Robot Task Primitive Segmentation from Demonstrations using only Built-in Kinematic State and Force-Torque Sensor Data

Simon Lyck Bjært Sørensen, Thiusius Rajeeth Savarimuthu and Iñigo Iturrate

***Abstract*— This paper presents a method for segmentation and classification of kinesthetic demonstrations of robot peg-in-hole tasks using a deep neural network. The presented method depends only on sensor data that is readily available on collaborative robots (kinematic state and force-torque readings) and does not need any additional sensors. Our method can be used to automatically derive program structures from a single demonstration of a robot task. This can reduce programming time, and make it easier to revise sections of a larger task.**

We introduce a combined architecture consisting of a Convolutional Neural Network block for raw feature extraction and a Long Short-Term Memory block for tracking the time-evolution of these features. We also extend the model from binary insertion segmentation to multi-class segmentation including *pick up*, *place*, *free air motions*, *insertions*, and *extraction*. Through an ablation study and comparison to previous work, we show that the new model performs better on the test set, containing one demonstrator, but significantly better on a generalization set consisting of ten previously unseen demonstrators, reaching an overall accuracy of 70%, which is 15 percentage points better than the closest-performing other method.

I. INTRODUCTION

The rise of collaborative robots in recent years can partly be attributed to their ease of programming compared to traditional industrial robots. For instance, movement via-points can be programmed by kinesthetically moving the robot to the desired point. However, for more complex motions and tasks, programming is still time-consuming. Here, Programming by Demonstration (PbD) can be used, as it allows users to simply demonstrate how a task is performed, after which the robot will learn a program that imitates the behavior [1]. PbD can be formulated at various levels of abstraction [2], depending on whether the focus is on learning motor control policies [3], [4], or semantic segmentations that describe a task symbolically [5], [6]. This paper focuses on the second.

We present a deep neural architecture that, given only the input data streams available on a typical industrial collaborative robot, i.e. kinematic state (position and velocity) and force-torque (FT) sensor data, without any additional sensors such as computer vision and/or hardware signals, is capable of segmenting a demonstration of the task and classifying these segments into one of five action categories. This is targeted at quick (re-)programming of collaborative industrial robots, where the output of the proposed system can generate an overall program structure that can be pre-initialized with different action primitives, but that is also easily understandable and editable by the end-user.

All authors are with the Maersk Mc-Kinney Moller Institute, University of Southern Denmark, Odense, Denmark {siso,trs,inju}@mmmi.sdu.dk

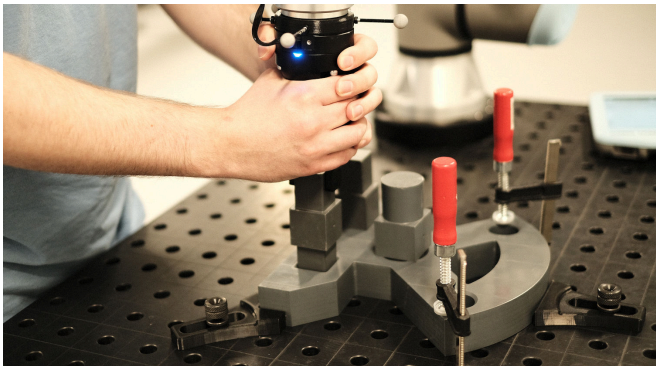


Fig. 1: Kinesthetic demonstration of an insertion using the Cranfield assembly benchmark.

A challenge for PbD at the semantic level is the complexity of the tasks, particularly when it comes to generalization to unseen circumstances. Many approaches address this by incorporating additional sensor modalities. Learning from images and/or video [7] is widespread, with methods like behavioral cloning from observation gaining interest in recent years [8], [9]. Adding computer vision makes it possible to adopt an object-centric representation [10], [6], [5], well-suited to capturing the semantics of a task that, by definition, is in itself object-centric. Other methods combine images with other multi-modal input data, allowing them to learn latent features that allow to discover skills automatically, i.e., without the need for annotations [11]. However, it is often not financially feasible to use computer vision or multi-modal sensory input in industrial applications, as it may require qualified personnel and expensive hardware for its setup, and often depends on a well-controlled environment.

Given the above, we identify a gap in the literature for methods that address this problem with limited low-dimensional input data. We constrain our problem to using only robot kinematic states and FT sensor data as input modalities, as they are widely available on most collaborative robots. As these modalities consists of continuous time-indexed data points, our problem is closer to time-series segmentation and classification [12].

Time-series classification is a widely-studied problem, with applications in many fields. While it has traditionally been approached algorithmically [13], [14], [12], recently, deep learning has made significant strides, as it circumvents the design of descriptive hand-crafted features and instead learns these in an unsupervised fashion [15], [16], [17]. Transformers [18] have shown great performance, but their

data-hungry nature creates a need for pre-training on large datasets, which are not available for our target domain.

