Resource-Aware Object Classification and Segmentation for Semi-Autonomous Grasping with Prosthetic Hands

Felix Hundhausen, Denis Megerle and Tamim Asfour

***Abstract*— Myoelectric control of prosthetic hands relies on electromyographic (EMG) signals captured by usually two surface electrodes attached to the human body in different setups. Controlling the hand by the user requires long training and depends heavily on the robustness of the EMG signals. In this paper, we present a visual perception system to extract scene information for semi-autonomous hand-control that allows minimizing required command complexity and leads to more intuitive and effortless control. We present methods that are optimized towards minimal resource demand to derive scene information from visual data from a camera inside the hand. In particular, we show object classification and semantic segmentation of image data realized by convolutional neural networks (CNNs). We present a system architecture, that takes user feedback into account and thereby improves results. In addition, we present an evolutionary algorithm to optimize CNN architecture regarding accuracy and hardware resource demand. Our evaluation shows classification accuracy of 96.5% and segmentation accuracy of up to 89.5% on an in-hand Arm Cortex-H7 microcontroller running at only 400 MHz.**

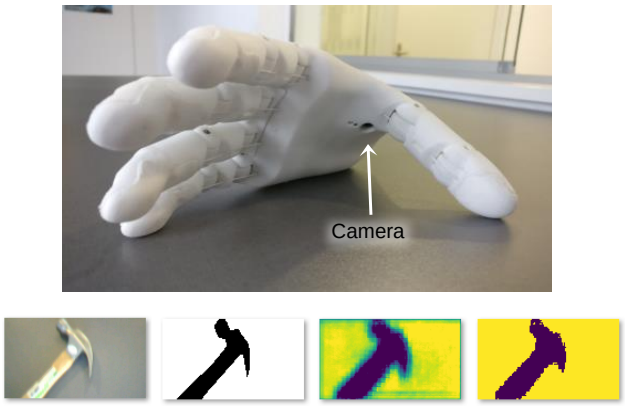


Fig. 1. Top: In-hand camera in the palm of the *KIT Prosthetic Hand*. Bottom: Camera image, ground truth, predicted mask (8 bit) and binary mask

I. Introduction

Recent research in the area of prosthetic hands aims to automate the grasping process using methods coming from robotic grasping ([1], [2], [3], [4], [5]). The system predicts the users intended grasp by perceiving environmental information as well as user-generated actions. Compared to robotic systems, where complete autonomy can be required, the use case of grasping in the context of prosthetics includes communication with the user, which leads to a semi-autonomous control scheme. However, in a typically used 2-channel electromyographic (EMG) interfaces, only primitive commands can be conveyed. Thus, complex hand configurations including control of multiple actuated DoF require a series of consecutive commands which results in delays of the grasping process. Setups with multiple electrodes for higher resolution EMG-pattern were realized in experiments but translating the results to activities of daily living poses several problems [6]. In many cases the signals are not long-time stable, special training is required and the generation of these detailed control pattern increases the cognitive burden for the user.

In this work, we address the problem of semi-autonomous grasping with hand prosthesis, which should be controlled by a few simple user commands instead of complete manual control which could be complex and increases the cognitive burden for the user, especially for control of multi-DoF hands. Therefore, the in-hand embedded system captures visual environmental information, that is intended to be evaluated in real-time. The captured visual scene information allows selecting a suitable offline design with possible online parametrization. This result in autonomous finger trajectory execution, adaptation of hand orientation and control of forces to be applied without further user interaction, when prior knowledge about object properties like size, shape as well as weight is available. Feedback loops are then shifted from the user to the prostheses. This allows to minimize the focus of the user on grasp control, gives a chance to speed up the grasping process and improving the reliability of grasps.

For scene interpretation often highly resource intensive algorithms with demand for powerful processing hardware are used. However, it is favorable to realize all data processing device-embedded, known as edge computing in literature [7]. The advantage of embedded processing is the lack of dependency on wireless network availability and uptime of external services.

The limited resources in embedded hardware require efficient algorithms optimized towards low resource demand. This includes a reduction of memory demand as well as a minimized number of operations. Algorithm complexity and resource-awareness are both important to guarantee real-time performance, which is critical since reaction time plays an important role in control of prosthesis.

The prototypical system presented in this work is realized

This work has been supported by the German Federal Ministry of Education and Research (BMBF) under the project INOPRO (16SV7665) and by the German Research Foundation (DFG) within the the Transregional Collaborative Research Centre Invasive Computing (SFB/TR 89).

