2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
September 24–28, 2017, Vancouver, BC, Canada

Detecting insertion tasks using convolutional neural networks during robot teaching-by-demonstration

Etienne Roberge[1] and Vincent Duchaine[1]

Abstract— **Today, collaborative robots are often taught new tasks through "teaching by demonstration" techniques rather than manual programming. This works well for many tasks; however, some tasks like precise tight-fitting insertions can be hard to recreate through exact position replays because they also involve forces and are highly affected by the robot's repeatability and the position of the object in the hand. As of yet there is no way to automatically detect when procedures to reduce position uncertainty should be used.**

In this paper, we present a new way to automatically detect insertion tasks during impedance control-based trajectory teaching. This is accomplished by recording the forces and torques applied by the operator and inputting these signals to a convolutional neural network. The convolutional neural network is used to extract important features of the spatio-temporal forces and torque signals for distinguishing insertion tasks. Eventually, this method could help robots understand the tasks they are taught at a higher level. They will not only be capable of a position-time replay of the task, but will also recognize the best strategy to apply in order to accomplish the task (in this case insertion). Our method was tested on data obtained from 886 experiments that were conducted on eight different in-hand objects. Results show that we can distinguish insertion tasks from pick-and-place tasks with an average accuracy of 82%.

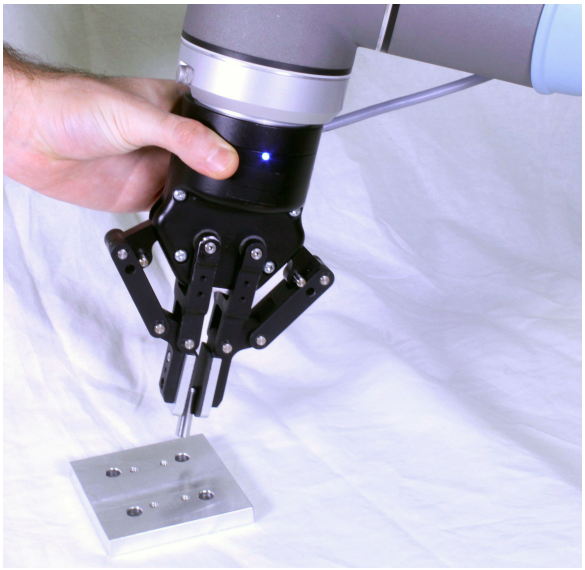

Fig. 1. Insertion task taught through impedance control.

I. INTRODUCTION

A recent trend in industry is the integration of collaborative robots into a variety of processes. This generation of robots is often present alongside workers in assembly lines to assist humans with repetitive tasks. The same robot may perform many different tasks in various workspaces during its operational lifetime. However, getting a robot to perform as expected typically requires a highly skilled programmer to manually program the robot's actions according to constraints imposed by its environment. If you add multiple robots to the equation, programming becomes a lengthy task requiring precise knowledge about the specific models of robot and the tools they will use.

In the search for ways to avoid manually programming the robot every time it must learn a new task, multiple studies have been done on the concept of "programming by demonstration" [1]. Peternel and al. [2] recently proposed a teleoperated teaching approach that uses force feedback to program new tasks. This procedure is particularly relevant to situations where the direct presence of a worker is not possible. The movement of the teacher can be remotely coupled with the control of the robot and registered for a future replay.

A more direct approach is kinesthetic teaching [3], which consists of teaching tasks using direct physical interactions between the teacher and the robot. This method has become increasingly common with the introduction of collaborative robots, most of which use admittance/impedance control-based algorithms for the teaching control mode [4]. One does not need prior expertise in programming or robotics to know how to guide the robot by holding its effector. However, although the teacher's role can be performed by any well-trained worker, some actions can be too difficult for the system to recreate from a time-position replay (e.g., very precise movements with force dimensions, such as precise, tight-fitting (or simply "tight") insertions).

Numerous works have attempted to overcome the challenge of tight insertion. Proposed methods include using force feedback [5], [6] or a combination of visual and force servoing [7] to resolve the uncertainty of the positioning during the assembly task. Although those methods are effective ways to perform the insertion task, they do not address the dilemma of detecting insertion tasks during teaching. What is needed is a way to automatically determine that an insertion was performed during the kinesthetic teaching phase, before the aforementioned insertion techniques can be used.

