Deep Learning in Robotics

by

Sepehr Valipour

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

New machine learning methods and parallel computation opened the door to many applications in computer vision. While computer vision is progressing rapidly because of that, there are not as many and as successful real world applications in robotics. In this thesis, we investigate two possible causes of this problem and we provide potential solutions. In particular, using recurrent networks to deal with temporal information and using human robot interaction for incremental learning.

Robotics is time dependent. The environment is perceived through time and tasks are being performed by time indexed trajectories. This important property has been generally neglected in the deep learning community. The main focus instead is on improving benchmarks on single image tasks or analyzing videos in batches. Consequently, real-time video analysis received less attention. But this is exactly what robots need to have. Processing single images will not provide sufficient information to observe the environment and processing a batch of images will be too delayed to be used for planning and decision-making in real-time. As a solution to this problem, we propose a recurrent fully convolutional neural network(RFCNN) for segmentation, which is very useful in different robotics scenarios. This type of network accepts a series of images ending with the current image to infer the segmentation for the last image. We showed how such networks can be designed and trained in an end-to-end fashion. An extensive set of experiments on several different architectures and benchmarks were made. We observed a consistent improvement by using RFCNN over non-recurrent counterparts.

While deep learning methods are the most general machine learning solutions available, they still suffer from the change in the data distribution between training and test. Even though it is not limited to robotics, their effect is most apparent

there. It is mostly due to the fact that robots need to learn complicated reasoning using a limited train data. This combination leads to severe overfitting which will become obvious during the test in a slightly different environment. To mitigate this problem, we suggest a novel paradigm in which new perception information can be thought to the robot through Human-Robot Interaction(HRI). A complete HRI system is developed that allows human-friendly communication using speech and gesture. An incremental learning method is designed to improve an object detection network by using human inputs. The system is tested on simulated data and real experiments with success. We showed that humans can easily teach new objects to the robot and the robot is able to learn and use this information later.

Preface

This thesis is an original work by Sepehr Valipour. Two referenced works in the Chapter 3 and one referenced work in Chapter 4 have been previously published. ”Fully Convolutional Recurrent Neural Networks for Video Segmentation presented in WACV 2017 [75] and ”Recurrent Fully Cnvolutional Network for Video Segmentation” to be presented at ICIP 2017 [67] , are collaborative works with Mennatullah Siam. Both works received equal contribution from the authors. I was mainly responsible for the system design. ”Incremental Learning for Robots perception through HRI to be presented at IROS 2017 [74], is a collaborative work with Camilo Perez. I was responsible for developing the the detection system and the incremental learning framework and integration.

Acknowledgements

I would like to thank my supervisor Dr. Martin Jagersand. He always helped me with his insightful feedbacks. Our meetings were constructive but moreover, it thought me to have analytical thinking and different perspective toward a problem. I also want to extend my gratitude toward Dr. Dana Cosbzas, Dr. Nilanjan Ray and Dr. Eleni Stroulia who advised me on various projects during my program.
I count myself lucky for having great friends and colleagues in the robotics lab whom, without them, these two years would have been much less exciting. Camilo, Masood, Rong, Oscar, Jun, Peter thank you for your help, support and fun times.
Finally, I want to thank my parents for being there for me anytime that I needed them. This accomplishment would not have been possible without them.

Sepehr Valipour

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Moving robotics more and more to unstructured environment and use them for multiple tasks rather than specific applications was, and is, a goal of the community. This goal, however, was beyond traditional engineering approaches where a robot is designed and programmed for a very limited task. To overcome this issue, machine learning is used as it can deal with a broader range of situations. This paradigm is growing in popularity especially after successful applications of deep learning in image analysis.

Main motivation behind using deep learning in robotics is that they are significantly more general than any other learning algorithm. It has been demonstrated that deep networks are capable high level reasoning and abstraction. Therefore, it makes them an idea choice for robotics in unstructured environment. Additionally, thanks to advances in parallel processing and sophisticated numerical libraries, deep networks have become very efficient. Time critical robotics tasks need processing module with high frequency response rate to be able to control the motion. Deep networks on GPUs can deliver this.

These being said, there is still a large gap between practical applications of deep learning in image processing compared to their usage in robotics. Below, I enumerate few reasons for this gap.

- **Temporal information**: Robotics applications are time dependent by nature. Robots work in dynamic environments, their actions are time indexed and what they perceive is a function of time. For example, a tennis player robot need to track the ball over time and based on its speed and angle generate a

time indexed trajectory for a successful hit. Note that, information for such an action cannot be derived from a single image as velocity is not observable. Majority of computer vision community are not concerned with such scenarios and therefore, most of deep architectures are designed for single image processing.

- Training data As it was mentioned, deep networks are very capable mapping function and are able to learn any non-linear function from input to output. This, however, is coming at a cost. The training data that is needed for a deep network to learn a general function for a task grows proportional to the task complexity. For example, distinguishing black from white pictures may only require a handful of samples with different shades but differentiating cats from dogs require thousands. Note that the accuracy of the network may suffer as more and more data sample with different appearance are provided. Robotics is an extreme case. Consider a household assistant robot and assume that we managed to collect enough data in the lab to train it to perform basic object handling task. The robot is then shipped to a customer house, and unlucky us, the objects in the house look very different than the objects that we had in the lab. Current solution of the computer vision community for this problem is to increase the training data to incorporate similar samples to the possible test scenario.

In this work, I tried to provide possible solutions to these problems by leveraging functionalities in the robots that are generally ignored by the vision community. Namely, the availability of temporal data (video, sensor signals, etc.) and the human robot interaction.

Firstly, in chapter 2 background information and related works needed to understand rest of the paper is presented. In the next chapter 3 we explain our possible solution for incorporating temporal data for video analysis. In particular, we describe a fully convolutional recurrent network for video segmentation. Later, in chapter 4 we introduce an incremental learning approach in which robots can learn from humans through natural interaction. Our contributions can be listed as follow:

- Novel video segmentation method based on recurrent networks, suitable for online processing in robotics.
- Extensive experiments showing the superiority of this method over single image segmentation.
- Introducing an incremental learning model to dynamically update a deep network using natural human interaction.
- Implementing and testing a complete interaction system and demonstrating its viability.

In the last chapter 5 summarize this manuscript and propose a unifying framework that can make use of the HRI through interactions and recurrent network for temporal signal analysis.

Chapter 2

Background and Literature Review

2.1 Deep Learning

The term Deep learning is loosely associated with a broad range of neural network architectures [28]. Common properties of these architectures can be listed as follow:

- A graph structure where each node (commonly referred to as neurons) is connected to either input signal or other nodes with weighted edges. The output of node is the linear combination of edge weights and the inputs and possibly followed by a non-linear function.
- This graph defines a cost function $C = F(\boldsymbol{X}, \boldsymbol{W}, \boldsymbol{Y})$. Where $\boldsymbol{X}$ is the input signal and $\boldsymbol{W}$ represents all the edge weights and $\boldsymbol{Y}$ is a set targets given the corresponding input.
- The cost function is minimized with respect to $\boldsymbol{W}$ using non-linear optimization techniques (mostly gradient decent or its variants).

While variety of graph structures were tested, still, a layered network is the most common design. In this design, each layer is only receiving signal from the nodes in the previous layers and its output is only fed into the next. Figure 2.1 showcases a very simple neural network with one hidden layer. Lets assume that the networks output $\boldsymbol{O}$ is a regression vector. The cost function can be defined in 2.1, where x is

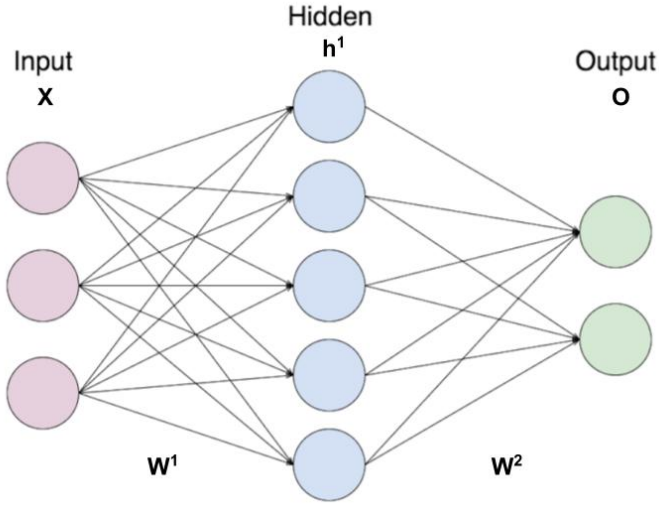

Figure 2.1: A simple example of a neural network.

a single input sample and y is the corresponding target.

$$C = \sum_{x} (\boldsymbol{O}(x_i) - y_i)^2 \tag{2.1}$$

To find $\boldsymbol{W}$ such that $\boldsymbol{W} = argmin_w C$ using gradient decent, the gradients of cost with respect to $\boldsymbol{W}$ should be computed and $\boldsymbol{W}$ optimized accordingly. This can be done using back-propagation. Formally, $\frac{\partial C}{\partial \boldsymbol{W}}$ needs to be computed.

While computing this derivative is straight forward for the last set of weights ($\boldsymbol{W}^2$), it is increasingly more difficult for the previous weights. To perform the back-propagation in a more systematic way, we can employ chain rule 2.2. In this equation n indicates the last layer and $h^n = \boldsymbol{O}$. Note that the computed derivatives for the higher layer can be reused as we back-propagate and there for the process can be done efficiently.

$$\frac{\partial C}{\partial \boldsymbol{W^i}} = (\prod_{j=n}^{i+1} \frac{\partial h^j}{\partial \boldsymbol{h^{j-1}}}) \frac{\partial h^i}{\partial \boldsymbol{W^j}} \tag{2.2}$$

Neural networks were used since 1990s. They have been revitalized thanks to, first, better processors (especially GPUs) and , secondly, introducing convolutional filters into neural networks (CNN). CNNs are bio-inspired variant of fully connected neural networks where they operate on a region of the input signal, in contrast with single element. Therefore they can exploit local connectivity in visual data. E.g.

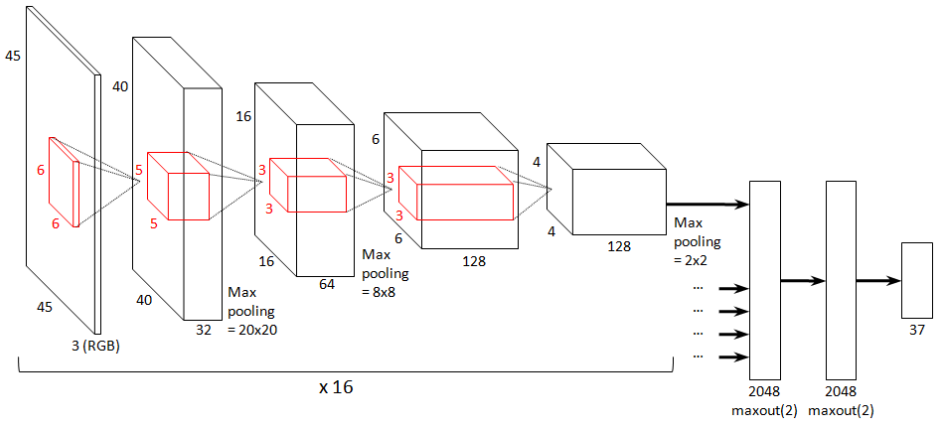

Figure 2.2: Modern classifier deep network. It incorporates multiple layers of convolutional filters with few fully connected layers.

the neighbor pixel values are related to each other. CNN provides a very efficient way to process images which lead to its popularity in the computer vision and machine learning community. Using CNN in conjunction with fully connected layers, produced modern neural network architecture for classification 2.2,

In the following sections, three specific neural network architectures will be discussed which later will be used in our systems.

Fully Convolutional Networks for Segmentation

In convolutional neural networks that are used for classification, the few last fully connected layers are responsible for the classification part. But, with pixel-wise labelling, there is a need for dense predictions on all the pixels. In [44] the idea of using a fully convolutional neural network that is trained for pixel-wise semantic segmentation is presented. It is shown that it surpasses the state of the art in semantic segmentation on PASCAL VOC, NUYDv2, and SIFT Flow datasets. The FCN method is briefly discussed in what follows.

FCN architecture is based on VGG [68] architecture due to its success in classification tasks. However, due to the fully connected layers that these networks have, they can only accept fixed size input and produce a classification label. To overcome this problem, it is possible to convert a fully connected layer into a convolutional layer. Convolution filters are independent of the input size and therefore can operate on any input spatial size. So if we substitute fully connected layers

with convolutional layers, the network can produce a coarse map instead of a single prediction. Moreover, the learned filters can be used on images with different resolutions to some extent(large difference will decrease the performance).

In order to have dense prediction from this coarse map, it needs to be up-sampled to the original size of the input image. The up-sampling method can be a simple bi-linear interpolation. In [44] a new layer that applies upsampling within the network was presented. It makes it efficient to learn the up-sampling weights within the network using back-propagation. The filters of the deconvolution layer act as the basis to reconstruct the input image. Another idea for up-sampling is to stitch together output maps from shifted version of the input. But It was mentioned in [44] that using up-sampling with deconvolution is more effective. In [50] the idea of having a full deconvolution network with both deconvolution layers and unpooling layers is presented. Figure 2.3 depicts the difference between a fully connected network used for classification and a fully convolutional networks used for segmentation.

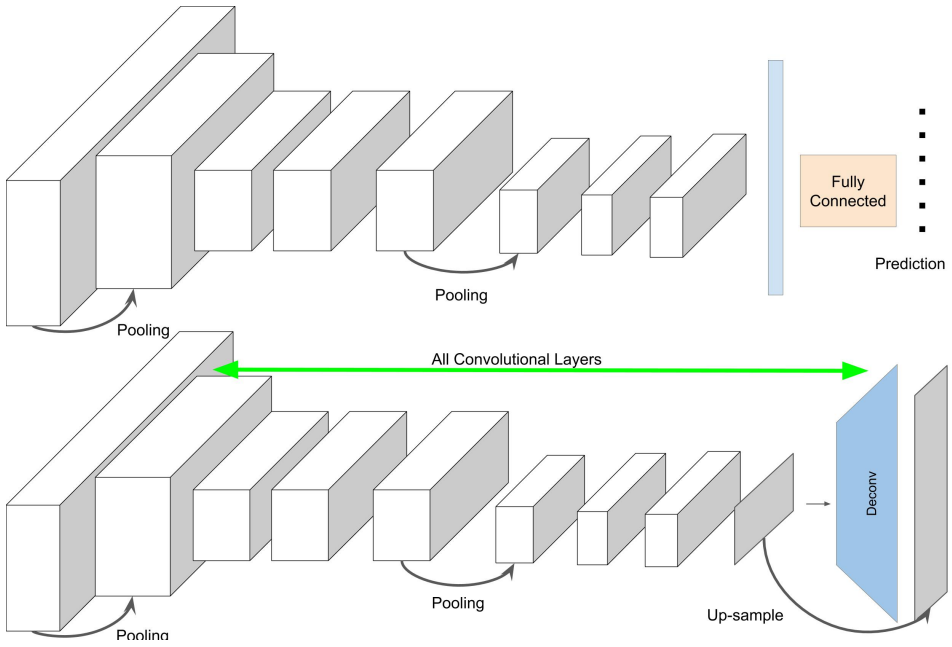

Figure 2.3: A fully connected network for classification (top) versus a fully convolutional network for segmentation (bottom). The main difference is the removal of fully connected layer in the FCN.

The FCN architecture has been tried in different applications. In [31] it is used

for object localization. A single FCN network was used to predict bounding box locations from a pyramid of the input image. It was shown that the network can be trained for multiple task (segmentation and localization) and perform better in either. In [78] a modified architecture was used for visual object tracking. Feature maps from different layers passed through two separate FCN branches and joined for a better tracking. Finally for semantic segmentation in [50] a full deconvolution network is presented with stacked deconvolution layers. Having multiple deconvolution layers showed to have positive effects an the accuracy of segmentation.

2.1.1 Recurrent Networks

Recurrent Neural Networks [76] are designed to incorporate sequential information into a neural network framework. These networks are capable of learning complex dynamics by utilizing a hidden unit in each recurrent cell. This unit works like a dynamic memory that can be changed based on the state that the unit is in. Accordingly, the process of each unit yields to two outcomes. Firstly, an output is computed from the current input and the hidden units values (the networks memory). Secondly, the network updates its memory based on, again, current input and hidden units value. The simplest recurrent unit can be modeled as Eq 2.3.

$$h_t = \theta\phi(h_{t-1}) + \theta_x x_t \tag{2.3a}$$

$$y_t = \theta_y \phi(h_t) \tag{2.3b}$$

Here, h is the hidden layer, x is the input layer and y is the output layer and ϕ is the activation function. θ, θ_x and θ_y are the unit's learnable weights.

