IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 13, 2021, accepted August 15, 2021, date of publication September 1, 2021, date of current version September 13, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3109733

Hardware-Aware Affordance Detection for Application in Portable Embedded Systems

EDOARDO RAGUSA[1], CHRISTIAN GIANOGLIO[1], STRAHINJA DOSEN[2], (Member, IEEE), AND PAOLO GASTALDO[1]

[1]Department of Electrical, Electronic and Telecommunications Engineering, and Naval Architecture, DITEN, University of Genoa, 16145 Genova, Italy
[2]Department of Health Science and Technology, Aalborg University, 9220 Aalborg, Denmark

Corresponding author: Edoardo Ragusa (edoardo.ragusa@edu.unige.it)

ABSTRACT Affordance detection in computer vision allows segmenting an object into parts according to functions that those parts afford. Most solutions for affordance detection are developed in robotics using deep learning architectures that require substantial computing power. Therefore, these approaches are not convenient for application in embedded systems with limited resources. For instance, computer vision is used in smart prosthetic limbs, and in this context, affordance detection could be employed to determine the graspable segments of an object, which is a critical information for selecting a grasping strategy. This work proposes an affordance detection strategy based on hardware-aware deep learning solutions. Experimental results confirmed that the proposed solution achieves comparable accuracy with respect to the state-of-the-art approaches. In addition, the model was implemented on real-time embedded devices obtaining a high FPS rate, with limited power consumption. Finally, the experimental assessment in realistic conditions demonstrated that the developed method is robust and reliable. As a major outcome, the paper proposes and characterizes the first complete embedded solution for affordance detection in embedded devices. Such a solution could be used to substantially improve computer vision based prosthesis control but it is also highly relevant for other applications (e.g., resource-constrained robotic systems).

INDEX TERMS Affordance prediction, CNNs, prosthetics, embedded systems.

I. INTRODUCTION

The invention and spread of deep learning made a profound impact in many fields of engineering. Computer vision techniques can now almost compete with humans in many recognition and classification tasks [1]. Accordingly, researchers could approach new or well-known problems in novel ways. Many important applications use extensively such technologies; e.g., healthcare, sentiment analysis, robotics, cybersecurity and autonomous driving.

Affordance detection is one of the problems that have been approached satisfactorily only after the development of deep learning techniques. This problem consists in detecting functional interactions between objects and humans [2]. More specifically, affordance detection identifies potential interactions that an object can afford and segments the object into parts according to these functions. It is solved by means of image segmentation: Fig. 1 presents the input-output relationship. In particular, Fig. 1(a) represents an input image, while Fig. 1(b) shows the segmentation masks, where each pixel is colored based on its functional role. The pixels of the background are black, and the scoop is divided in two parts, based on the functionality: the light blue individuates the handle, whereas the dark blue is the functional part that should not be grasped. Affordance detection is important in all those applications where an agent should manipulate or interact with objects. For example, knives should be grasped with an appropriate predisposition of the hand. Similarly, when approaching a suitcase, the agent should target the grip and not the whole suitcase.

Robotics literature presented excellent solutions for affordance detection using RGB cameras, deep learning tools, and powerful computing units [3]. However, embedded resource-constrained systems struggle in adopting those techniques. One of the main reasons is the computational requirements: in fact, in standard robotics and computer vision applications, the state-of-the-art solutions employ GPUs for scientific calculations also during the inference phase [3]. In practice, a hardware with the size of a desktop computer is needed, without mentioning the energy consumption. One could

The associate editor coordinating the review of this manuscript and approving it for publication was Yongjie Li.

argue that cloud based applications are a reliable option. However, the assumption of a stable, low latency and always-available connection is unrealistic. Accordingly, there is a demand for scalable solutions for affordance detection that are implementable on embedded systems subject to hard constraints in terms of power consumption and dimension. Fortunately, the market has recently made available hardware platforms suitable for the deployment of deep learning solutions in resource-constrained systems [4]. Thus, the compute node can be placed in the end device in accordance with the framework of edge computing.

A new generation of smart robotic prosthetic limbs [5]–[7] is an example of systems that would profit from advanced computer vision methods if they can be implemented on a platform with limited computing power. In principle, advanced prostheses with multiple degrees of freedom enable dexterous manipulations of daily life objects [8]. The methods and technologies for human-machine interfacing between the users and their bionic limbs have been improved significantly in recent years [9]. Nevertheless, the communication bandwidth is still insufficient to allow the users to effectively and intuitively exploit the functionality available in advanced prostheses with many degrees of freedom, in particular with non-invasive solutions [10]. In the conventional approach to prosthesis control, the user needs to command each prosthesis function explicitly by generating discriminable patterns of muscle activation, which can be slow and cognitively taxing especially with complex systems with multiple functions. Normally, in commercial prostheses, the user can control a single function at a time and to switch to another degree of freedom, he/she needs to produce a special switching signal (e.g., co-contraction). In recent commercial systems with pattern classification [11], [12] the user can select a degree of freedom directly, but he/she is still limited to controlling prosthesis movements sequentially (one movement at a time). Despite some controversies about embodiment [13], semi-autonomous control could substantially improve prosthesis performance. In this approach, the prosthetic limb is equipped with additional sensors (e.g., touch sensors, electromyography sensors, stereo and RGB-D cameras, and gyroscopes) and cognitive-like processing that enable the system to perform some tasks autonomously (e.g., preshape into an appropriate grasp); the overall goal is to reduce the cognitive burden of control for the user [6], [14]. As explained later in more detail (Sec. II), a typical semi-autonomous prosthetic limb relies on a camera to estimate the properties of the target object and, based on this, automatically assume an appropriate grasp. If the computer vision system in a prosthesis would implement the affordance detection, the limb controller could select the grasp configuration that is adapted to the functionality afforded by the object.

The present paper is an important step in this direction. More specifically, we propose a novel approach to affordance detection that is suitable for the deployment of the inference phase within a resource constrained embedded device. This paper therefore bridges the gap between embedded systems and robotics for the problem of affordance detection. The proposed approach is characterized with the following features. First, the novel solution is based on hardware-aware deep neural architectures that reduce the overall hardware requirements of the trained model, i.e., the model that should be deployed on the embedded device to support the inference. Second, affordance detection is achieved by elaborating only RGB images. Therefore, the framework does not use depth sensors that are more expensive, bulky, and power demanding compared to RGB cameras. Third, the hardware-aware affordance model is tested on three existing embedded devices suitable to host the real time inference system, namely, NVIDIA Jetson TX2, NVIDIA Jetson Nano, and smartphones. Finally, tests involve a setup that mimics the conditions of an envisioned application (i.e., prosthesis grasping control). Experimental outcomes show that the hardware-aware affordance model can achieve the same accuracy on standard benchmarks as the state-of-the-art solutions employed in robotics.

(a) Input (b) output

FIGURE 1. Example of the input (a) output (b) relation in affordance prediction problem. The input image (spoon) is segmented into background (black), handle (light blue) and bowl (dark blue). The handle can be grasped while the bowl can be used to scoop; hence, the object is segmented according to the affordance classes.

A. CONTRIBUTION

The present paper investigates factors that are important for the application of affordance models. The overall goal is to provide the reader with the pipeline for the design and deployment of these models on portable devices with limited resources.

The contribution of the paper can be summarized as the sum of the following points:

- The design of a framework based on hardware-aware deep learning architectures for affordance detection.
- A design space exploration for the deployment of affordance models on resource-constrained devices. The study evaluates three different affordance models supported by as many hardware-aware deep learning architectures.
- Real time implementation of the trained models for affordance detection on small size embedded devices, namely, NVIDIA Jetson TX2 and NVIDIA Jetson Nano, and smartphones.
- Characterization of computational performance of the embedded systems (latency, memory requirements and power consumption).

- An affordance model that can compete with solutions employed in robotics in terms of accuracy using substantially less computing resources.
- Experimental assessment of the proposed solution in real-life conditions (stress test) including different objects, background, illumination conditions and viewing angles.

II. RELATED WORKS

The research in robotics produced the best solutions for object detection, classification and manipulation using computer vision. However, most of these solutions require high–performance computations. In the following, we focus on reviewing the state–of–the–art affordance detection solutions for robotics, as well as the previous developments in the field of semi-autonomous control of prostheses employing computer vision.

The problem of understanding affordances at the pixel level is termed as ''affordance detection'' in robotics, where researchers focus on the real-world objects with which the robot can interact [2]. Conversely, in computer vision community this problem is also known as ''object part labelling'', and is not restricted to objects.

With the rise of deep learning, recent works relied on feature learning for affordance detection. In [15], two deep neural networks detected grasp affordances from RGB images. Later, deep features from CNNs were used for affordance detection in RGB-D images [16], obtaining significant gains with respect to previous works [2]. In [17], the authors introduced a multi-scale CNN to localize environment affordances. A weakly supervised deep learning approach to segment object affordances was used in [18]. This method avoided costly pixel ground truth labels. In [3], a deep learning-based object detector improved the affordance detection accuracy on a real-world dataset by automatically selecting objects inside an image.

Most of these studies did not focus on the computational cost of the proposed solutions. This is reasonable because the primary goal was to improve accuracy. Next, most of these solutions used RGB-D sensors, which are less suitable than simple RGB cameras for small size, low-power systems, with hard constraints in size and consumption. In addition, the contribution of the D component is mostly on the control strategy, rather than in affordance recognition. The constraints regarding memory and number of computations per single inference were not considered in these studies. And indeed, such approach can lead to high accuracy, however it produces solutions that cannot be implemented in an embedded system with limited computational resources.

