Softbot: Software-based Lead-Through for Rigid Servo Robots

Yackov Lubarsky[1], Amit Wolf[2], Lior Wolf[1], Curime Batliner[2], and Jake Newsum[2]

Abstract— **In this paper, we present an interactive control method for rigid robotics. The core of the method is a neural network classifier that maps position and torque readings to a force direction in 3D space. We show that running our method online allows a human to move the robot along a desired path by performing intuitive pushes and pulls on the robot's joints. The setup is sensorless: no additional sensors, other than those already integral to the rigid joint itself, are added to the manipulator or used in the process.**

I. Introduction

Lead-through learning underlies the smooth, sweeping, continuous motions of modern robots in many fields: from arc welding to paint spraying to the cinematic camera movement and control of Hollywood's latest creations. Lead-through learning involves a user that manually conducts the robot through an intentioned path, similar to leading someone by the hand. Conventionally, the method uses a syntaxeur - a mock-up proxy of the manipulator - that carries comparable joint position sensors but none of the actuators. This limitation of the need for a secondary manipulator towards programming has been increasingly overcome by soft robotics. They are also termed compliant robotics. These soft-jointed manipulators employ the mechanical elasticity of spring series elastic actuators (SEAs) towards environmental compliance and adoptability. The seven axes Baxter (by Rethink Robotics), Universal Robots' UR5 and UR10, and Hocoma's Lokomat exoskeleton, are a few examples of current day SEA, with lead-through applications ranging from straightforward programming of simple, everyday tasks, to industrial CoBoting, to medical rehabilitation of limbs [1].

Soft robotics is making its chief strides in solving problems of human-robot interaction as well as addressing the dynamic environments of material production. At the same time, SEAs entail a significant reduction in both repeatability and payload with respect to rigid joint robotics. The powers and limitations of SEAs are balanced by a new breed of joints offering doubly laid and opposing actuators. These variable-stiffness actuators (VSAs) can be seen to regain at least some of the performance benchmarks set by rigid joint industrial robotics [1]. However, these technologies are capital-intensive, requiring the incorporation of new robots.

In this work, we present an applied (add-on) AI software system that is able to give existing rigid medium and heavy payload robots soft capabilities. In the heart of our approach to active compliance, lies an artificial neural network. A specially designed device is used to collect training data easily and reliably.

Overall, our system holds the promise of versatility and universality that is unavailable with current integral or applied solutions. While this approach was developed using Stäubli's medium payload arm (TX90), because our deep neural networks are open-ended and allow for multiple forms of data and training modules, it may potentially be applied to any rigid joint, servo based manipulator.

[1]YL and LW are with the Blavatnik School of Computer Science, Tel Aviv University, Tel Aviv, Israel

[2]AW, CB, and JN are with the Southern California Institute of Architecture (SCI-Arc), Los Angeles, CA, USA

II. Related Work

In our work, we sense external force as it is applied to the manipulator. Some collision-detection systems rely on external sensing mechanisms. For example, in [2] a camera captures the manipulator as it operates and uses image analysis techniques to detect a collision state. Other works aim for a simpler collision detection system, which does not require external aids. These typically use only the available readings such as link position, velocity and torque, and apply rigid-robot dynamic equations [3] to detect differences between sensed and actual torque. A discrepancy indicates the existence of an external force.

An example of such a scheme for collision detection is [4]. Using the system's dynamic equations, the reference torque value is calculated based on the expected position, velocity and other parameters and compared to the actual torque signal which is calculated based on actual sensor readings. The assumption at the base of this work is that in absence of external interference, the actual and control torques should be very similar. A significant difference between the two triggers a collision report and some control tactic is being deployed. Similarly, in [5], [6], [7] a residual-based method is used for collision detection and avoidance.

In addition to the detection of a collision state, an estimation of the external force acting on the robot links can be estimated, as shown in [8]. In a more recent work [9], the force estimation is augmented with an external Kinect sensor. This 3D video sensor helps in determining the exact contact point along the link on which the external interference is detected. The approach in our contribution differs significantly from these model based methods because we do not use a priori knowledge of robot dynamics. In particular, the above works require inertia parameters, such as link mass, for calculating the dynamics of the manipulator system. In contrast, in the work presented here, no external parameters are required and all of the method's parameters (the neural network weights) are learned during the training phase.

There are some additional works that aim to extract to the exact location of interference. The work of [10] uses a probabilistic approach in order to estimate where contact occurs. In contrast, in [11] a tactile sensor network is used to estimate the contact forces. In the work presented here, we focus on applying contact force to the robot end effector.

