Technische Universität München
Institute for Cognitive Systems (ICS)
Prof. Gordon Cheng

Master Thesis

Learning to recognize new objects using deep learning and contextual information

Master Student:	Niklas Barkmeyer
Matrikelnummer:	03617655
Anschrift:	Klugstr. 91
	80637 München
E-Mail:	niklas.barkmeyer@tum.de
Betreuer:	Karinne Ramirez
Beginn:	30.03.16
Abgabe:	05.09.16

Abstract

Enabling robots to recognize objects in the real world is a very important and challenging problem, due to variability in lighting conditions, visibility constraints, and the availability of large amounts of labeled data. This problem has currently been addressed using hierarchical representations such as convolutional neural networks (CNN), which are multi-stage architectures inspired by how the human brain processes visual information. A subgroup of deep learning, these algorithms can be trained end-to-end and outperform other methods on many benchmarks. However, CNNs are not widely deployed in robotics yet. Additionally, existing robotic systems do not leverage contextual information to enhance their machine learning algorithms with knowledge and reasoning systems to understand and interact with known and unknown objects in their environment.

In this thesis, we present and analyze a method that combines deep learning algorithms with semantic reasoning techniques. This system enriches the visual recognition capabilities of a robot by including contextual information of an object such as material, shape, color, and affordance during a human-supervised learning process. We use deep convolutional networks and classifiers trained on extracted high-dimensional features to predict these attributes. Our experiments are evaluated on the iCub humanoid robot and tested on two datasets, namely the iCubWorld28 and the new TUM-ICS dataset.

The results show that our proposed system improves the overall recognition accuracy and learning capabilities of our iCub robot, both for known and unknown objects. Compared to a stand-alone deep learning network, the recognition performance increases from 85% to 98% for known objects and from 0% to 65% for unknown objects.

Contents

Chapter 1

Introduction

Artificial intelligence (AI) is the general science and engineering principle which deals with how to create intelligent machines. AI recently gained a lot of public attention, especially after Google DeepMind's AlphaGo defeated the South Korean master in the board game Go in early 2016. Research in AI started in the 1950's, where one of the biggest milestones was IBM's supercomputer Deep Blue, which beat Garry Kasparov in chess in 1997. A subgroup within AI, machine learning algorithms learn from large quantities of statistical data and make predictions on unknown data. Since the 1980s, algorithmic approaches in decision tree learning, inductive logic programming, clustering, reinforcement learning, and Bayesian networks resulted in many new technical advances and solutions for real-world problems, such as spam filtering, optimal control, image and speech processing, and many others. Within machine learning, deep learning is a new branch of algorithm which consists of multiple non-linear layers.

In recent years, especially deep learning algorithms have contributed to recent breakthroughs which have caused a hype within the AI and machine learning community [1]. However, the principles of deep learning, especially convolutional neural networks (CNNs), is not a new concept. The general ideas have existed since the 1990s [2, 3, 4, 5]. Since then, CNNs were very popular, but their importance decreased with the introduction of support vector machines (SVMs). Due to more powerful graphical processing units (GPUs), larger quantities of data, and better algorithms, Krizhevsky [6] achieved by far the best results reported in the image classification challenge ILSVRC in 2012. Since then, deep learning has become very popular again and the successes of deep learning have accelerated since then.

Convolutional neural networks are multistage architectures of non-linear layers that can be trained on large datasets. They can be used for many object recognition [6, 7] and computer vision tasks [8, 9]. Initially, a lot of research work focused on creating new architectures to beat other state-of-the-art networks in highly-competitive benchmarks evaluated on the ImageNet dataset [6, 7]. Other research studies concentrated on analyzing the underlying architectures of convolutional networks in general and evaluated why the networks work so

well [10, 11, 12, 13]. In current research, deep learning models are often linked and applied with other methods, such as natural language processing (NLP) [14, 15] or reinforcement learning (RL) [16, 17, 18, 19] algorithms, to provide solutions for new problem domains.
The ability to comprehend visual context is a critical and arguably the most complicated cognitive capability for understanding the real world. According to Li Fei Fei [20], computer vision is the key enabling technology to build future intelligent machines that can see and think like humans. When humans solve a given task, they often rely on their intuition and exploit available contextual information about the problem [21]. Humans use their knowledge to simplify the problem and solve the task by leveraging their experience about related concepts. In contrast, technical systems lack this ability to infer knowledge in multiple representations by nature. Concerning visual processing systems, the capabilities of classical object recognition methods are mostly limited to categorize known objects. In recent research, new ideas have emerged how additional information can be extracted from images to improve the visual recognition performance [22, 23, 24, 25]. Computer vision algorithms that describe objects with attributes and infer object properties based on contextual information provide new possibilities to learn and recognize new objects [26]. This attribute-centric approach has the advantage that new objects can be detected by high-level descriptions of the objects instead of training models on these images [24].
Our motivation in this thesis is to combine the superior performance of deep convolutional neural networks with logical reasoning components into one system. This system is used to teach our humanoid robot iCub new objects based on contextual information. According to the literature [23, 26, 27], attributes such as material, shape, color, and affordance are examined as the relevant context of a detected object. These attributes are predicted for each object by our deep networks or machine learning classifiers.
In summary, the main goals of this thesis are to:

- Compare and analyze the performance of deep learning algorithms and linear classifiers on objects, attributes and affordance.
- Improve the visual recognition capabilities of the humanoid robot iCub on known objects with knowledge and reasoning methods.
- Enable iCub to learn new or unknown objects based on deep learning algorithms and contextual information.

The stated goals were addressed and reached in several steps: first, the relationship between the analyzed deep learning algorithms and linear classifiers are evaluated in consideration of the chosen attributes and affordances. In this step, the performances are benchmarked on two datasets. Second, an enhanced deep network is proposed which combines the outputs of the machine learning algorithms with contextual information. The third point is the main research goal of this thesis: the created enhanced deep network enables the robot to learn new objects. This system is designed in consideration of both evaluated machine learning algorithms and all attributes.
The four main contributions in this work are the following:

- The TUM-ICS dataset, which contains 16,500 images that are divided into 11 object categories and 23 object classes. The dataset has been acquired in three different settings on multiple acquisition days and times in changing lighting conditions. The dataset is fully labeled for object categories, the attributes material, shape, and color, and affordance.
- The definition and implementation of classes, object properties, and Prolog predicates, based on the hierarchical ontology representation of the TUM-ICS dataset.
- A detailed analysis about the benchmarked deep learning algorithms and linear classifiers trained on objects and contextual information is provided on two datasets. Further experiments describe why a new dataset was needed, and underlying concepts of these algorithms are explained.
- An enhanced deep network is proposed which combines the benefits of deep learning algorithms with semantic reasoning techniques. This system equips the robot with on-line object recognition and learning capabilities based on semantic understanding of the objects' contextual information.

The structure of this work is described in the following. Chapter 2 describes the background of this thesis with a focus on convolutional neural networks, dimensionality reduction techniques, attributes and affordances, and semantic reasoning. In the last section of this chapter, we describe a similar experiment which was performed on the iCub robot by researchers from IIT. In chapter 3, we explain the major technical components that are needed to conduct our experiments. These are the iCub robot, machine learning toolboxes and other software frameworks, datasets used for training and evaluating our system performance, and our OWL ontology. The ontology contains the a-priori knowledge and contextual information our robot has about the known objects. In chapter 4, we analyze and evaluate our experiments. At first, we compare the object and context-recognition performance of our deep neural networks with other classifiers trained on extracted features on two datasets. While analyzing the performance on the collected TUM-ICS dataset, we take a deep dive to investigate why the methods performed so well or missed our expectations. Second, we present the results of our on-line object learning experiment, in which we used the predicted labels from our deep learning algorithms as input to our knowledge reasoning system to learn new objects. After evaluating the results, we discuss our findings in chapter 5, where some limitiations of the proposed system are also addressed. Finally, chapter 6 summarizes and concludes this thesis.

Chapter 2

Background and Related Work

In the first section, we describe convolutional neural networks. Based on [28, 29], we provide an in-depth analysis of the underlying concepts and the architecture that we use in this work. Furthermore, we give an outlook on current research and new application areas for state-of-the-art deep learning techniques. In the second part, we briefly explain the motivation and algorithms of dimensionality reduction techniques. The third section introduces the concepts of attributes and affordances. Background on semantic reasoning is described in the fourth section. The last section is the most important for our research goal: we describe related work that focuses on teaching new objects to robots, and show how their work differs to our main contribution in this thesis.

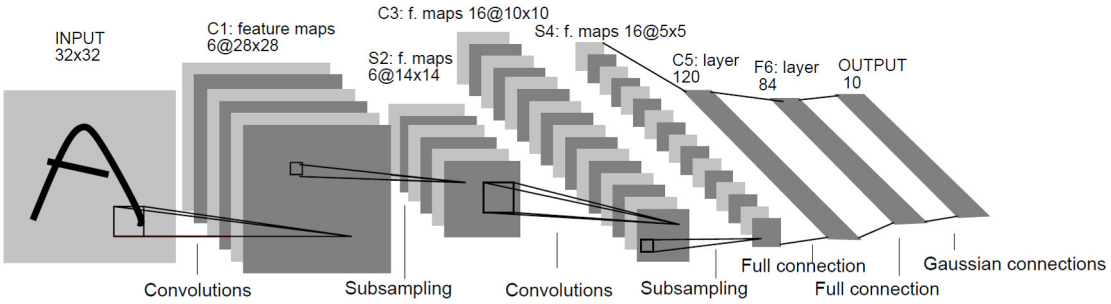


Figure 2.1: Yann LeCun is considered as the 'father' of convolutional neural networks. The image shows the famous LeNet-5 [30], which is applied to document recognition. A typical convolutional neural network consists of several non-linear layers. The input is an image and the output is a vector. The output vector contains the probability of the image belonging to each class. This network has four different types of layers which are named as: 'convolution', 'subsampling', 'full connection', and 'gaussian connection'. This network was introduced in 1998. Since the introduction of AlexNet [6] in 2012, networks are deeper and use dropout to reduce overfitting.

2.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a subgroup of deep learning algorithms, which focus on computer vision and image processing tasks. CNNs are loosely inspired by how the brain processes visual information. Hubel and Wiesel were the first who proposed deep models which are similar to a cat's visual system's structure [31]. These models identified simple cells with local receptive fields, similar to filters or kernels, and complex cells, which are similar to pooling layers. The first CNN was introduced in Fukushima's Neocognitron [32]. CNNs were later improved by Yann LeCun's LeNet-5 [30] (see Figure 2.1 for details) which applied gradient-based stochastic gradients to document recognition and was very successful for handwriting recognition tasks. A major drawback in the research and development of CNNs in the 1990s and 2000s has been the required computational power to apply them in large scale to high-resolution images. However, this has changed since the 2010s.

Three reasons exist why deep networks have become successful: 1) more computation power due to Moore's Law, especially found in today's GPUs, 2) more training data, and 3) novel and better algorithms. With the increase of computational power, researchers described new ways to train convolutional neural networks more efficiently that allowed deeper networks to be trained. Recently, the performance increased significantly for multiple image databases, approaching or even beating human performance, e.g. on digit-recognition (<0.25 percent) [33] and many other pattern-recognition tasks, most notably visual classification problems. In general, CNNs are very good at classifying objects such as particular dog or cat breeds based on fine-grained details, whereas humans have problems with this [6]. A drawback for deep learning is that recognizing objects in realistic settings requires very large datasets for training. Currently, the best CNNs struggle with objects that are small or thin or that have been distorted with digital filters. A recent study revealed the differences how deep neural networks and humans recognize objects. In this study, the images are encoded in a way which humans cannot recognize, but the deep neural networks detected the correct class of the object with almost 100% accuracy [11].

For training very complex convolutional neural networks on large datasets, the recent trend has been to increase the number of layers and layer size while using dropout [34] to address the problem of overfitting. Krizhevsky et. al and Szegedy et. al, the creators of the two most well known networks used in research, namely AlexNet [6] and GoogLeNet [7], stress how important it is to use a high dropout ratio during training. Another trend is to make the networks very deep, i.e. the networks have many layers. Zeiler et. al experimented in [10] with the depth of CNNs and concluded that the network's performance highly depends on the number of layers. The measured performance significantly decreases even if a single convolutional layer is removed. He states that the network's depth is more important than any other architectural component of the network. Consequently, the choice of the underlying architecture is the key to the network's performance. This can also be seen at GoogLeNet which was published in 2014 and has a depth of 22 layers,

not counting the pooling layers. In 2012, AlexNet has started the hype around progress in deep learning in particular with a network consisting of 8 layers. Although it achieved state-of-the-art results when it was published, many other networks have surpassed the performance since then. The limiting factor of a CNN's size is the amount of memory available on current Graphics Processing Units (GPUs) and the tolerated training time. Before GPUs were widely used for training deep networks, the total training time on CPUs was larger by a factor of 9. Nvidia's cuDNN library, which is optimized for fast image processing and efficient convolution operations in GPUs, further accelerates the training time by a factor of 1.9. Thus, state-of-the-art GPUs with the latest CUDA and cuDNN libraries installed increase the computation time by a factor of 17 compared to today's CPUs. Depending on the number of parameters of the network and the size of the dataset, training a convolutional neural network on a GPU with randomly initiated weights can take several days until a week. One of the fundamental differences between

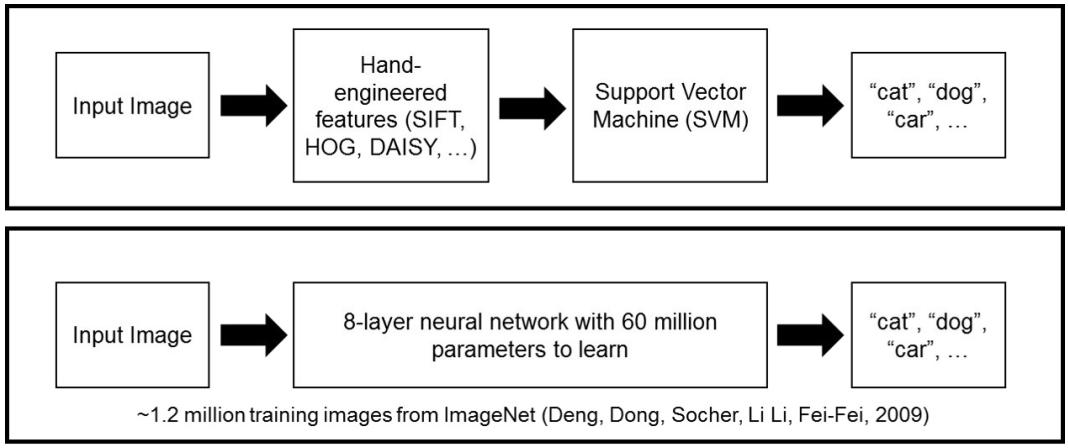


Figure 2.2: End-to-end learning with deep convolutional neural networks. Top: Traditional approach. Interest points are detected with e.g. SIFT and described in 128 dimensional descriptors. Classifiers such as SVM can be used to learn and predict from these features. Bottom: State-of-the-art approach since 2012. The object recognition pipeline is learned end-to-end with deep convolutional neural networks, which eliminates hand-engineering of features. The nets are trained with backpropagation. The figure is inspired by: [35]

how deep learning and traditional methods in image processing work is illustrated in the Figure 2.2. Until 2012, the state-of-the-art approach in computer vision was to use SIFT to extract hand-engineered features from images, which are encoded in 128 dimensional descriptors [36]. A classifier such as the popular support vector machines (SVM) [37] is trained on these descriptors for image classification. Deep convolutional neural networks combine both steps as the networks can be trained end-to-end from the images to the corresponding labels, as illustrated in Figure 2.3. The networks are trained via forward passes and backward-propagation, i.e. the computed gradients are added to the weighted neurons in the previous layer, which is further considered during training. The network is trained after the cost-function is minimized. The concept of end-to-end learning provides

the basis for many researchers in the artificial intelligence community who seek to apply this concept on new problem domains. Among them, Abbeel et. al apply end-to-end training to robotics to teach them how to learn and execute new tasks automatically with deep reinforcement learning [19]. Although end-to-end trained models have achieved impressive results, an alternative approach is to use pre-trained convolutional neural networks (which were trained on large datasets such as ImageNet) as feature extractors. In this case, the computed features are extracted from the second last layer before the softmax classifier computes the final probabilities. This highly-dimensional feature representation can be used for training machine learning algorithms from other toolboxes or libraries.

In this thesis, we compare both approaches: the end-to-end trained models and the

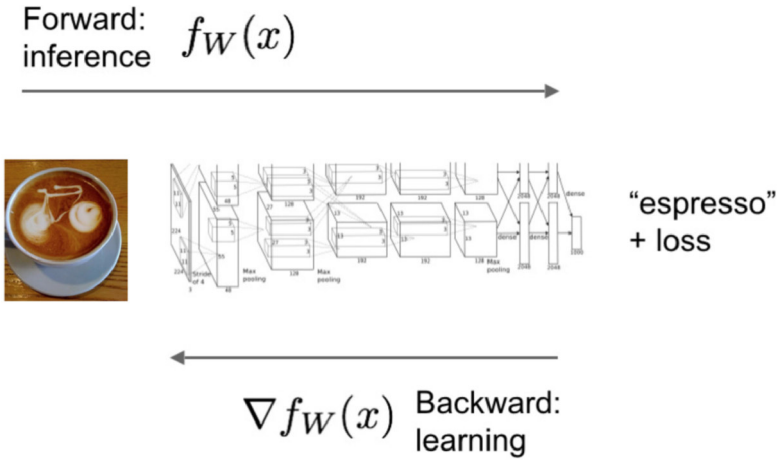


Figure 2.3: The principle of end-to-end learning, which is achieved by forward passes and backpropogation. Source: [38]

models from which we extracted high-dimensional feature vectors and used them to train classifiers.

2.1.1 Major Concepts

Based on [28, 29], this section gives an overview about the major concepts that are applied in convolutional neural networks. These models typically consist of several layers, and each layer is separated into multiple stages: convolutional layer, applying a rectified linear function, and max or average pooling. Figure 2.4 shows a convolutional layer with three stages. Each operation is explained in one of the following subsections.

2.1.1.1 Convolution

The name "convolutional neural network" indicates that the network employs convolution operations. Convolution is a linear operation that makes it possible to handle inputs of

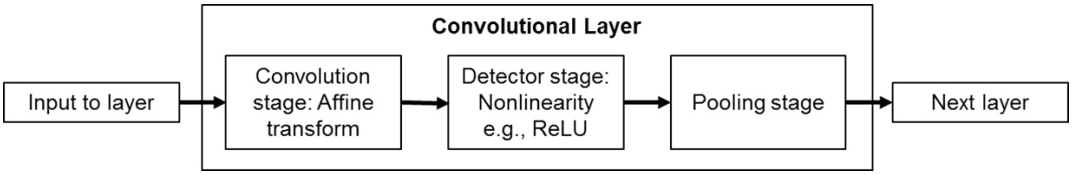


Figure 2.4: This visualization shows that a neural network layer consists of several stages: 1) Convolution, 2) ReLU, 3) Max Pooling. Normalization is not applied in this layer. This Figure is similarly shown in [29].

variable size. The input is usually a two-dimensional array of data and the kernel is usually a two-dimensional array of learnable parameters [29, p.199]. In CNNs, the two-dimensional inputs and kernels are images. The convolution operation is mathematically defined as

$$s[i,j] = (I \times K)[i,j] = \sum_m \sum_n I[m,n]K[i-m, j-n] \tag{2.1}$$

where I is the input image with dimensions m and n, K is the kernel, and s is the resulting output at location i and j.

Applying the convolution operation provides several benefits that are based on three important ideas: sparse interactions (or sparse weights), parameter sharing, and equivalent representations.

The term sparse interactions means that fewer connections between input and output values exist. Therefore, less parameters are needed which reduces the memory requirements and improves the statistical efficiency. This is achieved by applying smaller kernel sizes than the input image because fewer operations are required to produce the output. In contrast, traditional neural network layers, which do not employ convolutions, use a matrix multiplication to describe the relationships between each input value and each output value. This means that all outputs are connected to all inputs. Convolutions are very efficient operations to describe transformations of small receptive fields across input images.

Parameter sharing refers to using the same values for more than one function in a network. In traditional neural networks, each element of the weight matrix is used exactly once for computing the layer output. Each value of the kernel is used at every position of the input. The parameter sharing used by the convolution operation means that rather than learning a separate set of parameters for every location, only one set is learned. This does not affect the runtime but further reduces the storage requirements of the model to k parameters [29, p.202].

When convolution is applied on images, convolutions create 2-D feature maps that show where certain features appear in the input. Parameter sharing causes the layer to be equivariant to translation, i.e. if the input changes, the output changes in the same way. If the object in the input is moved, its representation will move the same amount in the

output.

2.1.1.2 Pooling

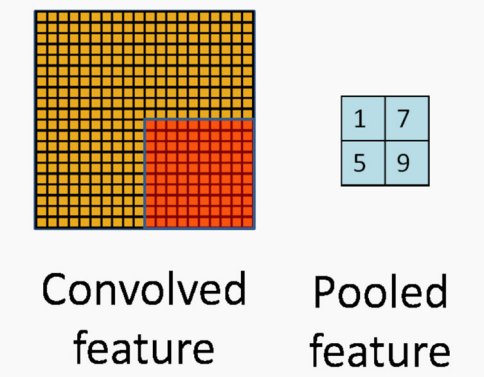


Figure 2.5: Max pooling. Pooling reduces the number of parameters by applying pooling units over 4 non-overlapping regions of the image, as the authors of [39] describe.

Pooling layers summarize the neighborhoods of output units and replace them with one value in the kernel map. Traditionally, these neighborhoods do not overlap. However, research shows that overlapping regions reduce overfitting [6]. A pooling layer is a grid that consists of pooling units spaced s pixels apart. Each grid summarizes a neighborhood of size z×z centered at the location of the pooling unit. Traditional non-overlapping pooling is obtained if s and z are set as s = z. If s is smaller than z, i.e. the center of a neighboring pooling region lies within this pooling grid, this method is called overlapping pooling.

Several statistical approaches are possible to describe features with pooling layers in large images. One could either compute the mean or the maximum value over an image region. As seen in Figure 2.5, pooled features have lower dimensionality and can also improve results due to less overfitting. The most widely-used pooling operations are 'max pooling' and 'average pooling'.

2.1.1.3 Learning

The typical way a cost function is minimized in convolutional networks is using a stochastic gradient descent (SGD) function. The standard form of a SGD is

$$\theta^{k+1} = \theta^k - \epsilon_k \frac{\partial L(\theta^k, z)}{\partial \theta^k} \tag{2.2}$$

where θ are the learning parameters and ϵ is the learning rate, [40]. A similar concept is the momentum: instead of the current gradient, a moving average of the past-gradients

is used in the updated equation. The amount of weight given to the momentum can be controlled by an additional parameter. AlexNet [6] is trained using a stochastic gradient descent function with momentum. The update rule for weight w is

$$v_{i+1} := 0.9v_i - 0.0005\epsilon w_i - \epsilon \left\langle \frac{\partial L}{\partial w}|_{w_i} \right\rangle_{D_i} \tag{2.3}$$

$$w_{i+1} := w_i + v_{i+1} \tag{2.4}$$

where i is the iteration index, v is the momentum variable, ϵ is the learning rate, and $\left\langle \frac{\partial L}{\partial w}|_{w_i} \right\rangle_{D_i}$ is the average over the i-th batch D_i of the derivative of the objective with respect to w, evaluated at w_i. Krizhevksy [6] found that the small weight decay of 0.0005 is important for learning. The momentum parameter for the gradient descent function is set to 0.9.

2.1.1.4 ReLU

The standard way to model a neuron's output f as a function of its input x is $f(x) = tanh(x)$ or $f(x) = (1+e^x)^{-1}$. However, both nonlinearities are saturating and are much slower than the non-saturating nonlinearity Rectified Linear Unit (ReLU) $f(x) = max(0, x)$, as defined in [41]. ReLUs are non-saturating, which means the neuron learns if there is at least one positive input to a ReLU. Networks with ReLUs train several times faster than those using *tanh* functions. The desired training error rates of CNNs with ReLUs are achieved several times faster than with saturating neurons [6]. ReLUs are important because the learning speed has a large influence on the model's performance, especially if the dataset is very large.

2.1.1.5 Dropout

'Dropout' is a technique which helps to avoid overfitting by setting the outputs of hidden neurons to zero with a pre-set probability. In AlexNet, the standard value for dropout is 0.5, and for GoogLeNet it is 0.7. The neurons which are set to zero do not contribute to the forward pass or back-propagation, i.e. they are 'dropped out'. Due to these 'dropped-out' neurons, the network samples a different architecture every time. This forces neurons to learn more robust features as they cannot rely on the presence of other neurons. A dropout ratio of 0.5 doubles the number of iterations required to converge, but reduces overfitting.

2.1.1.6 Softmax

In image classification tasks, the last layer in a convolutional network is typically a softmax layer. It is used to normalize a K-dimensional vector z of real values to a K-dimensional vector $\sigma(z)$ of real values between 0 and 1. The function's output represents the probabilities of the image belonging to class j

$$\sigma(z)_j = \frac{e^z_j}{\sum_{k=1}^{K} e^z_k} \tag{2.5}$$

for $j = 1, ..., K$. Due to the exponential component in the equation, the probability of the most probable label tends to be very large (ranging between 0.9 and 1.0 in many cases).

2.1.2 Architecture

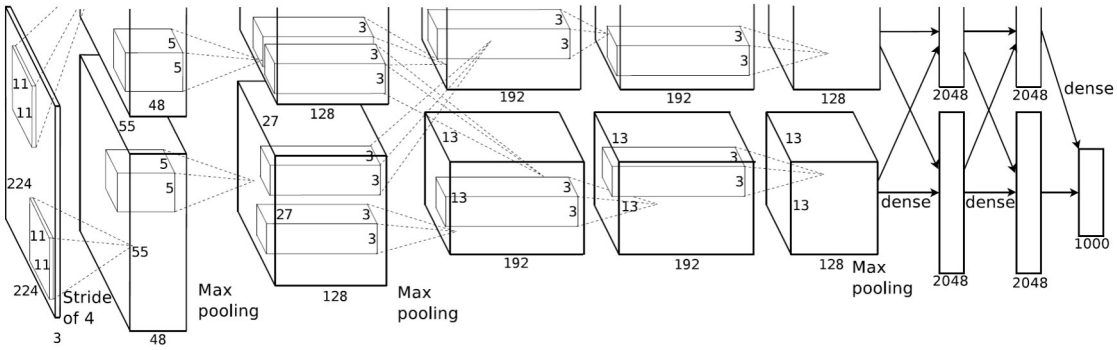


Figure 2.6: An illustration of the architecture of Krizhevsky's network, showing the separation in blocks for two GPUs running in parallel. One GPU runs the blocks at the top of the figure while the other runs the blocks at the bottom. The convolution, max-pooling and normalization operations take place between the blocks. The used filtering kernels of each layer are illustrated as cuboids in the previous layer. In this figure, the block's notations are three-dimensional: a block of size 13×13×192 is equivalent to 192 feature maps of size 13×13. Source: [6, 28]

Figure 2.6 presents Krizhevsky's AlexNet which contains five convolutional (Conv layer) and three fully-connected layers (FC layer). Before we analyze the overall eight layers of this network, we describe the major differences and important features that Krizhevsky considered very important in the network's architecture. Among them is the use of the ReLU Nonlinearity, Local Response Normalization, and overlapping pooling.