In this paper, we propose an insertion task detection system based on convolutional neural network (CNN) ar-

[1]The authors are with the Control and Robotics Laboratory (CoRo), Faculty of Automated Production Engineering, École de technologie supérieure (ÉTS), Montréal, Québec, Canada etienne.roberge.1@ens.etsmtl.ca vincent.duchaine@etsmtl.ca

978-1-5386-2682-5/17/$31.00 ©2017 IEEE

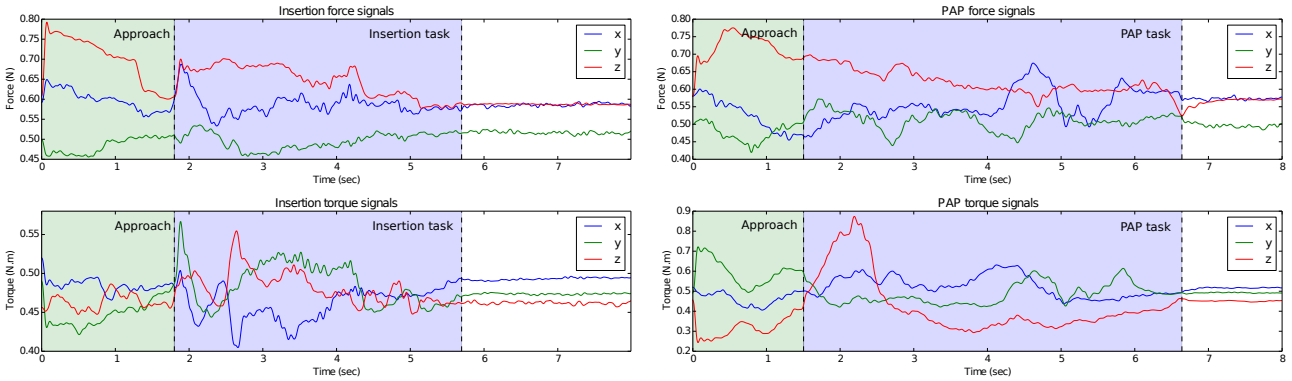

Fig. 2. Examples of normalized insertion and PAP signals.

chitecture that could be integrated in most robotic teaching frameworks. Our method uses the force and torque (F/T) feedback from the operator and environment during the demonstration stage, and involves analyzing this data to extract features associated with the action of inserting an object. After detecting this assembly event, the robot will automatically replace that portion of the taught routine with a more generic insertion method [5], instead of using an exact replay of the taught path, to ensure the success of the cycle. Please note that although the use of the CNN is a key part of our work here, the main contribution of this paper is not in applying a CNN to another application, but in developing a new method of detecting insertion tasks—the CNN is merely a means to this end.

The CNN architecture has proven to be an impressive machine learning technique in multiple fields. Examples of effective signal processing can be seen in solutions to musical and speech recognition problems [8–11], as well as a more spatio-temporal signal processing problem [12]. The use of this feature extraction technique is justified by the nature of the signal, since in our experiments, the same F/T sensor is used during the demonstration phase with impedance control and the data acquisition. Using the same F/T sensor means the dynamics of both the control and the external interaction of the end-effector are superposed.

In Fig. 2, we can see the resulting forces and torques generated during an insertion task and a pick-and-place (PAP) task. Solely by looking at the figure, it is quite difficult for one to determine the features that would characterize each task. A CNN could be used to automatically discover and learn the necessary features to discriminate between the tasks. To the best of our knowledge, this paper represents the first time an automatic insertion detection system has been applied to robot teaching methods.

To evaluate our approach, we used eight different objects to create a database containing F/T signals of insertions, pick-and-place tasks, and random movements done with eight different in-hand objects. We give an overview of the theoretical details of the applied techniques in Section II of this work. Section III describes the hardware and software and the data acquisition methods that were used during the experiments. Section IV presents the obtained results. Finally, Section V concludes with a discussion of the impact of this work and potential related future projects.

II. METHOD

The main goal of this work is to automatically determine the presence of an insertion task during kinesthetic teaching. Once a robotic system can automatically determine the moment when an insertion has occurred in the position-based demonstration, it may become possible for robots to replace the position-based insertion part of the demonstration with a solution that is better-suited to the task. We believe that this could be possible by studying the F/T signal feedback from the teaching phase to distinguish between insertion tasks and other tasks.

In our experiments, a CNN classifier is used to distinguish insertion tasks taught with impedance control. This type of classifier is known to be a powerful way to extract features in spatio-temporal signals. Works show that feature extraction through CNN is more appealing than hand-crafted feature methods because the latter requires specific knowledge of the nature of each signal [12]. Also, the CNN classifier is interesting to use in our particular case, since we want to detect patterns over spatio-temporal force/torque signals, which can mostly be represented by the generated features of the network.