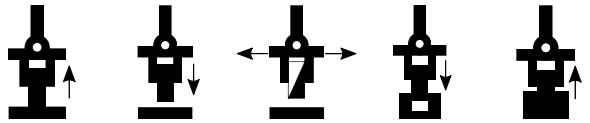
Within robotics, time-series analysis has been applied to, e.g., human motion prediction for human-robot cooperation [19], segmentation of robot action data from one-dimensional accelerometer and camera data [20], or classification of human-robot contact situations into intended interactions or unintended collisions [21]. Machine learning approaches are also common. For instance, Moreira et al. detect threading task failure from FT signals using Convolutional Neural Network (CNN) [22]. Eiband and Dongheui [23] tackle a problem similar to ours: classification of soft-contact skills based on robot position, force-torque and gripper opening and status signals. They incorporate tool-specific measurements (from a gripper) and use hand-designed features that are limited in their applicability. Furthermore, the ability of their classifier to generalize beyond the single demonstrator it was trained on is not evaluated. Other approaches have made use of unsupervised learning [24], [25]; however, the output of such methods will not necessarily generate classes that correspond to how a human would segment the task, and are therefore not user-interpretable.

The closest approaches to this work are those of Roberge and Duchaine [26] and our own previous work [27], as they consider the same kinesthetic teaching scenario and input data modalities as we do. Instead of the binary insertion segmentation considered in those papers, we extend our problem to consider five classes of task primitives consisting of insertion, extraction, pick up, place, and free air (see Figure 1). To achieve this, we introduce a combined network architecture by including both CNN and Long Short-Term Memory (LSTM) components, the first of which learns to extract data features, while the second considers their evolution across time. Such a combined architecture has previously shown promising results within the context of action recognition from mixed video and inertial input signals [28].

The remainder of this paper is organized as follows: Section II will formulate the problem. Section III will present our methodology for data collection and model training. Section IV will detail how our model is built up. First, we will present considerations on data pre-processing and then, we will compare our proposed model to two ablations and to our previous work [27] on a binary insertion classification and segmentation problem. We will evaluate our model against the baselines on the target five-class classification problem in Section V, and discuss and interpret the results in Section VI. Finally, we will conclude the article in Section VII.

II. PROBLEM FORMULATION

The problem we consider is that of, given an input demonstration consisting of n samples, where the i-th sample consists of: $\{\dot{\mathbf{p}}_i, \mathcal{F}_i\}$, where $\dot{\mathbf{p}}_i \in \mathbb{R}^3$ is the velocity of the robot end-effector in base frame, and $\mathcal{F}_i \in \mathbb{R}^6$ is a wrench vector containing the forces and torques at the end-effector, for each sample i, assigning a corresponding class label C_i. These class labels are drawn from a set of five different task primitives, as seen in Figure 2.



(a) Pick up (b) Place (c) Free air (d) Insert (e) Extract

Fig. 2: Illustrations of the different Tasks Primitives (TP) that our method can segment a robot task demonstration into.

A. Choice of Class Labels

Our class labels correspond, at a high semantic level, to five primitive actions present in a typical peg-in-hole task: picking up a workpiece (Figure 2a), setting down or placing the workpiece (Figure 2b), performing a free air motion without a workpiece or a transfer motion with a workpiece (Figure 2c), inserting a workpiece into another (Figure 2d), or extracting a workpiece from another (Figure 2e). Note that, when we refer to "binary insertion segmentation", we are referring simply to the problem of classifying samples into either the *Insert* class (Figure 2d) or any of the other classes (without specifying which).

Contrary to other approaches using unsupervised learning [24], [25], we pre-define the classes instead of letting the model autonomously discover them for two reasons: 1) The output of the model must fit with the types of actions programmable on an industrial robot, 2) Explainability of the output contributes to the user-friendliness of the system.

B. Coupling between Task, Demonstrator and Inertial Forces

An implicit issue that makes this problem challenging is the coupling between the dynamics of the demonstrator and of the task in the demonstration data [26][27]. Kinesthetic teaching often makes use of indirect force control, e.g., admittance control [29], where the demonstrator applies forces to the end-effector to move the robot around. This requires that the user grasp the robot beneath the FT sensor, resulting in the sensor measuring forces due to both the user and to the environment (i.e. task). Furthermore, as the learning from demonstration system should make it possible to interact with multiple (potentially unknown) objects, and the user might not be able to manually specify their mass, it is not possible to compensate for the inertial loads that they will induce, so this will also be captured by the FT sensor.

This issue makes it complicated, if not unfeasible, to identify features from the raw data, especially if these are to generalize well to other ways of demonstrating the task(s). This motivates the use of Machine Learning techniques, as they are able to learn features in a latent space. Similar to [26], [27], we adopt a deep neural network as a solution.