The authors are with the Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology, Karlsruhe, Germany. {felix.hundhausen, asfour}@kit.edu

on the Embedded System of the second generation of the *KIT Prosthetic Hand* (see Fig. 1 and [8]). The hand includes an embedded Arm Cortex-H7 processor with a connected miniature camera and a color display on the hand backside for user feedback. For experiments, we emulate a 2-channel EMG control.

In this paper, we present a resource-aware system for classification and localization of objects based on visual information obtained by an in-hand camera of an intelligent prosthetic hand. The system combines an optimized object classification network with an adaptable semantic segmentation network. In case of a user confirming correct classification, segmentation can be performed class-aware using an encoder-decoder network trained on a particular class. For incorrectly classified and unknown objects the network uses weights trained on the set of known objects. Highest accuracy results are achieved for object-aware trained segmentation. For unknown objects the accuracy drops but can still provide good accuracies.

The major contribution of this work is the realization of the related methods on a resource-limited embedded system. We present methods for optimization of the network architecture and investigate resource-aware deployment enabling execution on constrained systems. Our methods enable the use of cost and energy-efficient hardware platforms, which is an important step towards the realization of higher levels of autonomy in prosthetic hands.

The paper is structured as follows: In section II we give an overview of related work, in section III we describe the design of our system including the classification and segmentation network as well as the optimization of the network architecture. We evaluate the system in section IV where we analyze the system performance and describe its limitations. Finally, we conclude our work in section V.

II. Related Work

In research, different prosthetic hands using computer vision systems can be found. In the following we discuss different approaches addressing the design of visual perception systems for prosthetic hands and show state of the art in the field of embedded implementation of image processing algorithms especially in resource constrained environments.

One approach followed by [1], [2] and [3] is to directly classify the object from visual data into different *grasp categories*, each with a specifically assigned grasp type. Došen et al. [1] use a simple threshold value to segment objects from the background. The segmentation together with a depth value measured by an ultrasonic sensor is used to determine the longest and shortest object axis and select the grasp type (pinch, lateral, palmar or spherical) by using if-then rules. Ghazaei et al. [2] design a convolutional neural network to classify images from an image library into 4 grasp categories. They conclude that a 2 layer network leads to best results, however, this is presumably the case since the 32×32 images from the dataset do not include background and are captured in a controlled setup. In recent work DeGol et al. [3] classify RGB images with a 640×480 captured by a hand-internal camera using a deep neural network. The inference of this network is realized on a hand-external NVIDIA Tegra GPU. For training of the network, the authors create their own dataset but also use ImageNet dataset as training data which leads to significantly higher classification accuracy.

The design of artificial neural networks for classification and segmentation tasks is a highly active field of research. Modern deep neural networks include billions of multiply-accumulate (MAC) operations. For semantic segmentation of images, neural networks as [9], [10], [11] and [12] provide promising results. However, due to a large number of operations these networks are not suitable for deployment on resource-constrained embedded systems. Targeting mobile platforms, networks are adapted for mobile devices like smartphones [13], but still require over 1 M parameters and over 100 million MAC operations. Microcontroller based architectures as, for example, implemented in the *KIT Prosthetic Hand* only provide memory in the size about 1 MB and can execute only a few million MAC operations per second. The deployment of these types of platforms requires smaller, task-specific application which can be realized using a lower number of filters and layers.

To automatically create optimized network architectures, often reinforcement learning methods or evolutionary algorithms are used. [14] implements a learning agent and experience replay for reinforcement learning of convolutional neural networks, designing novel network structures. [15] and [16] serialize networks as genome and apply serialization-sensitive adapted evolutionary algorithms, both showing that model architectures still leave tremendous room for improvement.

State of the art deep networks are often optimized towards maximal accuracy while ignoring resource demand. However, when precision of filters and layer-data is reduced, memory footprint can decrease and improve runtime while having only a small impact on output accuracy. For example, [17] proposed switching to integer-arithmetic only inference, where near-identical accuracy levels are restorable after quantization, decreasing the model size by a factor of 4. Additionally, deep neural networks often contain redundant or negligible structures. It is beneficial to evaluate connection impact on the inference result to remove low impact weights [18] what results in increased generalization and lower runtime.

