Fully Convolutional One–Shot Object Segmentation for Industrial Robotics

Benjamin Schnieders†
University of Liverpool
Liverpool, United Kingdom
Benjamin.Schnieders@liverpool.ac.uk

Shan Luo
University of Liverpool
Liverpool, United Kingdom
Shan.Luo@liverpool.ac.uk

Gregory Palmer
University of Liverpool
Liverpool, United Kingdom
G.J.Palmer@liverpool.ac.uk

Karl Tuyls
University of Liverpool
Liverpool, United Kingdom
K.Tuyls@liverpool.ac.uk

ABSTRACT

The ability to identify and localize new objects robustly and effectively is vital for robotic grasping and manipulation in warehouses or smart factories. Deep convolutional neural networks (DCNNs) have achieved the state-of-the-art performance on established image datasets for object detection and segmentation. However, applying DCNNs in dynamic industrial scenarios, e.g., warehouses and autonomous production, remains a challenging problem. DCNNs quickly become ineffective when tasked with detecting objects that they have not been trained on. Given that re-training using the latest data is time consuming, DCNNs cannot meet the requirement of the *Factory of the Future (FoF)* regarding rapid development and production cycles. To address this problem, we propose a novel one-shot object segmentation framework, using a fully convolutional Siamese network architecture, to detect previously unknown objects based on a single prototype image. We turn to multi-task learning to reduce training time and improve classification accuracy. Furthermore, we introduce a novel approach to automatically cluster the learnt feature space representation in a weakly supervised manner. We test the proposed framework on the RoboCup@Work dataset, simulating requirements for the *FoF*. Results show that the trained network on average identifies 73% of previously unseen objects correctly from a single example image. Correctly identified objects are estimated to have a 87.53% successful pick-up rate. Finally, multi-task learning lowers the convergence time by up to 33%, and increases accuracy by 2.99%.

KEYWORDS

One-Shot Segmentation, Fully Convolutional Network, Industrial Robotics

ACM Reference Format:
Benjamin Schnieders†, Shan Luo, Gregory Palmer, and Karl Tuyls. 2019. Fully Convolutional One–Shot Object Segmentation for Industrial Robotics. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019*, IFAAMAS, 9 pages.

This paper is dedicated to the memory of our wonderful colleague and friend Benjamin Schnieders, a bright young scientist, who recently passed away.

1 INTRODUCTION

The Factory of the Future [18] imagines interconnected robots working alongside and in cooperation with humans. Tasks to be performed range from being in direct contact with humans [8] during assistance with repetitive or potentially dangerous tasks, to completely autonomous operations, in which mobile robots connect various other, pre-existing robotic workflows. To ensure these autonomous operations run without disturbance for as long as possible, detecting objects and their pickup point robustly is of high priority [20].

While Deep Convolutional Neural Networks (DCNNs) form the state-of-the-art in object detection [29], they typically suffer from long training periods. Rapid prototyping production, as in the Factory of the Future, can require turnaround times shorter than the training times of common neural networks. In this situation, the robot may be required to pick up produced objects before the network has finished training, halting the entire workflow. While many techniques exist to speed up the convergence of neural networks [27], dataset acquisition and labeling poses an additional delay. Thus, for workflows of a smaller batch size, training a robot to detect the produced objects can be infeasible due to temporal constraints.

Training a network to generalize the learned knowledge of recognizing pre-trained objects to correctly recognize untrained classes is the subject of *few-*, *one-*, and *zero*-shot object detection [17]. Using *few-* or *one*-shot detection, a robot can be shown one or a few examples of a prototype object, which it can then re-detect during autonomous operations.

Figure 1 showcases a model workflow of rapid prototyping production. Training the network becomes a bottleneck, as production has to be delayed until the robot can learn to detect, and thus to manipulate, the produced objects. Using a one-shot detection approach alleviates the bottleneck, and thus allows for faster design/production cycles.

While object detection networks are commonly trained to produce bounding boxes around the detected objects [26, 27], object segmentation approaches segment the image pixelwise into semantic classes, i.e., which object class each pixel belongs to. This segmentation output allows the robot to compute an object *pickup point*, and subsequently, to pick it up [30]. The state-of-the-art approaches in semantic segmentation typically build on fully convolutional neural networks [19]. Combining the semantic segmentation

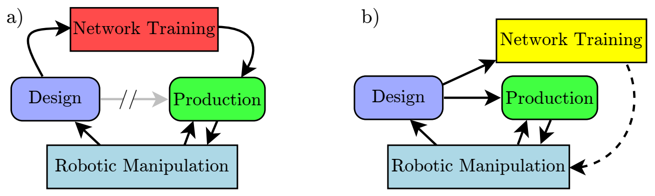

Figure 1: A simplified automated production workflow. a) Production of newly designed parts has to be delayed until the robot that manipulates them after production is trained. b) The manipulation/production cycle can begin with a one-shot detection network, while the new information is learned. Once training is complete, the robot can use the updated network.

approach with one-shot learning leads to a one-shot segmentation network. Such a network is trained to produce a per-pixel classification of whether each pixel shows part of a previously shown prototype object or not.

Research questions addressed in this paper are the following:

i To what extent can DCNNs extrapolate their learned object segmentation to unknown data?
ii How much does transforming the optimization process into a multi-task problem benefit the training time?

This paper introduces a method to rapidly train a one-shot object segmentation network to detect industrial objects based on a single prototype image. The method is validated on the new, publicly available[1] RoboCup@Work dataset [30], which includes objects present in a Factory of the Future simulation [16]. Results illustrate that our network architecture can identify previously unseen objects in a Factory of the Future simulation with 73% accuracy after being trained for only 7,000 iterations. As little as 4,000 iterations, or approximately 57 minutes of training on a NVIDIA 1080 Ti, suffice to re-train the network with all information, producing a 94% accuracy.

Our main contributions can thus be summarized as follows:

i A novel framework is proposed for rapidly training Siamese network based, fully convolutional networks to perform one-shot segmentation in industrial scenarios[2];
ii An innovative error metric supporting auto-clustering of objects' representations in feature space is introduced to improve convergence speed;
iii An auxiliary network architecture is showcased to automatically combine the different error metrics in order to reduce hyperparameters.