In [12], Machine Learning classifiers are trained on particular tasks, e.g., to detect events where the end effector reaches a barrier. Contrasting these results with those obtained through the conventional use of predefined torque thresholds for triggering identical state transitions, the research finds clear advantages towards the former in terms of execution speeds. The work therefore demonstrates the effectiveness of classifier based on torque readings. However, the scope of work is constrained to controlled and repetitive tasks. Our work goes a step further: the neural networks classify user interaction events in which the torque and manipulator positions are flexible, undetermined, and continuously varied, reflecting material production dynamics.

Rozo et al. [13] use an Hidden Markov Model operating on force/torque data alone to solve goal-driven manipulation tasks, which vary between executions. Training is based on demonstrations of the task using a haptic device. In our work, the data is used to predict the user's intention, using direct supervision. The trajectories are then computed based on these intentions.

The torque and position inputs that our method uses are already an essential component of current day joint mechanics since position and torque data feed the active compliance software at the base of some light-weight robots. For example, the DLR-KUKA Lightweight Robot and the DLR-HIT-Schunk Hand, obtain soft behaviors with an integrated approach that seamlessly combines software-based compliance, variable link stiffness, and joint mechanics [14]. In contrast to our work, the presented solutions are integrated rather than applied, and cannot offer comparable soft behaviors for existing robots. At the same time, the methods were not shown to be appropriate for heavy-weight, high payload and precision driven applications.

Within the focus of the present article, that of a software method for registering and acting upon external forces, SoftMove, by ABB Robotics, is of special interest [15]. Forgoing mechanical compliance add-ons and related investment costs, the software virtualizes soft robotics behaviors rather than physically altering the rigid joint configuration. Detection is torque based, with torque limits set for one distinct robot axis at a time. Thus conceived, the compliance is constrained to one Cartesian direction: detecting resistance along the vector direction, the manipulator's trajectory is terminated at the surface of the obstruction. Still, while ABB Robotics offers a software solution that softens the rigid joints of ABB manipulators without losing its rigid joint performance values, the setup is limited to one Cartesian direction, for which the original stiffness is varied. As described below, our approach achieves soft joint behaviors, and with it lead-through learning capabilities, by simultaneously tracking and predicting 10 direction vectors. In practice, these directions are enough to lead the robot freely in space.

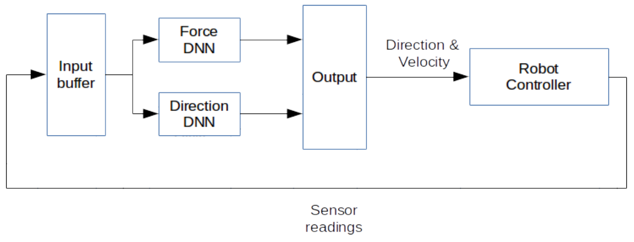

Fig. 1. The control loop. A VAL3 program running on the controller sends sensor readings to a PC program via TCP/IP. The signals are processed by two deep neural networks that detect the presence and the direction of external forces. The result is processed by an output module that produces new direction and velocity commands which are sent back to the controller

Deep neural networks are becoming increasingly popular in the field of robotics. Early use of neural networks in the field can be seen in [16], which uses neural networks in order to learn to follow a pre-defined trajectory that was created by an unknown set of control commands. In [17], a feedforward neural network is applied in the field of robot dynamics, in a specific system, where exact mechanical calculations are difficult to obtain. A soft conic manipulator driven by three cables is used. A neural network is trained to obtain seven parameters of the inverse kinematic model that extracts the forces acting on the robot's cables.

III. Our Approach

The control system that we present uses a neural network classifier in order to map the robot sensor readings to force vectors in 3D space. The basic setup includes Stäubli's medium payload arm TX90, with a CS8 controller running VAL3. Data was streamed by the LIVE TCP streaming platform [18]. The outline of this system is shown in Fig. 1.

The core of the system are two neural network classifiers. Both classifiers are Multi-Layer Perceptrons (MLPs), i.e., fully connected feedforward neural networks. The first MLP is trained to detect the presence of forces. It outputs a boolean value indicating whether an outside force is applied to the robot. The second MLP is trained to classify the direction of the outside force, based on a predefined set of direction vectors. During online classification, the force MLP is queried first to determine if an external force is being applied. If the answer is positive, the direction MLP is queried for the force direction.

Both MLPs employ 5 fully connected hidden layers with 100 neurons each. The ReLU activation function [19] is used for all layers. A subsequent output layer produces pseudo-probabilities using the softmax function.

We use a time-window approach to obtain the networks' input. The underlying assumption is that although some classification accuracy can be achieved by training only on the time frame for which prediction is requested, better results could be obtained when using information from past frames. Due to the nature of the classification task, the time-window used was relatively small.