The method proposed in this paper is to acquire the forces exerted by the operator on the robot during teaching as if they were the representation of the human's motor control model output. Several works support the idea that the human central nervous system stores motor control skills as models [13–15]. Assuming that those models can be captured through the force exerted on the robot, it might be possible, with neural networks, to differentiate those models from one another and to detect the intended insertion task.

We know that any task is a combination of multiple small movements. For example, a tight-insertion task consists of the following steps: approach, angular and translational alignment of the assembly, insertion, and halting of the movement (which we will call *the stop*). We will capture the F/T profile of the input trajectory of each step involved

in a task. Then we will test our hypothesis that CNNs are good for detecting the patterns in the F/T signal that are related to those steps.

In the next section we will briefly outline the concept of CNNs, and then explain how ours was trained and applied to our dataset.

A. Feature learning with CNNs

CNNs typically consist of one or more feed-forward convolutional layers combined with sub-sampling layers, which are often followed by fully connected layers. What makes a CNN unique (as compared to other neural networks) is that in each convolutional layer, each neuron is only locally connected to a subset of the previous layer, and it shares its connection weight with all other neurons in the same feature map. The output feature map is the combination of all the overlapping neurons sharing the same weight. In other words, the output feature map is the convolution of a filter over the entirety of the data, and it gives the spatial reaction of the kernel over the input. The use of multiple filters then creates different feature maps reacting to different patterns, and summarizes the input in terms of combinations of pattern reactions over maps indicating the position of those patterns.

The feature maps created after a convolution layer are generally fed into a max-pooling or average-pooling layer to reduce the size of the data and also to produce some shift invariance [16]. For example, let $X^{n\times a\times b}$ — where n is the number of feature maps with size $a \times b$ — be the input to a non-overlapping pooling $k \times j$ operation. This results in the output data being condensed to $X^{n\times \frac{a}{k}\times \frac{b}{j}}$.

The last pooling layer of the CNN is usually flattened and fed into a fully connected neural network (NN) layer to get the final predicted class of the input using a softmax activation function. The fully connected layer is used to combine the information from each feature map of the previous layers.

B. Applying a CNN to F/T signals for insertion task detection

To use a CNN for insertion detection, we trained the system with data from a time frame focusing on the whole insertion task, beginning with the approach and ending with the stop. We then generated a network that labels the fed signal in a binary way.

Consider two groups of three signals: $F_x, F_y, F_z \in \mathbb{R}^{n\times m}$ and $T_x, T_y, T_z \in \mathbb{R}^{n\times m}$, where n is the number of trials of force and torque sensor signals, and m is the length of time of the signal. Assume a matrix $X \in \mathbb{R}^{n\times m\times 6}$ to be the result of the concatenation of all six signals ($F_{x,y,z}$ and $T_{x,y,z}$). X is the input fed to the network that considers each F/T signal as an input channel. Finally, let $Y \in \{0, 1\}^n$ be the labels of X, where for the i-th input, $y_i = 1$ is the detection of an insertion task.

Typical CNN applications, such as in computer vision, require the use of a square filter. However, in our case we have one-dimensional signals, so the height of the filters is of course one. This detail simplifies the adjustment of the filter's parameters, since only the width of the filters needs to be set. The same simplification applies to the pooling layers of the network.

One technique, which we have borrowed from the field of computer vision, relates to the way a CNN usually manages color images. During convolution, filters have connections to all the layers of the image in its own neighbourhood, with the layers being the red, green, and blue channels of an RGB image. In our case, concatenating every force and moment signal in a 6-channel matrix means the filters can access all the F/T features in a certain time frame.

