Deep Learning in Robotics: Survey on Model Structures and Training Strategies

Artúr István Károly, *Member, IEEE*, Péter Galambos, *Member, IEEE*, József Kuti, *Member, IEEE*, and Imre J. Rudas, *Fellow, IEEE*

***Abstract*—The ever-increasing complexity of robot applications induces the need for methods to approach problems with no (viable) analytical solution. Deep learning (DL) provides a set of tools to address this kind of problems. This survey presents a categorization of the major challenges in robotics that leverage DL technologies and introduces representative examples of successful solutions for the described problems. We also consider the question when and whether to use modular, monolithic models or end-to-end DL, in order to provide a guideline for the selection of the correct model structure and training strategy. By doing so, the current role and adaptability of different techniques at different hierarchical levels of a robot-application can be highlighted, thus providing a well-structured basis to assist future approaches.**

***Index Terms*—Deep learning (DL), machine learning (ML), manipulators, mobile robots, neural networks, robot control, robot learning.**

I. Introduction

COMPUTERS can easily solve formal problems that are demanding for humans. However, the increasing need for adaptive systems requires the solution of tasks that are hard to formulate, but can be easily solved by humans, such as the recognition and manipulation of objects. In order to perform such tasks, a certain complex knowledge of the environment is inevitable. The automatic extraction of the required knowledge is called machine learning (ML). The way the data is presented to the ML system, heavily influences how well the extracted knowledge represents the given problem. The ML approaches that also perform feature extraction, using multiple hierarchical artificial neural network layers, are referred to as deep learning (DL) [1], [2].

The theoretical background of DL had been introduced for a long time, when it finally gained widespread popularity, i.a., thanks to the winner entry of the ImageNet Challenge 2012 (LSVRC-2012) [3], a deep convolutional neural network (CNN) proposed by Krizhevsky, Sutskever, and Hinton (later referred to as AlexNet). This model was able to significantly outperform the previous state-of-the-art approaches as well as other contestants, achieving a top-5 test error rate of 15.3%, which was more than 10% lower than the second best entry. Since then, DL has been successfully applied for several use cases, such as speech recognition [4]–[6], image processing [7]–[10], natural language processing [11]–[13], sentiment analysis, recommendation systems [14], [15], etc., and large companies like Google, Facebook, Amazon, IBM, etc., have also founded their own DL research teams.

During its development, DL brought innovations in various aspects of robotics as well. A general review on the frequently used DL methods for robotics can be found in [16], while other surveys cover the most outstanding results with sharper focus: Robot control through reinforcement learning (RL) [17], [18], robotic manipulation and grasping [18], [19], mobile robot navigation [19], and transfer learning for robotics [19], [20] to mention a few topics. Even though DL-based solutions can be utilized in such diverse set of problems, they suffer from the drawbacks of unpredictability and high computational complexity. In safety critical systems, such as self-driving cars and industrial robots DL methods are never used on their own and their output is always treated with uncertainty. As a response to that, such DL methods are tested against adversarial attacks and on benchmarks that tell about their robustness. As a response to high computational complexity and time-consuming training process, alternative model architectures, and corresponding training strategies are introduced, such as random vector functional link neural networks (RVFLNNs), proposed by Pao *et al.* [21], [22], which utilize a novel training strategy for a shallow network architecture to avoid the time consuming training process that is typical in DL systems. Based on the RVFLNN method Chen and Liu [23], [24] also introduced broad learning systems, that offer an incremental training strategy for the fast adaptation and retraining of such models. To overcome these problems, new approaches in the field of DL were also proposed, which usually leverage the modular nature of DL models with the help of an appropriate model structure and/or training strategy [25]. Apart from the training process, the collection and preparation of the data for DL also requires a huge amount of resources, especially if it is done manually. This issue is even more relevant in robotics, where in most cases the data collection can only be done by

Manuscript received January 26, 2020; accepted June 12, 2020. Date of publication September 17, 2020; date of current version January 12, 2021. The work of Artúr István Károly was supported by the Hungarian State and the European Union under Grant EFOP- 3.6.1-16-2016-00010 Project. The work of Péter Galambos and József Kuti was supported by the Hungarian State and the European Union under Grant 2019-1-3-1-KK-2019-00007 Project. This article was recommended by Associate Editor H.-X. Li. *(Corresponding author: Péter Galambos.)*

The authors are with the Antal Bejczy Center for Intelligent Robotics, Óbuda University, 1034 Budapest, Hungary (e-mail: peter.galambos @irob.uni-obuda.hu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSMC.2020.3018325

performing actions on an actual robot. This type of data collection can take up to months of robot hours with multiple robots and comes with great costs accordingly [26]. To reduce the amount of (labeled) data needed for the training, new DL approaches leverage unsupervised/semi-supervised methods and transfer learning [27]–[29], and attempt to decrease the resource needs by doing the data collection (mostly) in simulation instead of reality [30], [31]. In this survey, we provide a wide angle review, from the aspect of the utilized structures and training strategies for DL models in robotics. We elaborate this topic by considering the major challenges in robotics and the related DL solutions. Accordingly, a structured overview of the problems can be seen in Fig. 1 arranging the landscape along the three major categories: 1) perception; 2) motion; and 3) knowledge adaptation. This categorization is the result of the following considerations.

1) Common characteristics of any robot-application [excluding Human–Robot interaction (HRI)] is that the operation is manifested in motion and manipulation.
2) In order to plan desirable motion, the robot needs to have reliable knowledge about its environment that is gained through perception capabilities.
3) Having successful DL-based solutions concerning perception and motion in specific tasks does not mean that the obtained knowledge can be directly utilized in other problems. To avoid or at least reduce the tremendous effort of training from scratch, the general knowledge has to be extracted from the experience of former solutions and transferred to new ones.

Along these points, the three major challenges can be organized in a hierarchical structure (Fig. 1), because the motion or manipulation is carried out based on the results of the perception process, and for the adaptation of the extracted knowledge an already existing solution for a similar problem is necessary. The category of motion is further divided both vertically and horizontally. The vertical separation is due to the significantly different tasks of robotic manipulation and mobile robotics, while the horizontal arrangement shows the separation of the planning and the control of the motion. In our categorization, we consider perception related challenges as those, that can be fully independent from the robot, so these models do not incorporate any explicit or implicit knowledge on the robot kinematics and controls. Thus, for example, a model for grasp prediction that incorporates the information about the gripper structure is not considered as solution for a perception task, but a solution for planning. The models that also make the robot perform actions are considered to be solutions for control.

The problems highlighted in the different levels of the hierarchy are not to be considered as subsets of the specific challenge. For example, HRI is not a subset of perception, but a larger field that involves perception. These problems are presented in the hierarchy to show, what kind of tasks may be approached by DL solutions, and where the corresponding DL solutions are located in the hierarchy. Also, there are analytical methods for the solutions of most of these problems, but this article analyzes the challenges from the aspect of DL. The most outstanding practical advantage of DL methods over analytical ones is in vision systems, since DL solutions

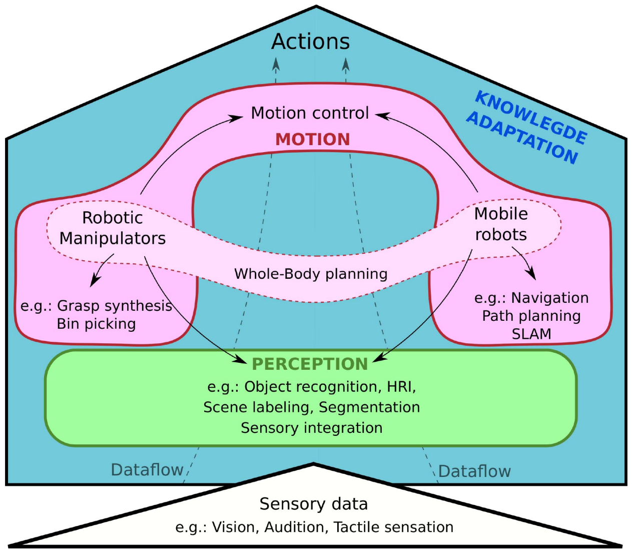


Fig. 1. Major challenges in DL for robotics.

for object detection and localization do not require expensive equipment, are quite robust to environmental variations, and they also save a lot of time, that would be spent on the manual design of visual feature detectors otherwise. An other area that is gaining more-and-more attention is grasp synthesis with DL.

The set of tools provided by DL can be applied at any level of the hierarchy shown in Fig. 1. Furthermore, the scope of a DL application can range from a single subtask to the whole hierarchy. When approaching a robot application with a DL solution, one has to decide which part of the application the DL model should be responsible for, and what strategies can be used for its training. In this survey, we aim to support these decisions by introducing successful solutions that can be applied at different levels of the presented hierarchy, highlighting the advantages and disadvantages of using DL at different scopes and by providing a systematic depiction of utilizing different training strategies, such as end-to-end or modular approaches.

II. Current Hard Challenges in Robotics

We consider the challenges of modern robot applications that usually benefit from DL solutions. The challenges are organized according to the scheme of Fig. 1, which helps us present the modular aspect of such applications.

A. Perception

In case of intelligent robot systems, high-level perception skills, such as the recognition of scenes and objects, localization, and interaction with humans are often required. These tasks are usually approached with the use of visual, depth, audio, and tactile information. In case of multimodal data, the challenge of sensory integration had to be addressed as well.