III. System Design

The system is designed to determine object class as well as a segmentation of the object in the camera image, which allows to obtain the object's orientation and shape. This information can then be used by a grasp control unit to determine and execute a suitable grasp with optimized hand orientation, finger trajectories and forces. We designed the system to allow extracting the above information from images captured by the in-hand camera only. The user shows his intent to grasp an object using an initial command when the object to be grasped is in view of the camera. The input

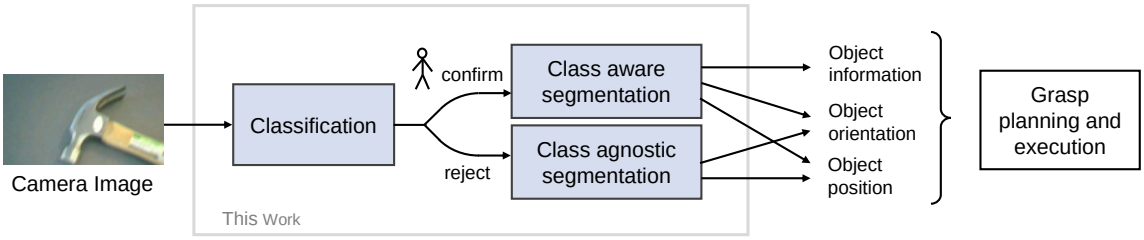


Fig. 2. Overview of the System-Architecture. The system consist out of a classification network followed by a segmentation network. In case of successful classification (class-aware), the use of a class-particular parametrized network is possible. In case of unknown class, a fall-back network for class agnostic segmentation is inferred.

image is first processed by an object classification block that outputs a vector of object class probabilities. Object classes are learned offline and include frequently used user-selected objects. The object class assigned with the highest probability is suggested to the user who can confirm the result by binary EMG-signal as feedback. Based on the result, an object-specific grasp can be executed. The second part of the system performs a semantic segmentation of the object in the camera image. This provides information about the orientation of known objects (*class aware*), in case of unidentified object class, the segmentation output (*class agnostic*) can be used for shape based grasp selection as well as hand orientation. When inferring class aware segmentation, object-specific trained parameters can be used, when the object class is unknown and incorrectly classified class agnostic segmentation uses parameters trained on the complete training data set. Since unknown objects can be segmented using class agnostic segmentation, there is no need to include all objects from daily life in the system, but only a subset of relevant ones. For those objects, a finer segmentation and object class, which allows object specific grasp parametrization, are provided. An overview of the system architecture is given in Fig. 2.

A. Classification

Classification is realized by a deep convolutional network optimized towards inference on the embedded processor. For our application of object classification two aspects are relevant: A set of known objects must be recognized with high accuracy within given real-time constraints. Taking considerations from [19] into account, we see a maximum of $\approx 150\ ms$ as an acceptable value for recognition.

1) Network Architecture Synthesis: To achieve real-time network inference, the classification network architecture is synthesized by using an evolutionary algorithm inspired by the algorithm proposed in [15]. The algorithm uses multiple evolution steps in which the algorithms evaluated the fitness of all networks, breeds offspring networks with crossover characteristics from two parent networks of high fitness and randomly mutates segment parameters.

Since real-time constraints and given hardware resources allow a maximum number of operations at which the highest accuracy can be expected, we modify the fitness function to target a given number of operations. Since the convolution layers are responsible for most of the resulting operations as stated in [20], only the operations resulting from convolution are regarded. The number of MAC operations per convolution layer can be calculated as

$$\beta_{\text{conv}} = \frac{I_H \times I_W \times I_C \times K_H \times K_W \times O_C}{S^2} \tag{1}$$

with input I, kernel K, output O as well as height, width and channels H, W, C. S is the filter stride. We derive the fitness of a network combining number of operations and accuracy by designing a fitness function using a generalised logistic function, in detail

$$\mathcal{F}(\alpha_{\text{final}}, \beta_{\text{conv}}) = \alpha_{\text{final}} + (1 + e^{\beta_{\text{conv}} - (1+\nu)})^{-\frac{1}{2+\nu}} \tag{2}$$

where α_{final} is the converged network accuracy and ν the target amount of MAC operations in millions, in our case, $\nu = 2.0$. The function is designed so that $0.0 < \mathcal{F}(\alpha_{\text{final}}, \beta_{\text{conv}}) < 2.0$ and strictly monotonously increases for smaller β_{conv} and ν. Fitness strongly decreases for for $\beta_{conv} > \nu$ to penalise higher number of operations than the preferred ν.

For mutation and crossover, CNN architectures are serialized into segments depicted in Fig. 4(a). Each segment consists of a convolutional, batch normalization, pooling and ReLu layer. This implementation proved to be extensible as well as easy to modify. The crossover process of two CNNs is depicted in 4(b). Two parents networks create two offspring networks, carrying on favorable traits of their parents. Mutation is implemented as random change in hyperparameters, including the adjustment of filters, kernels, strides, pooling types in a predefined range and even the complete removal of segments. The initial pool consists out of 20 randomly initialize network architectures.

