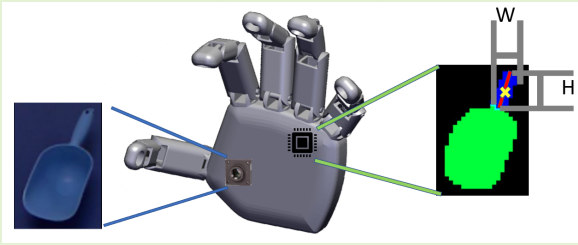
Affordance segmentation using tiny networks for sensing systems in wearable robotic devices

Edoardo Ragusa *(Member, IEEE),* Strahinja Dosen *(Member, IEEE),* Rodolfo Zunino, and Paolo Gastaldo

***Abstract*— Affordance segmentation is used to split object images into parts according to the possible interactions, usually to drive safe robotic grasping. Most approaches to affordance segmentation are computationally demanding; this hinders their integration into wearable robots, whose compact structure typically offers limited processing power. The present paper describes a design strategy for tiny, deep neural networks that can accomplish affordance segmentation and deploy effectively on microcontroller-like processing units. This is attained by specialized, hardware-aware Neural Architecture Search (NAS). The method was validated by assessing the performance of several tiny networks, at different levels of complexity, on three benchmark datasets. The outcome measure was the accuracy of the generated affordance maps and the associated spatial object descriptors (orientation, center of mass, size). The experimental results confirmed that the proposed method compared satisfactorily with the state-of-the-art approaches, yet allowing a considerable reduction in both network complexity and inference time. The proposed networks can therefore support the development of a teleceptive sensing system to improve the semi-automatic control of wearable robots for assisting grasping.**



***Index Terms*— Embedded Systems, Tiny CNNs, Microcontrollers, Wearable robots, Affordance segmentation, Grasping**

I. Introduction

Affordance segmentation consists in detecting [1] and identifying potential functional interactions that an object can afford, and segmenting that object into parts accordingly. Successful methods for affordance segmentation used RGB cameras, deep learning (DL), and powerful computing units [2]–[5]. The literature, however, seems to lack implementations relying on limited computational resources. This ultimately hinders applications to wearable robots that can assist, restore or augment grasping function, such as prostheses, exoskeletons, and supernumerary limbs [6].

These devices are mechatronically advanced systems with multiple degrees of freedom, but effective user control is still a challenge. Normally, the user needs to control all functions of the device by generating explicit commands, which can be slow and cognitively taxing, and wearable assistive robots are therefore rarely used outside of the lab or they are often rejected by their users [7]. One approach to addressing this challenge is to enhance wearable robots with teleceptive sensing capabilities (the sensing that occurs without physical contact with the object being tested [8]). Teleceptive sensing can be used to implement semiautonomous control, which can substantially reduce the users' physical and mental efforts while controlling these devices [6], [9]. For instance, a wearable robot (e.g., a prosthesis) can be endowed with a camera so that the device can "see" the target object. An image analyzer then estimates the object properties and directly configures the robotic hand for grasping, without any additional input from the user [10]–[12].

Affordance segmentation can be useful in this scenario, as it can highlight the graspable parts of an object (see Fig. 1), hence allowing the image analyzer to focus only on the relevant segments of the object. However, to enable this application, novel resource-aware methods for affordance segmentation that can be deployed onto the embedded controller of a wearable robot need to be developed.

The key contribution of this paper lies in applying hardware-aware neural architecture search (HW-NAS) methods to the enhancement of teleceptive sensing systems. As a result, inexpensive, low-resolution cameras integrated with low-performance computing units can support affordance segmentation using tiny networks. The HW-NAS framework applies an evolutionary algorithm to perform a block-wise search in a hardware-friendly search space, to pinpoint the best-performing network architecture that satisfies the specified constraints. To address the set of candidate networks compatible with the target device, the design approach presented here relies on an empirical model of the inference time.

The experimental assessment of the developed method on three established benchmarks [1], [2], [13] demonstrated that good accuracy was achieved by the resulting networks, even in the presence of a considerable reduction in the networks' sizes. The tiny neural models compared satisfactorily with much more complex solutions reported in the literature [14], whereas

E. Ragusa, R. Zunino, and P. Gastaldo are with DITEN, University of Genoa, Genova, Italy e-mail: edoardo.ragusa@unige.it

S. Dosen is with Department of Health Science and Technology, Aalborg University, 9220 Aalborg, Denmark

the required computational power for the run-time operation was reduced substantially. For assessing potential application, the affordance maps were also used to estimate the relevant parameters for hand-based grasping, namely, the barycenter of the graspable area, as well as its size and orientation. The experiments also addressed the impact of the image resolution on the estimates of the object properties.

The contribution of the paper can be summarized as follow:

- The development of advanced teleceptive devices supported by commercial microcontrollers, equipped with deep neural networks (DNNs) for real-time affordance segmentation.
- A software-hardware codesign strategy based on HW-NAS for teleceptive implementations.
- A strategy based on the empirical relationship between the number of required Flops and the associate inference time, thus yielding efficient, configurable implementations of the HW-NAS.
- A set of tiny DNNs for affordance segmentation, hosted by an stm32f746g-disco board, featuring a latency smaller than 0.25 s.
- Tests on established benchmarks, confirming the overall effectiveness of the developed method.

II. Related works

A. Grasp Affordance Prediction

Understanding an object's affordance at the pixel level is denoted as "affordance segmentation", "affordance detection", or "object part labelling". It is closely related to grasp selection [15], in particular, to segmentation-based grasping methods. The two problems overlap in defining the graspable regions of the object since a suitable grasp depends on the function afforded by the object [3]. Convolutional neural networks (CNNs) are a popular approach to that task [2]. In [2], a deep learning object detector improved the affordance detection accuracy by automatically selecting objects within images. The work presented in [16] considered the intrinsic dependence of affordance on the task, whereas [17] involved a rank-based strategy of affordances. Occlusions due to human-object-robot interactions were modeled in [18], and, with a generalization to consider unseen objects in [19]. When applying multitask learning [4], a CNN reconstructed 3D objects and predicted their affordable parts at the same time. The research described in [5] introduced a new formulation of the affordance prediction problem for RGB images with multiple objects. Likewise, [20] augmented affordance segmentation with keypoint detection. A simulator included in [21] generated a collection of grasping sequences, while [13] involved a large dataset with RGBD information. Synthetic images were also considered in [22]. The approaches discussed above mostly neglect computational constraints that, however, can severely limit the inclusion of affordance segmentation in wearable systems.

B. Semi-autonomous control of grasping

Semi-autonomous control was used to improve performance and user experience when interacting with a variety of wearable robots for assistance or restoration of grasping, including supernumerary limbs [6], exoskeletons [23] as well as arm and hand prostheses [24]. This approach makes the devices smart so that they can accomplish some tasks independently. The users can therefore perform complex functions by using simple commands, which improves control and practical utility, while decreasing the cognitive load.

