2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

SA-Net: Robust State-Action Recognition for Learning from Observations

Nihal Soans, Ehsan Asali, Yi Hong, and Prashant Doshi[1]

***Abstract*— Learning from observation (LfO) offers a new paradigm for transferring task behavior to robots. LfO requires the robot to observe the task being performed and decompose the sensed streaming data into sequences of state-action pairs, which are then input to LfO methods. Thus, recognizing the state-action pairs correctly and quickly in sensed data is a crucial prerequisite. We present SA-Net a deep neural network architecture that recognizes state-action pairs from RGB-D data streams. SA-Net performs well in two replicated robotic applications of LfO – one involving mobile ground robots and another involving a robotic manipulator – which demonstrates that the architecture could generalize well to differing contexts. Comprehensive evaluations including deployment on a physical robot show that SA-Net significantly improves on the accuracy of the previous methods under various conditions.**

I. Introduction

Recent robot learning methods for learning from demonstration [2], [3] allow a transfer of preferences and policy from the expert performing the task to the learner. These methods have allowed the learning robots to successfully perform difficult acrobatic aerial maneuvers [1], carry out nontrivial manipulation tasks [15], penetrate patrols [4], and merge autonomously into a congested freeway [13]. One way by which this transfer occurs is the learner simply observing the expert perform the task. Observing the expert engaged in the task is expected to yield trajectories of state-action pairs, which is then input to the algorithms that drive some of these methods. Consequently, recognizing the expert's state and action accurately from observations is crucial for the learner. If the learner is a robot, its observations are sensor streams. Very likely, these will be streams from camera and range sensors yielding RGB and depth (RGB-D) data. Thus, the learning robot must recognize sequences of state-action pairs quickly and accurately from RGB-D streams. This is a key component of the learning from observation (LfO) pipeline.

In this paper, we present SA-Net, a deep neural network that recognizes state-action pairs from RGB-D data streams with a high accuracy. This supervised learning method offers a general deep learning alternative to the current adhoc techniques, which often rely on problem-specific implementations using OpenCV. Figure 1 gives an overview of how SA-Net is deployed. SA-Net aims to recognize from a sensor stream, the expert's state and action. The state is often the 2D or 3D coordinates in a global reference frame and the orientation. The action is derived from the motion performed by the robot.

[1]Nihal Soans, Ehsan Asali, Yi Hong and Prashant Doshi is with Dept. of Computer Science, University of Georgia, Athens GA 30606, USA pdoshi@cs.uga.edu

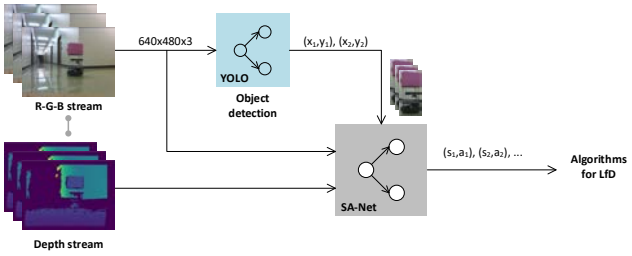


Fig. 1: Overview of the I/O of SA-Net for state-action recognition from RGB-D streams of a TurtleBot patrolling a corridor.

As the learner's position may not be fixed, SA-Net seeks to recognize the coordinates and orientation of the observed object(s) relative to the learner's location and project it on the global reference frame. While the RGB frame offers context, the depth data is relative to the observer. Coordinates are recognized by interleaving convolutional neural nets (CNN) and pooling layers followed by fully-connected layers input to a softmax. This allows the use of all four channels, RGB-D, in recognizing the coordinates. Identifying the expert's orientation and action is more challenging. Both of these rely on temporal data, and SA-Net utilizes frames from time steps $t-2$, $t-1$, and current time step t. Each frame is cropped previously by a network such as Faster R-CNN [8] or YOLO2 [16] to focus attention on the expert. The network backtracks the movement inside the bounding box for time step $t-1$ and $t-2$ using a layer of time-distributed CNNs followed by two convolutional long-short-term memory nets (LSTM) [10]. SA-Net continues to utilize the depth channel here by running an intercept to the previously described fully-connected nets that provides the relative distance.

