Universität Bamberg
Angewandte Informatik

Seminar KI: gestern, heute, morgen

A Critical Review of “Deep Learning for Detecting Robotic Grasps”

by

Ines Rieger

February 27, 2017

advised by
Professor Ute Schmid

Contents

This paper is a review of the paper of Lenz et al. (2015) and provides furthermore information about robotic grasping, dealing with multimodal input data and the two deep learning architectures convolutional network and neural network. Lenz et al. (2015) operate with a two-stage cascaded detection with two differently sized deep neural networks. They apply multimodal group regularization on the RGB-D input data and learn the weights with a sparse auto-encoder in a two-step learning process. Two different robots execute the grasps. The deep learning approach of Lenz et al. (2015) outperforms a machine learning approach on the same dataset, but advantages over other deep learning approaches are not apparent.

1 Introduction

Tackling the challenging problem of robotic grasping has evolved from treating it as a geometrical problem over to a data-driven problem. Lenz et al. (2015) were - to my best knowledge - the first to apply a deep neural network (DNN) to this problem. A deep learning approach has several advantages over a machine learning approach. The most important advantage is that in deep learning approaches the features don't need to be hand-engineered and novel objects can be grasped. As grasping is a basic and important feature for robots, it is only consequent to apply this approach on this task. This paper gives an overview of robotic grasping approaches and focuses especially on deep learning and the paper of Lenz et al. (2015). Lenz et al. (2015) used a two-step cascaded system with two DNNs. The output of the first network was re-evaluated in a second network, which computed the top single rectangle. Getting the right features for the network is most important and they used a two-step learning algorithm with a sparse auto-encoder (SAE) for it, applying unsupervised learning first and supervised learning second. They had RGB-D images for input and handled the multiple modalities with a structured regularization.

Deep learning was successfully applied in many areas such as speech recognition (Collobert et al., 2011; Hashimoto et al., 2016; Mikolov et al., 2011) and object recognition (Sohn et al., 2011; Szegedy et al., 2015; Sermanet et al., 2013b). Lenz et al. (2015), Eitel et al. (2015) and Levine et al. (2016) showed that deep learning can also be successfully applied to robotic grasping.

Concerning the structure of this paper, section two gives relevant background information on robotic grasping, multimodal input data and deep learning. The section first distinguishes the robotic grasping approaches in geometrical and data-driven approaches while giving an overview over the most important works. Then, it provides an overview over approaches with RGB-D data, where it focuses especially on approaches with deep learning. Finally, it explains the structure and mode of operation of DNNs and deep CNNs. In section three the implementation of Lenz et al. (2015) is described. It focuses on explaining the algorithms of the two-stage cascaded detection as well as of the two-phase weight learning and the regularization of the multimodal input data. Also, their results are discussed. Section four compares the approach of Lenz et al. (2015) to standard machine learning approaches and to CNN approaches.

2 Learning from Image Data

2.1 Robotic Grasping

For a successful robotic grasp, the best candidate grasp and the optimal gripper pose to approach this grasp needs to be computed. The algorithm has to deal with a wide variety of shapes and textures of the objects. The approaches which try to solve the domain of robotic grasping can be divided into two main categories, the geometrical approach and the data-driven approach.

Geometrical Approach The geometrical approach, although still an active research topic (Dogar et al., 2012; Weisz and Allen, 2012), is the earlier approach for robotic grasping in contrast to the data-driven approach. Following Shimoga (1996), the geometrical approach refers to methods, where force-closure grasps satisfy the constraints of the grasp synthesis. A force-closure grasp means that the robot fingers have gripped the object at certain places and are capable to balance the force exerted on the grasped object. The criteria of the grasp-synthesis are dexterity (configuration of grasping fingers), equilibrium (degree of hardness for squeezing), stability and dynamic behavior (degree of softness for squeezing for a certain task). It can be seen as a constrained optimization problem, because the algorithm is based on the criteria for the synthesis grasp and on the 3D model of the object (Ferrari and Canny, 1992; Weisz and Allen, 2012).

Data-driven Approach The data-driven approach or, as Shimoga (1996) refers to, the knowledge-based approach, means generally spoken the use of predefined rules. This extends also to the robotic finger configurations. The grasp experience, which leads to these rules, is usually gained through heuristics or simulation. The algorithms can be generalized in most times and are thus applicable to unknown objects. Machine learning and deep learning algorithms are used to realize these approaches.

One characteristic of machine learning approaches is that the features are mostly hand-engineered. Maitin-Shepard et al. (2010) for instance hand-engineered in their approach the features, so the robot could grasps a towel and manipulate it with re-grasps in such a way that he could fold it.

Some data-driven approaches were still close to the geometrical approaches. In these approaches, partially occluded objects were recognized in images (Glover et al., 2008) or the objects were in an unknown pose (Detry et al., 2009). But there were also machine learning approaches, where no 3D models were needed anymore. Saxena et al. (2006) did not build a 3D model, but their algorithm learned via supervised training from synthetic images to predict the best candidate grasp. Le et al. (2010) used a SVM model to rank all grasp candidates. They fed the top ten ranked grasps into a motion planner, which gave out the best grasp in a representation of multiple contact points, which simulated the way a human hand would grasp. But this representation had the drawback that the multiple contact points were not an exact representation of the 7-dimensional gripper configuration. The 7-dimensional gripper configuration tells for example the gripper width, its position and orientation. Therefore it lacked on robustness, which was further improved by Jiang et al. (2011).