1) Scene and Object Recognition, Localization: In order to perform tasks in robot applications, the information of what kind of objects are there in the environment and where those objects are, is necessary. In real-world systems, this information is often not provided beforehand. For example, in a bin-picking scenario the position and orientation of the objects to be manipulated is unknown. Similarly in a simultaneous localization and mapping (SLAM) application, the map of the environment and thus the location of the mobile robot and the obstacles are not known *a-priori*. For acquiring this kind of information, high-level perception capabilities are utilized. DL methods can work well with data structures like images and depth data, using mostly the CNN structure [7].

The recognition of scenes and objects, as well as the localization of objects are mostly performed through visual information [32]–[38]. Apart from that, depth, tactile, and audio modalities can also be utilized. While tactile information is particularly used for object recognition in manipulation [39]–[42], audio can be used for identifying directions and locations [43]–[45]. Depth data may be used by both robot manipulators or mobile robots [32], [46], [47].

2) Human–Robot Interaction: The purpose of HRI is to endow robots with social skills, so they can interact with people. To do so, the robot has to infer the intentions of humans and perform appropriate actions accordingly. However, the interpretation of human social behavior is a very complex task, for which hand-crafted solutions are extremely hard to formulate. That is why, the perception of human intentions is usually done with the help of an ML method like DL [33], [48].

3) Sensory Integration: Data from a single modality may not be sufficient for all kinds of problems. For example, in case of object detection, apart from visual information depth data may also be utilized. The use of multiple sources of information gives a richer representation of the environment, it introduces redundancy to the system and reduces uncertainty. However, for the proper composition of the available information from various data sources, an appropriate sensory integration should be performed [49], [50].

Due to the uncertainties and conflicting information, it is hard to decide which pieces of information should contribute to the decision making process to what extent. This is especially true in case of multimodal data. DL methods are able to learn a high-level common embedding of the inputs with different modalities. The DL models for sensory integration usually process different modalities separately in different data streams, until sufficiently high-level features are extracted to fuse the information [32], [39], [42], [51]. The best approach to choose the abstraction level (a layer of the DL model) at which the information is fused is application specific, so it may as well be learned with an appropriate DL model [52].

B. Motion

A common requirement for intelligent robot systems is to flexibly adapt to novel tasks and environments, while being reliable and computationally efficient. Conventional analytic solutions do not scale well for motion planning problems with high dimensionality, so for these tasks DL-based solutions are also sought [47], [53]–[58].

The tasks for robot motion that are often approached with DL methods are the detection of the target of the motion (grasp detection), path planning, and trajectory planning in dynamic environments, SLAM, learning a model for predictive control, and learning to control unconventional robot architectures.

1) Grasp Detection: For the planning of the motion, a target should be specified beforehand. However, in some applications the target may not be known *a priori* due to the requirement for flexibility. For example, if a robotic manipulator is required to grasp unknown objects, these objects have to be detected and their pose, relative to the manipulator, have to be inferred [53]–[56]. The output of the grasp detection process is usually the proposed pose of the end effector of the manipulator that yields the highest probability of a successful grasp. There are analytic solutions for grasping in the presence of uncertainty, such as detection and pose estimation of known objects or the detection of primitive shapes, but these solutions are time-consuming and hard to formulate, according to Lenz *et al.* [55]. So DL methods are used to deal with novel objects and occlusions.

It is important to note, that the output of the grasp detection process greatly depends on the geometric set-up of the gripper. For the same object, different gripper geometries and operation principles (parallel gripper, three-finger gripper, suction pad, etc.) result in different gripper positions and orientations for the highest probability of success rate [59]–[61]. This makes the proposed solutions gripper-specific, so if a DL model is utilized, it has to be retrained if the type of the gripper is changed.

2) Path and Trajectory Planning: A robot application that operates in a dynamic environment should plan paths and trajectories in a responsive manner [62], reacting to unexpected changes in the environment. There are conventional, non-DL-based methods for real-time motion planning, such as [63] and [64]. However, in some cases the path planning process should also generalize to yet unseen scenarios apart from being reactive, such as grasping novel objects in dynamic environments, or reacting to unseen moving obstacles. In such problems DL methods can be utilized. Due to there being no theoretical guarantee for the performance of a DL model, hybrid planning techniques can also be preferable [65].

Apart from modifying a path, that connects the current and target locations by always adapting to the current situation, a local path planning approach can be used as well. This setup is particularly used in case of mobile robots, where the navigation is only based on local landmarks and features of the environment. In this case, the output of the DL model is usually the heading direction or steering angle for the mobile robot [47], [57], [66]. These approaches are widely used in case of self driving cars, mobile robot navigation, and swarm robotics and are fundamental for SLAM applications as well [58]. The selection of the appropriate heading direction has to incorporate a vast amount of information that is hard to process with analytic approaches. In reality, for great distances even the discrimination of objects/obstacles and the ground becomes a challenge [37], [67].

3) Whole-Body Planning: There are examples for platforms that utilize the advantages of mobility and manipulative skills together [68]–[70]. In such robots legged locomotion, balance keeping, and/or manipulation with multiple arms at the same time can occur [71]. The problems associated with these kind of robots are referred to as whole-body planning and whole-body control.

The difficulty in whole-body planning is to carry out solutions for the previously mentioned challenges by combining multiple actions at once to adapt to various circumstances. The motion of the whole body influences the position of the target objects relative to the robot. This property must be taken into consideration during manipulation. So the control of the manipulators are interdependent through the extra objectives, such as keeping the balance of the robot [68], [68]–[70]. As it is an extremely hard task to formalize and compute, DL methods are often used for learning behavior of legged locomotion or keeping balance.

4) Motion Control: When the plan is successfully constructed, the next step is to perform the motion accurately by minimizing the error between the planned and the realized movements. This is mostly done through modeling the dynamic behavior of the system in question. In this case the control algorithm is able to compute the actuation needed to perform the desired motion, or predictions can be made based on the model to evaluate the most beneficial actions to take [70], [72]–[75]. There are several well-known analytic solutions for the challenges of model construction and motion control [76]. Sometimes, however, in case of unconventional robot architectures, it is easier to construct the model with learning-based methods such as DL. Also some applications require a level of flexibility from the control algorithm, that cannot be fulfilled without learned behaviors or the behavior is too difficult to describe analytically like the in-hand manipulation of objects [77].

Some of the recent learning-based solutions for motion control utilize a so-called end-to-end approach [26], [78]–[82]. Such an approach is based on a DL solution that integrates the whole dataflow into a single computational model. According to Fig. 1 an end-to-end model would take sensory data as its input, and output the motor control signals needed to perform the actions (gray dashed line). In a model like this, it is not possible to separate the different levels, like perception and motion planning. Also, the model includes the solution for problems, that could be solved with exact analytic methods as well. End-to-end approaches have a great significance, e.g., in the solution of learning locomotion for unconventional robot architectures [73], [74]. The training of the end-to-end structure for robot control is usually carried out in a try-and-error manner [82]–[84].

C. Knowledge Adaptation

The training of a DL model from scratch is a time consuming and data demanding process. If a robot application has a requirement, that is satisfied with a DL-based solution, it should be crafted in a way, that is easy to adapt to other similar problems as well. That is, why the adaptability of the model always have to be considered, and thus, it is represented in the background layer in Fig. 1. The area of transfer learning deals with approaches that leverage the knowledge gained by the solution of former problems in order to speed up the training process, or to enhance the performance of new solutions [85].

Transfer learning can be seen as the discovery and utilization of abstracted knowledge, which we will refer to as core knowledge from now on. The core knowledge can be represented as an abstract kinematic or dynamic model, a law or a set of rules, a formula or a function, an algorithm, a policy, etc. With the help of this knowledge we can create new, similar solutions for different problems. Taking the kinematics of serial manipulators as an example, the problem might be the forward kinematics task for a given robot arm. The core knowledge in this case can be represented as an abstract robot model (or a set of rules for creating a robot model) and the method for solving the forward kinematics task given a specific robot model in the defined format. With this knowledge new solutions can be constructed for the forward kinematics problem, but with different manipulators. However, there are problems, where the analytic formulation of such representations of the core knowledge is not possible or not yet available. In these cases, we can use DL methods to approximate the core knowledge [85].

The most important solutions for transfer learning in robotics are the use of pretrained DL models, sim-to-real approaches, the extraction of domain invariant features, and learning by imitation/demonstrations.

1) Pretrained Models: A very common method for transfer learning is the use of pretrained models. Pretrained models are used as general feature extractors [86], [87] as they represent a general knowledge required for a field of problems and they are used for the initialization of the DL-based solution for the new problem. During training, they may be fine tuned, or a smaller learning-based model may be trained from-scratch that processes the features extracted by the pretrained model. There are several publicly available DL models that were pretrained on the ImageNet dataset [88], such as AlexNet [3], GoogleNet [89], VGG model [90], etc. These models can be used for extracting high-level features from image data in various image processing tasks.

2) Sim-to-Real: The majority of learning-based solutions for motion control use a trial-and-error approach. These approaches usually cannot be carried out on the physical robot itself, because of safety, financial and time concerns [83], [84].

Due to the aforementioned difficulties, there is a strong need to decouple the preparation of the solution from the physical world. An obvious workaround is to create the solution in a simulation environment and apply it to the real-world problem.