C. Practical Applicability of our Methods

Note that, while a solution to the above problem of coupled forces is to simply add a second sensor modality (e.g. sensitive skin at the grasping point) to decouple demonstrator and task forces, this goes against our main philosophy of making the method usable with any generic industrial collaborative robot without additional hardware. This argument also applies to tracking the objects' positions

[24] in order to measure relative distances between objects, as this would require a tracking or computer vision system.

An important point to make regarding how we perceive the applicability of the proposed method in an industrial setting is that we do not intend for the model to be re-trained on site. Instead, it would be pre-trained on a selection of people and tasks and deployed in a frozen state. For this reason, it is important that the method generalizes well from a constrained set of training data to unseen generalization conditions, for example, from a single demonstrator in the training set to multiple unseen demonstrators.

III. METHODOLOGY

In this section, we will describe our methodology for data collection and how all of the models described in the remainder of the paper were trained.

A. DATA COLLECTION

As input for the model, the robot Tool Center Point (TCP) forces, torques, and velocities are used. These were chosen as they are available in most industrial robots and remove the need for additional sensors. Forces and torques contain significant information on the task contact state and dynamics. They will, however, be impacted by the coupling between operator and environment dynamics while performing the demonstrations. Velocities were chosen instead of positional data to make the data features position-invariant. In addition to the input data for the model, video recordings of the demonstrations were collected purely to aid in the annotation process.

The demonstrations were performed using four parts from the Cranfield assembly benchmark [30], two square pegs, one cylindrical peg and a backplate (shown in Figure 1). The demonstrator chooses the Tasks Primitives (TP) randomly to minimize the chance of the model learning the order of the performed TPs. For each placement of the pegs the operator chose a random position on the table. All data was collected using a Universal Robots UR5e collaborative robot.

For the training dataset, we collected data from 20 sessions, totaling around 1 hour of demonstrations, all performed by the same person. As there can be a wide range of variation in the way that individuals demonstrate a task kinesthetically, we decided to collect a second generalization dataset consisting of ten additional people, each providing approximately five minutes of additional demonstrations. Note that, while including these other demonstrators in the training and test dataset would likely improve model performance, we were particularly interested in evaluating the models' performance on data they were not trained on, as this better reflects the conditions we would expect the system to be used in (see Section II-C).

All data was annotated using the VGG Image Annotator (VIA) tool [31], and split with a 60/20/20 ratio into training, validation and test sets, ensuring that data from specific sessions was kept within the same set. As each session was approximately the same length, the split ratio also approximately holds true for the distribution of TPs.

The collected dataset **KI**nesthetic **D**emonstrations of **A**ssembly **T**asks (KIDAT) is published at [32]. An overview of the annotated data – total annotated time, occurrence of the different classes in the annotated data, etc. – can also be found there.

B. TRAINING

All models were trained over 800 epochs using the Adam optimizer with a starting learning rate (lr) of 1e-3. For each epoch, 2500 batches of 32 samples were trained on and 1250 batches were used for validation. The lr was scaled by 0.1 if the validation loss did not improve within 200 epochs. If no improvement was achieved within 500 epochs the model would be terminated. Categorical cross-entropy was used as a loss function. The models were trained ten times on random weight initialization seeds to minimize the effect of random initialization on the results. While training, the data given to the models was balanced by dropping samples of over-represented classes. To implement this the TensorFlow Dataset API was used with the transformation *rejection_resample* with a target distribution of $1/N_c$, with N_c being the number of classes.

IV. PROPOSED MODEL

We will now present our main contribution in terms of a new model architecture for task primitive classification and segmentation from robot peg-in-hole task demonstrations. First, we will describe our considerations in terms of data pre-processing, and evaluate different normalization and clipping pipelines. Next, we will present our proposed model itself. As our architecture consists of both a CNN and an LSTM, we will justify their combination based on an ablation study and on a comparison against our previous approach [27] for binary insertion segmentation.

A. DATA PRE-PROCESSING

To remove bias in the sensor, the force-torque data was zeroed with the first sample in each session (as each demonstration always starts with the robot out of contact). A key consideration in our data pre-processing is that, whereas, e.g., image-based data (where deep learning methods have achieved the greatest success) is bounded in the range of $[0, 255]$, our force-torque and robot kinematic data is continuous and of large dynamic range. While some of the action primitives may consist of force patterns of at most a couple of Newtons, where the distinguishing features could be components under 1 N, others may be characterized by transients that can reach 50 - 150 N, such as during contact. For this reason, we explored different normalization schemes. We considered no normalization as an option, as we hypothesized that normalization might compress the small components that are in part key to the classification. Similarly, we considered clipping, as we hypothesized that for the higher transients the magnitude of the peak might not make a significant difference for classification beyond a certain threshold, particularly as these contact forces will greatly depend on the person performing the demonstration.