The remainder of this paper is organized as follows: Section 2 provides the background on one-shot object detection and segmentation and discusses related work. Section 3 describes the network architecture used, and how the different tasks are trained simultaneously. Section 4 describes the experimental setup. Finally, Section 5 presents and compares the experimental results, and Section 6 concludes the paper.

[1]http://wordpress.csc.liv.ac.uk/smartlab/objectdetectionwork/
[2]We make our code available online: https://github.com/smARTLab-liv/ObjectDetect

2 BACKGROUND AND RELATED WORK

While general one-shot learning is well studied [28], one-shot object detection and segmentation is a very recent field of research [32]. While conventional object detection [7] and segmentation [19] neural networks operate only on classes the network has been trained on, the one-shot variants extrapolate the learned information to extract a previously unseen object class, of which only one example is shown. Research fields closest related to one-shot segmentation can be roughly subdivided into two categories, each attempting to solve this challenging task in a specific way.

Object co-segmentation is a weakly or entirely unsupervised approach to learn to extract objects based on visual similarity within a class of objects. Typically, image level classes are presented, i.e., the network is trained with the knowledge that presented images belong to the same class, and can learn to detect similar features [24, 25, 33]. However, while research in this field is able to detect an object based on the input of another image, co-segmentation networks are not designed to abstract the learned knowledge such that it can be transferred to previously unseen objects. Furthermore, the unsupervised nature of the approach requires large datasets and comparatively long training periods.

Siamese networks have been employed to efficiently perform one-shot object classification [35] or detection [13]. Siamese networks are trained to map image data onto a dense feature vector on a much smaller space, in a way that objects from the same category are mapped onto feature space positions close to each other with respect to a given a distance metric. This distance metric can be automatically learned [31]. Michaelis *et al.* [23] have shown Siamese networks to be able to perform semantic segmentation of color-coded symbols in cluttered environments.

The work of Shaban *et al.* [32] is the current state-of-the-art method to perform one-shot object segmentation, outperforming competitive approaches on the Pascal VOC dataset. However, we identify a number of issues preventing it from being employed effectively in a *Factory of the Future* environment: First, the authors of [32] assume that there is no overlap in classes between training and test set. While no class of the test set should appear in the training set, the reverse does not hold: Trained classes should be featured as distractors in the evaluation set as well [23]. As segmentation networks tend towards extracting known data, this assumption could prohibit any occurrence of trained objects in workflows exploiting one-shot detection, making it unfit in an industrial context. Secondly, the approach requires a segmentation mask to be applied to the image including the query object. In an autonomous production environment, a human-generated segmentation mask is unlikely to exist. The third identified issue is the high complexity of the resulting network. While this produces good results on the real-world Pascal VOC dataset, it is likely to suffer from exceedingly long training periods. This assumption is confirmed by [32] stating that memory constraints only allow for a batch size of 1, and training is done with a learning rate of 1×10^{-10}, a combination producing extremely long training periods. In contrast, our approach is tested on extracting an unknown object class from a set of at least two unknown classes and up to ten known classes. It is able to extract the object to be queried without the need for a human generated segmentation mask. Most importantly, the network architecture

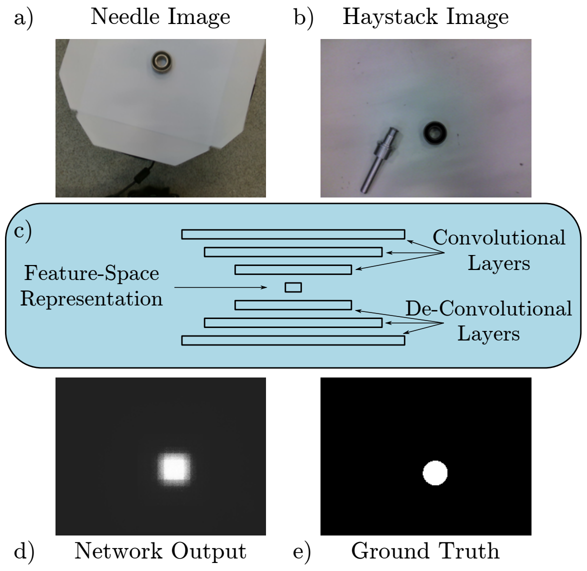

Figure 2: The general principle of a network performing one-shot object segmentation. A *needle* image a) and a *haystack* image b) are presented to the network c), which in turns extracts the *needle* object from the *haystack* image. The network output d) and the ground truth e) are compared and the error is propagated back. Evaluation is performed on object classes the network has not been trained on.

proposed in this paper can be trained quickly, with convergence in as little as 7,000 iterations for one-shot detection, or about 100 minutes of training on a NVIDIA 1080 Ti. If only recollection of known objects is required, re-training to an accuracy of 94% can be achieved in 4,000 iterations, or 57 minutes of training.

3 APPROACH

Our approach is, on its most basic level, composed of a deep convolutional neural network, which is trained on images featuring objects. For every training iteration, it is presented with an image showing a single object, and an image featuring multiple objects. As per convention in search and image retrieval [4], we call the single object to be retrieved the *needle* object, and the corresponding image the *needle* image. The second image, featuring the *needle* object amongst multiple others, is called the *haystack* image. The performance of the DCNN is evaluated on how well it extracts the *needle* object from the *haystack* image. The Intersection over Union (IoU) error, measuring to which percentage the desired segmentation and the ground truth differ, is commonly used to evaluate segmentation quality, and is used to train the network. Additionally, the Euclidean distance between predicted object pickup coordinates and the ground truth is measured during training in order to later derive the probability of a robotic pickup operation succeeding [30].

Figure 2 visualizes the nomenclature used on an abstracted segmentation network. Both *needle* and *haystack* images are presented to the network at the same training iteration. Due to the Siamese nature [15] of the network used, the feature extraction filters for both images are shared. Deconvolutional filters restore the original dimensions of the image, and highlight the extracted object. The network output is then compared to the ground truth segmentation, and the error is backpropagated. For the next iteration, a new *needle*/*haystack* combination is selected from the training set. The following section describes the network layout in detail.