Jiang et al. (2011) also ranked the candidate grasps. But instead of using a representation with multiple contact points they worked with a rectangle representation. They implemented a two-step learning process with the top ranked rectangle oriented in an image plane as an output. In the first step they applied features, which were fast to compute. In the second step they applied advanced features, which were more accurate but took longer to compute. The top ranked rectangles from the first step were the input for the second step, which had for the output the single top ranked rectangle. For the ranking they used a support vector machine (SVM) algorithm. The output rectangle from the second step was converted to the 7-dimensional gripper configuration to make the grasp feasible for the robotic gripper. Thereby, the two edges of the rectangle corresponded to the gripper pair.

Viola and Jones (2001) also used a multistage algorithm in the field of object detection. They first quickly analyzed the background of an image and then concentrated with more computational power and higher accuracy on regions, where the probability that it contained the object of interest was more likely.

These multistage algorithms enhance therefore the speed of the detection process, because it takes less computational power in the first step, where the whole data set is analyzed and use the more complex features in the second step on the highest ranked output data of the first step. This saves computational power. Lenz et al. (2015) were inspired by the multistage approaches and used a two-stage cascaded system with a rectangle representation.

The second data-driven method is deep learning. Deep learning refers to computational models, who operate with several layers on large data sets and categorize the input data. There are two types of deep learning, one operates with a deep neural network (DNN) and the other one operates with a deep convolutional neural network (CNN). As mentioned before, Lenz et al. (2015) implemented robotic grasping with a DNN. As far as I know, this was so far the first and only approach for robotic grasping with a DNN. There are also successful approaches for robotic grasping with deep CNNs. Some of them focus on large datasets (Levine et al., 2016; Pinto and Gupta, 2016) and others on dealing with different modalities (Eitel et al., 2015). In general, DNNs and CNNs are applied successfully in several areas. DNNs were used in the fields of speech recognition (Collobert et al., 2011; Hashimoto et al., 2016; Mikolov et al., 2011), face detection (Osadchy et al., 2007; Sermanet et al., 2013a) and pedestrian detection (Sermanet et al., 2013b). An implementation with deep CNNs was used in the fields of visual recognition (Sohn et al., 2011; Tompson et al., 2014; Szegedy et al., 2015) and also in speech recognition (Sainath et al., 2013).

2.2 Dealing with RGB-D Data

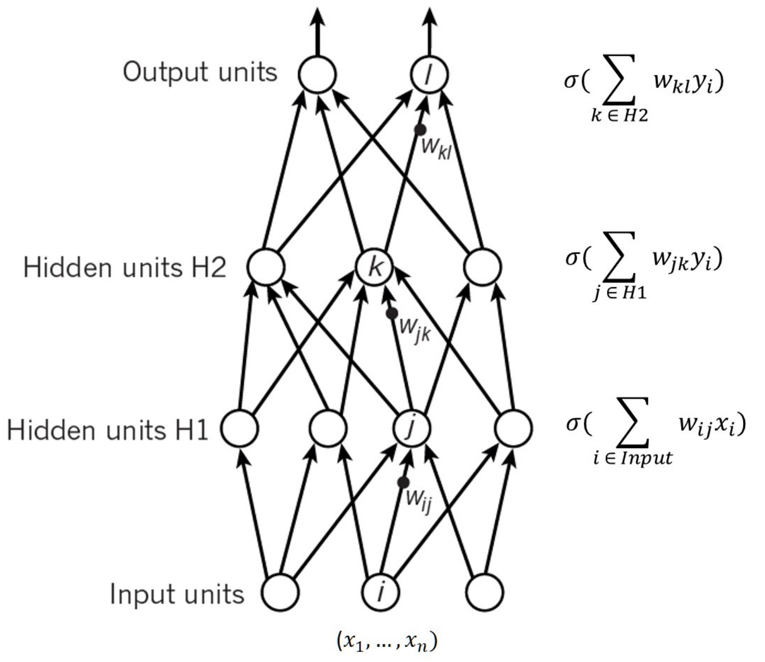
Dealing with RGB-D data means dealing with a color channel and a depth channel. Thus, an algorithm is needed to handle the multiple input data for feature learning. Other multiple input data besides RGB-D would be for instance audio and video data (Ngiam et al., 2011) or text and image data (Srivastava and Salakhutdinov, 2012). The machine learning approach of Jiang et al. (2011) in robotic grasping, with which I also later compare the approach of Lenz et al. (2015), also worked with multiple input data from RGB-D images to recognize the placement of the objects. They used the depth channel to divide the picture in spatial histogram features to recognize patterns of the object, for instance the stem of a martini glass.

RGB-D data is also used for various robotic applications in combination with deep learning, so called multimodal deep learning. Eitel et al. (2015) proposed for object recognition a deep CNN architecture to pre-process the two modalities of the RGB-D data separately in two streams. They fused the streams later consecutively together in a late fusion network. So, the two modalities were first trained seperately with a CNN and then later fused together and trained on a third CNN for fine-tuning. Socher et al. (2012) in contrast fused the two modalities already at the beginning together. They introduced a model based on a CNN and recurrent neural networks (RNN) for object classification on RGB-D images. They first used the CNN to learn low-level invariant features and then gave the output of the CNN to multiple RNNs to construct higher level features. For learning the features they concatenated the different modalities. Concatenating the different input modalities has been proven effective, where the modalities are more similar like in RGB-D data, according to Lenz et al. (2015). Lenz et al. (2015) therefore chose a similar approach with concatenating the different modalities.

2.3 Deep Learning (DNN and CNN)

Deep Neural Networks Neural networks were originally inspired by the brain, but differ in their computational structure from the biological learning system. According to Schmidhuber (2015), the first deep learning system of the type feedforward multilayer perceptron first arose around 1970. At the same time the concept of backpropagation, which means reducing the error through gradient descent, was originated. So although the basic concepts already existed, it was not until the computational power increased enough to process huge amount of data that deep neural network architectures could gain valuable results. This was around 2003, when the first successful DNNs arose. To differentiate a shallow neural network from a deep neural network, the hidden layers are counted. The amount of hidden layers, which characterize a shallow neural network, is not clearly defined throughout literature, but Delalleau and Bengio (2011) for instance set the threshold for a DNN at two layers.