TABLE I: Overview of normalization methods. Validation accuracy (acc) given as mean over 10 runs. The normalization was performed to [-1, 1]. The results are presented in the form of: $\mu \pm \sigma$ $(\min - \max)$

Method	Clipping			Normalization Range			Validation dataset	
	Force	Torques	Velocity	Force	Torques	Velocity	Accuracy	$F1_{ma}$
v0	Off						0.923±0.011 (0.90-0.93)	0.883±0.014 (0.85-0.90)
v1	[-50, 50]	[-5, 5]	[-0.5, 0.5]	[-50, 50]	[-5, 5]	[-0.5, 0.5]	0.909±0.021 (0.87-0.94)	0.869±0.024 (0.82-0.91)
v2	[-75, 75]	[-7.5, 7.5]	[-0.5, 0.5]	[-75, 75]	[-7.5, 7.5]	[-0.5, 0.5]	0.918±0.013 (0.89-0.94)	0.878±0.016 (0.85-0.91)
v3	[-100, 100]	[-10, 10]	[-0.5, 0.5]	[-100, 100]	[-10, 10]	[-0.5, 0.5]	0.913±0.027 (0.84-0.94)	0.871±0.031 (0.79-0.90)
v4	Off			[-50, 50]	[-5, 5]	[-0.5, 0.5]	**0.927±0.022 (0.86-0.94)**	**0.889±0.026 (0.82-0.91)**

In all, we explored five variants of normalization and clipping; see Table I: no normalization (v0), three different normalization and clipping ranges (v1-v3), and normalization without clipping (v4). No normalization and normalization without clipping showed approximately the same validation accuracy and F1 macro average ($F1_{ma}$) score, but the best run for normalization without clipping had a validation accuracy 1 percent point higher and an $F1_{ma}$ score 0.01 higher than the best run with no normalization. Therefore, normalization without clipping was used.

The resulting normalization procedure divides forces by a factor of 50 (i.e. 50 N $\mapsto$ 1), torques by 5 (i.e. 5 Nm $\mapsto$ 1) and velocities by 0.5 (i.e. 0.5 m/s $\mapsto$ 1).

B. MODEL ARCHITECTURE

Inspired by approaches from the time-series analysis community, we hypothesized that a combined architecture with both CNN and LSTM components might outperform these two separate models alone, as the CNN would find features, while the LSTM would react to how these features change over time. This could both help the model identify more subtle changes among classes (rendering better at multi-class problems), as well as make it more robust to generalizations not present in the training set, such as different demonstrators performing the various TPs at different speeds.

As an input to the model, data is first windowed with a sliding window that moves one sample per step. Each window is subsequently divided into five sub-windows with a 10% overlap. The architecture consists of a larger model with an inner submodel that is time-distributed over the sub-windows. The submodel is made up of four convolutional blocks, each with two convolutional layers, where each of these is followed by a batch normalization and a leaky ReLU. A max-pooling layer follows the output of each block. After the convolutional blocks in the submodel there is a dropout layer with 50% dropout. At the end of the submodel there is an LSTM layer with 10 nodes. The features from the time-distributed submodel are finally given to an LSTM layer with 64 nodes, followed by a Fully Connected (FC) classifier with a softmax activation function. A diagram of the model architecture are shown in Figure 3.

Through hyperparameter tuning, we found that reducing the sampling rate to 62 Hz and window to 5 s – resulting in 310 samples – improved the performance of the insertion segmentation. These parameters were subsequently adopted for the rest of our models. In contrast, Iturrate et al. [27] used a sample rate of 125 Hz with a window of 8 s. As the sampling rate and window size can be data-specific, for comparison, all other baseline models were also trained using the 62 Hz and a 5 s window.

Table II shows the results of an ablation test on our proposed network, comparing the full model against its individual CNN and LSTM components and against prior work [27], which we adopt as a baseline. The p-values of a T-test comparing the models are shown in Table III. As shown from these results, our proposed model shows a statistically significant improvement over the baselines ($p < 0.01$) for both accuracy and $F1_{ma}$ on the test set. On the generalization set, the improvement is statistically significant ($p < 0.02$) for $F1_{ma}$ against all baselines, and for accuracy against the two model ablations (CNN- and LSTM-only).

Figure 4 shows confusion matrices for our proposed architecture compared to that of our previous work [27] on the generalization dataset. We can observe an improvement in accuracy of around 8 percentage points (pp) for the *No insertion* class, while the *Insertion* class maintains similar performance. This shows that our proposed model is less likely to output false negatives, i.e., it is better at detecting anything other than insertion, which supports the notion that