The in-silico preparation of the solutions has several benefits, like full and accurate information of the environment and the great speed of computation and preparation of various environmental circumstances. The sole challenge is to adapt the solution to the real-world-problem in order to exploit the benefits of these methods. This is often referred to as bridging the "reality gap" [30], [31], for which there are two major approaches. One is randomization and the other is the close-to-real quality simulation and data generation [30], [31], [91]–[94].

Sim-to-real approaches that utilize randomization are based on the idea, that if a model is able to perform well in the highly randomized source domain, it must have great generalization properties, so it should be able to adapt to the real-world domain as well [30], [91]. The other approach to bridge the reality gap is to make the simulation environment as similar to the real world as possible, so the trained DL model can be applied for the real-world problem directly [31], [93], [94].

3) Domain-Invariant Features: Apart from sim-to-real, there are other approaches that try to overcome the challenge of different source and target domains. These approaches aim to discover features that are domain independent [14], [95]. The domain independent features can be extracted from several different domains $(\mathcal{D}_1, \mathcal{D}_2, \ldots,)$. Let's say $\mathcal{X}_1$ and $\mathcal{X}_2$ are the feature spaces of two domains, and the features are $\mathcal{X}_1 = \{\mathbf{x}_{11}, \mathbf{x}_{12}, \ldots, \}$ and $\mathcal{X}_2 = \{\mathbf{x}_{21}, \mathbf{x}_{22}, \ldots, \}$, respectively. $\mathbf{x}_{1i}$ and $\mathbf{x}_{2j}$ are common features, if they are identical and their marginal probability distribution in the two domains $(\mathcal{D}_1, \mathcal{D}_2)$ are the same. If we take a dataset of images of objects to be grasped as an example, identical features can be the red color intensity value of the same pixels of the images of different domains (if the images have the same resolution). However, only if the marginal probability distribution of this feature is the same across all domains, can we say that this feature is a common feature of the domains. Obviously, the common features are usually more complex than the color intensity of specific pixel, e.g., complex shapes and textures [86], [87]. Such features are hard to find analytically, so DL methods are utilized to extract them.

4) Imitation and Demonstration-Based Learning: Imitation learning is the term used for learning-based methods that create a solution based on demonstrations [20], [96], [97]. This approach can be successfully used for robots to learn manipulation skills [98], [99]. The demonstration is the solution for the source problem and the aim is to learn a solution for the target problem. The difference between the source and the target problems can be the differences in the embodiment of the agents. For example, imitation learning can be realized for a task of robotic manipulation, by providing demonstration from a human performing the task, and requiring the robot to perform the same task [96]. Most of the approaches for such problems suppose that the demonstration is close to the optimal solution of the source problem and thus the performance of the solution for the target problem is measured via a comparison to the demonstration [98]–[100]. This can be achieved by having a state description, and by constructing a solution for the target problem that achieves the objective through a sequence of states that is similar to the demonstration [98], [99]. The state description has to be independent of whether it is the source or target problem in order to make it comparable, so it can be made with the help of common features. This process of adapting the source solution to the target solution is also referred to as skill transfer and skill extraction.

In order to simplify the demonstration learning scenario, the source problem is usually defined in a way (the demonstrations are provided in a way), that is very similar to the target problem. This means, that instead of tracking the hand of a human demonstrating the task, the demonstration can also be provided by teleoperation. The human hand have a significantly different kinematic structure than most of the conventional robot manipulators and also the tracking of the hand can introduce errors to the system. Meanwhile, demonstrations provided in the form of teleoperation are very close to the target problem and can be easily used as starting points for later exploration [99]. Providing the demonstrations in such a way enables the collection of relevant internal data from the demonstration, such as forces, torques, velocities, etc. This data can be measured with high accuracy and most of the probability distributions of the errors of these measurements are the same hence they mainly come from the physical construction of the manipulator. That is why, in demonstration learning tasks, usually other modalities are utilized apart from vision such as tactile information [101].

III. Relevant Examples and Best Practices

A. Proposition

Preferably DL methods result in a model that is easy to adapt to new problems as well, thus avoiding most of the data exhaustive and time consuming training process that is a characteristic of such methods. For this to happen, the solution either has to be general enough that it can be applied to the new problem directly, or it has to be easily reusable/transferable. The choice of the scope and training strategy for the DL-model heavily influences the adaptability of the solution. We can highlight the advantages and disadvantages of different approaches from this aspect. We especially focus on the in-hierarchy scope of some approaches, along with their training strategies, and argue that larger DL models that incorporate multiple levels of the hierarchy presented in Fig. 1, such as end-to-end methods, are not always preferable over modular DL-based solutions. The use of transfer learning methods and the leveraging of the modular nature of DL models can help in the quick and data efficient preparation of new solutions. We support our argument with a set of relevant examples and best practices and demonstrate how these methods can be effectively utilized in robotics.

The examples are categorized according to the approaches for achieving such efficient adaptation capabilities, meanwhile the order of the presented examples roughly represents moving higher in the hierarchy in Fig. 1.

B. Feature Extraction

At the very bottom of the hierarchy, perception is carried out using the sensory data as input. If the sensory data has high dimensionality, it is reasonable to define a smaller set of features that still represent the problem well, but enable simplified calculations. The discovery of such features is referred to as feature extraction.

1) Pretrained Models and Modularity for Perception: Visual information is high dimensional data and its use is also very common in robotics, so the pretrained models that are used for image processing, such as the AlexNet, GoogleNet, and VGG networks are widely utilized in the solutions for robot applications as well. From these models, the CNN part is used as the pretrained model. These CNNs extract high-level features from the images, efficiently reducing the dimensionality. Girshick *et al.* [38] proposed the R-CNN which showed that the features extracted by these pretrained models can be used for visual object detection. Later, Girshick [102] also introduced the Fast R-CNN with which the training and inference is significantly faster than the R-CNN. In the R-CNN method, the pretrained model extracts features from regions of interest (RoI) in the input images and the extracted feature vectors are classified with object-specific support vector machines (SVMs). The RoIs are provided by a region proposal method. In case of the Fast R-CNN, the output is inferred by fully connected neural network layers from the feature maps of the RoIs. Girshick showed that the Fast R-CNN is able to perform more accurate and faster object detection than the R-CNN method (9 times faster training and 213 times faster inference) which is mostly the result of the model architecture, that the prediction layers can be optimized together with the feature extractor (single-stage training).

Ren *et al.* created a DL-based region proposal network (RPN) for the Fast R-CNN object detection method. They also fused the introduced RPN with the Fast R-CNN structure through an attentional interface and named it Faster R-CNN, a DL model that is capable of real-time object detection [103]. Their results show, that the Fast R-CNN fused with the RPN yielded superior accuracy compared to Fast R-CNN with other region proposal methods. They were also able to do inference with 17 frames per second (FPS) surpassing previous approaches. This example can demonstrate very well how the modular nature of DL methods can be leveraged to create better and complex solutions.

Johnson *et al.* [13] proposed a fully convolutional localization network. They used the Faster R-CNN method for proposing regions for a recurrent neural network that generates complex captioning of the image. Their model is called DenseCap. Valipour *et al.* [33] used this model in their incremental learning scenario as the basis of their localization network and the VGG-16 pretrained model for extracting features for the object detection and localization. Through this DL-based perception pipeline, the robot was able to retrieve (manipulate) various objects requested by the human operator via speech, that is recognized by the CMU Sphinx speech recognition module. They also utilized a speech synthesis method, so the robot is able to provide feedback on its current state verbally. For this purpose, the Festival speech synthesis system [104] was used. They also created a method for correcting the classifications of the localization network in an incremental fashion, through HRI, with the help of the speech synthesis, speech recognition, and a human gesture recognition system. This solution shows how the different DL modules can be combined to solve a very complex robotics problem, consisting of object detection, and HRI.

Redmon *et al.* [105] approached the topic of object detection from an other perspective and introduced their new network called YOLO. Contrary to the R-CNN and its variants, YOLO does not apply region proposals separate from feature extractors. Rather, the proposed bounding boxes for objects and the associated class probabilities are directly inferred from the global extracted features. So their network follows the popular architecture of having two parts, a CNN for feature extractor and a smaller neural network on top of the CNN that performs the actual classification and regression tasks. We refer to this part as the top network. This approach is integrated into a single model, so it can be trained in an end-to-end fashion, but it can still leverage the modular nature of DL models. The feature extractor and the top network can be separated and they are responsible for different subtasks. Also, instead of extracting local features (only from the region proposals), the model can infer predictions globally, based on the whole image. They also introduced a feature extractor CNN architecture for YOLO, that is inspired by the GoogleNet architecture and pretrained this feature extractor on the ImageNet dataset. The pretraining process took approximately a week to achieve the performance of the GoogleNet models as of 2012. Naturally, if there is not a specific reason to create a novel pretrained model for feature extraction, one can simply use the already available CNNs for that purpose avoiding the costs of the nearly one week training. During the evaluation of their method Redmon *et al.* also showed that a VGG-16-based YOLO is more accurate than their fastest model, but can only predict 21 FPS instead of the 155 FPS inference speed of the Fast YOLO. The comparison with other models revealed that the Fast YOLO model was able to perform object detection at a rate of 155 FPS and with a mean average precision (mAP) of 52.7, while the second fastest method, the deformable parts models (DPMs) could be run on 100 FPS with an mAP of 16.0. The most accurate method appeared to be the VGG-16-based Faster R-CNN with 73.2 mAP and 7 FPS, while the best YOLO architecture, the VGG-16-based YOLO achieved 66.4 mAP with 21 FPS. The authors have also showed that the YOLO model is able to generalize better, by comparing its performance to other detection systems on artworks and natural images from the Internet. These comparisons show that a single neural network that incorporates the whole problem may have a better performance than nonend-to-end approaches, both in terms of speed and accuracy. However, it is very important that the end-to-end model should also possess modularity, because the training of the whole network can be very resource exhaustive.