3.1 Network Design

Our one-shot segmentation network operates fully convolutional, without the requirement of fully connected layers or any non-differentiable operations. The DCNN layout is extending [34] by adding deconvolutional layers as proposed in [19]. Figure 3 presents the layout of the DCNN. Convolutional layers successively break down the input image from 256×192 pixels and 4 channels to a $1 \times 1 \times 4{,}544$ feature vector representation. The convolutional layers are arranged in eight same-size stacks, after which a 2×2 max pooling operation is performed. All filters feature 3×3 kernels until a one dimensional feature vector is achieved, after which the eighth stack uses 1×1 kernels to introduce additional nonlinearity. All activation functions are leaky ReLUs [22], with $\alpha = 0.02$. Dropout layers with a 15% dropout rate are employed after the third to eighth stack of convolutional layers. Depending on whether the *needle* or *haystack* image is presented, the grey shaded *selector* layers are either ignored, or multiplied onto the outcome of every filter of the layer before.

To perform one-shot object segmentation, we use a Siamese architecture of the feature extraction part of the network, meaning that during one training iteration, two images will be presented to the convolutional layers. However, we do not train a difference metric; instead, we train the network to enable or disable convolutional filters that produce the extraction of the desired object only. The following paragraph provides a step by step walkthrough of a training iteration.

First, the *needle* image is presented to the feature extraction part of the network, i.e., the convolutional layers. A 4,544 dimensional output vector is returned, providing an additional 4,544 filter weights w_i for extraction. Next, the *haystack* image is presented to the network. This time, every one of the 4,544 convolutional feature extraction filters is weighted by the weight w_i extracted from the *needle* image. This effectively allows the network to block the output of any filters that detect objects other than the one looked for. As the convolutional filter parameters are shared, the amount of variables to be learned is still manageable for graphics cards commonly used in deep learning. The dimensions of the *haystack* image are then reconstructed using deconvolutional layers, and a belief map of equal size is generated, highlighting the areas for which the network extracted similar features to the *needle* image. Figure 4 shows a simplified version of the approach. The belief map is then compared to the desired segmentation, and an error value is returned. The IoU measured between generated belief map and ground truth forms the main loss to be minimized. Other error metrics are extracted as well, as detailed in the following section.

3.2 Error Measures

In order to aid the network to produce a highly abstract representation and to speed up convergence [3], multiple error measures are combined. This transforms the segmentation task into a multi-task

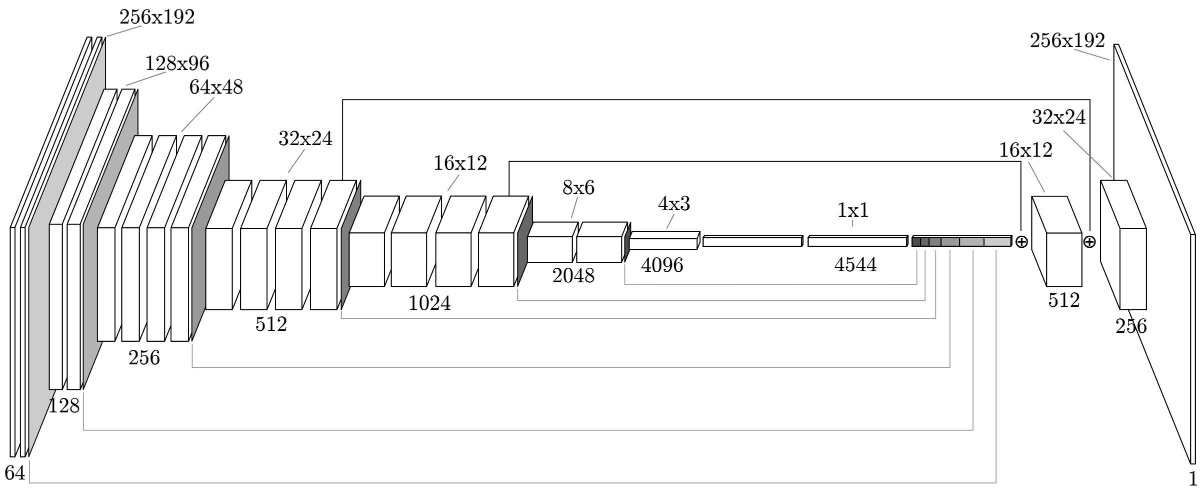

Figure 3: The DCNN used for one-shot object segmentation. Above the network, pixel dimensions are stated; below the network, the number of filters can be found for every stack of convolutional filters. After every such stack, but before the ensuing pooling operation, a *selection* layer is introduced (shaded in grey). Selection layers are inactive when feeding the *needle* image into the network. From the output of the last convolutional 1x1 layer, *selection* layer weights are derived and multiplied onto the result of every filter when the *haystack* image is presented. This can effectively disable all filters potentially extracting unwanted objects, and only keep filters that respond to the *needle* object active.

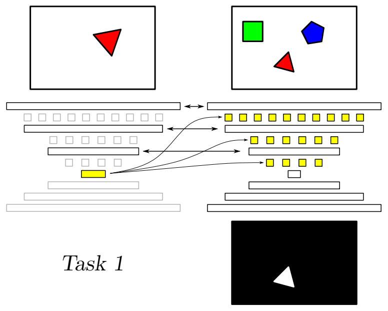

Figure 4: Our approach feeds the *needle* image through the convolutional stage first, achieving a 4,544 dimensional weight vector. This weight vector is used to weight each filter output during the convolutional deconstruction of the *haystack* image. Both networks displayed share their convolutional parameters. Reconstruction of the *needle* object on the *haystack* image denotes the main task, *Task 1*, of the network to be learned.