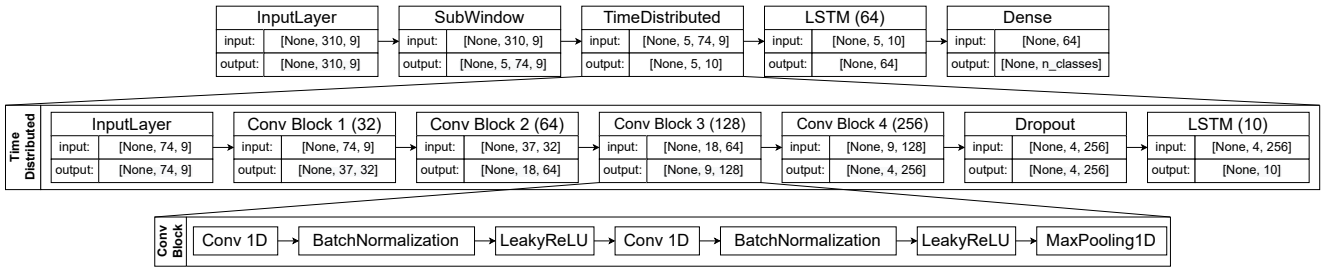
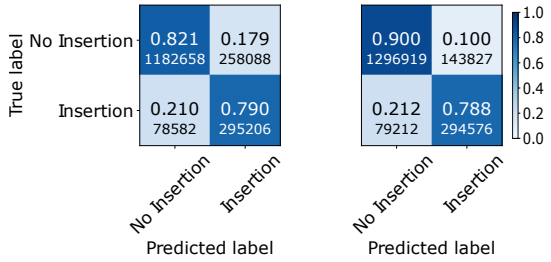


Fig. 3: Mixed model architecture. Two output nodes (n_classes) are used for Insertion and five for the multi-class problem.

it might better be suited to multi-class classification, as will be explored in the next section.



(a) Iturrate et al. [27] (b) Insertion Mixed (ours)

Fig. 4: Confusion Matrix for Iturrate et al. and our architecture on Insertion segmentation on the generalization dataset. Model with the highest validation accuracy of the ten runs are used.

TABLE II: Results for the Insertion problem. Statistics are based on ten runs for each model. The results are presented in the form of: $\mu \pm \sigma$ $(\min - \max)$.

Model	Accuracy	$F1_{ma}$
Test dataset		
Iturrate et al. [27]	0.951±0.018 (0.92-0.97)	0.914±0.050 (0.77-0.95)
CNN-Only	0.949±0.021 (0.90-0.97)	0.907±0.058 (0.77-0.95)
LSTM-Only	0.927±0.029 (0.87-0.96)	0.859±0.072 (0.74-0.94)
Mixed (ours)	**0.968±0.005 (0.96-0.98)**	**0.948±0.008 (0.94-0.97)**
Generalization dataset		
Iturrate et al. [27]	0.799±0.037 (0.73-0.85)	0.746±0.029 (0.69-0.79)
CNN-Only	0.792±0.066 (0.64-0.86)	0.740±0.053 (0.62-0.79)
LSTM-Only	0.737±0.065 (0.57-0.81)	0.687±0.053 (0.55-0.75)
Mixed (ours)	**0.817±0.063 (0.69-0.88)**	**0.774±0.058 (0.66-0.83)**

TABLE III: T-test p-values based on the each models 10 runs results on each session for the Insertion problem.

Compared Models		Metric	Test dataset (p-value)	Generalization dataset (p-value)
CNN-Only	LSTM-Only	Accuracy	0.0366	$2.76 \cdot 10^{-05}$
		$F1_{ma}$	0.0534	$1.11 \cdot 10^{-05}$
Mixed (ours)	CNN-Only	Accuracy	0.0069	0.0390
		$F1_{ma}$	0.0090	0.0036
	LSTM-Only	Accuracy	$2.56 \cdot 10^{-05}$	$1.94 \cdot 10^{-09}$
		$F1_{ma}$	$1.68 \cdot 10^{-05}$	$2.75 \cdot 10^{-12}$
	Iturrate et al. [27]	Accuracy	0.0048	0.1007
		$F1_{ma}$	0.0064	0.0112

V. RESULTS

This section will show the results of our proposed model on the target five-class problem. As we are not aware of a comparable baseline model for multi-class classification applicable to our problem domain restrictions (see Section II), we modified the model used in our previous work [27] to output multi-class labels, and compared against this as a baseline. Additionally, to justify our choice of architecture, we compared the proposed model against its two ablations.

Tables IV and V show the results of this evaluation. The accuracy and $F1_{ma}$ values for the four different models are shown on both the test and generalization datasets are shown in Table IV, while Table V shows p-values for a T-test comparing our proposed model against the other three candidates. The results show a statistically significant difference ($p < 0.01$) between our model and all other candidates for accuracy and $F1_{ma}$ on the test dataset, with our model outperforming the rest. For the generalization dataset, this difference is highly statistically significant ($p \ll 0.001$). The proposed architecture achieves an average accuracy of 77.7% (best run 82%) on the test set, which is on average five percentage points higher than the closest performing model. As hypothesized, the performance gap is widened on the generalization set, where our model achieves an average accuracy of 62.5% (best run 70%), which is on average 12 percentage points higher than the closest performing model, and 15 if the best runs are compared.