Figure 1 shows the basic architecture of a DNN with an input layer, two hidden layers and an output layer. The following explanations of the DNN is based on the survey paper of LeCun et al. (2015). The example DNN in figure 1 is very small. Typically tens of hundreds of thousands of units (counted together from all layers) are used for



Output units
Hidden units H2
Hidden units H1
Input units
l
k
j
i
w_{kl}
w_{jk}
w_{ij}
$\sigma(\sum_{k \in H2} w_{kl} y_i)$
$\sigma(\sum_{j \in H1} w_{jk} y_i)$
$\sigma(\sum_{i \in Input} w_{ij} x_i)$
$(x_1, \dots, x_n)$

Figure 1: Deep Neural Network

object recognition in a DNN. The input values x_i represent real values and are usually renormalized to a value range, for instance from 0 to 1. The values in the input array can consist for instance of pixel intensity values of an image. The output is a categorization of the input image with the help of feature layers. The first layer normally finds low level features like the presence or absence of edges and locations in the image. The second layer then consists of more complex features like particular arrangements of edges. So, every unit detects a different feature of the input image. The computation is for every layer the same. For the first layer for instance every input x_i is multiplied with their respective weights w_{ij}. A non-linear function is applied to each unit to compute their output. In most cases this is a sigmoid function $\sigma(t) = 1/1 + e^{-t}$. Other non-linear functions used for neural networks are the $tanh(t)$ or the rectified linear unit (ReLu). All units from one layer are connected with all units from the next layer. So the total output $\sum_{i \in input} w_{ij} x_i$ of the layer H1 is also the input of the next layer H2. The output layer at the end of the network has for instance as an output the probabilities for one or, in this case, two categories. A threshold can be used to determine whether the category holds for the input image or not.

The challenge is to find the correct weights to produce the desired output. A popular method is the backpropagation algorithm, which consists of two steps, propagation of the error and updating the weights. The goal is to minimize the error. When the input vector is propagated to the output, a loss function is used to calculate the error for each

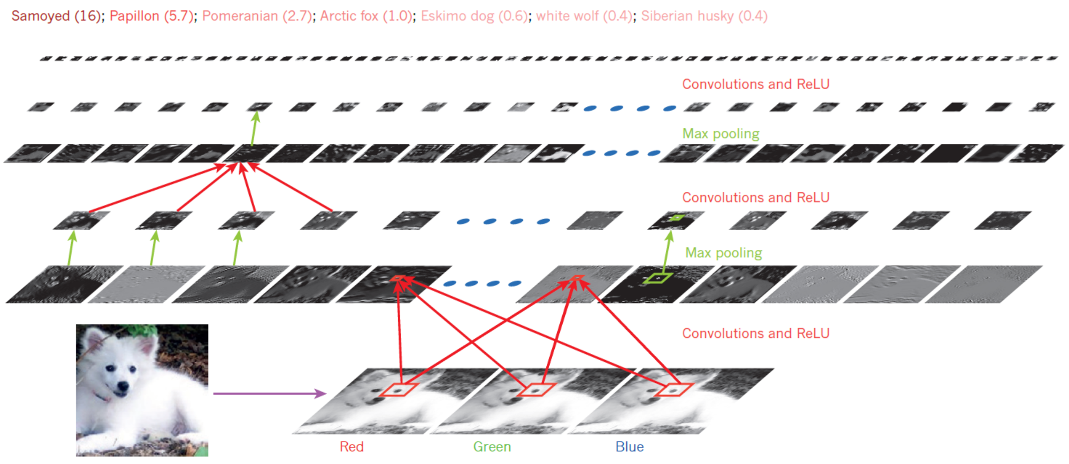

Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic fox (1.0); Eskimo dog (0.6); white wolf (0.4); Siberian husky (0.4)
Convolutions and ReLU
Max pooling
Convolutions and ReLU
Max pooling
Convolutions and ReLU
Red
Green
Blue

Figure 2: Deep Convolutional Neural Network (LeCun et al. (2015))

unit in the output layer. The error is the difference between the desired output and the actual output. These error values are, as the name backpropagation already suggests, then propagated backwards to the input layer. At each layer, the error derivative is computed over all unit outputs. With the error value the gradient of the loss function is computed. The gradient vector indicates for each weight, how the error would change, if a weight is de- or increased. The weights are then updated, but in the opposite direction of where the gradient points, because it points in the direction of the greatest increase of the error function. The backpropagation can be run repeatedly, but if executed too often, there is a risk of overfitting.

Backpropagation is a supervised learning method with labeled input data for the training phase and a guessed or random weight for initialization. There are also approaches with unsupervised learning procedures, which don't require labeled data. These procedures pre-learn the weights and the backpropagation is then only used for fine-tuning the weights. For pre-training Lenz et al. (2015) used a variant of a sparse auto-encoder and Hinton et al. (2006) used a restricted Boltzmann machine.

Deep Convolutional Neural Networks As introduced before, deep CNNs are applied to tasks like object detection such as pedestrian detection, object recognition such as face recognition and also to speech recognition. I introduce the concept of deep CNNs, because it is interesting in contrast to DNN, as they are both applied to robotic grasping. A deep CNN consists of several sets of convolutional layers and pooling layers and fully connected layers. The convolutional and pooling layers are inspired by visual neuroscience (Hubel and Wiesel, 1962). The following explanation is based on LeCun et al. (2015) and Nielsen (2015).