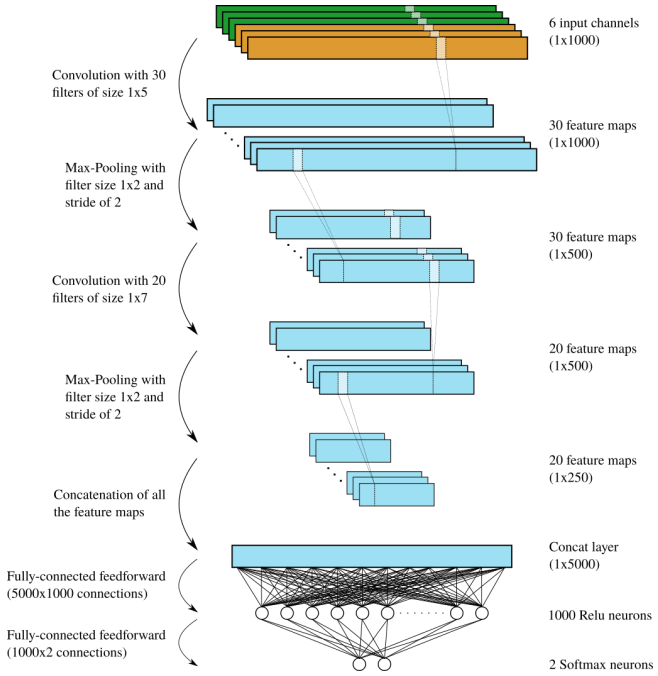

Fig. 3. CNN architecture used in this work. Labels beside arrows at left indicate operations; labels beside blocks at right indicate layers.

C. Proposed approach

The main goal of this work is to detect insertion tasks through the robot's regular teaching-by-demonstration sequence. To that end, the operator first shows the robot the task to be done by using the F/T sensor to move the robot's end-effector. While the teacher demonstrates the task to the robot, the complete set of sensor signals and the robot's positions are saved. The F/T signals are then used to find out if an insertion task was done during the teaching of the robot sequence. To do so, the data are first divided into time samples of $8s$ with a stride of $0.08s$. This generates windows of signals that are then input to the pre-trained CNN to determine if any of the signal windows relate to an insertion task. Ultimately, if a signal window is identified as indicative of an insertion, then the robot could learn to replace the operator's generated path for that time frame with a more advanced and generic approach to the insertion, for example a spiral-search-type insertion [5]. Fig. 4 depicts a diagram of the proposed approach.

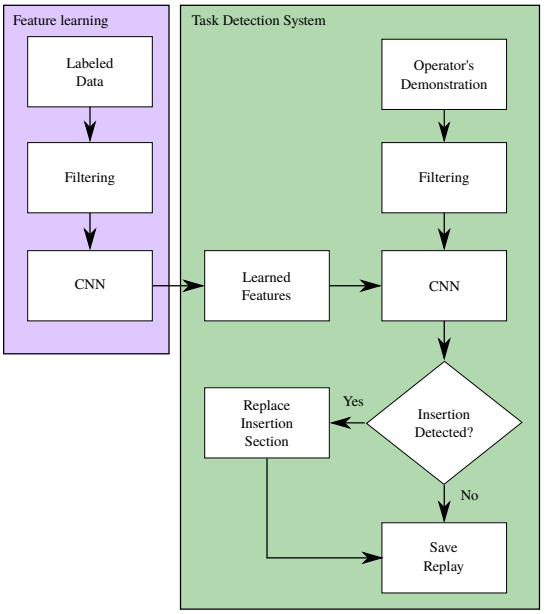

Fig. 4. Steps of the proposed approach.

Of course, a CNN must first be trained to apply the right filters (in this case, to filter the features that characterize insertions). So Section III describes how data were acquired for training and testing the CNN, and Section IV details the training process and the results of testing the CNN.

III. EXPERIMENTAL SETUP

A. Data acquisition

To acquire the training and test data, we set up a UR5 manipulator from Universal Robots, an FT 300 sensor, and a 2-finger gripper from Robotiq, Inc. The same F/T sensor was used for impedance control and data acquisition, meaning that all the dynamics of the control from the operator were mixed with the dynamics of the action performed. For example, if the operator caused the end effector to move across a hard surface such as a table, then for our purposes the operator's interaction with the end effector was inseparable from the end effector's interaction with the table.

We used eight objects for the insertion and PAP exercises. We conducted a total of 886 experiments: 302 insertions, 453 PAP, and 131 random movements. Since the goal of this work is to detect insertion tasks, we divided the experiments into two classes: insertions (class 1), and the group of PAP and random movements (class 2).

To create the dataset of class 1, we performed tight insertions of the dataset objects into a rigid slot. To do so, we attached a mechanical vice to a table, and fixed the slotted base of the assemblies in the vice. The object to be inserted was firmly grasped by the 2-finger gripper and guided during insertion by the operator using impedance control. The last eight seconds of data (sensor signal and robot position) from each insertion task were saved. The PAP portion of the class 2 data was collected in a similar manner, except the objects were placed on a precisely marked spot on a flat surface instead of being inserted in a slot. Lastly, the random movement portion of the class 2 data were generated by having the operator move the UR5 manipulator in long continuous movements, short jerky movements, no movement, and by shaking it in multiple directions.