TABLE IV: Results for the multi class problem. Statistics are based on ten runs for each model. The results are presented in the form of: $\mu \pm \sigma$ $(\min - \max)$.

Model	Accuracy	$F1_{ma}$
Test dataset		
Iturrate et al. [27]	0.725±0.026 (0.68-0.77)	0.686±0.029 (0.63-0.72)
CNN-Only	0.705±0.043 (0.60-0.77)	0.649±0.081 (0.46-0.75)
LSTM-Only	0.657±0.079 (0.48-0.76)	0.613±0.091 (0.43-0.74)
Mixed (ours)	**0.777±0.032 (0.70-0.82)**	**0.769±0.037 (0.68-0.82)**
Generalization dataset		
Iturrate et al. [27]	0.501±0.063 (0.34-0.55)	0.437±0.067 (0.26-0.49)
CNN-Only	0.498±0.030 (0.44-0.54)	0.412±0.030 (0.36-0.46)
LSTM-Only	0.396±0.062 (0.31-0.52)	0.323±0.065 (0.24-0.47)
Mixed (ours)	**0.625±0.065 (0.48-0.70)**	**0.604±0.074 (0.46-0.69)**

TABLE V: T-test p-values based on the each models 10 runs results on each session for the Multi-class problem.

Compared Models		Metric	Test dataset (p-value)	Generalization dataset (p-value)
Mixed (ours)	CNN-Only	Accuracy	0.0004	$3.04 \cdot 10^{-27*}$
		$F1_{ma}$	$5.91 \cdot 10^{-05}$	$3.94 \cdot 10^{-40*}$
	LSTM-Only	Accuracy	$4.67 \cdot 10^{-06}$	$1.29 \cdot 10^{-49*}$
		$F1_{ma}$	$8.04 \cdot 10^{-06}$	$4.61 \cdot 10^{-56*}$
	Iturrate et al. [27]	Accuracy	0.0001	$1.17 \cdot 10^{-21*}$
		$F1_{ma}$	0.0031	$7.80 \cdot 10^{-28*}$

* These numbers are below machine precision, and should therefore be interpreted as $p \ll 0.001$.

Figure 5 shows the Confusion Matrix (CM) for the proposed architecture on the generalization dataset. The proposed model performs best on the *Pick-up* and *Insertion* classes with accuracies $\approx 84\%$. The worst performing class is *Free air*, with an accuracy of $\approx 60\%$.

The match plot in Figure 6 shows how well the predictions overlap with the ground truth annotations for a particular test session. The errors present here are in general representative of the type of errors present for most typical sessions.

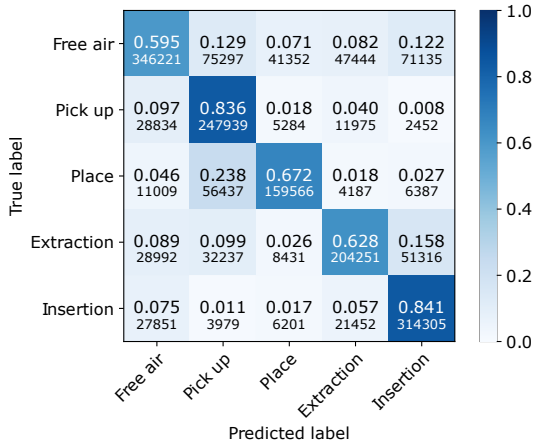


Fig. 5: Confusion Matrix for our architecture on all TP segmentation on the generalization dataset. Model with the highest validation accuracy of the ten runs are used.

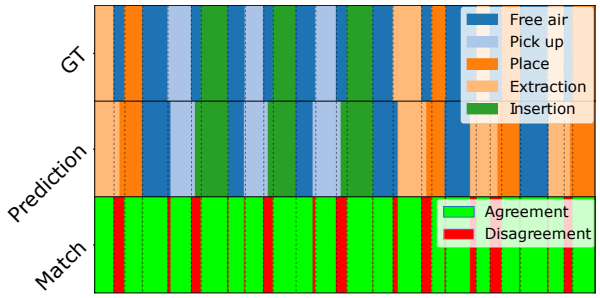


Fig. 6: Match plot showing the prediction of the Multi-Class model on an test session.

VI. DISCUSSION

A. Confusion between Classes

As can be seen from Figures 5 and 6, the largest source of confusion for the model are timing errors where the model disagrees with the ground truth on when a TP starts and stops. These are present in the first row and column of the CM. If the movement is demonstrated quickly, the *Free air* TP is often not predicted at all. Note that the timing of when a TP starts and stops can also differ between annotators.