The application in prosthetics is particularly interesting for the present paper, as these are compact battery-powered systems that need to work over a prolonged period (hence, both physical and computational constraints). RGB cameras [25], [26], stereopairs [27] and depth sensors [11] were added to or even integrated [28], [29] into robotic prosthetic hands. Image and point cloud analysis was then used to estimate object properties (shape, and size – width and height) and, based on this information, decide on the convenient grasp strategy using a simple set of rules (e.g., heuristically constructed decision trees) [11], [27]. Alternatively, the image data was processed using CNNs to support grasping classification [30], object segmentation [28], [29], and hand-pose estimation for bimanual interaction [31]. The uncertainty introduced by partial occlusions was modeled in [10]. A hardware-amenable deep neural network supported the segmentation of the affordable parts of objects [14], further extended by object-recognition abilities in [32].

C. Tiny deep networks

In modern, smart sensors equipped with deep learning, sensing nodes can mine sophisticated information at the local level [33]. This requires specialized software-hardware co-optimization to identify the best CNN architecture. When applying NAS [34], one should define a search space, namely, the set of admissible candidate networks. The popular MNAS was built from the MobileNet [35] design space.

Practical implementation may however exhibit a crucial issue: networks having the same number of Flops and parameters [36] may differ in latency values when they are ported on different devices. This is due to specific inference engines that may interfere with optimization techniques [37]. If optimization is not matched by hardware resources, this may even worsen the resulting performances. Likewise, running models with an arbitrary quantization on microprocessors can slow down execution, if the instruction set fails to support that representation [38]. The MCUnet applied a comprehensive optimization procedure to select the architecture and set up the computing layer, yielding excellent performances [39]; the custom software layer, however, made it difficult to use and customize the model [40].

The major issues when applying NAS stem from its computational cost and the diversity in target platforms, since effective approaches for designing tiny networks are mostly tailored to specific devices. Super networks span all the possible architectures spanned by a search space [41], and may represent a viable solution to the computational problem. Conversely, wearable devices host heterogeneous computing platforms and sometimes embed proprietary software, which might complicate the formulation of a general strategy. The approach presented in this paper therefore keeps a high level

of abstraction, to derive a design procedure for networks that may comply with heterogeneous computing platforms.

III. Material And Methods

The tiny DNNs run on microcontrollers and work out affordance segmentation maps from raw RGB images. This section first describes the smart sensing system, then outlines the NAS design approach.

A. Smart Sensing Device

Figure 1 illustrates the overall conceptual solution: a general wearable grasping robot hosts an RGB/RGB-D camera to allow teleceptive sensing, and a microcontroller. The acquired frames are stored in the memory of the microcontroller, which processes raw images, executes the control algorithm to select the grasp strategy, and prompts the resulting commands to the embedded actuators.

Electromyography (EMG) signals (not shown in the Figure) trigger and drive the control pipeline of the semi-autonomous system, to support volitional control. The EMG-based user interface prompts a volitional command (muscle contraction) to indicate the user's intention to grasp an object. Then the teleceptive sensor acquires an RGB image of the target, and the tiny DNN estimates the associated affordance map. The map allows the system to focus on the object parts that are labeled as graspable (highlighted in blue), and estimate the properties relevant for selecting the grasping strategy. For instance, simple processing can yield the dimensions of the graspable area, its barycenter (yellow) and orientation (red). Non-graspable parts are marked in green.

The map information can be directly fed into subsequent sections of the semi-autonomous control pipeline in combination with the output of other sensing devices to select the grasp preshape using, for instance, decision trees [11]. Affordance maps can be employed to estimate the width and height of the graspable area, and the previous work showed that this information can be used for the effective selection of grasp modality, as explained in Section II.B. The estimate of object orientation can drive a gripper orientation to properly align the hand to the object for grasping.

In summary, the result of tiny network processing can enhance the analysis of data provided by the camera, thus endowing the smart sensor with the capability of detecting the position and shape of the graspable object part.

B. Defining a suitable HW-NAS for affordance segmentation

State-of-the-art approaches to the design of tiny DNNs for embedded systems use HW-NAS procedures. Existing works on the deployment of specialized segmentation models require a co-optimization of the software layer that runs on the embedded device [39]. The research presented here yields a HW-NAS strategy for affordance segmentation that does not involve any tuning of the low-level software layer, yet enables to deploy effective networks on very constrained devices. This enhances general applicability and timing effectiveness.

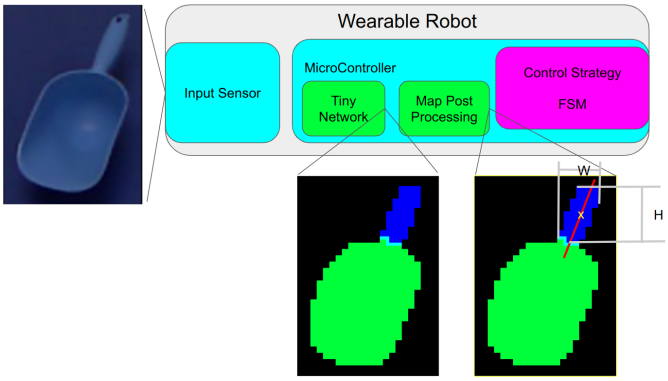


Fig. 1: Overall scheme of the proposed sensing system

Three main aspects characterize any HW-NAS strategy: the search space, $\mathcal{SS}$, of admissible candidates, the search algorithm $\mathcal{SA}$, and an evaluation criteria $\mathcal{EC}$. In this paper, the basic architecture schema relies on the conventional *backbone head* structure adopted in the segmentation literature. The set of possible backbones only includes a linear combination of parametric building blocks, i.e., a single-branch neural network [36]. A set of 6 parameters characterizes each building block, adhering to the MobileNetV2 model. LR-ASSP [42] supports the segmentation head; it includes a few upsampling layers and connects to the two layers of the backbone. One architectural parameter, namely, the number of filters in the last convolutional layer, does not affect the size of the output mask, and therefore, only that parameter of the segmentation head enters the search procedure. In addition to the architecture blocks, the size of the input image is included in $\mathcal{SS}$, as that quantity can affect accuracy, memory, and Flops significantly.

A standard evolutionary algorithm supports $\mathcal{SA}$; to explore candidate architectures, it applies random mutations on a parent architecture A_p according to a function R_m:

$$A_s = R_m(A_p) \tag{1}$$

$R_m()$ randomly performs one of a set of possible actions: 1) changing the input size, 2) changing the number of building blocks, 3) altering a building block in one of its parameters (kernel size, number of filters, expansion factor, activation function, stride), and 4) changing the number of filters in the segmentation head.