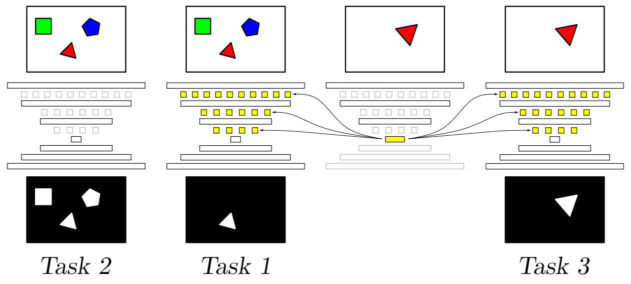

Figure 5: Multiple error measures can be derived from a single *needle/haystack* image combination. Learning multiple tasks from multiple error measures can provide more information to the optimizer every iteration. First, the constrained network should detect both instances of the *needle* object with the same weights. Second, the unconstrained network should detect every object, effectively learning a background model to be removed. The third task forces the network to extract features from the desired region only, i.e., from the *needle* object.

problem, which have been shown to be able to generate higher levels of abstraction than single-task learning. Figure 5 illustrates how three different IoU metrics are obtained and subsequently minimized during each training step. The first independent task for the network to learn is to extract the *needle* object from the *haystack* image, as described in Section 3.1 above. The second task is trained at the same time, being that the network should extract all the objects of the *haystack* image if no weights are provided, i.e., all filters are in effect. This combination aids the network in learning narrow-band filters, that only extract the required object type and ignore the background fully. Finally, the third task to be learned is making the network re-detect the *needle* object with activated weights. This task forces the network to learn filters at least wide enough to extract both *needle* objects in both images, regardless of lighting or scale. Training all three tasks, i.e., combining the narrowing and widening effects, the network can learn filters that are only sensitive to a certain range of object variability, ignoring objects that are outside the learned range in feature space representation.

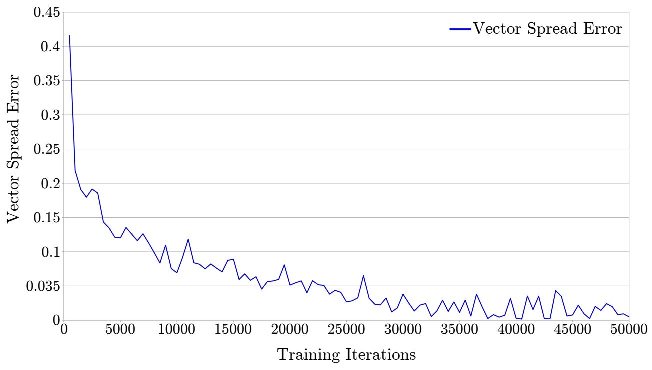

Figure 6: The decreasing vector spread error during training.

3.3 Vector Spread Loss

In order to further aid the network learning a mapping of input images into a spanning feature space, we add an additional fourth task, minimizing a *vector spread* loss term. The aim of the vector spread error metric is to minimize intra-class distances in feature space, while maximizing inter-class distances, thus spreading feature vectors of different classes. Inspired by the angular error used by [36], we model the vector spread error using a cosine distance, as shown in Equation 1, between object representation vectors in feature space. For every mini-batch of size b, the vector spread error is calculated for all the combinations of feature vectors gathered from the last convolutional layer. Equation 2 details the calculation of the vector spread error metric. The distance of every feature vector combination of X_i and Y_j is desired to be 1 if the classes of X_i and Y_j differ, or 0 if X_i and Y_j belong to the same class.

$$CosineDistance(X, Y) = \frac{cos^{-1}(\frac{X \cdot Y}{\|X\|\|Y\|})}{\pi} \tag{1}$$

$$\mathcal{L}_{VectorSpread} = \sum_{i=0}^{b}\sum_{j=0}^{b}\frac{D_{ij}}{b \times b}\text{, where}$$
$$D_{ij} = \begin{cases} CosineDistance(X_i, Y_j) & \text{if } Class(X_i) = Class(Y_j), \\ 1 - CosineDistance(X_i, Y_j) & \text{otherwise.} \end{cases} \tag{2}$$

The differences between calculated and desired values are summed up and normalized by dividing by the number of elements, $b \times b$, forming the vector spread loss term. This metric favors a distinct set of filters being activated for every class encountered by forcing their vector representations to be orthogonal in feature space. As filters shared for multiple object types will extract both objects at all times, learning separate filters for different object types is an advantage. Additionally, it forces the network to collapse feature space representations of objects of the same class. Figure 6 presents an example of the vector spread metric converging to a value near zero while training, indicating that the dimensionality of the feature space was high enough to fit all the classes collision free.

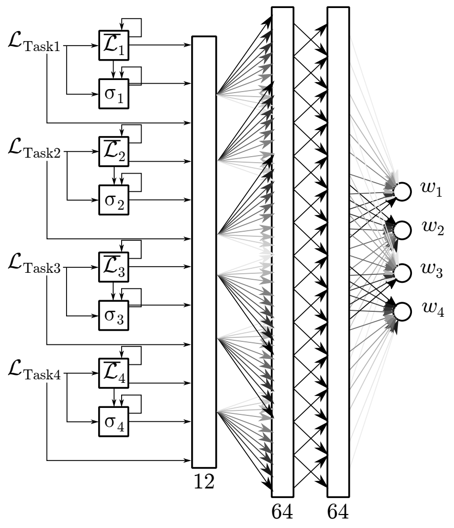
$\mathcal{L}_{\text{Task1}}$
$\mathcal{L}_{\text{Task2}}$
$\mathcal{L}_{\text{Task3}}$
$\mathcal{L}_{\text{Task4}}$
$\overline{\mathcal{L}}_1$
σ_1
$\overline{\mathcal{L}}_2$
σ_2
$\overline{\mathcal{L}}_3$
σ_3
$\overline{\mathcal{L}}_4$
σ_4
12
64
64
w_1
w_2
w_3
w_4

Figure 7: The layout of the auxiliary network to produce weights for every loss function used. The loss values for the four different tasks are averaged over time, and their variances are measured. The current loss values, average values and their variances are fed into the network, after which two fully connected layers produce 4 output weights.