In general, controlling a smart robotic prosthesis is a unique challenge compared to controlling fully autonomous robotic systems. Indeed, prostheses coexist with humans. Therefore, some parts of the control flow can be relegated to the user, thereby reducing the overall complexity of the control pipeline [19]. The semi-autonomous control of robotic prosthetic limbs is a relatively new research field. The literature presents several solutions where semi-autonomous systems employed computer vision. The initial studies proposing this concept [20], [21] used a web camera placed on a prosthesis to provide a snapshot of the target object and an ultrasound sensor to measure distance to the object. Upon triggering by the user (myoelectric signal), the prosthesis automatically selected grasp type and aperture size according to the object width and height. The later version of the system relied on a stereo-camera [6], [14] and depth sensor to obtain a 3D information. The algorithm used the extracted cloud of points to model target objects using predefined geometrical shapes (box, sphere, cylinder and line). The estimated model was then used to automatically adjust the prosthesis grasp type, aperture size and wrist orientation. In [5], instead of explicitly estimating the object properties, the authors adopted a grasp classification approach implemented using CNNs trained on the Columbia Object Image Library (COIL) dataset. The detected objects were clustered in four groups, each associated to a specific grasping strategy. A similar approach was followed by Taverne *et al.* [22] where deep learning framework was used to predict the action of grasping. In [23], an object detection and an image segmentation architecture were deployed into a prosthesis hand equipped with a camera. The two architectures were designed from scratch. The experiments showed good results in terms of object segmentation and classification accuracy. Similarly, in [24] an object detection network classifies different objects driving the selection of grasping patterns based on the object kind, without considering the actual grasping surface. The uncertainty introduced by partial occlusion and movements was explicitly modeled in [25]. In [26] a resource aware method was introduced to classify hand-grasping actions.

Therefore, the previous studies on the semi-autonomous prosthesis control all relied on estimating the global properties of the target object (e.g., object overall shape and size). Such global characterization substantially limits the subsequent reasoning about the preferred grasping strategy. In many cases, objects of daily life have complex geometry that cannot be captured with a single geometric primitive (e.g., a simple cylinder or a box). The affordance detection, if it can be implemented within a resource-constrained device, would allow segmenting and modeling the object component parts that are particularly relevant for grasping (e.g., a handle). This information is critical for an accurate selection of prosthesis grasp parameters.

III. MATERIAL AND METHODS

This research proposes a novel solution for affordance detection that is suitable for deployment of the inference system in an embedded device (e.g., prosthesis).

The paper divides the design of the system into three parts:

1) **Affordance Detection**: the first phase tailors state-of-the-art solution for affordance detection in robotics to the specific application in a semi-autonomous prosthesis. In practice, this phase defines simplifications, algorithms and algorithmic choices required to deal

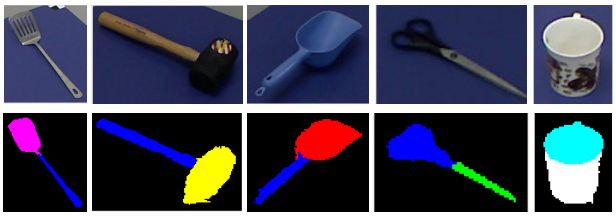

FIGURE 2. Example of objects with the corresponding affordance maps extracted from the UMD dataset.

with the affordance detection problem in this context. The outcome is an hardware-aware affordance model that -after the training process- can be deployed on edge devices.

2) **Deployment**: the second phase addresses the setup of the embedded device supporting the real-time implementation of affordance detection. Hence, this phase deals with the design of the eventual system for affordance detection.
3) **Solution Characterization**: the aim of the third phase is to characterize the hardware-aware affordance model and test it in different conditions.

Here in after, the three phases will be always identified using the three names proposed in the list.

In the following, Sec. III-A will explain the design choices. Section III-B will design the organization of the framework implementing affordance detection on resource-constrained embedded devices. Then, Sec. III-C will present the hardware-aware affordance models adopted in this research. Finally, Sec. III-D will analyze the edge devices involved in the deployment phase. The experimental protocol supporting solution characterization will be discussed in Sec. IV.

A. REQUIREMENTS FOR AFFORDANCE DETECTION IN PROSTHETICS

In robotics, most of the computing energy is spent in object localization because the robot must individuate objects from scratch, without a good framing. Instead, for prosthesis, object detection is not relevant since the user selects the target object for grasping. As described in Sec. II, the system can rely on the user to target the camera towards the object of interest. Therefore, in practice, one can assume that the graspable object is reasonably well-centered in the image under analysis.

In addition, in robotics affordance techniques encompass fine-grained classifications, as functional part recognition drives the whole inference process. For example, the University of Maryland (UMD) dataset [2] involves 7 affordance classes (grasp, cut, scoop, contain, pound, support and wrap-grasp), meanwhile in [16] the classification includes 9 classes. Figure 2 reports several representative objects with the corresponding affordance maps extracted from the UMD dataset. These classes are designed so that a fully autonomous robotic system receives a complete information about the

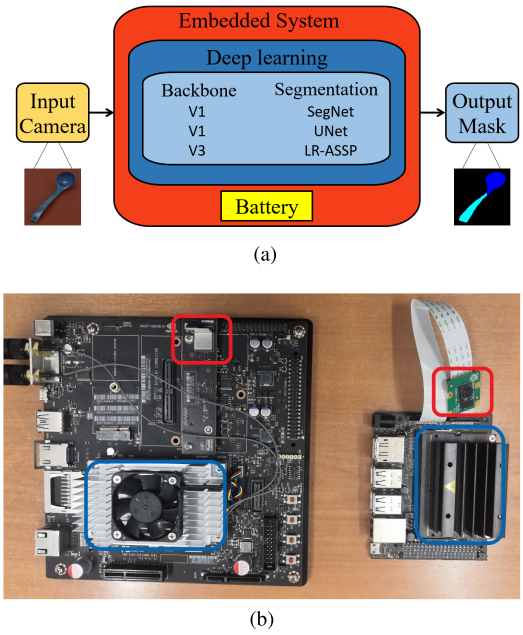

FIGURE 3. The proposed solution: (a) block scheme; (b) the embedded systems employed in the tests. In (b) Jetson TX2 is the device on the left, Jetson Nano is the device on the right. Red rectangles enclose the cameras; blue rectangles enclose the embedded devices hidden by the cooling systems of the development boards.

functions that are afforded by an object (e.g., an inside of a glass can "contain" water, a head of the hammer can be used to "pound"). However, fine-grained classification is not relevant in prosthetic applications since most of this information is known to the user. In practice, the main goal of the semi-autonomous system in prosthetics is to assist the user in grasping an object by automatically preshaping the prosthesis. Therefore, the prosthesis controller should only recognize which are the "graspable" elements of an object since this is a critical information for deciding an appropriate grasping strategy. Therefore, in this context, the affordance detection can be substantially simplified by reducing the number of classes to three: background, graspable and non-graspable. This simplification can in turn decrease the required processing resources.

Obviously, the hardware resources should be hosted on the prosthesis or in a device carried by the human user. For initial prototyping, such a device can be attached externally to the prosthesis socket, for instance, or worn in a pocket or a waist band. In the final solution, the system should be integrated within the prosthesis socket in the space normally allocated for electronics and batteries. Accordingly, weights and dimensions should be minimized. From this perspective, an RGB camera is preferable since it is smaller, cheaper and consumes less power with respect to RGB-D sensors. The battery is a critical component because the system should balance lifetime and weight. The embedded device should be fast enough to guarantee real time performance for the inference system. In the meantime, power consumption and

volume should be minimal. In practice, the right embedded system is a function of the selected algorithm for affordance detection.

Ideally, despite the limitations in hardware resources, the accuracy should still be high since wrong classification will decrease the reliability of the semi-autonomous system. This can negatively affect the user experience, as he/she would often need to correct the wrong decisions of the automatic controller. As explained in Sec. II the best solutions for affordance detection are based on computationally intensive deep learning. Nevertheless, the problem can be substantially simplified when considering the applications in prosthetics. In addition, the training phase can be managed offline by relying on powerful devices that can run computationally intensive procedures to determine the optimal hardware-aware network, which is then deployed on the embedded system. This was the driving point for the solution presented in this work.

B. PROPOSED FRAMEWORK

This paper presents a model for real-time affordance detection deployed on embedded accelerators. To this aim, the current research tackles the major challenges in the design of an affordance model that is expected to meet real-time constraints on resource-constrained devices. Figure 3 (a) shows the proposed framework. The image of the target object is acquired by an RGB camera and processed by the embedded system hosting a hardware-aware deep learning architecture for affordance detection. The system outputs a pixel map that associates each pixel in the input image with an affordance class. The deep learning components (backbone and segmentation) that comprise our solution are described in more details in Sec. III-C. Figure 3(b) shows the two embedded systems that hosted the inference engine in our tests. The components are described in Sec. III-D and the experimental assessment in section IV.

The scheme in Fig. 3 (a) emphasizes that the inference engine is hosted on a battery-operated embedded system that can be carried by a human user (portable solution). Indeed, the novel element of the proposed design is that the deep learning inference engine relies solely on hardware-aware networks, namely two different versions of the MobileNet architecture. This in turn is an enabling technology for many applications that involve resource-constrained devices. In fact, state-of-the-art deep learning frameworks for affordance detection cannot accommodate the constraints imposed by existing embedded accelerators.

C. AFFORDANCE DETECTION: HARDWARE-AWARE MODELS

Affordance detection is performed by means of image segmentation. Indeed, image segmentation architectures can be roughly divided into two main components: *backbone* and *segmentation*. The backbone network is the component entitled to perform low-level feature extraction. Then, a meta-architecture employing deconvolution operations [27] produces the segmentation masks, as per Fig. 1(b).