The data was collected using a Python program we wrote, running under Ubuntu 14.04 at a rate of 125 Hz. Every compiled sample contains $8s$ of data from every direction of the FT sensor, resulting in $\mathbb{R}^6$ matrices of length 1000. These windows of data were obtained by saving a snapshot of the sensors after each experiment was done. The CNN, which was later used to analyze those inputs, was generated using the open-source machine learning library Tensorflow.

B. Data pre-processing

To reduce noise induced by the F/T sensor during teaching, we used a fourth-order Butterworth lowpass filter. This was done in the work of Myers and al. [17] on the *teach-by showing* technique, because as they state, the motion of human limbs while performing voluntary movement trajectories is around 5 Hz, whereas involuntary movements (reflexes) typically take place at around 10 Hz. Since we only need low-frequency information to characterize the operator's movements while teaching, we can remove any unnecessary information in the data that would disrupt the feature learning sequence. We set a cut-off frequency of 12 Hz with a fourth-order Butterworth digital filter.

IV. EXPERIMENTS

A. Training and test methodologies

To validate our feature learning approach, we used the leave-one-subject-out scheme, as was done in the work of Rad and al. [12]. This technique consists of creating a test dataset out of all the data obtained for one object, and using the remaining data for the training dataset. This ensures that the generated network has never seen any example of the one test object. The test database is then used to evaluate the generalizability of the system. In our experiments, only four of the eight objects (metal pin, PVC tube, light dimmer, and x-shaped insert) were used as a test set, because these were the only objects that had both insertion task data and PAP data. The training datasets were boosted by sliding every data sample to the right and left on the time frame by a random amount of time with a maximum of $1s$. This action tripled the amount of data in the training sets.

In an attempt to obtain a result to compare with our feature extraction methods, the first experiment we conducted was to classify the test object without feature extraction. To do so, we directly showed the samples of F/T sensor data to a fully connected feedforward neural network, which is similar to the one used in the output of the convolution layers of our CNN. This network consists of one hidden layer of 1000 neurons. It outputs to another layer of two softmax neurons, each of which is associated with a class of data, and outputs the predicted class of the original input signal. To match the input of the feed-forward neural layers of the CNN, we concatenated every direction of F/T sensor data to get an

Results	Experiments	Metal pin	PVC tubes	Light dimmer	X-shaped insert	Means
Accuracy	*Benchmark*	0.7±0.03	0.73±0.2	0.79±0.04	0.89±0.05	0.78
	CNN	**0.74±0.02**	**0.79±0.04**	**0.85±0.04**	**0.91±0.03**	**0.82**
Precision	*Benchmark*	0.65±0.02	0.9±0.04	**0.95±0.03**	1.00	**0.88**
	CNN	**0.68±0.02**	**0.91±0.0**4	0.89±0.03	1.00	0.87
Recall	*Benchmark*	0.89±0.06	0.41±0.07	0.53±0.11	0.77±0.12	0.65
	CNN	**0.96±0.02**	**0.57±0.09**	**0.73±0.09**	**0.8±0.07**	**0.77**

TABLE I

RESULTS OBTAINED DURING THE FOUR TEST EXPERIMENTS.

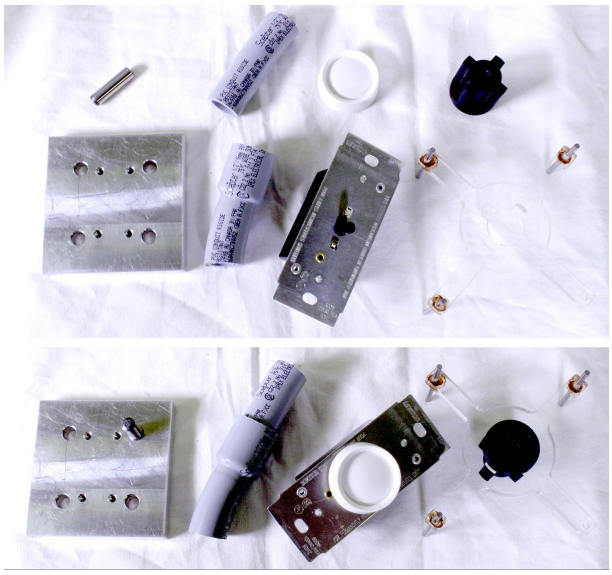

Fig. 5. Objects used in the test dataset. From left to right: metal pin, PVC tube, light dimmer, x-shaped insert.

array of 6000 data points in length. We will refer to this first experiment as the *benchmark* network for our CNN.