Recurrent networks were successful in many tasks in speech recognition and text understanding [69] but they come with their challenges. Unrestricted data flow between units causes problems with vanishing and exploding gradients [7]. During the back propagation through recurrent units, the derivative of each node is dependent of all the proceeding nodes. This is shown in equations 2.4,2.5 and 2.6 where E is the loss of the layer. To compute $\frac{\partial h_t}{\partial h_k}$ a series of multiplication from $k = 1$ to $k = t - 1$ is required. Assume that $\dot{\phi}$ is bounded by α then $||\frac{\partial h_t}{\partial h_k}|| < \alpha^{t-k}$

$$\frac{\partial E}{\partial \theta} = \sum_{t=1}^{t=S} \frac{\partial E_t}{\partial \theta} \tag{2.4}$$

$$\frac{\partial E_t}{\partial \theta} = \sum_{k=1}^{k=t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial \theta} \tag{2.5}$$

$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=k+1}^{t} \theta^T diag[\dot{\phi}(h_{i-1})] \tag{2.6}$$

A solution to this problem is to use gated structures. The gates can control back propagation flow between each node. Long-Short Term Memory [30] is the first such proposed architecture and it is still popular. A more recent architecture is Gated Recurrent Unit [13] which has simpler cells yet with competent performance [14].

Long Short Term Memory (LSTM)

As mentioned, LSTM uses a gated structure where each gate controls the flow of a particular signal. Each LSTM node has three gates that are input, output and forget gate each with learnable weights. These gates can learn the optimal way to remember useful information from previous states and decide the current state. In equations 2.7 the procedure of computing different gates and hidden states is shown, where i_t, f_t and o_t are input, forget and output gates respectively. While c_t denote the cell internal state, and h_t is the hidden state. The operator $\odot$ is the element-wise product.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \tag{2.7a}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \tag{2.7b}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \tag{2.7c}$$

$$g_t = \sigma(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{2.7d}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{2.7e}$$

$$h_t = o_t \odot \phi(c_t) \tag{2.7f}$$

Gated Recurrent Unit (GRU)

The Gated Recurrent Unit, similar to LSTM, utilizes a gated structure for flow-control. However, it has a simpler architecture which makes it both faster and less memory consuming. The model is shown in Figure 2.4 and described in 2.8 where r_t, z_t is the reset and update gate respectively. While h_t is the hidden state.

$$z_t = \sigma(W_{hz}h_{t-1} + W_{xz}x_t + b_z) \tag{2.8a}$$

$$r_t = \sigma(W_{hr}h_{t-1} + W_{xr}x_t + b_r) \tag{2.8b}$$

$$\hat{h_t} = \Phi(W_h(r_t \odot h_{t-1}) + W_x x_t + b) \tag{2.8c}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z \odot \hat{h_t} \tag{2.8d}$$

GRU does not have a direct control over memory content exposure while LSTM has it by having an output gate. These two are also different in the way that they update the memory nodes. LSTM updates its hidden state by summation over flow after input gate and forget gate. GRU however, assumes a correlation between how much to keep from the current state and how much to get from the previous state and it models this with the z_t gate.

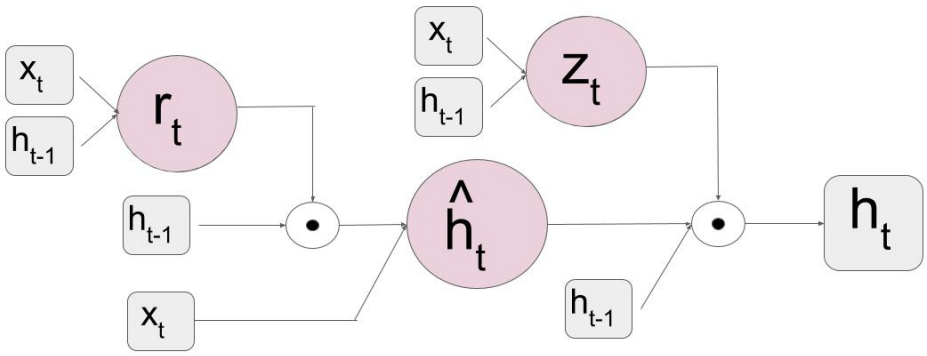

Figure 2.4: GRU Architecture.

2.1.2 Object Detection and Recognition Networks

The goal of object detection methods is to find the best-fit bounding boxes around ”objects” in a possibly cluttered scene. Traditionally, for still images, the object detection was done using image gradient and finding salient boundaries between an

object and the background or patches with a solid color. Using videos would allow detecting moving object much easier by incorporating temporal filters.

More recent works saw extensive use of SIFT [47] and HOG [19] features. Admitted by most researchers in the field, this period brought very little improvement mostly by using ensemble models and combining different low level and high-level filters. Region proposal was also a general theme with different detection techniques. These methods are either based on super-pixel grouping or sliding window. They, however, are extremely inefficient. Their accuracy depends on the fineness of sampling which directly affects the computation cost. To be more practical researchers usually limit the range of classes to be detected to avoid too small or very large objects.

The next leap in object detection framework came after the introduction of modern Convolutional Neural Networks by [38]. Deep networks were shown to be able to extract better features than traditional HOG and SIFT for object classification. The question then was how the success of deep networks in classification extends to detection. Multiple concurrent works were done to investigate this problem.

Szegedy et al. [71] formulates the detection problem as a direct regression to find a bounding box. However, they showed that this approach does not work well in practice. Sliding window was also tried in [66] for pedestrian detection and in overfeat [65] for general object detection. While they were state of the art at the time they had few drawbacks. Firstly, using sliding window caused computation problems as mentioned. Secondly, using convolutional networks with pooling layers put restrictions on the minimum size of the input image. Therefore, either the spatial resolution should be low or the depth of the network small. Both cases would decrease the accuracy.

To overcome problems with sliding window approach, Girshik et al. [60] proposed a within-network object proposal techniques. This method does not rely on an external proposal method such as the sliding window. Instead, it internally generates the proposals and evaluates them using a deep network. Therefore, region proposal could be learned and improved. It also reduced the inference time drastically as now the whole system can run on the GPU. Region proposal networks and their

variants are current state of the art in object detection.

Object detection systems are commonly accompanied by a recognition mechanism. A simple, yet inefficient, way to achieve recognition is to crop the detected region on the original image and input it to a classifier network. It is inefficient since we are recomputing very similar features to what was already extracted with the detection network. A better approach would be to use intermediate features of the detection network along with a classifier on top of them in order to recognize the object. Moreover, training a detection and recognition network together can deal better with overfitting.

2.2 Human Robot Interaction

Autonomous robots have become a consistent part of the manufacturing process in large companies. However, due to their limitations in terms of tasks that they can perform, their usage in smaller companies or household are less viable. Such environments require more flexible robots that can perform a range of tasks without needing to go through a rigorous reprogramming of the robot every time. Especially since the reprogramming needs to be done by experts. While robots with full autonomy are not yet available for these scenarios, there exists a great opportunity for other solutions. The human-robot collaboration paradigm is one of those solutions.

Human-Robot Interactions (HRI) spans a broad range of robots and applications. There are several factors that affect the design of an HRI system. The list below, enumerates the key elements.

- **Human Proximity**: Depending on the physical location of the robot and the user, the HRI system needs to be modified. If the robot and the user are in the same location, the human can preserve the environment through its own sensor. But, if the robot is in a remote location, the user's sole source of information is what the robot is transmitting.

- **Robot's Design**: The HRI system is limited by the robot's capabilities. For example, robotic arms are ideal for physical interaction but they have a very

short range. In the other end, there are UAV's with extreme range but close to none physical interaction.

- **Medium and Form of the Communication** HRI systems can be designed to support one or more type of communications medium (speech, gesture, physical, etc). They can also define their own communication protocols (how a message is interpreted by either party).

- **Application** The application directly influences the choice for the HRI system. For example, designing an interaction software for an arm robot for surgery and another arm robot for operating in space is very different. Even though the hardware is quite similar, the type of applications that are meant for drastically change the design. For the space robot, latency and gravity should be taken into account. For a surgical robot, accurate haptic feedback and exact workspace boundaries are important.

- **Autonomy** HRI systems can differ based on the level of the autonomy that is expected from the robot. In one end, there are fully autonomous robots that operate near humans. Roomba vacuum robot is a great example of such systems where it only operates based on it sensory inputs. At the other end of the spectrum, there are teleoperated robots that are fully controlled by the human user.

  A right combination of autonomy and interaction, however, could do much better than these two extremes. Semi-autonomy a incorporates advantages of both systems to overcome their disadvantages. In this paradigm, the human does not attain the full control of every actuator and will let the robot take care of laborious part of the control. At the same time, the robot can utilize human guidance which circumvents difficult high-level reasoning by the robot.

  A good example is [59]. Understanding the concept of making pizza, what are the ingredients are or how to combine them is extremely challenging for robots. At the same time, the physical aspect of the task is too repetitive and time consuming for humans. A good collaboration can form where the human instructs the robot on how to make a pizza and the robot takes care of

the physical part2.5.

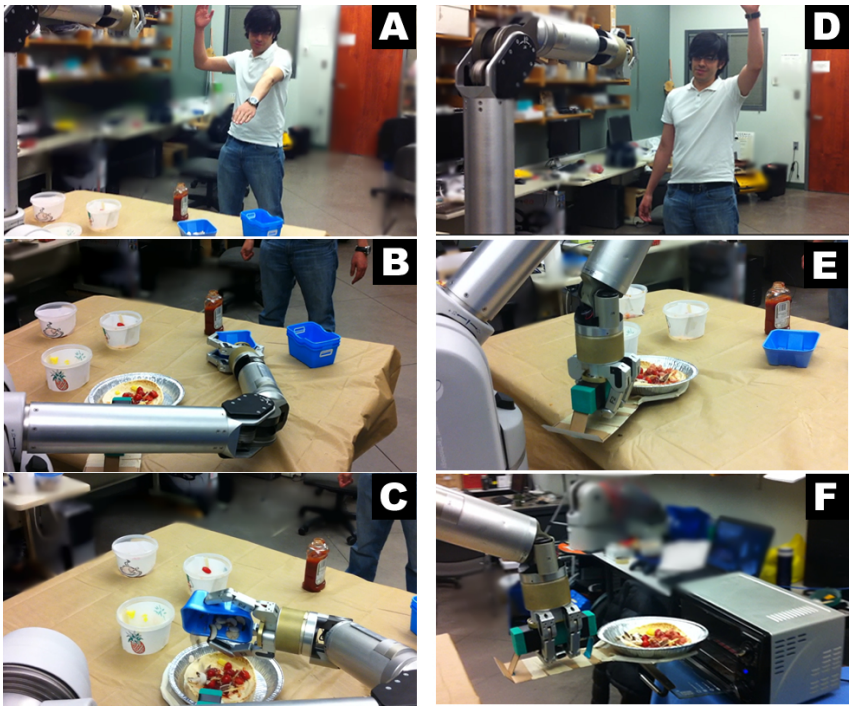

Figure 2.5: The pizza maker robot using human robot interaction proposed by [59]. The human instructs the robot about the process using a combination of gestures and pointing. The robot percepts these instruction and performs the action

In the past, the main focus of HRI was to develop more convenient ways for humans to control robots [22]. A shift in paradigm has brought the focus to ways to improve robot's understanding of humans' methods of communications for a more humanly natural interaction. Main modalities of natural communications are [10], speech, body language, facial expressions, and physical interaction. The term "natural" refers to the fact that no prior knowledge or specific device is needed for the human to be able to communicate with the robot. This requirement places a burden on the robot's perception module as it needs to understand human affect through these medians. Such robots are also called social robots.

In the following, I will review prominent works that have addressed perception problems for Social HRI systems. But first, lets look at three main components of a social HRI system.

- **Perception**: Possibly, the most important part of a social HRI robot. It is

responsible for converting raw signals that sensors have received from the environment to something meaningful for the robot. A few examples are mapping, image recognition/detection/tracking, voice recognition, and etc.

- **Intermediate**: This module encapsulates a robot's world representation. One can also refer it as the brain. It is responsible for using the information from the perception in decision making. For example, a nurse robot should give appropriate drugs to patients. A face recognition signal is provided by the perception. Intermediate here, should categorize patients by their faces and administer correct drug for each.

- **Action**: The last component is acting. At this point, the robot has perceived the world and made a decision based on it, now, it wants to perform an action or communicate its conclusion to the human. The action module will do so through different medias such as speech synthesis, gesture, digital (text, images), facial expression, etc.

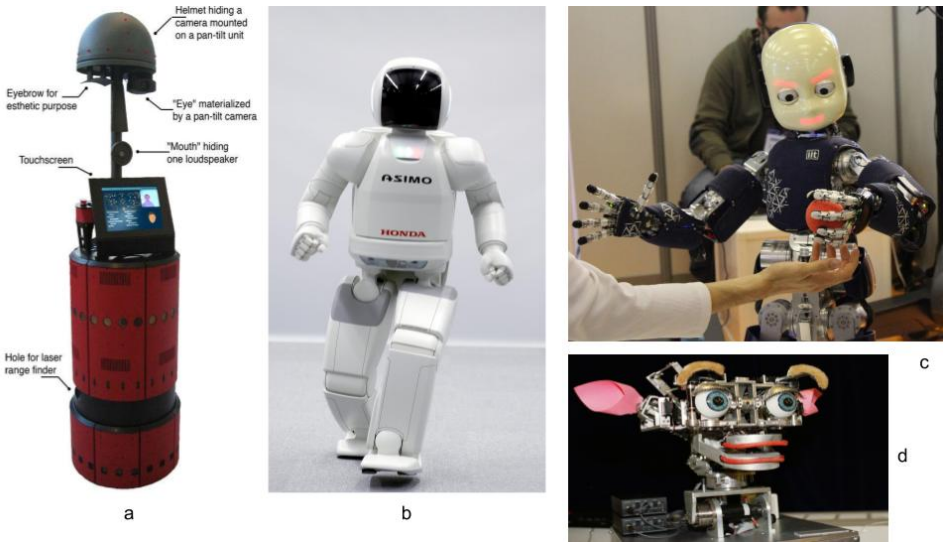

Figure 2.6: Examples of social robots. a) Rackham [15] is a representative robot that operates as a tour guide. b) Asimo [29] is a humanoid capable of showing human-like characteristics. It has been used for HRI studies. c) ICub [51] is a child-like robot with dexterous hands and several perception modules to detect and track objects of interests. d) Kismet [9] one of the earliest social robots. It can communicate using facial expressions.

Two of earliest successful works are Kismet by MIT [9] and Asimov by Honda [29]. Asimov is a humanoid robot capable of showing human-like characteristics. It has been used as an assistant that collaborates with humans. Kismet has facial features and can use it to some extent to communicate. It has been widely used for HRI studies.

ICub [51] is another example of social robots that is mostly designed for further study on this subject. It is built similar to a 5-year-old in shape. It has dexterous hands and neck and can manipulate objects and track them with its head. It is capable of recognizing faces and detecting and tracking humans.

Social robots have been used as representatives or tour guides as well. RobotX [32] and Rackham [15] were tour guides at Swiss national science expo and the BioSpace Exhibition, respectively. They were responsible for communicating with the guests and ask their desired destination and guide them to there. They were equipped with speech recognition and synthesis and navigation modules. Their other capabilities include, face recognition, motion detection, and gesture classification.

Chapter 3

Recurrent Fully Convolutional Networks for Video Segmentation

The recent trend in convolutional neural networks has dramatically changed the landscape in computer vision. The first task that was improved with this trend was object recognition [38] [68] [70]. An even harder task that greatly progressed is semantic segmentation, which provides per pixel labelling as introduced in [46] [81] [77]. In [46] fully convolutional network was introduced. These networks yield a coarse segmentation map for any given image, and it is followed by upsampling within the network to get dense predictions. This method enabled an end-to-end training for the task of semantic segmentation of images. However, one missing element in this recent trend is that real-world is not a set of static images. Large portion of the information that we infer from the environment comes from motion. For example, in an activity recognition task, the difference between walking and standing is only profound if you consider a sequence of images.

The conventional Convolutional Neural Networks (CNN) are not designed to include temporal changes. The simplest way to include temporal information in CNN is to concatenate multiple frames and feed it as a single input. Small variations of this method are used for context classification on one million youtube videos [35]. Surprisingly, it could not improve on single frame prediction by much which can indicate the inefficiency of this approach. Michalski et al. [48] created a network which learns transformations between frames of a long video sequence. In [72] a convolutional restricted Boltzman machine is introduced which learns optical-flow-

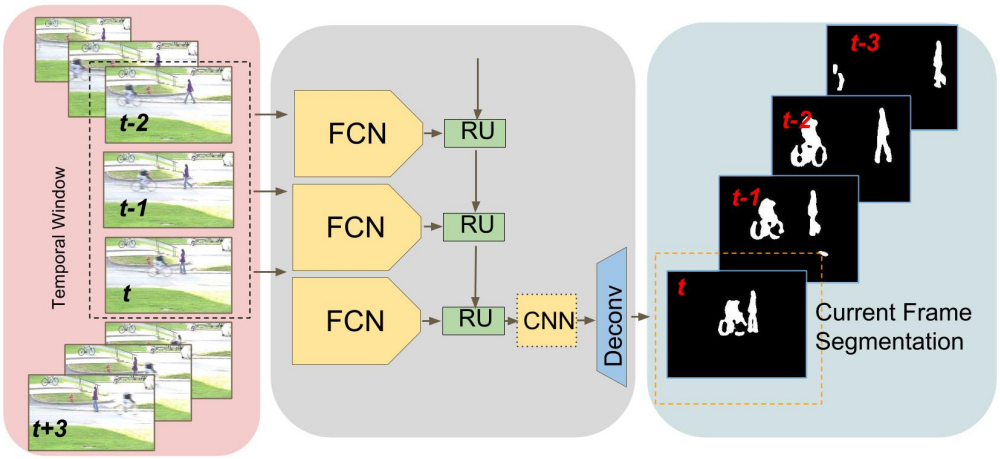

Figure 3.1: Overview of the Proposed Method of Recurrent FCN. The recurrent part is unrolled for better visualisation

like features from input image sequence. Another proposed method [52] uses Recurrent Neural Networks (RNN) which have shown their power in different tasks. It introduces a combination of CNN and RNN for video segmentation. However, their RNN architecture is sensitive to initialization, and training is difficult due to the vanishing gradient problem. Their design does not allow usage of a pre-trained network and it cannot process large input images as the number of parameters in their network grows exponentially with the input size.