The concepts of ReLU Nonlinearity and overlapping pooling are already described in the previous sections 2.1.1.4 and 2.1.1.2, respectively. The kernels of the third convolutional

layer are connected to all kernel maps in the second layer. The units in the fully-connected layers are connected to all units in the previous layer. The Local Response Normalization normalizes the activity of a neuron by the total number of kernels in the layer. This normalization is applied in the first two convolutional layers and reduces the error rates by approx. 1 percent. Overlapping max-pooling with a stride of 4 pixels is applied after response-normalization layers as well as after the fifth convolutional layer. ReLU non-linearity follows every convolutional and fully-connected layer.

The overall architecture of AlexNet is described in three-dimensional kernels, where the third dimension is the number of feature maps. For example, a kernel of size 32×32×10 is equivalent to 10 feature maps of size 32×32. Because the description about the architecture is based on [6, 28], we use Krizhevsky's notations in the following.

The network requires input images of size 224×224×3 (RGB image). The first convolutional layer C1 filters the image with kernels of size 11×11×3. Because of the stride of 4, these kernels overlap and 96 kernels are needed to filter the image. Because the network is illustrated as a two-GPU net, it divides the kernels into two parts. Instead of one block (or kernel map) with size 55×55×96, the resulting kernel is "split" into two blocks of size 55×55×48. The output is response-normalized and max-pooled, and passed to the second convolutional layer C2. Layer C2 uses 256 kernels of size 5×5×48, and normalization and max-pooling is applied again. This operation leads to two blocks of size 27×27×128. The following convolutional layers do neither apply max-pooling nor normalization. Layer C3, which is connected to both kernel blocks of Layer C2, filters the output with 384 kernels of size 3×3×256. The resulting two blocks are of size 13×13×192. Layer C4, where each kernel map is only connected to one of C3, uses 384 kernels of size 3×3×192 to filter the output. The fifth convolutional layer C5 has 256 kernels of size 3×3×192, the resulting kernel maps are of size 13×13×128. The next three layers are fully-connected layers. Layers FC6, FC7 and the last fully-connected layer FC8 have 4096 units each. Layer F8 produces an output that is passed to a 1000-way softmax that computes a probability distribution over 1000 class labels. 1000 classes are used because AlexNet competed in the ImageNet classification challenge, where 1000 classes exist.

The number of units in each layer can be computed as a multiplication of each layer's dimensions. The network's input is $224 \times 224 \times 3 = 150,528$ dimensional. The number of units in layers C1 and C2 are $55 \times 55 \times 96 = 290,400$ and $27 \times 27 \times 256 = 186,624$, respectively. The number of units in the remaining layers are 64,896 (both C3 and C4), 43,264 (C5), 4096 (both FC6 and FC7), and 1000 (FC8).

In the ImageNet LSVRC-2010 contest, the convolutional network achieved top-1 and top-5 error rates of 37.5 percent (62.5% accuracy) and 17.0 percent (83% accuracy), respectively. Most labels in Figure 2.7 are reasonable alternatives for each predicted label. For example, only other types of cat (jaguar, cheetah, snow leopard, and Egyptian cat) are considered in the top-5 for the correct label leopard. However, the cases 'grille' and 'cherry' show that some images could have multiple labels as ambiguities exist.

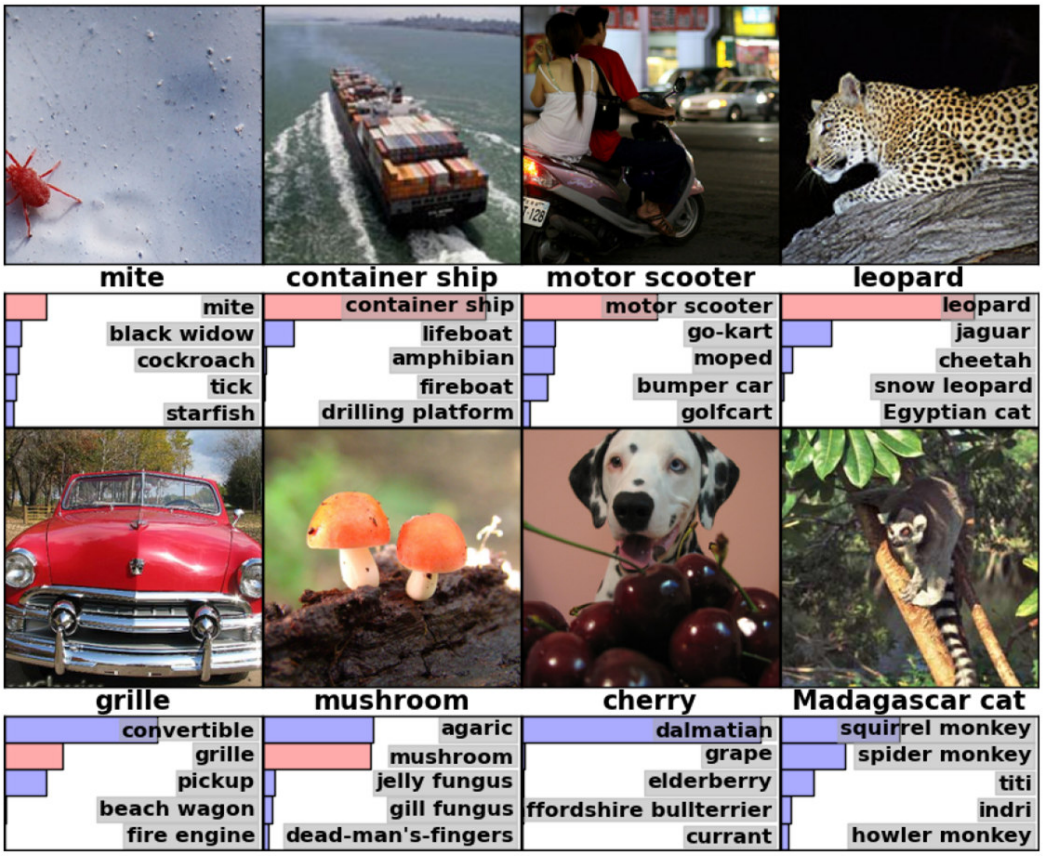


Figure 2.7: ImageNet test images with the five most probable labels. The correct label is shown with a red bar (if it is in the top 5). Source: [6]

2.1.3 New applications for deep learning

Due to the large success of deep learning in computer vision and natural language processing, deep learning algorithms have started to be commercialized [42] in many applications ranging from bank check reading systems, pedestrian and traffic sign detection [43], video surveillance, face detection, to off-road mobile robots for vision-based obstacle avoidance [44]. Motivated by many breakthroughs in recent years in this field, researchers are exploring new ways to leverage the benefits of deep learning and apply them on new problem domains. In this section, we will go into further detail about recent research trends in deep learning. We showcase a few examples where convolutional neural networks have recently been applied. These may very likely enable new application areas for artificial intelligence algorithms in the next years and decades to come.

Figure 2.8: The relocalization results for Alex Kendall's deep convolutional neural network camera pose regressor [45]. Top: input images. Middle: predicted camera poses of the corresponding visual reconstructions. Bottom: input images are shown again with middle images overlaid in red.

2.1.3.1 SLAM

One of the classic problems in mobile robotics is the Simultaneous Localization and Mapping (SLAM) problem. SLAM [46] allows robots to localize and navigate in unknown environments autonomously. SLAM is hard because it estimates both the pose of a robot and map the environment at the same time. Therefore, it is often referred as a 'chicken-or-egg' problem. Kendall et. al show in [45] that point-based SIFT registration fails when difficult lighting, motion blur, and different camera intrinsics occur. Their proposed system is a 23 layer deep network that is able to be used for out of image plane regression problems. The system regresses the 6-DoF camera pose from a RGB image and obtains 2m and 3 degrees accuracy outdoors and 0.5m and 5 degrees accuracy indoors. The system is trained end-to-end from RGB images to the 6-DOF camera pose without additional techniques such as hand-engineering features or graph optimisation. Four examples of predicted camera poses are shown in Figure 2.8.

2.1.3.2 Semantic Segmentation

Recent research has shown that convolutional networks, which are trained end-to-end and pixels-to-pixels, exceed the state-of-the-art in semantic segmentation [9, 47]. Figure 2.9 shows the SegNet encoder-decoder architecture which is used for pixel-based prediction in [47]. The encoder network is topologically identical to the VGG network [48]. The decoder network maps the feature maps from low resolution to full input resolution feature maps for pixel-wise prediction. SegNet is developed for autonomous driving applications to enable vehicles understand road scenes. Example images in Figure 2.10 demonstrate

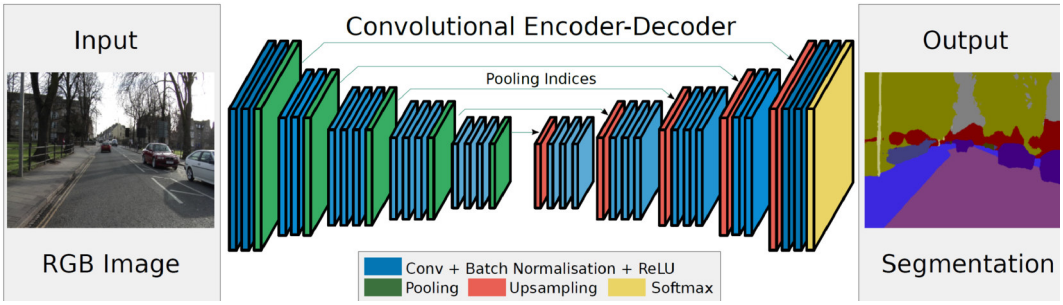


Figure 2.9: SegNet Architecture. The network is trained pixel-wise and segments the road scene in 11 different class color codes. Source: [47]

how SegNet can segment the input images into different class color codes. Compared to other methods, SegNet and its variations outperform the other benchmarked algorithms in 10 of 11 classes, with an average accuracy of 88.5% over all classes.

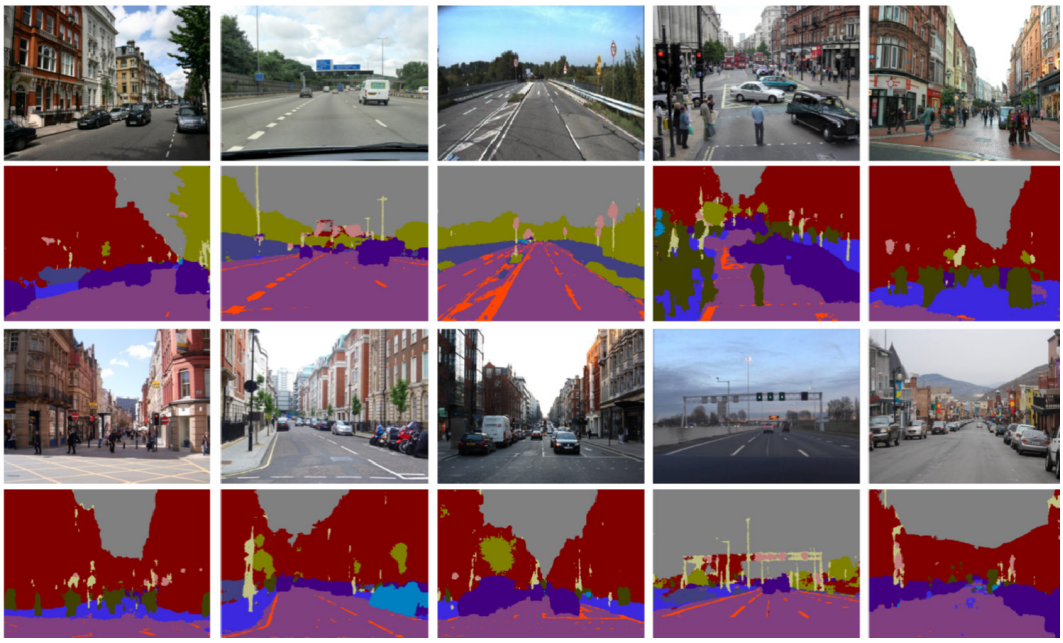


Figure 2.10: Example images for SegNet. The road scene images are segmented in 11 different class color codes. Source: [47]

2.1.3.3 Natural Language Processing and Sequences

Another application area for deep learning is natural language processing. In 2014, Vinyals et. al introduced the Neural Image Caption Generator (NIC) [15], which automatically describes the content of an image in one or two sentences. NIC consists of a deep convolutional neural network (CNN) and a recurrent neural network (RNN). This

Figure 2.11: Google's Neural Image Caption Generator [15], which consists of GoogLeNet [7] (top) and a language generating RNN (middle), is used to describe the content within images in sentences. Examples are shown in the bottom.

structure is mainly motivated by the recent advances of RNNs in machine translation tasks. One RNN can be used to translate a source sentence from one language into a vector representation, from which a second RNN can produce the target sentence. Instead of the first RNN, NIC uses a convolutional neural network to encode the input image into a rich feature representation. Their CNN is GoogLeNet, a network which finished 1st place in the ILSVRC 2014 classification challenge with a top-5 error rate of 6.67 percent. For comparison, Krizhevsky's network [6] achieved 16.4 % in the same competition in 2012. The name GoogLeNet is an homage to LeCun's LeNet-5 [30]. It consists of 22 layers and is created by stacking several, so-called Inception modules on top of each other. The author Szegedy argues that a high dropout rate of 0.7 is essential for a good performance of GoogLeNet. The NIC model is trained to maximize the likelihood p(S—I) of the target sentence given the training image. Figure 2.11 shows the architecture of GoogLeNet (rotated by 90 degrees), NIC, and good and bad examples of NIC. In the image notation examples, the sentences and images above the green box perform well. Others above the red box show classification errors.

Another example in recent research where CNNs have been applied with RNNs is [49]. As shown in 2.12, the proposed system is able to give correct answers based on an input image and a question.

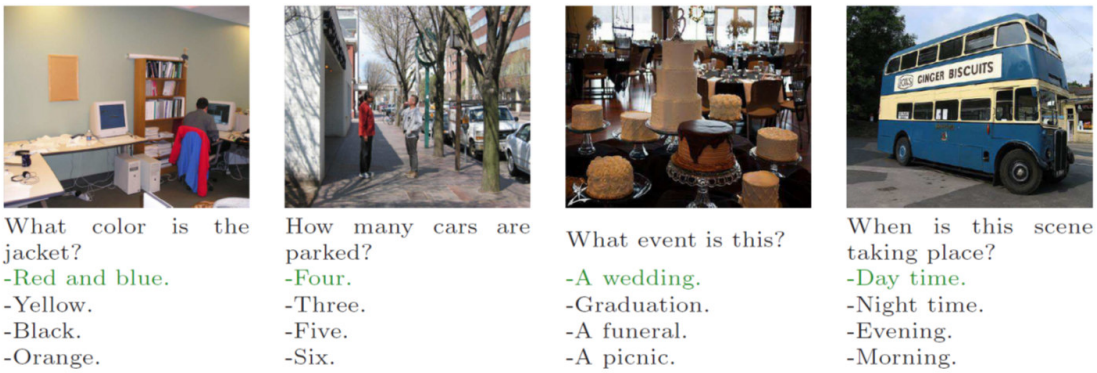
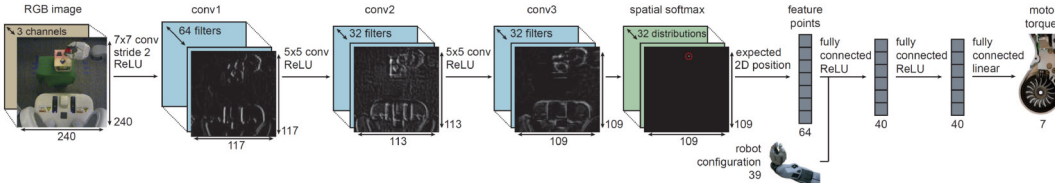


Figure 2.12: Four images are shown with associated questions and answers from the Visual7W dataset. The system provides correct answers to the questions given the input images. Source: Facebook AI Research [49]

2.1.3.4 Deep Reinforcement Learning

Deep Reinforcement learning, which is the combination of deep convolutional networks with reinforcement learning algorithms, has drawn a lot of attention in the media recently, especially for AlphaGo [18] and playing Atari games [17]. Levine et. al show in [19] that it is possible to apply deep reinforcement learning in robotics. Figure 2.13 shows the

architecture, which maps the input image to the robot actuators, and an overview of the four trained tasks.



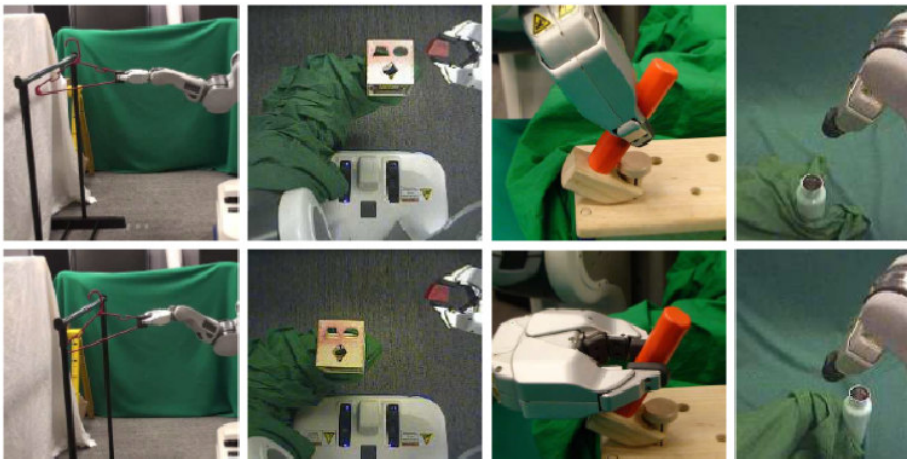


Figure 2.13: Visuomotor policy architecture. Source: [19].

The network consists of three convolutional layers, one spatial softmax and an expected position layer which transforms features from the pixel space to feature points, which are more beneficial to compute spatially. The computed points are passed through three fully connected layers with the robot configuration to control the robot's motor torques. The four basic tasks which are successfully learned end-to-end are placing the objects hanger, cube, hammer, and bottle into predefined positions and performing object manipulations.

Another research project which addresses end-to-end learning of raw image data to the action output is [50]. The used system learns to become a self-driving car, where the CNN directly produces the steering command. The system was never trained with lane markings, but is able to learn internal representations from the training data, such as detecting useful road features with only the human steering angle. Results show that the system has an autonomy value of 98%, meaning that the driver needs to intervene for 12 seconds total in a driving time of 600 seconds.
The previous examples show that neural networks are applied and tested in novel areas, where they can be trained end-to-end and outperform previous methods or provide a new way to quickly deploy a system which automatically learns the required features. As of

today, a lot of research and commercial interest focuses on this approach, which enables new applications that were not possible before.

2.2 Regularized Least-Squares Classification

In this section, we describe the Regularized Least-Squares Classification [51], which is the underlying machine learning algorithm in the GURLS library which we used in our experiments.
The Regularized Least-Squares Classification (RLSC) algorithm is based on the solution of binary classification problems via Tikhonov regularization using the square loss in a Reproducing Kernel Hilbert Space. The authors demonstrate that its performance matches the accuracy of Support Vector Machines (SVM) on several datasets. Instead of solving a quadratic problem when training an SVM, RLSC requires only the solution of a set of linear equations. The authors define the RLSC problem as

$$minF(c) = min_{c\in\mathbb{R}^n}\frac{1}{2n}\sum_{i=1}^{n}(y - Kc)^T(y - Kc) + \frac{\lambda}{2}c^T Kc, \tag{2.6}$$

where n is the number of training examples, K is the positive semidefinite kernel matrix, y is the labels vector, written as $y_i \in 1, ..., T$ for $i = 1, ..., n$, and c is the optimal solution to the above problem. This function is minimized by its derivative with respect to c. Because it is allowed to set the derivative to zero, it can be seen that the solution of a RLSC problem can be found by solving a single system of linear equations:

$$(K + \lambda lI)c = y \tag{2.7}$$

By setting the derivative of a differentiable Lagrangain function equal to zero, the maximum of the Lagrangian function with respect to a variable needs to be computed (not shown here). After solving the concave maximization problem for this variable, the resulting c satisfies the desired value for solving the equation 2.7.
In this thesis, we chose to use the RLSC algorithm instead of support vector machines because we first reimplemented our object recognition system based on [52]. The authors Pasquale et. al stress that they rely on GURLS to perform RLSC, which is the natural variant of the classic regularized logistic regression (RLS) algorithm. They experienced from previous work on the iCub that the used algorithm achieves comparable or better results than the previously tested LIBLINEAR [53] and SVM [54] libraries.

2.3 Dimensionality Reduction Techniques

In this work, we extract high dimensional features from our dataset. These extracted features are used to train our classifiers which make predictions on unknown data. Because

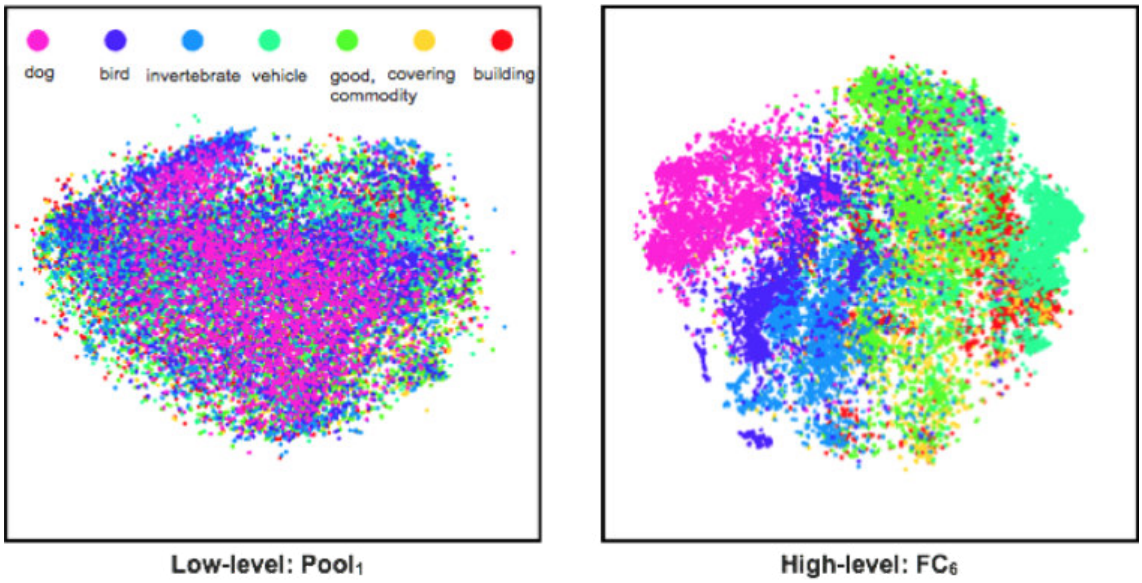


Figure 2.14: The visualization shows the result of a dimensionality reduction of high-dimensional features reduced to a two-dimensional subspace of the data. The deep features illustrate that deep networks are very effective as feature extractors. Source: [55]

the features are 4096-dimensional, it is very difficult and challenging to visually analyze the data and determine which of those features contain the relevant information. The problem of extracting information from high dimensional data is solved by dimensionality reduction techniques [56]. The goal is to extract only a few reasonable features which contain most of the information available in the 4096-dimensional features. For example, Figure 2.14 illustrates how high-dimensional features can be geometrically projected onto a two-dimensional subspace of the data. In this thesis, we apply a dimensionality reduction technique to visually analyze the relationship between the 4096-dimensional features. Mathematically, the problem of dimensionality reduction can be stated as [56]: For a p-dimensional real valued random variable $X = [X_1...X_p]^T$, an algorithm : $\mathbb{R}^p \rightarrow \mathbb{R}^k$ with $k << p$ reduces the dimensionality of the data, such that $S = f(X)$ contains as much of the original information, possibly expressed by the variance, as possible. In the following subsections, we first give an overview about PCA and afterwards explain the t-SNE algorithm, which we use in this work.