The YOLO model has a special connection to robotics as well, because the prediction process of the bounding boxes for its object detection is based on the MultiGrasp system proposed by Redmon and Angelova [106].

2) Multimodal Data and Unsupervised Pretraining for Perception: Apart from visual information, depth data is also significant in robotics. However, pretrained models for depth and point cloud information (e.g., PointNet [107]) are not as widely available as pretrained models for RGB data, so often clever tricks are used to deal with this modality.

Predicting a grasp can be carried out by estimating the pose of a known object. Zeng *et al.* [108] proposed a method with fully CNN segmentation of 2-D images from multiview RGB-D sensors for object 6-D pose estimation. Based on the segmentations of 2-D images a segmented 3-D point cloud is constructed and formerly scanned objects are fit on the segmented scene, so the pose for the objects can be estimated. With this method the authors were able to create a pose estimator model for multiple objects with occlusion in a cluttered environment. Their segmentation network is based on the pretrained VGG architecture. The output of the network is a dense probability map, with values for every pixel of the image for each of the given object labels. The probability maps are then thresholded, and the 3-D segmented point cloud is constructed from these segmentations. This approach was awarded third and fourth place in the 2016 Amazon Picking Challenge.

For the processing of multiple modalities one possible way is to use separate convolutional structures and fuse them into a common layer at a higher level of the DL model like in [109]. Schwarz *et al.* [32] encoded the depth data into color images to process them with the help of CNNs, that were pretrained on simple images. Based on the extracted features from both the images and the converted depth data, a classification is performed to identify objects on the image, and a regression method is used to predict their pose. The comparison of this approach with the method of Bo *et al.* which utilized unsupervised learning of hierarchical feature representations from RGB-D objects [110], revealed that this method was able to learn from much less data. Furthermore, the accuracy of the model for RGBD data was better than the one which trained its own feature extractor in an unsupervised manner and the RGB-based classification accuracy was also comparable (92% compared to 92.1% for the method of Bo *et al.*).

The method that Bo *et al.* utilized, the unsupervised pretraining of the feature extractor is also a very popular approach [40], [55], [110]. These methods usually use an autoencoder structure and the reconstruction error to extract meaningful and representative features from the inputs. The unsupervised training of a feature extractor to be used later as a pretrained model is important in problems where the inputs are not common structures and thus, there is no large labeled dataset available. Schmitz *et al.* [40] used a denoising autoencoder structure for pretraining their model for tactile object recognition. They concluded that the unsupervised pretraining together with dropout increased the recognition rate drastically and they were able to achieve 88% recognition rate compared to the 64.7% without pretraining and dropout.

3) Feature Extraction for Planning: The examples presented so-far can be considered as solutions for the perception problem. In case of grasp planning, however, the predictions may also depend on the kinematic model of the end effector, resulting in different strategies for different kinds of end effectors. Thus, according to our categorization in Fig. 1, such challenges are at a higher level in the hierarchy (motion planning). The following examples, that incorporate this knowledge inside the DL model provide solutions for planning problems.

The MultiGrasp system by Redmon and Angelova [106] predicts multiple grasps for various objects and also performs classification. A predicted grasp is parameterized by five values, that are the location (x, y), width and height, and the orientation of the proposed grasp. Each predicted grasp is weighted by a predicted probability of being a true grasp for the given object and the grasp with the highest weight is realized as the detected grasp. The MultiGrasp model is a single DL model performing grasp detection so it has similar advantages over other grasp predictors like the YOLO model has over the other object detectors. While the state-of-the art of that time, a grasp detection method based on two-stage cascaded deep neural networks and RGB-D data, proposed by Lenz *et al.* [55] was able to achieve 75% accuracy at 13.5 s per frame (0.074 FPS), the MultiGrasp model had 87% accuracy in the same dataset for grasp detection and it only took 75 milliseconds per frame (13.33 FPS) to detect the grasps. Just like the YOLO model, the MultiGrasp architecture is made up of a CNN for feature extraction and a top network performing the predictions, so it can also leverage the benefits of using pretrained models.

4) Similarity in Feature Spaces for Planning: Mahler *et al.* [59] introduced Dex-Net 1.0, a cloud-based large dataset for robotic grasping tasks and an algorithm to determine fast and robust grasping policies for novel objects. Their dataset consists of numerous objects (represented as 3-D object models) and several possible grasps per object, each labeled with corresponding probability of force closure. They used a so called multiview CNN (MV-CNN) method [111] to identify objects in their database that are similar to the object being grasped. The MV-CNN has the architecture of the AlexNet model and they initialized it by pretraining on the ImageNet dataset. The features extracted by the MV-CNN from several images of different objects are used to measure the similarity of these objects. The similarity is expressed as the Euclidean distance of compressed feature vectors representing the objects. Their algorithm is able to model the correlations of grasps on different objects, so it is able to utilize former grasps of similar objects recorded in their dataset. Later Mahler *et al.* also created deep CNNs for the prediction of the probability of a grasp being successful, for parallel [60] grippers and suction-based end effectors [61] as well. Both of these models were trained entirely on their synthetic dataset and reported outstanding results of being fast (0.8–3 s) with a high success rate (93%–99%). However, due to the different gripper types, parts of the model structure had to be altered and the network had to be retrained for the different tasks. In [112], they proposed a method for combining both models for more robust manipulation with two robot arms, one equipped with a parallel gripper and the other with a suction pad.

The utilization of the feature maps for measuring similarity of high-dimensional data is also present in the approach of Gao and Zhang for the detection loops in SLAM systems [113]. The proposed method utilizes a stacked denoising autoencoder for the re-localization of the mobile robot for SLAM. The compressed representation of the images, extracted by the encoder part, is used for the comparison with other images. If the similarity of the compressed features of two images is above a certain level, a loop is detected and the mobile robot platform can be relocalized to increase the accuracy of the SLAM process. The results show that the method can output more robust recognition of previously seen scenes than the FAB-MAP method [114], which was used as a baseline for the experiments.

5) Domain Invariant Features and Imitation Learning for Control: The transfer learning approaches that utilize the extraction of domain invariant features or imitation learning also leverage feature extractors for measuring similarity in a way. The following examples demonstrate, how these methods can be used in the context of transferring solutions for the challenge of controlling the motion.

Gupta *et al.* [83] created a method for skill transfer between robot agents (that were trained by RL) even with different morphologies. Based on tasks that both agents are able to solve, a common feature space is trained. Then this common feature space is used to transfer the skills of one agent to the other. From the tasks that can be solved by both agents, pairs of states can be extracted that are considered to be corresponding. A neural network for both agents maps the states to a common feature space. These neural networks are optimized to generate similar common features for the corresponding state pairs. In order to avoid trivial solutions, the encoding to the common feature space is decoded for both agents and the reconstruction error is also used in the loss function, similarly to an autoencoder structure. So this approach trains a feature extractor for both agents that can map the state into a feature space, in which similar feature vectors represent similar states of the two environments. The skill of one agent can be encoded to the common feature space and then decoded for the other agent to transfer skills. Gupta *et al.* also compare their method to an approach in which no transfer learning is carried out and the policies of the different agents are trained separately from scratch. The comparison revealed that the from-scratch training was unable to learn the solution for tasks that could be solved successfully by agents of different morphology using their proposed transfer learning method, even when it was trained on a significantly larger set of data.

Duan *et al.* proposed a method for imitation learning, which only requires a single demonstration for new, previously unseen tasks [98]. The motivation for their method is to develop a network that can be trained to perform actions to accomplish a potentially infinite number of tasks in contrast to the usual approach, when the learnt policies represented as deep networks are task-specific. Their method utilizes three neural networks. The demonstration and the context networks are responsible for creating a context embedding based on the demonstration and the current state. This context embedding is necessary for forming to fixed length input, independent of the length of the demonstration for the third part, the manipulation network. The context embedding in this case corresponds to the result of the feature extraction. The manipulation network is a simple deep neural network with fully connected layers that proposes the appropriate actions based on the context embedding. The modular nature of the system enables training the three networks separately. Although, during training the model is not enforced to learn an embedding that maps similar states of the demonstration into similar contexts, Duan *et al.* mentioned that their experimental analysis supports this interpretation of how the transfer from the demonstration to the learned policy is carried out internally.

C. Beyond Feature Extraction

Feature extraction enables the compact and informative representation of the environment. However, there are some tasks, like the control of robot systems, for which solely the representation of the environment is not enough, but the effects of certain actions on the environment should also be considered. In such problems, RL can be utilized.

1) Reinforcement Learning: The two major parts of RL are model-based and model-free RL. Model-based RL constructs a model of the environment with which the state transitions and the rewards can be predicted, thus allowing the agent planning ahead [72]. In case of model-free RL approaches, policy optimization, or Q learning can be performed.