We evaluate SA-Net on two diverse tasks. (i) It is used to identify the state-action sequences of two TurtleBots that are simultaneously but independently patrolling a hallway. SA-Net is deployed on a third TurtleBot that is observing the patrollers from a vantage point, and is tasked with penetrating the patrol [4]. (ii) SA-Net is used to identify the state-action sequences of a PhantomX robotic arm that is engaged in pick-and-place to sort objects. In both tasks, SA-Net exhibits high accuracy while being able to run on computing machines with limited processing power and memory on board a robot. Ablation and robustness studies demonstrate that the architecture is effective and that SA-Net can handle some typical adverse conditions as well. Consequently, SA-Net offers high-accuracy trajectory recognition to facilitate robots engaged in LfO for various tasks.

II. Related Work

Traditionally, the state and action of an observed robot is recognized by tracking a marker associated with the robot. For example, Bogert and Doshi [4] utilizes a colored box on top of the TurtleBot for blob detection in CMVision coupled with 3D point-cloud processing. This centroid-based method simplifies the estimation of the robot's trajectory but is not robust to occlusion of the marker and to noise in the context.

Recently, deep NNs have demonstrated significantly improved performance on tasks involving image and video analysis [9], [26]. Related to our method are the NN architectures in computer vision for recognizing human gestures and activities though these rely predominantly on videos. For example, Ji et al. [11] recognizes human actions in surveillance videos using a 3D-CNN. Furthermore, a recurrent NN (RNN) combined with 3D-CNN [12] classifies and temporally localizes activities in untrimmed videos. Rezazadegan et al. [18] introduced two-stream VGG16 based CNNs that utilize spatial and optical flow images to recognize robotic activities. Depth modality can be leveraged for gesture recognition using two separate CNN streams with a late fusion network [7]. Alternately, RGB and depth modalities can be superimposed to extract features for action recognition with CNNs [23]. These action recognition methods learn from videos by treating them as either 3D volumes with multiple adjacent frames [11], one or multiple compact images [14], [18], [23], or as an image frame sequence [12]. Our method belongs to the last category and handles the image and depth sequence with LSTMs, which learn temporal dependencies.

Among methods that use LSTMs, Wang et al. [24] adopts 3D-CNNs to extract features from video clips and uses the LSTM to extract dynamic features for recognizing actions in the video. Another related work [21] leverages a bidirectional LSTM to capture temporal changes. But, this may not distinguish between moving forward and backward or left vs. right motions. In comparison to these sophisticated NN designs, SA-Net's architecture for action recognition is simpler because it avails of an additional modality – depth streams. Furthermore, in contrast to these methods focusing on recognizing actions, SA-Net is tasked with recognizing the state and action pairs simultaneously in real time from just a few frames, as demonstrated by the experiments.

A recent deep NN architecture SE3-Nets [6] predicts the rigid body motion of objects in the robotic scene. However, SA-Net has a different focus: to recognize or summarize a robot's trajectory from RGB-D streams rather than estimate its visual representation using 3D point cloud or images.

III. SA-Net Architecture

As SA-Net is tasked with recognizing state-action pairs, this motivates a network design that efficiently mixes convolutional and recurrent NNs, which we describe below.

A. Problem Definition

We aim to automatically estimate the state and action pairs of an expert from RGB-D streams using deep NNs. Given the expert's three video frames captured by a learner at time points $t-2$, $t-1$, and t, our network jointly predicts the state (X, Y, Z, θ) and action (A) of the expert at the current time point t. Here, the tuple (X, Y, Z) in the state representation describes the location coordinate of the expert in a 3D environment; the Z dimension is ignored for 2D cases. The θ describes the orientation of the expert. In this paper, we consider discrete state and action spaces, which allows formulating the task as a multi-label classification problem. Formally, the problem can be formulated as:

$$(X, Y, Z, \theta, A) = f(I_{t-2}, I_{t-1}, I_t; \mathbf{\Theta}), \quad \text{where}$$
$$X \in \{0, ..., N_X - 1\}, Y \in \{0, ..., N_Y - 1\}, Z \in \{0, ..., N_Z - 1\},$$
$$\theta \in \{0, ..., N_\theta - 1\}, A \in \{0, ..., N_A - 1\}.$$

Here, f denotes the mapping function learned by our classification network; I_{t-2}, I_{t-1}, and I_t are the three frame inputs; $\mathbf{\Theta}$ represents the parameter set of the network for classifying the state and action jointly; N_X, N_Y, and N_Z are the discretized dimensions in each coordinate; N_θ is the number of the expert's orientations – for instance, we have four orientations including north, south, east, and west in the TurtleBot application; N_A is the number of actions, e.g., four actions including move forward, stop, turn right, and left. Overall, the network includes two coupled components for the state and action recognition, which are learned simultaneously. SA-Net's architecture is shown in Fig. 2.