2.3.1 Principal Componant Analysis (PCA)

The Principal Component Analysis (PCA) is among the most well-known unsupervised reduction techniques. Unsupervised means that the data does not have to be labeled prior to applying the algorithm. PCA [56, 57, 58] is a linear reduction technique which finds a reduced number of independent variables that contain as much of the original variance as possible. From a geometric interpretation, PCA projects the data onto a lower

dimensional subspace. The number of dimensions in the subspace is k, which can be freely chosen dependent on which lower dimension is desired or how much of the variance should be included in the reduced data. When choosing $k = 2$, Figure 2.15 shows the geometric

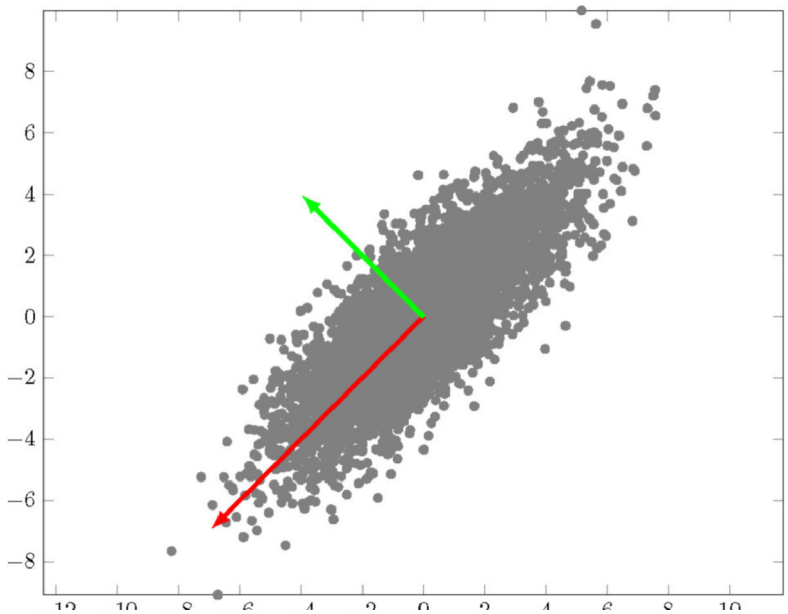


Figure 2.15: The geometric interpretation of PCA for $k = 2$. The first principal component, which is the eigenvector corresponding to the largest eigenvalue, is colored in red. The second principal component is orthogonal to the first and is colored in green. These principal components contain as much of the original variance as possible for two dimensions. Source: [56]

interpretation of the largest two eigenvectors, or principal components. To apply PCA, the first step is to apply the singular value decomposition (SVD)

$$X = U\Sigma V^T, \tag{2.8}$$

where U is a unitary matrix containing orthogonal column vectors, Σ contains the positive-semidefinite singular values σ of X in a rectangular diagonal matrix, and V is the matrix containing the right singular vectors of X, whose unit vectors are also orthogonal. Then, U_k contains the first k columns of U, which minimizes the equation

$$J(U_k) = \sum_{i=1}^{n} \left\| X_i - U_{[}kU_k^T x_i \right\|_2^2. \tag{2.9}$$

The reduced and uncorrelated data is computed as

$$S := U_k^T X. \tag{2.10}$$

2.3.2 t-Distributed Stochastic Neighbor Embedding (t-SNE)

In this thesis, we use the t-SNE algorithm [59, 60, 61, 62] to visualize our 4096-dimensional features in a two-dimensional map. We chose this method because leading researchers in deep learning have used this algorithm to show the underlying structure of large datasets in two or three dimensions. The authors of t-SNE argue that the algorithm provides significantly better results than other techniques on almost all datasets, especially when the data is high-dimensional and lie on different, but related, low-dimensional manifolds. This is the case in images of objects that are shown from different viewpoints and are split in separate classes [59]. The algorithm is based on Stochastic Neighbor Embedding [63] that is much easier to optimize. Several implementations of the algorithm exist. The Barnes-Hut approximation can speed up the computation time of the algorithm. Laurens van der Maaten applied the technique on datasets with up to 30 million examples. Figure 2.16 describes the basic concepts of the t-SNE algorithm. Graphical comparisons of dimensionality reduction and visualization techniques are shown in Figure 2.17.
In the following, we explain the background of the algorithm in more detail based on [59]:

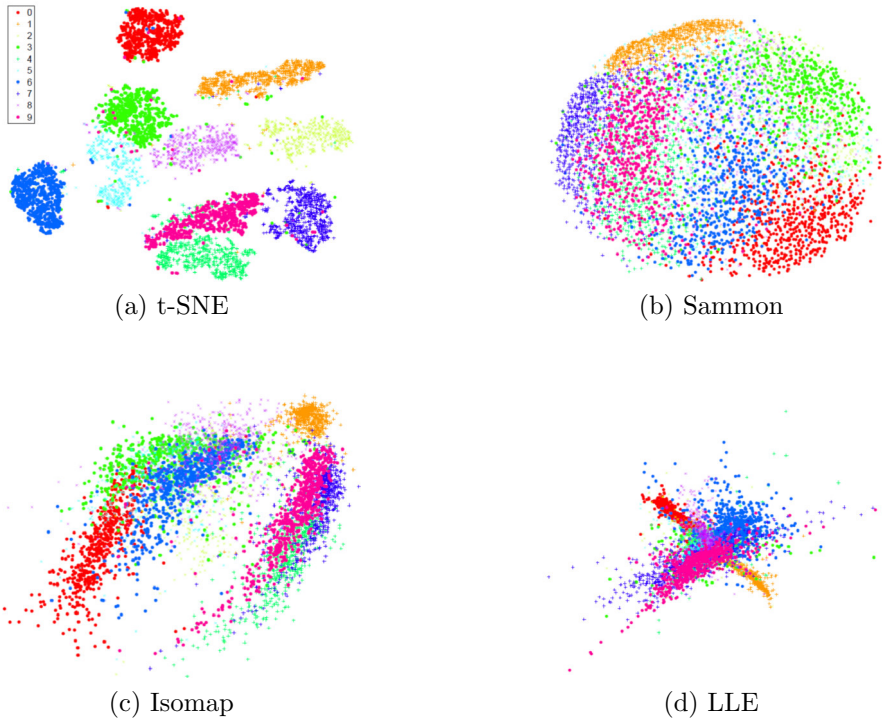
Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity *Perp* (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for *t=1* **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

Figure 2.16: Overview of the t-SNE algorithm in pseudo-code. Source: [59]

The t-SNE algorithm minimizes the difference between two distributions. The first measures pairwise similarities of the input objects and the second of the corresponding low-dimensional points. Assume $x_1, x_2, ..., x_N$ are a set of high-dimensional data points, the distance between a pair of objects is computed by the Euclidean distance $d(x_i, x_j) = \|x_i - x_j\|$. Joint probabilities p_{ij} are computed by symmetrizing

$$p_{j|i} = \frac{exp(-d(x_i, x_j)^2 / 2\sigma_i^2}{\sum_k exp(-d(x_i, x_k)^2 / 2\sigma_i^2)}, \quad p_{i|i} = 0 \tag{2.11}$$



(a) t-SNE (b) Sammon

(c) Isomap (d) LLE

Figure 2.17: Visualizations of the MNIST dataset with 6,000 handwritten digits. The t-SNE algorithm (top left) is able to provide the best two-dimensional map compared to the Sammon (top right), Isomap (bottom left), and LLE (bottom right) algorithms. Source: [59]

with

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}. \tag{2.12}$$

The goal is to find a s-dimensional embedding ϵ, where s is typically equal to 2 or 3. The similarities between two low-dimensional points are measured with a normalized heavy-tailed kernel:

$$q_{ij} = \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}}, \quad q_{ii} = 0. \tag{2.13}$$

According to [62], the locations of the s-dimensional embeddings are computed by minimizing the Kullback-Leibler divergence between the joint distributions P and Q:

$$C(\epsilon) = KL(P||Q) = \sum_{i \neq j} p_{ij}\ log\frac{p_{ij}}{q_{ij}}. \tag{2.14}$$

In the embedding ϵ, the objective cost function is non-convex and minimized by

$$\frac{\delta C}{\delta y_i} = 4 \sum_{j \neq i}(p_{ij} - q_{ij})q_{ij} \sum_{k \neq l}(1 + \|y_k - y_l\|^2)^{-1}(y_i - y_j). \tag{2.15}$$

2.4 Attributes and Affordances

In this section, the presented attributes and affordances describe the contextual information in our experiments. The authors of [26] propose to extend the classical object recognition problem from naming object classes to describing attributes. This approach makes it possible to learn and recognize new objects by being able to describe attributes on unknown objects. Similarly, the goal in this thesis goes beyond classical object recognition in our experiments. Objects are recognized and described with three attributes and one affordance. The models that predict the labels for attributes or affordance are trained on our dataset. A challenge that exists during training is generalization. It is important that the attributes can be learned correctly and wrong relationships are not learned instead during training.

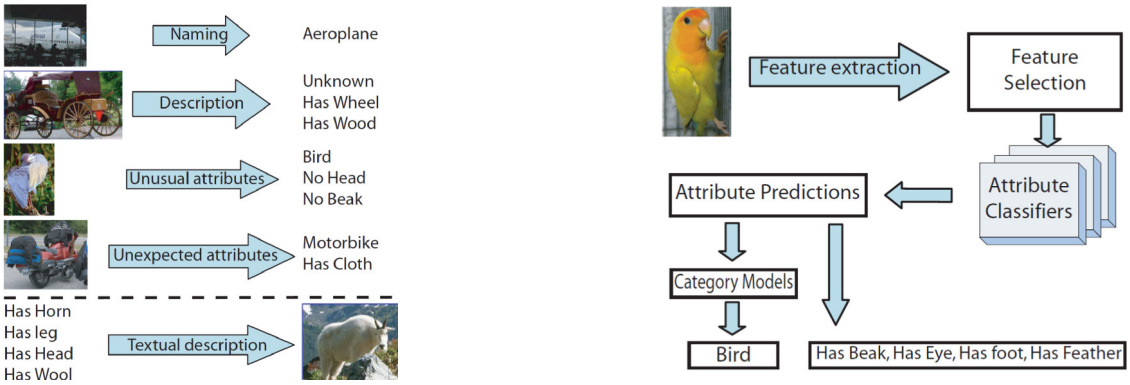


Figure 2.18: The methodology by the authors of [26]. Left image: the attribute-based approach allows to describe unknown objects with textual descriptions. Right image: extracted features of the image are used to train attribute classifiers. These can make predictions about attribute classes on unknown object categories. For each of the evaluated attributes, the authors selected beneficial features for training the classifiers.

The left diagram in Figure 2.18 gives an overview about the attribute-based approach of Farhadi et. al [26]. Recognizing attributes in images allows to describe images with unknown object categories, atypical attributes of known classes, and learn new models based on the predicted textual description. The right image shows that features are extracted from the images. The features are chosen that are beneficial for training attribute classifiers. The trained classifiers predict attributes in images. These predicted attributes can be either used to learn and describe object categories such as the class 'Bird', or describe unknown object classes with attributes. Farhadi et. al describe visual aspects with feature representations: they use color and texture for materials, visual words for parts, and edges for shapes. For each of those types, bag of words style features are used. They rely on texture descriptor [64] and the kmeans algorithm for quantization to 256 centers. HOG spatial pyramids are used to construct visual words. The HOG descriptors are also quantized to 1000 kmeans centers. Canny edge detectors are used to find edges and their orientations,

which are saved into 8 unsigned bits. Color descriptors, based on LAB values, are sampled for each pixel and also quantized to the nearest 128 centers using the kmeans algorithm. These base features are used to generate histograms for each feature type of each cell in a grid. Stacked together, the feature vector is 9751 dimensional.
In our system, we make use of the approach summarized in the right image. Unknown classes are described with attributes that are learned on our collected dataset. Our system differs that we do not preselect features which are relevant to the learned attributes. We train the models end-to-end from the images to the attribute labels in Caffe [55]. For GURLS [65], we extract 4096-dimensional features which are directly used to train the attributes. Thus, their feature vectors are more than twice larger than ours. In the following, we explain attributes and affordances based on [26].

2.4.1 Attributes

Attribute learning is difficult because the learned patterns must be generalized on unknown data. Three main types of semantic attributes are considered in [26]: material (13 classes), shape (5 classes), and part (46 classes). In total, they use 64 different attributes. 'Material' consists of the classes 'has wood', 'is furry', 'has glass', and 'is shiny'. For attribute 'Shape', they use 'is 2D boxy', 'is 3D boxy', and 'is cylindrical'. The attribute 'Part' describes visible parts, e.g. 'has head', 'has leg', 'has arm', 'has wheel', and 'has window'.
In comparison, we ignore the attribute 'Part'. Because our dataset has a small number of classes, our models would not be able to generalize the learned parts on unknown data. However, we adopted the material and shape labels and modified them to fit to our dataset. We use basic 3D shapes for labeling our objects. These are 'sphere', 'cuboid', 'cylinder', and 'none'. For material, we use 'textile', 'ceramic', 'plastic', 'organic', and 'none'.
Another attribute which is very popular is color, such as described in [66]. In our system, we have defined 10 different colors: 'black', 'brown', 'grey', 'yellow', 'red', 'orange', 'blue', 'green', 'white', and 'none'.

2.4.2 Affordance

The concepts of affordances is linked to action possibilities on an object or on the environment. Originally proposed by J.J. Gibson in [67] for use in psychology, his ideas about affordances influenced many different fields, including autonomous robotics, industrial design, communication studies and human-machine interfaces. Sahin et. al point out in [68] that there are three perspectives about affordances instead of one, and that Gibson's ideas on affordances were more a general theory than a specific theory of visual perception. In the field of human-computer interaction, affordance was defined by Norman on [68, 69] as 'the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used.' In robotics, a behavior is perceived as a sensory-motor mapping which is equal to a function from certain sensors to

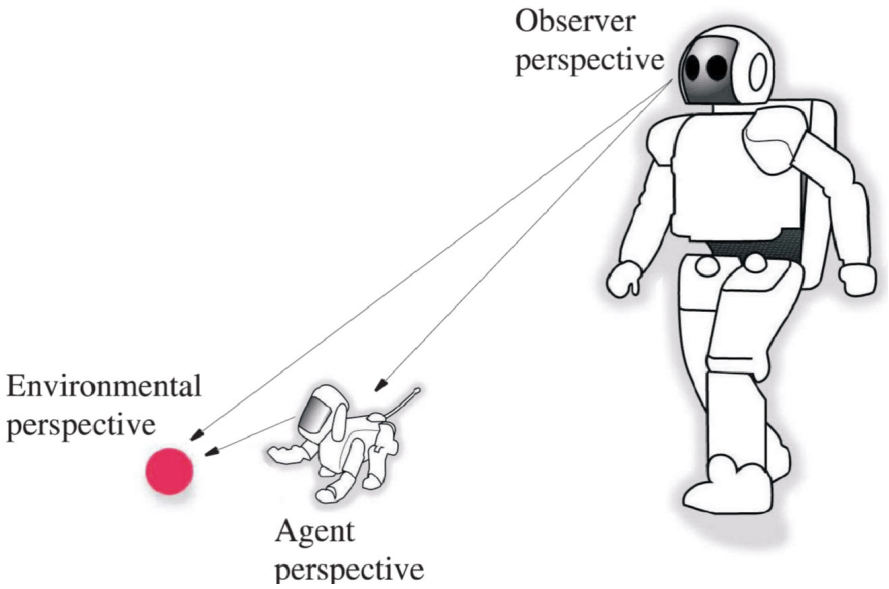


Figure 2.19: The three perspectives about affordances. 1) Observer, 2) Agent, and 3) Environmental. Source: [67]

certain actuators. Affordances in autonomous robotics are mostly used for behavior-based control. In our work, we use Norman's definition of affordance, which is the action that can be executed on the object with the robot. Because our dataset mostly contains household objects, our affordances are 'eat', 'drink', 'clean', 'work', and 'none'.

2.5 Semantic Reasoning

Semantic reasoning is the science which deals with inferring logical consequences from a set of facts or rules. These rules are commonly defined and specified in an ontology. The term ontology is historically originated from a branch of philosophy, which computer scientists adopted to reduce the complexity and organize the available knowledge for a particular domain. In information technology, an ontology is used as a knowledge representation model that defines the types, properties, and relationships of entities. Ontologies are written in the Web Ontology Language (OWL)[70, 71], which is the recommended ontology language by the World Wide Web Consortium (W3C). It is based on Description Logics [72] and supports the use of IRIs, XML schemas and importing of ontologies from the web. At the Institute for Cognitive Systems, we use semantic reasoning techniques to equip our iCub robot with high level understanding and cognitive reasoning capabilities [73]. In recent research, this enabled the robot to recognize activities and intentions of human activities [74, 75, 76] and transferring skills to humanoid robots [77]. In [75], a deep learning algorithm was used to improve the recognition performance of human actions with spatio-

temporal feature learning.
In this study, semantic reasoning is used to enhance the object recognition algorithms with logical reasoning capabilities. Our main goal is not only to enable our iCub robot to correctly classify shown objects, but to demonstrate that it can learn new objects by inferring semantic attributes based on contextual information from objects.

2.6 Related Experiments

In the previous sections, we described the state-of-the-art in research which is related to the individual components of our system. In this section, we describe related research which focuses on either teaching a robot to recognize new objects or combining classical object recognition with contextual information. We compare our system with related experiments made on the iCub from the Italian Institute of Technology (IIT) [52]. Another approach [78] which includes natural language descriptions as contextual information is presented. Furthermore, we look into another research study from the Institute for Artificial intelligence (IAI) at the University of Bremen, which is called RoboSherlock [79].

2.6.1 iCubWorld

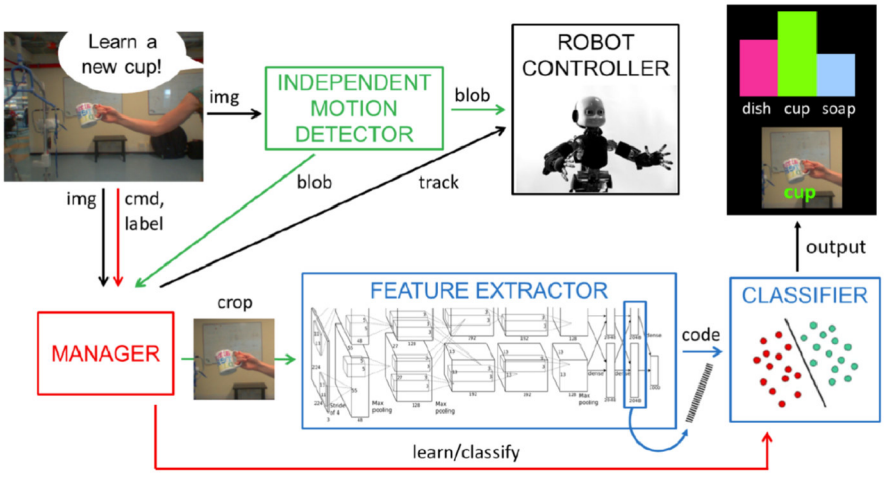


Figure 2.20: Visual recognition pipeline proposed by IIT's iCub team. Source: [52]

Pasquale et. al were the first who implemented deep convolutional neural networks to teach the iCub humanoid robot new objects [52]. They investigated how the good performance of CNNs can be leveraged to advance the visual recognition capabilities of the iCub. The performance of their system is benchmarked on the iCubWorld28 dataset. In their scenario, a human supervisor shows a new object to the robot and verbally annotates it. The robot

tracks the object and learns it. Figure 2.20 visualizes the system used on the iCub. In their system, features are extracted from the second last layer of a convolutional neural network that is pre-trained on the ImageNet dataset. The features are used to train a linear classifier, which predicts the class of the image. The central question they address in their work is: *"How many objects can the iCub actually recognize?"*[52]. The authors stress that they chose this approach since it is impractical to train deep neural networks in robotic settings. They evaluated the performance on images that were obtained on four different acquisition days for training. The results show that on average for all training days, the test accuracy is 70.3%. If a predictor is learned only on one acquisition day, the performance lies between 53.5% and 64.8%. Therefore, it is lower than when all days are trained. The authors also report the maximum number of objects that the iCub can recognize: for a confidence level of 98%, the iCub robot can recognize two (of 28) objects. The number of correctly recognized objects increases to 14 for a level of 50%, which is every 2nd object from the iCubWorld28 dataset. The authors mention that the visual recognition problem in robotics is still far from being solved. They propose to create a new dataset which contains more object instances per category and more acquisition sessions under different lighting conditions. Another proposal from the researchers is to apply an online active learning technique that is able to let the robot decide when it acquired enough images to correctly classify an object with the required confidence level. According to the authors, this idea goes one step in the direction to reduce the number of training images and improve the overall recognition by taking contextual information into account.

2.6.2 Visual one-shot learning

In [78], the authors Krause et. al present an approach to quickly teach a robot a new object by providing contextual information based on natural language descriptions from the human teacher. The authors state that the robot is immediately able to recognize the described object after learning, which is critical for many human-robot interaction scenarios in the real-world. The investigated method is called "one-shot learning", which enables an agent to learn something from just one example. Together with the visual input, the robot receives a linguistic label to be able to recognize other instances of the same object type. The system is motivated by recent research that started to utilize contextual information in one-shot learning in cognitive architectures. Their approach to detect objects is language-guided: objects are described in natural language that obtains *"object categories, object parts, surface patterns and symbols, object characteristics, spatial and merelogical relations, and others"*[78]. For example, "a cup can be described as a cylindrical container with a round handle attached on one side"[78]. The adjectives 'Cylindrical' and 'Round' are considered as visually perceivable properties of objects, which we consider similarly as attributes in this thesis. The words 'Container' and 'Handle' refer to parts of the perceived object. The difference between their work and ours is that they use natural language processing to deliver the relevant contextual information. When an object is new, the robot can ask questions about the object properties to the human supervisor. Based on

this information, the robot performs a visual search for the object. If the robot was able to recognize and learn the object, the robot can confirm the request.
In contrast, contextual information in our work is detected by prior trained deep neural networks. That means the human supervisor corrects misclassified labels if the robot is not able to guess the object.

2.6.3 RoboSherlock

RoboSherlock [79] is an open source software framework that can be used to implement perception systems for everyday manipulation tasks. The system (shown in Figure 2.21) is able to answer task-related queries about the objects located in the scene and supports reasoning about objects. The researchers recognize that it is insufficient for a robot to only

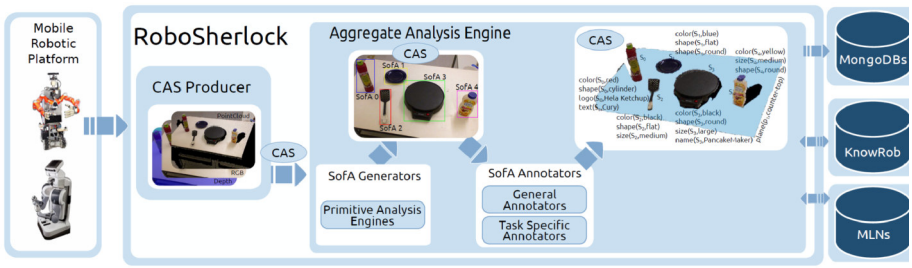


Figure 2.21: RoboSherlock. Overview of the system. Source: [79]

classify an object without further analyzing the object and decomposing it into its functional parts. RoboSherlock enables the robot to examine detected objects in a scene based on their 3D shape, pose, and state. It further enhances the perception of the robot with knowledge and reasoning based on the detected objects and environment context. RoboSherlock uses unstructured information management (UIM) to realize the perception system which is based on content analytics. The capabilities of the robot are tested in a kitchen scenario with everyday objects. The authors use OpenEase as knowledge base for their experiments. RoboSherlock receives perception tasks such as 1) looking for an object that can hold one liter of water or 2) finding a Kellog's cornflakes box on the table. These tasks are challenging and therefore require the robot to combine its perception capabilities with knowledge and reasoning methods. As sensor input, it uses RGB, RGB-D, stereo cameras. a thermo camera and other sensors. For each detected object, a designated data structure called Subject of Analysis (SOFA) is created, that collects and organizes information that different RoboSherlock components infer about the object hypothesis [79]. The RoboSherlock data structure Common Analysis Structre (CAS) consists of the images combined with meta data. The system is able to analyze a scene and runs SOFA annotators which labels the objects with color, shape, size, and if applicable, logo. Their proposed system is capable of performing everyday manipulation tasks by leveraging the use of unstructured information management. The resulting system processes information about the objects

located in a scene and can apply reasoning for interpreting perception information. However, their system is not developed to teach a robot to learn new objects. Furthermore, it does not use deep learning algorithms for recognizing objects and contextual information. The RoboSherlock system is a framework which is intended to be primarily used in object manipulation tasks. In this thesis, we concentrate on learning unknown objects.

Chapter 3

System Design

Figure 3.1: Researchers from IIT were the first who explored the use of deep convolutional neural networks for teaching iCub to recognize new objects [52].

This section provides an overview of the experimental setup and frameworks which have been used in this research work. The first part describes the humanoid robot iCub (shown in Figure 3.1) and the experimental setup at the Institute for Cognitive Systems at TUM. The second part describes the software frameworks Caffe, GURLS, and KnowRob which are needed to develop and realize the proposed system design. In modern machine learning and computer vision applications, large datasets with human-labeled pictures are required to enable the algorithms to learn the relevant features. Therefore, Part 3 introduces the widely known dataset ImageNet and explains in further detail the evaluated datasets iCubWorld28 and TUM-ICS. As described in the following sections, the final proposed system in Section 4.2 relies on the newly created dataset TUM-ICS. After introducing the dataset and its

prospective labels, Part 4 shows the knowledge base in the form of an OWL Ontology which contains the a priori knowledge the iCub robot has about objects. Finally, our proposed enhanced deep network is introduced in Part 5.

3.1 Experimental Setup

Figure 3.2: Our iCub robot. We tested our system on the humanoid robot platform iCub at the Institute for Cognitive Systems at TUM. Top left: iCub in standby-mode. Top right: Robot learns to recognize a laptop. Bottom left: We controlled and supervised the robot learning on 4 displays with a PC (Intel i7 CPU with 8 cores, nvidia GTX 750 TI GPU, 16GB RAM), one laptop, and speakers/headphones for the speech output. Bottom right: Alternative view on the experimental setup for recognizing a new object.

The technical implementations and experiments in this work have been performed on the iCub humanoid robot platform [73, 80]. It is designed by leading universities in Europe to support collaborative research in cognitive development and achieve a greater impact by focusing on one open-source platform. The iCub is the ideal robotic platform to develop rational, social and intelligent behaviors for robotic systems [73]. The iCub is designed

with perceptuo-motor capabilities with 53 degrees of freedom [77], a cognitive system for learning and development, a software architecture and a support infrastructure that encourage research collaboration and knowledge sharing. Figure 3.2 describes the robotic setting in which we performed the experiments. As the visual recognition pipeline in the experiments focuses on processing the images from the iCub cameras and not on leveraging the iCub's full motor control capabilities, we used a total 9 DoF, i.e. 6 DoF of the head and 3 DoF of the torso to center the region of interest in the experiments. The robot arms were manually put in a non-disturbing rest position. The iCub [80] and YARP [81] software libraries are used to control the system and exchange the iCub's camera data via YARP ports. The proposed technical system in this thesis can be further adapted to any robot or technical system which has a data connection to one or multiple cameras and a powerful GPU to process visual information.

3.2 Software frameworks used for the development

This section explains the used frameworks Caffe, GURLS, and KnowRob. Each subsection explains the choice of the framework for the evaluated system and provides background information.

3.2.1 Caffe

Caffe is a modifiable framework for deep learning algorithms and provides a collection of trained reference models. It was developed by Yangqing Jia et al. from the Berkeley Vision and Learning Center (BVLC) [82]. The underlying C++ library is based on a BSD-license with MATLAB and Python bindings. An active community of computer vision researchers contribute via open source code to Caffe and collaborate on research projects, industrial applications and develop new ideas to apply deep learning beyond vision in natural language processing, motor control, and neuroscience applications.

Figure 3.3 compares Caffe with other CNN frameworks. The creators of Caffe highlight two main differentiators: the first is that integration into existing systems and interfaces is easier as the complete implementation is C++ based. After the models are trained, they can be tested on non-specialized hardware in CPU mode. The second is that training large neural networks on large datasets is computationally expensive. Caffe provides pre-trained reference models off-the-shelf which deliver state-of-the-art results. These models can be further finetuned for related tasks or datasets, or can function as a semantic feature extractor [55].