The two MLPs were trained on the same input data, composed of the robot readings, available through the robot's API: (i) joint rotation values; (ii) torque values; (iii) flange

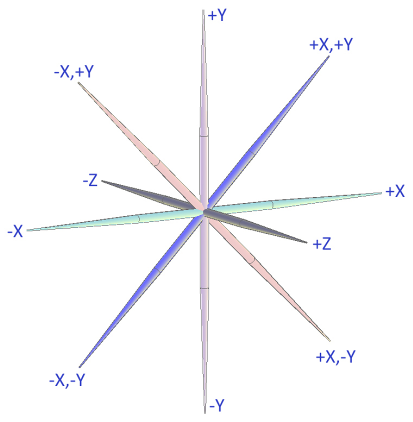

Fig. 2. The ten directions in space that are predicted by the second MLP.

position, orientation and velocity. Specifically, an input of size 440, which is a concatenation of the 11 recent time steps (including the current), is used at each time point. The input data at each time step consists of the following:

- 6 Torque values - one for each joint
- 6 Joint rotation values - one for each joint
- 3 Flange location in world coordinates (x,y,z)
- 3 Flange rotation in world coordinates (rx,ry,rz)
- 1 Flange velocity
- 1 Whether the robot is settled or not

In addition to the 20 raw signals above, the difference in values from the last time step was added to each data point. There are therefore a total of 40 features, that are collected at a rate of 25Hz.

The output of the first classifier is the probability of an external force being applied. The output of the second classifier is one of 10 direction vectors with respect to the flange coordinate system as illustrated in Fig. 2. These correspond to motion along the X,Y,Z axes in either a positive or a negative direction, as well as motion in each direction along the two diagonals of the XY plane.

A. *Training the System*

As is typical in machine learning approaches, large and comprehensive data sets are necessary in order to obtain good generalization from training data to test scenarios. In our case, the training data was collected by applying force by hand to a dedicated tool while moving the robot along random trajectories in 3D space.

The tool, which is shown in Fig. 3, contains six buttons. Instead of applying force directly to the flange, the force was applied, at each time point, to one of the tool buttons. This approach allowed rapid collection of sensor data points, labeled with the approximate force direction based on the button that was pushed. In order to move from six classes to ten, the tool was rotated 45 degrees in the XY plane.

Since the data was collected by hand, in order to reduce the time necessary for data collection, we constrained the data to a subset of possible robot configurations, namely a 3D area in front of the robot with the flange facing out. Fig. 4 depicts the random locations in 3D space from which the training

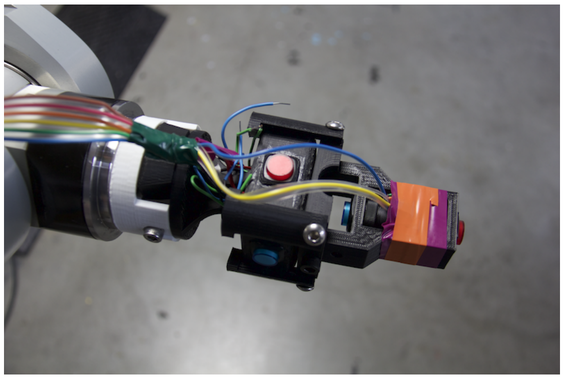

Fig. 3. The apparatus used to collect the training data. The tool has six buttons, four along the connection plane and two perpendicular.

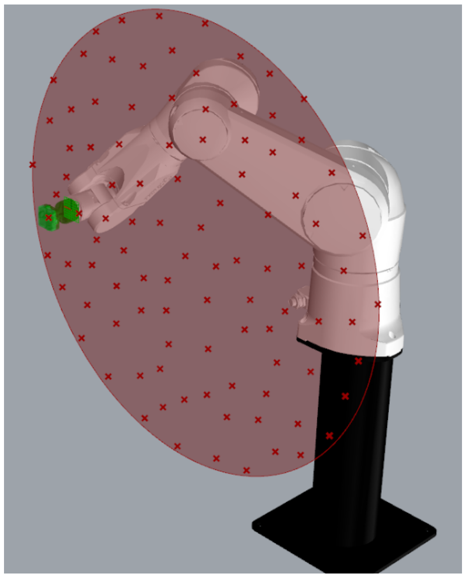

Fig. 4. The point cloud from which training data was collected. At each location, force readings at multiple directions were collected.

data are collected. Collection was also performed when the flange was moving in between the 3D points. The entire training data were collected in less than ten hours, and consist of 747,000 data points. Note that each push action creates multiple samples, and that data points are also collected when no force is applied.