B. State Recognition

State recognition aims to determine the expert's coordinate (X, Y, Z) and its orientation θ. Typically, the expert's coordinate can be identified on the basis of its surrounding environment. Therefore, we use one image frame without considering the temporal information in our coordinate recognition module. Different from state recognition, orientation recognition requires more than one image frame to recognize hard-to-distinguish orientations. As shown in Fig. 3, the TurtleBot is oriented differently in the two images, but the image difference is too subtle to correctly separate these two orientations of the TurtleBot. In such situations, image sequence plays an important role in recognizing the orientation. Therefore, in the state recognition of SA-Net, we separate the prediction of the coordinate (X, Y, Z) from that of the orientation θ, as one network stream takes the static image as input while the other takes the image sequence.

Coordinate recognition As shown in the top stream of the network in Fig. 2, only the image frame at time point t is used to predict the expert's location coordinate. We assign a pre-defined coordinate system for each environment; that is, each image frame will be classified into a unique coordinate, which is represented by an absolute location (X, Y, Z) with respect to the origin in the coordinate system. The expert's coordinates are learned from images captured by the learner; however, the learner's location may change in different situations. To improve the network's generalization, we leverage the relative distance between the expert and the learner to help in the recognition of the expert's coordinate.

In the coordinate recognition branch, we have two sets of coordinate-related predictions: the relative distance $(\Delta X, \Delta Y, \Delta Z)$ and the absolute coordinate (X, Y, Z). These

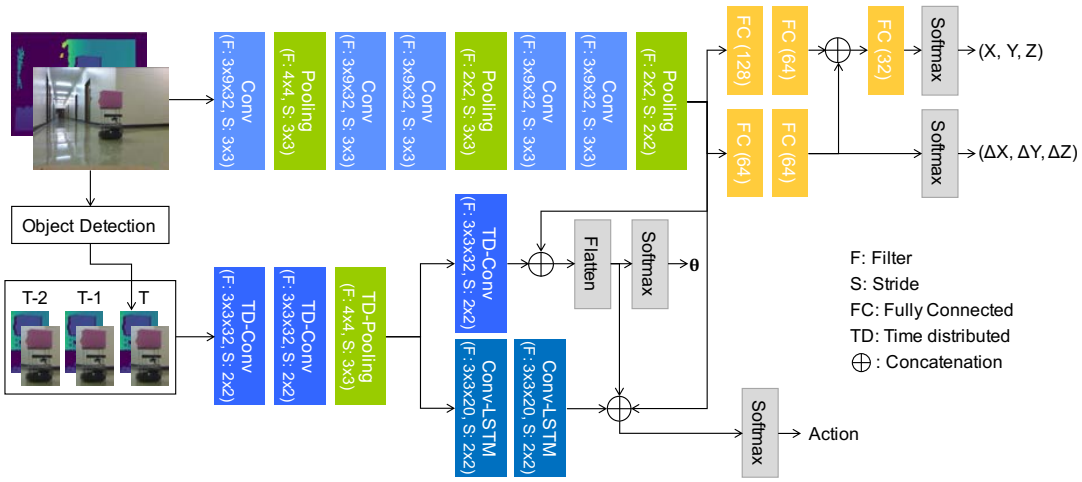


Fig. 2: An overview of the SA-Net architecture. This network jointly predicts the state and action of an expert using the observed RGB-D data streams and corresponding sequential data cropped by an object detection model. The final outputs of the network include the coordinate X, Y, Z, the orientation θ, and the action. The additional output of relative coordinate $\Delta X, \Delta Y, \Delta Z$ is used in the training.

two prediction tasks share the same process of image feature extraction, which includes five convolutional layers and three max pooling layers. The convolutional layers use 32 filters with the same kernel size 3×9 and the same 3×3 stride. The three pooling layers are located after the first, third, fifth convolutional layers, respectively, with filters of size 4×4, 2×2, and 2×2 and strides of size 3×3, 3×3, and 2×2. Following the convolutional and pooling layers, two fully convolutional (FC) layers are used in the classification. As the prediction of relative distance contributes to coordinate prediction, we have an additional FC in the stream of coordinate classification after concatenating the pre-activation of the softmax from the relative distance classification.