Several architectures are proposed that aim to solve the main bottleneck of recurrent networks namely, vanishing or exploding gradients. In [30] Long Short Term Memory (LSTM) is presented. It is used for various applications such as generating sequences for text [24], dense image captioning [33] and video captioning [21]. Another recently proposed architecture is the Gated Recurrent Unit (GRU) [13]. It was shown in [14] that LSTM and GRU outperform other traditional recurrent architectures, and that GRU showed similar performance to LSTM but with reduced number of parameters. One problem with these previous architectures is that they only work with vectorized sequences as input. Thus, they are incapable of handling data where spatial information is critical like images or feature maps. Another work [4] uses convolutional GRU for learning spatiotemporal features from videos that tackle this issue. In their work, experiments on video captioning and human action recognition were conducted.

The usage of fully convolutional networks (FCN) combined with recurrent gated units can solve many of the pitfalls of the previous approaches. In this paper, we present : (1) A novel architecture that can incorporate temporal data directly into FCN for video segmentation. We have chosen the Recurrent Neural Network as the foundation of our structure since it is shown to be effective in learning temporal dynamics. (2) An end-to-end training method for online video segmentation that does not need to process data offline. Overview of the suggested method is presented in Figure 3.1, where a sliding window over the frames is used and passed through the recurrent fully convolutional network(RFCN). To our knowledge, this is the first work that presents a recurrent fully convolutional network for pixel-wise labeling.

3.1 Methodology

In an abstract view, we use a recurrent fully convolutional network (RFCN) that utilizes the temporal as well as spatial information for segmentation. The general theme in the design is to use recurrent nodes that combine fully convolutional network with a Recurrent unit(RU). The recurrent unit denotes either LSTM, GRU or Conv-GRU (which is explained in 3.1.2). In all of our networks, we aim for online segmentation in contrast to batch/offline version which needs the whole video as input. This is done by using a sliding window over the frames. Then, each window is propagated through the RFCN and yields a segmentation corresponding to the last frame in the sliding window. The recurrent layer can be employed to a sequence of feature maps or heat maps where each element in the series is the result of an image forward pass through an FCN network (Figure 3.1). The whole network is trained end-to-end using pixel-wise classification logarithmic loss. We designed different network architectures to this end using conventional and also convolutional recurrent unit 3.1.

3.1.1 Conventional Recurrent Unit for Segmentation

Our first architecture uses the Lenet network converted to a fully convolutional network as the base. Lenet is a well known and shallow network and as it is common,

we used it for early experiments. We embed this model in a recurrent node to allow the network to use temporal data. In Table 3.1, RFC-Lenet is delineating this architecture. The output of deconvolution inside the fcn is a 2D map of dense predictions that is then flattened into 1D vector as input to the recurrent unit. The recurrent unit takes a vector from each frame in the sliding window and outputs the segmentation of the last frame (Figure 3.1).

Note that utilizing the first architecture requires using large weight matrix in the RU layer since it operates on the flattened vectors of the full sized image. To lessen this problem, we can apply the deconvolution layer after the recurrent node. This will lead to our second architecture. The RU receives a vector of the flattened coarse map as its input and outputs the coarse predictions of the last frame in the sliding window. Then, deconvolution of this coarse map is used to get dense predictions. This proved to be useful with larger input images that lead to large number of parameters in the RU. Decreasing their size, allows the optimizer to search in a smaller state space and find a more generalized local minima in a faster training time. RFC-12s in the Table 3.1 is an example of this approach. This network is slightly modified version of the Lenet FCN. The main difference is that now the deconvolution comes as the last step after recursion being done.

3.1.2 Convolutional Gated Recurrent Unit (Conv-GRU) for Segmentation

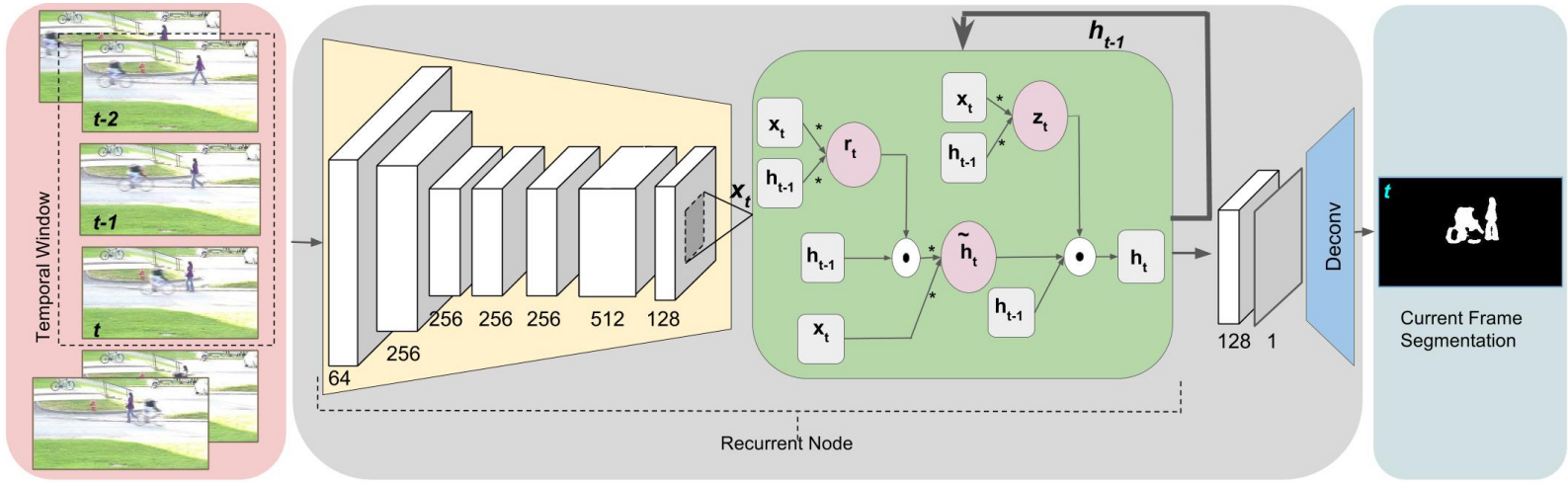

Figure 3.2: The architecture of RFC-VGG. Images are fed frame by frame into a recurrent FCN. A Conv-GRU layer is applied on the feature maps produced by the preceding network at each frame. The output of this layer goes to one more convolutional layer to generate heat maps. Finally, a deconvolution layer up-samples the heat map to the desired spatial size.

Conventional recurrent units are capable of processing temporal data however, their architecture is not suitable for working on images/feature maps for two reasons. 1) weights matrix size, 2) ignoring spatial connectivity. Assume a case where a recurrent unit is placed after a feature map with the spatial size of $h \times w$ and have a number of channels c. After flattening, it will turn into a $c \times h.w$ long matrix. Therefore, weights of the recurrent unit will be of size $c \times (h.w)^2$ which is power four of spatial dimension. These matrices for weights can only be maintained for small feature maps. Even if the computation was not an issue, such design introduces too much variance in the network which prevents generalization. In Convolutional recurrent units, similar to regular convolutional layer, weights are three dimensional and they convolve with the input instead of dot product. Accordingly, the cell's model, in the case of a GRU architecture, will turn into equations 3.1 where the dot products are replaced with convolutions. In this design, weights matrices are of size $k_h \times k_w \times c \times f$ where k_h, k_w, c and f are kernel's height, kernel's width, number of input channels, and number of filters, respectively. In Figure 2.4 the operations applied on the input and the previous step will all be convolutions instead. Since we can assume spatial connectivity in feature maps, kernel size can be very small compared to feature map's spatial size. Therefore, this architecture is much more efficient and weights are easier to learn due to smaller search space.

We employ this approach for segmentation in a fully convolutional network. It is possible to apply this layer on either heat maps or feature maps. In the first case, the output of this layer will directly feed into the deconvolution layer and produces the pixel-wise probability map. In the latter case, at least one CNN layer needs to be used after the recurrent layer to convert its output feature maps to a heat map.

RFC-VGG in the Table 3.1 is an example of the second case. It is based on VGG-F [68] network. Initializing weights of our filters by VGG-F trained weights, alleviates over-fitting problems as these weights are the result of extensive training on the imagenet. The network is cast to a fully convolutional one by replacing the fully connected layers with convolutional layers. The last two pooling layers are dropped from VGG-F to allow a finer segmentation. Then a convolutional gated recurrent unit is used followed by one convolutional layer and then deconvolution

for up-sampling. Figure 3.2 shows the detailed architecture of RFC-VGG.

$$z_t = \sigma(W_{hz} * h_{t-1} + W_{xz} * x_t + b_z) \tag{3.1a}$$

$$r_t = \sigma(W_{hr} * h_{t-1} + W_{xr} * x_t + b_r) \tag{3.1b}$$

$$\hat{h_t} = \Phi(W_h * (r_t \odot h_{t-1}) + W_x * x_t + b) \tag{3.1c}$$

$$h_t = (1 - z_t) \odot h_{t-1} + z \odot \hat{h_t} \tag{3.1d}$$

3.2 Experiments

This section presents our experiments and results. First, we describe the datasets that we used then, we discuss our training methods and hyper-parameters settings. Finally, quantitative and qualitative results are shown.

All the experiments are performed on our own implemented library. To our knowledge, there is no open source framework that accommodates RFCNN architecture. Therefore, we built our own library on top of Theano [5] to create arbitrary networks that have can have FCN as a recursive module. Key features of this implementation are: **(1)** Supports networks with the temporal operation for images. The architecture can be any arbitrary CNN and an arbitrary number of recurrent layers. The network supports any length input. **(2)** Three gated architecture, LSTM, GRU, and Conv-GRU is available for the recurrent layer. **(3)** Deconvolution layer and skip architecture to support segmentation with FCN.

3.2.1 Datasets

In this paper four datasets are used: 1) The Moving MNIST. 2) Change detection [23]. 3) Segtrack version 2 [43]. 4) Densely Annotated VIdeo Segmentation (Davis) [53]. Figure 3.4 shows samples from the latter two.

The Moving MNIST dataset is synthesized from original MNIST by moving the characters in random but consistent directions. The labels for segmentation is generated by thresholding input images after translation. We consider each translated image as a new frame. Therefore we can have arbitrary length sequence of images.

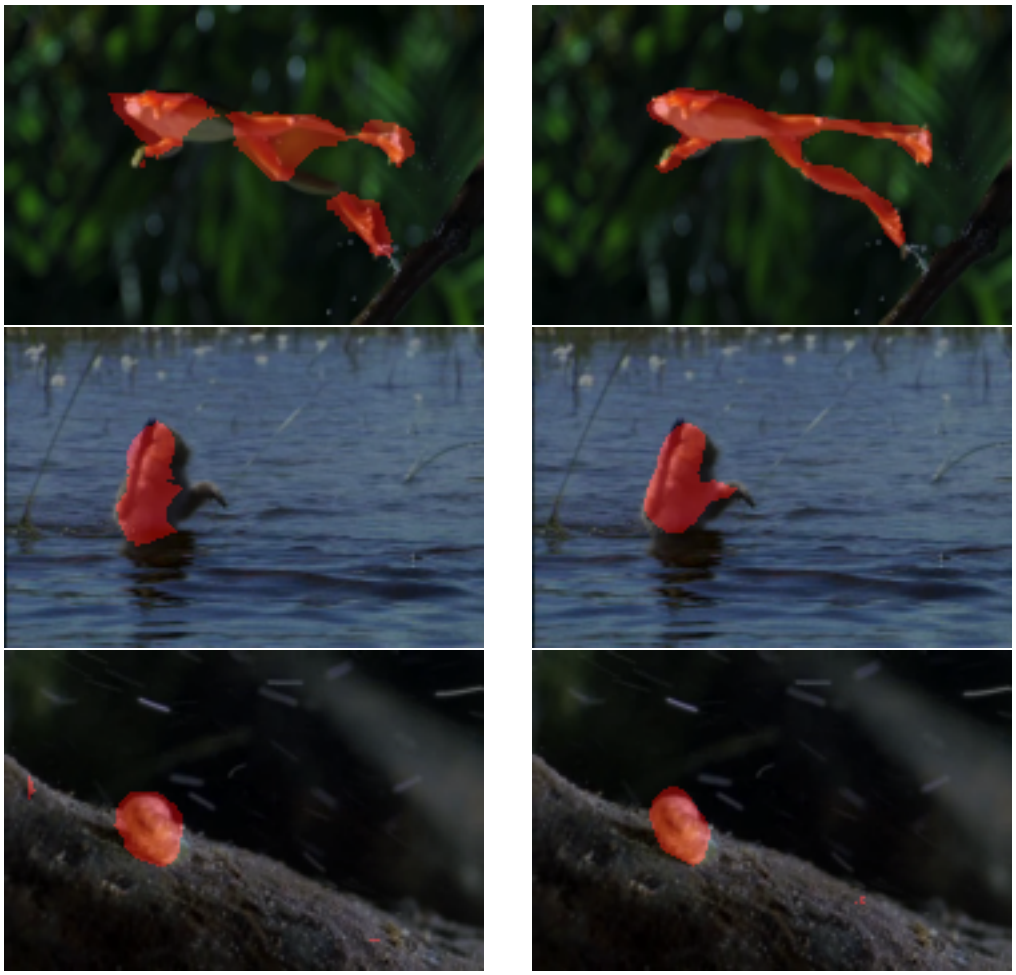

Figure 3.3: Qualitative comparison between FCN and RFCN on SegtrackV2

Change Detection Dataset [23] This dataset provides realistic, diverse set of videos with pixel-wise labeling of moving objects. The dataset includes both indoor and outdoor scenes. It focuses on moving object segmentation. In the motion detection, we were looking for videos that have similar moving objects, e.g. cars or humans so that there would be semantic correspondences among sequences. Accordingly, we chose six videos: Pedestrians, PETS2006, Badminton, CopyMachine, Office, and Sofa.

SegTrack V2 [43] is a collection of fourteen video sequences with objects of interest manually segmented. The dataset has sequences with both single or multiple objects. In the latter case, we consider all the segmented objects as one and we perform foreground segmentation.

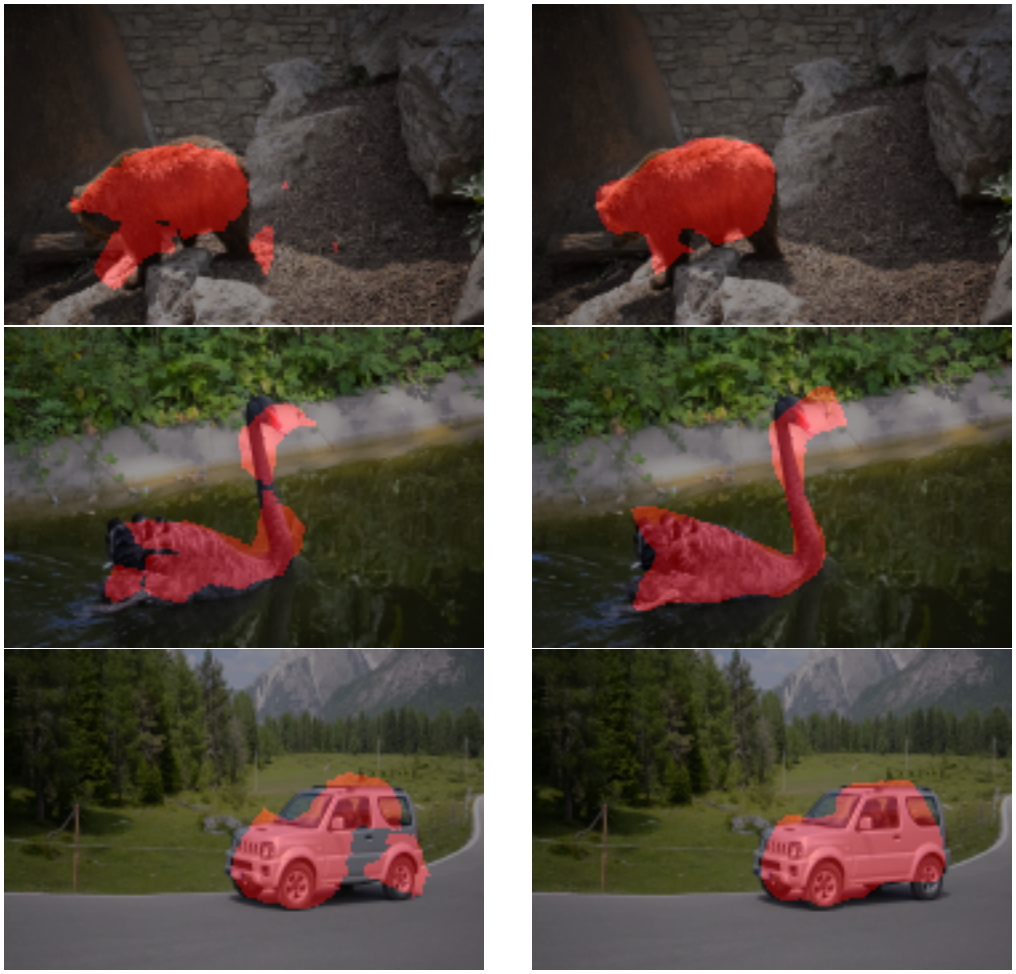

Figure 3.4: Qualtitative results over Segtrack V2 and Davis datasets, where top image has overlay of FC-VGG and bottom has RFC-VGG segmentation.