3.4 Weighting Loss Terms

Inspired by and extending the work of [30], we implemented an auxiliary neural network to produce an optimal weighting between the different loss terms. A network taking the current losses, their exponential moving averages and variances as inputs and featuring two fully connected layers of 64 nodes each produced quickest convergence based on limited trials. Figure 7 shows the layout of the auxiliary network. All activation functions are leaky ReLUs, and the same optimizer settings are used as for the main DCNN. Training the auxiliary network is done as detailed in [30].

4 EXPERIMENTS

We train a deep convolutional neural network on the new, publicly available RoboCup@Work dataset. We split up the dataset into 11 object classes to be used for training, and 2 object classes to be held out during training. The network is trained on all the 78 combinations of different classes to be held out. For every combination, we train the network for 50,000 iterations, as initial tests have shown the network converging well within this span. The evaluation set contains *needle* images featuring exclusively held out classes, and *haystack* images featuring at least both held out object classes, and up to 10 known objects of known classes as distractors.

He initialization, as proposed in [9], is used to initialize all weights of the neural network. Due to memory constraints of the GPUs used, no larger mini-batch size than 6 could be employed. The established Adam optimizer [14] is used to minimize the loss term, with a learning rate of 1×10^{-5}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1 \times 10^{-8}$. Every 500 iterations during training, we evaluate the

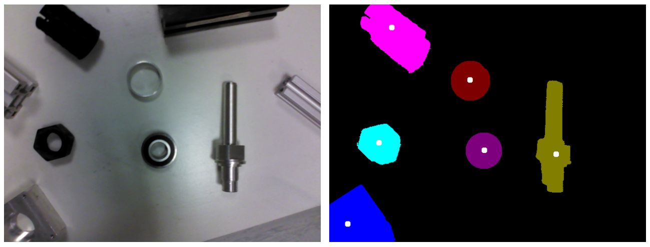

Figure 8: Left: An example image from the RoboCup@Work dataset. Right: Automatically acquired segmentation and manually labeled pickup points. Note that the automatic segmentation frequently includes shadows with the object.

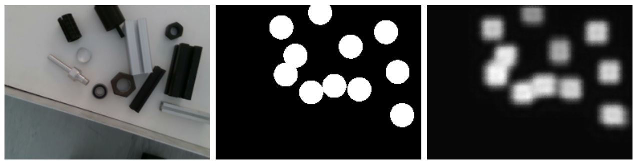

Figure 9: Left: An example *haystack* image from the evaluation set. Center: The ground truth, approximated segmentation for learning Task 2, i.e., all filters are active. Right: The segmentation output of the network.

network's capability to extract unknown classes on the entire evaluation set. This allows us to produce charts detailing the learning progress. In order to evaluate the performance gains in abstractive power and convergence speed, we repeat the experiments with the vector spread error disabled, and on a singe IoU error for the first identified task only. Results of these tests can be found in Section 5.

4.1 Datasets

While a multitude of object segmentation datasets exists [6, 12], almost none include industrial objects, an exception being T-LESS [10]. T-LESS includes 38,000 training and 10,000 evaluation RGBD images of 30 industrial object types. However, due to the nature of the dataset, all objects are texture-less and of the same color.

The RoboCup@Work dataset [30] currently contains 36,000 RGBD images of industrial objects, of which 15,800 are manually labeled with a center of gravity, or *pickup point*. The objects featured are the 13 object classes used in the annual RoboCup@Work challenge [16]. As object detection and picking performance on this dataset can be tested with our robotic setup, most of our results are derived from this dataset. Figure 8 shows an example of this dataset and segmentation derived from previously world-cup winning RoboCup@Work software [2]. As the automatic segmentation fails when objects are placed closer than approximately 3 cm apart, we obtained an alternative, approximate segmentation by extracting a circular region around the pickup point instead. This way, every object is represented by the same amount of pixels, preventing the tendency to select larger objects frequently happening when training on IoU [32]. Figure 9 shows an example of the approximated segmentation labels used, and the output of the trained network.

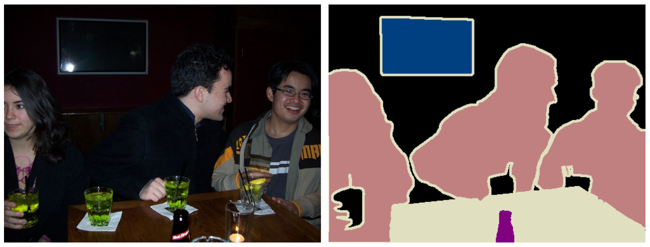

Figure 10: Left: An example image taken from the Pascal VOC dataset featuring three instances of the person object, one screen, and one bottle. Right: The corresponding ground-truth segmentation.

At the time of writing, the most widely used image segmentation dataset is the Pascal VOC dataset [5], featuring over 2,900 RGB images semantically segmented into 20 object classes. The object classes range from outdoor vehicles, over animals, to objects commonly found indoors. Depth information is not available, and the intra-class variation is high, as is to be expected for real world everyday objects. This makes the Pascal VOC dataset very challenging. Figure 10 shows an example image featuring multiple classes and its corresponding ground-truth segmentation. Using the Pascal VOC dataset as an independent benchmark for one-shot segmentation poses yet another challenge. To ensure the network learns a true one-shot object segmentation rather than a zero-shot foreground detection based on saliency [11], the *haystack image* has to include multiple objects, so that the network can be penalized for extracting both. The Pascal VOC dataset only contains 1,050 possible *haystack* images, with the majority of those images containing just two object classes. While the same is true for the RoboCup@Work dataset, not only does it contain more (1,513) possible *haystack* images, but the distribution of object classes is much more favorable, as shown in Figure 11. The more different objects are present in a *haystack*-image, the more information is available per iteration for the network to learn. Additionally, the detection task becomes harder, as it is more likely to mislabel an object. *Haystack* images containing multiple objects can be used in conjunction with every class they feature, further increasing the number of possible *needle/haystack* combinations to train.