The neural networks were implemented using the keras deep learning framework [20]. The training of the neural networks was done using the Stochastic Gradient Descent method with momentum. Learning rates of 0.1, 0.01, 0.001 were used, each for 40 epochs[1]. A batch size of 512 and a momentum value of 0.9 were used. A random subset of 90% of the data was used as the training set, and a well-separated random subset of 10% was used as the validation set. The same train/test split was used for testing multiple network architectures, as detailed in Sec. IV-A.

B. *Refinement of Classifier Output*

While the MLPs provided reasonable predictions, we found that added robustness can be gained by incorporating

[1]One pass over the entire training data is called an epoch.

TABLE I. Validation errors of alternative neural networks. LSTM1L400 is an LSTM model with a single layer of size 400. LSTM2L200 is an LSTM model with 2 layers of size 200. LSTM2L300 is a two layer LSTM with layer size 300. LSTM5L100 - five layer LSTM with layer size 100. MLP4L, MLP5L, and MLP6L are MLP networks with layer size 100 and a total of 4,5 and 6 layers respectively (MLP5L is the network used throughout this work). MLP5L_ND is a five layer MLP with layer size 100 but trained without the signal difference features. Each network was trained for two separate classification tasks, detecting external force existence ('Force' column) and classification of input signal to one of the ten direction vectors ('Direction' column). The 'Combined' column shows the error after combining the output from both networks to a single 11-class label. The values show mean error across the participating classes.

Method	Force	Direction	Combined
LSTM1L400 (1 x 400)	0.128	0.070	0.141
LSTM2L200 (2 x 200)	0.125	0.085	0.153
LSTM2L300 (2 x 300)	0.140	0.064	0.136
LSTM5L100 (5 x 100)	0.200	0.072	0.196
MLP4L (4 x 100)	0.049	0.026	0.062
MLP5L (5 x 100)	**0.038**	**0.016**	**0.045**
MLP6L (6 x 100)	0.052	0.027	0.066
MLP5L_ND (5 x 100)	0.049	0.022	0.062

two simple heuristics. These heuristics help to compensate for the unavoidable gap between the data collected during the training phase and the actual data seen at deployment.

First, the threshold of the force/no-force classifier was lowered in order to allow a better action detection rate, at the expense of increasing the number of false positives. This considerably improves sensitivity to light touch.

Second, in order to reduce noise, we used a buffer storing the five previous classifications. The actual classification is taken to be the absolute majority label (if it exists) from that buffer. In other words, if the first MLP predicted the existence of an external force, the direction is recorded. Out of the last five readings, if the same direction was obtained three times, the system recognizes an external force in this direction. Otherwise, the system reports no external force.

The two heuristics operate in different directions: while the first supports more liberal predictions of force, the second is more conservative. The obtained system is reliable with respect to the direction of the motion and is responsive to outside forces. Changing the above parameters controls the system's sensitivity and the output direction's reliability.

Once a force is detected, the robot is programmed to move in a constant velocity along the direction of the detected force. Fig. 5 shows the typical behavior of the final online system as the flange is being moved by a person.

IV. Results

A. Classifier Training

We experimented with two types of neural networks: (i) the feedforward MLPs described above, and (ii) recurrent networks, which are designed for time series. Specifically, we also tested the performance of LSTM (Long-Short Term Memory) networks [21], which is a type of recurrent neural network that has been shown to achieve good results in many time-series tasks [22], [23]. In the past, it was demonstrated [24] that a time-windowed MLP may perform better than LSTM in some but not all cases.

We tested multiple architectures with varying depth and number of hidden units per layer. Note that a hidden unit in an LSTM network contains four times the number of parameters of a hidden neuron in an MLP.

Table I depicts the results obtained for multiple network configurations. In these results, the MLP networks seem to outperform the LSTM ones by a large margin. The results also indicate that detecting the presence of an external force is a more difficult task than distinguishing the force vector. This is despite the former being a binary classification task while the latter is a multiclass one. Note, however, that the error rates in both tasks are rather low, varying from 1.6% to 4.5% for the best network, depending on the task. These numbers are prior to the refinement described in Sec. III-B.

The proposed network of 5 MLP layers of size 100 outperforms the other architectures tested. As mentioned, it has an input vector of size 440. Half of this consists of the raw signals and half contains the difference in values from the last time step. In order to demonstrate the contribution of the second half, Table I also contains the results of a network trained to predicted based solely on the 220 raw signals. This network is significantly outperformed by the network that employs the full input vector.

Below we focus on the deep MLP network with five hidden layers. Fig. 6 (a) and (b) show the training and validation results as a function of the training epoch. As can be seen in Fig. 6(a), the binary network that captures the existence of force somewhat overfits the training data.

Fig. 6(c) displays the ROC curves obtained for the force classifier. The y-axis is scaled to start at a true classification rate of 90%, and the overall performance is good. The plots suggest that force detection with a static manipulator is slightly easier than detection when in movement.