The procedure starts with a preset parent architecture, A_p, which is initialized to a minimal configuration, holding the smallest number of blocks to prioritize fast networks [43]. The algorithm completes a fixed number, N_g, of iterations. At each iteration, a set of N_c children architectures is spawned

$$C_s = \{A_i\}, i = 1, ..., N_c; A_i = R_m(A_p) \tag{2}$$

The architectures in C_s are all trained on a labeled training set $\{X_t, Y_t\}, t = 1, ..., Z$, where X_t is the input image and Y_t is the affordance segmentation mask. A penalty function P_f allows to pinpoint the best child A_b:

$$A_b = \operatorname{argmin} P_f(A_i), A_i \in C_s. \tag{3}$$

The best child, A_b, takes the place of its parent, A_p, in (2) and spawns the subsequent generation, hence $\mathcal{SA}$ can scan a wide selection of candidates. Eventually, the best architecture, A^*, is the one that scored the smallest value of P_f from among all comparisons.

C. Empirical modelling of the computational cost

The evaluation function, P_f, typically includes two penalty terms: one that considers the error on the validation set, $\mathcal{L}_V$, and another that penalizes the computational cost $\mathcal{L}_C$. Such a reasonable approach, however, might not best fit the problem at hand, especially when tight constraints are involved.

A viable solution might lie in modelling the correlation between latency and network architecture in $\mathcal{SS}$, for the target embedded system. The general procedure consists in drawing a set of networks randomly from $\mathcal{SS}$; that set need to be large enough to ensure a large variability of both quantities. In the present research, inference time ranged from less than one second to more than 15 seconds, thus covering any possible application considered for the target device; the actual values shall be set depending on the application domain and the specific HW platform adopted. Then one deploys the DNNs on the target device and measures the inference time, thus building an empirical model of the Flops/time relationship. This analysis can turn the specific NAS design into a decision-based problem, removing the computational penalty term from P_f: any network featuring a number of Flops higher than a threshold can be discarded.

In practice, the designer sets the largest admissible inference time and retrieves the associate number of Flops by inverting the experimental modelling relation. The latter quantity sets the threshold for network candidate rejection, while the accepted networks can be compared on validation errors only. The selection constraint enters the generation mechanism (2) of the set C_s. The procedure avoids to prioritize smaller networks as soon as the networks meet the constraint.

IV. Experimental Setup

The experimental setup covered several aspects. A preliminary analysis completed the modelling of the computational cost, as described in the previous Section. A subsequent assessment compared the obtained affordance segmentations with benchmarks. An additional experiment considered to what extent the "low resolution" affordance maps affected the estimation of grasp parameters (object descriptors). A final empirical session addressed the actual computational performance, by measuring the real-time performances obtained after deploying the created networks on the target microcontroller.

A. Computational cost model

In the case of the STM32F746NG board, the microcontroller unit (MCU) hosts a single-precision floating-point unit. The amount of RAM and flash memory in that target device is an order of magnitude lower than in standard devices, and the single floating point unit prevents parallelism.

In the analysis described in the previous Section, the inference time constraint proved dominant, as no configuration in the spanned search space ever met the latency constraints, when involving an excessive memory footprint. Figure 2 shows the measured relationship between MFlops and inference time. The markers refer to the values measured for the various deployments, supporting a linear regression model. Equation 4 expresses the relationship between time and MFlops.

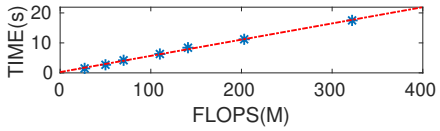


Fig. 2: Relationship between MFlops and inference time when using Cortex-M7 32-bit RISC core

$$time = (Flops/10^6) * 0.054 + 0.2648 \quad (4)$$

That linear relation yielded an estimate of the inference timing for every candidate DNN, hence one needed not to optimize the model with the STM32 toolchain. This made making the selection process much faster and more stable.

A final remark concerns the main aspects determining the eventual inference time, namely, MAC operations and memory access. The above analysis only involves external RAMs, which seems a reasonable assumption when dealing with portable electronics. Memory access time may therefore show up larger as compared with cases in which all operations are supported by on-chip memories. In practice, this trend becomes an upper bound for small DNNs; moreover, exact information about internal memory management is not often available, which greatly affects the general applicability of design strategies.

B. Generalization Performance

The present paper aims to prove that tiny networks can support affordance segmentation in the conditions in which the affordances are well defined. Therefore, the problem at hand focused on foreground objects with good framing settings. The UMD [1] is a well-known benchmark that contains 28.843 RGB-D images of 7 categories of objects that fulfill the aforementioned requirements. The dataset includes many framing angles for each object, providing valuable insights about the capability of the models to handle the framing issues. Two additional datasets have been included to evaluate the impact of factors that may complicate the segmentation, for instance, more object classes with ambiguous affordance definitions and heavy occlusions. The MW dataset [13] accounts for 23605 RGB-D images collected using 37 classes of objects. This dataset contains ambiguous affordances as the collection contains objects like boxes, balls, and books where the borders between graspable and non-graspable surfaces are not self-evident. Finally, the IIT dataset (IIT) [2], which contains 8835 images featuring different framing and resolutions, was used for the assessment. In summary, the three datasets contain very different sets of objects, providing de facto three different benchmarks. The validation sets were generated from the training set using a standard hold-out procedure. The test

patterns were never involved in the tuning of any parameters or hyperparameters.

Following the approach proposed in [14], all the grasping affordances were grouped into a unique class named *grasp* while the other affordances were assigned to "*do not grasp*" class. The learning problem, therefore, consists of a three-class pixel-wise classification where the predictor should discriminate between *grasp*, *do not grasp*, and *background*. The foreground images of the object were extracted using the corresponding segmentation masks extracting the bounding box containing the object. Different objects led to boxes with different eight-width ratios. The image ratio was adjusted to a square ratio using 0 padding to fit the network's input size. After the processing, training size and test size were 23.708 and 5135 for UMD, 9186 and 1969 for IIT, and 12.401 and 5888 for MW, respectively.

Five baseline solutions were considered. The version of MobileNetV3 (MobV3) presented in [14] for affordance segmentation features the lowest computational requirements, compared to the models used previously [14], while maintaining satisfactory accuracy on the benchmarks. A tiny model for object segmentation [28] intended for implementation on prosthetic hands, not designed for affordance segmentation, sets the reference for models developed specifically for application in prosthetics. Then, two models based on EfficientNetB0 and VGG16, paired with Unet segmentation head were selected to provide a reference for large-scale robotics approaches and related learning problems like grasp selection. The architecture uses a segmentation head connected to 4 feature maps corresponding to as many layers of the backbone. We opted for these two general configurations with respect to single instances of recent solutions from robotics literature because their target setup is substantially different making the comparison unfair [44]. The last baseline was SegFormer [45]. This model is based on transformer architectures [46] that are proving to be valuable alternatives to CNNs for computer vision tasks. Among existing transformer-based segmenters (e.g., Swin-Transformer, Mask2Former, and MaskDINO [46]), the SegFormer balances computational costs and generalization performance and this makes it an interesting comparison for the proposed approach.