Caffe is built for modularity: the software can be extended to new data formats, network layers and loss functions. Caffe models are defined in the Google Protocol buffer format, which features a human-readable text format and interface implementations with Python

Framework	License	Core language	Binding(s)	CPU	GPU	Open source	Training	Pretrained models	Development
Caffe	BSD	C++	Python, MATLAB	✓	✓	✓	✓	✓	distributed
cuda-convnet [7]	unspecified	C++	Python		✓	✓	✓		discontinued
Decaf [2]	BSD	Python		✓		✓	✓	✓	discontinued
OverFeat [9]	unspecified	Lua	C++,Python	✓				✓	centralized
Theano/Pylearn2 [4]	BSD	Python		✓	✓	✓	✓		distributed
Torch7 [1]	BSD	Lua		✓	✓	✓	✓		distributed

Figure 3.3: Comparison of popular deep learning frameworks [82]: Core language is the main library language, while bindings have an officially supported library interface for feature extraction, training, etc. CPU indicates availability of host-only computation, no GPU usage (e.g., for cluster deployment); GPU indicates the GPU computation capability essential for training modern CNNs.

and C++. A layer defined in Caffe takes one or more data blobs as input, and yields more than one data blob as output. Layers perform forward passes and backward passes. A forward pass takes an input and produces an output, and a backward pass obtains the gradient to the output in order to compute the gradient with respect to the parameters and previous inputs. The gradients are back-propagated to earlier layers in the network.

For brewing a network with Caffe, we need to prepare and label our data. We labeled our training and validation data numerically for the respective object, attribute or affordance class. Within the Caffe framework, the images are resized to a resolution of 256×256 and a mean image over the dataset is computed. The GPU is run in batches of 16 for a total of 10,320 iterations (10 epochs). One epoch equals the number of images divided by the batch size = 16,500/16, which is the number of iterations needed to process the entire training data once. For every 1,000 iterations, we test the learned network on the validation data. The initial learning rate is set to 0.02, and is decreased by a factor of 10 every 2,500 iterations. The network is trained with momentum 0.9 and a weight decay of 0.0005. A chart visualizing the test accuracy and loss for one network is provided in Figure 3.4.

3.2.2 GURLS

In previous work, researchers from IIT successfully used the Grand Unified Regularized Least Squares (GURLS) [65] machine learning library to teach an iCub to recognize new objects [52]. The GURLS machine learning library is based on the Regularized Least Squares (RLS) loss function and comes in four different versions: the standard version of GURLS is Matlab-based and works for datasets that fit into the computer's RAM memory. bGURLS uses memory-mapped storage to compute RLS on very large matrices in Matlab. GURLS++ and bGURLS++ are the corresponding software libraries to GURLS and bGURLS, but implemented in C++ for faster performance. Previous research [52]

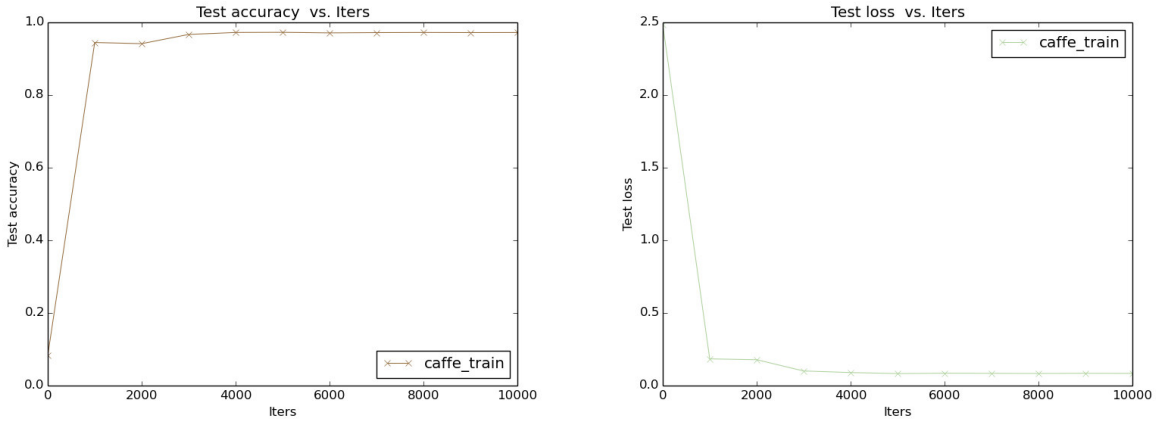


Figure 3.4: Test accuracy and test loss of our deep learning model finetuned to recognize objects. We trained the network for 10 epochs. All layers except the FC 8 layer were kept frozen during training. Thus, the performance reached high values very quickly, at about 1000 iterations.

with the iCub shows that RLS [83] achieves a comparable or higher performance than the liblinear or SVM library [37]. In [52], GURLS was used to learn objects online with incremental training data. In this thesis, we trained GURLS classifiers offline with the bGURLS library, as the machine learning classifiers need to be fully trained on the given dataset in order to learn new objects with contextual information. We chose bGURLS over GURLS because the matrices with the extracted deep features have a high dimensionality and are too large for the RAM memory. In this work, we show that the deep learning network trained on ImageNet functions as a feature extractor and GURLS is able to be trained on the features with supervised learning with good performance.

3.2.3 KnowRob

KnowRob [84, 85, 86, 87] is a knowledge processing system for robots and provides tools for knowledge acquisition, representation and reasoning. It has been developed in the IAS group at TUM and is maintained by the Institute for Artificial Intelligence in Bremen, Germany.

Figure 3.5 explains how knowledge is acquired and represented in the KnowRob system. The knowledge representation supports reasoning capabilities, provides multimodal interaction methods with humans and is integrated with the robot via ROS and JSON Prolog. KnowRob's implementation is based on the Web Ontology Language (OWL), Prolog, and Java. OWL is a description language, which represents classes of objects, properties, actions and the robot capabilities or action requirements in a XML file. The logical programming language Prolog is used to interact with OWL. It is especially good for loading,

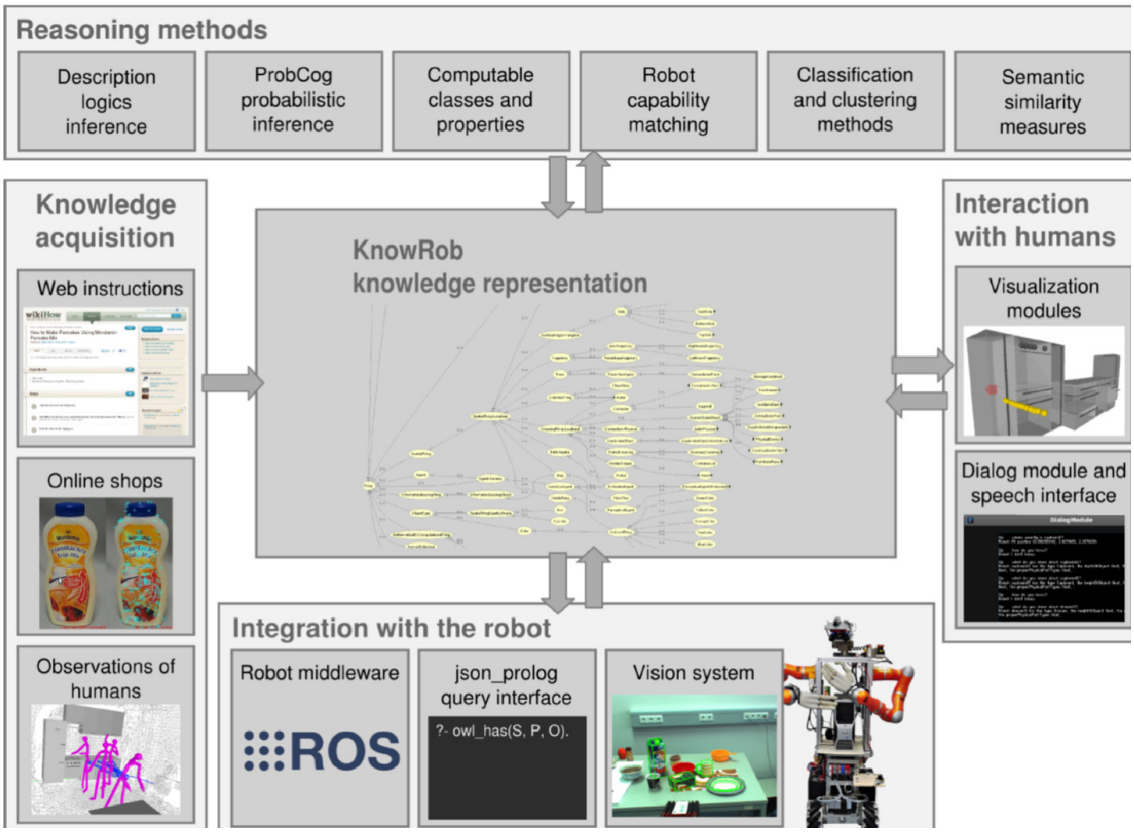


Figure 3.5: KnowRob Overview. This overview illustrates the different components in the knowledge processing system which are needed for knowledge acquisition, representation in OWL, and reasoning. Source: [84].

storing, and reasoning about the ontology. When starting KnowRob, OWL files are parsed into Prolog triples and asserted in the knowledge base as facts. To assert the knowledge, the command rdf_assert(S, P, O) is used, where S is the Subject, P is the Predicate, and O is the Object. This knowledge can be accessed with the right predicates, e.g. owl_has(S,P,O) or rdf_has(S,P,O). The reasoning engine is based on SWI-Prolog, which is a logic programming language that has bindings to Java and C++. It is possible to interact with KnowRob either from the command line via SWI Prolog or with C++ or Python interfaces via the JSON Prolog module. When dealing with KnowRob, two important concepts are considered, such as the open-world and closed-world theorems. The web ontology language OWL uses the open-world theorem, meaning that everything is assumed to be true unless we can prove that it is false. Prolog uses the closed-world theorem, i.e. everything is considered false unless something is stated or inferred that it is true.

Figure 3.6 shows the content of the standard KnowRob ontology. The ontology is divided in subclasses and provides the robot with general terms it can reason in: subclasses of owl:Thing are SpatialThing, TemporalThing, Agent-Generic, and MathematicalOrCom-

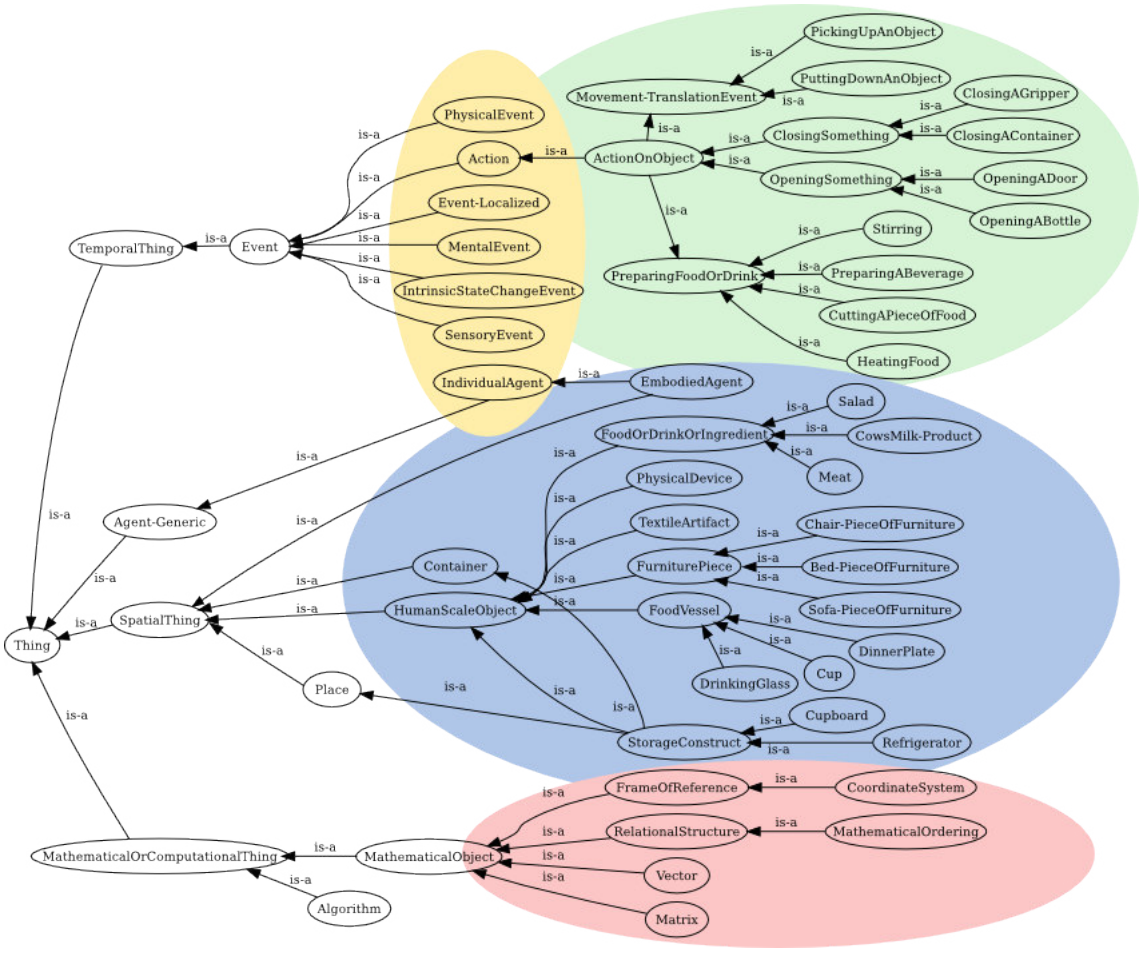


Figure 3.6: KnowRob Upper Ontology. This ontology contains the available knowledge of a robot, in which the robot can perform semantic reasoning. Source: [88]

putationalThing. SpatialThing combines all classes that are maps, points, places, trajectories, regions, and objects that are tangible and can be localized [88]. Examples for such objects are Container or FoodVessel, which can further be separated into subclasses. TemporalThing describes temporal related situations, events, and actions. MathematicalThing or InformationBearingThing defines units, coordinate systems, matrices, vectors, and other statistical facts. ObjectProperties, which can not be seen in this Figure, connect classes with objects or attributes in the rdf_triple(S, P, O) format.

3.3 Datasets

Historically, scientific datasets were by-products of research projects and were sometimes not published together with the analyzed results [89]. In recent years with the rise of

machine learning, datasets have become more important. Because algorithms learn features and properties from a dataset and the task is to make predictions on new or unknown data, the quantity and quality of a dataset is important for the algorithm's performance. In computer vision, classic data sets are the MNIST handwritten digits database [90], CIFAR10/100 [91], Caltech 101 [92], Caltech 256 [93], Pascal VOC [94], and ImageNet [95]. Because deep learning requires very large amounts of data with labels, the ImageNet dataset has received the highest attention in deep learning in recent years.

3.3.1 ImageNet dataset

The dataset ImageNet [95] is one of the largest and well known datasets available to the machine learning community. It consists of over 15 million high resolution images in 22,000 categories which are labeled by humans with Amazon's Mechanical Turk tool. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual competition in object classification and detection, which started in 2010. It uses a subset of ImageNet with about 1000 images in each of 1000 classes, with a total of 1.2 million images. The Figure 3.7 shows an overview of the dataset's 1000 classes.

Figure 3.7: ImageNet. This collage shows one picture for each of the 1000 classes. Source: [96]

The objects in ImageNet are centered in the images with variable backgrounds and settings. The benchmarks are compared with two error rates: top-1 and top-5, where for the top-5 error rate the correct label is not among the five most probable labels. In this work, the GURLS classifiers are trained on the extracted features from the CNN model provided in the standard Caffe version [52, 82] which is trained on the ImageNet dataset by Jeff Donahue from BVLC. The network BVLC Reference ICS-CaffeNet is based on the network architecture proposed in [6] and described in the previous Section.

3.3.2 iCubWorld28 dataset

The iCubWorld28 dataset consists of 7 distinct categories which are divided into 28 classes. Figure 3.8 lists the names of the 7 categories and shows the differences of four objects per category. More classes or different objects per category can make the computer vision algorithms more robust to detect the correct object category. The dataset consists of typical everyday objects that can be used in a household or kitchen environment for eating, drinking, and cleaning. It is the intention of the creators of the iCubWorld dataset series [97] to create a dataset that reflects the iCub's daily visual experience. About 220 images are acquired in each train and test set for one object per acquisition period. The acquisition period was repeated for four consecutive days, resulting in a dataset that contains 25831 training and 24884 testing images, separated in 4 days (Day 1 through Day 4).

Figure 3.8: iCubWorld28 example images. One category consists of four classes, i.e. 'Plate1', 'Plate2', 'Plate3', 'Plate4' are four classes and they belong to the same category Plate. This collage shows one picture for each of the 28 classes and 7 object categories. Source: [52]

During the acquisition of the dataset, the robot tracks the shown objects which leads to images that are centered in the dataset. The supervisor rotates the objects randomly in front of the robot to collect a large variety of different views of the object. This acquisition setup means that the hand of the human supervisor is always visible in the images, which represents a big difference between the iCubWorld28 and ImageNet datasets, since the

iCubWorld28 dataset			
Object	Material	Shape	Affordance
Laundry Detergent	Plastic	Cylinder	Cleaning
Plate	Ceramic	Cylinder	Eating
Dishwashing Detergent	Plastic	Cylinder	Cleaning
Sponge	Textile	Cuboid	Cleaning
Cup	Ceramic	Cylinder	Drinking
Soap	Textile	Cuboid	Cleaning
Sprayer	Plastic	Cylinder	Cleaning

Table 3.1: Overview of iCubWorld28 objects with assigned labels. In total, there are 7 objects, 3 materials, 3 shapes, and 3 affordances. The models are not trained for the attribute color. The dataset has no class 'Background', which means that no attribute can have the label 'None'.

ImageNet dataset does not contain hands in the images. The creators of the iCubWorld28 dataset provided the images with object labels. In the beginning of our experiments, we use this dataset to train and test our models on object classes and contextual information. Therefore, we additionally labeled the iCubWorld28 dataset for two attributes and affordance. Table 3.1 shows the chosen labels for 'Object', 'Material', 'Shape', and 'Affordance'. Pasquale et. al did not consider contextual information in the iCubWorld28 dataset in their experiments [52].

3.3.3 TUM-ICS dataset

In order to learn new objects with contextual information based on detected attributes, it is important to have a large dataset with objects that have different labels. In this section, we introduce our collected dataset TUM-ICS, which was collected with the iCub cameras in our experiments and contains 16,500 training images, 8,250 validation images and 8,250 testing images. These images were randomly split into training, validation, and test sets from a total amount of 300,000 collected images. The images were either collected with the iCub's 320×240 or high resolution 640×320 cameras. In the high resolution case, the pictures were resized to 320×240 before the whole dataset was cropped to 240×240. The 16,500 training images are created by 500 images per setting × 3 settings × 11 objects: Each object class contains 500 training images per setting or 1,500 total per object. If one object has multiple objects in its category, i.e. 'Plate' consists of 'Plate white', 'Plate red', 'Plate orange', 'Plate green', the 1,500 images per object (e.g. 'Plate') consist of a random number of images of the white, red, orange, or green 'Plate' class each. Following the methodology of the creators of iCubWorld28, we held different everyday objects in front of the iCub's cameras and recorded the pictures with the yarpdatadumper module provided by YARP. In contrast to iCubWorld28, the objects are not tracked by the robot during

the acquisition period. Therefore, the items are not centered in the images and located in all areas of the cropped images, as they are randomly 2D and 3D rotated and translated by the supervisor. Table 3.2 shows the full list of 23 object classes in the dataset with the assigned labels. If multiple object classes exist in one category, the models trained on object categories do not distinguish between object classes. For example, the machine learning algorithms are trained on the object category 'Apple' for both classes 'Apple green' and 'Apple red'. In one object category, the difference between two objects can be the color, for example a yellow or green 'Cleaning Cloth', or the material (e.g. 'Plate' made of 'Plastic' or 'Ceramic'). As the attribute labels shown in Table 3.2 are a very important prerequisite for our experiments, this table is shown in this section. The experimental results are dependent on our system design, which includes the choice of the attribute labels.

TUM-ICS dataset				
Object	Material	Shape	Color	Affordance
Apple	Organic	Sphere	Green Red	Eating
Banana	Organic	Cylinder	Yellow	Eating
Cleaning Cloth	Textile	Cuboid	Blue Green Yellow	Cleaning
Cup	Ceramic	Cylinder	Black Blue White	Drinking
Dishwashing Detergent	Plastic	Cylinder	Blue Green White	Cleaning
Laptop	Plastic	Cuboid	Grey	Working
Orange	Organic	Sphere	Orange	Eating
Plate	Plastic Plastic Plastic Ceramic	Cylinder	Green Orange Red White	Eating
Smartphone	Plastic	Cuboid	Black	Working
Sponge	Textile	Cuboid	Brown Green Yellow	Cleaning
Background	none	none	none	none

Table 3.2: Overview of all objects with assigned labels. In total, there are 11 objects, 6 materials, 4 shapes, 10 colors, and 5 affordances in the dataset.

We recorded objects in three experimental setups with three different backgrounds: 1) white background, 2) table, and 3) table with white background. The white background is used to make sure that the robot can find and extract the important features in an object. We realized in the table setting that it is easier to conduct our experiments with white background as the lab can look different everyday because of ongoing research activities and experiments in the background. Figure 4.8 displays the three different settings and shows examples of the objects per class. The recordings for the first setting were taken on consecutive days in very different daytime and light settings, as can be seen in the images in the upper part of the figure. The second setting was obtained on two different days with no notable change in light conditions, which can be seen in the similarity of the 'Background' class. The images in the third setting were taken on a single day. In contrast to the iCubWorld28 dataset, we included a 'Background' class for each setting which improves the total object recognition performance of our CNN and lets the robot identify that no object is in front of it during the experiments.

During the acquisition period, all images that were acquired in the first setting contain the supervisor's hand in the image. Because a table was used for the second and third setting, objects were both placed on the table without a hand in the image and were rotated with a visible hand in the images. Figure 4.8 illustrates these differences: hands are visible in all pictures in the upper part except for the 'Background' class, and are sometimes missing in the middle and bottom part of the images. It is notable that the black table in the second and third setting makes it difficult to see very dark objects in the images. Therefore, some objects like 'Smartphone' or 'Cup black' have been lifted and held in front of the white background.

3.4 OWL Ontology

Similar to the original KnowRob ontology, Figure 3.10 shows our defined ontology in the OWL format. In our system, there is no necessity to let the robot know and reason about the entire KnowRob ontology. Therefore, the ontology only contains the prior knowledge of the robot about our created TUM-ICS dataset. The node 'Objects' contains all 11 objects that are defined in the dataset. Following the same methodology as in the KnowRob base ontology, we used subclasses such as 'Container' or 'FoodOrDink' to summarize certain object groups and obtain the same categorization as KnowRob, if possible. Subclasses can have specific attributes, e.g. all classes that are subclasses of 'CleaningTool' have the affordance 'Cleaning'. The attributes material, shape, color, and affordance are also shown in the ontology. The attribute 'Labels' has two classes (True, Predicted) to differentiate whether an object is assigned the true labels (true), or predicted labels (predicted) if the output of the machine learning classifiers have detected different labels for an object, which are not asserted for this object in the knowledge base. In the latter case, our system is able to learn a known or unknown object with both true and predicted labels. The idea

Figure 3.9: ICS DATASET. This collage shows one picture for each of the 11 classes. The dataset was obtained in three different settings: 1) white background, 2) table, and 3) table with white background. The images shown in the first setting are taken with low-resolution cameras, images in the second and third setting are acquired with high resolution cameras. This is done because better lighting conditions were required for the black table. In the first setting, the objects are held closer to the cameras than in the second and third settings.

is that it still needs to be possible to correctly recognize an object although the predicted labels are not correct. The system is designed such that it can still find the correct object

even when incorrect labels are obtained from the classifier.

	Predicted Labels				
	Object	Material	Shape	Color	Affordance
True	Banana	Organic	Cylinder	Yellow	Eating
Predicted	Banana	Organic	Cylinder	Green	Cleaning

Table 3.3: The difference between true and predicted labels. The true labels are set during definition of the object, either as initially declared and shown in Table 3.2, or as specified in written form while a new object is being learnt. The predicted labels are the outputs of the machine learning classifiers.

Table 3.3 illustrates an example. Consider an object has the set of true labels: 'Banana', 'Organic', 'Cylinder', 'Yellow', 'Eating'. However, the classifiers recognize it with different labels: 'Banana', 'Organic', 'Cylinder', 'Green', 'Cleaning'. Then, our system is able to detect the 'Banana' because the system has been taught the object with both true and predicted labels before. To give the true labels a higher importance, the Prolog search query gives higher priority to the objects with true labels than the detected and finds the result first. Not listed in Figure 3.10 are object properties. We have created five object properties, one for each attribute or affordance and one for 'Labels'. The object property helps to assign the correct attribute to an object. For example, if we assume that *data* represents the defined namespace of our dataset, the query

$$rdf\_assert(data:Cup,\ data:hasMaterialType,\ data:Ceramic) \tag{3.1}$$

asserts new knowledge into KnowRob. The object property 'hasMaterialType' assigns the material attribute 'Ceramic' to the object 'Cup'.
In our system, the four most important Prolog predicates are 1)'setDefinitions', 2) 'learnNewObject', 3) 'checkObject', and 4) 'checkObject_all'. The predicates are called from the C++ program we use in our final experiments via the JSON Prolog interface. 1) The predicate 'setDefinitions' is called in the beginning of the C++ program which initializes the objects in our dataset with the assigned attributes. After this predicate is called, knowledge reasoning can be performed.
2) The predicate 'learnNewObject' is very important in our system, as it enables to learn new objects on-line. When querying Prolog, all labels for 'Object', 'Material', 'Shape', 'Color', 'Affordance', and 'TrueOrPredictedLabel' need to be provided. This predicate is called when the robot learns a new object with the corresponding labels provided by the human supervisor. Algorithm 1 presents this algorithm in detail.
3) 'checkObject' is used to retrieve the object from the knowledge base with two object properties and two true or predicted labels. For example, the two attributes with the highest probabilities are queried. Alternatively, the labels with the maximum and minimum probabilities can be queried as well. The code for this predicate is almost identical to checkObject_all', with the exception that two attributes from material, shape, color, and affordance must contain an empty string.

4) The predicate 'checkObject_all' requires all attributes to be provided as input to this query. When this predicate is called, it asks for the object that matches all four attribute labels in addition to the 'True' or 'Predicted' label flag. This algorithm is summarized in Algorithm 2.

Algorithm 1 learnNewObject

Input: ?A, +Object, +Material, +Shape, +Color, +Affordance, +TrueOrPredictedLabel
Output: Learned object A

(atom_concat(data, Object, B)),
(atom_concat(data, Material, C)),
(atom_concat(data, Shape, D)),
(atom_concat(data, Color, E)),
(atom_concat(data, Affordance, F)),
(atom_concat(data, TrueOrPredictedLabel, G)),

owl_subclass_of(A, B), rdf_assert(A, rdf:type, owl:'Class'),

rdf_assert(A, data:'hasMaterialType', C),
rdf_assert(A, data:'hasShapeType', D),
rdf_assert(A, data:'hasColor', E),
rdf_assert(A, data:'isUsedFor', F),
rdf_assert(A, data:'isTrueOrPredicted', G).