Table II displays a confusion analysis of the errors over the eleven classes: the no-force case and the ten directions. It is clear that the directions are mostly not confused among themselves, and confusion mostly occurs with regards to the presence of a force.

B. System behavior

With the online system in place, the robot exhibits much of the desired properties: it is sensitive to touch and moves in the direction of the applied force, up to the limitations imposed by the ten direction vectors we used. The force required to move the robot can be as light as a touch of a hand resting on the flange. However, it seemed that due to a slight response lag, a person would usually apply more force than necessary to move the robot. The classifier gave good results in various speed settings.

Fig. 7 shows an example of the torque signal data and the classifier output. The joint torque signals vary substantially regardless of whether the robot is stationary or moving and whether force is applied or not on the robot. It does seem, however, that the neural networks are able to properly determine the force from the signal.

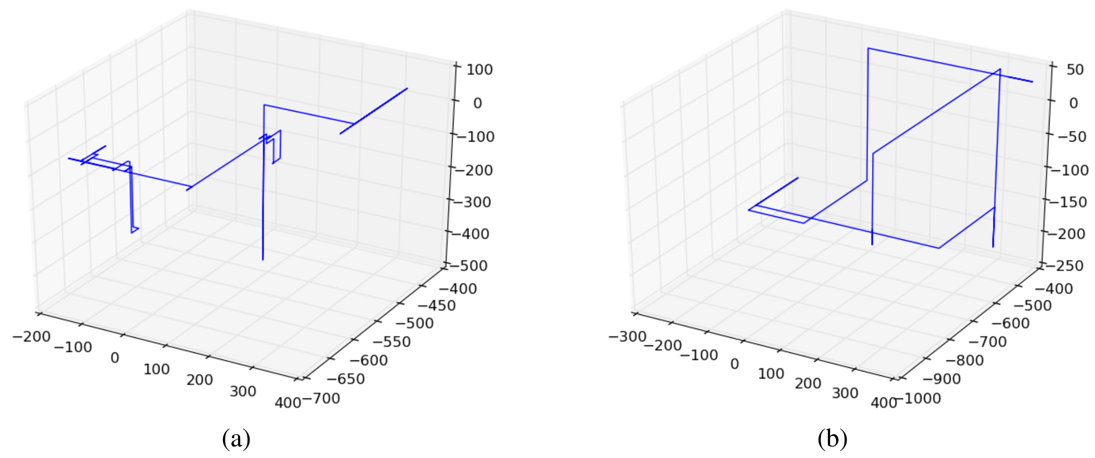

Fig. 5. Example of flange trajectory while running the online control system. The robot is being moved by push-pull forces applied by hand to the flange. In (a), some noise can be observed near points of direction change.

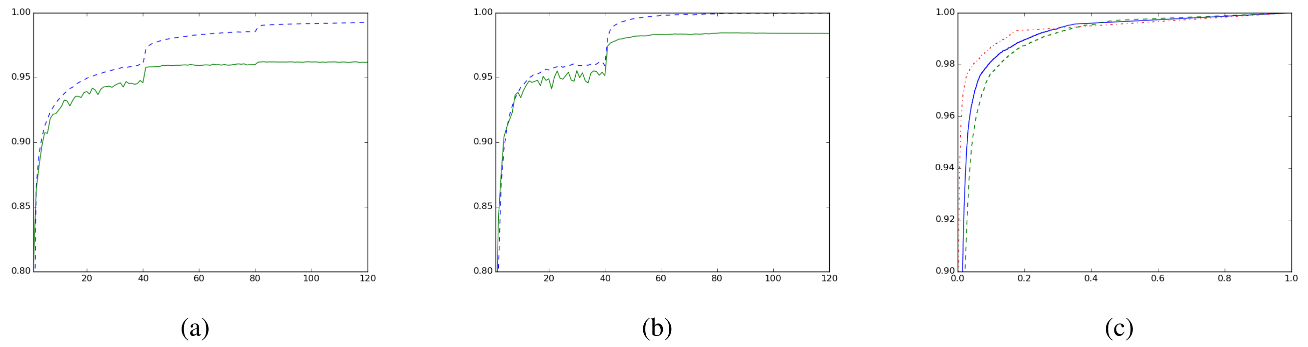

Fig. 6. (a) The train and validation accuracy for the external force MLP classifier as a function of the training epoch. The dashed line depicts the train error; while the solid line depicts the validation error.(b) The train and validation accuracy for the direction MLP classifier as a function of the training epoch. (c) An ROC curve for the MLP force classifier (false positive rate vs. true positive rate). The solid line shows the overall ROC, the dashed line is for the force signal applied while the robot is in motion and the semi-dashed is for the force signal applied while the robot is stationary

TABLE II. The confusion matrix of the combined output of the MLP classifier networks. The row labels are the actual classes; the column headers present the predicted classes. See Fig. 2 for an illustration of the ten direction labels.