Table 3.1: Details of proposed networks. $F(n)$ denotes filter size of $n \times n$. $P(n)$ denotes total of n zero padding around the feature map. $S(n)$ denotes stride of length n for the convolution. $D(n)$ denotes number of output feature maps from a particular layer n for a layer (number of feature maps is same as previous layer if D is not mentioned).

Network Architectures					
RFC-Lenet		RFC-12s		RFC-VGG	
input: 28×28		input: 120×180		input: 240×360	
Recurrent Node	Conv: F(5), P(10), D(20)	Recurrent Node	Conv: F(5), S(3), P(10), D(20)	Recurrent Node	Conv: F(11), S(4), P(40), D(64)
	Relu		Relu		Relu
	Pool 2×2		Pool 2×2		Pool 3×3
	Conv: F(5), D(50)		Conv: F(5), D(50)		Conv: F(5), P(2) D(256)
	Relu		Relu		Relu
	Pool(2×2)		Pool(2×2)		Pool(3×3)
	Conv: F(3), D(500)		Conv: F(3), D(500)		Conv: F(3), P(1) D(256)
	Relu		Relu		Relu
	Conv: F(1), D(1)		Conv: F(1), D(1)		Conv: F(3), P(1) D(256)
	-		-		Relu
					Conv: F(3), P(1) D(256)
					Relu
					Conv: F(3), D(512)
					Conv: F(3), D(128)
	DeConv: F(10), S(4)		Flatten		ConvGRU: F(3), D(128)
	Flatten		GRU: W(100×100)		Conv: F(1), D(1)
	GRU: W(784×784)		DeConv: F(10), S(4)		DeConv: F(20), S(8)

Davis [53] dataset has fifty high resolution and densely annotated videos with pixel accurate groundtruth. The videos include multiple challenges such as occlusions, fast motion, nonlinear deformation and motion blur.

Synthia [61] is a synthetic semantic segmentation dataset for urban scenes. It contains pixel level annotations for thirteen classes. It has over 200,000 images with different weather conditions (rainy, sunset, winter) and seasons (summer, fall).

Since the dataset is large only a portion of it from Highway sequence for summer condition is used for our experiments.

CityScapes [17] is a real dataset focused on urban scenes gathered by capturing videos while driving in different cities. It contains 5000 finely annotated 20000 coarsely annotated images for thirty classes. The coarse annotation includes segmentation for all frames in the video and each twentieth image in the video sequence is finely annotated. It provides various locations (fifty cities) and weather conditions throughout different seasons.

3.2.2 Results

The main experiments are conducted using Adadelta [80] for optimization that practically gave much faster convergence than standard stochastic gradient descent. The logistic loss function is used and the maximum number of epochs used for the training is 500. The evaluation metrics used are precision, recall, F-measure and IoU, shown in equation 3.2 and 3.4 where tp, fp, fn denote true positives, false positives, and false negatives respectively.

$$precision = \frac{tp}{tp + fp}, recall = \frac{tp}{tp + fn} \tag{3.2}$$

$$F - measure = \frac{2 * precision * recall}{precision + recall} \tag{3.3}$$

$$IoU = \frac{tp}{tp + fp + fn} \tag{3.4}$$

Table 3.2: Comparison of RFC-VGG with its baseline counterpart on DAVIS and SegTrack

		Precision	Recall	F-measure	IoU
SegTrack V2	FC-VGG	0.7759	0.6810	0.7254	0.7646
	RFC-VGG	**0.8325**	**0.7280**	**0.7767**	**0.8012**
DAVIS	FC-VGG	0.6834	0.5454	0.6066	0.6836
	RFC-VGG	**0.7233**	**0.5586**	**0.6304**	**0.6984**

In this set of experiments, a fully convolutional VGG is used as a baseline denoted as FC-VGG. It is compared against the recurrent version RFC-VGG. The initial five convolutional layers are not finetuned and are from the pretrained VGG architecture to avoid overfitting the data. The results of the experiments on SegTrackV2 and Densely Annotated VIdeo Segmentation (DAVIS) datasets are provided in 3.2. In these experiments, the data of each sequence is divided into two splits with half as training data and the other half as keep out test data. It is apparent in the results that RFC-VGG outperforms the FC-VGG architecture on both datasets with about 3% and 5% on DAVIS and SegTrack respectively.

Figure 3.4 shows the qualitative analysis of RFC-VGG against FC-VGG. It shows that utilizing temporal information through the recurrent unit gives better segmentation for the object. This can be explained due to the implicit learning of the motion that segmented objects undergo in the recurrent units. It also shows that using conv-GRU as the recurrent unit can enable the extraction of temporal information from feature maps due to the reduced parameter set. Thus the recurrent unit can learn the motion pattern of the segmented objects by working on richer information from these feature maps. It's also worth noting that the performance of the RFCN network depends on its baseline fully convolutional counter part. Thus with an enhanced fully convolutional network using skip architecture, to get finer segmentation, the recurrent version should improve as well.

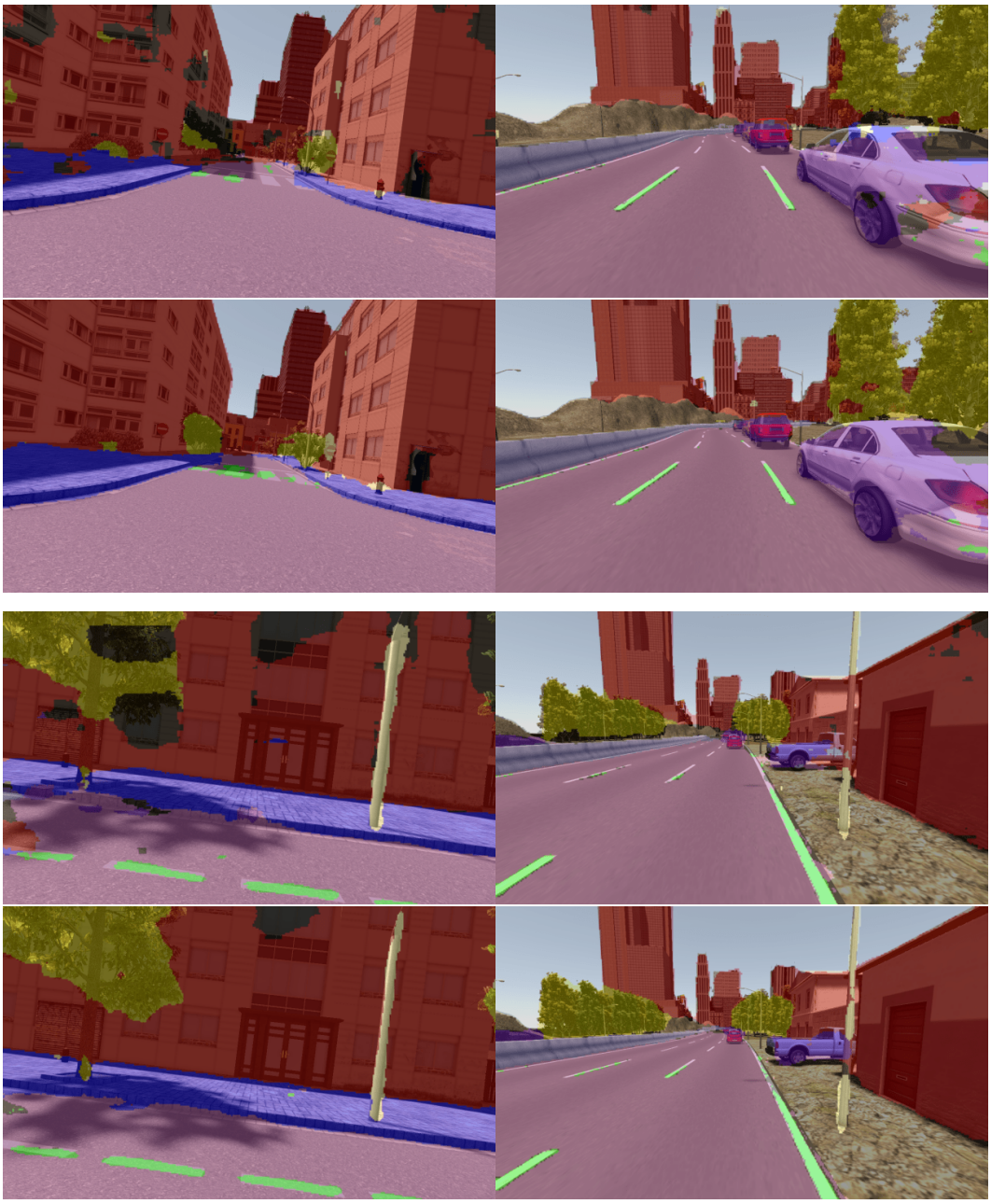

Figure 3.5: Qualitative results of experiments with Synthia, where network prediction are overlaid on the input. Odd rows are FC-VGG output and Even rows are their corresponding RFC-VGG outputs.

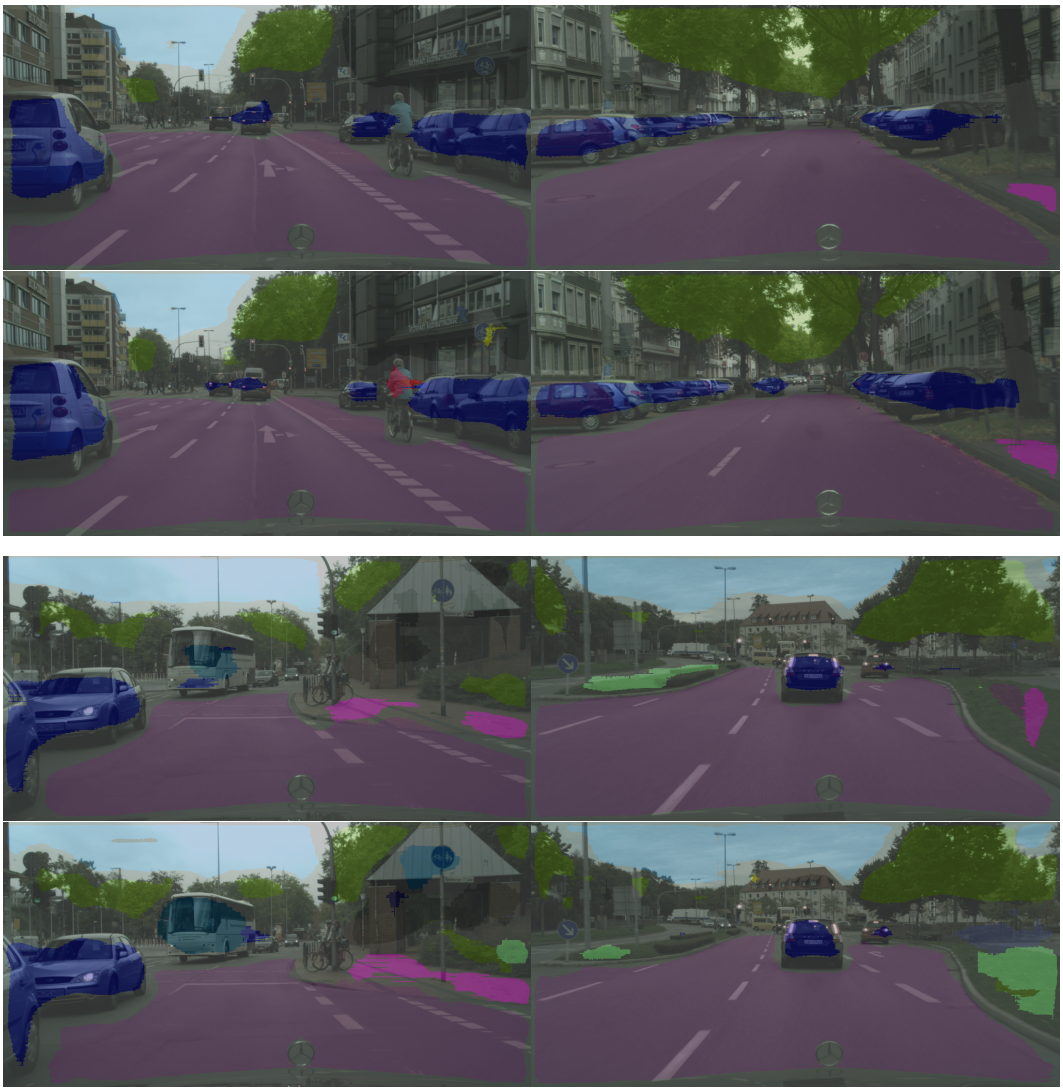

Figure 3.6: Qualitative results of experiments with cityscapes datasets, where network prediction are overlaid on the input. Odd rows are FCN-8s and even rows are their corresponding: RFCN-8s output.

Table 3.3: Semantic Segmentation Results on Synthia Highway Summer Sequence for RFC-VGG compared to FC-VGG

	Mean Class IoU	Per-Class IoU								
		Car	Pedestrian	Sky	Building	Road	Sidewalk	Fence	Vegetation	Pole
FC-VGG	0.755	0.504	0.275	0.946	0.958	0.840	0.957	0.762	0.883	0.718
RFC-VGG	**0.812**	**0.566**	**0.487**	**0.964**	**0.961**	**0.907**	**0.968**	**0.865**	**0.909**	**0.742**

Table 3.4: Semantic Segmentation Results on CityScapes for RFCN-8s compared to FCN-8s

	Category IoU	Per-category IoU				
		Flat	Nature	Sky	Construction	Vehicle
FCN-8s	0.53	0.917	0.710	0.792	0.683	0.585
RFCN-8s	**0.565**	**0.928**	**0.739**	**0.814**	**0.719**	**0.652**

The same architecture was used for semantic segmentation on synthia dataset after modifying it to support the thirteen classes. A comparison between FC-VGG and RFC-VGG is presented in terms of mean class IoU and per-class IoU for some of the classes. Table3.3 presents the results on synthia dataset. RFC-VGG has 5.7% over FC-VGG in terms of mean class IoU. It also shows the per-class IoU generally improves in the case of RFC-VGG. Interestingly, the highest improvement is with the car and pedestrian class that benefits the most from a learned motion pattern compared to sky or buildings that are mostly static. Figure3.6 first row shows the qualitative analysis on Synthia. The second image shows the car's enhanced segmentation with RFC-VGG.

Finally, experimental results on cityscapes dataset using FCN-8s and its recurrent version RFCN-8s is shown in Table3.4. It uses mean category IoU and per-category IoU for the evaluation. It clearly demonstrates that RFCN-8s outperforms FCN-8s with 3.5% on mean category IoU. RFCN-8s generally improves on the per-category IoU, with the highest improvement in vehicle category. Hence, again the highest improvement is in the category that is affected the most by temporal data. Figure 3.6 bottom row shows the qualitative evaluation on cityscapes data to compare FCN-8s versus RFCN-8s. The third image clearly shows that the moving bus is better segmented with the recurrent version. Note that the experiments were conducted on images with less resolution than the original data and with a reduced version of FCN-8s due to memory constraints. Therefore, finer categories such as human and object are poorly segmented. However, using original resolution will fix this problem and its recurrent version should have better results as well.

3.2.3 Further Analysis

In this section, we present our experiments using conventional recurrent layers for segmentation. These experiments provide further analysis on different recurrent units. They also compare end-to-end architecture versus the decoupled one. We use moving MNIST and change detection datasets for this part.

Images of MNIST dataset are relatively small (28×28) which allows us to test our RFC-Lenet(a) network 3.1. A fully convolutional Lenet is compared against RFC-

Lenet(a). Table 3.5 shows the results that were obtained. The results of RFC-Lenet with GRU is better than FC-Lenet with 2% improvement. But segmentation of MNIST characters is an easy task as it ends up to learning thresholding. Note also that GRU gave better results than LSTM.

Table 3.5: Precision, Recall, and F-measure on FC-Lenet, LSTM, GRU, and RFC-Lenet tested on synthesized MNIST dataset

	Precision	Recall	F-measure
FC-Lenet	0.868	**0.922**	0.894
LSTM	0.941	0.786	0.856
GRU	0.955	0.877	0.914
RFC-Lenet	**0.96**	0.877	**0.916**

The second set of experiments is conducted on real data from the motion detection benchmark using RFC-12s. Throughout these experiments, the training constitutes 70% of the sequences and 30% for the test set. The experiments compared the recurrent convolutional network trained in an end-to-end fashion denoted as RFC-12s with its baseline, the fully convolutional network FC-12s. It is also compared against the decoupled training of the FC-12s and the recurrent unit. Where GRU is trained on the probability maps output from FC-12s. Table 3.6 shows the results of these experiments, where the RFC-12s network had a 1.4% improvement over FC-12s. We observe less relative improvement compared to using Conv-GRU because in regular GRU spatial connectivities are ignored. However, incorporating the temporal data still helped the segmentation accuracy.