The second experimental procedure was for classifying the same test set as in the first experiment, but this time using our proposed feature extraction technique from Section II-C. The input of the network is fed in two sequences of convolution and pooling layers to extract the feature of the signal, and the generated feature maps are then flattened out to be fed to the feed-forward neural layers with the same layout as in the benchmark experiment. By comparing this work with the first experiment, we will be able to evaluate if feature extraction grants better results in terms of accurate insertion task detection.

In both experiments, we optimized our networks using an Adam algorithm [18] to compute gradients. To maximize the generalizability of the trained network and prevent the network from overfitting, we used a dropout rate of 0.3 [19] in every neural layer.

Since both networks involve randomness (such as the normal distribution of the weight initialization, and the stochastic first-order gradient-based optimization algorithm) the networks will not always give the same result (for the test set data) with each training. Thus, we generated ten different networks using the training dataset for every test object and evaluated the accuracy of each of those networks on the object dataset. We present the mean and standard deviation of the results in Table I, which we believe is a fair representation of the expected results should anyone attempt to recreate the experiment.

B. Experimental results

As can be seen from Table I, the general accuracy of our CNN, at 82%, beats the result obtained with the benchmark experiment, 78%, by a margin of 4 percentage points. The fact that the average result of our CNN is higher clearly shows the benefit of feature extraction. We can also consider the general accuracy result to be an indicator of the CNN's good performance.

If we look more closely at the results obtained with the test on the first object (metal pin), we can see that the precision of the network is rather low (68%) and the recall is reasonably high (96%), meaning most of the errors made by the system were false positives. This is probably because the movements generated during the approach step of the PAP task are similar to movements generated during the approach step of the insertion task. Since the object is small, and the PAP tests required the object to be placed in a very precise location, the placement of the object required a great deal of fine adjustments to be made during the approach step, and similar adjustments are required during the approach step of the insertion task.

The inverse phenomenon can be observed with object 2 (PVC pipes). This test resulted in excellent precision (91%) and a poor recall rate (57%), meaning that some of the insertions were not detected. This can be explained by the fact that performing the insertion action with the pipe was much easier than with the other objects. Since few adjustments were made while inserting one half of the pipe into the other, and the network considers adjustments to be the main characteristic of the insertion task, the lack of adjustments made this task difficult to detect.

V. EXAMPLE USE OF THE LEARNED FEATURES ON FULL DEMONSTRATIONS

The previous experiment showed how a CNN trained on insertion and PAP signals can be used to detect those tasks with good accuracy. However, the developed network was only capable of analyzing 8 seconds' worth (1000 data points) of F/T signal, which is a considerably less than what would be required to analyze the typical teach-by-demonstration task data. In this section, we explore how our

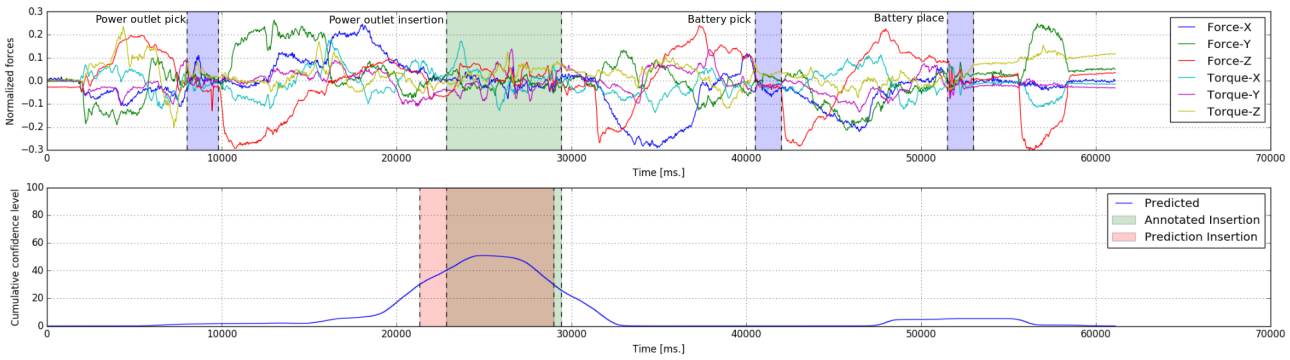

Fig. 6. Top: complete F/T signals for a kinesthetic teaching demonstration, with tasks annotated. Bottom: output vector containing the cumulative softmax values for insertion task detection.

network can identify an insertion task for input data of a much longer time frame.