The NAS executed 100 generations for every dataset. Seven values of the threshold parameter were used. The first one was 13,400K Flops which yielded an expected inference time close to 1 s. The remaining 6 thresholds were obtained by multiplying this threshold by 6,4, 2, 0.5, 0.33, and 0.25, respectively. Hereafter, we will call the different models propαS where α indicates the multiplication factor. As explained before, the threshold did not necessarily coincide with the inference time because the selected architecture could have a lower inference time compared to that given by the threshold. All networks were trained for 10 epochs following the early stop strategy [35]. The best architecture as selected by NAS was then trained for a maximum of 100 epochs with an initial learning rate of 10^{-3}, the learning rate reduction on the plateau, and early stopping using the validation loss as the metric. The generalization performance was measured using the test set[1].

C. Estimating grasp relevant parameters

As explained in section III.A, the affordance maps provide useful information for selecting an appropriate preshape as they allow focusing the analysis on the graspable surface of an object. However, the decrease in the complexity of the networks that predict the affordance map inevitably leads to a lower accuracy, which can produce errors when estimating the grasp relevant descriptors (dimensions, barycenter and orientation). To assess this, we computed the error when estimating the three descriptors from the generated affordance maps. This analysis also considered the impact of image resolution which plays an important role in the quality of estimation.

The tests were performed using the images extracted from the UMD dataset. This set was selected because none of the datasets provided information about the object size in physical units. The UMD dataset, however, uses a fixed camera that frames the object always from the same position, and therefore, in this case, the object's size could be estimated with high accuracy using the depth maps. The other two datasets used different framing angles and positions, making the reconstruction of the physical properties more challenging.

The first considered descriptor was the barycenter of the graspable surface, computed as the average position of the pixels marked as "graspable". The second was the physical dimension of the object including its width (W) and height (H) computed as the maximum and minimum coordinates of the pixels classified as belonging to the object, as shown in figure 1. The last descriptor was the orientation estimated as the angle of the principal component (PC) of the mask containing the graspable pixels. Figure 1 shows this information using a red segment superimposed on the image. This descriptor is useful when the object under analysis has an elongated shape (e.g. a handle), while the orientation angle is undefined for approximately spherical objects. The orientation was therefore computed only for classes of objects containing handles or having elongated shapes.

D. Deployment

The models generated using the proposed procedure were deployed on the target board, the stm32f746g-disco. The deployment was performed in two steps. First, the network was converted via TFLite; then, the STM32 X-Cube-AI suit was exploited to optimize the model. The memory indexing was tuned to use the external memory, when necessary, to host the tensors during the propagation along the layers of the network. Data representation was set to 32-bit because STM32 X-Cube-AI supports 8-bit representation only for fully connected layers, but the architectures tested do not use these operators. Eventually, one can consider that the measured performance corresponds to the worst-case analysis considering that the quantization can decrease latency. All the measurements were performed by using the STM32 design suite utility for testing.

[1]Generated architecture available at: `https://github.com/SEAlab-unige/IEEESensorsJournal2023`

model	back.	grasp	don't gr.	H7	KFlops	Params
UMD						
Prop1s	0.980	0.827	0.863	✓	12.46M	10.8K(10.4K)
Prop0.5	0.989	0.820	0.880	✓	7.02M	18.9K(18.5K)
TinySeg	0.99	0.598	0.636	✓	63.44M	4K(N.A.)
MobV3	0.985	0.833	0.930	✓	0.2GM	196K(184K)
SegFor.	0.979	0.872	0.914	✗	8.4G	3.8M(3.4M)
EFF	0.989	0.886	0.941	✗	1.32G	13M(4M)
VGG16	0.996	0.825	0.857	✗	10.94G	20M(15M)
MW						
Prop1s	0.970	0.804	0.618	✓	13.65M	50.8K(49.7K)
Prop0.5	0.967	0.813	0.563	✓	5.91M	15.5K(14.7K)
TinySeg	0.987	0.853	0.041	✓	63.44M	4K(N.A.)
MobV3	0.980	0.857	0.550	✓	0.2G	196K(184K)
SegFor.	0.980	0.851	0.777	✗	8.4G	3.8M(3.4)
EFF	0.988	0.900	0.810	✗	1.32G	13M(4M)
VGG16	0.988	0.862	0.606	✗	10.94G	20M(15M)
IIT						
Prop1s	0.970	0.459	0.686	✓	12.63M	55.3K(54.7K)
Prop0.5s	0.970	0.421	0.654	✓	6.66M	27.4K(26.8K)
TinySeg	1	0	0	✓	63.44M	4K(N.A.)
MobV3	0.977	0.647	0.808	✓	0.2G	196K(184K)
SegFor.	0.981	0.832	0.917	✗	8.4G	3.8M(3.4M)
EFF	0.990	0.890	0.953	✗	1.32G	13M(4M)
VGG16	0.992	0.856	0.923	✗	10.94G	20M(15M)

TABLE I: Generalization performance for UMD, MW and IIT

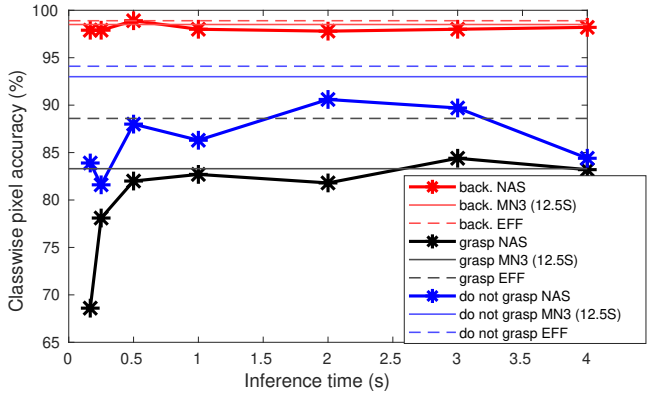


Fig. 3: Performance for dataset UMD. NAS stands for the generated networks while MN3 and EFF denote the benchmark.

V. Results

A. Generalization Performance Analysis

Table I reports the results obtained when testing the networks on UMD, MW and IIT datasets. The columns indicate the model, three pixel-wise accuracies for the different classes, the capability of the target microcontroller to support the model, the number of Flops, and the number of Parameters respectively. The last column, between the brackets, reports the number of parameters of the encoder/backbone summarizing the distribution of the weights inside the networks. Only the intermediate thresholds $1s$ and $0.5s$ were considered.