The selection of the AI techniques is critical because they affect the accuracy and the computational costs. VGG [28] and ResNet [29] are possibly the most famous CNNs for object classification. VGG employs up-to 138M, while ResNet involves from 10M to 59M parameters, based on the version. The number of parameters influence linearly the memory requirements, as parameters should be stored on the hardware device. In addition, the number of operations to be executed in the inference phase grows with the number of parameters. The strict relation depends on the network architecture. For this reason, in many applications one should deal with the trade-off between the accuracy of the eventual predictor and the computational cost.

The literature, though, provides other interesting models. For example, [30] demonstrated an excellent trade-off between accuracy and computational cost for classification, while BiSeNet architectures [31], [32] are probably the most appealing solution for real time image segmentation using high performance hardware.

When the focus moves to real-time inference on resource constrained devices, the family of MobileNets architectures [33]–[35] represents the state–of–the–art solution for computer vision, thanks to the use of depthwise convolutions. This family includes three different architectures, and the present research considered versions V1 and V3 as the backbone.

Many pre-trained versions of V1 are available; this is important since training a deep network from scratch is a time consuming task. Thus, it is reasonable to adopt V1 even if it is not the best option in terms of accuracy. V3 has been proposed more recently; hence, only a few pre-trained versions are available. The version Small of V3 [35] was selected in this work since it is the least computationally demanding release of V3. However, it is also expected to be less accurate.

Three meta-architectures for image segmentation were considered in the present work: SegNet [36], Unet [37] and LR-ASPP [35]. Notably, SegNet and UNet avoid fully-connected layers. Consequently, the number of parameters of the model is relatively small, especially when hardware-aware networks support the feature extraction process. LR-ASPP is the image segmentation meta-architecture tailored for V3; it can be considered as the reference for image segmentation on resource-constrained devices. Importantly, to the best of our knowledge, the present study is the first to investigate the application of the aforementioned backbone and segmentation networks in the context of affordance detection. More specifically, three alternative combinations backbone, meta-architecture were proposed for the hardware-aware affordance model. The following notation will be adopted to identify the resulting different architectures:

- *V1_Se*: MobileNetV1 + SegNet;
- *V1_U*: MobileNetV1 + UNet;
- *V3_LR*: MobileNetV3 + LR-ASPP.

TABLE 1. Available hardware configurations for Jetson Tx2 and Nano.

Dev.	Mode	Denver2		Cortex		GPU F.
		cores	GHz	cores	GHz	(GHz)
Tx2	Max-N	2	2.0	4	2.0	1.30
	Max-Q	0		4	1.2	0.85
	Max-P C.-All	2	1.4	4	1.4	1.12
	Max-P ARM	0		4	2.0	1.12
	Max-P Denv.	1	2.0	1	2.0	1.12
Nano	Max-N	-	-	4	1.5	0.92
	5W	-	-	2	0.9	0.64

The first two architectures are well-established solutions. Therefore, pre-trained models can be easily retrieved and fine-tuned for the specific problem. Conversely, V3_LR has been introduced recently. It is expected to be very effective in terms of computational cost. However, there are a limited number of sources for this solution. In particular, pre-trained configurations for low-resolution images are not yet available. The reader is referred to the Appendix for further details about the structure of the three architectures.

In summary, the eventual solutions were selected based on the trade-off between the two major design constraints: accuracy for the specific application, and computational cost.

D. DEPLOYMENT: EDGE DEVICES

The market offers several edge devices that can be used for deep learning. They differ in power constraints, dimensions, memory size, overall architecture, and software support. In this paper, we explore two classes of commercial solutions. The first class involves high performance systems for embedded deep learning. In particular we tested Jetson TX2 and Jetson Nano. The second class is represented by smartphone devices.

Jetson TX2 is a System On Module (SOM) composed of two processors, namely a dual-core NVIDIA Denver2 and a quad-core ARM Cortex-A57; it provides a 8GB 128-bit LPDDR4 and integrated 256-core Pascal GPU. The device hosts a custom operative system derived from Ubuntu 16 that enables a high level of control of hardware resources. The entire Jetson TX2 module measures 5 cm $\times$ 9 cm.

Jetson Nano can be considered a constrained version of the Jetson TX2 SOM. It contains an ARM A57 quad-core, a 1.43 GHz CPU, and a Maxwell 128 core GPU equipped with 4 GB of RAM memory. This device also hosts a custom operative system derived from Ubuntu 16 that enables a high level of control of hardware resources. The entire Jetson Nano module measures 7 cm $\times$ 4.5 cm.

Table 1 summarizes the configurations for both devices. The columns specify the device name (Dev.), the configuration (Mode), the number of operative cores and the clock frequency for the Denver microprocessor, the number of operative cores and the clock frequency for the Cortex microprocessor, and the clock frequency of the GPU. The upper part of the table reports the five configurations of TX2, while the bottom two rows describe the configurations for Nano.

Nvidia proposes five optimized configurations of the hardware resources of Jetson TX2 that can be selected from a software interface. The five configurations are very different. The impact of the GPU clock frequency alone is likely to be more substantial than all the other configuration parameters together because most of the computationally intensive operations are carried by the GPU. The microprocessors mostly perform data fetching and host the operating system kernel. Accordingly, configuration 1 is expected to yield the lowest power consumption. Meanwhile, configuration 0 should lead to better throughput. Configurations 2, 3, and 4 should provide a trade-off between the two extremes. Nvidia proposes two optimized configurations of the hardware resources of Jetson Nano that can be selected from a software interface.

For both the devices, Nvidia provides a dedicated SDK and TensorRT for the optimization and quantization of standard neural network layers. It is important to note that the software engine uses native TensorFlow. Accordingly, almost all the layers are supported on these devices. This aspect allows deploying recent solutions that are not supported by most of the embedded systems relying on custom inference engines.

Both these devices provide valid solutions for the problem under analysis because they offer high computing power with small form factor. Power consumption might represent a critical issue in Jetson TX2. However, hardware measures will confirm that the battery supply is still an option for these devices, especially if inference rate is suitably managed. Similar observations hold for the heating. In fact, both devices reach high temperature values during inference. Therefore, when integrating these devices in portable systems care should be taken to avoid contact with user skins (e.g, encasing it within a hollow area of a prosthetic socket).

The second class of devices are the smartphone processors. Those computing units are less efficient than Nvidia devices for deep learning. However, they can represent a reliable solution due to the excellent ecosystems, composed of dedicated libraries, and high-level of integration with standard deep learning tools. In this paper we consider only smartphone processors, avoiding the use of dedicated GPUs and NPUs that are efficient but non-standard. Three different solutions have been tested, namely, Honor 10, Samsung Galaxy S8, and Samsung Galaxy A40, to demonstrate that the proposed architectures can be deployed to the systems with different processing capabilities.

The deployment on the devices was performed by the TF-lite library that supports the vast majority of the standard layers for deep learning. In addition, the suit allows adjusting the settings for post training quantization of the deployed model.

IV. EXPERIMENTAL SETUP

The individual components of the proposed solution as well as the overall system were experimentally evaluated as described in the sections below.

A. AFFORDANCE DETECTION TEST

The proposed hardware-aware approach to affordance detection was compared to the state-of-the-art methods adopted in robotics. Robotics applications served as the benchmark since in this context real-time affordance detection runs on embedded systems (which are however not necessarily wearable/portable). In addition, this paper used as the baseline for the segmentation architectures two of the latest approaches proposed in the literature.

More specifically, the experiments presented in [3] have been used as the benchmark. The experiments involved data from UMD Dataset [2], which contains around 30000 images and a total of 17 object categories (e.g., workshops objects, gardens objects, and daily kitchen objects). The images were captured by the authors using a Kinect camera on a table in a clutter-free setup.

As anticipated in Sec. III-A, the UMD dataset contains 7 affordance classes: grasp, cut, scoop, contain, pound, support and wrap-grasp. Such fine-grained classification is useful in robotics. In prosthetics, though, one may focus on a simplified problem, as the goal is to detect the graspable parts of an object. Therefore, the proposed affordance models were tested on two different setups. In the robotics setup, the original 7 affordance classes were utilized. In the prosthetics setup, all the classes apart from ''grasp'' and ''wrap-grasp'' were merged into a single class.

The affordance models V1_Se and V1_U relied on available pre-trained checkpoints [38]. The checkpoints refer to networks trained on the ADE20K dataset [39]. As no pre-trained checkpoints are available for the V3_LR affordance model, the architecture has been trained on ADE20K for 100k steps using Adam optimizer with learning rate of 0.0001 and batch size equal to 32.

All three architectures were fine-tuned by exploiting the UMD training set for 10K steps using Adam optimizer with learning rate of 0.0001 and batch size equal to 32. As per [3], the patches containing the objects were isolated from the images using the affordance masks. The experiments involved only the RGB images of this dataset and followed the original train and test split proposed in [2].

In this research, the solution that obtained the best results in [3] (namely AffordanceNet) was used as the reference. AffordanceNet achieved state-of-the-art performance in terms of accuracy and compared favorably with respect to the previous works [2], [16], [40]. The performance was measured using F_β^W [41], which is a well-established metric for pixel-wise classification tasks with the values in the range [0,1]. When $F_\beta^W = 1$, the prediction map is identical to the ground-truth, while $F_\beta^W = 0$ means that there is not a single match between the pixels of the prediction map and the ground-truth. This metric is commonly adopted in papers dealing with affordance detection in robotics [3].

The baseline solution representing the latest neural network architecture for segmentation were EfficientNet B0 [30] and BiSeNet [31]. EfficientNet was combined with Unet, hereafter named Eff_U. The pretrained version was downloaded from [42]. The pretrained version of BiSeNet (BiSe) was retrieved from [43]. Both models were fine-tuned following the same procedure described previously.

B. HARDWARE DEPLOYMENT TEST

Dedicated tests were designed to measure the performance of the affordance detection system, i.e., the embedded device supporting the real-time implementation of affordance detection (as per Sec. III-C). The setup focused on prediction, i.e., all the tests refer to trained affordance models that are deployed on the embedded platform. The tests covered two aspects separately: 1) characterization of the hardware deployment in terms of latency, memory consumption, and power consumption; 2) assessment of the effect of quantization on generalization performance.