3.3 Discussion

We proposed a recurrent fully convolutional network for video segmentation. The designed network were tested on several datasets and improved the results on all of them. In the list below different choices for hyper-parameters and their effect on the results are analyzed.

Table 3.6: Precision, Recall, and F-measure on architectures FCN-12s, GRU pre-training on coarse map from FCN-12s, RFC-12s on six sequences from motion detection benchmark on the test set. (D) and (EE) indicate the decoupled and the end-to-end integration of recurrent units with the FCN, respectively.

	Precision	Recall	F-measure
FC-12s	0.827	0.585	0.685
RFC-12s (D)	**0.835**	0.587	0.69
RFC-12s (EE)	0.797	**0.623**	**0.7**

- **Number of Layers**: It is known that increasing number of layers may cause overfitting. Yet, the network should have a sufficient depth to be able to learn the desired task. As the dataset gets more challenging (more classes, smaller regions, more variance in images) the minimum required depth increases. For example, in the relatively simple SegTrackV2 (binary segmentation, dominant ROI), even the shallow LeNet could get a good performance. Cityescapes and Synthia needed significantly deeper networks for a decent performance. RFCN and FCN acted similarly in this regard.

- **Location of the Recurrent Layer**: We varied the position of the recurrent layer from the third layer all the way to last layer. This choice has a significant impact on both computation and performance. On early layers where the feature maps spatial size are large, recurrent layer becomes too expensive to train. However, we suspect that it could have better captured the dynamics. Because, multiple pooling layers will make the output insensitive to locations of the objects in the image which contradicts with extracting dynamics. Generally, we saw that putting the recurrent layer earlier helps performance given that it is trained properly (it takes much longer to train as it gets closer to the firs layer). We were restricted for the choice of this layer by the skip architecture. The recurrent layer should come either after all the skip layers or before the first one.

- **Size of the Moving Window**: We tested windows sizes between 1 to 5 (size 1

would be just an extra convolutional layer with a gate. We observed marginal change for size 1, as expected. As the size grows, the performance improves at the cost of more difficult training which eventually reduces the performance. We saw a peak at window size of 3. To jump start the training for larger window sizes, we used trained recurrent layers from networks with smaller window size as initial weights.

- **Training FCN vs RFCN**: RFCN networks took about twice as much time as FCN networks to train. Both networks were trained better using Adadelta vs SGD. We did not see much sensitivity to the batch-size in either one. Interestingly, we observed that once RFCN gets to a reasonable accuracy, it tends to improve steadier and smoother than FCN.

RFCN architecture is very generic and allows many extensions. It can be used to incorporate non-visual series. It can prove very useful for mobile robots or self-driving car where accurate pose information is available from their SLAM system. Network can potentially learn a next image predictor from this pose information. To enforce RFCN to learn the dynamics, one can add optical flow as an auxiliary objective. This task can be learn simultaneously with the main segmentation objective.

Chapter 4

Incremental Learning through Human Robot Interaction

A current research aim is to develop robot assistants for helping humans in everyday manipulation tasks. However, only a few robot platforms have been successful in home environments. Commercial robots have been designed to perform a specific task, e.g., robot vacuums, robot lawnmowers. By contrast, multi-purpose robots, that can perform a broad variety of household tasks are still not realized. One key reason is that current robots do not interact well with humans. By considering HRI as a core component during the system design, it will be easier to integrate robots within humans' daily activities. Yanco etal [79] conducted a study during the recent DARPA robotics challenge [55]. Their results show that although every team used similar methods and technologies, one of the main reasons for different performance was the quality of the human-robot interface presented to the human operators during the challenge. Researchers have realized this and instead of aiming for autonomy, have shifted focus to the user interaction focusing on the human-in-the-loop paradigm. In this paradigm, the human's knowledge and guidance are used whenever the robot cannot make a decision on its own [25, 40, 58]. To provide this guidance, it is important to have the capacity to establish a common ground knowledge (discourse) shared between the human and robot, just as humans do. For example, a new apprentice in a metal workshop needs to quickly get familiar with different types of materials (Steel, bronze, nylon, etc.) and machines (Lathe, drilling, milling, boring, shaping etc.). Otherwise, it will be difficult to understand

the information that is passed to it. Therefore, a more experienced worker tries to first establish a common ground knowledge, like names of tools and materials, by interacting with the apprentice.

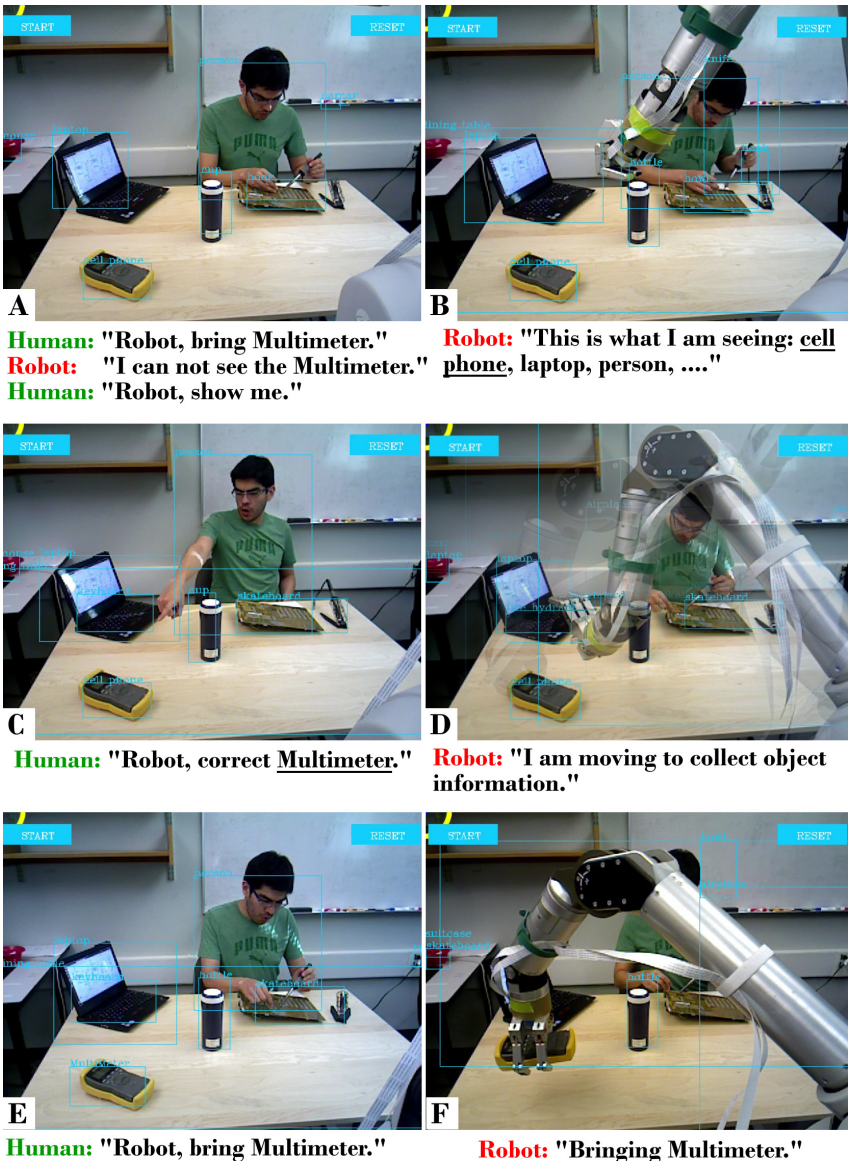

Figure 4.1: **Incrementing robot knowledge through HRI:**
A) The human asks the robot to bring the multimeter while working on a circuit board. The robot does not know what is a multimeter. The human asks for the robot's world representation.
B) The robot iterates through the detected objects by pointing and saying each object label.
C) The human points and corrects the "multimeter" label which was initially recognized as a "cell phone".
D) The robot gets close to the pointed object and collects images of the corrected object.
E,F) The human asks again to bring the multimeter. This time the robot succeeds in his task. Demonstration of our interface can be seen in the supplementary video [3].

The same principals should also be considered for robots. However, robots need

to have a basic world understanding before they can start learning from guidance. This basic understanding can be in terms of object localization and object class recognition. Deep learning was shown to outperform many classic computer vision approaches in this matter. Overfeat [65] was one of the earliest attempts. A convolutional neural network (CNN) was used to generate bounding boxes of each object and label them according to their class. This work used a regression network to predict a possible bounding box for an object given a grid of proposed regions with different scales. In this approach, increasing number of region proposals proved to be both crucial for accuracy and at the same time, unfortunately, computationally expensive. Therefore, more recent attempts [60] [33] utilized Region Proposal Networks (RPN) to regress bounding boxes. RPN can operate on feature maps instead of the input image and by doing so, it bypasses the need for recomputing the feature maps.

CNN models have achieved the state of the art in object recognition on a wide variety of datasets. However, their implicit assumption is that all the possible object categories are included in the dataset for batch off-line training. Unfortunately, real world recognition is different. At prediction time, the algorithm will frequently face objects that are not in the training data or they can look very different compared to training examples. Accordingly, the need for updating robot's knowledge over time arises. In the literature these problems are addressed by Incremental Machine Learning (IML) [18, 54] and Open Set Recognition(OSR) [6, 11, 63, 64]. The main focus of the IML is to handle new instances of known classes. OSR methods, however, need to deal with two more challenges. One is to continuously detect novel categories and two is to update the method so that it will include the new category. These are difficult problems to solve, especially, recognizing an object as a novel class since it could involve a long reasoning chain. For example, a sugar box and a detergent box look very similar and without semantic cues like reading the label or using the context that they are in, it is near impossible to differentiate them. However, we can utilize a robot's discourse as a solution to this problem. In particular, robots can interact with humans and get guidance or instructions from them. They are also able to explore the environment using an onboard camera.

Along with this general direction, we propose a new method to improve the robot's visual perception incrementally. Our purpose is to recognize and localize objects. Furthermore to have the capacity of learning new objects and correcting false interpretations through HRI. Our contribution is two-fold. (1) A deep learning based localization and recognition method that uses our robotic-vision interface to incrementally improve its object knowledge through interaction with a human. (2) A robot-vision system capable of interacting naturally with a human to establish a common ground knowledge of the objects in a shared environment.

4.1 Incremental Learning from Humans

In the literature, robot learning from humans is commonly named as Programming by Demonstration (PbD), Learning from Demonstration (LfD) or Imitation Learning. It should be contrasted with the automatic learning methods that do not involve active human interaction during the learning process such as reinforcement learning. The main body of work in this field focuses on learning motion trajectory from user demonstrations to perform tasks. While improving the robot's perceptual prowess is neglected in the field, still, some of the fundamental challenges are shared. In particular, interfaces for information exchange and methods for incremental learning. A brief summary of research on these challenges is provided below.

Since the main focus in the field was to learn trajectories that were required to do actions, most interfaces are optimized to capture teacher's trajectories. Tracking human body motion [39] [73] [36] [49], using kinesthetic teaching where the human guides the robot to perform the task [62] and teleoperation where the teacher controls robots effectors directly through an interface [16] [26] are the main approaches for this purpose. Even though these interfaces are good for recovering the demonstrated trajectory, they are not natural in terms of human interaction. So other, more natural, interfaces were developed. Using speech [12] [20] and gestures [27] as

additional information are among these. In most incremental learning methods, the robot uses new demonstration to bootstrap its current skill and gradually improving it [82].

4.1.1 Deep Learning for Robotics

The success of deep learning in computer vision has inspired some application in robotics. One of the main challenges of using robots in an unrestricted environment was the lack of robust and generic algorithms. Deep learning (DL) methods showed that now it is possible to achieve accurate and robust performance in many applications. Particularly, image classification and natural language processing have benefited greatly from this new technology.

DL models can be used as a pre-processing module, mostly on images and language, that converts raw sensor data into a lower dimensional feature space that the can be used for control. Detecting robotic grasp [41] is using raw images to detect grasping point for various objects which can later be used for grasping. DL can also be used in an end-to-end fashion where the network is responsible for processing raw input all the way to generating the control signal for the robot's effectors. Levine et al. [42] implemented such network and demonstrated its feasibility by completing a few tasks. The idea of end-to-end networks for control is appealing, as in principle, it alleviates the need for detailed engineering for each module. However, it became apparent that in these networks many other hand tunings of hyperparameters and special care for training is needed and at the end, they are very limited in practice. The main reason is that all the approaches that have proposed so far, need much more training data than what is practically viable. This forced the designer to come up with training speed-ups such as learning from demonstrations techniques. Moreover, it is yet to be shown that deep network are capable of learning highly complex decision making needed in robots even if enough data is available. Therefore, we think that using DL enablers as modules is more practical at the moment. Because, these modules have trained on appropriate amount of data for a particular task and have a decent performance on it.

Object Localization and Detection Networks

The goal of object localization and detection methods is to localize all the objects in a scene, commonly by bounding boxes and label them according to their class. Such methods are quite useful for scene understanding and many applications of them can be envisioned in robotics. Overfeat [65] was one of the earliest attempts for the used CNN for this task. This work used a regression network to predict a possible bounding box for an object given a grid of proposed regions with different scales. In this approach, increasing number of region proposals proved to be both crucial for accuracy and at the same time, computationally expensive. Therefore, more recent attempts [60] [33] utilized Region Proposal Networks (RPN) to regress bounding boxes. RPN can operate on feature maps instead of the input images and by doing so, it bypasses the need for recomputing the feature maps. The details of this method will be further discussed later.

4.1.2 Multi-Class Open Set Recognition

CNN models have achieved the state of the art in object recognition on a wide variety of datasets. Their implicit assumption is that all the possible object categories are included in the dataset. However, the situation in the real world recognition is widely different. At prediction time, the algorithm will face objects that were not in the training datasets. This is especially important for household and assistive robots. Objects in each household and workplace are different and having an inclusive dataset of all of these objects is not possible. The problem is not only about objects. For example, an assistant robot in a hospital should treat each patient differently and it needs to recognize the patient to do so. But, the current recognition methods cannot do better than recognizing the patient only as a person.

This problem is addressed by Incremental Machine Learning(IML) and Open Set Recognition(OSR) in the literature. Main focus of the IML [54] [18] is to handle new instances of known classes. OSR methods [64] [63] [6] however, need to deal with two more challenges. The first is to continuously detect novel categories and the second is to update the method so that it will include the new category.

4.2 system

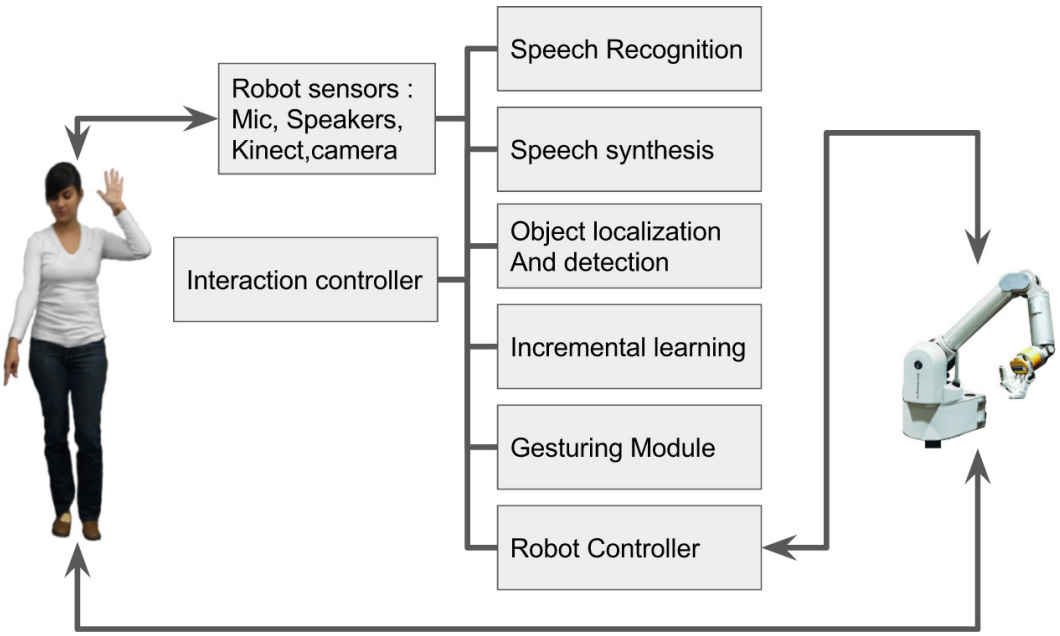

Figure 4.2: System block diagram.

Our system uses a 7-DOF WAM arm [2] instrumented with eye-in-hand-cameras, a microphone, Kinect camera and speakers. It is composed of 7 modules as shown in Fig. 4.2. All modules are fully integrated with ROS [56].

- The **speech recognition module** integrates the CMU Sphinx toolkit [1]. It provides basic word and sentence recognition used by the robot to shift states during the interaction.

- The **speech synthesis module** relies on the Festival speech synthesis system [8] and provides feedback to the human in a verbal channel.

- The **object localization and detection module** provides labels and 2D locations of the objects in the scene. For a detail description see section 4.3.

- **The Incremental learning module** uses HRI, to permit changes in the robot's world representation. For a detail description see section 4.3.