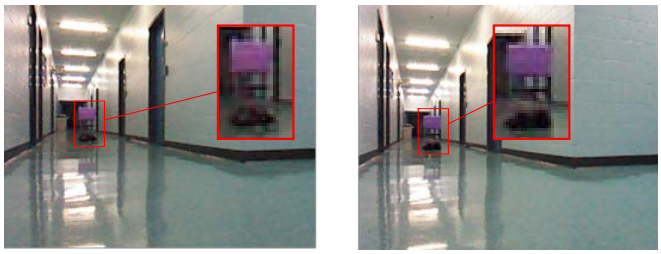


Fig. 3: An example of a TurtleBot in two similar images but having different orientations.

Orientation recognition Different from the coordinate parameter, the orientation of the expert guides its movement particularly so in the context of mobile robots. Therefore, in both orientation and action recognition we would like the network to have its attention on the expert itself, especially when the expert is far away from the learner and relatively small in the whole image frame. To achieve this goal, we adopt object detection to make the expert stand out for perceiving its behavior. More details about the object detection are given in Section III-D. After object detection, we have three new sequential frames, which are cropped from the original RGB-D image inputs and re-sized to images of size 150×100 to facilitate orientation and action recognition of the expert. The sequential frames are essential in orientation recognition to differentiate hard examples as shown in Fig. 3.

To handle the sequential image inputs, we use time-distributed convolutional (TD-Conv) layers in the stream for recognizing orientation. These layers collect image features required for orientation recognition from all three time steps. In particular, we have two TD-Conv layers, followed by one time-distributed max pooling layer and another TD-Conv layer. Each TD-Conv layer has 32 filters of size 3×3 and stride of 2×2, and the pooling layer uses a filter of size 4×4 and stride of 3×3. We observe that the orientation and action recognitions are connected to coordinate recognition, albeit loosely. To motivate this, note that the TurtleBot is less likely to turn left or right if it is in the middle of a corridor. Thus, we concatenate the whole-image features extracted from the coordinate recognition with the spatio-temporal features extracted from the cropped image sequence to predict the expert's orientation. A similar operation is performed in action recognition, as discussed next.

C. Action Recognition

Similar to the orientation recognition, actions are recognized using the same three sequential cropped images after object detection (Section III-D). The goal is to determine the expert's action – for example, in which cardinal direction is the expert moving. Because the orientation and action recognition are working on the same input, they share the first three layers for extracting lower-level features from cropped images at all time steps. The action recognition then uses two convolutional LSTM layers to further compose higher-level features and capture temporal changes in the image sequence. These two new layers also use 32 kernels of size 3×3 and stride of 2×2. In this branch we leverage all features extracted from the state (both coordinate and orientation) recognition to support the action recognition. In tasks involving mobile robots, the orientation and action

are often coupled and we use concatenation to make full use of extracted features. Orientation helps in predicting the action but is, of course, not sufficient. A mobile robot moving toward the observer exhibits the same orientation, which does not reveal the move-forward action. If the state and action are known to be independent, these connections would be removed and SA-Net handles the two independently.

D. Expert Detection

This object detection module provides inputs for the orientation and action recognition of the expert. We use the RGB data stream at the current time point t to perform the object detection using an existing model, YOLO2 [16]. Using the predicted bounding box for the expert $[(x_1, y_1), (x_2, y_2)]$, we crop the images from the frames $t-2$, $t-1$, and t. We keep a small amount of surrounding environment background; this buffered cropping is calculated by the linear equation, $\Delta_a = r \times \Delta_b + c_{min}$. Here, $r \geq 1$ is the cropping factor that determines how aggressively the users want to crop the image; Δ_b is the width $|x_1 - x_2|$ or the height $|y_1 - y_2|$ of the bounding box before the buffered cropping, while Δ_a is the corresponding value after the cropping; and c_{min} is the minimum amount of cropping, e.g., 10 pixels. In all of our experiments, we set $r = 1.1$.