Model-based RL methods are more data efficient than model-free approaches, whereas the proposed solutions may be suboptimal compared to the model-free methods because of the model bias, according to [73], [74], and [115]. There are multiple approaches to combine the benefits of these two fields of RL. Pong *et al.* [73] introduced temporal difference models (TDMs) that can be trained via model-free RL and used for model-based (predictive) control. The results of the proposed method were demonstrated on robotic manipulation tasks in simulation and real-life environment and on simulated locomotion tasks as well and revealed that on the given problems TDM is generally more data efficient than model-free RL approaches (converging faster) while it also achieves at least the accuracy of the policies trained by a model-free method.

An other solution to leverage both model-based and model-free RL was proposed by Nagabandi *et al.* [74]. They introduced a model-based RL method that can be trained efficiently to implement a deep neural network dynamics model. Then this predictive model was used to form an initial policy for a model-free RL process for fine tuning, to achieve model-free performance with better efficiency. They also carried out benchmark experiments on locomotion tasks, that show their method is able to achieve state-of-the-art performance but with a 3–5 times greater data efficiency.

Levine *et al.* [26] proposed an end-to-end DL-based method for hand-eye coordination in robotic grasping problems. Their method consists of a grasp prediction system and a servoing function that is responsible for the continuous control of the robot, according to the output of the grasp prediction. This approach is a good example for how the planning and the control tasks can be separated in DL approaches for robotic systems. Their grasp prediction system is a deep CNN that can infer the grasp success probability from a series of images and motion commands. The servoing function samples motor commands and selects the one with the highest associated probability for a successful grasp, with the help of the cross entropy method. They explained that the grasp prediction network can be interpreted as an approximation of the Q-function defined by the servoing method, with the assumption that there is no difference between moving from point A to point C directly or moving from point A to point C through point B. This assumption is not always true in a grasping problem, because intermediate movements may relocate objects in the scene, but it enables to reduce the fitting of the Q-function to a simple prediction task. In their grasp prediction network they did not use a pretrained model. The network was trained on a large-scale real-life dataset of over 800 000 grasp attempts collected in the course of two months with the help of multiple (6–14) real life robotic manipulators.

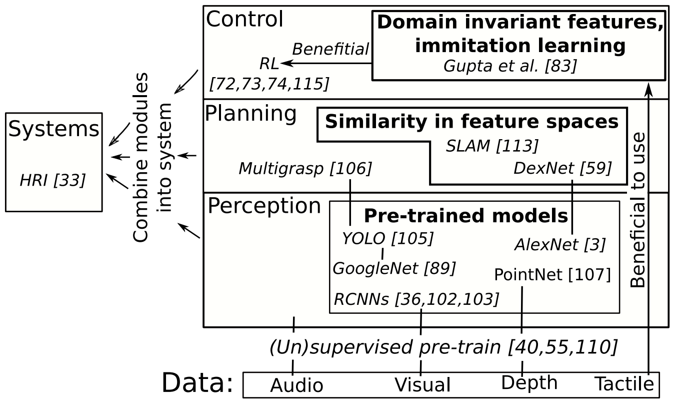


Fig. 2. Summary of examples.

The results show, that the system is able to develop intelligent grasping strategies (e.g., different strategy for grasping soft and hard objects) and can generalize to grasping unseen objects, different camera placements, and slightly different robot configurations, but Levine *et al.* highlight that the method may fail to generalize significant differences in the robot platform, such as a different manipulator or gripper, or the changes in the environment, like grasping from a shelf instead of a planar surface. In case of these problems, the grasp prediction model had to be retrained with additional training samples for the new problem.

Levine *et al.* [82] also proposed a method for robotic manipulation in which a single model is responsible for both the planning and the control of the motion. It takes images of the environment and the robot configuration as its input and outputs the torques for the motors of the robot. The learnt policy is implemented by a deep CNN consisting of a convolutional visual processing part and a fully connected control network on the top of the visual processing part. Even though the proposed model is monolithic it leverages modular pretraining. The filters of the first visual processing layer were initialized with the weights of the GoogleNet model, and the whole visual processing part was initialized by training it to perform 3-D pose regression. For the initialization of the control network the visual processing part was removed from the policy and the full state was provided instead, for which the control network could be optimized. After their individual initialization, the visual processing part, and the control network were combined into the policy. At first the control network was trained while the weights of the visual processing part were left frozen, and then the whole policy was optimized by fine-tuning it in an end-to-end manner. Levine *et al.* compared the performance of the end-to-end fine tuned policy with one that was only fine tuned down to the pose regression features and an other one in which only the predicted poses are fed to the control layers. They found that the policies with the end-to-end fine-tuning outperformed both other approaches and the ones that used the pose regression features were superior to those that only relied on the predicted poses. This implies that if the fine tuning is performed up to deeper layers, the model can be better adapted to the specific problem. With this approach Levine *et al.* were able to create policies for different manipulation tasks in a matter of 3–4 h, of which only a fraction is actual robot movement for data collection, so it is significantly faster than collecting data with multiple robots for multiple months. However, they also note that the trained policies are not able to generalize to significantly different scenes and that the training of a single policy with multiple robots working in different environments, advanced pretraining techniques and better data augmentation may be necessary for a more general policy.

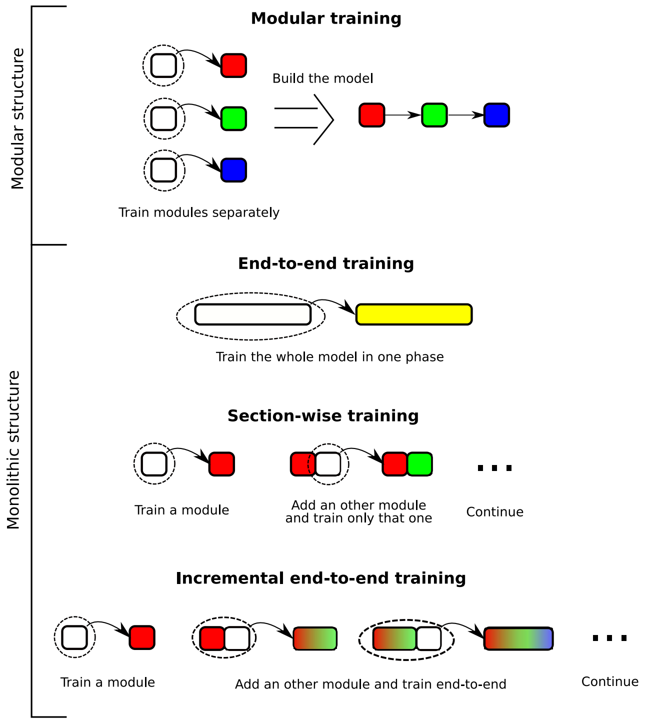


Fig. 3. Model structures and training strategies (best viewed in color).

Fig. 2 summarizes the findings derived from the examples, and gives an insight on how the knowledge adaptation methods are related to the different levels of the hierarchy in the selected examples. In the figure, the knowledge adaptation approaches were highlighted in bold font and some examples were given with italic.

IV. Discussion

From the introduced representative examples, it can be seen that different model structures and training strategies can be used for similar problems effectively, each having their own advantages and drawbacks. However, there seems to be no general consensus on the definition of the terms that are important to discuss these matters. For example end-to-end DL is often associated with the model structure, meaning that a single DL model performs the task, but it is also frequently used to describe the training strategy when the model is trained as a whole. This can lead to misconceptions as there are approaches which use a single DL model for the solution, but it is not trained as a whole. We recommend the separate discussion of the model structure and the training strategy and according to the introduced best practices, we differentiate two model structures and four kinds of training strategies. The proposed terminology can be seen in Fig. 3.

For the solution of a problem either a composition of multiple models, or a single model may be used. We call solutions with multiple models the modular structure, and the single-model solutions are called monolithic structures. The modules in the modular structure are responsible for subtasks of the problem and they can also be considered as individual modular or monolithic structures. Examples for the modular approach are the RCNN, and the Fast RCNN for the object detection task, and the proposal of Valipour *et al.* [33] for an incremental learning system via HRI. From the comparison of the RCNN variants and the YOLO model, the conclusion can be drawn that as the solution approaches the monolithic structure (fully connected layers instead of SVMs in Fast RCNN; RPN in Faster RCNN; and finally the completely monolithic YOLO), the speed and performance of the method improves. However, there is a practical limit to creating arbitrary large monolithic models, which is determined by the complexity of the task and the amount of the available training data. While a single monolithic model can be sufficient for object detection, a complex system that includes object detection, HRI via speech recognition and speech synthesis, human pose recognition, etc., cannot be implemented as a monolithic structure. In this case, the modules should be trained separately and the system should be combined from the individual modules. Thus, the fundamental building blocks of every modular approach eventually will be smaller monolithic structures. This can also be easily seen in Tree-CNN, the approach of Roy *et al.* [116] that uses a hierarchical tree structure of DCNN modules to provide adaptation capability. By leveraging modularity, the system can be given an adaptive nature by incrementally training new modules upon receiving data from yet-unseen classes. Also, the better control over what a module is responsible for, makes it possible to build hierarchical structures, where modules in different layers of the hierarchy can classify according to super-classes or subclasses. Due to the advantages and disadvantages of both structures neither of them is generally superior and the proper structure should be selected according to the nature of task.

Assessing the computational and time complexity of training DL models is a challenging task due to various reasons, such as the several existing model architectures, training strategies, and hardware environments. However, in order to better understand what methods can be used to reduce these values, recent research is carried out to find a method to predict them [117]. Although regarding monolithic models, it is fairly simple to see that as a result of the reuse of the modules, there are parts which do not have to be trained again, and this results in less computations and reduced training time. Also in case of the try-and-error RL methods the data collection is simultaneous with the training process, so a large monolithic model with many parameters, that requires a lot of data to train, requires a lot of time to train as well.