Return: A

Algorithm 2 checkObject_all

Input: ?A, +Material, +Shape, +Color, +Affordance, +TrueOrPredictedLabel
Output: Inferred object(s) A

```
ObjectProperty1='hasMaterialType';
ObjectProperty2='hasShapeType';
ObjectProperty3='hasColorType';
ObjectProperty4='isUsedFor';
ObjectProperty5='isTrueOrPredicted';

(atom_concat(data, Material, B)),
(atom_concat(data, Shape, C)),
(atom_concat(data, Color, D)),
(atom_concat(data, Affordance, E)),
(atom_concat(data, TrueOrPredictedLabel, F)),

(atom_concat(data, ObjectProperty1, B_ObjP)),
(atom_concat(data, ObjectProperty2, C_ObjP)),
(atom_concat(data, ObjectProperty3, D_ObjP)),
(atom_concat(data, ObjectProperty4, E_ObjP)),
(atom_concat(data, ObjectProperty5, F_ObjP)),

owl_has(A, B_ObjP, B),
owl_has(A, C_ObjP, C),
owl_has(A, D_ObjP, D),
owl_has(A, E_ObjP, E),
owl_has(A, F_ObjP, F),
atom_concat(data,Obj,A).
```

Return: A

Figure 3.10: OWL Ontology. This tree illustrates the prior knowledge our robot iCub has at the beginning of our experiments. Object properties describe the connections between objects, attributes, and affordance. After learning a new object, the knowledge base is expanded by the new object with its corresponding attributes.

3.5 Enhanced deep network

Our proposed system combines the machine learning algorithms and the knowledge base, which we call enhanced deep network. As shown in Figure 3.11, we use five deep convolutional neural networks in parallel that are trained end-to-end. For querying the database, the predicted labels are sent to the knowledge and reasoning system via YARP. Figure 3.12 illustrates the system design for the version with GURLS: we only use one

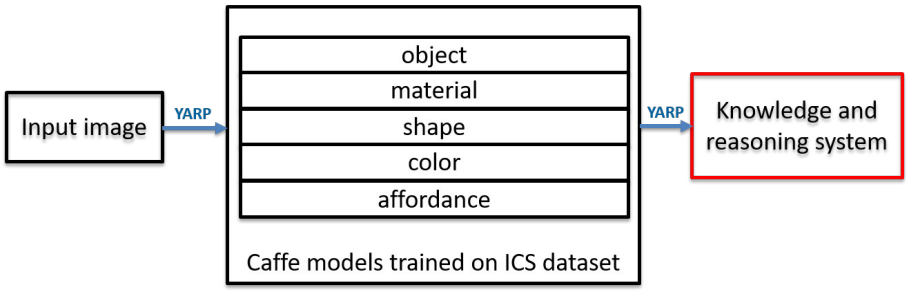


Figure 3.11: Caffe and knowledge reasoning.

deep neural network, which is trained on ImageNet, and five GURLS classifiers which make predictions on the extracted features from the fully connected layer FC7. An

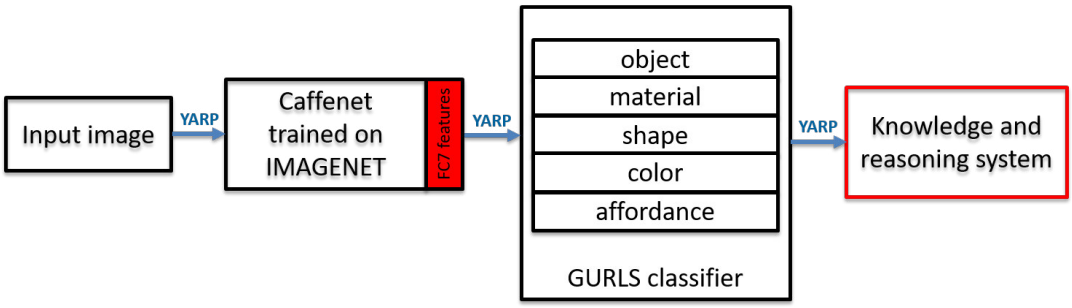


Figure 3.12: GURLS and knowledge reasoning. The GURLS classifiers are trained on extracted 4096-dimensional feature vectors.

overview of the interaction with the robot during learning is shown in Figure 3.13. In this system, it is necessary to supervise the robot learning with a human teacher. The supervisor is responsible to either approve the robot guesses or to give input to the robot by correcting certain predicted labels. In the following, we describe the workflow of our proposed system in detail.

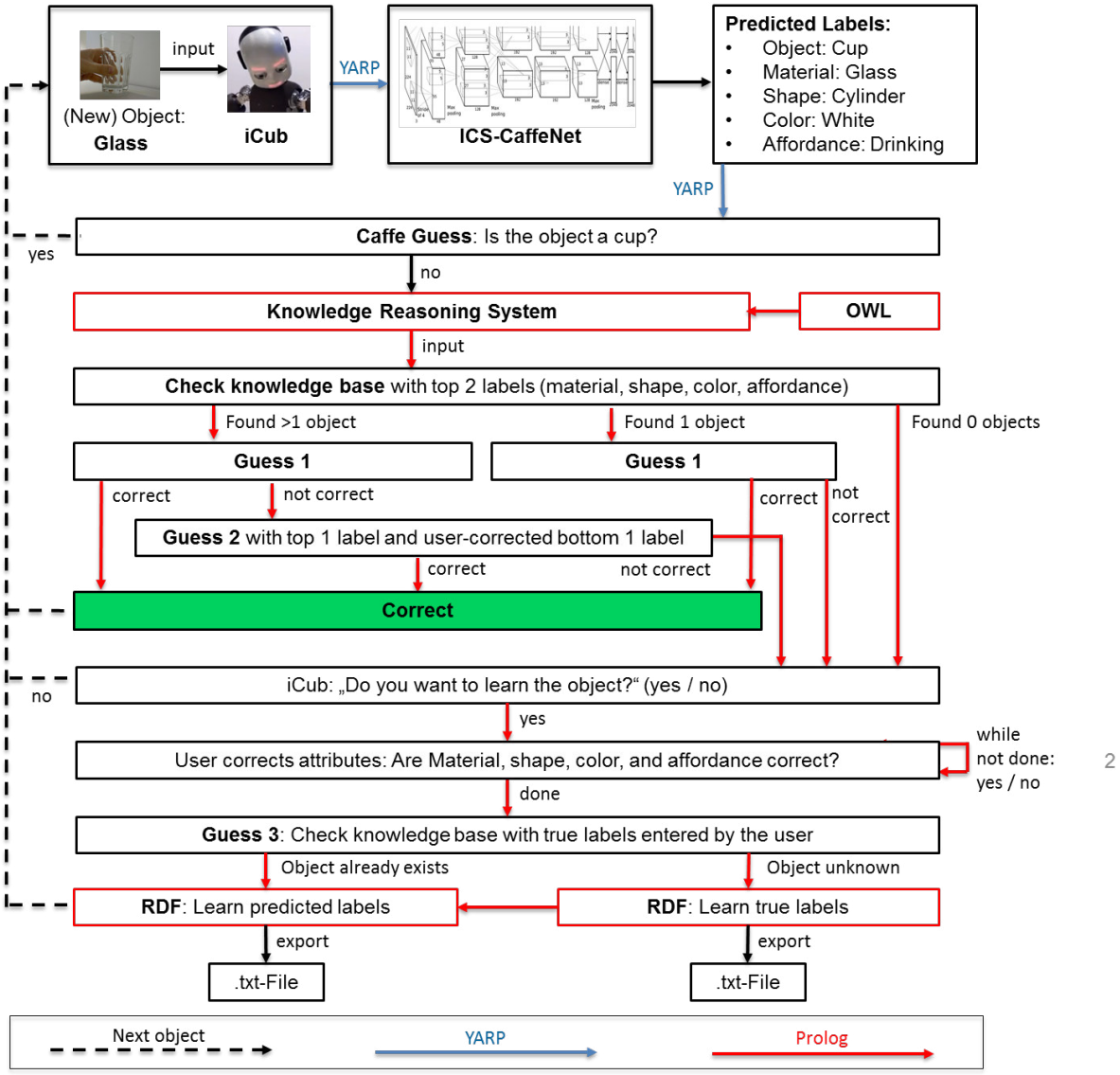


Figure 3.13: System overview from a UX/UI perspective. This example shows the version with ICS-CaffeNet. We have also developed the system version with the ImageNet-trained model and GURLS classifiers. The shown object to the robot can either be known or unknown. iCub makes guesses based on the predicted labels by ICS-CaffeNet and queries to the knowledge base. When the robot guesses the correct object or a new object has been learned, the program restarts from the beginning.

An object is shown to the robot which is classified by our machine learning algorithms. Five labels for object, material, shape, color, and affordance are predicted. Based on this information, the robot makes a first guess based on the detected object. If the object recognition algorithm performs well, this guess is expected to predict the correct object if the shown object is known to the robot. When the machine learning classifiers fail to

predict the known object or are not trained on this object category, our proposed system can make further guesses based on contextual information, i.e. the object's attributes and affordances. In this case, the knowledge processing system reasons about the objects and properties in the OWL ontology. It checks the knowledge base for objects that share the same top two labels from material, shape, color, and affordance, based on the highest probabilities. If no object is in the knowledge base with the top two labels, the object is learned directly by the robot. If one object is found, the robot guesses the object - if it is correct, the robot can continue with the next one. If it is false, the program goes to the starting point of the object learning system. If more than one object is found in the knowledge base, which is very likely after many new objects have been learned, the robot makes a guess based on the first inferred object. This guess needs to be different than the Caffe-guess, otherwise the second inferred object is guessed. Similarly with the other options, a correct guess results in the end of this iteration and a new object can be shown. If the first system guess is not correct, a second guess is made based on the attribute with the highest probability and the attribute with the lowest probability. Before the second guess, Active Learning [98] is applied by asking the supervisor if the predicted label with the lowest probability is correct or not. This gives the user the chance to manually correct the predicted label with the lowest probability. Active Learning produces a very high likelihood that the resulting two labels are correct and thus finds the correct object in the knowledge base. If multiple answers from the queried knowledge base exist, the guess is made on the first inferred answer, requiring that this guess is different than the Caffe Guess and Guess 1. If this is not the case, Guess 2 is based on the second or third inferred answer from the knowledge base. When Guess 2 is incorrect, which happens especially in cases when the tested object is unknown, the program jumps to the learning part of the proposed system: the user is asked to manually type the name and correct the labels of the 1) object, 2) material, 3) shape, 4) color, and 5) affordance. The system makes a third guess to the knowledge base if an object exists with the labels of the attributes of number 2)-5). If an object is found, which means that the object with true labels, that are entered by the user into the system, already exists. In this case, the object is learned with the original predicted labels that are detected by ICS-CaffeNet. We do this because the object is known in the knowledge base with the true labels but not with the detected labels. If Prolog finds multiple objects after Guess 3, the guess is based on the first object which is not identical with Caffe Guess, Guess 1, and Guess 2. Learning objects with the predicted labels enables the robot to identify objects although they possess other labels in the truth. If no known objects are found, the system learns first the true labels, which are entered by the user, and afterwards the predicted labels. This is done to ensure that objects with the correct labels are given higher priorities than objects with predicted labels. Otherwise, the robot would incorrectly reason to recognize an object based on the predicted labels, although other objects exist which are labeled with the correct attributes. The true and predicted labels of the objects are learned via RDF. For accessing the learned knowledge about the objects in the next session, these queries are exported in two separate text files and are loaded into the knowledge base when the system is launched. Table 4.15 illustrates how new objects are learned with the enhanced deep network over several trials.

Learning Overview: ICS-CaffeNet and Enhanced deep network					
Method	Example Object	Trial 1	Trial 2	Trial 3	No. Trials
ICS-CaffeNet	Glass	0	-	-	-
	Watermelon	0	-	-	-
Enhanced deep network	Glass	0	1	-	2
	Watermelon	0	0	1	3

Table 3.4: This Table briefly explains how objects are learned with the enhanced deep network. The system predicts the labels for object, material, shape, color, and affordance. If the object is unknown, the deep learning network is not capable to recognize the correct class. The system always 'fails' to recognize a new object in the first 'trial', as the first column in this table shows. When the object is not recognized, we use the proposed learning mechanism to learn both the predicted and user-corrected true labels with Prolog queries. If the predictions are inaccurate, the predicted labels are different than the true labels. An object is learned until it is recognized by the system in either the first or second system guess. Two mock-up examples show how this system is trained. When the object is detected, the cell is marked with 1, otherwise with 0. Glass is learned within two trials and Watermelon with three trials. Immediately after the objects are learned, new objects are trained.

Deep learning networks are not capable to learn unknown objects.

Chapter 4

Experiments and Results

This section is structured in two main parts. The first part compares the object recognition performances of deep neural networks with the GURLS machine learning library on two different datasets. Additionally, the obtained results are described and evaluated in more detail to explain why the machine learning methods succeed or fail in these experiments. The second part introduces and analyzes the proposed enhanced deep network, which combines the output of machine learning algorithms with semantic reasoning to improve the object recognition performance for both known and unknown objects. All experiments have been done using both deep learning networks and GURLS classifiers. For simplicity, we say a deep neural network is trained, although it would be more correctly to say that the models were finetuned on the dataset, as all layers in the network except one were kept frozen during training. In this thesis, the finetuned deep learning networks which are trained end-to-end on the evaluated datasets are referred as 'ICS-CaffeNet'. In contrast, GURLS classifiers are trained on extracted FC7 features of a deep learning network which is pre-trained on the ImageNet dataset.

4.1 Object Recognition Performance

The object recognition performances are obtained on two different datasets, namely the iCubWorld28 dataset published by Natale et. al [97] and the TUM-ICS dataset, which was collected over the period of this thesis. The Caffe framework is used to train the deep convolutional networks for the object, attributes and affordances in the images. For each dataset, the models are compared with GURLS classifiers, which have been trained on extracted FC7 features of an ImageNet-trained convolutional neural network.

4.1.1 Results on the iCubWorld28 dataset

Before we obtained the TUM-ICS dataset, we trained, tested and evaluated our models first on the iCubWorld28 dataset. This subsection provides the performance results for 28 classes, 7 classes, two attributes and affordance. Table 4.1 shows the object classes, categories, and labels for the iCubWorld28 dataset.

iCubWorld28 dataset				
Class	Category	Material	Shape	Affordance
LaundryDet1 LaundryDet2 LaundryDet3 LaundryDet4	Laundry Detergent	Plastic	Cylinder	Cleaning
Plate1 ... Plate4	Plate	Ceramic	Cylinder	Eating
DishwashingDet1 ... D.4	Dishwashing Detergent	Plastic	Cylinder	Cleaning
Sponge1 ... Sponge4	Sponge	Textile	Cuboid	Cleaning
Cup1 ... Cup4	Cup	Ceramic	Cylinder	Drinking
Soap1 ... Soap4	Soap	Textile	Cuboid	Cleaning
Sprayer1 ... Sprayer4	Sprayer	Plastic	Cylinder	Cleaning

Table 4.1: Overview of the iCubWorld28 datasets with all classes, categories and defined labels. The attribute 'Color' is not considered for this dataset. Four object classes can be summarized into one object category, in which all classes have the same attributes. The category 'Laundry Detergent' illustrates this in more detail. The class names of the other objects are listed in abbreviated form, but also contain four objects per category.

4.1.1.1 Object (28 classes)

Training our models for 28 classes means that the models need to distinguish between categories such as 'Plate' and 'Soap', for example, but also between the classes 'Plate1' and 'Plate2', for example. It is a much more challenging task to differentiate between classes than categories. In this case, more fine-grained features in the images need to be considered by the models. The differences between classes in categories can be different colors (e.g. 'Plate1-4') or different patterns ('Cup1-4'). Table 4.2 shows that the ICS-CaffeNet model which is directly trained on iCubWorld28 performs better than GURLS on the test set.

iCubWorld: 28 classes		
Method	ICS-CaffeNet	GURLS
Accuracy	**0.88**	0.8

Table 4.2: ICS-CaffeNet and GURLS on iCubWorld28. ICS-CaffeNet performed better.

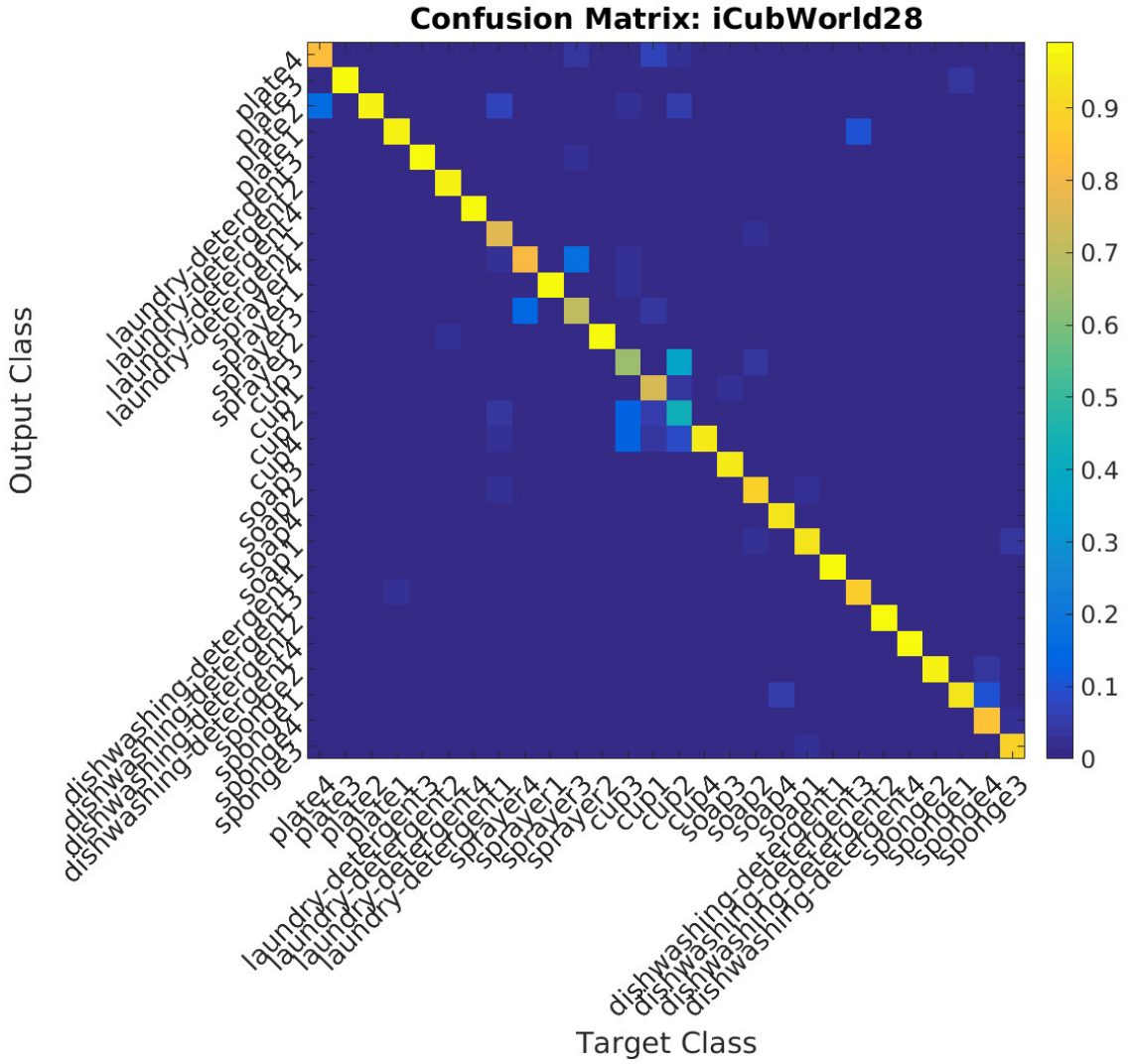


Figure 4.1: Confusion Matrix of iCubWorld28. Misclassifications only occur within the same categories, especially between different cups, sprayers and plates. This means, for example, 'Cup1' is recognized instead of 'Cup2'. Based on this finding, reducing the number of classes in this dataset from 28 to 7 classes improves the performances of the tested algorithms.

In the confusion matrix shown in Figure 4.1, we can see that the model misclassified some classes, e.g. it recognized 'Cup 1' instead of 'Cup 2'. This means that the classification errors are only within the same object categories, such as 'Cup'. In order to avoid this, we train our models on a smaller number of classes. In the following, we test if the performance of our models increases when the we train the models on the 7 categories instead of 28 classes. In the next subsection, we check if this assumption is true.

4.1.1.2 Object (7 classes)

In this experiment, we propose to reduce the amount of classes from 28 to 7 by combining similar objects such as 'Plate 1' and 'Plate 2' into the category 'Plate'. The goal is to analyze if this change improves the object recognition accuracy, as the object category and not the object class is important for our experiments. Following the same methodology as described for 28 classes, we make a performance comparison.

iCubWorld: 7 classes		
Method	ICS-CaffeNet	GURLS
Accuracy (28 classes)	0.88	0.8
Accuracy (7 classes)	**0.98**	0.91

Table 4.3: iCubWorld: 7 classes. The deep neural network achieves the highest performances.

Table 4.3 compares the results: the experiment confirms our assumption that reducing the number of classes leads to a higher overall performance - for both ICS-CaffeNet and our GURLS classifier. At 98 percent accuracy, ICS-CaffeNet reaches nearly optimal performance results for 7 classes. The GURLS performance is slightly lower, but also improved by 10 percent.
In the remaining part of this thesis, we take only the 7 classes into consideration instead of the original 28 classes of iCubWorld28. Therefore, when results are compared between 'iCubWorld' and our dataset in this thesis, the classes within iCubWorld are treated as 'Plate', 'Laundry Detergent', 'Sprayer', 'Cup', 'Soap', 'Dishwashing Detergent', and 'Sponge'.

4.1.1.3 Comparison with Pasquale et. al

Compared to the literature [52], our trained models achieve higher results than the reported results from IIT, as shown in Table 4.4. Pasquale et. al trained their predictors on all days (Day 1-4). We also trained both ICS-CaffeNet and GURLS on all four days, but additionally included manually cropped images into the training set. These images are provided in the standard iCubWorld28 dataset by Pasquale et. al, and are more closely

cropped around the shown object. In contrast to their work, we used bGURLS for off-line training. The deep learning network achieves the best results in this comparison.

iCubWorld28 dataset: Performance Comparison		
Method	28 classes	7 classes
GURLS (Pasquale et. al)	0.70	-
GURLS	0.80	0.91
ICS-CaffeNet	**0.88**	**0.98**

Table 4.4: Performance comparison with Pasquale et. al [52] from IIT. We report higher results than IIT. The end-to-end trained ICS-CaffeNet model achieved the highest performance, for both 7 and 28 classes.

Table 4.5 further compares how many objects our robot is able to recognize. The data illustrates that iCub can recognize more objects with our trained models. For 28 objects, the number of recognized objects with 98% confidence is similar to the number of recognized objects from Pasquale et. al. However, with a lower confidence of 90% or 80% in the case of [52], the recognition performance is higher with our models. There is only one object which we can not recognize among 28 objects above 50% confidence with our models, which is 'Cup3'. It is a white cup with a small black and red 'Segafredo' logo (see Figure 3.8), and often misclassified as 'Cup2', which is a white cup with a spotted pattern in black and red colors. In comparison, the data for 7 objects shows that we can recognize 4 of 7 objects above 98% and all objects above 90% accuracy, which is a significant improvement compared with 28 classes.

	Confidence					
Method	98%	90%	80%	70%	50%	# Total
Pasquale et.al	2	4	6	7	14	28
GURLS 28	1	10	16	20	27	28
ICS-CaffeNet 28	**6**	**18**	**22**	**25**	**27**	**28**
GURLS 7	**4**	**7**	**7**	**7**	**7**	**7**
ICS-CaffeNet 7	**4**	**7**	**7**	**7**	**7**	**7**

Table 4.5: The maximum number for objects that iCub is able to recognize on the iCub-World28 dataset is 4 objects using Pasquale et. al with 90% of confidence. Our method obtains a better accuracy of identifying 18 objects with the same 90% confidence. Notice, that when 7 classes are used instead of 28, our method is able to recognize all 7 with 90% confidence. The performance is compared with the Table from [52].

iCubWorld: Attributes and Affordances		
Method	ICS-CaffeNet (%)	GURLS (%)
Material	**90**	**90**
Shape	**94**	90
Affordance	**95**	90

Table 4.6: iCubWorld: Attributes and Affordances. Both methods obtain very high results.

4.1.1.4 Attributes and Affordances

In the previous three subsections, we considered the training and testing of our models based on the object recognition performance, i.e. the accuracy our models achieved when they were specifically trained for recognizing the correct objects. As the goal of this work is to learn unknown objects, the current models trained on known objects can never correctly recognize an unknown object. For example, if we show an orange to the robot, this would not be able to recognize this new object since it was not previously trained. Therefore, we proposed to include contextual information in our system. This subsection provides the results of our models which are trained to recognize contextual information, which Pasquale et. al did not consider in their experiments. The attributes material, shape, and affordance are evaluated, since they are the most significant extracted from images as stated in the literature []. We did not analyze the attribute color in the iCubWorld28 dataset.

As shown in Table 4.6, we were able to train ICS-CaffeNet and GURLS to achieve at least 90 percent on the test set. The accuracy of ICS-CaffeNet for shape and affordance are the highest ones with 94 percent and 95 percent, respectively.

4.1.1.5 Need for a new dataset

Correctly recognizing known objects is a very critical and important prerequisite for learning new objects. When our robot iCub looks at an object, it needs to know if the object is new or if it is already included in the knowledge base. If the robot falsely assumes it does not know the object, the robot will start to learn the object although it was known before. In this case, the supervisor spends a lot of time interacting with the robotic teaching system, especially when the robot fails to recognize the correct attributes due to the incorrect deep network model.

For this reason, we tested the trained models on images that were made within our lab environment (see Figure 4.2 for an example). We tested the same 7 objects and performed experiments to check if the robot can detect the correct object. Table 4.7 shows the results: the accuracy is significantly lower than before. The performance values for ICS-CaffeNet lie between 14 and 47 percents, compared to 90 to 98 percents before. This means the accuracy is between 50 and 70 percentage points lower when having different test data for

iCubWorld: Object, Attributes and Affordances		
Method	ICS-CaffeNet (%)	GURLS (%)
Object	29	**41**
Material	47	**53**
Shape	29	**71**
Affordance	14	**61**

Table 4.7: iCubWorld28: We achieved low performance with an overfitted iCubWorld model on test images from another setting. Figure 4.2 shows two image examples. GURLS performed better due to the overfitted CNN.