As for training, a dataset of 13 classes, 300 samples per class annotated with class labels is used. This dataset is split $70/30$ for training and testing. The training data set is augmented by segmenting the background and replacing it with random pixel values. Additionally, we use flipping, cropping, affine translation and random Gaussian noise to further augment training images. The performance of each network is evaluated by training upon convergence until accuracy delta is smaller then 1% for 3 epochs. To select the final network architecture, once the generations do not improve, the highest fitness architecture with an accuracy higher than a threshold (98.5 %) was taken from the pool.

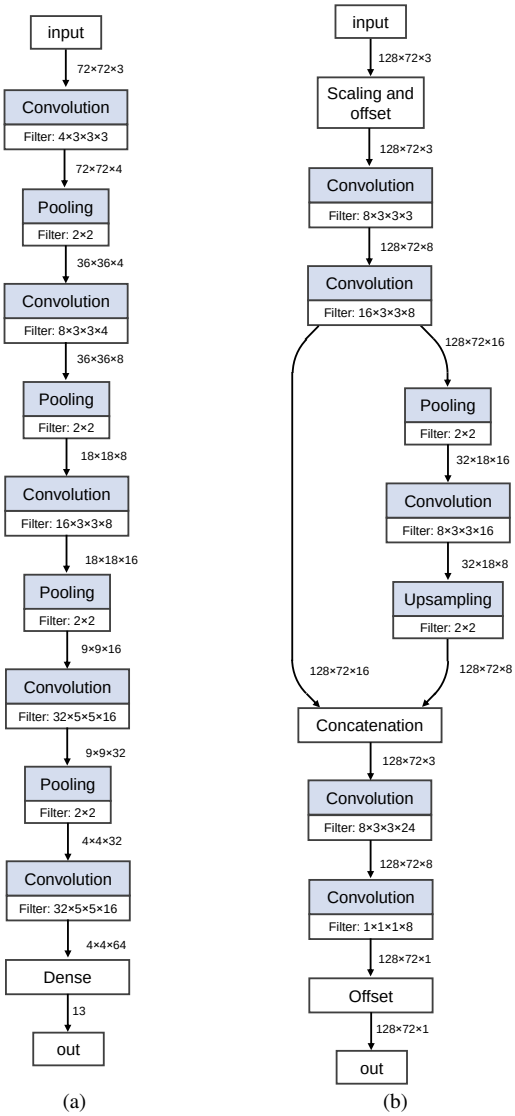
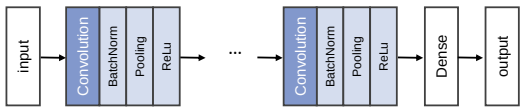


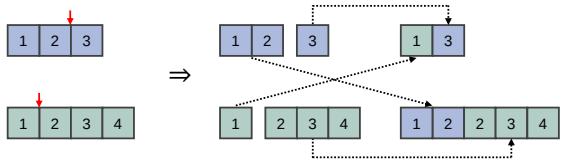
Fig. 3. Classification (a) and segmentation (b) network architecture

2) Deployment Optimization: For deployment of the classification network, we use the CMSIS:NN library[21] which offers Cortex-H7 optimized network kernels, especially exploiting SIMD instructions and includes a quantization framework. For efficient inference, convolution and pooling filter and kernel sizes are adapted to the next suitable dimension supported by the library. The resulting and finally deployed architecture is depicted in Fig. 3(a).

The model obtained by the evolutionary algorithm is further trained using a decaying learning rate starting at 0.01 until convergence. After training of the network, we fuse batch normalization into preceding convolutions, to avoid operations needed for training in inference. Iterative pruning is employed to further enhance performance until the accuracy decreases by 0.1 %. Then, we find the optimal quantization range with 8 bit precision by applying quantization sweeps as described in [21].



(a) Serialization of CNNs for evolution, the input passes an arbitrary amount of segments, each segment is defined as Convolution-BatchNorm-Pool-ReLu block. After feature extraction, a final fully connected layer serves as a classifier.



(b) Breeding two child CNNs from two parent CNNs. Each block represents a whole segment. Random cut locations are determined (red arrows) and according to these, the networks are intertwined.

Fig. 4. Serialization and crossover for the evolutionary algorithm.