We define three different training strategies regarding the monolithic models. The first one is the end-to-end training, when the entire model is trained (all the parameters are updated) for the specific problem. For the end-to-end training of a model, large amounts of training data is necessary, so this strategy is typically applied for the creation of feature extractors, for which large, labeled datasets are available, like the ImageNet dataset. An example for such a model is the pretrained part of the YOLO. If there is no such training dataset for the desired task, the data collection can take up plenty of time, like in [26], where Levine *et al.* collected 800 000 grasp attempts in the course of two months with the help of multiple (6–14) real life robotic manipulators to train a grasp detector in an end-to-end manner.

In case of the section-wise training strategy, the monolithic model is composed of smaller modules. However, contrary to the modular approach, in the section-wise training strategy the modules are not independent on their own (except the first one), because later modules are trained with respect to the result of the training of the previous modules in the model. A typical example for this approach is when a pretrained model is used, and the training is performed with the weights of the pretrained model frozen, so only the top network is updated. For example, in the VGG-16-based YOLO model, the top network is trained to perform the object detection with respect to the given VGG-16 feature extractor, while in the original YOLO model, the top network is trained with respect to the feature extractor introduced by Redmon *et al.* [105], so these modules are not interchangeable, because they depend on the feature extractor.

For the adaptation of an end-to-end-trained monolithic model to new problems the time consuming and data exhaustive training process usually have to be repeated from scratch, because the whole model is optimized for a specific task. However, in a monolithic model which is trained with the section-wise approach, only the higher-level modules will be specialized for a given task. So, in order to repurpose the model, one could simply retrain the higher-level modules only. Naturally, due to the working principle of deep neural networks, even the end-to-end-trained monolithic models possess this property, so lower layers of the network extract more general features than the layers above them. The difference is, that in section-wise training we have control over which part of the model is responsible for what subtask, allowing us the engineering of a more transparent decision process, thus making the knowledge adaptation easier. The advantage of training a model in an end-to-end fashion, provided that a sufficiently large training dataset is available, is that it is more likely to achieve a better performance due to the sheer number of parameters that are optimized for the specific task. In order to combine the strengths of both approaches, the incremental end-to-end training strategy can be utilized.

In case of the incremental end-to-end training strategy the model is constructed like the section-wise-trained monolithic models, but instead of training the top network only, end-to-end training is utilized. this way the whole model can be optimized for a specific problem for better performance, but we can also enforce the model to develop a solution for certain tasks at certain levels of its hierarchy.

There is also a middle-ground between the section-wise and the incremental end-to-end training strategy, when neither the whole model is trained in an end-to-end fashion, nor just the top network is updated, but a part of the already trained weights are also modified along with the top network weights. Levine *et al.* proved in [82], that the deeper one goes with the training the better performance the model has on the specific task, eventually making the incremental end-to-end training approach the best possible choice, provided that a sufficiently large training dataset is available. In order to reduce the data needs of the incremental end-to-end training strategy Levine *et al.* also utilized the section-wise approach to initialize the model, and the end-to-end training was applied only after that.

The mentioned training strategies are, among other things, determined by the model structure, the amount of available training data, the complexity of the task, etc. Although hybrid approaches already exist [82], one has to select a training strategy according to these criteria. Since the nature of the tasks for which DL is used and the fundamental structure of DL models and training procedures are not expected to change significantly, we expect all of these training strategies to stay relevant in the future as well.

Based on our findings, we provide a brief checklist that can be followed when designing a DL-based solution for a robot application.

1) Specify the requirements that the application should fulfil and check if it could benefit from a DL-based solution (this could be done according to Section II).
2) Identify the specific task that the DL-model should be responsible for and locate it in the hierarchy of Fig. 1.
3) Search for methodologies and previous solutions from Section III or elsewhere.
4) Decide on the model structure (monolithic or modular) based-on the scope of the problem (Fig. 1) and the available resources.
5) Carry out a training strategy for the model. If possible use pretrained models, feature extractors and/or transfer learning, and the incremental end-to-end training approach. If the available training dataset is not large enough for the end-to-end fine-tuning, then use the section-wise training strategy.
6) Build complex systems with the modular approach if the monolithic models for specific tasks are already trained.

V. Summary

In this survey, we proposed a widespread analysis of the field of DL in robotics. We categorized the major challenges according to the most significant features a robotic task like perception, motion, and the adaptation to variable tasks and domains. We found that this categorization helps us to classify problems, place them in a hierarchy, and search for solutions suitable for the given type of problems. We also classified the DL methods according to the structure of the DL-model and the strategy used for its training. This way, we were able to give examples of successful solutions for each category of challenges and see the influence and role of the introduced DL structures and training strategies.

We also provided a guideline for creating new DL-based solutions for robot applications in Section IV. We hope that our discussion on the DL model structures and training methodologies can dissolve the misconceptions about end-to-end models and end-to-end learning and that our paper can provide a useful hint for the future research in DL and robotics.

References

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[2] L. Deng and D. Yu, "Deep learning: Methods and applications," *Found. Trends Signal Process.*, vol. 7, nos. 3–4, pp. 197–387, 2014.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[4] L. Deng, G. Hinton, and B. Kingsbury, "New types of deep neural network learning for speech recognition and related applications: An overview," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2013, pp. 8599–8603.

[5] G. Hinton *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, Nov. 2012.

[6] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2013, pp. 6645–6649.

[7] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, *Object Recognition With Gradient-Based Learning*. Berlin, Germany: Springer, 1999, pp. 319–345. [Online]. Available: https://doi.org/10.1007/3-540-46805-6_19

[8] J. Dean *et al.*, "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1223–1231.

[9] Q. V. Le *et al.* (2011). *Building High-Level Features Using Large Scale Unsupervised Learning*. [Online]. Available: http://arxiv.org/abs/1112.6209

[10] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 1701–1708.

[11] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 160–167.

[12] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Comput. Intell. Mag.*, vol. 13, no. 3, pp. 55–75, Aug. 2018.

[13] J. Johnson, A. Karpathy, and F. Li. (2015). *Densecap: Fully Convolutional Localization Networks for Dense Captioning*. [Online]. Available: http://arxiv.org/abs/1511.07571.

[14] X. Glorot, A. Bordes, and Y. Bengio, "Domain adaptation for large-scale sentiment classification: A deep learning approach," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 513–520.

[15] R. Socher *et al.*, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conf. Empiric. Methods Nat. Lang. Process.*, 2013, pp. 1631–1642.

[16] H. A. Pierson and M. S. Gashler. (2017). *Deep Learning in Robotics: A Review of Recent Research*. [Online]. Available: http://arxiv.org/abs/1707.07217.

[17] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, Sep. 2013. [Online]. Available: https://doi.org/10.1177/0278364913495721.

[18] S. Amarjyoti. (2017). *Deep Reinforcement Learning for Robotic Manipulation—The State of the Art*. [Online]. Available: http://arxiv.org/abs/1701.08878.

[19] L. Tai and M. Liu. (2016). *Deep-Learning in Mobile Robotics—From Perception to Control Systems: A Survey on Why and Why Not*. [Online]. Available: http://arxiv.org/abs/1612.07139.

[20] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robot. Auton. Syst.*, vol. 57, no. 5, pp. 469–483, 2009.

[21] Y.-H. Pao and Y. Takefuji, "Functional-link net computing: theory, system architecture, and functionalities," *Computer*, vol. 25, no. 5, pp. 76–79, 1992.

[22] Y.-H. Pao, G.-H. Park, and D. J. Sobajic, "Learning and generalization characteristics of the random vector functional-link net," *Neurocomputing*, vol. 6, no. 2, pp. 163–180, 1994.

[23] C. P. Chen and Z. Liu, "Broad learning system: An effective and efficient incremental learning system without the need for deep architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 10–24, Jan. 2018.

[24] C. L. P. Chen and Z. Liu, "Broad learning system: A new learning paradigm and system without going deep," in *Proc. 32nd Youth Acad. Annu. Conf. Chin. Assoc. Autom. (YAC)*, May 2017, pp. 1271–1276.

[25] A. Yamaguchi and C. G. Atkeson, "Differential dynamic programming for graph-structured dynamical systems: Generalization of pouring behavior with different skills," in *Proc. IEEE-RAS 16th Int. Conf. Humanoid Robots (Humanoids)*, Nov. 2016, pp. 1029–1036.

[26] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. (2016). *Learning Hand-Eye Coordination for Robotic Grasping With Deep Learning and Large-Scale Data Collection*. [Online]. Available: http://arxiv.org/abs/1603.02199.

[27] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proc. ICML Workshop Unsupervised Transfer Learn.*, 2012, pp. 17–36.

[28] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proc. ICML Workshop Unsupervised Transfer Learn.*, 2012, pp. 37–49.

[29] G. E. Hinton, *A Practical Guide to Training Restricted Boltzmann Machines*. Berlin, Germany: Springer, 2012, pp. 599–619. [Online]. Available: https://doi.org/10.1007/978-3-642-35289-8_32

[30] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. (2017). *Domain Randomization for Transferring Deep Neural Networks From Simulation to the Real World*. [Online]. Available: http://arxiv.org/abs/1703.06907.

[31] K. Bousmalis *et al.* (2017). *Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping*. [Online]. Available: http://arxiv.org/abs/1709.07857.