Figure 2 shows a deep CNN, where the input consists of a RGB picture of a Samoyed dog. The input image is split up in three images according to their color channels in red, green and blue. The network consists of sets of one convolutional layer and one

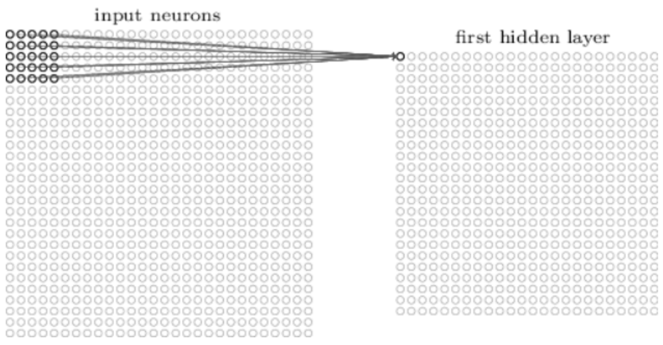


Figure 3: Convolutional Layer (Nielsen (2015))

pooling layer, which together reduce the pixel size of the image. If this set is repeated often, the CNN is called a deep CNN. The task of the convolutional layer is to detect local conjunctions of features. The image in the convolutional layer is organized in units, which each contain several pixels. As shown in figure 3, the field of pixels of for example the input layer is connected to one neuron of the next layer. This so called window slides pixel per pixel over the whole pixel matrix. For every pixel group, which is propagated to the next layer, the same weight is used. This is the case, because the same motif can appear in different parts of the image. Thus, the weight is called a shared weight and the input image is called a feature map. Over each local weighted sum from the pixel group a non-linear function such as ReLU (Rectified Linear Unit) is applied. The ReLu eliminates all negative values. The task of the pooling layer is to merge semantically similar features. It thereby merges neighbored units and reduces the dimension of representation. By doing that, the pooling layer converts then again in a convolutional layer. After the sets of convolutional layers and pooling layers, some deep CNNs add more convolutional layers without a pooling layer in between. At the end of the network, there are fully connected layers. This means, in contrast to the convolutional and pooling layers, here every unit is connected with every unit of the next layer. Using support vector machines or a softmax activation function, the fully connected layer classifies the output. In figure 2 the class “Samoyed” has the highest output value. This is correct, because the input image shows a Samoyed dog. For learning the weights in CNNs, backpropagation with propagating gradients back is used similar to the DNNs.

3 Implementation of Robotic Grasping with Deep Learning by Lenz et al. (2015)

Lenz et al. (2015) showed a DNN implementation to detect the single best grasp of an object in a RGB-D view. The best grasp was represented in a rectangle, which information’s were used by the robot to grasp the novel object. They used a two-step cascaded system with two DNNs. The overall idea was to take the top grasps of the smaller first network and re-evaluate them in the more complex second network, where the output is the single best grasp. They also presented a method to handle the multimodal input data from the RGB-D camera. Lenz et al. (2015) claim that their approach with using DNNs for detecting robotic grasps is the first to do so. Their research was in the year 2014 and to my best knowledge, there exists no further work in this area since then. The newer approaches with deep learning and robotic grasping (Levine et al., 2016; Pinto and Gupta, 2016) were realized with deep CNNs.

3.1 Algorithms

For the object detection Lenz et al. (2015) used a two-stage cascaded detection. A successful detection means to find a box, which tightly contains the desired object, in this case a grasp. The two-stage cascaded detection works as the following. At first, the system gets an image from the RGB-D camera the robot wears. From this, many possible grasps in rectangles are generated. Every RGB-D image in the rectangle is then split into a color, a depth and a surface normals image. Out of these three images the input vector for the first DNN is generated. The highest ranked rectangles from the first DNN are the input for the second DNN. For learning the weights they use a variant of a sparse auto-encoder (SAE) and for learning the features they apply a multimodal regularization method. They used an updated version of the Cornell grasping dataset for the supervised training. This dataset contains 1035 images of 280 graspable objects. Each image was annotated with several rectangles and it was indicated whether the feature in the rectangle was graspable or not.

3.1.1 Two-Stage Cascaded Detection

The two neural networks have the same architecture of layers, but come in different sizes. The first network makes a first selection of rectangles and has a size of 50 units, whereas the second network searches for the single best rectangle and has a size of 200 units. As figure 4 shows exemplarily for one network, the networks have both an input layer x, two hidden layers $h^{[1]}$ and $h^{[2]}$ with K_1 and K_2 sigmoidal units and an output layer $\hat{y}^{(t)} \in \{0,1\}$, which predicts, whether a rectangle $G^{(t)}$ is graspable or not. Each layer l has a set of weights $W^{(l)}$, which are multiplied with the output $h_i^{[l](t)}$ from the layer before or at the first hidden layer $h_j^{[1](t)}$ with the input values x_i . This means for instance that the sum over all units for the first hidden layer $h_j^{[1](t)}$ is $\sum_{i=1}^{N} x_i^{(t)} W_{i,j}^{[1]}$. Each unit computes their output with the help of the sigmoid function $\sigma(a) = 1/(1+exp(-a)$. This result is then the input of the connected unit from the next layer. For the result of the output layer, a logistic classifier is used on the outputs of the second hidden layer $h^{[2]}$. This logistic classifier predicts $P(\hat{y}^{(t)}|x^{(t)};\Theta)$, where $\Theta = \{W^{[1]}, W^{[2]}, W^{[3]}\}$ and $x^{(t)}$ are the inputs. So the binary prediction $\hat{y}^{(t)}$ depends on the weights and input values. The rectangles with the possible grasps are ranked due to the results of the network. The goal of the second network is to find the single best grasping rectangle G^* (1) with

$$h_j^{[1](t)} = \sigma\left(\sum_{i=1}^{N} x_i^{(t)} W_{i,j}^{[1]}\right)$$

$$h_j^{[2](t)} = \sigma\left(\sum_{i=1}^{K_1} h_i^{[1](t)} W_{i,j}^{[2]}\right)$$

$$P(\hat{y}^{(t)} = 1|x^{(t)};\Theta) = \sigma\left(\sum_{i=1}^{K_2} h_i^{[2](t)} W_i^{[3]}\right)$$