Other sources of confusion are similarities between TPs. *Place* and *Pick up* are confused, as the only difference is the presence of an object in the gripper at the start of the TP. As our model does not receive gripper signals (to keep hardware agnosticism), the force from the weight of the peg could be a potential feature, but it is likely obscured by the coupling with the demonstrator's forces, as the mass of the pegs is low (194 to 277 g). We hypothesize that the model may be using the transversal forces generated by the gripper fingers closing around the peg to differentiate between the two TPs. A similar confusion is seen with *Extraction* and *Insertion*, which can mainly be distinguished based on the direction of the movement. As a kinesthetic demonstration of an extraction/insertion may not be a single fluid motion but multiple back-and-forth motions due to the tight tolerances, the clear sense of direction that differentiates *Insertion* and *Extraction* disappears for some demonstrators. Here, *Insertion* outperforms *Extraction*, probably because many *Insertions* will collide with the backplate before the demonstrator manages to align the peg with the hole. *Extraction* is also confused with *Pick up*. This could be due to the similarity of the movements, where the only difference are sideways forces for *Extraction*, which may become small if the *Extraction* is performed as a single smooth movement.

B. Generalization Performance and Model Architecture

Our results show that our model is more robust to conditions unseen in the training set, particularly to different demonstrators, compared to the baselines. We argue that this is highly relevant in the learning from demonstration community for two reasons: First, there is a large variation in quality, precision and speed between demonstrators. Second, collecting large training sets of demonstrations that capture a variance in the task conditions and methods of demonstrating comparable to real-world applications is highly time-consuming and costly, and in many situations infeasible.

Initially, we motivated our choice of architecture by arguing that the CNN component would focus on finding the features, while the LSTM component would learn to react to their evolution over time, rendering the network more robust to variances across demonstrators. From Tables IV and V, we see that our proposed model performs better than the baselines on the generalization dataset for the target five-class problem; this difference is highly statistically significant. We therefore believe that such a combined CNN+LSTM model holds high promise within this domain. While the overall accuracy of the best performing iteration of our model is 70%, which could be considered low in the wider deep learning community, we would like to draw attention to the complexity of the problem due to the limitations on the inputs and training set. In fact, we argue that, in practice, this performance might be sufficient if the target application is to quickly populate a robot program structure, as the user could correct the model in case of incorrect predictions.

C. Future Improvements

Following from the above, we envision using an adaptation of continual active learning [33] in order to on-the-fly tailor the model to new demonstrators. Similarly, user corrections to the robot program (i.e. corrections to incorrectly classified primitives) could be used as feedback to the model.

A topic that is key to our particular problem, but we argue is of high value to the wider robot learning community is how to encode time-invariant representations of robot skills. While Recurrent Neural Networks (RNN) such as LSTMs have been shown to be able to do this – and this is the approach we adopted – traditionally, this has been accomplished with sequence alignment methods, such as Dynamic Time Warping for alignment [34] or DTW Barycentric Averaging (DBA) for temporal averaging [35]. In future work, we would like to investigate if such methods could be integrated within our network architecture.

VII. CONCLUSIONS

We have proposed a method for segmentation and classification of kinesthetic robot demonstrations into five Task Primitives. Contrary to the majority of the State of the Art (SOA), the model requires only robot velocities and force-torque measurements at the end-effector as input, both of which are readily available on most collaborative robots. Our model utilizes a CNN for feature extraction, combined with an LSTM to track the time-evolution of these features.

Through comparison against its ablations and an existing baseline, we showed that our model performs better (5 percentage points improvement in accuracy with respect to the closest-performing competitor) on the test set, and significantly better (15 percentage points improvement) on demonstrators that were not part of the training. Thus, our work takes steps towards overcoming demonstrator variance, a significant problem in the learning from demonstration community.

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[2] S. Chernova and A. L. Thomaz, "Robot learning from human teachers," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 8, no. 3, pp. 1–121, 2014.

[3] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[4] S. Calinon, F. Guenter, and A. Billard, "On learning, representing, and generalizing a task in a humanoid robot," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 2, pp. 286–298, 2007.

[5] K. Ikeuchi and T. Suehiro, "Toward an assembly plan from observation. i. task recognition with polyhedral objects," *IEEE transactions on robotics and automation*, vol. 10, no. 3, pp. 368–385, 1994.

[6] T. R. Savarimuthu, A. G. Buch, C. Schlette, N. Wantia, J. Roßmann, D. Martínez, G. Alenyà, C. Torras, A. Ude, B. Nemec, *et al.*, "Teaching a robot the semantics of assembly tasks," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 48, no. 5, pp. 670–692, 2017.

[7] P. Sun, Z. Yang, T. Zhang, S. Guo, and F. Chen, "Primitive-contrastive network: data-efficient self-supervised learning from robot demonstration videos," *Applied Intelligence*, pp. 1–16, 2021.

[8] F. Torabi, G. Warnell, and P. Stone, "Behavioral cloning from observation," *arXiv preprint arXiv:1805.01954*, 2018.