- The **gesturing module** is based on our previous work [57,59], where we proposed a non-verbal robot-vision system capable of inferring human pointing

and perform simple pick and place tasks based on human gesture commands. We integrate this capability into our system to reduce speech description and make the interaction more human like.

- The **robot controller module** commands robot movements and generates: pointing gestures, robot data collection and pick-up object actions [59].

- The **interaction controller module** is in charge of orchestrating the complete system. It supports the different interactions that are shown in Fig. 4.1 and it is based on a finite state machine that is triggered by gesture or speech coming from the human and/or robot.

In Fig. 4.1 four important interactions of our system are highlighted. **Verbal interaction**, by using both speech recognition and synthesis the human and the robot can establish basic verbal communication. The **world representation** refers to both verbal and gesture interaction performed by the robot. The recognition and localization module provides 2D bounding boxes and labels of the detected objects. The system verbally informs the object class and at the same time the robot arm points to the object 3D centroid. The centroid is calculated by using the RGB to depth camera correspondence from the objects bounding box (See Fig. 4.3). In the **correct interaction**, human uses both verbal and gesture communication to annotate a particular object in the scene that needs to be corrected. Fig 4.3 shows both RGB and Point cloud visualization. The head and hand of the human are tracked as two points. With a verbal triggering, a 3D ray is constructed from these two points, hitting the target object. The robot is then commanded to collect data with this 3D object location. The data collection is performed through the eye-in-hand camera by moving the end-effector in a parameterized helix curve keeping the camera facing to the object location. During the collection state, a TLD tracker [34] is used to guarantee the cropping of the object during the data collection (See Fig.4.6). The initial bounding box of the tracker is given as the whole image when the robots starts the collection close to the object.

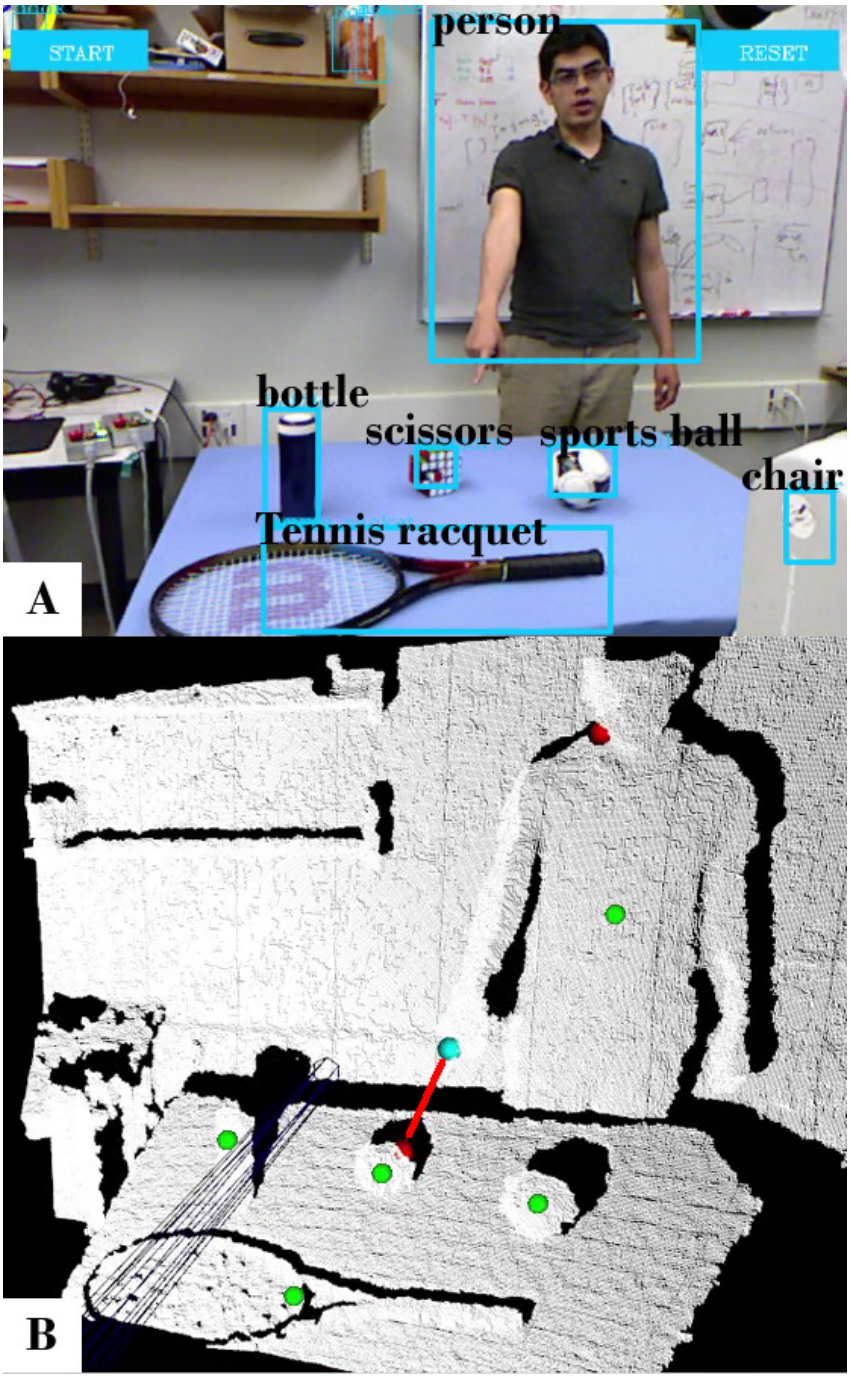

Figure 4.3: A) RGB visualization. Objects in the scene are detected and localized. The human points to the rubik's cube to correct its label. *Note: the font and the line-width of the bounding boxes are enlarged from the original image for clarity.* B) Point cloud visualization. Using the 2D to 3D correspondence, 2D bounding boxes centroids are used to find 3D objects centroids (green spheres).

4.3 Methodology

An interaction example using our proposed interface is shown in Fig. 4.1. Our example shows a person soldering a circuit board. During this work, he needs to use a multimeter, that is out of his reach. He asks his robot assistant to bring the multimeter. (Fig. 4.1A). The robot is equipped with a recognition module. The multimeter class was not found as a recognizable object. The person then asks the robot what it sees (given its current world representation) (Fig. 4.1A), to figure out if the robot is detecting the object under another name/category or if it does not detect it at all. Through speech, the robot communicates the type of detected objects in the scene, and by using pointing gestures the robot provides the objects' locations (Fig. 4.1B). Pointing allows the robot to skip complex spatial sentences, (e.g. "from my point of view the laptop is on the left top corner of the table"). After communicating which objects were recognized and localized by the robot the human can interact through gestures and speech to correct or add a particular object (Fig. 4.1C), in this case, the multimeter. After the addition or correction, the robot collects several images on-line of the target object (Fig. 4.1D) which is used to modify the current world representation. This procedure needs to be done only once for each new class but can be repeated to achieve better performance. Finally, the person asks again to bring the multimeter (Fig. 4.1E). This time the robot succeeds on recognizing the object and executes a picking task to bring the multimeter. (Fig. 4.1F).
In the following two subsections, we describe our localization and recognition network and our open-set recognition approach.

4.3.1 Localization and Recognition Network

The network can be divided into three parts, see Fig.4.4. The first part is a fully convolutional network that extracts feature maps from the input image. Second, is the localization network that takes feature maps from the first part and finds the possible bounding boxes of objects and objectness score corresponding to each bounding box. The third part is the recognition network. It takes a fixed size feature map input corresponding to each bounding box and produces predictions for each bounding

box.

We used the first 30 layers of vgg-16 [68] (counting pooling and activation layers), trained on image-net [38] for the first part of the network. We chose vgg-16 due to its state of the art performance in object recognition.

The structure of our localization network is based on the Densecap [33] which in turn is a modified version of the Faster RCNN [60]. In this model, the localization receives a feature map that is computed by a convolutional layer. Using this feature map and convolutional anchors, a regression network finds the transformation that is required to take an anchor to a bounding box of an object. Convolutional anchors can be viewed as a fixed and multi-scale proposed bounding boxes in the image space that are centered to the spatial correspondence of a pixel in the feature map. Using this scheme, greatly improves the inference time. After predicting a bounding box, the recognition network finds a fixed size feature map corresponding to that bounding box. It does so by first, projecting the bounding box into feature map space, then transferring the arbitrary size selection of the feature maps into a fixed size, using bi-linear interpolation. The network also has a regressor branch that predicts the objectness of the fixed size feature map.

For the recognition network, we again used the last three fully connected layer of vgg-16 as our recognition structure with the weights initialized by the weights of a trained vgg-16 model. Same as the Densecap, we predict the objectness score and position of bounding box one more time in the recognition network. Lastly, this network predicts the category of the object inside of each bounding box.

For the base training of our model, we used Microsoft COCO dataset [45]. This dataset consists of about 328 thousand images with 2.5 million labeled instances of 80 common objects in their common context. We used the objects' bounding box data and their category as our supervised data. The loss function is defined in 4.1. In this equation $\hat{y}^{bb1}$, $\hat{y}^{bb2}$,$\hat{y}^{o1}$, $\hat{y}^{o2}$, $\hat{y}^{c}$ are prediction for first and second bounding box regression, first and second objectness score regression and classification probability, respectively. y^{bb}, y^{o}, y^{c} are ground truth for locations of bounding boxes, binary value indicating the existence of an object and the one hot vector indicating the category, respectively. k is the number of proposed regions for each image and

$f_{logloss}$ is the logarithmic loss function.

$$
\begin{aligned}
F_{loss}(\hat{y}^{bb1}, \hat{y}^{bb2}, \hat{y}^{o1}, \hat{y}^{o2}, \hat{y}^{c}, y^{bb}, y^{c}) = w^{c} \sum_{i=1}^{k} f_{logloss}(\hat{y}_i^{c}, y_i^{c}) \\
+ w^{bb}(\sum_{i=1}^{k} ||\hat{y}_i^{bb1} - y_i^{bb}|| + \sum_{i=1}^{k} ||\hat{y}_i^{bb2} - y_i^{bb}||) \\
+ w^{o}(\sum_{i=1}^{k} f_{logloss}(\hat{y}_i^{o1}, y_i^{o}) + \sum_{i=1}^{k} f_{logloss}(\hat{y}_i^{o2}, y_i^{o}))
\end{aligned}
\tag{4.1}
$$

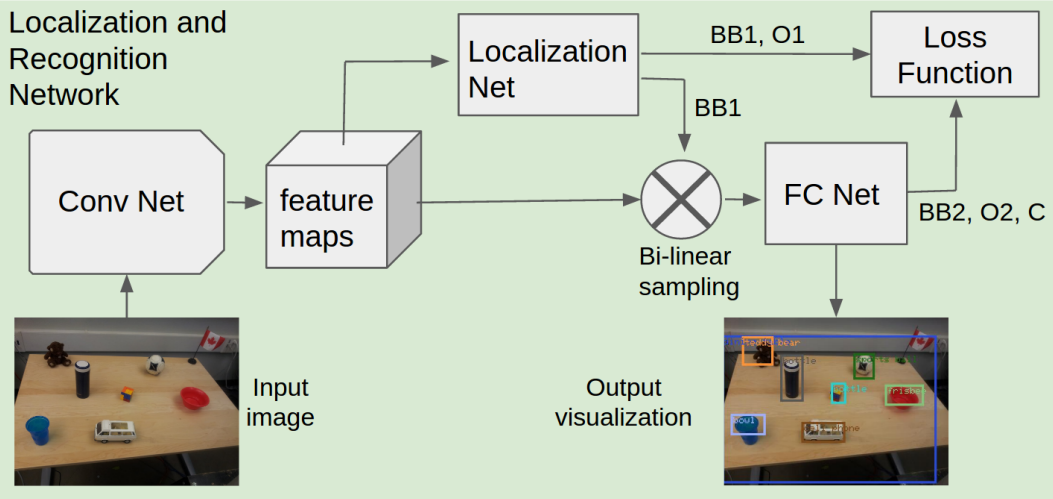

Figure 4.4: Main recognition and localization model. Components from left to right, Conv Net: first 30 layers of VGG16 [68] (counting pooling and activation layers). Localization Net: Object proposal and detection network based on Densecap [33]. FC Net: Last 6 fully connected layers of VGG16. Loss Function: explained in Eq.4.1

During the training, the weights of the first part of the network are kept constant. Since it is taken from a trained model, they are suitable for extracting features. The weights of the rest of the network are initialized with samples taken from a normal distribution with zero mean and 0.01 deviation [38]. As for the optimizer, we use ADAM [37] due to its easy tuning. The initial learning rate is set to 10^{-4}. To reduce inference time, we reshape all images to 400×400 which is smaller than conventional implementations and we also decreased the number of bounding box proposals to 200.

4.3.2 Open-Set Recognition Facilitated by Human Guidance

As mentioned before, discriminating between a novel and known objects is one of the main challenges in Open-set Recognition. However, human guidance can circumvent this problem. The user can reliably introduce novel objects to the system and therefore the incremental learning approach can use this guidance as the ground truth. Accordingly, in this section, we assume that reliable positive samples of a novel object is given to the incremental learning module through the user's interaction with the robot.

The incremental learning method has to have four main features for it to be feasible for a continuous and life long learning through HRI. 1) Allow Adding new classes. 2) Update its knowledge on already known classes as new human guidance comes in. 3) Preserve its learned knowledge after possibly many training phases. 4) Finally, it should not require storing the base data and new data inputs.

We investigated two approaches and compared their performance. **First Approach** We start off with our recognition network explained in 4.3.1 and trained with base dataset. We aim to improve it over time through HRI. To enable the network to recognize a new class, we modify the last fully connected layer that performs the classification by adding a new set of weights corresponding to the new class. Concretely, if we denote the weights of this layer as $\Theta_n = [\theta_1\theta_2...\theta_n]$ where θ_i is the ith column of the weight matrix then,

$$\Theta_{n+1} = [\theta_1\theta_2...\theta_n\theta_{n+1}]$$

θ_{n+1} is a randomly initialized vector. For adding the new category only the weights in this layer is modified and the rest of the network stays constant. Using one-vs-all classification scheme, the newly added weight vector is trained. We use the first part of the recognition and localization network to extract features from input images and using these features, we optimize the weights for the classification loss function. Stochastic Gradient Descent is used for training this layer.

To perform one-vs-all classification, in addition to positive samples, a set of negative samples is also needed. We extract a subset of images from MS-COC0 and images that were taken for the already added classes. We assign a probability for

drawing samples from each known class based on how likely it is that they get mistaken with the new class 4.2. In equation 4.2 C_i denotes the i-th class and X_{n+1} is the positive samples set. This sub-sampling procedure forces the classifier to discriminate better between similar classes.

It is expected that the recognition accuracy degrades as number of new objects increases. It is mostly due to the fact that the network is only partially trained for the new object. It is not a recommended approach in a batch dataset scenario where the data for all classes is available. However, as it is explained earlier, having an exhaustive dataset of all object is close to impossible. In addition, for most cases in robotics, a local representation of objects is enough, and even desired, since the robot is mostly interacting with one particular environment.

$$P_{draw}\{C_i\} = P\{pred(X_{n+1}) = C_i\} \tag{4.2}$$

Second Approach

Same as the first approach, we start off with a base recognition network but we employ a different incremental learning technique. Similar to [11], instead of storing the training data itself, we store the number of seen samples $\boldsymbol{C_i}$ for the ith class and the mean $\boldsymbol{\mu_i}$ and the standard deviation $\boldsymbol{\sigma_i}$ of their corresponding extracted convolutional features. These variables will be updated as a new sample comes in. If the new sample is from a novel class, a new node will be added to the last layer of the recognition network as shown in Fig. 4.5. $\boldsymbol{C}, \boldsymbol{\mu}, \boldsymbol{\sigma}$ and the network weights W will also be extended appropriately to include the new class. We use the loss function defined in 4.3. Where, $\{x, y\}$ is a new sample image and its label that is indicated by the user.

$$L = L_s(x, y, W) + L_m(\boldsymbol{C}, \boldsymbol{\mu}, \boldsymbol{\sigma}, W) \tag{4.3}$$

L_s 4.1 is the loss attributed to the incoming sample. o is the output of the Recognition network indicating the probability of the input x belonging to each class.

$$L_s = f_{logloss}(o(x, W), y) \tag{4.4}$$

Optimizing 4.4 pushes the network to correctly predict the new sample. However,

Algorithm 1 Incremental Learning

procedure INPUT: $x, y, W, \boldsymbol{C}, \boldsymbol{\mu}, \boldsymbol{\sigma}$.
for each new sample $\{x,y\}$ **do**:
    if y is a new class **then**
        append $\boldsymbol{C}, \boldsymbol{\mu}, \boldsymbol{\sigma}$ with a new element
        append a new output node to the network.
        Append the new node weights to W.
    end if
    $L = L_s(x, y, W) + L_m(\boldsymbol{C}, \boldsymbol{\mu}, \boldsymbol{\sigma}, W)$
    $W = W + adam(\frac{\partial L}{\partial W})$
    update $\boldsymbol{C}, \boldsymbol{\mu}, \boldsymbol{\sigma}$
end procedure

if only this loss is used, the network will soon start drifting and would forget the previously learned weights. To prevent this, a second loss with respect to the stored statistics is added in equation 4.5. The first summation over n (number of current classes) is weighted according to the number of seen sample for each class $\boldsymbol{C_i}$. The inner summation is the average logarithmic loss of the network from m independently and identically distributed (i.i.d) samples drawn from $\mathcal{N}(\mu,\, \sigma^2)$. Note that i is the label of the ith class. We realized that if only the loss w.r.t the $\boldsymbol{\mu_i}$ is measured, the weights will overfit to that overtime. So instead, we draw m random samples from the distribution and compute their average loss. This decreases the bias and prevents overfitting. We found m ranging between 50 to 200 works well. If chosen smaller it may cause instability during training and if larger, it would not be variant enough to prevent overfitting. The incremental learning procedure for each new batch of incoming data is delineated in the Algorithm 1.