	No Force	+X	-Y	-X	+Y	+X,-Y	-X,-Y	-X,+Y	+X,+Y	+Z	-Z
No Force	0.96	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01
+X	0.04	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
-Y	0.04	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
-X	0.04	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00	0.00
+Y	0.03	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00	0.00
+X,-Y	0.02	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00	0.00
-X,-Y	0.03	0.00	0.00	0.00	0.00	0.00	0.97	0.00	0.00	0.00	0.00
-X,+Y	0.02	0.00	0.01	0.01	0.00	0.00	0.00	0.96	0.00	0.00	0.00
+X,+Y	0.02	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.96	0.00	0.00
+Z	0.06	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.94	0.01
-Z	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.94

Fig. 8 shows a closer look at the classifier behavior when the flange is being pushed in the positive X direction. The classifier provides mostly correct predictions throughout the movement. Small time lags can be seen at the start of force application and at the end during which the classification system is misclassifying the input. The two heuristics described in Sec. III-B are being used here, and contribute to the observed lag duration.

When at rest, the robot shows occasional random noise related moves. This could be attributed to internal joint forces or to misclassification of the sensor signals. In turn, tasks requiring considerable precision may require a velocity control. These velocity controls can be implemented either manually or heuristically based on touch duration.

We also noticed that the classifier's sensitivity and noise behavior are closely related to the robot's configuration. These correlations are to be expected since different configurations entail different rest torques in the robot joints. In addition, it may indicate less than optimal generalization of the classifier model in configurations lacking training data,

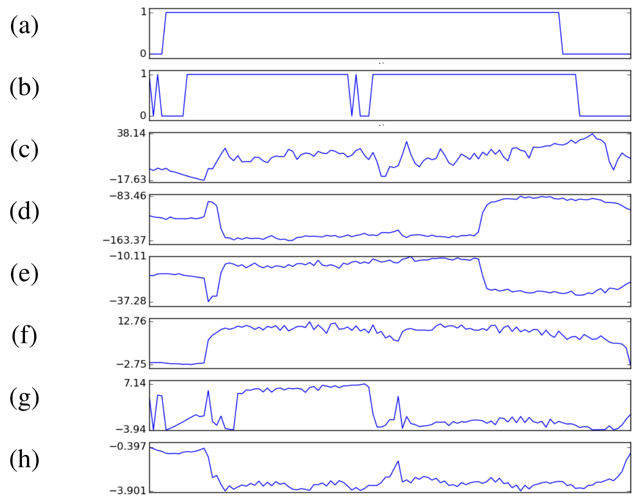

Fig. 7. Torque sensor readings when force is applied to the flange in the X positive direction (a) Force signal: 1 when force is being applied (b) Classifier output: 1 when the combined classifier returns the correct direction label, 0 otherwise. (c)-(h) Torque readings for each of the 6 joints.

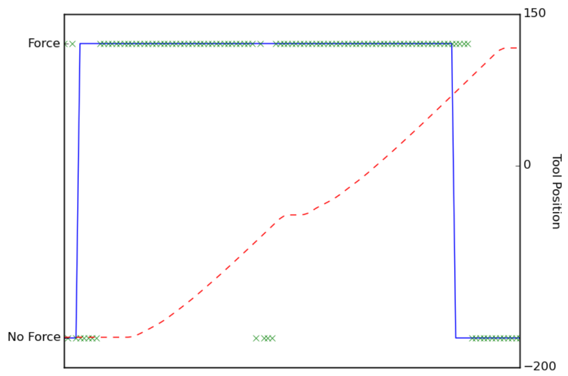

Fig. 8. The classifier output during a push event in the positive X direction. The solid line depicts when a force is being applied to the robot flange; the + ticks depict the classifier output (correct/incorrect) during this period; the dashed line shows the robot flange movement along the X axis. A small lag appears between the actual force and the classifier detecting it, resulting in a corresponding lag in movement response.

e.g., at the limits of the manipulator's range of motion.

C. Sample tasks

We evaluated the online system on a few tasks involving lead-through of the robot by a human towards some goal. Each experiment was repeated by two human operators interacting with the robot, and was conducted multiple times.

1) Moving the robot among predefined points in space: In this experiment, four predefined points are randomly chosen in space in front of the manipulator. The only constraint is that all points are reachable by the robot end effector. We used plastic cups hanging on threads to signify the point locations. The goal of a human user is to guide the flange by hand as close as possible to each of the points.