[9] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson, "Implicit behavioral cloning," in *Conference on Robot Learning*, pp. 158–168, PMLR, 2022.

[10] C. Devin, P. Abbeel, T. Darrell, and S. Levine, "Deep object-centric representations for generalizable robot learning," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7111–7118, 2018.

[11] Y. Zhu, P. Stone, and Y. Zhu, "Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4126–4133, 2022.

[12] A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data mining and knowledge discovery*, vol. 31, no. 3, pp. 606–660, 2017.

[13] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Proc. IEEE international conference on data mining*, pp. 289–296, 2001.

[14] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "Segmenting time series: A survey and novel approach," in *Data mining in time series databases*, pp. 1–21, World Scientific, 2004.

[15] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[16] J. C. B. Gamboa, "Deep learning for time-series analysis," *arXiv preprint arXiv:1701.01887*, 2017.

[17] M. Längkvist, L. Karlsson, and A. Loutfi, "A review of unsupervised feature learning and deep learning for time-series modeling," *Pattern Recognition Letters*, vol. 42, pp. 11–24, 2014.

[18] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, 2020.

[19] C. Pérez-D'Arpino and J. A. Shah, "Fast target prediction of human reaching motion for cooperative human-robot manipulation tasks using time series classification," in *IEEE international conference on robotics and automation (ICRA)*, pp. 6175–6182, 2015.

[20] S. Lenser and M. Veloso, "Non-parametric time series classification," in *Proc. of the IEEE International Conference on Robotics and Automation*, pp. 3918–3923, 2005.

[21] G. Cioffi, S. Klose, and A. Wahrburg, "Data-efficient online classification of human-robot contact situations," in *European Control Conference (ECC)*, pp. 608–614, 2020.

[22] G. R. Moreira, G. J. Lahr, T. Boaventura, J. O. Savazzi, and G. A. Caurin, "Online prediction of threading task failure using convolutional neural networks," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2056–2061, 2018.

[23] T. Eiband and D. Lee, "Identification of common force-based robot skills from the human and robot perspective," in *2020 IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, pp. 507–513, 2021.

[24] E. G. Herrero, J. Ho, and O. Khatib, "Understanding and segmenting human demonstrations into reusable compliant primitives," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9437–9444, IEEE, 2021.

[25] S. Thrun, J. Langford, and D. Fox, "Monte carlo hidden markov models: Learning non-parametric models of partially observable stochastic processes," in *ICML*, vol. 99, pp. 415–424, Citeseer, 1999.

[26] E. Roberge and V. Duchaine, "Detecting insertion tasks using convolutional neural networks during robot teaching-by-demonstration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3210–3216, 9 2017.

[27] I. Iturrate, E. Roberge, E. H. Ostergaard, V. Duchaine, and T. R. Savarimuthu, "Improving the Generalizability of Robot Assembly Tasks Learned from Demonstration via CNN-based Segmentation," in *IEEE International Conference on Automation Science and Engineering (CASE)*, vol. 2019-Augus, pp. 553–560, 8 2019.

[28] N. Dawar and N. Kehtarnavaz, "Action Detection and Recognition in Continuous Action Streams by Deep Learning-Based Sensing Fusion," *IEEE Sensors Journal*, vol. 18, pp. 9660–9668, 12 2018.

[29] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics: Modelling, Planning and Control*, ch. 8. London: Springer London, 2009.

[30] K. Collins, A. J. Palmer, and K. Rathmill, "The Development of a European Benchmark for the Comparison of Assembly Robot Programming Systems," in *Robot Technology and Applications* (K. Rathmill, P. MacConaill, P. O'Leary, and J. Browne, eds.), pp. 187–199, Berlin, Heidelberg: Springer Berlin Heidelberg, 1985.

[31] A. Dutta and A. Zisserman, "The VIA annotation software for images, audio and video," in *Proc. of ACM International Conference on Multimedia*, MM '19, (New York, NY, USA), ACM, 2019.

[32] "Kinesthetic Demonstrations of Assembly Tasks (KIDAT)." `https://gitlab.sdu.dk/sdurobotics/medical/KIDAT`, 2022.

[33] M. Perkonigg, J. Hofmanninger, and G. Langs, "Continual active learning for efficient adaptation of machine learning models to changing image acquisition," in *Information Processing in Medical Imaging* (A. Feragen, S. Sommer, J. Schnabel, and M. Nielsen, eds.), pp. 649–660, Springer International Publishing, 2021.

[34] H. Sakoe, "Dynamic-programming approach to continuous speech recognition," in *Proc. the International Congress of Acoustics, Budapest*, 1971.

[35] F. Petitjean, A. Ketterlin, and P. Gançarski, "A global averaging method for dynamic time warping, with applications to clustering," *Pattern recognition*, vol. 44, no. 3, pp. 678–693, 2011.