E. Masking for Learning from Multiple Experts

In practice, we may have more than one expert. However, SA-Net is not explicitly designed to recognize the state and action pairs of multiple experts. Despite this, we can allow multiple experts by using masking that ensures only one expert exists in the images for recognition. Specifically, we leverage the object detection (Section III-D) to separate the experts and generate a new image for each. To generate the image for one expert, we remove all others using their detected bounding boxes and replace the removed regions with the background image stored in memory. Thus, we have new images for each expert to pass through the net for state-action recognition. As shown later, this use of masking to segregate experts in each frame does not negatively impact the simultaneous tracking of multiple robots across frames.

IV. Experiments

SA-Net exhibits a general architecture useful in multiple domains. We evaluate it on two tasks offline and online on a physical robot and report on our extensive experiments.

A. Tasks

We evaluated SA-Net on two diverse LfO tasks. First, it was deployed on a TurtleBot tasked with penetrating cyclic patrols by two other TurtleBots in a hallway as shown in Fig. 4(a). Each patroller can assume one of 4 orientations and 4 actions. This task replicates a well-known testbed for evaluating LfO methods such as inverse reinforcement learning [4], [5]. The other task involves observing a PhantomX arm mounted on a TurtleBot (Fig. 4(b)) and performing pick-and-place to sort objects of two types. An overlooking Kinect 360 RGB-D sensor observes the arm [20]. This task adds a third dimension, the height of the end effector, to the state, and the arm has 6 possible orientations and 6 actions that correspond to the motion of the end effector in the 3D space.

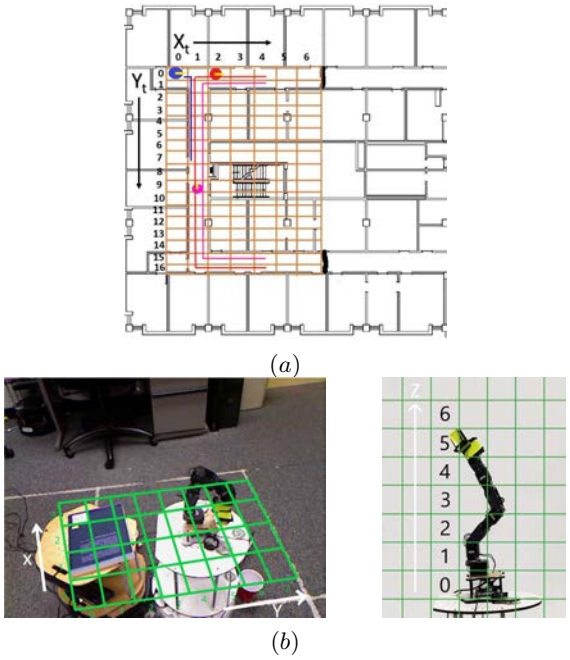


Fig. 4: (a) A map of the hallway patrolled by two TurtleBots. The learner, shown in blue, observes the patrols from its vantage point using a Kinect 360 RGB-D sensor. A 2D grid is superimposed on the hallways. (b) SA-Net is deployed on a computer connected to a Kinect 360 that observes a PhantomX arm mounted on a TurtleBot. A 3D grid is superimposed for the coordinates of its end effector.

B. Formative Evaluation

We evaluated SA-Net using stratified 5-fold cross validation for both tasks. 500 RGB and depth image pairs, each annotated with a bounding box only, were utilized to train a Faster R-CNN [17] whose output then trained a YOLO2 net to learn the bounding boxes for cropping the images. The complete data had 60K annotated sets of RGB and depth image frame pairs for the patrolling task and 10K sets for the manipulation task. Each set consists of an uncropped pair and three cropped pairs at time points $t-2$, $t-1$, and t.

Table I shows the prediction accuracy on the 2D or 3D coordinates and orientation that make up the state, and on the action for each task. We show the mean and standard deviation across the 5 runs. Notice that in both problems, SA-Net generates predictions of state and action with very high accuracy, with those for the manipulation task being slightly less accurate than those for the patrolling. This is generally consistent across all folds yielding low standard deviations. The action recognition compares favorably to that by the two-stream CNN [18] previously discussed in Section II.