$$L_m = \sum_{i=1}^{n} \boldsymbol{C_i} \frac{1}{m} \sum^{m} f_{logloss}(o(iid(\boldsymbol{\mu}_i, \boldsymbol{\sigma_i}^2), W), i) \tag{4.5}$$

4.4 Experiments

To validate our approach we tested three main components of our method. Namely, The object detection and recognition baseline, incremental learning algorithm and finally incremental learning through human guidance. Additionally, a human-human

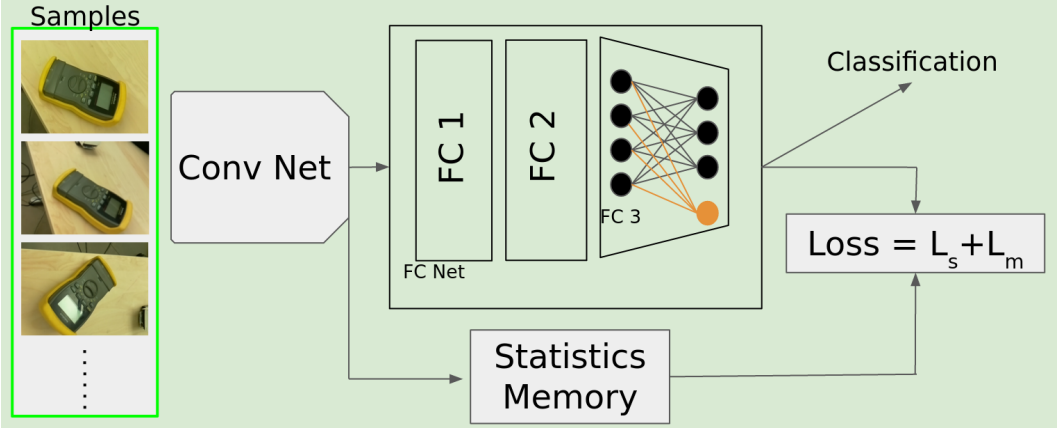

Figure 4.5: Incremental Learning model. Sample images from the HRI system are fed to the convolutional network and their features are extracted. Samples can be from a new class or a seen class. In case of the new class, a node and its weights will be added to the last fully connected layer. The memory is also extended. The features and the class labels will be used to first compute the loss with given the statistics memory. Then, the memory is updated.

interaction experiment was conducted to determine the gold standard for an object sorting and handling assistive robot.

4.4.1 Incremental Learning Through HRI

In this experiment we aim to emulate a real scenario with an assistant robot in an electronics workshop. The robot starts by having a pre-trained baseline object detection and recognition ability and we try to teach it to recognize new objects in the workshop. Accordingly, using our system 4.2, we introduced new objects, one at a time, to the robot. The robot collected images from each new object using its eye-in-hand camera. Samples of these images can be seen in Fig. 4.6. After each collection, the new object is appended to the robots detection module in real-time and then we move to append another object in the same fashion. After adding each new class, we re-evaluate the recognition accuracy on the MS-COCO test set plus the test portion of the newly added class. After evaluation of each new class, we include their test portion into our dynamic test dataset. Since the MS-COCO test set is significantly larger than test portion of new classes, to capture the variance in accuracy, we vary the ratio between number of samples from the new object and number of samples from old objects. We change this ratio from 0.05 to 0.5 with

the step size of 0.02. It gives a better estimation of the network's performance in a real scenario with an unknown distribution of encountering different classes. Both incremental learning approaches were tested in a same manner

Figure 4.7 and 4.8 summarizes the results of this experiment for the first and the

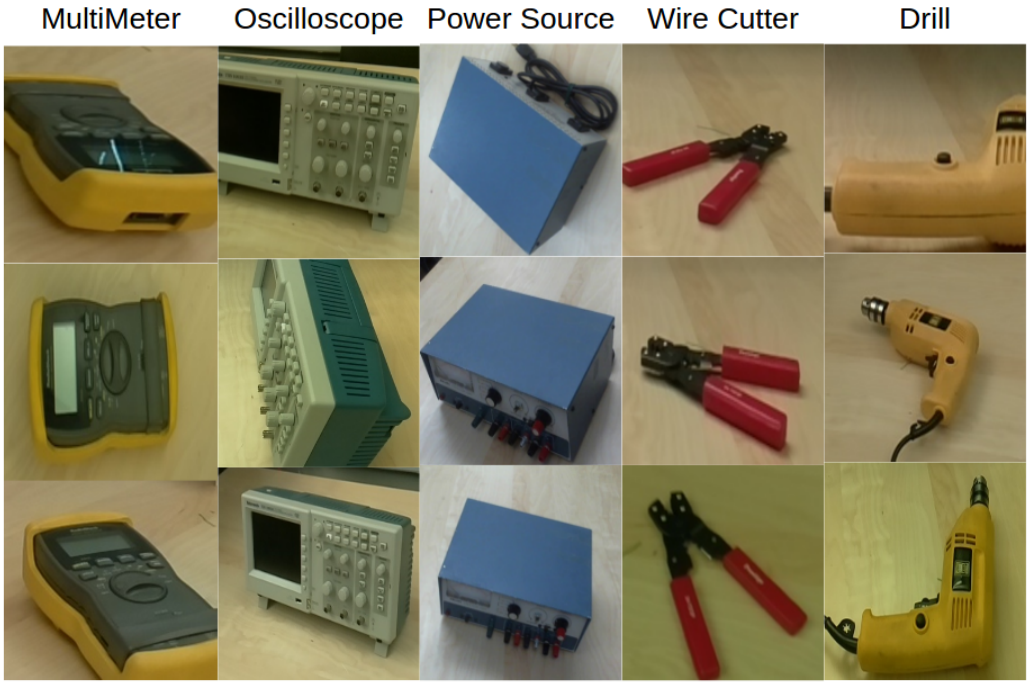

Figure 4.6: Sample images of the data collected by the robot and TLD tracker.

second approach respectively. All the shown values are top 1 accuracy. As we expected, the accuracy is decreasing slightly by adding new objects however, this drop is not significant and the slope is low especially in the second approach. In which, we only observed a total of 5% drop after adding 11 new objects/ Accordingly, we can say that the accuracy stays in a usable range even after adding many new objects. Also note that the baseline model already includes 80 common objects and therefore not too many additions is expected.

4.4.2 Object Detection and Recognition Baseline

We can get near real-time performance from our detection and recognition system with the simplifications that was made in the network4.3.1. Our model's inference

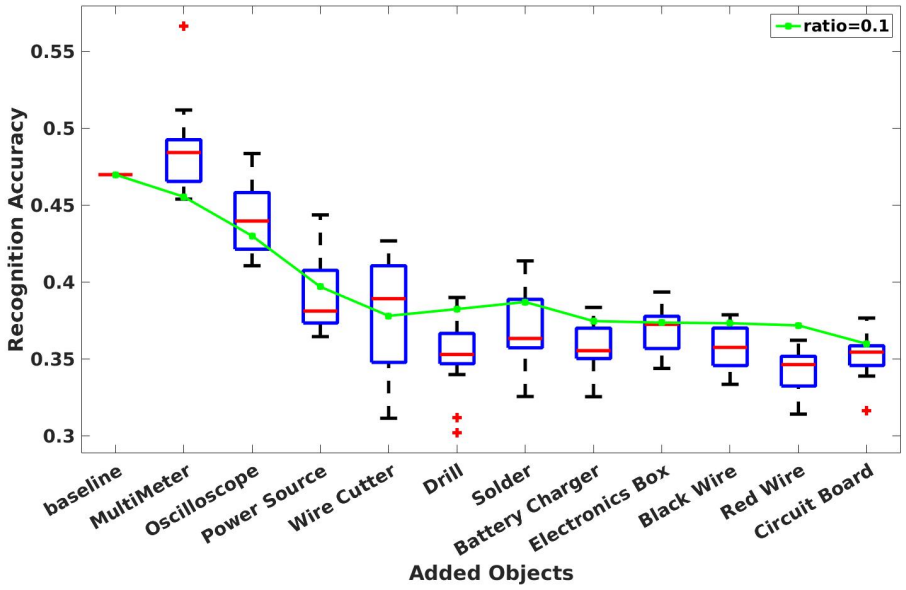

Figure 4.7: Recognition performance in an incremental learning scenario using the first approach. New objects are introduced by user one by one and their images are collected by the robot. Shown values are top 1 accuracies. Boxplots capture the variance in accuracy as the ratio between test samples from new class and test sample from old classes changes from 0.05 to 0.5. Green line is the accuracy for when this ratio is 0.1.

time is 150ms for doing both the detection and the recognition on a GeForce 960 GPU. It is more than twice as fast as R-CNN with 350ms on Titan X GPU [33]. In the object detection task, we reached Average Precision (AP) of 0.2026 which is comparable with the state of the art at 0.224 [60]. Based on AP metric, a detection is counted correct if it has IoU of more than 0.5 with the ground truth.

We measured baseline model top-1 accuracy for recognition. The prediction is correct if the class with the highest probability is similar to the ground truth. We achieved recognition accuracy of 0.45. We are not aware of any recognition performance reported on MS-COCO. But considering the difficulty of MS-COCO compared to imagenet, an accuracy close to this value is expected.

The total accuracy of the model is 0.1704. To compute this accuracy we use the precision metric as follow. A prediction is counted correct if it satisfies both localization with IoU greater than 0.5 and also recognizes the object.

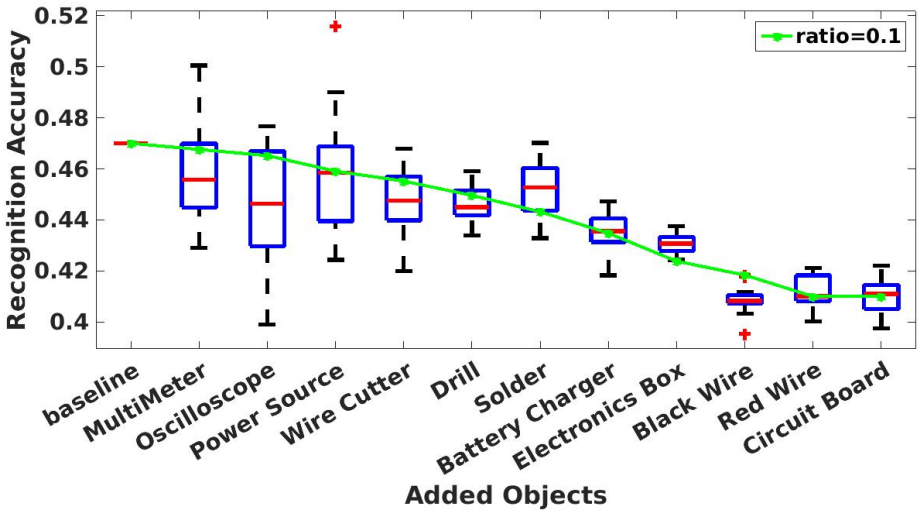

Figure 4.8: Recognition performance in an incremental learning scenario using the second approach. New objects are introduced by user one by one and their images are collected by the robot. Shown values are top 1 accuracies. Boxplots capture the variance in accuracy as the ratio between test samples from new class and test sample from old classes changes from 0.05 to 0.5. Green line is the accuracy for when this ratio is 0.1.

4.4.3 Our Incremental Learning Approach

To make our approach verifiable by researchers, we tested our incremental learning system using publicly available datasets. In this experiment, we started with our baseline model trained on MS-COCO and then we incrementally added new classes to the model. These new classes are randomly chosen from imagenet [38]. We followed the same procedure for evaluation as the first experiment 4.4.1 with one exception that now, the labeled data for new object comes from imagenet rather than HRI. The results are summarized as the same fashion in Figures 4.9 and 4.10 for the first and the second approach respectively.

By comparing real collected images and imagenet images we observe a slight performance gain with the imagenet data compared to the robot's collected data. It can be attributed to the diversity of the images in imagenet and therefore, decreasing the chance of over-fitting. However, this small difference shows that the images that the robot has collected work nearly as good as a generic dataset, at least for a local use which is our intended purpose.

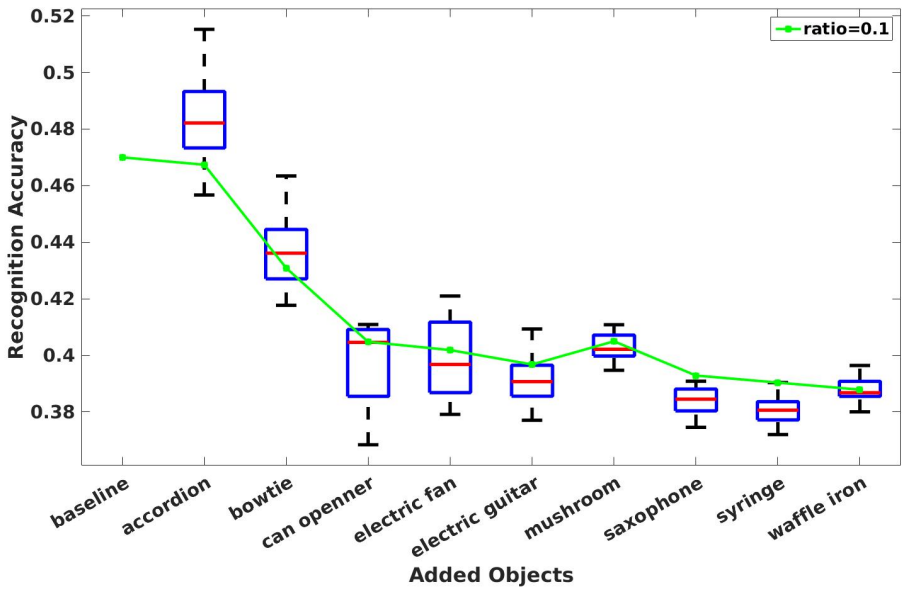

Figure 4.9: Recognition performance after incrementally adding new classes using the first approach. The data for new classes are taken from imagenet. Shown values are top 1 accuracies. Boxplots capture the variance in accuracy as the ratio between test samples from new class and test sample from old classes changes from 0.05 to 0.5. Green line is the accuracy for when this ratio is 0.1

4.4.4 Mock Human-Human Interaction Performance Evaluation

We designed and conducted an experiment to find a gold standard interaction interface for a collaborative sorting task. The goal of the task is to sort 10 common household objects into 2 bins. Two humans engage in the experiment. One is the instructor that tells the actor what object to pick and where to drop them. Each experiment session starts with the objects randomly placed on a table and finished when all objects are sorted into two bins. Four interaction medians were tested and are listed below.

- **Pointing**: The instructor indicates the objects and target bins by pointing at respectively. The actor follows the hand pointings and possibly the direction of the gaze to locate the object and the bin.

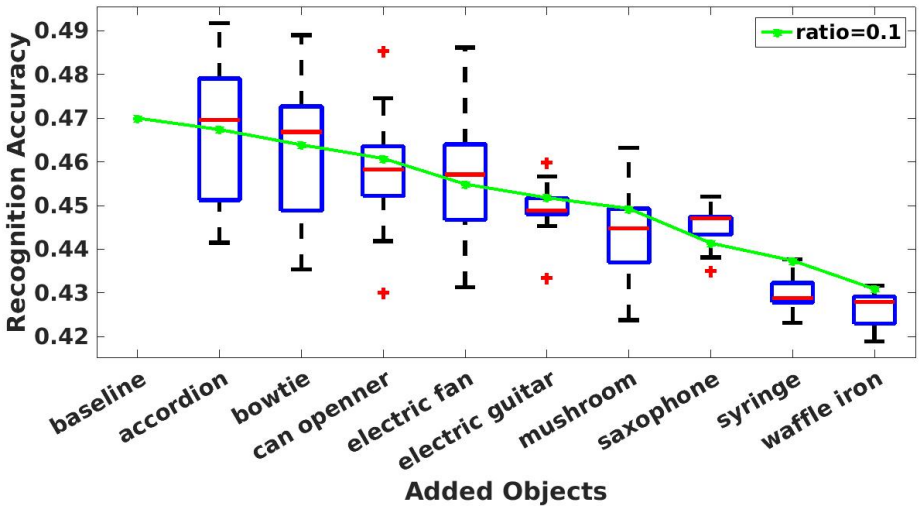

Figure 4.10: Recognition performance after incrementally adding new classes using the second approach. The data for new classes are taken from imagenet. Shown values are top 1 accuracies. Boxplots capture the variance in accuracy as the ratio between test samples from new class and test sample from old classes changes from 0.05 to 0.5. Green line is the accuracy for when this ratio is 0.1

- **Speech**: The instructor describes the object to be picked up by speech and indicates the bins by saying left are right which are the relative locations of the bins with respect to the instructor. The actor only performs the task only by listening to the instructor without any eye contact or other indicative body gestures.