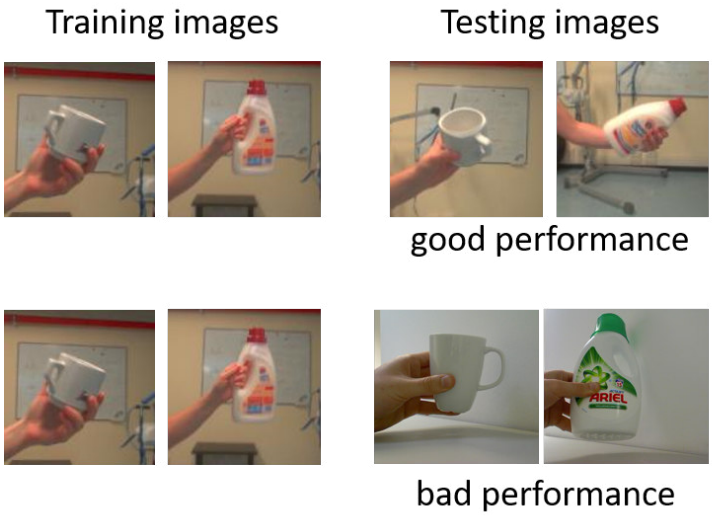
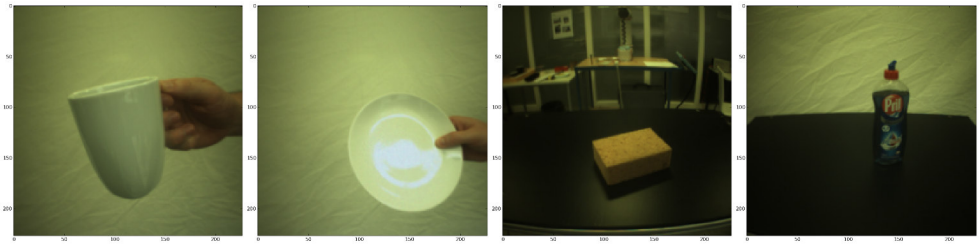


Figure 4.2: An example how low accuracies were obtained in this section. Top: The performance is high for an iCubWorld model and an iCubWorld dataset. Bottom: The accuracy is low for the same model tested on images from our lab environment.

ICS-CaffeNet. The performance of GURLS is between 41 and 71 percent, which is between 19 and 49 percentage points lower. The reason is that the GURLS classifier is trained on extracted features from a deep neural network that has been trained on over 1 million images. This makes the feature vector of an object highly descriptive. The results show that we need to improve the object recognition performance for known objects. Otherwise, the robot would too often struggle to correctly recognize an object when it needs to learn a new one. One approach to improve the object recognition performance is to test if the accuracy is higher on another dataset, which is explored in the next subsection.

The low accuracy data on the shown images were the initial reason why the TUM-ICS dataset was collected. In this thesis, the results shown in the later sections **??** prove that it was necessary to collect a new dataset. In fact, there are two reasons why the performance in this experiment was low:



(a) Cup (b) Plate (c) Sponge (d) Dishwashing Det

Figure 4.3: Testing four images from the TUM-ICS dataset with the corresponding mean image file on finetuned models from different datasets.

- The results shown in this subsection are produced with an overfitted model. This is the main reason why the performance is so low in this section for the deep neural networks. In section 4.1.2.6, we provide more detail how this problem was solved and describe our experiences with overfitting.
- Section 4.1.2.3 evaluates the performances of models trained and tested on two different datasets. Due to the performance comparison, this experiment shows that creating a new dataset is necessary.

The next section analyzes the performances on the TUM-ICS dataset.

Object	ImageNet model	iCubWorld28 model	TUM-ICS model
Cup (a)	Cup	Cup	Cup
Plate (b)	Toilet Seat	Plate	Plate
Sponge (c)	Studio Couch	Sponge	Sponge
Dishwashing Det. (d)	iPod	Laundry Detergent	Dishwashing Det.

Table 4.8: Predictions of models trained on different datasets on the four test images shown in Figure 4.3 from the TUM-ICS dataset.

4.1.2 Results on the TUM-ICS dataset

For our proposed system in this thesis, it is very important the robot can recognize the objects from our knowledge base. Obtaining a new dataset can be a work-intensive task as light and room conditions, the availability of objects and the robot's hard- and software need to be stable during the acquisition period. A new dataset has the following two main advantages: first, it gives the researchers control over the dataset. In this work, it means that we can add more objects with different attributes such as different materials or colors to the dataset. This is very important as we rely on contextual information to learn new

objects. The second advantage is that the new dataset is acquired in the same experimental setup as we do the final experiments on the proposed system. This means our models are optimized for this environment and can expect to obtain the maximum possible amount of performance in the final experiments.
For our main dataset, we conducted a series of experiments, starting with the same methodology as with iCubWorld28. This subsection is divided into several experimental parts:

- First we evaluate the object recognition performance of the dataset.
- Secondly, we analyze the performance of the trained networks on contextual information, i.e. the attributes material, shape, color, and affordance.
- Afterwards, we do a crosstesting experiment to test if the iCubWorld28 and TUM-ICS trained models can recognize known and unknown objects from the other datasets. Additionally, we compare these result with the performance of the trained models on their own datasets.
- In the next subsection, we provide a detailed feature analysis, which visualizes the relation between the 4096-dimensional feature vectors with different sets of labels.
- We use a prior trained network called GoogLeNet-CAM to highlight heatmaps in our images where convolutional nets look at to predict the class.
- An extra paragraph is committed to describe how we overfitted a deep learning network and showcase the differences in performance and in visualized layers.
- Finally, we test the trained networks on another set of images which contains images of both known and unknown objects. These images were obtained during testing the enhanced deep network, and are not included in the TUM-ICS dataset, i.e. they were not used to train the model.

4.1.2.1 Object (11 classes)

Table 4.9 compares the results on the TUM-ICS dataset for 11 classes. With an accuracy of 99 percent, both ICS-CaffeNet and GURLS achieve nearly optimal results on the validation dataset. Note that during training, the loss of ICS-CaffeNet is optimized on the validation set. On the test set, the performance is slightly lower for ICS-CaffeNet. The GURLS classifier also achieved a lower classification accuracy.

4.1.2.2 Attributes and Affordances

The accuracies of the algorithms in this subsection are more important for the performance of our final system than the accuracies obtained on classifying objects. Comparing the results in Table 4.10 for both validation and test set, the data shows the same pattern as with 11 objects. ICS-CaffeNet and GURLS achieved very high performances on the

TUM-ICS: 11 classes		
Method	ICS-CaffeNet (%)	GURLS (%)
11 classes - Validation set	**99**	99
11 classes - Test set	**84**	67

Table 4.9: The images in the validation set were used to compute the loss of the model during the optimization. The images from the test set are independent images used to quantify the performance of the model.

TUM-ICS: Attributes and Affordances				
	Methods			
	ICS-CaffeNet (%)		GURLS (%)	
Attributes	Val	Test	Val	Test
Material	99	**93**	99	62
Shape	99	**93**	99	62
Color	99	**68**	99	61
Affordance	99	**77**	99	63

Table 4.10: TUM-ICS: Attributes and Affordances. The performance on the validation set is nearly optimal. On the test set, ICS-CaffeNet performs better, especially for the attributes material and shape.

validation set with 99 percents for all attributes. On the test data, the values of 93 percent indicate that ICS-CaffeNet is able to generalize the correct prediction for the attributes material and shape. The performance is worse for color and affordance, at 68 and 77 percent, respectively. For GURLS, all attributes and affordances reach a performance of about 62 percent, showing that the deep neural networks are superior to the GURLS classifiers.

4.1.2.3 Crosstesting: iCubWorld and TUM-ICS models

As mentioned previously, a crosstesting experiment is performed with the deep neural networks that are finetuned on the iCubWorld28 and TUM-ICS datasets. Figure 4.4 summarizes how the experiment is performed with two models on two datasets in four tests. The goal is to evaluate if the models can recognize the correct objects, although the models were not trained on the respective datasets. In this experiment, about 1000 images from the iCubWorld28 dataset and roughly 1000 images from the TUM-ICS dataset are used (including unknown objects to the respective model). The images are evenly distributed across the acquisition days and object classes. In this experiment, we choose one model (iCubWorld or TUM-ICS) and one dataset (iCubWorld or TUM-ICS) including the dataset's corresponding mean image file. In all four resulting cases, both the '4 similar' and '3 different' objects are tested. The '4 similar' objects are contained in both datasets, which are 'Cup', 'Dishwashing Detergent', 'Plate' and 'Sponge'. The '3 different' objects

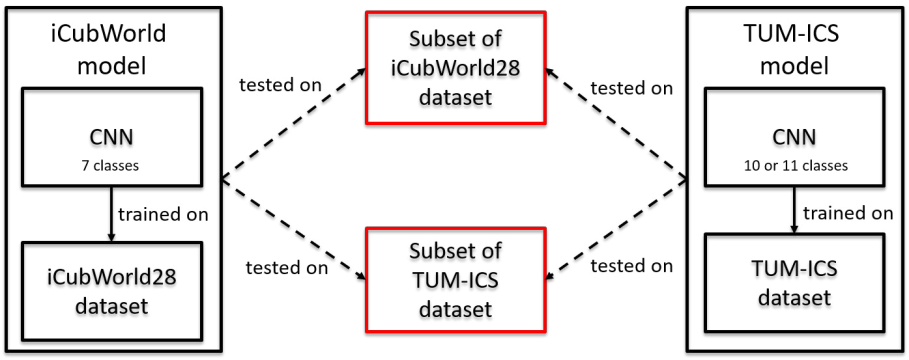


Figure 4.4: Overview of the performed crosstesting experiment. The iCubWorld model and TUM-ICS model are CaffeNets which are pretrained on ImageNet and finetuned on the respective datasets. The models are tested on both their own and opposite dataset. This experiment is done to evaluate how good the models perform on images from different experimental settings. Four tests are done in total. The TUM-ICS model can either be trained on 10 classes or 11 classes (including class 'Background'). The latter models are compared in Table 4.12.

Crosstesting Experiment (in %)								
	iCubWorld Dataset				TUM-ICS Dataset			
	iCubWorld model		TUM-ICS model		iCubWorld model		TUM-ICS model	
Attributes	4 similar	3 different	4 similar	3 different	4 similar	3 different	4 similar	3 different
Object	69	-	29	-	57	-	**85**	-
Material	74	86	33	26	64	7	**84**	**98**
Shape	84	84	59	58	87	6	**85**	**98**
Affordance	**91**	**94**	53	70	65	38	51	76

Table 4.11: Crosstesting: Object, material, shape, and affordance are tested in this experiment. Because we use two different datasets and two different models, there are 4 tests for which we perform the crosstesting experiment. The second row describes from which dataset the tested images and corresponding image mean file come from. The image mean files are important for classification because they are different for the datasets. The models which are tested on this dataset are found in 'the third row. The names of the models describe on which dataset these models were trained. The iCubWorld dataset consists of 7 classes, the TUM-ICS dataset has 11 classes (from which we only use 7, corresponding to the 4 similar and 3 different). The objects 'Cup', 'Plate', 'Dishwashing Detergent', and 'Sponge' are included in both datasets. The category "4 similar" contains the images from these objects of the corresponding dataset. "3 different" includes objects which are from the same dataset, but on which the tested models are not trained on. Therefore, it is impossible for a deep learning network to correctly recognize the object in the category "3 different".

are different for each dataset: the iCubWorld7-trained model is unfamiliar with 'Banana', 'Cleaning Cloth' and 'Laptop' from the TUM-ICS dataset. The model trained on TUM-ICS tests its performance on the '3 different' objects 'Laundry Detergent', 'Soap', and

'Sprayer'.
The overview of the comparison is presented in Table 4.11. As expected, both models perform well on their own dataset, reaching top 1-accuracies of about 70-90%. The TUM-ICS model performs better on its dataset compared to the iCubWorld model, except for the metric affordance: only three classes (i.e. 'Eating', 'Drinking', 'Cleaning') exist for affordance in the iCubWorld dataset, with 5 out of 7 objects are labeled as 'Cleaning'. In contrast, TUM-ICS contains five affordance classes ('Eating', 'Drinking', 'Cleaning', 'Working', 'None'). Surprisingly, the iCubWorld-trained model performs very well on the TUM-ICS dataset. For the object and material values on '4 similar', the achieved values are only about 10 percentage points lower than on the iCubWorld dataset, while shape remains about the same and affordance is 26 percentage points lower. However, for the '3 different' objects, the classification accuracy is significantly lower: the iCubWorld-trained model achieved a merely 7% on material and 38% on affordance, while the shape was classified well with 66%. The performance of the TUM-ICS model on the iCubWorld dataset is lower than the other combination on the known objects. This makes sense as the TUM-ICS model is trained on the class 'Background' which improves the recognition of the other classes due to the softmax classifier in the network's last layer. The performance of the TUM-ICS model on the unknown objects reaches a similar magnitude in accuracy: material is recognized with 26%, the correct shape with 58% and the affordance with 70%. The reason for this result is the variability of our collected TUM-ICS dataset. The advantage is the size: TUM-ICS contains more objects than iCubWorld which also has a higher variability in the labels set. Most objects in iCubWorld have a long, cylindrical shape that is made from 'Plastic' and can be used for 'Cleaning', for example the class 'Laundry Detergent'. In the TUM-ICS dataset, only one object fits this description, i.e. the class 'Dishwashing Detergent'. Inspired by a previous dataset from the original ICUBWORLD dataset series [97], some objects in our datsets are edible fruits or are different forms of cups and plates, which enrich the dataset with a variety of shapes, materials and colors.
We performed the crosstesting experiment because the iCubWorld model had previously scored low accuracies on the recorded set of images, as described in section 4.1.1.5. In conclusion of this experiment, it was necessary to obtain a new dataset. The data shows that the iCubWorld model reaches accuracies on our dataset in the range of 57, 64, 87, and 65% while our model achieved 85, 84, 85, and 51%, respectively. The iCubWorld model performs well, but for our specific research problem it is very benificial to obtain the maximum amount of possible performance. This is due to the fact that our research goal is to learn new objects by recognizing the contextual information of the questioned object - which is based on the attributes and affordances of the known objects located in our OWL ontology. The main advantage is that we can include new objects to the dataset that provide new attributes, classes or labels our model can learn. Thus, a larger dataset with a more unique set of labels means that some objects can be queried easier in the knowledge base. However, for objects that share the most common labels, it can be very difficult for the robot to guess the correct object since multiple objects fulfil the desired criteria, for example the shapes 'Cuboid' or 'Cylinder' and material 'Plastic'.
Table 4.12 provides an additional perspective in analyzing the results of the crosstesting

experiment. The prior shown results were performed with a TUM-ICS model trained on 11 classes, including the class 'Background'. Because the iCubWorld model was not trained on a background class which could improve the recognition accuracy for the other classes, we evaluate the results of the model trained on 10 classes. The data indicates that the model trained on 10 classes achieved a lower accuracy on objects on the TUM ICS dataset. However, it slightly improves the performance of the other model on material and shape by 1 percent. For affordance, it increased the performance of the model by 20-30%.
On the iCubWorld dataset, the opposite is true: the model trained on background has a higher performance for object, material, and shape for both known and unknown objects. For unknown objects, the accuracy for affordance with 70% is also higher than 60%. However, the results show that the performance of the TUM-ICS models are relatively low on the iCubWorld dataset compared to the TUM-ICS dataset. As the iCubWorld-models also achieve a lower performance on the TUM-ICS dataset compared to the TUM-ICS models, it can be seen that the new dataset was needed to achieve a higher overall recognition performance.

Crosstesting Experiment - Comparison TUM-ICS model trained on 10 and 11 classes (in %)								
	iCubWorld Dataset				TUM-ICS Dataset			
	TUM-ICS model				TUM-ICS model			
	10 classes		11 classes		10 classes		11 classes	
Attributes	4 similar	3 different	4 similar	3 different	4 similar	3 different	4 similar	3 different
Object	24	-	29	-	69	-	**85**	-
Material	31	30	33	26	**86**	97	84	**98**
Shape	56	48	59	58	**88**	97	85	**98**
Affordance	60	60	53	70	**85**	**95**	51	76

Table 4.12: Crosstesting experiment Add-On. This table compares the performances of the TUM-ICS models trained on 11 classes (including class 'Background') and 10 classes (without class 'Background"). The results shown for the model trained on 11 classes is the same as previously shown in Table 4.11.

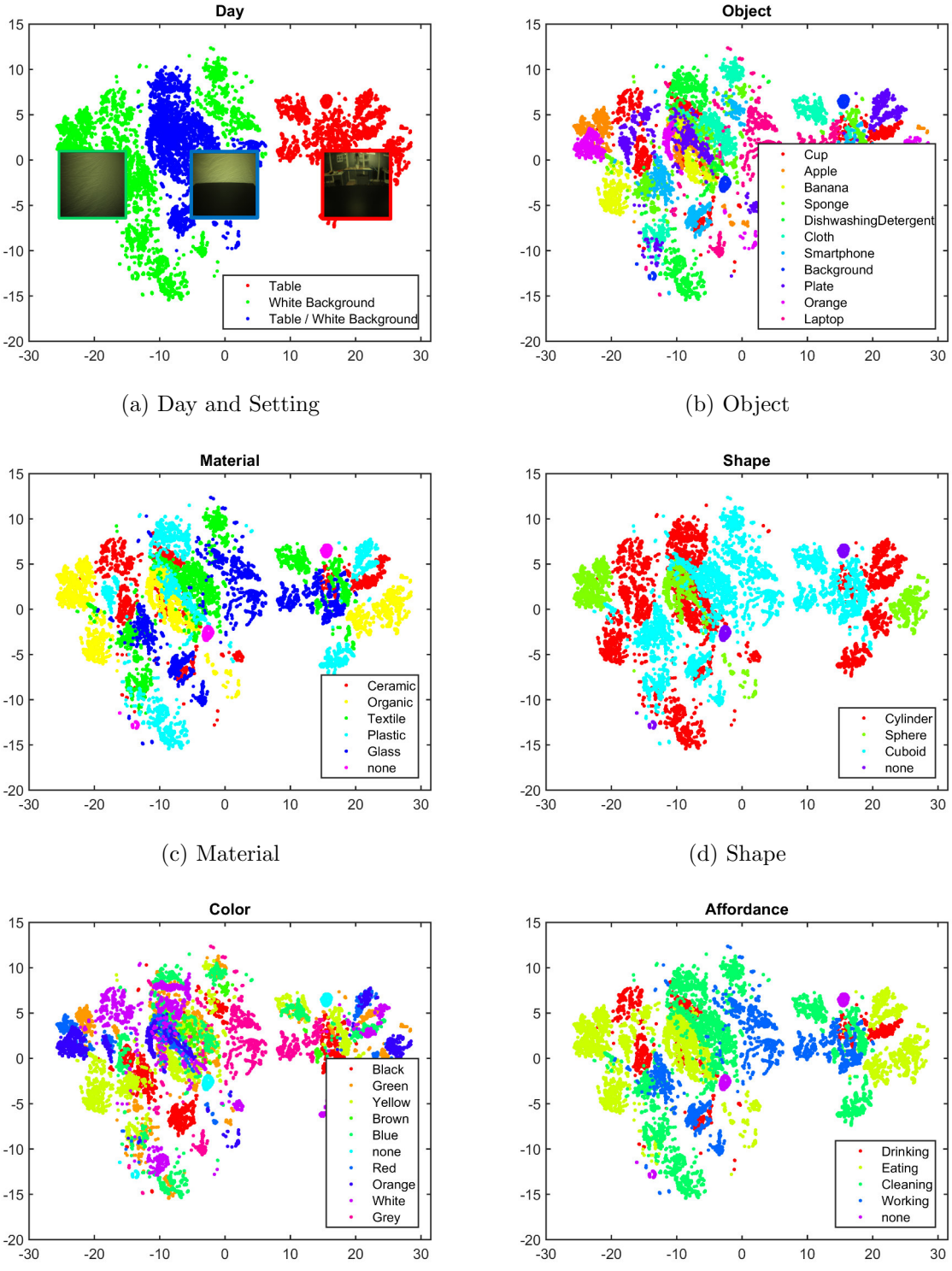
In the future, the possibly best and most reasonable approach to enrich the iCub's perception capabilities from a dataset point of view is to only collect datasets that have few images per category and combine it with existing datasets for training. This small amount of data is much cheaper and easier to obtain than large datasets and can be used for finetuning the networks. Finetuning does not require many images for training when the learning rates are adjusted. Furthermore, the models can optimize their recognition performance on the experimental setup settings.

4.1.2.4 Analysis of extracted features

In this subsection, we visualize extracted features and show how the ImageNet-trained ICS-CaffeNet is able to serve as a feature extractor. In related research work, these visualizations are usually done with extracted features from the validation data of the same dataset. The resulting figures show that a neural network can function as a very good

feature extractor. The figures illustrate this observations with clear boundaries between clusters. In this thesis, we use an ImageNet-pretrained network to extract the FC7 features from the training data of our collected TUM-ICS dataset. Although ImageNet and the TUM-ICS dataset are completely different datasets, our purpose is to show that this deep learning model can extract descriptive FC7 features from the TUM-ICS images. The better the model separates the classes in clusters between each other, the easier it is for GURLS to learn from the extracted features. This means that if the training data can be nicely separated in 2-dimensions using a dimensionality reduction technique, we expect that the classification accuracy for GURLS is higher for those attributes. For this experiment, we used the training images of TUM-ICS, which consist of 16,500 images total. These images were taken in three different settings: 1) the lab background with a table, 2) a white background without a table, and 3) a white background with a table. Alternatively speaking, the 16,500 images consist of 5500 training images for each of the three settings.

Figure 4.5(a) shows 6 visualization plots which are obtained using the tSNE - Barnes Hut implementation. Because this dimensionality reduction algorithm is unsupervised, the 4096 dimensional features are reduced to the same 16,500 2D point locations in all figures. The colors in the figures are dependent on the labels of the corresponding attributes.
Figure 4.5(a) shows the clearest separation of all clusters: the green group contains all images that belong to the background images with white background. These images were taken on three different daytimes: in the morning, in the afternoon and at night. The smaller green cluster between the blue and red group belongs to the images that were taken at night in low light conditions, thus differing from the other two daytimes. The blue group contains the features of the background images that are black and white, which lies between the images with white background and the images in the lab with the darker lab background. The latter group does not have white background images. The fact that this group lies furthest away from the others proves that the trained neural network can distinguish when and where an image has been acquired in this dataset.
The five remaining figures are more cluttered, as more classes exist. For object, we see in Figure 4.5 (b) that the algorithm is able to divide the features into small clusters. The clustering works well for the object's 11 classes, but it is notable here that many small subclusters or single points exist which do not lie in any larger group within the area.
The attribute material has six classes which are fairly divided in the visualization shown in Figure 4.5 (c). The classes 'Organic', 'Textile', and 'Glass' are almost perfectly divided into three parts. The feature points for 'Ceramic' are also well separated. Here, it is interesting to see how close the groups for the settings white background and table with white background lie together. The red 'Ceramic' cluster of the table setting with white background overlaps with the classes 'Organic', 'Plastic', 'Textile' in the center of the image. This overlap shows that the 4096-dimensional feature vectors are similar in these images. The purple-colored dots represent the background images, which lie very close together. The purple clusters are larger for the setting when a table is used, indicating that more than one color in the image leads to features that have a higher variance. As the white background cloth only contains one color, the corresponding features remain a very tiny



(a) Day and Setting

(b) Object

(c) Material

(d) Shape

(e) Color

(f) Affordance

Figure 4.5: Visualization of 16,500 4096-dimensional feature vectors reduced to 2 dimensions. The used t-SNE dimensionality reduction technique is unsupervised, thus it does not consider the labels during the optimization.

cluster on the visualization graph.
Figure 4.5 (d) shows a good separation of the four different shapes. The 2D feature points for 'Sphere' are located close together, with the exception of a small subgroup of single points in the center of the image. The points with 'Cuboid' labels tend to form large groups of points in the center of the image. The clusters for cylinders are separated well.
The visualization for 'Color' is shown in Figure 4.5 (e). Because 10 different colors exist, many small but well separated groups exist in the image. In the center part of the visualization, features of 'Brown', 'Blue', 'White', 'Orange' and 'Yellow' overlap, which indicates that misclassifications may happen for color in the setting of table with white background.
The affordances 'Eating', 'Cleaning' and 'Working' are well separated in 4.5 (f). The affordance 'Drinking' has the least amount of points except for 'Background', but seems to be the only one of the large four affordances to be separated over large areas compared to the number of points. Besides the three main groups, two smaller point groups and single red dots exist. These findings could give us a hint that 'Drinking' is misclassified more often than the other three affordances, which would explain the lower performance results for affordance detected earlier.

4.1.2.5 Analysis of CNN Heatmaps

In the ImageNet dataset, objects are located in the center of the images with good light conditions. No hands or other unintended objects disturb the images. Bolei Zhou et. al have introduced a technique to expose the implicit attention of convolutional neural networks in an image [99]. Their models, which are either trained on ImageNet or Places205, highlight the most informative image regions relevant to a predicted class. The highlighted region can be interpreted as the part of the image, based on which the neural network decides to which class the image belongs to. In this subsection, we use the pretrained network GoogLeNet-CAM, which is trained on ImageNet for over 120,000 iterations, to test where the most informative image regions are located in our dataset images. The results show us which parts of the shown objects draw the highest attention of the network, and if the hand of the supervisor holding the object disturbs the learning and classification of the network.