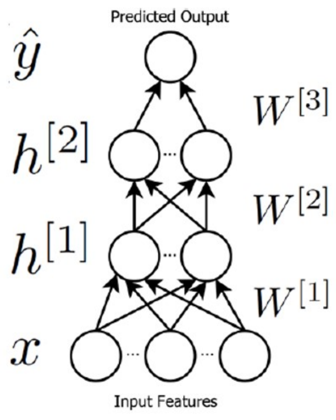


Figure 4: Deep Neural Network (Lenz et al. (2015))

the maximum grasping probability. $\phi(G)$ is the input representation for the respective rectangles G.

$$G^* = argmax_G P(\hat{y}^{(t)} = 1|\phi(G);\Theta) \tag{1}$$

3.1.2 Two-Phase Weight Learning

For learning features they used 24 x 24 pixel images as an input, where each of them represented one image in a rectangle. They extracted seven channels from these images. The first three channels represented the color space, the fourth channel was the depth and the last three channels were the surface normals. The surface normals were based on the depth channel. They also whitened the data and preserved the aspect ratio when they inserted the images into the network.

Regularization of Multimodal Data Input The regularization was applied while learning the weights in the two-phase learning approach with the SAE, which is described next. Regularization protects against overfitting with penalizing large weights and trying to reduce them to zero. This diminishes the number of modalities, which are used for one feature. The system is thus prevented to learn weak correlations between the modalities and therefore the generalization ability improves. Through experimenting Lenz et al. (2015) found out that learning low-level correlations can lead to more robust features for RGB-D data. This means, the three input modalities are simply concatenated in the first layer (fig. 5) instead of training them separately first. For the structured multimodal regularization, they used the following function:

$$f(W) = \sum_{j=1}^{K}\sum_{r=1}^{R} II\{(max_i S_{r,i}|W_{i,j}|) > 0\} \tag{2}$$

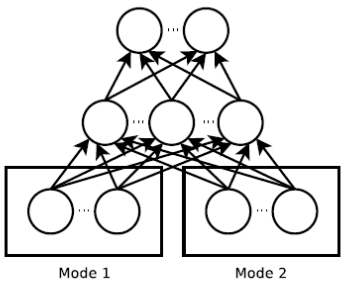


Figure 5: Simple Concatenation (Lenz et al. (2015))

The function gives back a weight vector over all input modalities for one feature. The output features correspond either to a positive grasp or to a negative grasp. The matrix S shows all R distinct modalities. Every element in $S_{r,i}$ indicates a unit x_i in a particular modality r. So, a weight vector is created by considering for every modality every unit. The max norm max sets an upper bound to the weight and if a weight is larger than max, all weights of this unit are renormalized by division. On top of the max norm the L_0 norm is applied. The L_0 norm ensures that the network has more non-zero weight values than zero weight values by indicating that the argument is true or not.

Lenz et al. (2015) described the effects of the structured regularization on the feature generation. The structured regularization diminished for example in many features the weights for the U and V color channels by giving them zero weight. This means that the system is more robust for objects, which are colored differently. The depth channel was very important and many features had a non-zero weight for depth. For example, raised regions such as handles were correlated as a positive feature and lowland was correlated as a negative feature. But overall they achieved the highest accuracy by using all input modalities.

Two-Phase Learning Approach For learning the weights, Lenz et al. (2015) used a two-phase learning approach and included the regularization function described before. For the first phase they pre-trained the weights $W^{[1]}$ and $W^{[2]}$ with unsupervised feature learning and initialized them. Pre-training with unlabeled data reduces the risk of overfitting and means generally spoken finding clusters (similar objects) in the data. For pre-training Lenz et al. (2015) used a variant of a sparse auto-encoder (SAE) (figure 6) with one hidden layer h. This means that no features had to be engineered manually. As described by Ng (2011), an auto-encoder applies backpropagation and the target values of the layer $\hat{x}$ are the same as in the input layer x. So, the auto-encoder tries to reconstruct the input x by finding correlations between the input features. The aspect of a sparsity constraint comes into play with large layers of hidden units. The sparsity ensures that structures will be found in data when networks with a large number of units are used. Lenz et al. (2015) didn't specify the size of the hidden layer in their sparse auto-encoder, but I assume, because of the explanation of Ng (2011), that it is a rather large one. Lenz et al. (2015) described that in this first phase of unsupervised

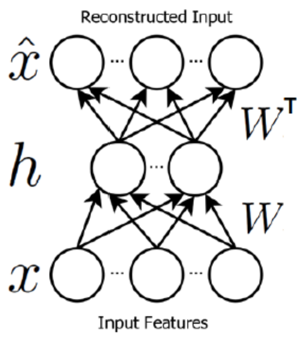


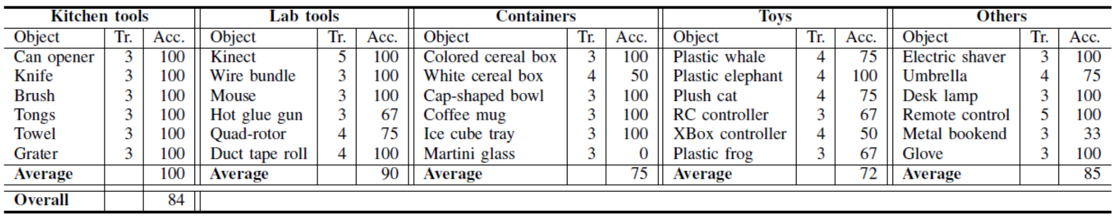
Figure 6: Sparse Auto-Encoder (SAE) (Lenz et al. (2015))