C. Ablation Study

We performed an ablation study on a smaller data set to understand the sensitivity of SA-Net's performance on its key components. The ablation study removes a part of the network and conducts experiments on the revised model.

Task	Method	X	Y	Z	θ	Action
Patrolling	SA-Net	98.85±0.02	99.97±0.014	—	99.99±0.01	**99.74±0.01**
	Two-Stream CNN [18]	—	—	—	—	87.53±0.61
Manipulation	SA-Net	97.64±0.02	95.22±0.01	96.16±0.49	98.17±0.04	**99.13±0.02**
	Two-Stream CNN [18]	—	—	—	—	84.05±3.39

TABLE I: Mean and standard deviation of SA-Net's performance from a 5-fold cross validation for the patrolling and manipulation tasks. We show the prediction accuracy of state and action for both tasks. This is compared with another deep NN based method that performs action recognition only from spatial and optical flows [18]. Note that ' — ' denotes not applicable.

Ablation	X	Y	θ	Action
SA-Net w/o Relative X and Y	**81.38±1.50**	91.44±1.46	91.24±1.26	**83.63±1.86**
SA-Net w/o data from $t-1$, $t-2$	96.56±0.01	98.32±0.01	**79.43±1.33**	**78.86±3.57**
SA-Net w/o depth channel	**87.23±1.50**	95.12±1.49	**83.56±1.52**	**81.12±1.00**
SA-Net w/o object detect	**68.74±2.26**	**69.95±1.26**	**21.65±2.39**	**33.89±0.96**

TABLE II: Four ablation experiments in the patrolling task and the impact on the prediction accuracy of state and action.

Relative X and Y In this experiment on the patrolling task, we eliminate that part of SA-Net which contributes to establishing the 2D grid coordinate of the observed robot relative to the observer's location. This part relies more on the depth data. Consequently, we may expect the network to memorize the location by relying more on RGB data but unable to detect changes in its own deployed position. Row 1 of Table II shows a significant drop in the prediction accuracy of state and action with a more pronounced drop in the accuracy of predicting the X-coordinate and action. These two rely significantly more on the relative distances.

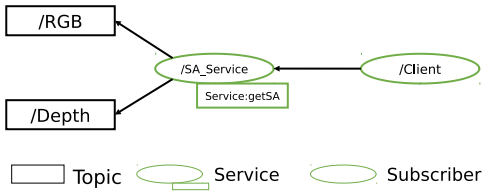
Temporal sequence data In this experiment, we eliminate the part of SA-Net responsible for processing temporal data from previous time steps $t-1$ and $t-2$. This also eliminates those two input channels and keeps input from time step t only. We hypothesize this removal to significantly impact the recognition of orientation θ and action, both of which are thought to rely on sequence data. On the other hand, a single frame could be sufficient to identify the orientation in many cases. Table II, row 2 presents prediction accuracies that are significantly lower for θ and action, while recognizing the 2D coordinates is generally not affected. As such, the temporal data is indeed important for SA-Net in general.

Multimodal data Next, we study if depth data is needed for the predictions and how the network will behave when its removed. Can we make the network learn the state and action from RGB data only? Row 3 of Table II shows that the predictions of X-coordinate, θ, and action are significantly degraded in the absence of the depth channel. The Y-coordinate is least impacted as we may expect. As a patroller approaches the observer, there are multiple states for which the RGB frames are similar. In the absence of depth, the network memorizes certain features and overfits on those.

Object detection Finally, we precluded the object recognition performed by YOLO2, resulting in no cropped images as input. The drastic drop in prediction quality of all coordinates, orientation, and action (row 4) gives evidence that object detection is required. Coordinate recognition suffers because object detection is needed for masking each expert in the context of multiple experts. In recognizing the orientation and action, object detection plays a more integral role focusing SA-Net's attention, which is demonstrated by a larger degradation in their prediction accuracy.

D. Summative Evaluation on Physical Robots

We deployed the trained SA-Net with masking (Section III-E) on a physical TurtleBot that observed two other TurtleBots patrolling the hallway and on a TurtleBot that is connected to a Kinect 360 overlooking a PhantomX arm picking and placing small objects. SA-Net can be used in ROS as a service and the corresponding component architecture is shown in Fig. 5(a).