B. Segmentation

For semantic segmentation, the captured camera image as well as the user evaluated result of the classification network is used as input. The output is a binary pixel-wise segmentation of the object in the input image. The layers of the network are shown in Fig. 3(b). The network follows the structure of an encoder-decoder network. Encoder and decoder are connected by a residual connection. This connection short-cuts a down-sampling layer and a convolution-layer with reduced layer sizes. The decoder network is using deconvolution layers realized by an up-sampling layer which is concatenated by the layer before down-sampling. The concatenated layers are merged by using depth-wise convolution. The final pixel-wise result is generated by applying a binary threshold to the output image.

As a dataset for the training of the network, we captured a set of 100 images per object class. The labels were generated manually by creating a ground truth mask for every image. This results in a set of 1.300 available images for the training and test dataset which was split in a relation of $1/3$ to $2/3$.

The network is trained using a loss function to minimize binary cross-entropy for the predicted pixel class

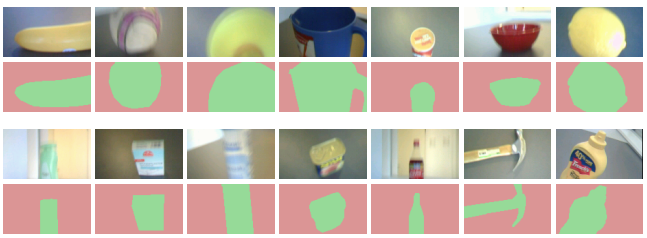


Fig. 5. For offline training of the system, RGB-images were captured by the in-hand miniature camera and transferred to a PC. For training 1.300 images of 13 object-classes from the YCB and KIT object model database were recorded. The images are annotated with binary labels, providing ground truth for the training of the segmentation network. Bottom right: Example from the afterwards captured holdout dataset (YCB mustard bottle)

probability and ground truth. To take later quantization of weights and layers into account, the network is trained using fake quantization layers that allow quantization aware training. During training the network weights are stored as floating points and allow high resolution for the backward propagation algorithm, while for evaluation, quantization is simulated [17]. Training is stopped when the filtered loss improvement falls under a certain threshold. For quantized deployment, the network is converted using a converter provided by Tensorflow Lite. This reduces all weights to 8-bit resolution taking into account minimum and maximum values. The converted weights are exported as constants to be flashed to microcontroller static memory.

The network structure is common for segmentation of different objects but parametrized by weights trained on different training sets. For known object classes the network parameters are learned exclusively from the corresponding object class. In case of unknown objects, the network weights trained on all other object classes. In case of a rejected segmentation of a known object class, also the class agnostic network is inferred.

IV. Evaluation

To evaluate our system we discuss results for the classification network architecture obtained using the proposed evolutionary algorithm. Further, we evaluate the accuracy of the segmentation network on the test dataset and an additionally generated holdback dataset consisting out of 5 additional object classes.

A. Classification Results

Results of the evolutionary algorithm used for the design of the optimized classification network architecture, as average accuracy and MAC per generation, are depicted in Fig. 6. The results show, that evolution as expected decreases the operation count while increasing accuracy. The operation count, according to the target count defined in equation 2, converges towards ν. Since accuracy and amount of operations improved evenly, the fitness function does not overrate either one and is therefore beneficial. On a GTX970 GPU the training per generation takes approximately 2 hours. We observe that with increasing number of generations the variance in the models is decreasing, network architectures tend to converge and share similar layer setups. After 10 generations progress significantly slows down and first generations do not increase the best fitness. This suggests that a local extremum is reached after a short evolution time. We believe that hyperparameters of the evolutionary algorithm, especially the mutation rate, need to be further adjusted to ensure higher variance. More importantly though, the results indicate that the chosen breeding function in fact carries on favorable characteristics. Results show that evolutionary architecture synthesis does not reach an optimal architecture, but provides a good result without manual hyperparameter optimization. The algorithm allows reducing execution time of networks which can be created automatically, allowing the creation of user specific network architectures for individual requirements.

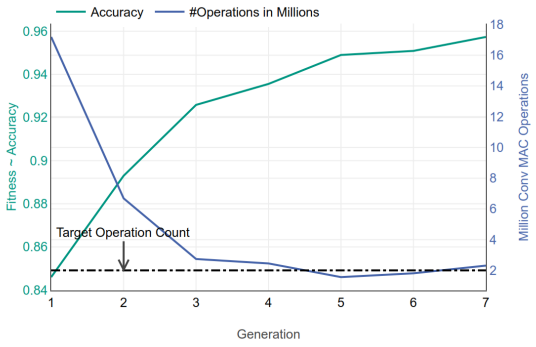


Fig. 6. Average accuracy and MAC operations in millions per generation. The evolutionary algorithm successfully generates networks that have increased accuracy and lower operation count. The convergence of operations to 2 is a result of the chosen fitness function.