A. Using the CNN for inputs of longer time frames

Since the trained CNN takes a fixed-sized input (of 8 s), due to its fully-connected layers, the lengthier (in terms of time frame) input needs to be segmented or redimensioned into a format that is compatible with the network.

One popular way of doing so is with the use of a region-based CNN (R-CNN) [21]. This technique, typically used in image processing for object localization, uses region proposal algorithms to select the potential positive regions of the image. These regions can be found using multiple techniques, notably objectness [23] or selective search [22]. However, since our data is not nearly as substantial as that used in the typical image processing problem, there is no need to use the region proposal method.

Instead, we simply segmented the whole signal into windows of 8 s (here equivalent to 1000 data points). The segmentation was done on the whole signal by convolving the input window over the signal with a stride of 10 (0.08 s).

After the segmentation is done, we input all the newly-segmented signals to the CNN, obtained the insertion task softmax output values, and accumulated the latter in an output vector.

In general, assuming the length of the demonstration is n, the steps above can be enumerated as follows:

1) Arrange the demonstration's force signals in a matrix $F^{6\times n}$.
2) Build an output vector y^n to accumulate the confidence levels of the CNN over the full length of the demonstration.
3) Make $n/10$ new matrices, $N^{6\times 1000}$, by sliding a window of length 1000 on F with a stride of 10.
4) Feed the new matrices N to the previously trained CNN and save the output value, r, of the insertion softmax neuron (this value is a confidence level between $[0, 1]$).
5) Add the value r to the corresponding values of the output vector y. For example, the output of matrix $N(0:1000)$ is added to $y(0:1000)$ and the output of matrix $N(t:t+1000)$ is added to $y(t:t+1000)$.

Following the steps above will result in an output vector of the same length as the input, and will indicate where, for the original signal, an insertion task is most likely to have occurred. Each point of the output vector is the sum of the results obtained by inputting 100 signals, each of which overlaps the given point in time of the input signal, to the pre-trained CNN. It is important to note that the first and last 1000 output points of the vector have smaller values than the rest, due to the fact that they contain the sum of fewer than 100 outputs.

Fig. 7. Objects used for the full teaching demonstration experiment. Top left: power plug; right: battery charger; bottom left: battery.

B. Example of CNN use for a full teaching demonstration

In this subsection, we describe how we used the method above to analyze the F/T signals of a complete demonstration. For this example, we took the signal from a demonstration where both an insertion task and a PAP task were performed. The objects used in this demonstration are shown in Fig. 7. The demonstrated task consisted of picking up the battery charger's power plug and slotting it into place on the charger, then picking up the battery and placing it on a precise spot on the workbench. As with the previous experiments, this demonstration was done using objects that were not used to train the network.

Fig. 6 shows the output vector of the complete demonstration, as well as the F/T signals associated with the demonstration. In this example, a cumulated value of 30 in the output vector was arbitrarily chosen to be the value that results in the detection of an insertion task.

We can observe in the output vector that surprisingly enough, the predicted time for an insertion task almost completely covers the annotated time for the insertion task. With a threshold value of 30, 93.8% of the annotated insertion was considered to be an insertion by the CNN, and only 19.5% of the predicted insertion region was not annotated as an insertion.

Although the insertion task was not perfectly detected, by knowing the general position in time of the task, it is possible to find the moment in that region where the robot stopped, and thus find the stop point of the insertion. By doing so, one could possibly replace that part of the demonstration in a position-based replay with a more suitable insertion technique.

The second thing we can observe is the small value added to the output vector during the PAP task. In this case, the output value of this region is clearly not enough to be considered an insertion task, and thus it is correctly classified.

VI. CONCLUSION

In this work, we have shown that it is possible to automatically detect insertion tasks while teaching a robot using impedance control. Our proposed method of using neural networks to analyze the forces and torques applied by the operator during teaching has shown promising results. Our results also indicate that feature extraction of the signals can slightly improve the overall detection rate compared to that of a simpler network.

Although the CNN method used in this paper looks promising for the detection of insertion task in kinesthetic teaching demonstrations, it is important to note that other types of neural networks could probably perform as well. Notably, recurrent neural networks, like LSTMs, have shown great results in time based tasks like speech recognition [24]. This type of network could be compared with the technique used here in future work.