- **Clicking**: This interface uses a monitor which is visible by both the instructor and the actor. The instructor selects objects by clicking on them and then select the appropriate bin. The actor sees the monitor and performs the actions according to the clicks.

- **Pointing and Speech** In this interface, the instructor can use both speech and pointing to communicate its intentions. The actor both listens to the instructors and also follows their pointing and eye gaze direction to find the object and the bins.

We tested these medians in the same fashion with the same test group. The actor kept the same for all experiments. To evaluate the interfaces, we used standard

NASA Task Load Index (TLX) surveys [**?**]. We also timed the task completion. A summary of results can be seen in Figure 4.11. While there is no clear winner, the combination of pointing and speech has a minor edge over the rest. In particular, in terms of the task completion time and effort. Pointing would be second due to its clear edge in the mental demand axis. The next one would be clicking. It was the most accurate yet it required the most effort and took long to finish. The least favored interface was the speech. Even though it had an obvious advantage in terms of physical demand (which is important for physically impaired), it performs badly in terms of time and mental demand. Clearly, describing an object tended to more challenging than pointing in many cases. A take home lesson from this experiment is to incorporating bodily gestures into the interface is much more natural and convenient for humans. Also, including the speech helps especially if the task was more complicated and required higher level communications between the instructor and the actor.

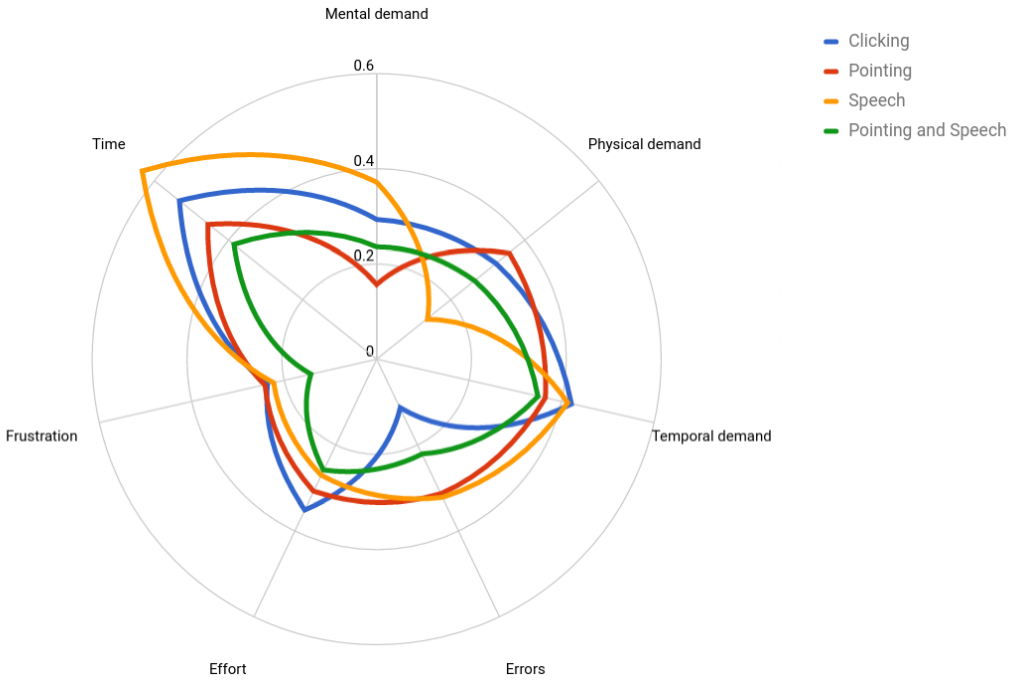

Figure 4.11: NASA Task Load Index evaluation for 4 tested interfaces. Lower values are better in all dimensions.

4.5 Discussion

We proposed and developed a complete HRI system that aims to use human knowledge to improve robot's perception. Our system is able to learn appearance of a new object by human guidance and add it to its knowledge base. The natural interface allows humans to easily see the holes in the robots perception and correct it just by gesturing and speech.

We proposed two suitable incremental learning framework to be used in our setup. The methods were tested and compared. The second incremental learning approach proved to be more efficient by showing less decrease in performance as new objects being added.

While we only showed the incremental learning for the visual tasks, the idea is not limited to it and can be extended to teaching actions. In fact, transferring trajectory generation skills from humans to robot is very well studied. We worked on learning from demonstration systems and have used it on our robots. We are yet to incorporate it into the incremental learning system.

Chapter 5

Conclusion and Future Work

In this thesis we reviewed some issues with using deep learning methods in robotics. Namely the lack of utilizing temporal information and relying solely on the train data that is initially available. Then we discussed how robots can circumvent these problems to some extent by using their sensors and the capability of interacting with humans and the environment.

We introduced recurrent fully convolutional networks for video segmentation that fits nicely with many robotics tasks such as self-driving cars. We showed how such a network can be designed and trained and then the experiments were made on popular video segmentation benchmarks. We observed consistent improvement by using our method compared to regular fully convolutional networks.

We also looked into object detection and recognition problem in a robotics scenario and proposed a novel method for using human robot interaction to improve robots perception. We designed a full interaction interface that allows the human and robot to communicate through, speech, gesture, and vision. We showed how such interaction can provide additional data for the robot to learn from. Two different incremental learning methods were incorporated into deep object detection-recognition module and tested. We showed that how our system is capable of improving the robot's perception on both a simulated scenario using imagenet data and also real tests with everyday objects.

An important aspect of robots is their ability to manipulate their environment. But learning such skills has proven difficult. While there exist engineers approaches for some manipulation tasks a general solution is still out of the reach. But, similar

to perceiving objects, the robot does not need to know how to perform all the manipulations tasks to be useful as long as it can learn these skills easily when needed. With this approach, one can imagine an incremental learning process for actions and trajectory generation. We are experimenting with a Learning from Demonstration system that uses a deep network (an RNN to be exact) for trajectory learning. This allows easier incorporation of different signals and possibly a more general system. We have developed a simulation environment to test such a system. A 7 DoF robot is being controlled for a reaching task with the simulator path planner. The trajectories are recorded and used to train a multi-layered GRU which given the current states, produces the next control signal. Early results are promising and indicate that the network is able to learn the task. However, we see a lack of generalization that one expects from a deep network. A possible solution is to use the visual input directly in the system. An RFCNN is suitable as it can incorporate image sequences as well as other sensory signals. The segmentation task can be used as an auxiliary objective to guide the training while the main objective is to produce correct control signal. For the future work, we will investigate this path further and explore possible techniques that can be used to guide and improve the network. Some possibilities are, memory networks to store different skills separately and load them in appropriate time. A combination of visual servoing and LfD can also be beneficial.

Bibliography

[1] CMU Sphinx Open Source Toolkit for Speech Recognition project by carnegie mellon university. http://cmusphinx.sourceforge.net/. Accessed: 2016-06-25.

[2] Wam robotic arm. http://www.barrett.com/products-arm-articles.htm. Accessed: 2016-09-11.

[3] https://webdocs.cs.ualberta.ca/~vis/HRI/HRI_learn.mp4, Sept 2016.

[4] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432*, 2015.

[5] Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, James Bergstra, Ian J. Goodfellow, Arnaud Bergeron, Nicolas Bouchard, and Yoshua Bengio. Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012.

[6] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1893–1902, 2015.

[7] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on*, 5(2):157–166, 1994.

[8] Alan W. Black and Paul A. Taylor. The Festival Speech Synthesis System: System documentation. Technical Report HCRC/TR-83, Human Communciation Research Centre, University of Edinburgh, Scotland, UK, 1997. Available at http://www.cstr.ed.ac.uk/projects/festival.html.

[9] Cynthia Breazeal. Toward sociable robots. *Robotics and autonomous systems*, 42(3):167–175, 2003.

[10] Cynthia Breazeal, Guy Hoffman, and Andrea Lockerd. Teaching and working with robots as a collaboration. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1030–1037. IEEE Computer Society, 2004.

[11] Lorenzo Bruzzone and D Fernandez Prieto. An incremental-learning neural network for the classification of remote-sensing images. *Pattern Recognition Letters*, 20(11):1241–1248, 1999.

[12] Maya Cakmak, Crystal Chao, and Andrea L Thomaz. Designing interactions for robot active learners. *IEEE Transactions on Autonomous Mental Development*, 2(2):108–118, 2010.

[13] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[14] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[15] Aurélie Clodic, Sara Fleury, Rachid Alami, Matthieu Herrb, and Raja Chatila. Supervision and interaction. In *Advanced Robotics, 2005. ICAR'05. Proceedings., 12th International Conference on*, pages 725–732. IEEE, 2005.

[16] Adam Coates, Pieter Abbeel, and Andrew Y Ng. Learning for control from multiple demonstrations. In *Proceedings of the 25th international conference on Machine learning*, pages 144–151. ACM, 2008.

[17] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3213–3223, 2016.

[18] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7(Mar):551–585, 2006.

[19] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[20] Peter Ford Dominey, Manuel Alvarez, Bin Gao, Marc Jeambrun, Anne Cheylus, Alfredo Weitzenfeld, Adrian Martinez, and Antonio Medrano. Robot command, interrogation and teaching via social interaction. In *5th IEEE-RAS International Conference on Humanoid Robots, 2005.*, pages 475–480. IEEE, 2005.

[21] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2625–2634, 2015.

[22] Michael A Goodrich and Alan C Schultz. Human-robot interaction: a survey. *Foundations and trends in human-computer interaction*, 1(3):203–275, 2007.

[23] Nil Goyette, Pierre-Marc Jodoin, Fatih Porikli, Janusz Konrad, and Prakash Ishwar. Changedetection. net: A new change detection benchmark dataset. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*, pages 1–8. IEEE, 2012.

[24] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[25] Mona Gridseth, Oscar Ramirez, Camilo Perez Quintero, and Martin Jagersand. Vita: Visual task specification interface for manipulation with uncalibrated visual servoing. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3434–3440. IEEE, 2016.

[26] Daniel H Grollman and Odest Chadwicke Jenkins. Incremental learning of subtasks from unsegmented demonstration. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 261–266. IEEE, 2010.

[27] Verena V Hafner and Frédéric Kaplan. Learning to interpret pointing gestures: experiments with four-legged autonomous robots. In *Biomimetic neural learning for intelligent robots*, pages 225–234. Springer, 2005.

[28] Trevor Hastie. *Neural network*. Wiley Online Library, 1998.

[29] Masato Hirose and Kenichi Ogawa. Honda humanoid robots development. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 365(1850):11–19, 2007.

[30] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[31] Lichao Huang, Yi Yang, Yafeng Deng, and Yinan Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015.

[32] Björn Jensen, Nicola Tomatis, Laetitia Mayor, Andrzej Drygajlo, and Roland Siegwart. Robots meet humans-interaction in public spaces. *IEEE Transactions on Industrial Electronics*, 52(6):1530–1546, 2005.

[33] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. *arXiv preprint arXiv:1511.07571*, 2015.

[34] Zdenek Kalal, Jiri Matas, and Krystian Mikolajczyk. Pn learning: Bootstrapping binary classifiers by structural constraints. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 49–56. IEEE, 2010.

[35] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[36] Seungsu Kim, ChangHwan Kim, Bumjae You, and Sangrok Oh. Stable whole-body motion generation for humanoid robots to imitate human motions. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2518–2524. IEEE, 2009.

[37] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[39] Dana Kulić, Wataru Takano, and Yoshihiko Nakamura. Incremental learning, clustering and hierarchy formation of whole body motion patterns using adaptive hidden markov chains. *The International Journal of Robotics Research*, 27(7):761–784, 2008.

[40] Adam Eric Leeper, Kaijen Hsiao, Matei Ciocarlie, Leila Takayama, and David Gossow. Strategies for human-in-the-loop robotic grasping. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 1–8. ACM, 2012.

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

[42] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[43] Fuxin Li, TaeyouGng Kim, Ahmad Humayun, David Tsai, and James M Rehg. Video segmentation by tracking many figure-ground segments. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2192–2199, 2013.

[44] Guosheng Lin, Chunhua Shen, Ian Reid, et al. Efficient piecewise training of deep structured models for semantic segmentation. *arXiv preprint arXiv:1504.01013*, 2015.

[45] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[46] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[47] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[48] Vincent Michalski, Roland Memisevic, and Kishore Konda. Modeling deep temporal dependencies with recurrent grammar cells"". In *Advances in neural information processing systems*, pages 1925–1933, 2014.

[49] Shin'ichiro Nakaoka, Atsushi Nakazawa, Fumio Kanehiro, Kenji Kaneko, Mitsuharu Morisawa, Hirohisa Hirukawa, and Katsushi Ikeuchi. Learning from observation paradigm: Leg task models for enabling a biped humanoid robot to imitate human dances. *The International Journal of Robotics Research*, 26(8):829–844, 2007.

[50] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[51] Giulia Pasquale, Carlo Ciliberto, Francesca Odone, Lorenzo Rosasco, Lorenzo Natale, and Ingegneria dei Sistemi. Teaching icub to recognize objects using deep convolutional neural networks. *Proc. Work. Mach. Learning Interactive Syst*, pages 21–25, 2015.

[52] Mircea Serban Pavel, Hannes Schulz, and Sven Behnke. Recurrent convolutional neural networks for object-class segmentation of rgb-d video. In *Neural Networks (IJCNN), 2015 International Joint Conference on*, pages 1–8. IEEE, 2015.

[53] F Perazzi, J Pont-Tuset1 B McWilliams, L Van Gool, M Gross, and A Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation.

[54] T Poggio and G Cauwenberghs. Incremental and decremental support vector machine learning. *Advances in neural information processing systems*, 13:409, 2001.

[55] Gill Pratt and Justin Manzo. The darpa robotics challenge [competitions]. *Robotics & Automation Magazine, IEEE*, 20(2):10–12, 2013.

[56] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5, 2009.

[57] Camilo Perez Quintero, Romeo Tatsambon Fomena, Azad Shademan, Nina Wolleb, Travis Dick, and Martin Jagersand. Sepo: Selecting by pointing as an intuitive human-robot command interface. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1166–1171. IEEE, 2013.

[58] Camilo Perez Quintero, Oscar Ramirez, and Martin Jägersand. Vibi: Assistive vision-based interface for robot manipulation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4458–4463. IEEE, 2015.

[59] Camilo Perez Quintero, Romeo Tatsambon, Mona Gridseth, and Martin Jägersand. Visual pointing gestures for bi-directional human robot interaction in a pick-and-place task. In *Robot and Human Interactive Communication (RO-MAN), 2015 24th IEEE International Symposium on*, pages 349–354. IEEE, 2015.

[60] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[61] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3234–3243, 2016.

[62] Eric L Sauser, Brenna D Argall, Giorgio Metta, and Aude G Billard. Iterative learning of grasp adaptation through human corrections. *Robotics and Autonomous Systems*, 60(1):55–71, 2012.

[63] Walter J Scheirer, Anderson de Rezende Rocha, Archana Sapkota, and Terrance E Boult. Toward open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(7):1757–1772, 2013.

[64] Walter J Scheirer, Lalit P Jain, and Terrance E Boult. Probability models for open set recognition. *IEEE transactions on pattern analysis and machine intelligence*, 36(11):2317–2324, 2014.

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

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

[67] Mennatullah Siam, Sepehr Valipour, Martin Jagersand, and Nilanjan Ray. Convolutional gated recurrent networks for video segmentation. *arXiv preprint arXiv:1611.05435*, 2016.

[68] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[69] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024, 2011.

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

[71] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. In *Advances in neural information processing systems*, pages 2553–2561, 2013.

[72] Graham W Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler. Convolutional learning of spatio-temporal features. In *Computer Vision–ECCV 2010*, pages 140–153. Springer, 2010.

[73] Aleš Ude, Christopher G Atkeson, and Marcia Riley. Programming full-body movements for humanoid robots by observation. *Robotics and autonomous systems*, 47(2):93–108, 2004.

[74] Sepehr Valipour, Camilo Perez, and Martin Jagersand. Incremental learning for robot perception through hri. *arXiv preprint arXiv:1701.04693*, 2017.

[75] Sepehr Valipour, Mennatullah Siam, Martin Jagersand, and Nilanjan Ray. Recurrent fully convolutional networks for video segmentation. In *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*, pages 29–36. IEEE, 2017.

[76] Oriol Vinyals, Suman V Ravuri, and Daniel Povey. Revisiting recurrent neural networks for robust asr. In *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*, pages 4085–4088. IEEE, 2012.

[77] Francesco Visin, Kyle Kastner, Aaron Courville, Yoshua Bengio, Matteo Matteucci, and Kyunghyun Cho. Reseg: A recurrent neural network for object segmentation. *arXiv preprint arXiv:1511.07053*, 2015.

[78] Lijun Wang, Wanli Ouyang, Xiaogang Wang, and Huchuan Lu. Visual tracking with fully convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3119–3127, 2015.

[79] Holly A Yanco, Adam Norton, Willard Ober, David Shane, Anna Skinner, and Jack Vice. Analysis of human-robot interaction at the darpa robotics challenge trials. *Journal of Field Robotics*, 32(3):420–444, 2015.

[80] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[81] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip HS Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

[82] R Zoliner, Michael Pardowitz, Steffen Knoop, and Rüdiger Dillmann. Towards cognitive robots: Building hierarchical task representations of manipulations from human demonstration. In *Proceedings of the 2005 IEEE International Conference On Robotics and Automation*, pages 1535–1540. IEEE, 2005.