learning, they first use the algorithm to initialize $W^{[1]}$ and hereby reconstruct x in the sparse auto-encoder. Then, they use the learned weight $W^{[1]}$ to generate outputs from the input layer x. With these outputs, respectively inputs for the second layer $h^{[2]}$, they learn $W^{[2]}$ to reconstruct $h^{[1]}$. After this unsupervised phase, they use the supervised phase to fine-tune the weights $W^{[1]}$ and $W^{[2]}$ and to learn the weight $W^{[3]}$. In the supervised phase, they work with labeled data, where the rectangles are annotated as graspable or non-graspable.

3.2 Results

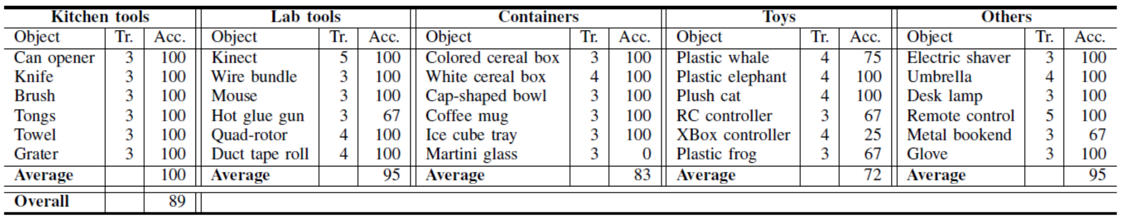
To perform the experiment Lenz et al. (2015) used the two robots Baxter Research Robot ("Yogi") and PR2 Robot ("Kodiak") with parallel grippers. The two robots had different features. Yogis arms had a maximum reach of 104cm and interchangable grippers, so they used two different grippers for him. Kodiaks arms had a maximum reach of 1m and had grippers with a maximum width of 8cm. His grippers were fixed, but he could close them completely from the widest span. They both had a Kinect sensor mounted on their head, which could deliver RGB-D images. The robots had a set of 35 objects to grasp from and from which most of the objects had not been in the training data set, so they were completely novel for the trained network. They had to sort out 5 objects out of the 35 objects, because the robots grippers could not grasp them. This was for example the hammer, who was too imbalanced and slipped out of the gripper. The results in the table (fig. 7, fig. 8) are after 100 trials.

Overall, Kodiak was able to grasp 89% of the objects successfully and Yogi successfully grasped 84% of the objects. So Kodiak was more successful, but in Yogi's case, the algorithm detected in 8% of the trials a valid grasp, which Yogi could not execute due to his physical limitations. For example, as is also indicated by the low accuracy for this object in the table (fig. 7), Yogi had problems to grasp the region between the handle and body of an umbrella, because he couldn't close his gripper completely. In contrast, Kodiak achieved 100% percent accuracy by grasping the umbrella. This was the same



Kitchen tools			Lab tools			Containers			Toys			Others		
Object	Tr.	Acc.	Object	Tr.	Acc.	Object	Tr.	Acc.	Object	Tr.	Acc.	Object	Tr.	Acc.
Can opener	3	100	Kinect	5	100	Colored cereal box	3	100	Plastic whale	4	75	Electric shaver	3	100
Knife	3	100	Wire bundle	3	100	White cereal box	4	50	Plastic elephant	4	100	Umbrella	4	75
Brush	3	100	Mouse	3	100	Cap-shaped bowl	3	100	Plush cat	4	75	Desk lamp	3	100
Tongs	3	100	Hot glue gun	3	67	Coffee mug	3	100	RC controller	3	67	Remote control	5	100
Towel	3	100	Quad-rotor	4	75	Ice cube tray	3	100	XBox controller	4	50	Metal bookend	3	33
Grater	3	100	Duct tape roll	4	100	Martini glass	3	0	Plastic frog	3	67	Glove	3	100
Average		100	**Average**		90	**Average**		75	**Average**		72	**Average**		85
Overall		84												

Figure 7: Results for experiment for Yogi (Lenz et al. (2015))



Kitchen tools			Lab tools			Containers			Toys			Others		
Object	Tr.	Acc.	Object	Tr.	Acc.	Object	Tr.	Acc.	Object	Tr.	Acc.	Object	Tr.	Acc.
Can opener	3	100	Kinect	5	100	Colored cereal box	3	100	Plastic whale	4	75	Electric shaver	3	100
Knife	3	100	Wire bundle	3	100	White cereal box	4	100	Plastic elephant	4	100	Umbrella	4	100
Brush	3	100	Mouse	3	100	Cap-shaped bowl	3	100	Plush cat	4	100	Desk lamp	3	100
Tongs	3	100	Hot glue gun	3	67	Coffee mug	3	100	RC controller	3	67	Remote control	5	100
Towel	3	100	Quad-rotor	4	100	Ice cube tray	3	100	XBox controller	4	25	Metal bookend	3	67
Grater	3	100	Duct tape roll	4	100	Martini glass	3	0	Plastic frog	3	67	Glove	3	100
Average		100	**Average**		95	**Average**		83	**Average**		72	**Average**		95
Overall		89												

Figure 8: Results for experiment for Kodiak (Lenz et al. (2015))

problem with the plush cat, where the single best grasp was a thin leg, which Yogi could grasp only with difficulty. The advantage from Kodiak was therefore that he could close his grippers completely from the widest span and that he had a larger gripper force. The larger gripper force of Kodiak made the difference with the white cereal box. The white cereal box was hard to detect via the algorithm, because the background for all objects in the learning dataset was white. But Kodiak was nevertheless able execute all times the not ideal grasp, because he applied larger gripper force and thereby crashed the box a bit. A problem for both robots was the martini glass. Because of it's glossiness the camera failed to detect the depth. Difficult objects, which where grasped successfully by both robots were for instance a crumpled cloth towel, the plastic baseball cap and a coffee mug.