(a)

Memory usage	742MB± 3MB
Faster R-CNN → SA-Net	6s± 0.4s
YOLO2 → SA-Net	**1.1s± 0.3s**

(b)

Fig. 5: (a) ROS nodes architecture for SA-Net on a robot. (b) SA-Net resource utilizations on a TurtleBot2 standard ASUS notebook with Intel Core i3, 4GB RAM. Note the run time benefit of YOLO2.

Though, it is generally challenging to report the prediction accuracy in online physical experiments, we logged the RGB-D stream and SA-Net's predictions for each frame in the stream. These predictions were later verified manually. Table III reports the prediction accuracy on the observed state-action pairs. We compared SA-Net's performance on the patrolling task with a traditional CMVision based implementation that detects the centroid of the colored box on each robot in the Lab color space and analyzes the depth data. Notice that this method, which was utilized previously

Task	Method	X	Y	Z	θ	Action
Patrolling	SA-Net	**97.23±0.29**	**98.12±0.49**	—	**96.25±0.67**	**96.16±0.68**
	Centroid method [4]	94.15±0.00	96.13±0.00	—	93.16±0.00	78.26±0.68
Manipulation	SA-Net	87.56±0.02	89.25±0.02	91.12±0.03	88.32±0.01	91.18±0.01

TABLE III: SA-Net's accuracy in physical experiments for the two tasks under *typical conditions*. Action prediction is much improved over the baseline method for the TurtleBots.

Test	X	Y	θ	Action
SA-Net w/ Noise	**92.65±0.87**	**96.65±0.72**	**95.23±0.40**	**95.12±0.76**
Centroid method w/ Noise	34.20±6.62	44.43±1.42	23.23±1.31	42.45±1.88
SA-Net w/ Occlusion	**45.15±0.87**	**54.60±0.76**	**64.12±0.99**	**46.36±1.00**
Centroid method w/ Occlusion	18.23±2.13	17.34±1.57	14.42±0.15	43.12±0.80

TABLE IV: Robustness testing of SA-Net and the centroid method [4] on the patrolling task with background noise and occlusion.

for this task [4], is particularly poor in recognizing the patroller's action and SA-Net improves on it drastically. As such, SA-Net should lead to improved LfO. SA-Net's reduced accuracy in the manipulation task is due to the increased complexity from a third coordinate and more manipulator actions. Next, we evaluate SA-Net's predictions under various conditions:

Noise test In this experiment, we test if background noise impacts the prediction accuracy of the network. The noise is defined as objects that look like or have similar characteristics as the target, and dimmed ambient light. Such background objects, shown in Fig. 6(a), include a human wearing a similar-colored shirt and boxes of same color.

Fig. 6: Robustness tests involving background noise via similarly colored boxes on the floor, low ambient light, human sharing the space, and observed robot is partially occluded.

Occlusion test In this experiment, the target is covered partially to approximate 50% occlusion; we cover the TurtleBot by a cardboard box or a white cloth as shown in Fig. 6(b). These robots then patrol the hallways as before.

In Table IV, we show SA-Net's prediction accuracy in each context. For the noise test, the predictions are average of 15 runs split into 5 with a human, 5 with boxes, and 5 with dimmed ambient light. For the occlusion test, again an average of 15 runs is shown with the object partially covered to approximate 50% occlusion. Notice that SA-Net's predictions degrade and rather dramatically under occlusion of the target object. The latter drop is because of SA-Net's reliance on RGB data, which get curtailed under occlusion. Nevertheless, it's predictions remain significantly better in both tests than the traditional centroid-based blob detection method. In particular, the centroid-based method fails to detect the observed robots under occlusion. Similarly colored boxes do not excessively impact SA-Net demonstrating that the object detection is not critically dependent on the marker.

How much memory is consumed by the ROS deployment of SA-Net? Figure 5(b) reports the total amount of RAM held by the ROS service for good performance on state-action recognition. We also show the maximum time in seconds taken by SA-Net for prediction when paired with Faster R-CNN and paired with YOLO2 in the patrolling task having two targets. Notice that pairing with YOLO2 speeds up the prediction by a factor greater than five.