UMD dataset confirms the suitability of tiny models when framing conditions are good, highlighting small differences with respect to computationally demanding models from robotic literature. All models tested on MW exhibited a significant gap between the performance obtained on the validation set and that achieved on the test set. This behavior, which was not observed for other datasets, is probably due to a bias in the labeling process rather than a limitation of the trained DNNs. The IIT dataset contains heterogeneous images and in this case, the labeling also introduced a non-negligible amount of noise. The results confirm that this dataset was indeed the most challenging as the gap between larger nets and the tiny networks is most pronounced. However, the drop in accuracy is very likely due to features that fall outside the scope of the paper, for example, occlusions and difficult framing settings. Nevertheless, this result establishes the boundaries of the proposed work: the generated tiny networks can support affordance segmentation on foreground images and can be useful blocks in the overall control pipelines but should not be considered as a stand-alone solution. The analysis of parameters and Flops provides a quantitative characterization of the distinction between the models supported by the target microprocessor and the other solutions. This clarifies the gap in terms of computing requirements that hinders the deployment of the latter models. In addition, the distribution of the weights illustrates the major role played by the encoder/backbone for the tiny networks.

Figure 3 displays the pixel-wise accuracy versus inference time for the UMD dataset. The x-axis shows the estimated inference time for the NAS-generated models. The accuracy of the benchmark solution (MobV3) is indicated as a constant (full line) with the inference time written in the legend. TinySeg is not included because the results are much worse compared to the other options. Between the three unconstrained solutions, that are not supported by the target platforms, EFF was selected because it scored better. The y-axis shows the accuracy for the three classes (*background*, *do not grasp*, and *grasp*) plotted using different colors.

The results demonstrate that for the classes *grasp* and *background* the inference time can be substantially decreased (almost 10 times) with respect to the MN3, i.e. the largest model supported by the target platform, without the noticeable loss of performance. The drop in accuracy arises only for the smallest of networks. The accuracy for class *do not grasp*, however, remains consistently smaller when using the tiny networks compared to that achieved with MobV3. Nevertheless, this class is not relevant for estimating the grasp parameters. Interestingly, the largest architecture tends to become sub-optimal, i.e. as shown in the figure for *no grasp* class, the largest network does not lead to the best performance. This result can be explained by the structure of the proposed $\mathcal{SS}$ because the segmentation head is designed to minimize hardware requirements, which then limits its scaling capability. In addition, the admissible backbones can have many layers. However, the proposed search space neglects those building blocks that could significantly enhance the performance of medium size networks but are less important for tiny networks; e.g., skipped connections. Eventually, EFF sets a gold standard that cannot be achieved using networks supported by constrained devices.

B. Grasp parameters analysis

Figure 4 shows the average errors in the estimation of the grasp relevant parameters. The x-axis and the representation

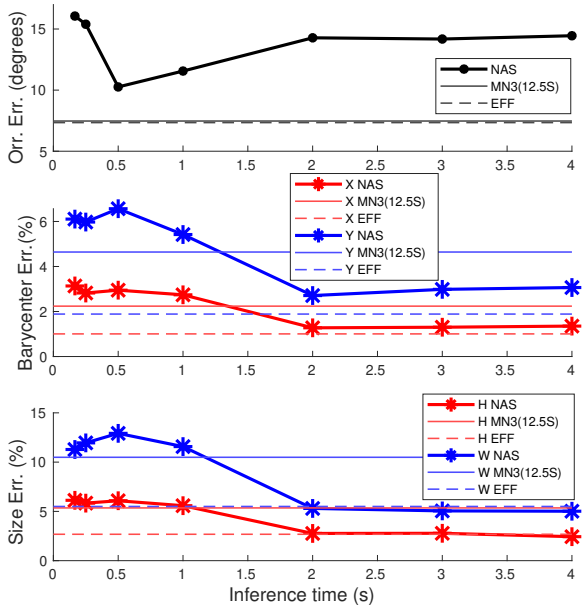


Fig. 4: Performance for grasp parameters. NAS stands for the generated networks, MN3 and EFF denote the benchmark.

of the performance of MobV3 and EFF are the same as in figure 3. EFF establishes the performance that one could reach without considering computational constraints. The first subplot reports the orientation error, in degrees, defined as the difference between the angles of the principal component of the reference graspable mask, i.e. the ground truth mask provided in the dataset, and the principal component of the predicted mask. The average error was lower than 16 degrees for all tiny networks tested and this confirms the suitability of the proposed approach in retrieving object orientation. Considering the envisioned application, the estimation of object orientation is robust to some deviations since they can be compensated by the user of a wearable robot. For instance, the user of a prosthesis can slightly rotate the arm from the shoulder to compensate for some misalignment between the hand and object orientation. The standard error (SE) has been computed to evaluate the statistically significant of the proposed results. The difference between the predictor was always larger than two times the sum of the SE, confirming that average error is a reliable estimator.

The central plot is the percent error in estimating the barycenter position, computed as the error in pixels divided by the total number of pixels of the image, which depends on the input resolution of the network selected by the NAS. Two colors indicate the two coordinates of the barycenter. Overall, the average estimation error is small ($< 7\%$) and the plots reveal an interesting trend, where the tiny networks with the longer inferences time ($> 1.5s$) score better accuracy (with a difference larger than the 3 times the sum of the SE) than the benchmark (MobV3), despite the latter is still the most demanding architecture. SE analysis confirmed the statistical significance of the results.

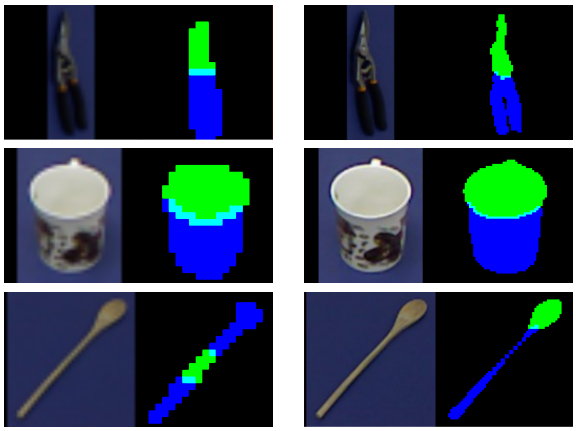


Fig. 5: Examples of the input/output relationship for Prop$\frac{1}{6}$S (left) and MobV3 (right).