1) DEPLOYMENT ON DEVICES

The first deployment pipeline involved the Jetson devices. The pipeline started from the trained model described using Keras API. The affordance model was converted in TensorFlow and frozen. A deployment ready version of the network architecture was built, thus removing all the software structures supporting the training phase. The frozen model was then optimized for Jetson TX2 and Jetson Nano using TensorRT tool [44]. This tool provides optimized implementation of common deep learning layers for Jetson devices. The output is again a TensorFlow frozen graph where the computed layers are replaced with optimized versions.

TensorRT can adopt different data sizes when deploying a network: standard floating-point representation (FP32), half-precision floating point (FP16) and 8-bit integer representation (INT8). The experiments were conducted with the FP16 format since this provides a good trade-off between accuracy and power consumption [45]. In addition, the results proved that FP16 is indeed sufficient to reach high frame rates in these high-end devices.

The Jetson TX2 development kit provided the hardware prototype for the experiments. The built-in Jetson Camera 5MP CSI module (with Omnivision OV5693) was used to capture the input images. The Jetson Nano toolkit was utilized for the second set of experiments; in this case, the toolkit was connected to a Raspicam module.

The code was developed in Python using CV2 module and TensorFlow utilities to measure memory usage and power consumption in combination with latency. The latency did not consider any batching strategy. Each frame was elaborated in real time when acquired.

Hardware measures were averaged over 20000 frames. Memory usage and power consumption were assessed using the Tegrastats utility [46]. In particular, power consumption was measured using the built-in sensors interfaced via I2C to monitor the input current to the SOC and to the GPU, respectively. The optimal clock configuration was set using Jetson_clock utility. Video and WiFi interfaces were always

disabled during the experiments. The tests were executed by running scripts through an SSH connection.

The test involving Jetson TX2 encompassed the three models (V1_Se, V1_U, and V3_LR) and the five available hardware configurations (as per Table 1). The test involving Jetson Nano focused on the V3_LR model; both available hardware configurations were analyzed (as per Table 1).

The second set of experiments considered the deployment on smartphones. The pipeline started from the Keras model converted into TFLite graph using Keras API. Similarly to TensorRT, TFLite supports different setups in terms of quantization. In this paper, INT8 quantization was considered. The host devices were an Honor10, a Samsung S8, and Samsung Galaxy A40. The model was deployed by using an Android application written in Java with a set of 20 images saved on the disk. The TFLite interpreter was set to run the inference on the processor core, avoiding dedicated units like GPUs or NPUs. The inference was performed on images stored into the device memory. The inference time only was measured in these experiments. In fact, quantities like memory usage and power consumption are heavily dependent from the operative system scheduling. As in the case of Jetson Nano, the V3_LR model was deployed.

2) EFFECTS OF QUANTIZATION

Post-training quantization performance depends heavily on the quantization approach. In fact, the same level of quantization can lead to different levels of accuracy depending on the operations performed by the inference engine. In this paper, the tests involved TFlite because it represents one of the mainstream solutions for post training quantization. The test followed the same experimental protocol used on the UMD dataset.

Quantization test considered 3 levels of quantization, namely, INT8, FP16 and FP32. The first two models were executed by using the TFLite interpreter. The FP32 version corresponded to the native Keras layer. According to TFLite documentation, the intermediate layers are anyway cast to floating point at runtime to avoid truncation errors. Accordingly, the major advantage is in storage size.

C. ASSESSMENT IN REALISTIC CONDITIONS

The third test was designed to characterize the performance of the affordance detector when the conditions mimicked those of prosthesis control, i.e., the envisioned application of the system. In this context, the camera will be mounted on a prosthesis moved by the user to target objects that he/she would like to grasp. Therefore, the affordance detection needs to accommodate different objects, illumination settings, background and viewing angle as well as the distortions caused by user movements. To simulate as much as possible such scenario, the assessment involved an experimenter holding the camera in his hand. The following subsections explain how the impact of illumination, background, objects, and viewing angle was evaluated.

1) IMPACT OF ILLUMINATION, BACKGROUND AND OBJECTS

The objects were placed on a flat surface. An experimenter held the camera and positioned it to the minimum distance at which the entire target object was contained in the image; to this aim, the user adjusted the distance based on the video streamed on a monitor. The camera was always placed directly above the objects during the experiments. Procedure 1 summarizes the procedure.

Procedure 1 Evaluation Protocol

Initialization

1. Select background and illumination settings.
2. Place the object under analysis on the background.

Evaluation

3. The user moves the camera on top of the object and adjust the framing to contain almost only the object under analysis using the video streaming on a monitor.
4. The affordance mask and the video stream are displayed.
5. Human annotators output the score based on the proposed rules.

The performance of the affordance detector was assessed by the experimenters (step 5 in Procedure 1). Since this was a dynamic test with a new set of objects, the information on the ground truth (i.e., annotated affordance masks) was not available. Hence, two annotators evaluated the classification quality by looking at the image and the corresponding mask in real-time. They eventually provided a score ranging from 0 to 3 by adopting the following rules:

$0 \Rightarrow$ the detector does not separate the object from the background. The prediction does not convey any useful information.

$1 \Rightarrow$ The object is segmented from the background, but the affordance information is not satisfactory.

$2 \Rightarrow$ The network segments the object from the background and identifies the majority of the affordance information, however the prediction is influenced by camera movements.

$3 \Rightarrow$ The network always segments satisfactorily the object from the background and the affordance information is consistent for all the frames analyzed during the test.

The tests were performed in an indoor environment. Two illumination settings were considered. The light environment was illuminated by 14 neons and the approximated light intensity was 75 lux. In the dark one, all the neons were switched off and the approximated light intensity was 3 lux.

Four different settings were adopted for the background: red fabric with light reflection close to 0; blue fabric with light reflection close to 0; black, reflective piece of plastic; reflective wooden table.

Only the V1_U model was employed for this test, as it proved to be the most reliable option given the outcomes of all the experimental sessions. In this case, the model was fine-tuned by using the IIT dataset [16] for 10K steps. The IIT dataset includes multiple backgrounds and illumination settings; thus this dataset was selected to enhance the overall robustness of the model.

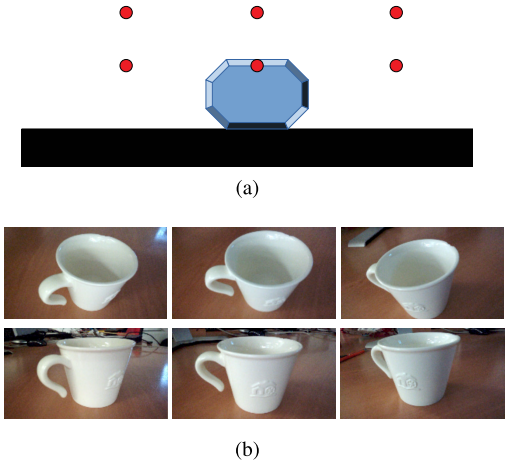

FIGURE 4. Framing positions schema employed during experiments (a) and an example containing the captured frames (b).

2) IMPACT OF VIEWING ANGLE

The object under test was placed on a table; illumination conditions were set to light. An experimenter moved the camera to frame the object from six different viewpoints. Figure 4(a) shows the framing positions using red dots. Figure 4(b) depicts an example of the six frames captured from the different positions. To minimize variability in capturing across trials, the experimenter always placed the feet in the same position to adjust the camera. Then by holding the camera in the hand, he positioned it to consistently achieve the desired viewing angle. Despite careful placement, this procedure does not guarantee exactly the same position across recordings. Nonetheless, this is not necessarily a drawback since the same effect is to be expected in the future functional applications, when the camera is embedded in a prosthesis moved by the user. A total of six objects were used in this test.

For each position, two human annotators assigned a score for the affordance mask, following the procedure detailed above. Again, only the V1_U model was employed as affordance detector. As a side effect, this test also stressed the role of background and light direction. In fact, the six frames included in 4(b) prove that light direction and the corresponding reflection phenomena changed significantly when varying the framing position. In addition, the overall structure of the background changed as the framing position was varied.

V. RESULTS

A. AFFORDANCE DETECTION

Table 2 shows the summary results of the experiments described in Sec. IV-A for the robotics setup (i.e., 7 classes); see Fig. 2 for examples. The table compares the performance of the benchmark (AffordanceNet) to that of the three hardware-aware models (V1_Se, V1_U and V3_LR). For each affordance class, the quantity F_β^W is reported. As a major advantage, it has been employed in the previous papers with the same experimental setup, and it therefore allows a fair comparison with previous works. Moreover, under each class name the table gives the total percentage of pixels originally labeled with that class.

TABLE 2. Results for comparison with benchmark. All results refer to F_β^W.

Model	grasp	cut	scoop	cont.	pound	sup.	wrap
	24.4	10.0	9.0	29.7	3.7	7.0	16.0
Aff. [3]	0.73	0.76	**0.79**	0.83	**0.83**	**0.82**	0.81
Eff_U	0.77	0.72	0.64	0.94	0.82	0.53	0.86
BiSe	0.79	0.87	0.66	**0.95**	0.79	0.52	0.93
V1_Se	0.75	0.88	0.57	0.93	0.64	0.56	0.92
V1_U	**0.80**	**0.89**	0.50	0.92	0.57	0.53	**0.94**
V3_LR	0.67	0.75	0.09	0.79	0.42	0.49	0.89

The experiments proved that the V1-based architectures led to excellent results in the classification of grasp, cut, contain and wrap-grasp classes; both V1_Se and V1_U were able to outperform the benchmark. However, a substantially lower F_β^W compared to the benchmark was achieved for scoop, pound and support. Also solutions based on Eff_U and BiSe show a similar trend, although the drop is less evident due to the better feature extraction capabilities. Such outcome is not surprising if one considers that those three classes are less represented in the dataset. In fact, only 3.7% of the pixels were marked as pound. Therefore, the proposed solutions might be less suitable for unbalanced distribution of the data across classes. The performance drop for less represented classes is even more evident in V3_LR. Many solutions to tackle this problem exist, starting from the use of class weighted cost function, moving to Online Hard Negative Mining [47]. However, addressing this problem is out of the scope of the present paper.