V. Concluding Remarks

SA-Net brings the recent advances in deep supervised learning to bear on a crucial step in LfO. It represents a general architecture for recognizing state-action pairs from RGB-D streams, which are then input to underlying methods for LfO such as inverse reinforcement learning [3]. SA-Net demonstrated recognition accuracies on two diverse LfO tasks that are significantly better than previous conventional techniques and a recent architecture that analyzes videos. This is expected to benefit the subsequent LfO. While minor changes in components may be beneficial, an ablation study revealed that the major architectural parts of the NN are indeed needed. A low resource utilization signature allows SA-Net to be deployed on board robotic platforms. Though SA-Net is shown to predict discretized state and action pairs, it could predict continuous ones by changing the classification network to a regression network.

SA-Net brings another benefit to LfO. Recent techniques, such as maximum entropy deep inverse reinforcement learning [25], utilize a NN. SA-Net can be merged with the NN for inverse reinforcement learning, potentially producing the first end-to-end deep learning approach for LfO. As future work, SA-Net's performance could be improved by improving the LSTM [22] or learning under weak supervision [19].

Acknowledgements

This research was partially supported by NSF grant #IIS 1830421. We thank Kenneth Bogert for help with evaluation of the CMVision based baseline method.

REFERENCES

[1] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng. An application of reinforcement learning to aerobatic helicopter flight. *Advances in Neural Information Processing Systems (NIPS)*, pages 1–8, 2007.

[2] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[3] S. Arora and P. Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *CoRR*, abs/1806.06877, 2019.

[4] K. Bogert and P. Doshi. Multi-robot inverse reinforcement learning under occlusion with interactions. In *International Conference on Autonomous Agents and Multi-Agent Systems*, pages 173–180, 2014.

[5] K. Bogert and P. Doshi. Toward estimating others' transition models under occlusion for multi-robot irl. In *Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1867–1873, 2015.

[6] A. Byravan and D. Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 173–180. IEEE, 2017.

[7] A. Eitel, J. T. Springenberg, L. Spinello, M. Riedmiller, and W. Burgard. Multimodal deep learning for robust RGB-D object recognition. In *Intelligent Robots and Systems (IROS)*, pages 681–687. IEEE, 2015.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[11] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, Jan 2013.

[12] A. Montes, A. Salvador, S. Pascual, and X. Giro-i Nieto. Temporal activity detection in untrimmed videos with recurrent neural networks. *arXiv preprint arXiv:1608.08128*, 2016.

[13] T. Nishi, P. Doshi, and D. Prokhorov. Merging in congested freeway traffic using multipolicy decision making and passive actor-critic learning. *IEEE Transactions on Intelligent Vehicles*, 4(2):287–297, 2019.

[14] E. Park, X. Han, T. L. Berg, and A. C. Berg. Combining multiple sources of knowledge in deep cnns for action recognition. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–8. IEEE, 2016.

[15] N. S. Pollard and J. K. Hodgins. Generalizing demonstrated manipulation tasks. *Algorithmic Foundations of Robotics V*, pages 523–539, 2004.

[16] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[17] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[18] F. Rezazadegan, S. Shirazi, B. Upcrofit, and M. Milford. Action recognition: From static datasets to moving robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3185–3191, 2017.

[19] A. Richard, H. Kuehne, and J. Gall. Weakly supervised action learning with rnn based fine-to-coarse modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 754–763, 2017.

[20] M. Trivedi and P. Doshi. Inverse learning of robot behavior for collaborative planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[21] A. Ullah, J. Ahmad, K. Muhammad, M. Sajjad, and S. W. Baik. Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE Access*, 6:1155–1166, 2017.

[22] V. Veeriah, N. Zhuang, and G.-J. Qi. Differential recurrent neural networks for action recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 4041–4049, 2015.

[23] P. Wang, W. Li, Z. Gao, Y. Zhang, C. Tang, and P. Ogunbona. Scene flow to action map: A new representation for RGB-D based action recognition with convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 416–425, 2017.

[24] X. Wang, L. Gao, J. Song, and H. Shen. Beyond frame-level cnn: saliency-aware 3-d cnn with lstm for video action recognition. *IEEE Signal Processing Letters*, 24(4):510–514, 2016.

[25] M. Wulfmeier, P. Ondruska, and I. Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

[26] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4694–4702, 2015.