Five heatmap activations from five of our eleven objects are shown in Figure 4.6. For the object 'Cup', the network focuses on the center part of the object in the first four activations. In the last one, the cup's handle lies in the attention of the network. Because no hand is located in the image, the net's attention focuses completely on the object in the image.
The second test image contains the supervisor's hand with arm, which clearly disturbs the heatmap. As 'Sponge' is a very small object, it does not grab the network's attention in any of the heatmap activations. The first three maps focus on the hand, the fourth on the dark colored shirt, and the last one on the forearm.
The next example shows that it looks different for larger objects. Although the arm is

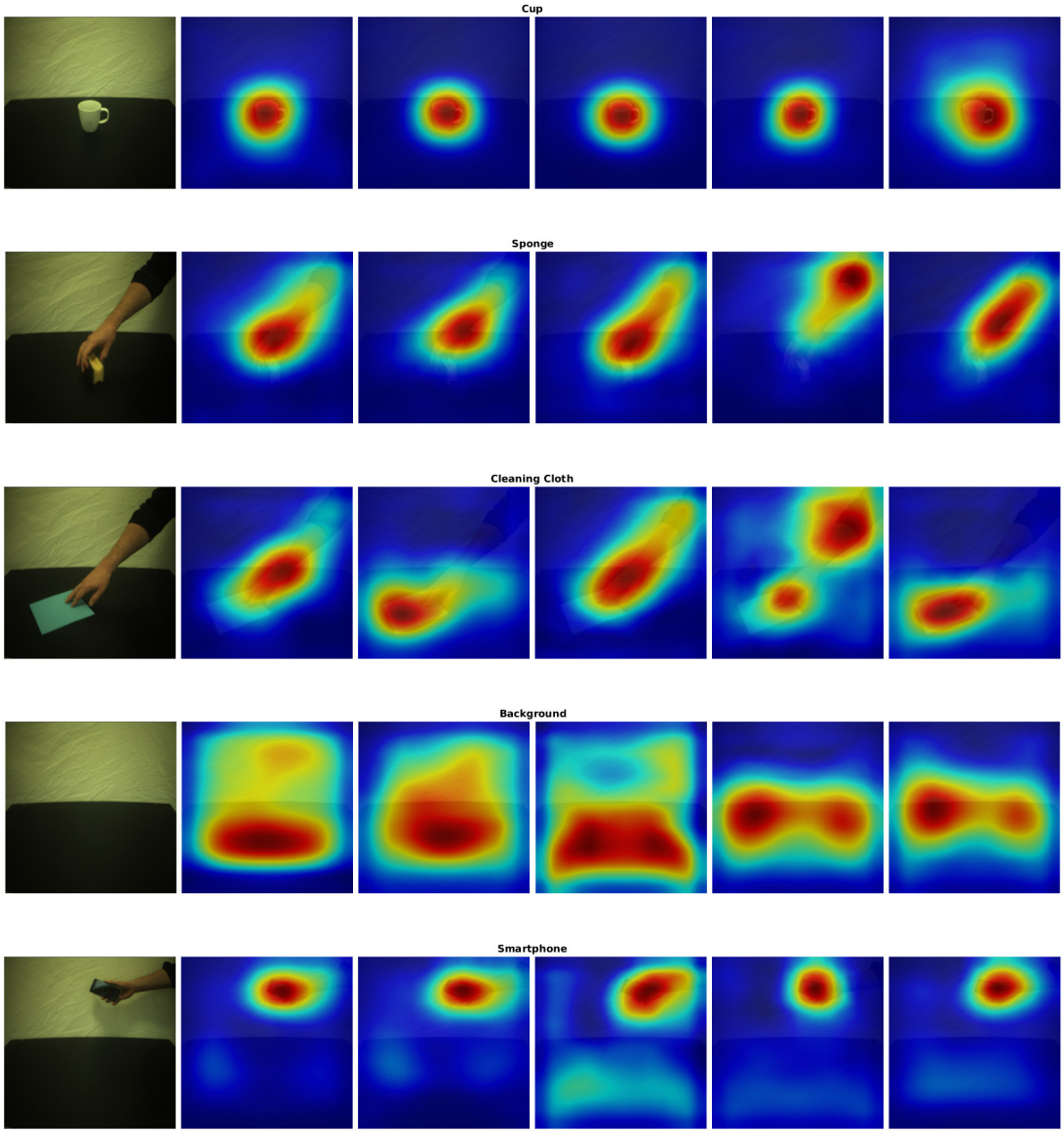


Figure 4.6: Our generated CAM heatmaps, produced with the pretrained GoogLeNet network from [99]. The shown objects are from the classes 'Cup', 'Sponge', 'Cleaning Cloth', 'Background', and 'Smartphone'. The left images are from the robot camera. The five images shown next to the robot camera image on the right are activation maps. The heatmaps highlight the regions where the convolutional neural network looks at when classifying the image.

located in about the same position in the image as in the previous example, the network recognizes the blue 'Cleaning Cloth' in the image. Two of the five activations solely focus on the object, namely the second and the last one. The first and third heatmap center the forearm while the fourth has two separate foci of attention: the fingers of the hand and the black shirt.
The last test image is from the class 'Background', which consists of the white fabric in the upper part of the image and the black table on the bottom. Although the network is not trained on our dataset, it is interesting to see that the heatmaps' main area of interest lies on the center part of the table where we would place our objects for classification. The dark red parts of the heatmaps are on the table's center in the first three images, and on the left and right parts of the table in the last two images. The white fabric is mostly ignored, except in the first activation map.
Due to the dark color of the table, it is hard to recognize black objects on the table. Therefore, black objects are held in front of the white fabric in the table settings. Although these are the only objects that are not placed on the table, the figure shows that the neural network looks at the right parts in the image.

4.1.2.6 Overfitted Model

Based on our experiences in this work with training deep convolutional neural networks, we donate one subsection in this thesis to the concept of overfitting. This subsection describes how overfitting occured in the beginning of our experiments. All results shown in this thesis are obtained with non-overfitted models, except the section 4.1.1.5 in which we explained why we need a new dataset.
One reason why overfitting can occur is the dataset: during the acquisition period of the TUM-ICS dataset, we estimate to have collected between 200,000 and 300,000 images. Many images in our collected dataset were redundant, i.e. they look almost identically due to our image acquisition frequency of 15 frames per second. Therefore, we only used a subset of 16,500 for training, 8,250 for validation, and another 8,250 for testing from the large dataset. Because of the "low" number of images for training a deep learning network and our solver settings, the model overfitted during training. Overfitting occurs when the model is too complex for the dataset, which means that the number of its parameters is too high relative to the number of images or observations. When a very complex model is overfitting, it achieves high performances on the validation set by memorizing the training data which it was trained on. However, it is not able to generalize the learned patterns in the data to correctly predict unknown examples. Because our data was randomly split into training, validation, and test sets from our recordings, the overfitted models performed very well on all sets. Thus it was not recognizable that the models were overfitting until they were tested with relatively poor performance on unknown objects. The performance on known objects was also lower for overfitted models, but still above 50 percent.
In this subsection, we provide some results we obtained on the TUM-ICS dataset with the overfitted models and compare it with the finetuned models. Afterwards, a look into the

	ICS-CaffeNet model			
	Overfitted (%)		Finetuned (%)	
	Known	Unknown	Known	Unknown
Object	47	-	**78**	-
Material	56	11	**87**	**62**
Shape	72	23	**90**	**65**
Color	75	4	**82**	**17**
Affordance	**55**	8	22	**38**

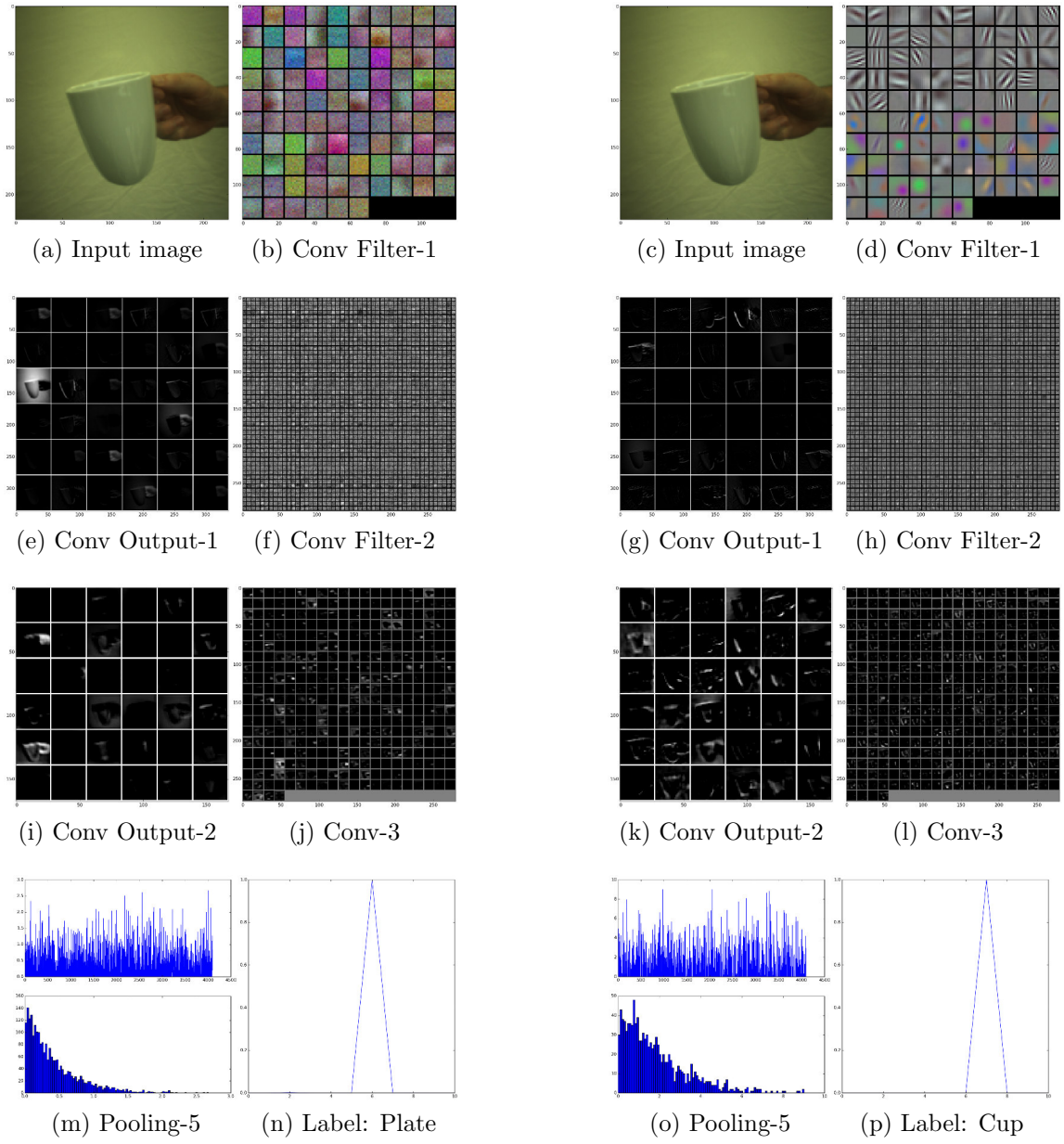
Table 4.13: Performance of our overfitted model: it performs worse than the finetuned model but still well on recognizing known objects. However, it is not able to generalize its knowledge on unknown objects. The finetuned model surpasses the overfitted model in all but one benchmarks.

visualized layers of the networks is taken for a comparison in Figure 4.7. The overfitted network, whose layers were all trained, is compared with the finetuned network, whose layers 1-7 were kept frozen and the last fully connected layer was finetuned.

Table 4.13 provides a performance overview of the overfitted model. It is able to correctly predict the object, material and affordance of known objects for accuracies between 47 and 55 percents. Shape and color are classified with 72% to 75%. Despite the good performance on known data, the performance on unknown objects shows that the model is overfitting: the model is supposed to accurately predict the detected attributes and affordances. The classification accuracy lies merely between 4 and 23 percent, which is worse than randomly guessing the attributes in all cases. This data shows that the model memorizes the training data without understanding the underlying concepts during classification. The following example shows why we need a nearly optimal classification accuracy: the probability to make the correct prediction based on all four labels is $0.11 \times 0.23 \times 0.04 \times 0.08 = 0.00008096$ percents and for the top 2 $0.11 \times 0.23 = 0.0253$ percents. The accuracy is too low to retrieve the right detected objects from our knowledge base. The goal is to reach values close to 1, so that the robot can make accurate guesses and knows if the object is known or not. The right columns in the table represent the performance values for the finetuned model. It outruns the overfitted value in each entry except for affordances of the known objects. The low performance for affordance displays that it is not possible to predict the correct affordance in images based on the objects that are recorded in our dataset.

4.1.2.7 Testing on known and unknown objects

Before we test the entire system, which consists of a combination of either ICS-CaffeNets or GURLS classifiers and knowledge, we check the performance of our machine learning models on an extra dataset of known and unknown objects. This set of images was obtained while we tested our knowledge system. The images differ to the original TUM-ICS dataset



(a) Input image (b) Conv Filter-1 (c) Input image (d) Conv Filter-1

(e) Conv Output-1 (f) Conv Filter-2 (g) Conv Output-1 (h) Conv Filter-2

(i) Conv Output-2 (j) Conv-3 (k) Conv Output-2 (l) Conv-3

(m) Pooling-5 (n) Label: Plate (o) Pooling-5 (p) Label: Cup

Figure 4.7: Layer Visualizations: Overfitted vs. Finetuned. The left side in the figure represents selected layers of the overfitted model, the right side shows the same layers of the finetuned model. Note the different learned filters on the top right images: the finetuned features are more distinctive. The overfitted model falsely predicts 'Plate' instead of 'Cup'.

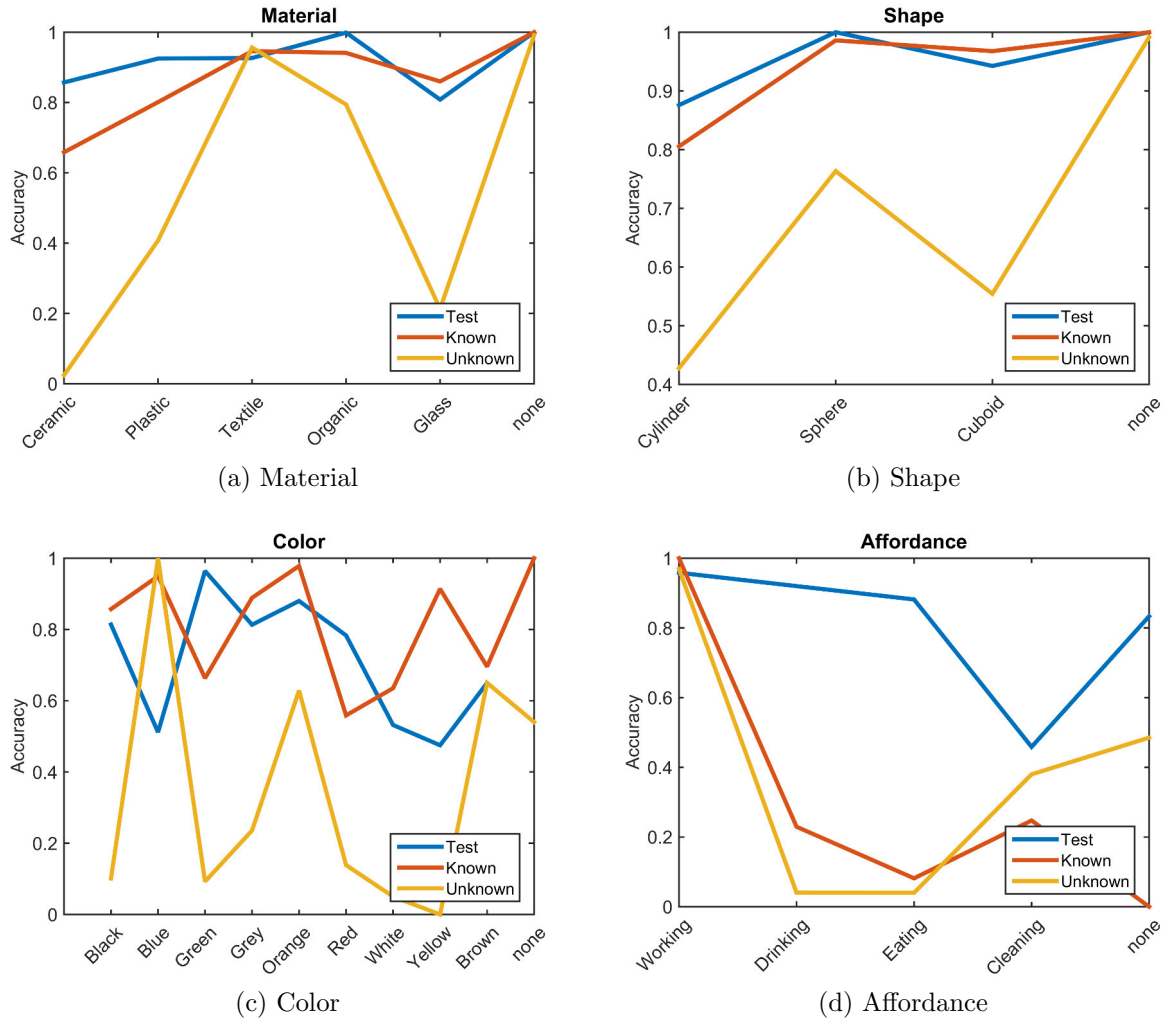
Figure 4.8: Unknown objects. The shown objects are put on the black table and are tested with both our finetuned ICS-CaffeNets and our proposed system. Beginning from the left, the objects are from the classes: 'Bowl', 'Box', 'Display', 'Glass', 'Headphones', 'Ice Tea', 'Keyboard', 'Kindle', 'Kiwi', 'Laundry Detergent', 'Lemon', 'PC Mouse', 'Paprika', 'Soap', 'Soap Box', 'Towel', 'Waterbottle', 'Watermelon'. Not displayed but also used for learning are 'Book', 'Cornflakes', and 'Sprayer'.

Comparison: Known and Unknown Objects				
	ICS-CaffeNet method (%)		GURLS method (%)	
	Known	Unknown	Known	Unknown
Object	**78**	-	11	-
Material	**87**	62	19	18
Shape	**90**	65	33	26
Color	**82**	17	11	12
Affordance	22	38	**31**	19

Table 4.14: Performance of our finetuned ICS-CaffeNet and GURLS on known and unknown objects. ICS-CaffeNet obtains the best results on object, material, shape, and color. GURLS achieved a higher accuracy on affordance. For both ICS-CaffeNet and GURLS, the attribute shape is the best metric.

that they are moved by hand in horizontal lines along and across the visible parts of the table. The objects are not rotated, but the extreme positioning of the objects in the corners of the table is challenging for the machine learning classifiers. Figure 4.8 shows the unknown objects when standing still or being held in front of the robot camera. This performance analysis is based on 761 images of the known objects and 1041 images from the unknown images.
Visualizations of this experiment are shown in Figure 4.9 for ICS-CaffeNet. The results for GURLS are not shown as ICS-CaffeNet produces much better results, as compared and summarized in Table 4.14. The Figure for the attribute 'Material' shows that especially 'Ceramic' and 'Glass' can not be correctly predicted on unknown objects. These attributes could therefore not be learned and generalized as desired.
The attribute 'Shape' suggests that the performances of the classes 'Cylinder' and 'Cuboid' are lower than for 'Cylinder' and 'Cuboid'. The class 'None', which only the object class 'Background' has, and 'Sphere' are recognized well.
Unknown, blue objects are correctly predicted while no yellow object could be detected. In the other cases, the data follows the same pattern: the algorithm's accuracy on the unknown objects is between 20 and 40 percentage points lower than the accuracy on known objects.
The graph for affordance shows that the predictions for 'Working' and 'Cleaning' are nearly the same for all three image datasets. Otherwise, the prediction accuracy on known and unknown objects is about the same, although that the class 'None' is never recognized.

Following the analysis of the performances in Figure 4.9, Table 4.14 provides the numerical results for the performance analysis on known and unknown objects. As we have seen in the prior shown visualizations, ICS-CaffeNet performs very well on the known objects, with the exception of affordance. The performance on the unknown objects is good for material and shape, but very low for color with 17 percents. Affordance reaches 38 percents, which is better than for the known objects. Both values for color and affordance indicate that the underlying relationships of the metrics were not learned properly by the model. For



(a) Material (b) Shape

(c) Color (d) Affordance

Figure 4.9: The performance of ICS-CaffeNet on attributes and affordance is compared on a set of extra images of known (red line) and unknown (yellow line) objects on the same testing day. The models are trained on 'Known' objects, but on other images. The blue line represents the accuracy we obtained on our test set. Top left: The materials 'Ceramic' and 'Glass' could not be generalized on unknown objects. Top right: 'Cylinder' obtains the lowest accuracy. Bottom right: This chart illustrates that our dataset does not have enough object classes with different colors. The accuracy for unknown objects is very low. Bottom right: Because of the low accuracy for both known and unknown objects, it becomes visible that it cannot be seen in an image for which tasks an object can be used. The class 'Working' ('Laptop', 'Smartphone', 'Kindle', 'Keyboard', 'Display', 'Book') obtains a nearly perfect score. It is possible that other relationships have been learned to distinguish these objects from the others. Possibly, because most of them consist of 'Plastic' material, have a screen and their shape is rectangular.

color, this means that 1) the dataset did not have enough images with different colors available for training, or 2) that the set of images which we tested in this experiment were not chosen by testing all colors known to the model or may be extreme cases such that wrong predictions are made compared to the initial labeling.
In contrast, the results for GURLS are very low, ranging only between 11% for the known objects to 33% for the known shape. When comparing the data of the known and unknown objects for GURLS, the values of unknown objects are similar for each attribute. This shows that the linear, regularized least-squares loss function of GURLS was not able to learn the intended relationships from the high-dimensional feature data. The resulting values are also very close to randomly guessing the correct attribute: in rounded form these are for object $1/11 =\sim 9\%$ (GURLS (known)/(unknown): 11%/-), for material $1/6 =\sim 16.7\%$ (GURLS (known)/(unknown): 19%/18%), for shape $1/4 = 25\%$ (GURLS (known)/(unknown): 33%/26%), for color $1/10 = 10\%$ (GURLS (known)/(unknown): 11%/12%) and for affordance $1/5 = 20\%$ (GURLS (known)/(unknown): 31%/19%). Consequently, we expect to achieve a much better overall system performance of our proposed system with our ICS-CaffeNet models in the final experiments.

4.2 Enhanced deep network results

This section presents the experimental results of the proposed system. It combines the machine learning algorithms and the knowledge base, which we call enhanced deep network. We first provide an overview of the system from a user-interface and user-experience perspective before we analyze the results for ICS-CaffeNet and GURLS in the following subsections. An overview of the interaction with the robot during learning is shown in Figure 3.13. In this system, it is necessary to supervise the robot learning with a human teacher. The supervisor is responsible to either approve the robot guesses or to give input to the robot by correcting certain predicted labels. In the following, we describe the workflow of our proposed system in detail.

4.2.1 ICS-CaffeNet and Semantic Reasoning

The results of our experiment with ICS-CaffeNet are described in this subsection. The performance of the five ICS-CaffeNets is benchmarked with our proposed ICS-CaffeNet and reasoning system. The results are evaluated based on four values: we compare the object recognition accuracy of ICS-CaffeNet (object only) on the known and unknown objects and provide the same for our proposed system. The data displayed in the table 4.15 compares the number of detected objects to the number of trials and percentage points. Note that the performance of ICS-CaffeNet on unknown objects is 0.00 because it cannot predict a class which it was not trained on. In the first trial of our reasoning system, the system always fails to classify the right object, but it learns it to be able to recognize it

the next time. Because we tested each object for at least 2 trials and the system always fails to recognize it in the first iteration, the performance of our proposed system cannot be much higher than 0.5, except we tested a correctly predicted object multiple times.

ICS-CaffeNet and knowledge reasoning system				
Guess	ICS-CaffeNet		**ICS-CaffeNet + Reasoning System**	
Accuracy Known	11/11	100 %	11/11	**100 %**
Accuracy Unknown	0/49	0 %	27/49	**55 %**

Table 4.15: ICS-CaffeNet and KnowRob. ICS-CaffeNet cannot recognize an object which is unknown. Our reasoning system always fails in the first trial when recognizing an unknown object, but learns it for the next tests.

The system was tested for eleven trials (or iterations) with ICS-CaffeNet on known objects. The deep neural network was able to always classify the correct object. Thus, the performance achieved for the Caffe-only version is 11/11 or 100%. Because the deep neural network makes one Caffe guess before we do up to three guesses afterwards, the correct object is detected every time, also leading to a perfect recognition performance. Even when the Caffe guess is not used in the enhanced deep network, we also achieved 100% performance by using semantics with our first and second guesses in a separate test, which is not listed in the table.
A much more challenging task is to be able to learn and recognize new objects, which is the main motivation for this research project. Because the deep networks are not trained on unknown objects, they are not able to recognize new objects. Thus, the performance is 0/49 or a mere 0%. It is the drawback of convolutional neural networks that they need a lot of data per object for classification. When seeing a new or unknown object, the reasoning also fails because its knowledge base does not contain an object which matches the name of the object with or without the desired attributes. However, the robot learns the object with the text input provided by the supervisor and is able to recognize all objects in the first or second guess of our reasoning method. In the experiment, we achieved a performance of 27 detected objects in 49 trials, reaching a performance of over 55%. In a few cases, when the recognition failed in the system, a third iteration was needed to recognize the new object. In this case, a common reason is that the predicted labels have changed in the second trial after the object was learned in the first trial. Consequently, the object was also considered unknown in the second iteration. However, this leads to more sets of predicted labels which improves the overall recognition for this object.

In Table 4.16, we provide the testing results of the enhanced deep network. Previously, we described how many trials we need with our system to learn an object until it is recognized. In this table, we measure the ICS-CaffeNet performance only for testing, which we did not do in the previous part. For each class in the iCubWorld7 and TUM-ICS dataset, we performed five tests to recognize the object. The comparison between ICS-CaffeNet and the enhanced deep network shows that the reasoning improves the recognition. Similar to

the crosstesting experiment described in section 4.1.2.3, we tested the deep learning model which is trained on TUM-ICS on the iCubWorld dataset. The "4 similar" category contains the objects which the model is also trained on in our dataset. "3 different" contains the classes which are unknown for our model. In the performed experiment, our model failed to detect similar objects from the iCubWorld dataset, resulting in 0 % accuracy. This is due to the different background and lighting conditions. Meaning that the network cannot generalize towards a different dataset. For unknown objects in both datasets, the model cannot recognize any classes. At 85%, the performance on known objects from our dataset was as good as expected. The data of the reasoning system shows that it improves the performance of the standalone deep network in all cases. For known objects, it achieves the nearly perfect score of 98%. Although the deep network failed in the other cases, the enhanced deep network is able to recognize many of them (65%, 67%, 47%). This data demonstrates that the system was successfully learned with the new objects as described earlier. It obtains the highest values on the TUM-ICS dataset, and scores the lowest on unknown objects from the iCubWorld dataset. This experiment demonstrates that the enhanced deep network increases the object recognition performance, especially when the predictors' classifications fail.

Comparison: ICS-CaffeNet and ICS-CaffeNet + Semantic Reasoning				
	TUM-ICS Dataset (%)		iCubWorld Dataset (%)	
Methods	11 known	22 unknown	4 known	3 unknown
ICS-CaffeNet	85	0	0	0
ICS-CaffeNet + Reasoning	**98**	**65**	**67**	**47**

Table 4.16: Comparison between ICS-CaffeNet and ICS-CaffeNet with Semantic Reasoning. The performance of recognizing objects is compared on known and unknown objects, both on the iCubWorld and TUM-ICS datasets. The enhanced deep network is already trained on the unknown objects, as illustrated in the previous table. We tested each object class five times with our on-line learning system. During the testing, the predicted attributes are not learned.