The outstanding generalization performance obtained by AffordanceNet over the seven classes are also due to the multi-task training. The network was trained simultaneously both for affordance detection and object detection. Accordingly, the feature set is less subject to over-fitting. However, this advantage comes with a prohibitive computational cost when considering the constraints related to the use in wearable embedded systems.

Table 3 gives the summary results of the experiments for the prosthetics setup, where the number of classes was decreased to 3. The table compares the performance of the three hardware-aware models (V1_Se, V1_U and V3_LR) with Eff_U and BiSe. Again, for each affordance class the quantity F_β^W is reported. The results confirmed that the proposed models are reliable candidates for the simplified problem, where the aim is to distinguish graspable parts of the image from those that are non-graspable. It is worth stressing that the worst F_β^w score was 85 for "grasp" class using V3_LR architecture. However, even this result is better than the state-of-the-art results for grasping, as per Table 2. In addition, the better feature extraction capabilities of Eff_U and BiSe did not yield better performance in this simplified problem.

TABLE 3. Experimental results for prosthetic problem. All results refer to F_β^W.

Model	background	grasp	non-grasp
V1_Se	0.99	0.86	0.94
V1_U	0.99	0.90	0.94
V3_LR	0.99	0.85	0.93
Eff_U	0.99	0.85	0.94
BiSe	0.99	0.86	0.90

Indeed, the memory requirements and the number of floating point operations (flops) for a forward phase of the hardware-aware models are significantly lower compared to those characterizing AffordanceNet. In this regard, the second column in Table 4 reports the number of parameters for the three proposed solutions, two recent baseline solutions (*Eff_U* and *BiSe*), an architecture using the Vgg network as backbone and SegNet as meta-architecture (*Vgg_Seg*), an architecture using the Vgg network as backbone and UNet as meta-architecture (*Vgg_U*), and the AffordanceNet model implemented on the WalkMan robot [3]. Since AffordanceNet solves multiple tasks simultaneously, the model is inevitably more demanding in terms of memory requirements. The results confirm that the recent solutions *Eff_U* and *BiSe* were more efficient than the solution based on Vgg. However, V1_Se and V1_U architectures have two times less parameters. In fact, by using MobileNets as backbone one can further reduce the number of parameters while achieving satisfactory performance in terms of classification (see Table 2 and Table 3). Finally, the *V3_LR* architecture uses less than 200k parameters. Even considering the differences in the application, one can easily appreciate that, in terms of memory, V1_Se requires 24 times less parameters than AffordanceNet. In the case of V3_LR, the decrease is 750 times. Accordingly, these architectures are more suitable for embedded implementations. The third column of Table 4 reports the number of flops required for one inference phase. This value is omitted for AffordanceNet because the number of flops for a prediction depends on input images. In fact, one image with multiple regions of interest requires a larger number of computations. The other tested models, however, use a fixed number of computations. The results highlight that *Eff_U* uses a small number of flops with respect to models with a similar number of parameters. The two hardware-aware affordance models based on MobilenetV1 use a slightly larger number of flops compared to *Eff_U*. However, when considering the combination of the number of parameters and flops, the proposed hardware-aware affordance models remain favorable. In addition, the architecture based on MobileNetV3 is the most appealing solution also in terms of flops count.

B. HARDWARE DEPLOYMENT

1) JETSON DEVICES

Experimental results for Jetson TX2 are presented using four figures. Figure 5 reports the average latency (in ms) introduced by the CNN for a single inference. Figure 6 reports the average memory consumption (in MB) for a single inference. Figures 7 and 8, respectively, report the average power consumption of the SOM and GPU, respectively. The SOM consumption takes into account the GPU consumption. In each figure, the x-axis gives the five hardware configurations of Jetson TX2, as per Table 1. For each configuration, the measures obtained with the three proposed solutions are shown. As described in Sec. IV-B, the average results over 20000 frames are reported. The standard deviation was negligible and therefore it is not shown.

Figure 5 shows that V3_LR introduces an important advantage in terms of speed. Regardless of the configuration, it is always the fastest solution. In configuration 0, the average inference time for V3_LR is 21 ms, which roughly corresponds to 48 FPS. The other two models also achieve small latency in configuration 0, with the worst results of 26 ms (38 FPS) for V1_Se. Importantly, the overall worst result of 38 ms was scored by V1_U.

The results in Fig. 6 demonstrate that V3_LR is also the best model in terms of memory consumption. V1_Se always allocates more memory than the other two models, even if this model has less parameters than V1_U. Actually,

TABLE 4. Number of parameters and flops for one inference phase of the deployed networks.

Model	parameters	flops
AffordanceNet [3]	146,654,400	n.a.
Eff_U	10,115,791	7,274,700,546
BiSe	26.340.900	10,093,952,344
Vgg_Seg	11,552,264	152,224,917,126
Vgg_U	12,326,408	164,455,507,590
V1_Seg	5,544,840	7,584,227,590
V1_U	6,318,984	10,358,759,686
V3_LR	195,768	642,135,722

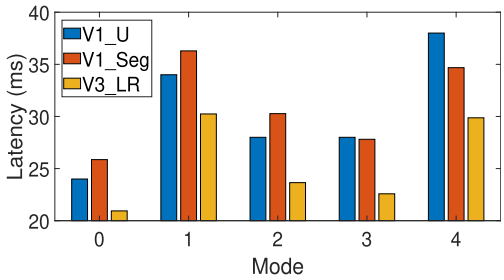

FIGURE 5. Average latency measured for a single inference.

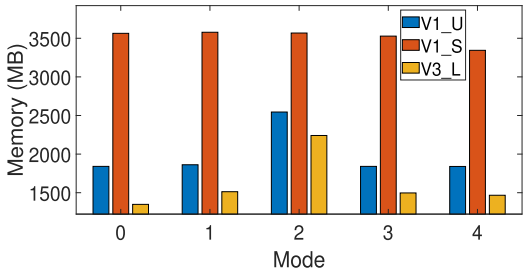

FIGURE 6. Average memory consumption measured for a single inference.

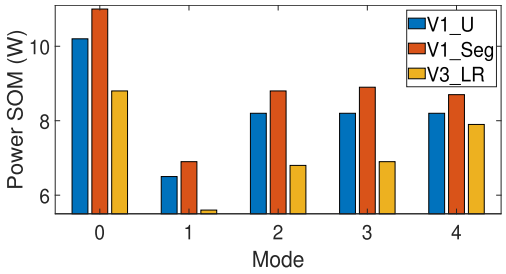

FIGURE 7. Average power consumption of the SOM for a single inference.

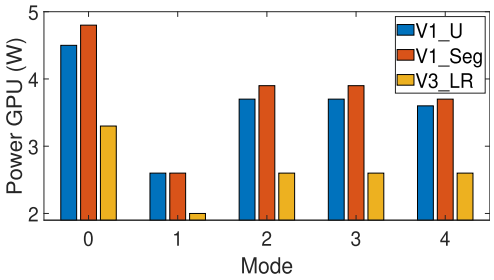

FIGURE 8. Average power consumption of the GPU for a single inference.

TABLE 5. Measured hardware performance for Jetson Nano.

HW setup	Latency (ms)	Memory (Mb)	Power SOM (W)	Power GPU (W)
Max-N	68	1782	5.2	1.6
5W	104	1870	3.6	0.8

run-time memory consumption is not only a function of the number of parameters, but of the overall network architecture. In any case, all the models are compliant with the hardware resources available in Jetson TX2. It is worth noting that the measure reported in this plot is the difference between the memory usage before the start of the script and the average memory usage during the inference. In this way, the offset due to sub-tasks run by the operative system is removed.

Figures 7 shows that the overall power consumption ranges from 5 to 11 Watts, while 7W are consumed by the GPU. As expected, power consumption in Jetson TX2 is higher. Considering 2 LiOn batteries with 3.6V output and 2900 mAh capacity, used in series, the continuous prediction span-time ranges from 2 to 4 hours. A single prediction stands for 21 ms, and therefore, more than 680K predictions are possible within the span-time?

Table 5 shows the experimental outcomes for the test involving the Jetson Nano. The first column of the table gives the hardware configurations (see Table 1). Columns from two to five give, respectively, the average latency, the average memory usage, the average power consumption of the SOM, and the average power consumption of the GPU. As above, all the values are for single inference averaged over 20,000 frames; standard deviation was negligible.

The latency, even in the case of 5W-configuration, is 104 ms. This corresponds to a frame rate of 10 FPS.

TABLE 6. Measured latency on smartphones.

Model	Honor10	S8	A4
Time (ms)	225	278	285

Memory consumption is bounded by 2 GB for both configurations, and the power consumption goes to up-to 3.6 Watts in the best case. Considering again the same set of batteries, the continuous prediction time increases to 5.8 hours. It is important to note that continuous prediction is not required for real-time applications. Accordingly, the effective lifetime could be much longer.

2) SMARTPHONES

Table 6 reports the inference time scored by the V3_LR model when deployed on the three smartphones. The first and the second rows show the name of the smartphone and the average time in ms, respectively.

The inference time is higher than that achieved on dedicated GPUs like Jetson devices. In practice, these processors cannot guarantee continuous prediction; the affordance surfaces can be estimated up to three times per second. Nevertheless, 3 FPS are still sufficient for many applications, as explained in Sec. 6 (e.g., prosthesis grasping). The average measured power consumption was in the order of mWatts [48]. As a consequence, the heating problems are highly reduced while the battery life time is improved.