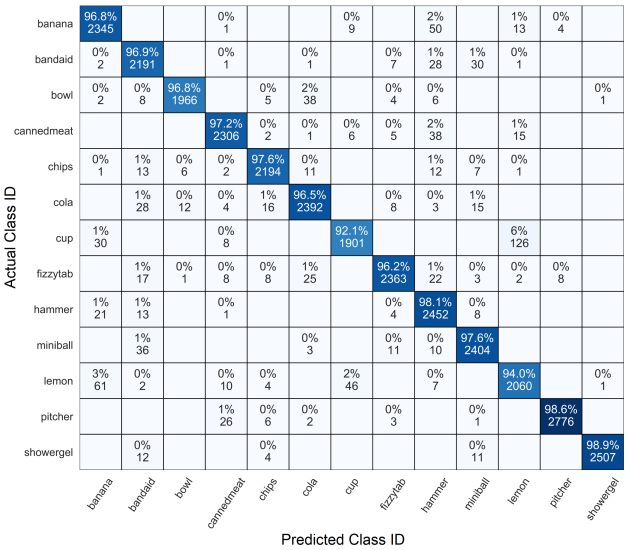


Fig. 7. Confusion matrix of the classification network. All objects are from the YCB object set.

The resulting architecture with the highest test accuracy on our dataset consists of 4 convolutional layers, subsampling the $72\times72\times3$ input image after each convolution and uses a combination of maximum and average pooling. Subsequently, a fully connected layer yields the classification output.

The synthesized network architecture, after fusing the batch normalization, achieves an accuracy of 96.75 % on a separately captured evaluation set including 100 samples per class. With weights, bias and activations quantized to 8 bit, we achieve an accuracy of 96.51 %. The classification results and the confusion matrix is provided in Fig. 7.

Over the course of quantization, accuracy only dropped 0.4 % reducing the weight and activation footprint by a factor of 4.

An overview of run-time and data usage per layer is provided in table I. In total, the network achieves a real-time satisfying inference in 115 ms, classifying 13 objects.

Layer Type	Filter Shape	Output Shape	Ops	Runtime -O1 quantized kernel (not) optimized
Input	N.A.	$72 \times 72 \times 3$ (15.6 kB)	N.A.	N.A.
Convolution	$3 \times 3 \times 3 \times 4$ (0.1 kB)	$72 \times 72 \times 4$ (20.3 kB)	0.6 M	(110 ms) 40 ms
MaxPooling	N.A.	$36 \times 36 \times 4$ (5.1 kB)	23.3 k	(8 ms) 3 ms
Convolution	$3 \times 3 \times 4 \times 8$ (0.3 kB)	$36 \times 36 \times 8$ (10.2 kB)	0.4 M	(76 ms) 13 ms
MaxPooling	N.A.	$18 \times 18 \times 8$ (2.5 kB)	11.7 k	(4 ms) 1 ms
Convolution	$3 \times 3 \times 8 \times 16$ (1.1 kB)	$18 \times 18 \times 16$ (5.1 kB)	0.4 M	(60 ms) 10 ms
MaxPooling	N.A.	$9 \times 9 \times 16$ (1.3 kB)	11.7 k	(2 ms) <1 ms
Convolution	$5 \times 5 \times 16 \times 32$ (12.8 kB)	$9 \times 9 \times 32$ (2.5 kB)	1.1 M	(135 ms) 24 ms
AvgPooling	N.A.	$4 \times 4 \times 32$ (0.5 kB)	2.3 k	(1 ms) 1 ms
Convolution	$5 \times 5 \times 32 \times 64$ (51.2 kB)	$4 \times 4 \times 64$ (1 kB)	0.8 M	(96 ms) 23 ms
AvgPooling	N.A.	$2 \times 2 \times 64$ (0.3 kB)	1.2 k	(<1 ms) <1 ms
Dense	$2 \times 2 \times 64 \times 13$ (3.3 kB)	13 (13 B)	1.2 K	(<1 ms) <1 ms
Total	68.9 kB weights	48.8 kB activations	3.2 M	(492.7 ms) 115 ms

TABLE I

CLASSIFICATION NETWORK RUNTIME AND MEMORY BREAKDOWN

Thereforee, it uses 69 k parameters, accounting for 69 kB of memory and 44 kB RAM usage.