In other eventual works, we could integrate different types of sensors in the loop, in addition to the current sensor, to improve the characterization of the task by the system. For example, another type of sensor was used by Roberge and al. [20], who found that the slippage of handheld objects on an external surface can be detected through tactile feedback. This tactile-feedback-based detection method could be used to strengthen the prediction rate of our work.

We believe the experiments shown in the paper indicate progress is being made toward automatic higher-level detection of tasks in kinesthetic teaching. The detection of other problematic tasks that are hard to repeat with time-position based replay might also be possible with techniques like the one shown here.

REFERENCES

[1] Billard, Aude, et al. "Robot programming by demonstration." Springer handbook of robotics. Springer Berlin Heidelberg, 2008. 1371-1394.

[2] Peternel, Luka, Tadej Petric, and Jan Babic. "Human-in-the-loop approach for teaching robot assembly tasks using impedance control interface." Robotics and Automation (ICRA), 2015 IEEE International Conference on. IEEE, 2015.

[3] Hersch, Micha, et al. "Dynamical system modulation for robot learning via kinesthetic demonstrations." IEEE Transactions on Robotics 24.6 (2008): 1463-1467.

[4] Duchaine, Vincent, and Clment Gosselin. "Safe, stable and intuitive control for physical human-robot interaction." Robotics and Automation, 2009. ICRA'09. IEEE International Conference on. IEEE, 2009.

[5] Jasim, Ibrahim F., Peter W. Plapper, and Holger Voos. "Position identification in force-guided robotic peg-in-hole assembly tasks." Procedia Cirp 23 (2014): 217-222.

[6] Broenink, Jan F., and Martin LJ Tiernego. "Peg-in-hole assembly using impedance control with a 6 dof robot." (1996).

[7] Hamner, Brad, et al. "An autonomous mobile manipulator for assembly tasks." Autonomous Robots 28.1 (2010): 131-149.

[8] Li, Tom LH, Antoni B. Chan, and A. Chun. "Automatic musical pattern feature extraction using convolutional neural network." Proc. Int. Conf. Data Mining and Applications. 2010.

[9] Schluter, Jan, and Sebastian Bock. "Improved musical onset detection with convolutional neural networks." Acoustics, speech and signal processing (icassp), 2014 IEEE international conference on. IEEE, 2014.

[10] Lee, Honglak, et al. "Unsupervised feature learning for audio classification using convolutional deep belief networks." Advances in neural information processing systems. 2009.

[11] Abdel-Hamid, Ossama, et al. "Convolutional neural networks for speech recognition." IEEE/ACM Transactions on audio, speech, and language processing 22.10 (2014): 1533-1545.

[12] Rad, Nastaran Mohammadian, et al. "Convolutional neural network for stereotypical motor movement detection in autism." arXiv preprint arXiv:1511.01865 (2015).

[13] Hikosaka, Okihide, et al. "Central mechanisms of motor skill learning." Current opinion in neurobiology 12.2 (2002): 217-222.

[14] Imamizu, Hiroshi, et al. "Human cerebellar activity reflecting an acquired internal model of a new tool." Nature 403.6766 (2000): 192-195.

[15] Doya, Kenji. "Complementary roles of basal ganglia and cerebellum in learning and motor control." Current opinion in neurobiology 10.6 (2000): 732-739.

[16] Scherer, Dominik, Andreas Mller, and Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition." International Conference on Artificial Neural Networks. Springer Berlin Heidelberg, 2010.

[17] Myers, Donald R., Michael J. Pritchard, and Mark DJ Brown. "Automated programming of an industrial robot through teach-by showing." Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on. Vol. 4. IEEE, 2001.

[18] Kingma, Diederik, and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

[19] Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." Journal of Machine Learning Research 15.1 (2014): 1929-1958.

[20] Roberge, Jean-Philippe, et al. "Unsupervised feature learning for classifying dynamic tactile events using sparse coding." Robotics and Automation (ICRA), 2016 IEEE International Conference on. IEEE, 2016.

[21] Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[22] Uijlings, Jasper RR, et al. "Selective search for object recognition." International journal of computer vision 104.2 (2013): 154-171.

[23] Alexe, Bogdan, Thomas Deselaers, and Vittorio Ferrari. "Measuring the objectness of image windows." IEEE transactions on pattern analysis and machine intelligence 34.11 (2012): 2189-2202.

[24] Graves, Alex, Abdel-rahman Mohamed, and Geoffrey Hinton. "Speech recognition with deep recurrent neural networks." Acoustics, speech and signal processing (icassp), 2013 IEEE international conference on. IEEE, 2013.