3) QUANTIZATION EFFECTS

Table 7 shows the effect of quantization on UMD benchmark. The first column identifies the quantization type. The following columns report the F_β^W values for the classes under analysis. The latest column shows the size in KB of the model saved on disk. Experimental results showed that the difference in terms of F_β^W is negligible. A possible explanation is that the network performs a relatively small number of multiply and accumulate operations, reducing significantly the propagation error. In addition, the activation functions employed have been designed to limit the truncation error with small number of bits representations. Therefore, the quantization can reduce the model size without significant decrease in performance.

C. ASSESSMENT IN REALISTIC CONDITIONS

1) IMPACT OF ILLUMINATION, BACKGROUND AND OBJECTS

Table 8 reports the scores registered in this test. The first column shows the objects involved in the experiment. Columns from two to nine refer to the different backgrounds. For each background, the scores are reported for the two illumination

TABLE 7. Experimental results for quantization.

Quant.	back.(F_β^W)	grasp(F_β^W)	non-grasp(F_β^W)	Size(KB)
INT8	0.988	0.848	0.929	238
FP16	0.988	0.850	0.930	421
FP32	0.988	0.850	0.930	954

TABLE 8. Outcomes of experiments assessing the impact of illumination, background and objects.

Letters 'd' and 'l' refer to dark and light illuminations, respectively.

Objects	Red		Blue		Black		Table	
Setting	l	d	l	d	l	d	l	d
Hammer	3	3	3	3	3	2	3	3
Pan	2	2	3	2	2	2	3	2
Bowl	1	3	3	3	1	3	1	1
Spat. 1	3	2	2	1	2	2	2	1
Spat. 2	3	3	3	2	3	1	3	3
Spat. 3	3	3	2	1	3	1	3	3
Scoop	1	3	1	2	2	1	1	3
Knife	1	1	1	1	1	1	1	1
Mug	1	2	3	2	1	2	1	2
Screwd.	1	1	1	1	1	1	0	1
Racket	2	2	2	2	2	2	2	2
Pen	3	3	3	3	2	2	2	2
Avg.	2.00	2.33	2.25	1.92	1.92	1.66	1.82	2.00
Std	0.95	0.78	0.87	0.80	0.79	0.65	1.03	0.85

settings. The last two rows refer, respectively, to the average score over the objects per {background, illumination} and the corresponding standard deviation.

The average scores are close to 2 in all the conditions except for the setup {black background, dark illumination}, which was expected to be the most critical scenario. Therefore, in general the network successfully distinguished the objects and associated affordances, even if a component of noise and some distortion existed in the segmentation mask. The table surface in combination with light illumination was also a challenging condition, because the table surface was textured and reflective.

Figure 9 reports a set of representative examples across conditions. Each panel shows a test image and the corresponding affordance map (3 classes). The images captured under dark illumination are characterized by significant noise. The sub captions report the background and lighting conditions, associated with the score assigned to the corresponding stream between the brackets. It is important to recall that the score was not assigned using the single snapshot but a continuous stream.

Several insights about the network behavior can be obtained from these representative examples. In most cases, the network successfully segments the object and separates the graspable (blue) from the non-graspable part (green), even in some rather challenging conditions. However, the knife is an example where the network could segment the object from the background, but the functional parts were not retrieved correctly. The system did not separate the handle from the blade; instead, the entire object was labeled as graspable. A possible explanation is that the blade of the knife is a small metallic rectangle, which is therefore similar to a handle of metallic spoons and forks. Note that this is still appropriate for the application in prosthetics since the graspable surface is in fact the relevant part of the object. Further encouraging results are demonstrated by the cup and the pan example. In the former, the networks were able to retrieve the handle despite only a fraction of it is seen in the image. In the latter, the functional (non-graspable) part of the object was affected by substantial noise, but the handle was still extracted almost completely.

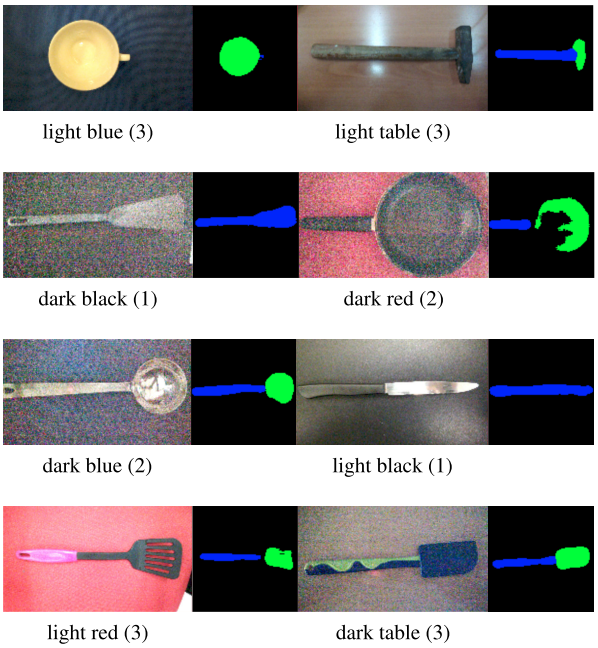

FIGURE 9. Examples of the real time performance with different configurations of illumination and background. Each one of the sub images reports a test and the corresponding affordance map (black - background, blue - graspable and green - non-graspable segments).

TABLE 9. Outcomes of experiments for angle. Sc. indicates the framing positions. Sc.A the average result.

Object	sc.1	sc.2	sc.3	sc.4	sc.5	sc.6	sc.A
mug	3	3	3	2	2	2	2.5
spoon	3	3	3	3	1	0	2.2
mug 2	1	2	2	3	2	2	2
hammer	3	3	1	3	1	3	2.3
pan	3	3	2	3	3	3	2.8
racket	3	2	1	2	1	1	1.7
fork	3	1	1	3	3	3	2.3

2) IMPACT OF VIEWING ANGLE

The results are reported in Table 9. The objects are in the rows. The column reports the scores for different framing as well as the average results. The only object with an average score lower than 2 is the racket. Importantly, this object has a pronounced texture and it does not belong to the set of daily life objects (hence it is less relevant).

VI. DISCUSSION

We have developed, deployed and tested a novel solution for affordance detection based on deep learning that is convenient for applications in embedded systems with limited resources. The experimental results confirmed that the proposed hardware-aware models provide high accuracy, good online performance on embedded devices, and reliable detection under realistic conditions (objects, background and illumination, and viewing angles). The deployment was

performed on several hardware solutions and pros and cons were analyzed. This study paves the way to the setup of deployment strategies where given a set of constraints it is possible to retrieve the best solution by performing an optimization procedure [49]. In fact, such strategies may represent a critical step when focusing on specific applications of the proposed system.

One of the envisioned applications for the developed systems is in the semi-autonomous prosthesis control. The affordance detection enables extracting the graspable part of an object, thereby identifying the part that should be further analyzed to decide about an appropriate grasping strategy. Such information would substantially improve the capabilities of a smart robotic limb. For instance, it would allow a prosthesis to preshape and grasp an object for the handle, whenever a handle is available (Fig. 9). The systems presented in literature [6], [14] estimate the object properties (size and shape) from the point cloud. With affordance detection, the object point cloud could be reduced to only those points that correspond to a graspable surface, thereby improving the estimate. The input source of the system proposed in the present study was an RGB camera. A possible alternative solution could be based on a LIDAR depth sensor. Combining the two input sensors to improve the estimation of affordance classes and/or grasp parameters would be a relevant future task in this line of work.

The proposed hardware-aware models for affordance detection perform worse than the benchmark in the case of fine-grained classification of objects parts (9 classes). However, all three models achieved satisfactory results when the number of classes was reduced to that relevant for prosthetic applications (graspable versus non graspable object parts). Accordingly, these models establish a better trade-off between computing cost and accuracy compared to previous approaches when the system is customized to a specific context (prosthetics versus robotics). Extensive experiments on the real time embedded system with different objects and changes in background, lighting and viewing angle were performed to remove eventual biases about the performance measured using offline benchmarks. The test confirmed that the system could correctly detect the graspable parts in most cases, even under challenging conditions.

Semi-autonomous control using computer vision can allow simple control of advanced bionic limbs, thereby improving user experience and potentially decreasing rejection rates. In addition, recent studies in conventional human-machine interfacing also propose to equip a prosthesis with advanced sensors that might require complex data processing; e.g., inertial measurement units [50], electronic skins [51], musculoskeletal modeling [52], high-density EMG [53]. To support these developments as well as general user demands for better control and functionality [54], modern prostheses will need to integrate improved processing capabilities (see [5]). The platforms considered in the present study (and similar solutions) can be regarded as the potential candidates for such a high-end prosthesis controller. Hardware deployment on high-end embedded systems confirmed that these devices can perform the inference phase in real time with power consumption, and dimension acceptable for wearable applications. If integrated into a prosthesis, these systems could implement the complete semi-autonomous pipeline including all other computer vision and signal processing functions (in addition to affordance detection). Using a smartphone as an extra processing module can be especially attractive. Several state of the art prostheses (Michelangelo from Otto Bock, i-Limb from Ossur) are equipped with a wireless interface (e.g., Bluetooth link) which allows connecting to a smartphone to offload data intensive processing. Accordingly, in the proposed scenario, the embedded system hosted in the prosthesis can indeed work locally but it could also connect from time to time to the mobile network to use additional resources, as in the paradigm of cloud computing. Such setup would enable online training, for example.