The results convincingly show that we were able to bring the flange to the desired locations without difficulty, for any set of points and regardless of the order of points. This was done by moving the arm by hand rather than using the teaching pendant or any other additional means.

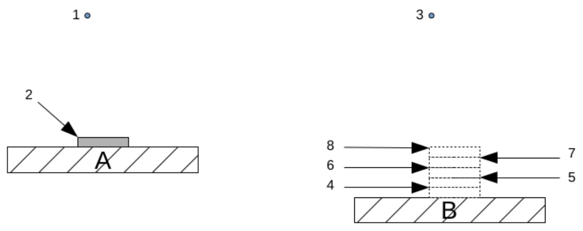

Fig. 9. Pick and place task. Metal disks are picked from A using a suction cup tool attached to the end effector and moved to B where they are stacked on top of each other. The human leads the suction tool by hand to the marked positions and records the robot configuration at each point. The robot is then switched to playback mode and instructed to move through the recorded positions in order 1,2,1,3,4,3,1,2,1,3,5,... and so on, picking the disks from A and stacking them at their corresponding heights at B. In this setting, we assume that disks are fed to point A from an external source rather than being stacked there. Thus, there is only one pick position (2).

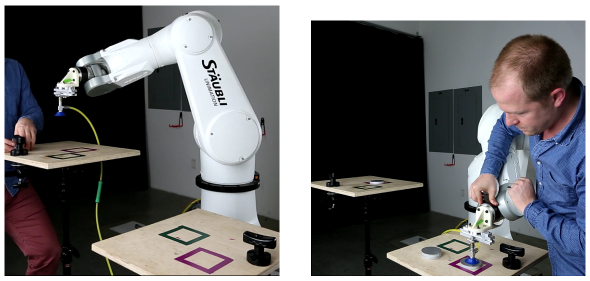

Fig. 10. The pick and place experiment setup

2) Teaching a pick and place task: In this setting, a user guides the robot through the desired key positions and records each key position as it is reached. Once the entire sequence is recorded, the robot can be switched into playback mode, moving throughout the predefined sequence of positions as expected.

We chose a pick-and-place stacking task for this experiment. For this task, two locations are arbitrarily selected in the space in front of the robot - a pickup location and a destination. The task involves moving items from the pickup location and stacking them at the destination point. We used five 70mm metal disks as the designated items to be moved and an air-pressured suction cup was used as the gripping mechanism. The suction cup was attached to the flange such that it points down and can uplift the disks.

The experiment was conducted in two stages. In the recording stage, a human guides the manipulator through a series of predefined key positions. An outline of the positions to be taught is shown in Fig. 9, and the experiment is depicted in Fig. 10. As each position is reached, a record signal is sent to the software control program, which records the current configuration of the manipulator and assigns it a label. Once all of the positions are recorded, the robot is switched to a 'playback' mode in which the control program instructs it to move through the recorded positions in the order appropriate for completing the stacking task.

In this experiment, beyond the lead-through interface, explicit signals had to be passed to the software control program to record a position, switch the robot mode from recording to playback, and repeat the recorded sequence when in playback mode. In these cases, we used custom buttons on the teaching pendant, which when pressed sent the corresponding signal to the program. In addition, air pressure of the suction cup was to be turned on and off at the appropriate locations. Although it would be straightforward to expand our control program to record turning the air pressure on/off in recording mode, similar to how positions were recorded, we chose not to implement this and rather turned the air on and off manually during the experiments.

The setup allowed the user to train the robot to perform the task with minimal effort, and the task was subsequently performed with full success (see video supplementary).

V. Discussion

We advocate for an AI (specifically Machine Learning) solution for the inference of peripheral forces from internal states. Perception capabilities, e.g., in the domain of video analysis, are going through a phase of rapid development, primarily due to the advent of deep learning. Using a similar set of techniques, applied introspectively to the existing inner sensors of a manipulator, we are able to empower existing robots with new and desirable behaviors.

Basing our method on Machine Learning rather than physical modeling, allows us to avoid the assumptions and approximations associated with the latter. However, learning algorithms require the collection of training data. We have demonstrated that using an appropriate apparatus, we are able to collect such data easily and efficiently.

The platform we rely on, namely Stäubli's medium payload TX 90 arm, did not offer any praticular advantage with respect to comparable medium payload systems, and was used solely due to its availability to the research team at the SCI-Arc robotics lab. We conjecture that any system that enables control and inner sensor reading through an API would support the successful application of our methods.

VI. Conclusions and Future Work

We presented an online method for detecting outside forces and reacting in a way that provides compliant behavior. We show that Machine Learning techniques can be successfully applied to the robot sensor data without much preprocessing or feature engineering. A combination of two neural networks provides us with accurate predictions. Several network architectures are tested, and a time window based approach seems preferable to a leading recurrent neural network technique. Since we have used only the basic internal readings from the robot sensors, our method may be applicable to other rigid servo based robots.