Lenz et al. (2015) mentoined in their paper also some criticism on their approach. Their approach is not applicable to robots with multi-fingered hands, because the output of the system is only a gripper pose. Also the algorithm doesn't take into account that some objects have preferable areas to grasp like a hot glue gun by the barrel. Another point was to have different background colors for the images in the test dataset.

4 Comparing DNN with Standard Machine Learning and CNN for the Robot Grasping Domain

Comparing DNN with Standard Machine Learning Approach As noted before, Jiang et al. (2011) used the same dataset as Lenz et al. (2015). They both implemented a two-step learning algorithm based on RGB-D images as input, which returned the single best grasp. For learning the weights Jiang et al. (2011) first manually labeled the rectangles with ranks from 1 to 3. Then, they used a support vector machine (SVM) algorithm to learn the weights of the possible grasps.

Lenz et al. (2015) compared the results of both approaches by the detection accuracy for rectangles. The compared accuracy was measured by using the top-ranked rectangle G and the ground-truth rectangle G^*, using only the rectangles from G^*, which have a lower orientation error from G than $30°$. The remaining rectangles were evaluated by their intersection divided by union. Lenz et al. (2015) classified a prediction as correct when it covered 25% of the evaluated space. According to these results, Lenz et al. (2015) had a 13% higher accuracy detecting results in the dataset with using the SAE and a structured two-step regularization for feature learning than Jiang et al. (2011) with a machine learning approach.

Besides comparing their approach with a machine learning approach, Lenz et al. (2015) also compared it with different deep learning strategies. For instance, they applied the SAE with only an L_1 regularization and the SAE with applying the structured regularization on the first cycle. This still outperformed the machine learning approach of Jiang et al. (2011). But applying SAE with no regularization has for an output an almost 30% lower accuracy than the described machine learning approach. This shows the advantages of non-hand-engineered features but also the importance of taking the information's of the different modalities into account and regularize them. The accuracy in the approaches of Lenz et al. (2015) and Jiang et al. (2011) were the highest when all input modalities were used. Lenz et al. (2015) used as different modalities color, depth and surface normals. Jiang et al. (2011) proved in their paper to have the highest accuracy when using not only the color, but also depth.

In conclusion, considering these two approaches, using the depth channel and using a two-step approach clearly enhanced the accuracy. Using in addition a DNN enhanced the accuracy further. So, Lenz et al. (2015) picked the promising features of the machine learning algorithms and added a DNN for further improvement.

Comparing DNN with CNN Eitel et al. (2015) proposed an object recognition approach for robotic grasping by using a deep CNN. They also used RGB-D input images and a two-step approach like Jiang et al. (2011) and Lenz et al. (2015). In the first step, Eitel et al. (2015) used two deep CNNs to process the two input modalities color and depth seperately. In the second step, they fuse the two CNNs into a third CNN together. In their approach it is crucial to treat the modalities differently at first. Other approaches with a deep CNN and multiple modalities for input did not pre-process the modalities in different CNNs and hence were not as successful (Schwarz et al., 2015; Socher et al., 2012). So, considering these two approaches, treating the modalities differently and making a late fusion enhances the success of deep CNNs. Working with different modalities as an input both enhanced also the accuracy of the DNN approach and the machine learning approach. But there are also approaches with deep CNNs, which didn't focus on different modalities but on large datasets. Levine et al. (2016) for instance used visual feedback to learn robotic grasping. They collected 800,000 grasp attempts to train their network. For the images they used a monocular vision, where the depth perception is limited. Pinto and Gupta (2016) also collected 50,000 grasp attempts for training and didn't use a depth sensor.

It is difficult to say, if DNNs or deep CNNs work better on the task of robotic grasping. The more recent approaches (Pinto and Gupta, 2016; Levine et al., 2016; Eitel et al., 2015; Redmon and Angelova, 2015) in this field worked with deep CNNs. Overall, in object recognition and object detection, the deep CNNs dominate the field at the moment (Szegedy et al., 2015; Tompson et al., 2014; LeCun et al., 2015). But the frequency of using one approach over another has increased and decreased before (Schmidhuber, 2015). It is apparent that certain patterns are used in each of the approaches such as dealing with different modalities und using multi-step approaches.

LeCun et al. (2015) proposed a combination of a deep CNN with a recurrent neural network (RNN), where the deep CNN is applied before the RNN. The RNN would get as an input a specific image extract from the deep CNN to recognize the object in this area. So, the area of the object is already detected by the deep CNN, but the RNN categorizes it. An unfolded RNN can also be seen as a very deep feedforward network, where all the layers share the same weight.

5 Conclusion

This paper gave an overview over the most important approaches in the fields of machine learning and deep learning regarding the task of robotic grasping. Thereby, the approach of Lenz et al. (2015) was the center of this paper, grouping the related subjects such as robotic grasping, deep learning with DNNs and CNNs and learning from multimodal data around. Important aspects of Lenz et al. (2015) approach were the usage of a two-step learning algorithm and the multimodal group regularization in combination with a two-step feature learning algorithm with an unsupervised step at first. As pointed out, the main approaches such as multistep algorithms and dealing with multimodal input data were also applied successfully in other machine learning and deep CNN approaches in the field of robotic grasping.