The bottom plot shows the relative error in estimating object dimensions, namely, height (H) and width (W), computed as the difference between the estimates and the true value retrieved by the dataset labels in pixels. The error is normalized with respect to the number of pixels of the image. The trend is similar to that obtained for the prediction of the barycenter. The error difference between the proposals and the baseline is less than 2% and for the longer inference times, the tiny networks outperform the benchmark.

Figure 5 shows a few representative examples of input/output (I/O) results. Each row contains 4 figures divided into pairs. The first column shows the input image and the affordance map estimated using the smallest network tested i.e. the network obtained by setting the inference time threshold to $\frac{1}{6}$S. The second column shows the same information for MobV3. The images were resized in post-processing to 620x620 for the sake of visualization. The examples clearly show the different granularity of the input and output images that the networks use and generate, respectively. Importantly, despite the low-quality input images, the tiny networks successfully distinguished the parts of the objects in the foreground. The output is a low-resolution affordance map that nevertheless still contains enough information to estimate the most important geometrical features of the objects. The spoon in line 3 represents a case where the tiny network failed to correctly segment the object parts but it nevertheless successfully recognized the silhouette of the object.

C. Computational performance

Figure 6 summarizes the results of the deployment tests. The plots are organized in a 2 x 2 grid. The first row contains plots showing memory occupation in KiB, while the second row refers to inference time in seconds. The first column comprises the plots with the total number of parameters on the x-axis, while the second one shows KFlops on the x-axis.

The four plots aim to show the relationship between the two features characterizing the network complexity, namely, the number of parameters and Flops, shown on the x-axis and computed using standard deep learning tools, and the three main constraints for real-time performance on embedded

This article has been accepted for publication in IEEE Sensors Journal. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/JSEN.2023.3308615

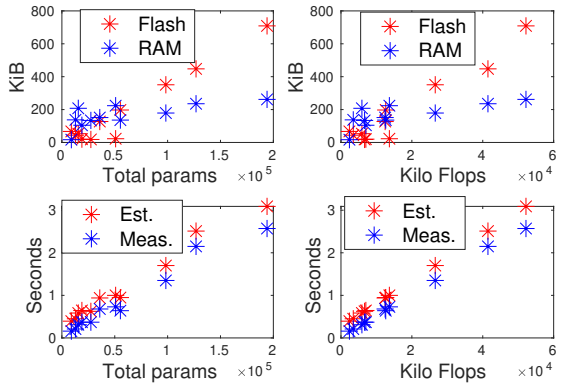


Fig. 6: Relationship between the architectures' number of parameters and Flops and hardware occupation.

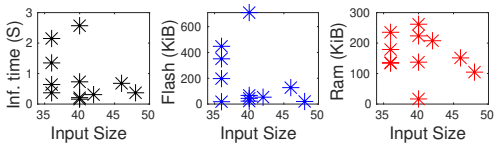


Fig. 7: Relationship between input and hardware measures.

devices, i.e., inference time, RAM, and Flash occupation, shown on the y-axis. The stars represent the models generated with the proposed NAS procedure. Blue and red colors indicate the quantities compared in each subplot, i.e., Flash and RAM occupation, or estimated and measured inference time.

The figure highlights that the inference time linearly increases with the Flops and the number of parameters in agreement with the model presented in eq. 4. In a few cases, a slight increase in the number of parameters yields a faster inference. This somewhat counterintuitive result can be explained by the fact that the number of Flops is not set only by the number of parameters in CNNs but depends also on the exact network architecture, for instance. This confirms that simple models of the inference time can yield useful apriori measures for the behavior of a model when deployed on a microcontroller-like device in combination with the proposed $\mathcal{SS}$. Memory occupation, however, is less predictable for small models, while the dependency seems to be linear for larger models. A second important observation is that the figure confirms that the model generated with the proposed approach always satisfies the memory constraints for both Flash and RAM. Therefore, good results can be obtained without external memories, thereby reducing power and weight. Similarly, the measured inference time for the selected models is always significantly smaller than the upper bound set by the threshold.

Figure 7 investigates the role of input image size on the hardware measures. The x-axis shows the selected input sizes. Note that each input size can have several associated architectures (hence, more points above each size in the plot). The $y-axis$ shows the measures, i.e. inference time in seconds and the two memories in KiB. Accordingly, the figure is divided into three plots, one per descriptor. As shown in the figure, the NAS procedure opted for small input sizes ranging between 36 and 48 pixels independently of the threshold. The measures do not show particular trends probably because even though the input resolution is important, other variables such as network depth, and kernel size, have a more decisive impact.

VI. Discussion

The present work shows that tiny DNNs generated by a tailored HW- NAS procedure can run on embedded microcontrollers to estimate fine-grained information about target objects. The resulting affordance masks segment the object into non-graspable and graspable parts. By estimating the spatial properties of the graspable segments (orientation, position, and size), a semi-autonomous control system can select a suitable grasping strategy. This research, therefore, aims to fill a gap in the literature, which mostly focuses on large-scale robotics. When addressing wearable robots, processing capabilities can in fact be rather limited. An additional important contribution lies in applying HW-NAS to support specific constraints of target applications.

The research only considered RGB inputs, although one might note that in segmentation tasks RGB-D inputs can improve accuracy [47]. In a recent study about the impact of depth information on UMD and MW datasets, however, the results exhibited a negligible improvement when compared with RGB-only approaches, at the expense of a significant increment in computational cost [48]. The present work focused on RGB because that domain seemed to yield the most convenient scenario. In addition, the available benchmarks were not really designed to highlight the benefits of RGB-D inputs. Nevertheless, the described design methodology can easily cover RGB-D data by extending the structure of the search space $\mathcal{SS}$ accordingly.

The analysis of generalization ability considered three established datasets, with various types of images, mostly to get a comprehensive assessment of the generated network models. As the main goal was to assess the network's ability to perform affordance with limited processing resources, the analysis did not involve framing, occlusion, and illumination issues, which anyway remain important tasks for future work. The results on the IIT and UMD datasets proved that, in the presence of those issues, the performance of the solutions degraded with respect to larger networks, which was especially true for tiny networks. At the same time, one should consider that the affordance module developed in this research will integrate into well-designed control pipelines. Those complex systems typically include components designed to mitigate some of the above-mentioned issues (e.g., by including an object detector [32]). These modules would also need to be downscaled and tuned to the desired inference time.

In the envisioned application of wearable robotics, satisfactory framing quality can be obtained by training the user to use the system for aiming. It is likely that users can attain effective framing strategies after short periods of training. The presented research has shown that effective framing can boost the performances of tiny networks to benchmark levels, even when the network complexity and inference time are limited. This aspect requires further investigation by a specific assessment, in which participants would use a wearable robot equipped

with the proposed smart teleceptive module to accomplish functional tasks. The assumption in the present work was that the user will position the wearable robot close to and in front of the object, and therefore, the analysis of the input image size did not consider the possibility of distant objects.