[32] M. Schwarz, H. Schulz, and S. Behnke, "RGB-D object recognition and pose estimation based on pre-trained convolutional neural network features," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2015, pp. 1329–1335.

[33] S. Valipour, C. P. Quintero, and M. Jägersand. (2017). *Incremental Learning for Robot Perception Through HRI*. [Online]. Available: http://arxiv.org/abs/1701.04693.

[34] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. (2016). *Deep Learning of Local RGB-D Patches for 3D Object Detection and 6D Pose Estimation*. [Online]. Available: http://arxiv.org/abs/1607.06038.

[35] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, Aug. 2013.

[36] P. H. Pinheiro and R. Collobert, "Recurrent convolutional neural networks for scene labeling," in *Proc. 31st Int. Conf. Mach. Learn. (ICML)*, 2014, pp. 82–90.

[37] S. Ghosh and J. Mulligan, "A segmentation guided label propagation scheme for autonomous navigation," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 895–902.

[38] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. (2013). *Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation*. [Online]. Available: http://arxiv.org/abs/1311.2524.

[39] Y. Gao, L. A. Hendricks, K. J. Kuchenbecker, and T. Darrell. (Nov. 2015). *Deep Learning for Tactile Understanding From Visual and Haptic Data*. [Online]. Available: http://arxiv.org/abs/1511.06065.

[40] A. Schmitz, Y. Bansho, K. Noda, H. Iwata, T. Ogata, and S. Sugano, "Tactile object recognition using deep learning and dropout," in *Proc. IEEE RAS Int. Conf. Humanoid Robots*, 2014, pp. 1044–1050.

[41] B. Hollis, S. Patterson, and J. Trinkle. (Sep. 2016). *Compressed Learning for Tactile Object Classification*. [Online]. Available: http://arxiv.org/abs/1609.07542.

[42] L. Pinto, D. Gandhi, Y. Han, Y. Park, and A. Gupta. (2016). *The Curious Robot: Learning Visual Representations via Physical Interactions*. [Online]. Available: http://arxiv.org/abs/1604.01360.

[43] A. Valada, L. Spinello, and W. Burgard, *Deep Feature Learning for Acoustics-Based Terrain Classification*. Cham, Switzerland: Springer Int. Publ., 2018, pp. 21–37.

[44] S. Uemura, O. Sugiyama, R. Kojima, and K. Nakadai, "Outdoor acoustic event identification using sound source separation and deep learning with a quadrotor-embedded microphone array," in *Proc. Int. Conf. Adv. Mechatronics Toward Evol. Fusion IT Mechatronics (ICAM)*, Dec. 2015, pp. 329–330. [Online]. Available: https://www.jstage.jst.go.jp/article/jsmeicam/2015.6/0/2015.6_329_article/-char/ja/

[45] E. Mumolo, M. Nolich, and G. Vercelli, "Algorithms for acoustic localization based on microphone array in service robotics," *Robot. Auton. Syst.*, vol. 42, no. 2, pp. 69–88, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889002003251

[46] Y. Liao, L. Huang, Y. Wang, S. Kodagoda, Y. Yu, and Y. Liu. (2016). *Parse Geometry From a Line: Monocular Depth Estimation With Partial Laser Observation*. [Online]. Available: http://arxiv.org/abs/1611.02174.

[47] A. Dubrawski and I. Siemiatkowska, "A method for tracking the pose of a mobile robot equipped with a scanning laser range finder," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 3, May 1998, pp. 2518–2523.

[48] H. Abdelkawy, N. Ayari, A. Chibani, Y. Amirat, and F. Attal. (2018). *Deep Hmresnet Model for Human Activity-Aware Robotic Systems*. [Online]. Available: http://arxiv.org/abs/1809.07624.

[49] W. Elmenreich, *An Introduction to Sensor Fusion*, Master Sci. Elect. Eng., Vienna Univ. Technol., Vienna, Austria, 2002.

[50] K. Nagla, M. Uddin, and D. Singh, "Multisensor data fusion and integration for mobile robots: A review," *IAES Int. J. Robot. Autom.*, vol. 3, no. 2, p. 131, 2014.

[51] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard, "Multimodal deep learning for robust RGB-D object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2015, pp. 681–687.

[52] V. Vielzeuf, A. Lechervy, S. Pateux, and F. Jurie. (2018). *Multi-Level Sensor Fusion With Deep Learning*. [Online]. Available: http://arxiv.org/abs/1811.02447.

[53] A. Bicchi and V. Kumar, *Robotic Grasping and Manipulation*. Berlin, Germany: Springer, 2001, pp. 55–74. [Online]. Available: https://doi.org/10.1007/3-540-45000-9_3

[54] A. Sahbani, S. El-Khoury, and P. Bidaud, "An overview of 3D object grasp synthesis algorithms," *Robot. Auton. Syst.*, vol. 60, no. 3, pp. 326–336, Mar. 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0921889011001485

[55] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *Int. J. Robot. Res.*, vol. 34, nos. 4–5, pp. 705–724, 2015.

[56] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *Int. J. Robot. Res.*, vol. 27, no. 2, pp. 157–173, 2008.

[57] A. R. Puthussery, K. P. Haradi, B. A. Erol, P. Benavidez, P. Rad, and M. Jamshidi, "A deep vision landmark framework for robot navigation," in *Proc. 12th Syst. Syst. Eng. Conf. (SoSE)*, Jun. 2017, pp. 1–6.

[58] D. DeTone, T. Malisiewicz, and A. Rabinovich. (2017). *Toward Geometric Deep SLAM*. [Online]. Available: http://arxiv.org/abs/1707.07410.

[59] J. Mahler *et al.*, "DEX-Net 1.0: A cloud-based network of 3D objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016, pp. 1957–1964.

[60] J. Mahler *et al.* (2017). *DEX-Net 2.0: Deep Learning to Plan Robust Grasps With Synthetic Point Clouds and Analytic Grasp Metrics*. [Online]. Available: http://arxiv.org/abs/1703.09312.

[61] J. Mahler, M. Matl, X. Liu, A. Li, D. V. Gealy, and K. Goldberg. (2017). *DEX-Net 3.0: Computing Robust Robot Suction Grasp Targets in Point Clouds Using a New Analytic Model and Deep Learning*. [Online]. Available: http://arxiv.org/abs/1709.06670.

[62] D. Morrison, P. Corke, and J. Leitner. (2018). *Closing the Loop for Robotic Grasping: A Real-Time, Generative Grasp Synthesis Approach*. [Online]. Available: http://arxiv.org/abs/1804.05172.

[63] J. Bruce and M. M. Veloso, "Real-time randomized path planning for robot navigation," in *Robot Soccer World Cup VI (RoboCup)*, G. A. Kaminka, P. U. Lima, and R. Rojas, Eds. Berlin, Germany: Springer, 2003, pp. 288–295. [Online]. Available: https://doi.org/10.1007/978-3-540-45135-8_23

[64] S. Macfarlane and E. A. Croft, "Jerk-bounded manipulator trajectory planning: Design for real-time applications," *IEEE Trans. Robot. Autom.*, vol. 19, no. 1, pp. 42–52, Feb. 2003.

[65] A. H. Qureshi, M. J. Bency, and M. C. Yip. (2018). *Motion Planning Networks*. [Online]. Available: http://arxiv.org/abs/1806.05767.

[66] D. K. Kim and T. Chen. (2015). *Deep Neural Network for Real-Time Autonomous Indoor Navigation*. [Online]. Available: http://arxiv.org/abs/1511.04668.

[67] M. Ollis, W. H. Huang, M. Happold, and B. A. Stancil, "Image-based path planning for outdoor mobile robots," in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 2723–2728.

[68] M. Fallon *et al.*, "An architecture for online affordance-based perception and whole-body planning," *J. Field Robot.*, vol. 32, no. 2, pp. 229–254, 2015.

[69] D. Berenson, J. Chestnutt, S. S. Srinivasa, J. J. Kuffner, and S. Kagami, "Pose-constrained whole-body planning using task space region chains," in *Proc. 9th IEEE-RAS Int. Conf. Humanoid Robots*, Dec. 2009, pp. 181–187.

[70] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, Nov. 2014, pp. 295–302.

[71] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization based full body control for the atlas robot," in *Proc. IEEE-RAS Int. Conf. Humanoid Robots*, 2014, pp. 120–127.

[72] C. Finn and S. Levine, "Deep visual foresight for planning robot motion," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 2786–2793.

[73] V. Pong, S. Gu, M. Dalal, and S. Levine. (2018). *Temporal Difference Models: Model-Free Deep RL for Model-Based Control*. [Online]. Available: http://arxiv.org/abs/1802.09081.

[74] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. (2017). *Neural Network Dynamics for Model-Based Deep Reinforcement Learning With Model-Free Fine-Tuning*. [Online]. Available: http://arxiv.org/abs/1708.02596

[75] A. Yamaguchi and C. G. Atkeson, "Model-based reinforcement learning with neural networks on hierarchical dynamic system," in *Proc. Workshop Deep Reinforcement Learn. Front. Challenges 25th Int. Joint Conf. Artif. Intell. (IJCAI)*, 2016, pp. 1–5.

[76] M. Spong, S. Hutchinson, and M. Vidyasagar, *Robot Modeling and Control*. Wiley, 2005. [Online]. Available: https://books.google.hu/books?id=A0OXDwAAQBAJ.