In a comparison, the approach of Lenz et al. (2015) was more successful than the machine learning approach of Jiang et al. (2011) on the same dataset. An interesting question is the comparison of DNNs and CNNs. At the moment, it seems that both deep learning approaches are successfully applicable for object recognition and detection. A direct comparison of a DNN and deep CNN on the same dataset in the area of object detection could shed some light on whether to prefer DNNs or deep CNNs. A combination of both, as LeCun et al. (2015) proposed, would also be thinkable for future implementations. Comparing deep learning with machine learning in this area, it became clear that the huge advantage of the deep learning algorithms is the automatic feature learning. Therefore, deep learning approaches may arise more often in the future for object detection, also probably with focusing on large datasets to take full advantage of automatic learning.

References

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12, Aug (2011), 2493–2537.

Olivier Delalleau and Yoshua Bengio. 2011. Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*. 666–674.

Renaud Detry, Emre Baseski, Mila Popovic, Younes Touati, N Kruger, Oliver Kroemer, Jan Peters, and Justus Piater. 2009. Learning object-specific grasp affordance densities. In *Development and Learning, 2009. ICDL 2009. IEEE 8th International Conference on*. IEEE, 1–7.

M Dogar, Kaijen Hsiao, Matei Ciocarlie, and Siddhartha Srinivasa. 2012. Physics-based grasp planning through clutter. (2012).

Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. 2015. Multimodal deep learning for robust rgb-d object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 681–687.

Carlo Ferrari and John Canny. 1992. Planning optimal grasps. In *Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on*. IEEE, 2290–2295.

Jared Glover, Daniela Rus, and Nicholas Roy. 2008. Probabilistic models of object geometry for grasp planning. *Proceedings of Robotics: Science and Systems IV, Zurich, Switzerland* (2008), 278–285.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2016. A Joint Many-Task Model: Growing a Neural Network for Multiple NLP Tasks. *arXiv preprint arXiv:1611.01587* (2016).

Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural computation* 18, 7 (2006), 1527–1554.

David H Hubel and Torsten N Wiesel. 1962. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology* 160, 1 (1962), 106–154.

Yun Jiang, Stephen Moseson, and Ashutosh Saxena. 2011. Efficient grasping from rgbd images: Learning using a new rectangle representation. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 3304–3311.

Quoc V Le, David Kamm, Arda F Kara, and Andrew Y Ng. 2010. Learning to grasp objects with multiple contact points. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 5062–5069.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

Ian Lenz, Honglak Lee, and Ashutosh Saxena. 2015. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research* 34, 4-5 (2015), 705–724.

Sergey Levine, Peter Pastor, Alex Krizhevsky, and Deirdre Quillen. 2016. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *arXiv preprint arXiv:1603.02199* (2016).

Jeremy Maitin-Shepard, Marco Cusumano-Towner, Jinna Lei, and Pieter Abbeel. 2010. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on.* IEEE, 2308–2315.

Tomáš Mikolov, Anoop Deoras, Daniel Povey, Lukáš Burget, and Jan Černockỳ. 2011. Strategies for training large scale neural network language models. In *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on.* IEEE, 196–201.

Andrew Ng. 2011. Sparse autoencoder. *CS294A Lecture notes* 72, 2011 (2011), 1–19.

Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. 2011. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11).* 689–696.

Michael A Nielsen. 2015. Neural networks and deep learning. *URL: http://neuralnetworksanddeeplearning.com/(visited: 27.02. 2017)* (2015).

Margarita Osadchy, Yann Le Cun, and Matthew L Miller. 2007. Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research* 8, May (2007), 1197–1215.

Lerrel Pinto and Abhinav Gupta. 2016. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on.* IEEE, 3406–3413.

Joseph Redmon and Anelia Angelova. 2015. Real-time grasp detection using convolutional neural networks. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on.* IEEE, 1316–1322.

Tara N Sainath, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran. 2013. Deep convolutional neural networks for LVCSR. In *Acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on.* IEEE, 8614–8618.

Ashutosh Saxena, Justin Driemeyer, Justin Kearns, and Andrew Y Ng. 2006. Robotic grasping of novel objects. In *Proceedings of the 19th International Conference on Neural Information Processing Systems.* MIT Press, 1209–1216.

Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural networks* 61 (2015), 85–117.

Max Schwarz, Hannes Schulz, and Sven Behnke. 2015. RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 1329–1335.

Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. 2013a. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229* (2013).

Pierre Sermanet, Koray Kavukcuoglu, Soumith Chintala, and Yann LeCun. 2013b. Pedestrian detection with unsupervised multi-stage feature learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3626–3633.

Karun B Shimoga. 1996. Robot grasp synthesis algorithms: A survey. *The International Journal of Robotics Research* 15, 3 (1996), 230–266.

Richard Socher, Brody Huval, Bharath Putta Bath, Christopher D Manning, and Andrew Y Ng. 2012. Convolutional-Recursive Deep Learning for 3D Object Classification.. In *NIPS*, Vol. 3. 8.

Kihyuk Sohn, Dae Yon Jung, Honglak Lee, and Alfred O Hero. 2011. Efficient learning of sparse, distributed, convolutional feature representations for object recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2643–2650.

Nitish Srivastava and Ruslan R Salakhutdinov. 2012. Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*. 2222–2230.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1–9.

Jonathan J Tompson, Arjun Jain, Yann LeCun, and Christoph Bregler. 2014. Joint training of a convolutional network and a graphical model for human pose estimation. In *Advances in neural information processing systems*. 1799–1807.

Paul Viola and Michael Jones. 2001. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, Vol. 1. IEEE, I–I.

Jonathan Weisz and Peter K Allen. 2012. Pose error robust grasping from contact wrench space metrics. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 557–562.