B. Segmentation Results

To show the benefits of our approach in combining object classification and selecting weight parameters based on the classification result, we compare the accuracy of segmentation results on the test data set for class aware and class agnostic segmentation.

The average accuracy of the resulting object mask is compared to ground what gives true/false positive and negative pixel-wise classification. The mean accuracy is calculated over all image-pixels:

$$acc_{mean} = \frac{\sum_{h=1}^{I_H} \sum_{w=1}^{I_W} \frac{\mathrm{tp}_{h,w} + \mathrm{tn}_{h,w}}{\mathrm{tp}_{h,w} + \mathrm{fn}_{h,w} + \mathrm{fp}_{h,w} + \mathrm{tn}_{h,w}}}{I_H \times I_W} \quad (3)$$

For object class agnostic segmentation the accuracy ranges from 78.0 % to 86.4 % with a mean of 82.8 %. With awareness about class, results improve for all objects. Class aware segmentation results from 81.3 % to 95.2 %, the mean is at 89.5 %, this exceeds results for class agnostic segmentation by 8.1 %. Detailed results can be found in table II

The effects of quantization on the network accuracy are measured by first evaluating the direct network output before applying the threshold for obtaining the binary output mask. The error is obtained by averaging the pixel-wise error which is calculated as

$$error_{mean} = \sum_{h=1}^{I_H} \sum_{w=1}^{I_W} \frac{out_{h,w} - out\_quantized_{h,w}}{out_{h,w}} \quad (4)$$

For the segmentation test set we obtain an average error of $error_{mean}$ as 2.33 %. Since small changes do not directly influence the output segmentation mask due to the applied threshold, we separately calculate the accuracy of the quantized result on the un-quantized network. Here an accuracy of 94.7 % is obtained. The number of MAC operations is 29 million, showing the need for further optimization, since it exceeds the time budget enforced by real-time requirements.

Following the design of the system, we evaluate the accuracy on additionally recorded data which we use as our holdout dataset. To this end, we recorded image data from 5 unknown objects using a handheld prosthesis, with 5 images and manually annotated ground truth masks per evaluation object. All unknown objects are not included in training and test dataset. The output is predicted by the class agnostic segmentation network trained on all 13 objects from the training data set. The resulting class-wise mean accuracy (see 3) is included in table II.

Object class	Class agnostic segmentation acc_{mean}	Class aware segmentation acc_{mean}	Improvement by using classification result
Banana	0.864	0.928	7.4 %
Ball	0.799	0.813	1.8 %
Cup	0.849	0.952	11.7 %
Pitcher	0.801	0.913	15 %
Chips	0.83	0.883	6.7 %
Bowl	0.839	0.936	11.6 %
Lemon	0.821	0.942	13.8 %
Shampoo	0.831	0.908	9.6 %
Bandaids	0.82	0.854	4.1 %
Fizzies	0.829	0.897	8.5 %
Spam	0.869	0.917	5.6 %
Cola	0.78	0.823	5.4 %
Hammer	0.834	0.865	3.7 %
Mean	0.828	0.895	8.1 %
Holdout:			
Baseball	0.882	-	-
Green cup	0.934	-	-
Mustard Bottle	0.839	-	-
Salt Box	0.906	-	-
Sugar Box	0.893	-	-

TABLE II

ACCURACY ON TEST DATASET

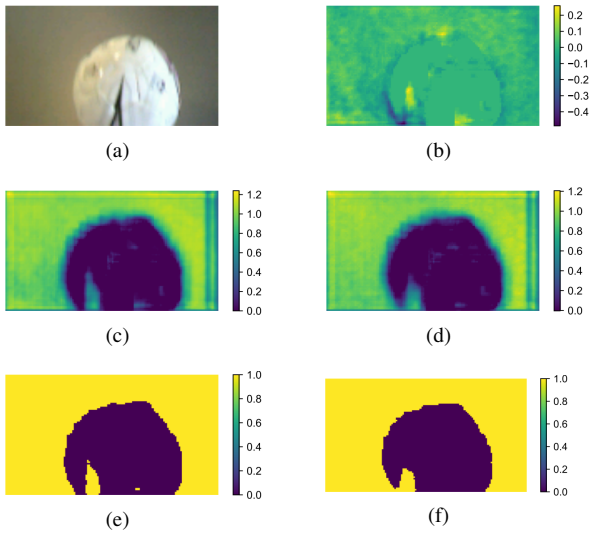


Fig. 8. Outputs from original (c) and quantized (d) segmentation network for an exemplary input image of the ball (a). The pixel wise quantization induced error is shown in (b). In (e) and (f) the output mask obtained by applying a binary threshold is shown with and without network quantization.

