gradient descent," in *Proc. Int. Conf. Learn. Representations ICLR*, 2015, pp. 1–15.

[32] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.