The results demonstrated that the tiny DNNs can estimate the affordance map with an accuracy that is comparable to that of the benchmark, especially regarding the *grasp* class. This is an encouraging result as the correct identification of the graspable part of the object is indeed the most relevant to decide the grasping strategy. The *do not grasp* class, for instance, was estimated with the lowest accuracy but this information is less relevant for the grasp selection.

Next, we have also demonstrated that the estimated affordance maps allow extracting the "physical" dimensions of the object with good accuracy, as the difference between the estimation error of the proposed networks and the baseline for position and object size were all lower than a few percent. Correctly estimating object orientation was, however, more challenging but this parameter is also less sensitive as the errors can be tolerated. The angle of the hand does not need to be perfectly aligned with the object tilt for successful grasping and the potential discrepancy can be corrected easily by the user through compensatory motion. The simple physical descriptors estimated in the present study were selected because they can be calculated using low computational cost. Even the extraction of the principal component (orientation descriptor) has a computational cost that is negligible with respect to that involved in evaluating the CNNs. In addition, similar descriptors were already used to inform the grasp type selection in semi-autonomous control [27]. Overall, the methods proposed in this work can lead to the development of a smart teleceptive system that can make the semi-autonomous control pipeline more precise and versatile. By focusing the processing on the graspable segment of an object, the grasp type selection can be responsive to the finer details of the object morphology. For instance, instead of modelling the object as a whole, the computer vision system enhanced with the affordance mask can focus the processing on the relevant part of an object (e.g., the handle in Fig. 1). This processing could be based on the RGB image, as demonstrated in the present study, but one could also analyze the point cloud "hidden" behind the affordance mask. The latter analysis could provide 3D measures, enabling thereby a more sophisticated approach to grasp selection.

The deployment results confirm that the network generated using the proposed algorithm can be executed in real-time on the target device. The smallest generated network achieved an inference time of 0.21 s. In this context, we considered real-time inference duration smaller than 0.5 s because the actual timing constraints are application dependent. We think that this delay is a good starting point for a large set of semi-autonomous control applications, where some additional latency in the system response can be tolerated in exchange for a decreased cognitive load on the user. As expected, the measured inference time is always lower than the estimated value because the generated network did not require external RAM usage. We opted to maintain this possibility because the same exact methodology could be applied to many related problems. Some applications could require a larger input resolution. Empirical evidence confirms that external RAM memories are required for larger input sizes.

The previous studies [14] used a hardware accelerator or high-range portable microprocessor. The empirical model related to the number of Flops developed in the present study could be as well used in these cases, but it needs to be updated considering the peculiarities of the deployment tools ranging from the software libraries to the physical organization of the memories and cores. All the models generated would achieve a very high frame rate when deployed in this kind of device. As highlighted by figure 3, the proposed search space could be suboptimal in this case. A reasonable strategy, if possible, would be to switch to a larger $\mathcal{SS}$ including more building blocks or branches in the architecture ensuring better feature extraction capabilities.

VII. Conclusion

The paper presented a novel approach to the design of tiny convolutional neural networks (CNNs) for affordance segmentation. The resulting neural models can fulfil the hard constraints imposed by the micro-controllers that are typically embedded in wearable robotic devices (prostheses, exoskeletons, and supernumerary limbs). A tailored HW-NAS procedure generated the networks while relying on an empirical model to express the relationship between network complexity and inference time. The experiments confirmed that the resulting tiny DNNs showed accuracy comparable to those attained by existing solutions, but could run on microcontrollers and complete their task in less than 250ms.

Future research steps will include a thorough treatment of the framing issue, as well as the integration of affordance detection within the full semi-autonomous control pipeline equipped with other sensing sources (e.g., depth camera). Clinical campaigns will investigate how the user framing skills can impact performance, and if these skills can be improved by dedicated training sessions.

References

[1] A. Myers, C. L. Teo, C. Fermüller, and Y. Aloimonos, "Affordance detection of tool parts from geometric features," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1374–1381.

[2] 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 *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 5908–5915.

[3] M. Hassanin, S. Khan, and M. Tahtali, "Visual affordance and function understanding: A survey," *ACM Computing Surveys (CSUR)*, vol. 54, no. 3, pp. 1–35, 2021.

[4] Z. Jiang, Y. Zhu, M. Svetlik, K. Fang, and Y. Zhu, "Synergies between affordance and geometry: 6-dof grasp detection via implicit representations," *arXiv preprint arXiv:2104.01542*, 2021.

[5] E. Corona, A. Pumarola, G. Alenya, F. Moreno-Noguer, and G. Rogez, "Ganhand: Predicting human grasp affordances in multi-object scenes," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 5031–5041.

[6] Z. Tang, L. Zhang, X. Chen, J. Ying, X. Wang, and H. Wang, "Wearable supernumerary robotic limb system using a hybrid control approach based on motor imagery and object detection," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 30, pp. 1298–1309, 2022.

[7] S. Salminger, H. Stino, L. H. Pichler, C. Gstoettner, A. Sturma, J. A. Mayer, M. Szivak, and O. C. Aszmann, "Current rates of prosthetic usage in upper-limb amputees–have innovations had an impact on device acceptance?" *Disability and Rehabilitation*, vol. 44, no. 14, pp. 3708–3713, 2022.

[8] N. E. Krausz and L. J. Hargrove, "A survey of teleceptive sensing for wearable assistive robotic devices," *Sensors*, vol. 19, no. 23, p. 5238, 2019.

[9] Y. Sun, T. Fei, X. Li, A. Warnecke, E. Warsitz, and N. Pohl, "Real-time radar-based gesture detection and recognition built in an edge-computing platform," *IEEE Sensors Journal*, vol. 20, no. 18, pp. 10 706–10 716, 2020.

[10] B. Zhong, H. Huang, and E. Lobaton, "Reliable vision-based grasping target recognition for upper limb prostheses," *IEEE Transactions on Cybernetics*, 2020.

[11] M. N. Castro and S. Dosen, "Continuous semi-autonomous prosthesis control using a depth sensor on the hand," *Frontiers in Neurorobotics*, vol. 16, 2022.

[12] J. Zheng, J. Zhang, K. Yang, K. Peng, and R. Stiefelhagen, "Materobot: Material recognition in wearable robotics for people with visual impairments," *arXiv preprint arXiv:2302.14595*, 2023.

[13] Z. O. Khalifa and S. A. A. Shah, "Towards visual affordance learning: A benchmark for affordance segmentation and recognition," *arXiv preprint arXiv:2203.14092*, 2022.

[14] E. Ragusa, C. Gianoglio, S. Dosen, and P. Gastaldo, "Hardware-aware affordance detection for application in portable embedded systems," *IEEE Access*, vol. 9, pp. 123 178–123 193, 2021.