A promising future direction is to deploy the proposed hardware-aware models to more constrained devices, e.g. GAP-8 and STM32. The preliminary experiments revealed two major bottlenecks. First, the inference engine of these devices requires a low-level implementation of each layer of the model. During our tests, we discovered that four of the key operations of V3_LR were not supported by the SDKs of both STM32 and GAP-8. Secondly, even when using the smallest model among the proposed ones, i.e. V3_LR, and changing all the unsupported operations to the closest supported ones, leading to an important decrease in generalization performance, the computational cost was still too high for STM32. In the case of GAP-8, we did not succeed in updating the model in order to make it compliant with the SDK.

If a suitable control strategy that evaluates only limited number of frames is implemented then the battery lifetime would be sufficient for an entire day of usage. To this aim, we can follow the approach of ''traded autonomy'' that was already used in literature for semi-autonomous prosthesis control. In this scheme, to grasp an object, the subject aligns the prosthesis camera towards the object and ''triggers'' the automatic control by generating a specific myoelectric signal (e.g., hand opening command). The computer vision module makes one or more snapshots, analyses the images (e.g., estimates the affordance class), selects the desired grasp parameters and preshapes the hand. After that, the control switches back to manual operation so that the user can close the hand to generate the desired grasping force and manipulate the object, as desired. This would substantially improve the control and user's experience.

The next step in this work is to integrate the computer vision system with a prosthesis, and assess the performance in functional tasks. To this aim, the camera will be placed on the prosthesis (as in [5], [21]), e.g., on the top of the palm of the prosthetic hand or on the socket, to have an unobstructed view in front of the hand. The user will be instructed to approach the target object and then trigger the system. In present tests, the experimenter performed the aiming with the assistance

of visual feedback (camera stream on the monitor). Similar information could be delivered to a prosthesis user via wearable glasses [7]. Otherwise, with some practice the user could be trained to properly aim even without any feedback, as demonstrated in a recent work [5]. Nevertheless, in future functional tests with the camera on the prosthesis, the system performance might depend on the exact placement of the camera as well as on the skills of an individual user regarding the accuracy and steadiness in ''aiming'' at the target object.

VII. CONCLUSION

This paper proposes a hardware-aware affordance detection framework for embedded devices with limited computational resources. This work starts by selecting a state-of-the-art solution from robotics and then simplifies the overall framework to cope with constrains and requirements of a specific application (prosthesis control). Three hardware-aware deep learning architectures were proposed. Experimental results confirmed that these models tackle the affordance problem with high accuracy. In addition, the inference system was implemented on a Jetson TX2 and Nano, achieving real time performance with a power consumption manageable from a smartphone-like battery. Two experimental assessments confirmed that the proposed hardware-aware models are reliable and robust across many combinations of background, illumination, framing conditions and objects. The main outcomes of the paper are the following:

- The development of hardware-aware deep learning architectures yielding compact models for affordance detection that can be deployed to edge devices for real-time applications. To the best of the authors' knowledge, the literature does not provide approaches leading to the same result.
- The characterization of the affordance detection models on different classes of embedded devices that could be integrated in novel smart prostheses equipped with deep learning.
- A comprehensive experimental assessment of the deployment pipeline in realistic conditions to evaluate the performance of the affordance detection when the conditions mimic those that will be encountered in semi-automatic prosthesis control.

APPENDIX A
BACKBONE: MobileNets

MobileNets architectures represent the state-of-the-art solution for deployment of CNNs on embedded devices. In the original paper presenting the first version of MobileNets [33], V1, the authors introduced the concept of depth-wise convolution. In practice, standard convolution was replaced by a set of channel-wise convolutions and a 1–D convolution; such arrangement significantly lowered the computational cost. Given M inputs of size $H \times W \times C$ and a convolutional layer characterized by N kernels of size $D_k \times D_k \times C$, the computational cost C_{sc} of standard convolution is:

$$C_{sc} = H \times W \times M \times N \times D_k \times D_k. \quad (1)$$

Conversely, when using depth-wise convolution the cost C_{DSC} becomes

$$C_{DSC} = H \times W \times M \times (D_k^2 + N) \quad (2)$$

which is markedly smaller than (1). The second version of MobileNets, V2 [34], introduced the linear bottleneck and the inverted residual structure to enhance data representation while maintaining low dimensional spaces. Finally, the third version, V3 [35], enriched data representation capability introducing also lightweight attention modules based on squeeze and excitation into the bottleneck structure [55]. In addition, the architecture adopted new hardware-aware non-linearity functions.

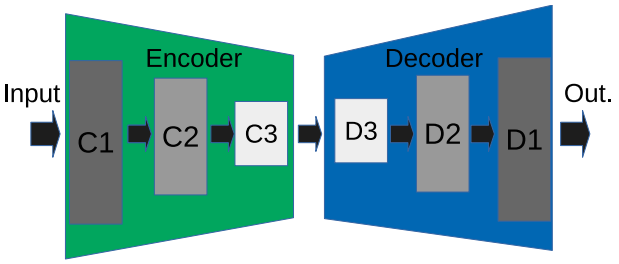

FIGURE 10. Example of SegNet meta architecture.

APPENDIX B
META-ARCHITECTURES FOR IMAGE SEGMENTATION

In this paper three meta-architecture for image segmentation are considered: SegNet [36], Unet [37] and Lite Reduced design of the Atrous Spatial Pyramid Pooling module (LR-ASPP) [35]. SegNet is an encoder–decoder architecture; its structure is schematized in Fig. 10 using a simple example with 3 layers. The encoder, marked in green in the scheme, computes features using a deep network, i.e., a sequence of convolution layers. The decoder, marked in blue in the scheme, localises patterns in the image. In the decoder, deconvolutions progressively reduce the number of feature maps. The feature maps become larger and wider, until the last up-convolution produces the segmented image, i.e., the image where each pixel corresponds to a class. In general, the encoder relies on a given architecture as backbone to extract features. In the original paper of SegNet [36] authors employed VGG as backbone.

UNet [37] is also an encoder–decoder architecture; its structure is schematized in Figure 11. Again, the encoder exploits a backbone to extract features. However, the feature maps generated by both the low levels and the intermediate levels of the deep architecture are also forwarded to the decoder to avoid loss of information. A complementary approach is adopted in the design of the decoder. Finally, the architecture uses a 1×1 convolution to process the feature maps, generating a segmentation map and thus categorising each pixel of the input image. Notably, both SegNet and

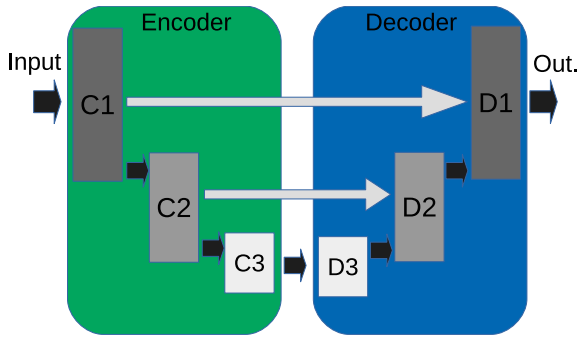

FIGURE 11. Example of UNet architecture.

TABLE 10. Summary of the computational cost of MobileNet V3 LR_ASPP.

Layer (type)	Output Shape	Param #
Input_1	(224, 224, 3)	0
Conv2d_1	(224, 224, 16)	448
Conv2d_2	(224, 224, 16)	272
Depthwise_conv2d_1	(112, 112, 16)	160
Dense_1	(16)	272
Dense_2	(16)	272
Multiply_1	(112, 112, 16)	0
Conv2d_3	(112, 112, 16)	272
Conv2d_4	(112, 112, 72)	1224
Depthwise_Conv2d_2	(112, 112, 72)	720
Conv2d_5	(112, 112, 24)	1752
Conv2d_6	(112, 112, 88)	2200
Depthwise_Conv2d_3	(112, 112, 88)	880
Conv2d_7	(112, 112, 24)	2136
Add_1	(112, 112, 24)	0
Conv2d_8	(112, 112, 96)	2400
Depthwise_Conv2d_4	(56, 56, 96)	2496
Dense_3	(96)	9312
Dense_4	(96)	9312
Multiply_2	(56, 56, 96)	0
Conv2d_9	(56, 56, 40)	3880
Conv2d_10	(56, 56, 240)	9840
Depthwise_Conv2d_5	(56, 56, 240)	6240
Dense_5	(240)	57840
Dense_6	(240)	57840
Multiply_3	(56, 56, 240)	0
Conv2d_11	(56, 56, 40)	9640
Add_2	(56, 56, 40)	0
Conv2d_26	(56, 56, 128)	5248
Conv2d_27	(1, 1, 128)	5248
Bilinear_up_sampling2d_1	(56, 56, 128)	0
Multiply_10	(56, 56, 128)	0
Bilinear_up_sampling2d_2	(112, 112, 128)	0
Conv2d_29	(112, 112, 4)	516
Conv2d_28	(112, 112, 4)	100
Add_7	(112, 112, 4)	0

UNet avoid fully-connected layers. Hence, the number of parameters involved in these models is relatively small.

Finally, LR_ASPP is a meta-architecture proposed in the original paper of MobileNet V3 [35]. It consists of three main branches that elaborate features from two different levels of the backbone architecture. In particular, two branches process the highest level of the backbone architecture; a third branch elaborates low-level features. This custom architecture proved effective in finding a trade-off between computation cost and generalization performance.

Table 10 summarizes the characteristics of MobileNet V3 LR_ASPP architecture. The three columns show, respectively, the layer name, the output shape, and the number of network parameters for the layer. It is worth noting that the table does not provide a complete topological description of the architecture.