[77] OpenAI *et al.* (2019). *Solving Rubik's Cube With a Robot Hand*. [Online]. Available: https://arxiv.org/abs/1910.07113.

[78] P. Long, W. Liu, and J. Pan. (2016). *Deep-Learned Collision Avoidance Policy for Distributed Multi-Agent Navigation*. [Online]. Available: http://arxiv.org/abs/1609.06838.

[79] T. P. Lillicrap *et al.* (2016). *Continuous Control With Deep Reinforcement Learning*. [Online]. Available: http://arxiv.org/abs/1509.02971.

[80] M. Bojarski *et al.* (2016). *End to End Learning for Self-Driving Cars*. [Online]. Available: http://arxiv.org/abs/1604.07316.

[81] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 739–746.

[82] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2016.

[83] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine. (2017). *Learning Invariant Feature Spaces to Transfer Skills With Reinforcement Learning*. [Online]. Available: http://arxiv.org/abs/1703.02949.

[84] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, no. Jul, pp. 1633–1685, 2009.

[85] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[86] M. Huh, P. Agrawal, and A. A. Efros. (2016). *What Makes Imagenet Good for Transfer Learning?*. [Online]. Available: http://arxiv.org/abs/1608.08614.

[87] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. (2014). *How Transferable are Features in Deep Neural Networks?*. [Online]. Available: http://arxiv.org/abs/1411.1792.

[88] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, 2009, pp. 1–2.

[89] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. (2015). *Rethinking the Inception Architecture for Computer Vision*. [Online]. Available: http://arxiv.org/abs/1512.00567.

[90] K. Simonyan and A. Zisserman. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. [Online]. Available: https://arxiv.org/abs/1409.1556.

[91] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. (2017). *Sim-to-Real Transfer of Robotic Control With Dynamics Randomization*. [Online]. Available: http://arxiv.org/abs/1710.06537.

[92] S. James and E. Johns. (2016). *3D Simulation for Robot ARM Control With Deep Q-Learning*. [Online]. Available: http://arxiv.org/abs/1609.03759.

[93] B. Planche *et al.* (2017). *Depthsynth: Real-Time Realistic Synthetic Data Generation From CAD Models for 2.5D Recognition*. [Online]. Available: http://arxiv.org/abs/1702.08558.

[94] Y. You, X. Pan, Z. Wang, and C. Lu. (2017). *Virtual to Real Reinforcement Learning for Autonomous Driving*. [Online]. Available: http://arxiv.org/abs/1704.03952.

[95] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He, "Supervised representation learning: Transfer learning with deep autoencoders," in *Proc. IJCAI*, 2015, pp. 4119–4125.

[96] J. Lee. (2017). *A Survey of Robot Learning From Demonstrations for Human–Robot Collaboration*. [Online]. Available: http://arxiv.org/abs/1710.08789.

[97] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends Cogn. Sci.*, vol. 3, no. 6, pp. 233–242, 1999.

[98] Y. Duan *et al.* (2017). *One-Shot Imitation Learning*. [Online]. Available: http://arxiv.org/abs/1703.07326.

[99] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. (2017). *Overcoming Exploration in Reinforcement Learning With Demonstrations*. [Online]. Available: http://arxiv.org/abs/1709.10089.

[100] A. Y. Ng and S. J. Russell, "Algorithms for inverse reinforcement learning," in *Proc. ICML*, 2000, pp. 663–670.

[101] G. Sutanto *et al.* (2018). *Learning Latent Space Dynamics for Tactile Servoing*. [Online]. Available: http://arxiv.org/abs/1811.03704.

[102] R. B. Girshick. (2015). *Fast R-CNN*. [Online]. Available: http://arxiv.org/abs/1504.08083.

[103] S. Ren, K. He, R. B. Girshick, and J. Sun. (2015). *Faster R-CNN: Towards Real-Time Object Detection With Region Proposal Networks*. [Online]. Available: http://arxiv.org/abs/1506.01497.

[104] P. Taylor, A. W. Black, and R. Caley, "The architecture of the festival speech synthesis system," in *Proc. 3rd ESCA Workshop Speech Synthesis*, 1998, pp. 147–151.

[105] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. (2015). *You Only Look Once: Unified, Real-Time Object Detection*. [Online]. Available: http://arxiv.org/abs/1506.02640.

[106] J. Redmon and A. Angelova. (2014). *Real-Time Grasp Detection Using Convolutional Neural Networks*. [Online]. Available: http://arxiv.org/abs/1412.3128.

[107] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. (2016). *Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation*. [Online]. Available: http://arxiv.org/abs/1612.00593.

[108] A. Zeng *et al.*, "Multi-view self-supervised deep learning for 6D pose estimation in the amazon picking challenge," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 1383–1386.

[109] R. Socher, B. Huval, B. Bath, C. D. Manning, and A. Y. Ng, "Convolutional-recursive deep learning for 3d object classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 656–664.

[110] L. Bo, X. Ren, and D. Fox, *Unsupervised Feature Learning for RGB-D Based Object Recognition*. Heidelberg, Germany: Springer, 2013, pp. 387–402. [Online]. Available: https://doi.org/10.1007/978-3-319-00065-7_27

[111] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, "Multi-view convolutional neural networks for 3D shape recognition," in *Proc. ICCV*, 2015, pp. 945–953.

[112] J. Mahler *et al.*, "Learning ambidextrous robot grasping policies," *Sci. Robot.*, vol. 4, no. 26, 2019, Art. no. eaau4984.

[113] X. Gao and T. Zhang, "Unsupervised learning to detect loops using deep neural networks for visual slam system," *Auton. Robots*, vol. 41, no. 1, pp. 1–18, 2017.

[114] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *Int. J. Robot. Res.*, vol. 27, no. 6, pp. 647–665, 2008.

[115] P. Dayan and Y. Niv, "Reinforcement learning: The good, the bad and the ugly," *Current Opinion Neurobiol.*, vol. 18, no. 2, pp. 185–196, 2008.

[116] D. Roy, P. Panda, and K. Roy. (2018). *Tree-CNN: A Deep Convolutional Neural Network for Lifelong Learning*. [Online]. Available: http://arxiv.org/abs/1802.05800.

[117] D. Justus, J. Brennan, S. Bonner, and A. S. McGough. (2018). *Predicting the Computational Cost of Deep Learning Models*. [Online]. Available: http://arxiv.org/abs/1811.11880.

Artúr István Károly (Member, IEEE) received the B.Sc. and M.Sc. degrees in mechatronic engineering from the Budapest University of Technology and Economics, Budapest, Hungary, in 2016 and 2018, respectively. He is currently pursuing the Ph.D. degree in deep learning in robotics with the Doctoral School of Applied Informatics and Applied Mathematics and the Antal Bejczy Center for Intelligent Robotics, Óbuda University, Budapest.

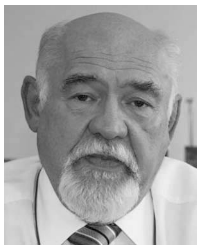
His research interests include the development of new deep learning methods and the study of their functioning principles, the enhancement of neural networks via other machine learning approaches and unsupervised learning techniques, and the realization of general knowledge representation.

Péter Galambos (Member, IEEE) received the M.Sc. and Ph.D. degrees in mechanical engineering from the Budapest University of Technology and Economics, Budapest, Hungary, in 2006 and 2013, respectively.

He was a Research Intern with the Toshiba Corporate Research and Development Center, Kawasaki, Japan, from 2007 to 2008, then joined to the Institute for Computer Science and Control, Hungarian Academy of Sciences (MTA-SZTAKI), Budapest, where he held a "Young Researcher" Scholarship from 2010 to 2012. From 2011 to 2015, he served as a Team Leader with MTA SZTAKI and coordinated the development of the VirCA VR system and its research applications. In 2013, he has joined Óbuda University, Budapest, where he participates in robotics-related Research and development activities and education. He is currently the Director of the Antal Bejczy Center for Intelligent Robotics and the Deputy Director of the University Research and Innovation Center, Óbuda University. His current research interests include advanced industrial robotics and control systems, cyber–physical systems, and virtual reality.

József Kuti (Member, IEEE) received the B.Sc. and M.Sc. degrees in mechatronical engineering from the Budapest University of Technology and Economics, Budapest, Hungary, in 2011 and 2013, respectively, and the Ph.D. degree in computer science from the Óbuda University, Budapest, in 2018.

He is currently a Research Assistant with the Antal Bejczy Center for Intelligent Robotics, Óbuda University. His research interests include LPV/qLPV modeling, TP model transformation and linear matrix inequality-based control design of nonlinear, optimal filtering, robot modeling and control.



Imre J. Rudas (Fellow, IEEE) received the M.Sc. degree in applied mathematics from Bánki Donát Polytechnic, Budapest, Hungary, in 1971, the master's degree in mathematics from the Eötvös Loránd University, Budapest, the Ph.D. degree in robotics from the Hungarian Academy of Sciences, Budapest, in 1987, and the Doctor of Science degree in robotics from the Hungarian Academy of Sciences, in 2004.

He is a Professor Emeritus with Óbuda University, Budapest, where he is the President of the University Research and Innovation Center. He has published six books, 12 university books, more than 850 papers in various journals and international conference proceedings, and received more than 5500 citations. His present areas of research activities are computational cybernetics, robotics, and computational intelligence.

Prof. Rudas is a President Elect of IEEE Systems, Man, and Cybernetics Society.