5 RESULTS

We obtain classification data by analyzing if combining the prediction with the desired segmentation mask produces a higher IoU measure than combining it with any other object segmentation mask. Training the network on all 78 possible combinations of included and held out objects in the RoboCup@Work dataset resulted in an averaged 69% classification rate of previously unseen data. Investigating the confusion matrix, shown in Figure 13 (left), quickly identifies recalling the *Distance Tube* object as the weakest contributor. The low recall of 18.5% on this object shows the main limitation of this and any likewise operating one-shot object detectors, as the object inhabits a point outside the learned feature space, spanned by training on the remaining objects. This is due to a unique feature of the *Distance Tube*, being near invisible on the depth channel, as shown by Figure 12. This object being held

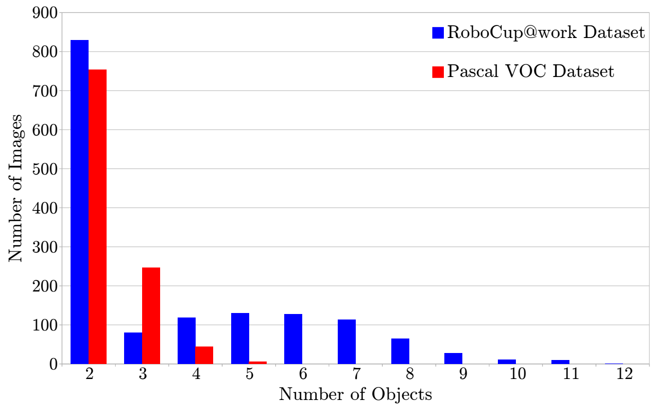

Figure 11: A histogram of object counts in the RoboCup@Work (blue) and Pascal VOC (red) datasets, with the X axis enumerating the number of occurring objects, and the Y axis denoting the number of images featuring this number of objects. Single object images excluded for clarity.

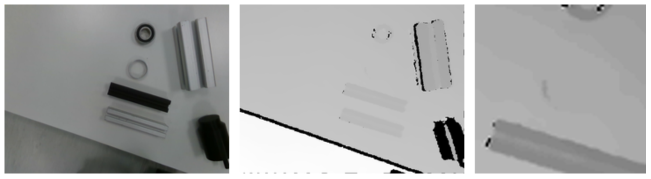

Figure 12: The *Distance Tube* object (small grey aluminium ring) next to other objects in visible light (left) and as occurring on the depth channel (center). The right image shows a contrast enhanced magnification of the small portion of the distance tube that is visible on the depth channel.

out, the network overtrains on the depth channel otherwise being a good indicator for object presence, and resorts to marking another object in the *haystack* image. Removing the *Distance Tube* object from the dataset raises the accuracy to 81.15%.

Disabling the depth channel forces the network to take only RGB information into account, and raises the recall of the *Distance Tube* to 48.9%. This configuration leads to an overall accuracy of 73.1%, and an average IoU of 0.45 after convergence. Using the pickup error metric presented in [30], we estimate the probability for successfully picking an identified object. The one-shot pickup rate achieved is 87.53%. Figure 14 shows the learning progress of the trained classifiers by plotting the accuracy and IoU metrics obtained on the held out validation set every 500 steps during training. Figure 13 (right) shows the according confusion matrix, and Table 1 lists the precision and recall measures per object type.

We repeated the tests with disabled Task 4, and with disabled Tasks 1-3. The resulting IoU and accuracy curves are shown in Figure 15. Training on all 4 Tasks converges faster and to a lower error than training on fewer. Table 2 provides the convergence points and average accuracy after convergence. We observe that adding the *vector spread* metric does not influence the reached

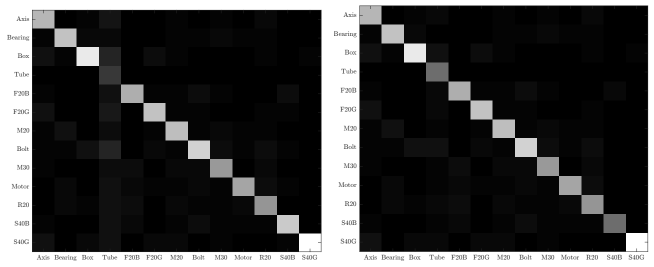

Figure 13: Left: The confusion matrix of correctly one-shot detecting the *needle* object in the *haystack* image when training on all RGBD channels. The *Distance Tube* object shows a very low recall of 18.5%. As it is the only object not showing up in the depth channel, training the network without it results in overfitting on depth features. Right: The same confusion matrix, when trained on RGB information only.

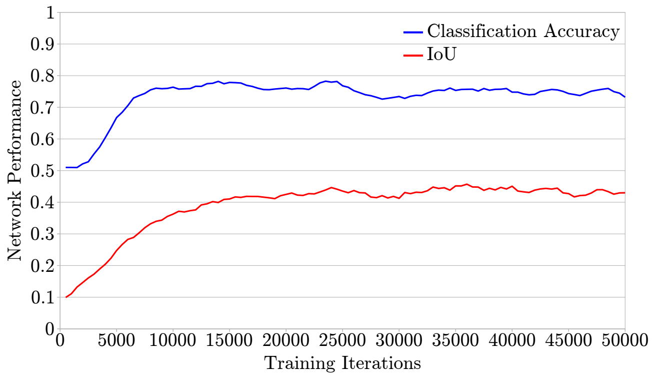

Figure 14: Classification accuracy (blue) and IoU (red) obtained on untrained classes averaged over the 78 training sessions and plotted for every 500 training iterations. Training and validation data contained RGB images only.

Table 1: Precision and Recalls trained on RGB

Object class	Precision	Recall
Axis	0.78	0.65
Bearing	0.78	0.75
Bearing Box	0.74	0.80
Distance Tube	0.92	0.49
F20_20_B	0.73	0.75
F20_20_G	0.75	0.78
M20	0.73	0.70
M20_100	0.68	0.75
M30	0.70	0.68
Motor	0.69	0.79
R20	0.66	0.65
S40_40_B	0.61	0.78
S40_40_G	0.78	0.91

accuracy in a significant manner, but, compared to training only on Tasks 1-3, reduces the iterations to convergence by 22.22%.