[15] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-1billion: A large-scale benchmark for general object grasping," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 444–11 453.

[16] P. Ardón, E. Pairet, R. P. Petrick, S. Ramamoorthy, and K. S. Lohan, "Learning grasp affordance reasoning through semantic relations," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4571–4578, 2019.

[17] F.-J. Chu, R. Xu, L. Seguin, and P. A. Vela, "Toward affordance detection and ranking on novel objects for real-world robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4070–4077, 2019.

[18] L. Liu, W. Xu, C. Lu *et al.*, "Fpha-afford: A domain-specific benchmark dataset for occluded object affordance estimation in human-object-robot interaction," in *2020 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2020, pp. 1416–1420.

[19] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," *The International Journal of Robotics Research*, vol. 41, no. 7, pp. 690–705, 2022.

[20] R. Xu, F.-J. Chu, C. Tang, W. Liu, and P. A. Vela, "An affordance keypoint detection network for robot manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2870–2877, 2021.

[21] F. Vasile, E. Maiettini, G. Pasquale, A. Florio, N. Boccardo, and L. Natale, "Grasp pre-shape selection by synthetic training: Eye-in-hand shared control on the hannes prosthesis," *arXiv preprint arXiv:2203.09812*, 2022.

[22] F.-J. Chu, R. Xu, and P. A. Vela, "Learning affordance segmentation for real-world robotic manipulation via synthetic images," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1140–1147, 2019.

[23] M. A. Vélez-Guerrero, M. Callejas-Cuervo, and S. Mazzoleni, "Artificial intelligence-based wearable robotic exoskeletons for upper limb rehabilitation: A review," *Sensors*, vol. 21, no. 6, p. 2146, 2021.

[24] J. W. Sensinger and S. Dosen, "A review of sensory feedback in upper-limb prostheses from the perspective of human motor control," *Frontiers in Neuroscience*, vol. 14, 2020.

[25] S. Došen and D. B. Popović, "Transradial prosthesis: artificial vision for control of prehension," *Artificial organs*, vol. 35, no. 1, pp. 37–48, 2011.

[26] P. Weiner, J. Starke, S. Rader, F. Hundhausen, and T. Asfour, "Designing prosthetic hands with embodied intelligence: The kit prosthetic hands," *Frontiers in Neurorobotics*, vol. 16, 2022.

[27] 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," *Journal of neural engineering*, vol. 12, no. 6, p. 066022, 2015.

[28] F. Hundhausen, D. Megerle, and T. Asfour, "Resource-aware object classification and segmentation for semi-autonomous grasping with prosthetic hands," in *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2019, pp. 215–221.

[29] J. Starke, P. Weiner, M. Crell, and T. Asfour, "Semi-autonomous control of prosthetic hands based on multimodal sensing, human grasp demonstration and user intention," *Robotics and Autonomous Systems*, vol. 154, p. 104123, 2022.

[30] G. Ghazaei, A. Alameer, P. Degenaar, G. Morgan, and K. Nazarpour, "Deep learning-based artificial vision for grasp classification in myoelectric hands," *Journal of neural engineering*, vol. 14, no. 3, p. 036025, 2017.

[31] E. Ragusa, C. Gianoglio, R. Zunino, and P. Gastaldo, "Data-driven video grasping classification for low-power embedded system," in *2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2019, pp. 871–874.

[32] T. Apicella, A. Cavallaro, R. Berta, P. Gastaldo, F. Bellotti, and E. Ragusa, "An affordance detection pipeline for resource-constrained devices," in *2021 28th IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*. IEEE, pp. 1–6.

[33] S. S. Saha, S. S. Sandha, and M. Srivastava, "Machine learning for microcontroller-class hardware-a review," *IEEE Sensors Journal*, 2022.

[34] H. Benmeziane, K. E. Maghraoui, H. Ouarnoughi, S. Niar, M. Wistuba, and N. Wang, "A comprehensive survey on hardware-aware neural architecture search," *arXiv preprint arXiv:2101.09336*, 2021.

[35] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2820–2828.

[36] L. L. Zhang, Y. Yang, Y. Jiang, W. Zhu, and Y. Liu, "Fast hardware-aware neural architecture search," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 692–693.

[37] M. Li, Y. Liu, X. Liu, Q. Sun, X. You, H. Yang, Z. Luan, L. Gan, G. Yang, and D. Qian, "The deep learning compiler: A comprehensive survey," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 3, pp. 708–727, 2020.

[38] C. Li, Z. Yu, Y. Fu, Y. Zhang, Y. Zhao, H. You, Q. Yu, Y. Wang, and Y. Lin, "Hw-nas-bench: Hardware-aware neural architecture search benchmark," *arXiv preprint arXiv:2103.10584*, 2021.

[39] J. Lin, L. Zhu, W.-M. Chen, W.-C. Wang, C. Gan, and S. Han, "On-device training under 256kb memory," in *Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[40] C. Banbury, C. Zhou, I. Fedorov, R. Matas, U. Thakker, D. Gope, V. Janapa Reddi, M. Mattina, and P. Whatmough, "Micronets: Neural network architectures for deploying tinyml applications on commodity microcontrollers," *Proceedings of Machine Learning and Systems*, vol. 3, pp. 517–532, 2021.

[41] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," in *European conference on computer vision*. Springer, 2020, pp. 544–560.

[42] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan *et al.*, "Searching for mobilenetv3," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1314–1324.

[43] W. Jiang, L. Yang, S. Dasgupta, J. Hu, and Y. Shi, "Standing on the shoulders of giants: Hardware and neural architecture co-search with hot start," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 4154–4165, 2020.

[44] Y. Li, P. Wang, R. Li, M. Tao, Z. Liu, and H. Qiao, "A survey of multi-fingered robotic manipulation: Biological results, structural evolvements and learning methods," *Frontiers in Neurorobotics*, p. 53, 2022.

[45] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 077–12 090, 2021.

[46] X. Li, H. Ding, W. Zhang, H. Yuan, J. Pang, G. Cheng, K. Chen, Z. Liu, and C. C. Loy, "Transformer-based visual segmentation: A survey," *arXiv preprint arXiv:2304.09854*, 2023.

[47] X. Hu, K. Yang, L. Fei, and K. Wang, "Acnet: Attention based network to exploit complementary features for rgbd semantic segmentation," in *2019 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2019, pp. 1440–1444.

[48] E. Ragusa, M. P. Ghezzi, R. Zunino, and P. Gastaldo, "Affordance segmentation using rgb-d sensors for application in portable embedded systems," in *Applications in Electronics Pervading Industry, Environment and Society: APPLEPIES 2022*. Springer, 2023, pp. 109–116.