As future work, we are interested in inferring the user's subsequent intentions and not just the current actions. For example, the user might start reducing the amount of applied force slightly before arriving at the desired manipulator destination, allowing the software to anticipate the upcoming event. As always with cobotting, safety concerns are to be addressed. We are similarly interested in applying our methods to the problem of collision detection, which would allow for a safer human-robot interaction. Thus conceived, the flexibility of software based solutions could lead to a continuous collaborative environment of introduced work flows that are always evolving, unrestricted by temporal boundaries of task recording and playback.

References

[1] E. Eitel, "The rise of soft robots and the actuators that drive them," *Machine Design*, 2013.

[2] D. Ebert and D. Henrich, "Safe human-robot-cooperation: Imagebased collision detection for industrial robots," in *IEEE/RSJ Int Conf. on Intelligent Robots and Systems*, 2002, pp. 239–244.

[3] R. Ortega and M. W. Spong, "Adaptive motion control of rigid robots: A tutorial," *Automatica*, vol. 25, no. 6, pp. 877–888, 1989.

[4] S. Morinaga and K. Kosuge, "Collision detection system for manipulator based on adaptive impedance control law," in *ICRA*, 2003.

[5] A. D. Luca and R. Mattone, "Sensorless robot collision detection and hybrid force/motion control," in *ICRA*, 2005.

[6] A. D. Luca, A. Albu-Schäffer, S. Haddadin, and G. Hirzinger, "Collision detection and safe reaction with the dlr-iii lightweight robot arm," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2006.

[7] S. Haddadin, A. Albu-Schäffer, A. De Luca, and G. Hirzinger, "Collision detection and reaction: A contribution to safe physical human-robot interaction," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2008, pp. 3356–3363.

[8] A. D. Luca and F. Flacco, "Integrated control for pHRI: Collision avoidance, detection, reaction and collaboration," in *IEEE Int. Conf. on Biomedical Robotics and Biomechatronics*, 2012, pp. 288–295.

[9] E. Magrini, F. Flacco, and A. De Luca, "Estimation of contact forces using a virtual force sensor," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014, pp. 2126–2133.

[10] A. Petrovskaya, J. Park, and O. Khatib, "Probabilistic estimation of whole body contacts for multi-contact robot control," in *ICRA*, 2007.

[11] A. D. Prete, L. Natale, F. Nori, and G. Metta, "Contact force estimations using tactile sensors and force/torque sensors," in *Human Robot Interaction*, 2012.

[12] A. Stolt, M. Linderoth, A. Robertsson, and R. Johansson, "Detection of contact force transients in robotic assembly," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2015, pp. 962–968.

[13] L. Rozo, P. Jiménez, and C. Torras, "Robot learning from demonstration of force-based tasks with multiple solution trajectories," in *International Conference on Advanced Robotics (ICAR)*, 2011.

[14] A. Albu-Schäffer, O. Eiberger, M. Grebenstein, S. Haddadin, C. Ott, T. Wimböck, S. Wolf, and G. Hirzinger, "Soft robotics: From torque feedback controlled lightweight robots to intrinsically compliant systems," in *Int. Conf. on Control Automation and Systems*, 2010.

[15] "SoftMove: cartesian soft servo," https://library.e.abb.com/public/74f4e5050f189f82c12573f00054efd0/Data%20sheet%20SoftMove%20LR.pdf, accessed: 2015-09-13. ABB, Västerås, Sweden. 2008.

[16] T. D. Sanger, "Neural network learning control of robot manipulators using gradually increasing task difficulty," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 3, pp. 323–333, 1994.

[17] M. Giorelli, F. Renda, G. Ferri, and C. Laschi, "A feed-forward neural network learning the inverse kinetics of a soft cable-driven manipulator moving in three-dimensional space," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2013, pp. 5033–5039.

[18] C. Batliner, J. M. Newsum, and M. C. Rehm, "Live," in *Robotic Futures*. Tongji University Press, 2015.

[19] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Int. Conf. on Artificial Intelligence and Statistics*, 2011.

[20] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18.5, pp. 602–610, 2005.

[23] F. Gers, "Long Short-Term Memory in Recurrent Neural Networks," Ph.D. dissertation, Federal Polytechnic School of Lausanne, Department of Computer Science, Lausanne, Switzerland, 2001.

[24] F. A. Gers, D. Eck, and J. Schmidhuber, "Applying LSTM to time series predictable through time-window approaches," in *Artificial Neural Networks-ICANN*. Springer, 2001, pp. 669–676.