Summarizing the results for ICS-CaffeNet, this experiment shows that deep learning networks are very powerful for classifying objects for which they are trained on. However, these models fail on classifying new objects. Our semantic reasoning system can be used to teach the robot unknown objects. The results also show that our system is able to detect known objects. Thus, it improves the results for both known and unknown objects.

4.2.2 GURLS and Semantic Reasoning

In this subsection, we repeat the same experiment with GURLS classifiers. In the previous section, we have seen that the overall recognition performance of GURLS is lower than ICS-CaffeNet, especially on unknown objects. Our proposed reasoning system with ICS-

CaffeNet has shown that it can improve the performance of our deep learning system. We first evaluate the system performance as before and analyze if the reasoning system can neutralize the lower accuracy of GURLS.

GURLS and knowledge reasoning system				
	GURLS		**GURLS + Reasoning System**	
Accuracy Known	4/28	14 %	22/28	**79 %**
Accuracy Unknown	0/53	0 %	14/53	**26 %**

Table 4.17: GURLS and KnowRob. GURLS cannot recognize an object which is unknown. The reasoning system always fails in the first trial when recognizing an unknown object, but learns it for the next tests.

In this experiment, we needed to perform 28 trials for known objects since it took longer in some cases to recognize the correct object. GURLS recognized only 4 out of 28 objects or 14%, which is a big difference to the previous 100% ICS-CaffeNet was able to recognize. In contrast, our reasoning system performed well: despite the poor classification accuracy, the system could recognize 22 out of 28 objects, or 79%. This value is lower than the 100% accuracy which ICS-CaffeNet achieved. It shows that the system is capable to include contextual information into the reasoning process which improves the recognition accuracy. As expected for the unknown objects, GURLS is not able to classify objects which it is not trained on: it achieves 0%. Out of 53 trials, the reasoning system learned and recognized 14 objects, which represents an accuracy of 26%. The 53 trials were made on 14 objects, which means that an average of almost four trials were needed to correctly recognize an object. This means that the system needed to learn about three sets of predicted labels for the object in order to robustly recognize it in future trials. One particular problem this creates is the large number of objects that exist in the knowledge base. After having learned many objects, the query for a particular object can lead to 10 to 20 inferred solutions (some with multiple entries). In some cases, almost all trained objects appear in the answers from the knowledge base. One GURLS guess and three system guesses are not enough to retrieve the correct object, even if it is in the inferred objects. The reason for the large amount of inferred objects is the larger number of trials we performed for each object. Thus, we learned a different set of predicted labels multiple times. In every case, the predicted labels are different than before due to the inconsistency of the GURLS classifiers during recognition.
The results show that the reasoning system also improves the recognition performance achieved by GURLS only. However, the resulting performance of recognizing unknown objects is below 26%, which makes it a lot of manual work to teach the robot to learn new objects. In a few cases, the system fails to recognize the correct object because of the large number of inferred objects. One solution to overcome this problem is to use a better machine learning classifiers, such as our ICS-CaffeNet models, and have a higher number of labels for each attribute. This gives each object in the database unique characteristics. Following this approach, each query will result in only one or a few inferred objects, which

results in a much higher system performance.

Chapter 5

Discussion

In this work, we presented our experiments and analyzed the results in detail to extract important insights into how well our algorithms performed. In this chapter, we discuss our findings: we first interpret our results compared to our expectations based on related work and previous experiments. We provide an overview about the key results and give explanations to possible causes and consequences. In the next section, we describe limitations of our research work. Suggestions for possible future research are made in the last section.

5.1 Interpretation of results

Because deep learning algorithms achieve close-to-optimal results for many object recognition tasks, we expected and achieved very high performance of our models trained on objects. A much more challenging task is to learn attributes and affordance from our relatively small dataset and generalize the learned relationships on unknown data. On the test set, the ICS-CaffeNet models trained on objects and contexts achieved accuracies between 68 and 93 percents. GURLS classifiers scored lower accuracies. Although the ImageNet-trained ICS-CaffeNet functions as a deep feature extractor and separates the features in a high dimensional space, the GURLS's underlying RLS loss function is not able to generalize the learned concepts. The problem is that multiple settings within our dataset makes it more difficult for the linear RLS loss function to differentiate between the classes.
Among known and unknown objects, material and shape were the best attributes which could be visually recognized from the images. The results on unknown objects indicate that color could not be generalized. This is surprising as we included objects with different colors in one category to weaken the relationship between learned objects and learned colors. Consequently, more objects per color are needed to improve the recognition accuracy. The values for affordance indicate that it is not possible to predict the affordance of an unknown object in a single image. Prior knowledge about the object is required,

or semantic reasoning about other attributes could be used to infer the affordance. For example, the attribute "has parts" could infer that a cup can be used for 'holding' because of a recognized handle.
To prevent overfitting on a small dataset, it is required to freeze all layers of the network except the last one during training. Otherwise, the network learns and optimizes its parameters on the noise of the training data and does not "understand" the underlying relationship. Visualizations show that the learned features of a finetuned ICS-CaffeNet model are more distinctive than the features of an overfitted model, due to the larger and more diverse ImageNet dataset. In contrast to the overfitted model, the finetuned model is able to generalize the learned relationships on unknown data.
We created the new dataset TUM-ICS, which contains 23 object classes within 11 object categories. It provides a greater variety of labeled attributes than the iCubWorld28 dataset, which is beneficial for training our models on contextual information. The crosstesting experiment proves that the reason we obtained poor performance with iCubWorld28-trained models on test images from another setting is due to overfitting. In the beginning of our experiments, we falsely assumed that the background, room settings and lighting conditions were the main cause for the lower classification accuracies.
Heatmaps indicate that our models look both on the object and the supervisor's hand in the images. However, when no hand can be seen in the camera's field of view, the heatmaps highlight only the areas around the object. To ensure that learning occurs on the objects and not on the hand positions, we included images into our dataset which show our objects without scene-disturbing hands. We conclude from the high classification performance that the models have learned to distinguish between the classes based on the characteristics of the shown objects.

Both the deep learning network and the proposed enhanced deep network are able to recognize all known objects in the final experiment. However, the stand-alone deep learning network has one drawback despite its performance: it fails when the object is unknown. In contrast, our system is able to learn the new object based on the object's contextual information. In the CNN version, every object could be learned and recognized by the system in no more than three trials. Although the system is not able to recognize an unknown object in the first trial, the system learns it and is able to recognize it in one of the next trials. Because the GURLS classifiers perform worse than the deep learning models, the recognition performance of the enhanced network on known objects was lower. In this case, the reasoning system improves the overall recognition accuracy significantly. Even if the predicted labels are wrong, our system enables the robot to learn the object with the predicted labels and recognize it in a later trial. Figure 5.1 provides an overview of the benefits of the enhanced deep network.

Comparison		
	Deep Learning	Enhanced Deep Network
Recognizes known obj	✓	✓
Performance known obj (%)	85	**98**
Ability to learn unknown obj	✗	✓
Recognizes unknown obj	✗	✓
Performance unknown obj (%)	0	**65**
Considers visually similar objs	✓	✓
Considers context: material	✗	✓
Considers context: shape	✗	✓
Considers context: color	✗	✓
Considers context: affordance	✗	✓
Considers contextually-similar objs	✗	✓
Recognizes obj w/ wrong prediction	✗	✓
Improves recognition w/ semantics	✗	✓
Total Score	2	**13**

Table 5.1: Comparison between the standalone deep learning network ICS-CaffeNet, which was the best-performing machine learning method in our experiments, and our proposed enhanced deep network. This overview summarizes the overall benefits of the proposed system design. The system improves the recognition of known objects and enables the robot to learn new objects by including contextual information in the recognition process with semantic reasoning.

5.2 Limitations of the proposed method

The combination of deep learning with semantic reasoning is a very good approach to give cognitive systems the ability to understand and interpret objects within an environment. Nonetheless, applying deep learning in robotics also provides drawbacks and limitations. Although our collected and multi-labeled dataset is among the most important enablers of our system, it also embodies a limiting factor. For our use case, the limiting factor is not primarily the number of our training images, but the variety of different objects with distinctive and unique attributes which we used for training. If an attribute or label does not exist in our initial dataset, it is not possible to predict this class on unknown objects. In a dataset, each represented class of an object or attribute must have several different examples to ensure that the right relationships can be learned. Otherwise, a correlation between the recognized object class and the detected attribute exists, as we have experienced during our first tests with e.g. the object 'Apple' and the color 'Green'. Including more objects per category with different attributes solves this problem.
Learning new objects on-line using our enhanced deep network requires human supervision.

If no teacher, user, or co-worker is present, the system can not be used to teach our robot new objects. Supervision of our system requires either reading or listening to the robot's speech output and interacting with the robot via text input. When the robot learns a new object, the corrected labels must be provided by the supervisor. Enabled speech recognition on the robot or further automation of the object learning process could reduce the amount of effort needed from the teacher.
When the classification output is false and changes during the learning of an object, our enhanced deep network learns the unknown objects multiple times with different predicted labels. This enables the robot to recognize the object in the future, even if the classification fails. On the other hand, it is a problem if many objects are inferred that have been learned with many attributes. These objects with the same predicted (but not true) attributes make it hard to query and retrieve the correct object from the knowledge base. To prevent this, we equipped our system with four guesses the robot is allowed to make during testing. Because our deep learning networks predict very accurately, the system learns the new objects very fast. On the other hand, our experimental results of GURLS show that in some cases the number of guesses is not enough. This happens especially if the classification fails and many objects have been learned. Additionally, objects in the knowledge base are prioritized which have true labels. That means, if the correct object could be inferred in our reasoner with the same predicted labels such as e.g. 'Plate' has as true labels, the solution 'Plate' has a higher priority than the new object. This problem can be solved by either increasing the number of system guesses, introducing a larger number of unique attributes and/or labels, or using more accurate classifiers.
Lastly, simultaneously recognizing objects and contextual information with deep learning requires computation power. Because of our technical equipment, this issue has not been a problem for us in our research experiments. However, for autonomous mobile robots which have very limited energy resources and/or low-performing CPU/GPUs, it is not possible to use deep learning in robotics (yet). Specialized hardware and current advances in efficient computing architectures for deep learning will resolve this problem in the near future.

5.3 Outlook

The performance of our overall system is highly dependent on the 1) deployed algorithms, 2) the dataset which we use for training, 3) our overall system architecture, and 4) the amount of required supervision. Each of these points provides a starting point to increase the system performance by improving only one of them. Theoretically, by using a better model, a dataset which contains twice as many objects and labels per category, and a slightly changed system architecture, we could very likely achieve an incremental increase in system performance.
However, the goal of a promising follow-on research project would not be to gain a small percentage increase in performance. The goal of this thesis is to demonstrate that combin-

ing deep learning with semantic reasoning improves the object recognition performance of a robot and enables to learn new objects on-line. Our results indicate and prove that our proposed system enriches the robot's visual recognition capabilities. Consequently, new ways need to be found to overcome the necessary hurdles which prevent the adoption of similar perception systems in today's robotics.

A promising approach would be to provide one common, but modified ImageNet-trained model. For each image, the network predicts not only the object but also the contextual information. It is enough and much more efficient to use one deep learning network to predict an object's attributes. However, as we have experienced during the experiments, the classification of this network on known and unknown objects will not always be correct. Here, the approach from this thesis could be used to learn new objects based on the predicted labels. Human supervision could be reduced by decreasing the involvement of the teacher and substituting it with a cloud-based system. Through parallel processing of large amounts of data, the cloud-system is able to more accurately predict an object class than the robot. When a new object is learned by a robot, certain objects can be synchronized with the cloud and the knowledge can be shared among the robots.

The application of natural language processing algorithms in robotics could further make the human-robot interaction more natural. Teaching robotic systems to learn new objects or tasks can be performed in a gamified manner with speech commands.

For the reasoning part, accessing the internet as a knowledge resource could significantly increase the object learning capabilities. Based on the existing KnowRob ontology, learned objects can be easier integrated and grouped within the ontology. By following this approach, the knowledge base can be extended to many 100s, 1000s or millions of objects with specifically labeled-attributes. The technical potential of utilizing deep learning networks and semantic reasoning to learn about object properties and the robot's environment could be fully leveraged. Knowledge sharing among robots can make the inferred knowledge available to all robots.

Based on this outlook, promising future research could involve how natural language processing can be used to make the supervision easier. Open-source deep learning networks which predict object and contextual information would be beneficial to enhance the visual recognition performance in robotics. From an architectural point of view, research also needs to answer which role the internet can play to enhance a robot's learning capabilities without making the robot dependent on it. Knowledge sharing across the cloud is very promising, but it is important to find out to which degree a robot should rely on the knowledge which is inferred locally compared to updates it receives from the cloud. Therefore, "knowledge fusion" and semantic reasoning are needed to update the available knowledge and enable a robot to autonomously and intelligently act in an environment.

Chapter 6

Summary and Conclusion

In this thesis, we presented and analyzed deep learning algorithms that are trained on objects, attributes and affordances. We combine deep learning and semantic reasoning techniques to improve a robot's visual recognition capabilities and teach iCub to recognize new objects based on contextual information.
To achieve our goal, we first evaluated the performance of finetuned convolutional neural networks and RLS classifiers, which are trained on extracted high-dimensional feature vectors. For this analysis, two datasets are used for training and testing, namely the iCubWorld28 and the introduced TUM-ICS dataset. For both evaluated datasets and all metrics, analysis shows that the deep learning models were able to generalize the learnings and achieved better results than RLS classifiers. Visualizations demonstrate that deep networks can detect the relevant parts of an image and extract semantic features from our dataset.
Lastly, we proposed and compared two versions of enhanced deep networks that combine the benefits of machine learning algorithms with semantic reasoning techniques. The results validate that the systems improve the overall recognition accuracy and learning capabilities of our iCub robot. For known and unknown objects, the recognition performance improved from 85% to 98% and from 0% to 65%, respectively. Even if the classifiers' predictions are wrong, the system makes it possible to recognize the shown object.
Although human supervision is required, robotic systems benefit from this system due to the evidence that semantic reasoning techniques enhance stand-alone machine learning algorithms. In the future, concepts of our system can be adapted to enrich autonomous perception systems with contextual understanding and on-line learning capabilities of new objects.

List of Figures

List of Tables

Bibliography

[1] Michael Copeland. What is the difference between artificial intelligence, machine learning, and deep learning? https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/, July 2016.

[2] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In David Touretzky, editor, *Advances in Neural Information Processing Systems (NIPS 1989)*, volume 2, Denver, CO, 1990. Morgan Kaufman.

[3] Y. LeCun, L. D. Jackel, B. Boser, J. S. Denker, H. P. Graf, I. Guyon, D. Henderson, R. E. Howard, and W. Hubbard. Handwritten digit recognition: Applications of neural net chips and automatic learning. In E. Sanchez-Sinencio and C. Lau, editors, *Artificial Neural Networks*, pages 463–468. IEEE press, 1992.

[4] Q.Z. Wu, Y. LeCun, L. D. Jackel, and B.S. Jeng. on-line recognition of limited vocabulary chinese character using multiple convolutional neural networks. In *Proc. of the 1993 IEEE International Symposium on circuits and systems*, volume 4, pages 2435–2438. IEEE, 1993.

[5] Y. LeCun and Y. Bengio. Pattern recognition and neural networks. In M. A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1995.

[6] Alex Krizhevsky, Ilya Sutskever, and Hinton Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25 (NIPS2012)*, pages 1–9, 2012.

[7] Christian Szegedy, Scott Reed, Pierre Sermanet, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. pages 1–12.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik, and U C Berkeley. Rich feature hierarchies for accurate object detection and semantic segmentation. 2012.

[9] Jonathan Long, Evan Shelhamer, and Trevor Darrell. [Slices] Fully convolutional networks for semantic segmentation. *Cvpr 2015*, 2015.

[10] Matthew D. Zeiler and Rob Fergus. Visualizing and Understanding Convolutional

Networks arXiv:1311.2901v3 [cs.CV] 28 Nov 2013. *Computer Vision–ECCV 2014*, 8689:818–833, 2014.

[11] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images. 2014.

[12] Arnab Paul and Suresh Venkatasubramanian. Why does deep learning work? - A perspective from group theory. *CoRR*, abs/1412.6621, 2014.

[13] H. W. Lin and M. Tegmark. Why does deep and cheap learning work so well? *ArXiv e-prints*, August 2016.

[14] Nitin Indurkhya and Fred J. Damerau. *Handbook of Natural Language Processing*. Chapman & Hall/CRC, 2nd edition, 2010.

[15] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June-2015:3156–3164, 2015.

[16] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei a Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[18] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the Game of Go with Deep Neural Networks and Tree Search. (1):1–37.

[19] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-End Training of Deep Visuomotor Policies. *Arxiv*, page 6922, 2015.

[20] Li Fei Fei. If we want machines to think, we need to teach them to see. http://www.wired.com/brandlab/2015/04/fei-fei-li-want-machines-think-need-teach-see/, 2015.

[21] Yoshua Bengio. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127, January 2009.

[22] Olga Russakovsky and Li Fei-fei. Attribute learning in large-scale datasets.

[23] Victor Escorcia, Juan Carlos Niebles, and Bernard Ghanem. On the relationship between visual attributes and convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1256–1264, 2015.

[24] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by betweenclass attribute transfer. In *In CVPR*, 2009.

[25] Vittorio Ferrari and Andrew Zisserman. Learning visual attributes.

[26] Ian Sheshadri, Aashish, Endres. Describing Objects by their Attributes. pages 1778–1785, 2012.

[27] Iman Awaad, Gerhard K Kraetzschmar, and Joachim Hertzberg. Finding Ways to Get the Job Done: An Affordance-Based Approach. *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 499–503, 2014.

[28] Niklas Barkmeyer. Deep learning: Convolutional neural networks for object recognition. TUM Advanced Seminar. Supervisor: Andreas Holzbach, TUM Institute for Cognitive Systems, February 2015.

[29] Yoshua Bengio, Ian J. Goodfellow, and Aaron Courville. Deep learning. Book in preparation for MIT Press, 2014.

[30] Yann Lecun, Leon Bottou, Yoshua Bengio, and Patrick Ha. Gradient-Based Learning Applied to Document Recognition. (November):1–46, 1998.

[31] Receptive Fields in the Cat's Visual Cortex Hubel and Wiesel.pdf.

[32] Kunihiko Fukushima. Biological Cybernetics. 202, 1980.

[33] Dan Cires and Ueli Meier. Multi-column Deep Neural Networks for Image Classification arXiv : 1202 . 2745v1 [cs . CV] 13 Feb 2012 Multi-column Deep Neural Networks for Image Classification. (February), 2012.

[34] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR)*, 15:1929–1958, 2014.

[35] Pieter Abbeel. Uc berkeley eecs. Research website.

[36] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[37] Christopher K. I Williams. Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. *Journal of the American Statistical Association*, 98(3):489–489, 2003.

[38] Evan Shelhamer. This Business of Brewing : Caffe in Practice. *Stanford course CS231.*

[39] Deep learning tutorial. http://deeplearning.net/tutorial/. Accessed December 18th, 2014.

[40] University of Montreal Yoshua Bengio. Introduction to gradient-based learning. http://www.iro.umontreal.ca/ pift6266/H10/notes/gradient.html. Accessed January 17th, 2015.

[41] Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines vinod nair.

[42] Yann Lecun and Koray Kavukcuoglu. Convolutional Networks and Applications in Vision. pages 253–256, 2010.

[43] Dan Cires, Ueli Meier, and Jonathan Masci. A Committee of Neural Networks for Traffic Sign Classification. 1(1).

[44] Raia Hadsell, Pierre Sermanet, Jan Ben, and Ayse Erkan. Learning Long-Range Vision for Autonomous Off-Road Driving. 1(1).

[45] Alex Kendall, C V May, and King College. Convolutional networks for real-time 6-DOF camera relocalization.

[46] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localisation and mapping (slam): Part i the essential algorithms. *IEEE ROBOTICS AND AUTOMATION MAGAZINE*, 2:2006, 2006.

[47] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *The Astrophysical Journal*, 815:43, 2015.

[48] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[49] Allan Jabri, Armand Joulin, and Laurens Van Der Maaten. Revisiting Visual Question Answering Baselines. 2016.

[50] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[51] Ryan Rifkin, Gene Yeo, and Tomaso Poggio. Regularized least-squares classification.

[52] G Pasquale. Teaching iCub to recognize objects using deep Convolutional Neural Networks. pages 1–5, 2015.

[53] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.

[54] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[55] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *Icml*, 32:647–655, 2014.

[56] Martin Kleinsteuber, Hao Shen, and Matthias Seibert. Information Retrieval In High Dimensional Data. 2014.

[57] B. C. Moore. Principal Component Analysis in Linear Systems: Controllability, Observability, and Model Reduction. *IEEE Trans. Automatic Control*, AC-26:17–32, 1981.

[58] Jonathon Shlens. Shlens2006_PCATutorial. pages 1–13, 2005.

[59] L J P Van Der Maaten and G E Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[60] Laurens Van Der Maaten. Learning a Parametric Embedding by Preserving Local Structure. *JMLR Proceedings vol. 5 (AISTATS)*, pages 384–391, 2009.

[61] Laurens Van Der Maaten and Geoffrey Hinton. Visualizing non-metric similarities in multiple maps. *Machine Learning*, 87(November):33–55, 2012.

[62] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *The Journal of Machine Learning Research*, 15:3221–3245, 2014.

[63] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. *Advances in neural information processing systems*, pages 833–840, 2002.

[64] Manik Varma and Andrew Zisserman. A statistical approach to texture classification from single images. *International Journal of Computer Vision*, 62:61–81, 2005.

[65] Andrea Tacchetti, Pavan K Mallapragada, Matteo Santoro, and Lorenzo Rosasco. GURLS: a Least Squares Library for Supervised Learning. (1):1–5, 2013.

[66] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalanditis, Li-Jia Li, David a. Shamma, Michael S. Bernstein, Li Fei-Fei, Yannis Kalantidis, Li-Jia Li, David a. Shamma, Michael S. Bernstein, and Fei-Fei Li. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *Arxiv*, page 44, 2016.

[67] James Jerome Gibson. The Theory of Affordances, 1977.

[68] E. Sahin, M. Cakmak, M. R. Dogar, E. Ugur, and G. Ucoluk. To Afford or Not to Afford: A New Formalization of Affordances Toward Affordance-Based Robot Control. *Adaptive Behavior*, 15:447–472, 2007.

[69] D.A. Norman. *The Psychology of Everyday Things*. The Psychology of Everyday Things. Basic Books, 1988.

[70] Frank Van Harmelen Deborah L. McGuinness. Owl web ontology language overview. *W3C recommendation 10.2004-03*, 2004:1–12, 2004.

[71] From SHIQ and RDF to OWL The Making of a Web Ontology Language. (0).

[72] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[73] Karinne Ramirez-amaro, Ewald Lutscher, Andreas Holzbach, and Gordon Cheng. iCub @ ICS-TUM : Semantic Reasoning , Constrained Manipulation and Humanoid Vision enable on the iCub. (June):2014, 2014.

[74] Karinne Ramirez-amaro, Eun-sol Kim, Jiseob Kim, Byoung-tak Zhang, Michael Beetz, and Gordon Cheng. Enhancing Human Action Recognition through Spatio-temporal Feature Learning and Semantic Rules. *IEEE-RAS International Conference on Humanoid Robots*, pages 456–461, 2013.

[75] Jiseob Kim Byoung-Tak Zhang Michael Beetz Gordon Cheng Karinne Ramirez Amaro, Eun-Sol Kim. Enhancing human action recognition through spatio-temporal feature learning and semantic rules. In *IEEE-RAS International Conference on Humanoid Robots*, Atlanta, USA, 2013. IEEE.

[76] Karinne Ramirez-Amaro, Michael Beetz, and Gordon Cheng. Understanding the intention of human activities through semantic perception: observation, understanding and execution on a humanoid robot. *Advanced Robotics*, 29(00):345–362, 2015.

[77] Gordon Cheng Karinne Ramirez-Amaro, Michael Beetz. Transferring skills to humanoid robots by extracting semantic representations from observations of human activities. *Artificial Intelligence*, 2015.

[78] E. Krause. Learning to recognize novel objects in one shot through human-robot interactions in natural language dialogues. *Artificial Intelligence*, 2014.

[79] Author Names Omitted and Anonymous Review. ROBOSHERLOCK: Unstructured Information Processing for Robot Perception. pages 1549–1556, 2015.

[80] Giorgio Metta, Lorenzo Natale, Francesco Nori, Giulio Sandini, David Vernon, Luciano Fadiga, Claes von Hofsten, Kerstin Rosander, Manuel Lopes, José Santos-Victor, Alexandre Bernardino, and Luis Montesano. The iCub humanoid robot: An open-systems platform for research in cognitive development. *Neural Networks*, 23:1125–1134, 2010.

[81] Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. YARP: yet another robot platform. *International Journal on Advanced Robotics Systems*, 3(1):43–48, 2006.

[82] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, Trevor Darrell, and U C Berkeley Eecs. Caffe: Convolutional architecture for fast feature embedding. 2014.

[83] Data Mining. The Elements of Statistical Learning. *The Mathematical Intelligencer*, 27:83–85, 2009.

[84] Moritz Tenorth and Michael Beetz. KNOWROB — Knowledge Processing for Autonomous Personal Robots.

[85] Moritz Tenorth, Dominik Jain, and Michael Beetz. Knowledge Representation for Cognitive Robots. *K{ü}nstliche Intelligenz*, 24:233–240, 2010.

[86] Moritz M Tenorth. Knowledge Processing for Autonomous Robots. page 225, 2011.

[87] Moritz Tenorth, Ulrich Klank, Dejan Pangercic, and Michael Beetz. Web-enabled Robots – Robots that use the Web as an Information Resource. *Robotics & Automation Magazine, IEEE*, 18:58–68, 2011.

[88] Mortiz Tenorth. Overview of the knowrob upper ontology. Website. http://www.knowrob.org/doc/knowrob_taxonomy, August 2016.

[89] *The importance of having data-sets.*, 2006.

[90] Yann LeCun. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

[91] Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. *... Science Department, University of Toronto, Tech. ...*, pages 1–60, 2009.

[92] Caltech 101 dataset. http://www.vision.caltech.edu/Image_Datasets/Caltech101/.

[93] AD. Perona P. Griffin, G. Holub. The caltech 256. caltech technical report. Technical report, Caltech, 2006.

[94] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010.

[95] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[96] Andrej Karpathy. What i learned from competing against a convnet on imagenet. http://karpathy.github.io/2014/09/02/what-i-learned-from-competing-against-a-convnet-on-imagenet/, September 2014.

[97] Intituto Italiano di Technologia. icubworld28 dataset. http://old.iit.it/projects/datasets.

[98] B Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison, 2009.

[99] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Computer Vision and Pattern Recognition*, 2016.