Table 2: Multi-task learning effect

Tasks Trained	Iterations to converge	Avg. Accuracy
Only IoU	10,500	72.40%
Tasks 1-3	9,000	75.31%
Tasks 1-4	7,000	75.47%

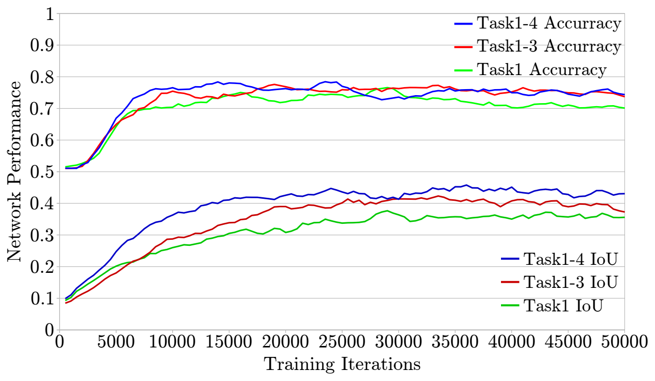

Figure 15: Accuracy and IoU on the evaluation set during training for three training configurations. Blue: All 4 Tasks identified are trained. Red: Tasks 1-3 were trained. Green: Only IoU was trained. The top three charts indicate accuracies, the lower charts IoU.

To evaluate the performance of the network when re-trained with all the available information, we repeated the network training with no classes withheld and instead splitting the dataset into a training and evaluation set at the ratio of 4 to 1. Figure 16 shows the rapid learning of the network, achieving peak performance of 94% correct classification after as few as 4,000 iterations. The pickup rate for correctly identified objects is estimated to be 97.77%. This clearly shows the advantage of re-training over one-shot segmentation, and further confirms the necessity to be able to re-train as quickly as possible.

6 CONCLUSION

In this paper, we proposed a novel framework for one-shot object segmentation using fully convolutional Siamese Deep Neural Networks. We demonstrated its effectiveness in segmenting new objects in industrial settings, and its rapid re-training capabilities, to replace one-shot segmentation with more accurate fully trained segmentation as quickly as possible. We have shown that our approach can successfully extrapolate how to segment unknown object types based on training and evaluating it on different object classes. We have also shown the limitations of this approach, being that it can overfit on specific details prevalent in the training set. However, this limitation can easily be overcome in an industrial setting, where a true spanning set of object classes can be combined to form a training set, allowing the network to robustly detect and segment previously unknown objects based on a single example image.

By applying multi-task learning techniques, our network can converge after as few as 4,000 training iterations. In an industrial

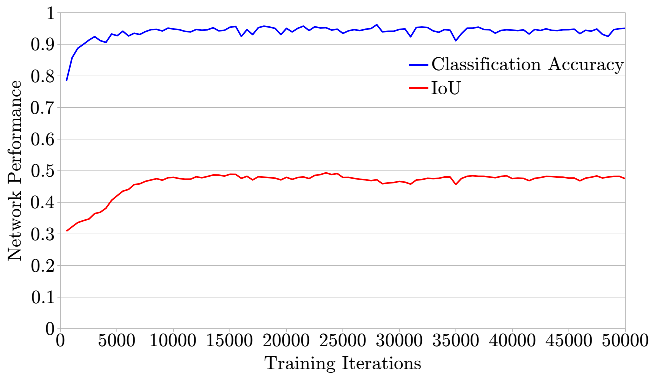

Figure 16: Classification accuracy (blue) and IoU (red) obtained on known classes, plotted for every 500 training iterations. Training and validation data contained RGB images only.

setting, our approach can be used as a one-shot segmentation network to pick up objects while re-training is in effect. After the very short training time of 57 minutes, a 94% accurate classifier can pick up identified objects with an estimated rate of 97.77%. Before re-training is completed, the network can robustly operate in a less accurate, one-shot segmentation manner, fully alleviating any training time bottleneck. We therefore conclude that robots in the *Factory of the Future* can greatly benefit from our presented one-shot object segmentation approach, allowing them to autonomously produce, locate, and pick novel objects without human supervision.

Our future work will extend the one-shot segmentation to a few-shot segmentation approach, in which multiple shots of the same prototype object are provided. Other research [32] has shown that object identification can greatly benefit from this transition. Further future work includes extending the RoboCup@Work dataset with arbitrary backgrounds and decoy objects, and providing full segmentation data. As this will provide an even better *Factory of the Future* approximation, our approach will be re-evaluated on the extended dataset and comparable datasets such as T-LESS [10]. The findings in the paper will also be applied in pick-and-place tasks on real robot platforms, possibly with interactive perception [1] and other sensing inputs (e.g., tactile sensing [21]).

7 ACKNOWLEDGEMENT

We are very grateful to James Butterworth and Jacopo Castellini for helpful comments and discussions. This work was supported by the EPSRC project “Robotics and Artificial Intelligence for Nuclear (RAIN)" (EP/R026084/1).

REFERENCES

[1] Jeannette Bohg, Karol Hausman, Bharath Sankaran, Oliver Brock, Danica Kragic, Stefan Schaal, and Gaurav S Sukhatme. 2017. Interactive perception: Leveraging action in perception and perception in action. *IEEE Transactions on Robotics* 33, 6 (2017), 1273–1291.
[2] Bastian Broecker, Daniel Claes, Joscha Fossel, and Karl Tuyls. 2014. Winning the RoboCup@Work 2014 Competition: The smARTLab Approach. In *RoboCup 2014: Robot World Cup XVIII.* Springer, 142–154.
[3] Rich Caruana. 1998. Multitask Learning. In *Learning to Learn.* Springer, 95–133.

[4] Ondřej Chum, Michal Perd'och, and Jiří Matas. 2009. Geometric min-Hashing: Finding a (Thick) Needle in a Haystack. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 17–24.

[5] Mark Everingham, Luc Van Gool, Christopher K.I. Williams, John Winn, and Andrew Zisserman. 2010. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision* 88, 2 (2010), 303–338.