ACKNOWLEDGMENT

Code available on the authors' web site: https://github.com/SEAlab-unige.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.
[2] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 1374–1381.
[3] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Object-based affordances detection with convolutional neural networks and dense conditional random fields," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 5908–5915.
[4] J. Chen and X. Ran, "Deep learning with edge computing: A review," *Proc. IEEE*, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.
[5] G. Ghazaei, A. Alameer, P. Degenaar, G. Morgan, and K. Nazarpour, "Deep learning-based artificial vision for grasp classification in myoelectric hands," *J. Neural Eng.*, vol. 14, no. 3, Jun. 2017, Art. no. 036025.
[6] M. Markovic, S. Dosen, C. Cipriani, D. Popovic, and D. Farina, "Stereovision and augmented reality for closed-loop control of grasping in hand prostheses," *J. Neural Eng.*, vol. 11, no. 4, Aug. 2014, Art. no. 046001.
[7] M. Markovic, H. Karnal, B. Graimann, D. Farina, and S. Dosen, "GLIMPSE: Google glass interface for sensory feedback in myoelectric hand prostheses," *J. Neural Eng.*, vol. 14, no. 3, Jun. 2017, Art. no. 036007.
[8] I. Vujaklija, D. Farina, and O. C. Aszmann, "New developments in prosthetic arm systems," *Orthopedic Res. Rev.*, vol. 8, p. 31, Jul. 2016, doi: 10.2147/ORR.S71468.
[9] N. Parajuli, N. Sreenivasan, P. Bifulco, M. Cesarelli, S. Savino, V. Niola, D. Esposito, T. J. Hamilton, G. R. Naik, U. Gunawardana, and G. D. Gargiulo, "Real-time EMG based pattern recognition control for hand prostheses: A review on existing methods, challenges and future implementation," *Sensors*, vol. 19, no. 20, p. 4596, 2019.
[10] D. Yang, Y. Gu, N. V. Thakor, and H. Liu, "Improving the functionality, robustness, and adaptability of myoelectric control for dexterous motion restoration," *Exp. Brain Res.*, vol. 237, no. 2, pp. 291–311, Feb. 2019.
[11] *Ottobockus*. Accessed: Sep. 3, 2021. [Online]. Available: https://www.ottobockus.com/prosthetics/upper-limb-prosthetics/solution-overview/myo-plus/myo-plus.html
[12] *Coaptengineering*. Accessed: Sep. 3, 2021. [Online]. Available: https://coaptengineering.com/
[13] J. W. Sensinger and S. Dosen, "A review of sensory feedback in upper-limb prostheses from the perspective of human motor control," *Frontiers Neurosci.*, vol. 14, p. 345, Jun. 2020.
[14] M. Markovic, S. Dosen, D. Popovic, B. Graimann, and D. Farina, "Sensor fusion and computer vision for context-aware control of a multi degree-of-freedom prosthesis," *J. Neural Eng.*, vol. 12, no. 6, Dec. 2015, Art. no. 066022.
[15] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *Int. J. Robot. Res.*, vol. 34, nos. 4–5, pp. 705–724, 2015.
[16] A. Nguyen, D. Kanoulas, D. G. Caldwell, and N. G. Tsagarakis, "Detecting object affordances with convolutional neural networks," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 2765–2770.
[17] A. Roy and S. Todorovic, "A multi-scale CNN for affordance segmentation in RGD images," in *Proc. Eur. Conf. Comput. Vis.* Cham, Switzerland: Springer, 2016, pp. 186–201.
[18] J. Sawatzky, A. Srikantha, and J. Gall, "Weakly supervised affordance detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2795–2804.
[19] L. Seminara, P. Gastaldo, S. J. Watt, K. F. Valyear, F. Zuher, and F. Mastrogiovanni, "Active haptic perception in robots: A review," *Frontiers Neurorobotics*, vol. 13, p. 53, Jul. 2019.
[20] S. Došen and D. B. Popović, "Transradial prosthesis: Artificial vision for control of prehension," *Artif. Organs*, vol. 35, no. 1, pp. 37–48, Jan. 2011.
[21] S. Došen, C. Cipriani, M. Kostić, M. Controzzi, M. C. Carrozza, and D. B. Popović, "Cognitive vision system for control of dexterous prosthetic hands: Experimental evaluation," *J. Neuroeng. Rehabil.*, vol. 7, no. 1, p. 42, 2010.

[22] L. T. Taverne, M. Cognolato, T. Bützer, R. Gassert, and O. Hilliges, "Video-based prediction of hand-grasp preshaping with application to prosthesis control," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 4975–4982.
[23] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural Comput.*, vol. 29, no. 9, pp. 2352–2449, Sep. 2017.
[24] Y. He, R. Kubozono, O. Fukuda, N. Yamaguchi, and H. Okumura, "Vision-based assistance for myoelectric hand control," *IEEE Access*, vol. 8, pp. 201956–201965, 2020.
[25] B. Zhong, H. Huang, and E. Lobaton, "Reliable vision-based grasping target recognition for upper limb prostheses," *IEEE Trans. Cybern.*, early access, Jun. 10, 2020, doi: 10.1109/TCYB.2020.2996960.
[26] E. Ragusa, C. Gianoglio, R. Zunino, and P. Gastaldo, "Data-driven video grasping classification for low-power embedded system," in *Proc. 26th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Nov. 2019, pp. 871–874.
[27] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1520–1528.
[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556
[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[30] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," 2019, *arXiv:1905.11946*. [Online]. Available: http://arxiv.org/abs/1905.11946
[31] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang, "BiSeNet: Bilateral segmentation network for real-time semantic segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 325–341.
[32] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen, and N. Sang, "BiSeNet V2: Bilateral network with guided aggregation for real-time semantic segmentation," 2020, *arXiv:2004.02147*. [Online]. Available: http://arxiv.org/abs/2004.02147
[33] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*. [Online]. Available: http://arxiv.org/abs/1704.04861
[34] 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.*, Jun. 2018, pp. 4510–4520.
[35] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, H. Adam, and Q. Le, "Searching for MobileNetV3," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1314–1324.
[36] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.
[37] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proc. Int. Conf. Med. Image Comput. Comput.-Assisted Intervent.* Cham, Switzerland: Springer, 2015, pp. 234–241.
[38] *Divamgupta*. Accessed: Sep. 3, 2021. [Online]. Available: https://github.com/divamgupta/image-segmentation-keras
[39] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ADE20K dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 633–641.
[40] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2017.
[41] R. Margolin, L. Zelnik-Manor, and A. Tal, "How to evaluate foreground maps," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 248–255.
[42] *Qubvel*. Accessed: Sep. 3, 2021. [Online]. Available: https://github.com/qubvel/segmentation_models
[43] *Kirilcvetkov92*. Accessed: Sep. 3, 2021. [Online]. Available: https://github.com/kirilcvetkov92/Semantic-Segmentation-BiSeNet
[44] *Tensorrt*. Accessed: Sep. 3, 2021. [Online]. Available: https://developer.nvidia.com/tensorrt
[45] E. Ragusa, C. Gianoglio, R. Zunino, and P. Gastaldo, "Image polarity detection on resource-constrained devices," *IEEE Intell. Syst.*, vol. 35, no. 6, pp. 50–57, Nov. 2020.
[46] *Jetson*. Accessed: Sep. 3, 2021. [Online]. Available: https://docs.nvidia.com/jetson/archives/l4t-archived/
[47] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 761–769.
[48] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proc. Annu. Tech. Conf. (USENIX)*, Boston, MA, USA, vol. 14, Jun. 2010, p. 21.
[49] S.-I. Mirzadeh and H. Ghasemzadeh, "Optimal policy for deployment of machine learning models on energy-bounded systems," in *Proc. IJCAI*, 2020, pp. 3422–3429.
[50] G. K. Patel, J. M. Hahne, C. Castellini, D. Farina, and S. Dosen, "Context-dependent adaptation improves robustness of myoelectric control for upper-limb prostheses," *J. Neural Eng.*, vol. 14, no. 5, Oct. 2017, Art. no. 056016.
[51] M. Franceschi, L. Seminara, S. Dosen, M. Strbac, M. Valle, and D. Farina, "A system for electrotactile feedback using electronic skin and flexible matrix electrodes: Experimental evaluation," *IEEE Trans. Haptics*, vol. 10, no. 2, pp. 162–172, Apr. 2016.
[52] M. Sartori, G. Durandau, S. Došen, and D. Farina, "Robust simultaneous myoelectric control of multiple degrees of freedom in wrist-hand prostheses by real-time neuromusculoskeletal modeling," *J. Neural Eng.*, vol. 15, no. 6, Dec. 2018, Art. no. 066026.
[53] M. Barsotti, S. Dupan, I. Vujaklija, S. Došen, A. Frisoli, and D. Farina, "Online finger control using high-density EMG and minimal training data for robotic applications," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 217–223, Apr. 2018.
[54] F. Cordella, A. L. Ciancio, R. Sacchetti, A. Davalli, A. G. Cutti, E. Guglielmelli, and L. Zollo, "Literature review on needs of upper limb prosthesis users," *Frontiers Neurosci.*, vol. 10, p. 209, May 2016.
[55] M. Tan, B. Chen, R. Pang, V. Vasudevan, M. Sandler, A. Howard, and Q. V. Le, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2820–2828.

EDOARDO RAGUSA is currently an Assistant Professor with the University of Genoa. His research interests include machine learning in resource constrained devices, convolutional neural networks, and applications of artificial intelligence.

CHRISTIAN GIANOGLIO is currently a Postdoctoral Researcher with the University of Genoa. His research interests include embedded systems, machine learning, and advanced signal processing.

STRAHINJA DOSEN (Member, IEEE) is currently an Associate Professor with the Center for Sensory-Motor Interaction, Department of Health Science and Technology, Aalborg University. His research interests include movement restoration and control, rehabilitation robotics, sensory feedback, and human–machine interfacing for sensory-motor integration.

PAOLO GASTALDO is currently an Associate Professor with the University of Genoa. His research interests include machine learning, embedded computing systems, and artificial tactile sensing.