C. Limitations and Future Work

The number of recognizable object classes is restricted by the network size. We proofed good classification results on a

set of 13 objects which satisfies real-time constraints on our current hardware platform. An increase of the number of object classes with currently implemented architecture, would lead to a decreasing accuracy. The increase of network-size, namely number of filters, can counteract this problem. Since real-time constraints do not allow increasing the number of operations for the algorithm, this does not provide a valid solution. Especially for semantic segmentation, a high need for computational power was identified, which cannot satisfy real-time requirements with current hardware. However, the designed system can be expected to scale well in the case of upgraded hardware resources.

A limitation of our currently implemented system is that only tabletop scenarios are regarded. We do not train or test on cluttered scenes since this is not focus of our work at this stage. However, this will be investigated in the future to allow robust real-world application. In addition, we will work on including multiple object instances per object class. This will allow us e. g. defining the category of bottles instead of one specific type of bottle.

V. Conclusion

In this paper, we presented a system for scene information extraction from visual data captured by a camera integrated into a prosthetic hand. To this end, we designed, implemented and tested a system combining object classification and semantic segmentation. The system includes user feedback after an initial classification by which the user can confirm or reject the classification result. In the case of a confirmed classification a class aware segmentation is inferred. Segmentation is realized by a convolutional encoder-decoder network. The output of the system includes multiple components: Object information can be derived from the identified class while the segmentation mask allows estimation of object orientation and position. We describe the methods used to implement our system in a resource-constrained computing environment. Therefore we propose an adaptation of an evolutionary algorithm for optimizing the classification network architecture taking limited hardware-resources into account. We apply methods for optimized implementation including quantization, pruning and use of optimized kernels for network deployment on an embedded Arm processor.

To evaluate the designed system, we test the performance after training with a dataset generated by using the in-hand camera. We include evaluation of required hardware resources and evaluate the accuracy of classification. Segmentation is evaluated by analyzing the classification result on an unseen test dataset. We further compare the results of the un-quantized trained network with the quantized version for deployment and observe only minor deviation.

To conclude, we presented a resource-aware system that allows gathering relevant scene information required for the automatic generation of grasp candidates by the KIT Prosthetic Hand. Our system design allows in-device implementation which we consider important for realizing integrated prosthetic hands with semi-autonomous grasping abilities.

References

[1] 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," *Journal of neuroengineering and rehabilitation*, vol. 7, no. 1, p. 42, 2010.

[2] G. Ghazaei, A. Alameer, P. Degenaar, G. Morgan, and K. Nazarpour, "An exploratory study on the use of convolutional neural networks for object grasp classification," 2015.

[3] J. DeGol, A. Akhtar, B. Manja, and T. Bretl, "Automatic grasp selection using a camera in a hand prosthesis," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 431–434, IEEE, 2016.

[4] M. Markovic, S. Došen, 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.

[5] M. Esponda and T. M. Howard, "Adaptive grasp control through multimodal interactions for assistive prosthetic devices," *arXiv preprint arXiv:1810.07899*, 2018.

[6] D. Farina, N. Jiang, H. Rehbaum, A. Holobar, B. Graimann, H. Dietl, and O. C. Aszmann, "The extraction of neural information from the surface emg for the control of upper-limb prostheses: emerging avenues and challenges," *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, vol. 22, no. 4, pp. 797–809, 2014.

[7] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[8] P. Weiner, J. Starke, F. Hundhausen, J. Beil, and T. Asfour, "The KIT Prosthetic Hand: Design and Control," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2018.

[9] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, pp. 818–833, Springer, 2014.

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[11] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[12] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.

[13] 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," *arXiv preprint arXiv:1704.04861*, 2017.

[14] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.

[15] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Automatically designing cnn architectures using genetic algorithm for image classification," *arXiv preprint arXiv:1808.03818*, 2018.

[16] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 497–504, ACM, 2017.

[17] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2704–2713, 2018.

[18] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, pp. 598–605, 1990.

[19] T. R. Farrell and R. F. Weir, "The optimal controller delay for myoelectric prostheses," *IEEE Transactions on neural systems and rehabilitation engineering*, vol. 15, no. 1, pp. 111–118, 2007.

[20] L. Lai, N. Suda, and V. Chandra, "Not all ops are created equal!," *arXiv preprint arXiv:1801.04326*, 2018.

[21] L. Lai, N. Suda, and V. Chandra, "Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus," *arXiv preprint arXiv:1801.06601*, 2018.