[6] Alberto Garcia-Garcia, Sergio Orts, Sergiu Oprea, Victor Villena-Martinez, and José García Rodríguez. 2017. A Review on Deep Learning Techniques Applied to Semantic Segmentation. *CoRR* abs/1704.06857 (2017).

[7] Yanming Guo, Yu Liu, Ard Oerlemans, Songyang Lao, Song Wu, and Michael S. Lew. 2016. Deep Learning for Visual Understanding: A Review. *Neurocomputing* 187 (2016), 27–48.

[8] M. Harant, M. Millard, M. Sreenivasa, N. Šarabon, and K. Mombaur. 2018. Comparison of Objective Functions for the Design of an Active Spinal Exoskeleton using Motion Capture Data and Optimal Control. *Submitted to RA-L only Intelligent Human-Robot Interaction for Rehabilitation and Physical Assistance for the IEEE Robotics and Automation Letters (RA-L)* (2018).

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In *Proceedings of the IEEE International Conference on Computer Vision.* 1026–1034.

[10] Tomáš Hodaň, Pavel Haluza, Štěpán Obdržálek, Jiří Matas, Manolis Lourakis, and Xenophon Zabulis. 2017. T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects. *IEEE Winter Conference on Applications of Computer Vision (WACV)* (2017).

[11] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. 2015. Online Tracking by Learning Discriminative Saliency Map with Convolutional Neural Network. In *International Conference on Machine Learning.* 597–606.

[12] Joel Janai, Fatma Güney, Aseem Behl, and Andreas Geiger. 2017. Computer Vision for Autonomous Vehicles: Problems, Datasets and State-of-the-Art. *arXiv preprint arXiv:1704.05519* (2017).

[13] Gil Keren, Maximilian Schmitt, Thomas Kehrenberg, and Björn Schuller. 2018. Weakly Supervised One-Shot Detection with Attention Siamese Networks. *arXiv preprint arXiv:1801.03329* (2018).

[14] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the International Conference on Learning Representations (ICLR), San Diego, 2015.*

[15] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. 2015. Siamese Neural Networks for One-Shot Image Recognition. In *ICML Deep Learning Workshop*, Vol. 2.

[16] Gerhard K. Kraetzschmar, Nico Hochgeschwender, Walter Nowak, Frederik Hegger, Sven Schneider, Rhama Dwiputra, Jakob Berghofer, and Rainer Bischoff. 2015. RoboCup@Work: Competing for the Factory of the Future. In *RoboCup 2014: Robot World Cup XVIII*, Reinaldo A. C. Bianchi, H. Levent Akin, Subramanian Ramamoorthy, and Komei Sugiura (Eds.). Lecture Notes in Computer Science, Vol. 8992. Springer International Publishing, 171–182.

[17] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. 2009. Learning To Detect Unseen Object Classes by Between-Class Attribute Transfer. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009.* IEEE, 951–958.

[18] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. 2014. Industry 4.0. *Business & Information Systems Engineering* 6, 4 (2014), 239.

[19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully Convolutional Networks for Semantic Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* 3431–3440.

[20] Dominik Lucke, Carmen Constantinescu, and Engelbert Westkämper. 2008. Smart Factory - A Step towards the Next Generation of Manufacturing . In *Manufacturing systems and technologies for the new frontier.* Springer, 115–118.

[21] Shan Luo, Joao Bimbo, Ravinder Dahiya, and Hongbin Liu. 2017. Robotic tactile perception of object properties: A review. *Mechatronics* 48 (2017), 54–67.

[22] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. 2013. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, Vol. 30. 3–8.

[23] Claudio Michaelis, Matthias Bethge, and Alexander S Ecker. 2018. One-Shot Segmentation in Clutter. *arXiv preprint arXiv:1803.09597* (2018).

[24] George Papandreou, Liang-Chieh Chen, Kevin P. Murphy, and Alan L. Yuille. 2015. Weakly- and Semi-Supervised Learning of a Deep Convolutional Network for Semantic Image Segmentation. *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), 1742–1750.

[25] Pedro O. Pinheiro and Ronan Collobert. 2015. Weakly Supervised Semantic Segmentation with Convolutional Networks. In *CVPR*, Vol. 2. 6.

[26] Joseph Redmon and Ali Farhadi. 2017. YOLO9000: Better, Faster, Stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* IEEE, 6517–6525.

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 6 (2017), 1137–1149.

[28] Ruslan Salakhutdinov, Joshua Tenenbaum, and Antonio Torralba. 2012. One-Shot Learning with a Hierarchical Nonparametric Bayesian Model. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning.* 195–206.

[29] Jürgen Schmidhuber. 2015. Deep Learning in Neural Networks: An Overview. *Neural Networks* 61 (2015), 85–117.

[30] Benjamin Schnieders and Karl Tuvls. 2018. Fast Convergence for Object Detection by Learning how to Combine Error Functions. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE, 7329–7335.

[31] Eli Schwartz, Leonid Karlinsky, Joseph Shtok, Sivan Harary, Mattias Marder, Sharathchandra Pankanti, Rogerio Feris, Abhishek Kumar, Raja Giries, and Alex M. Bronstein. 2018. RepMet: Representative-based metric learning for classification and one-shot object detection. *arXiv preprint arXiv:1806.04728* (2018).

[32] Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots. 2017. One-Shot Learning for Semantic Segmentation. In *Proceedings of the British Machine Vision Conference (BMVC) 2017.*

[33] Tong Shen, Guosheng Lin, Lingqiao Liu, Chunhua Shen, and Ian Reid. 2017. Weakly Supervised Semantic Segmentation Based on Web Image Cosegmentation. *arXiv preprint arXiv:1705.09052* (2017).

[34] Karen Simonyan and Andrew Zisserman. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556* (2014).

[35] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. 2016. Matching Networks for One Shot Learning. In *Advances in Neural Information Processing Systems.* 3630–3638.

[36] Jian Wang, Feng Zhou, Shilei Wen, Xiao Liu, and Yuanqing Lin. 2017. Deep Metric Learning with Angular Loss. In *2017 IEEE International Conference on Computer Vision (ICCV).* IEEE, 2612–2620.