Sequential Prediction with Logic Constraints for Surgical Robotic Activity Recognition

Md Masudur Rahman[1], Richard M. Voyles[2], Juan Wachs[3], and Yexiang Xue[1]

***Abstract*— Many real-world time-sensitive and high-stake applications (e.g., surgical, rescue, and recovery robotics) exhibit sequential nature; thus, applying Recurrent Neural Network (RNN)-based sequential models is an attractive approach to detect robotic activity. One limitation of such approaches is data scarcity. As a result, limited training samples may lead to over-fitting, producing incorrect predictions during deployment. Nevertheless, abundant domain knowledge may still be available, which may help formulate logic constraints. In this paper, we propose a novel way to integrate domain knowledge into RNN-based sequential prediction. We build a Markov Logic Network (MLN)-based classifier that automatically learns constraint weights from data. We propose two methods to incorporate this MLN-based prediction: (i) PriorLayer, in which the values of the hidden layer of the RNN are combined with weights learned from logic constraints in an additional neural network layer, and (ii) Conflation, in which class probabilities from RNN predictions and constraint weights are combined based on the conflation of class probabilities. We evaluate robotic activity classification methods on a simulated OpenAI Gym environment and a real-world DESK dataset for surgical robotics. We observe that our proposed MLN-based approaches boost the performance of LSTM-based networks. In particular, MLN boosts the accuracy of LSTM from 71% to 84% on the Gym dataset and from 68% to 72% on the Taurus robot dataset. Furthermore, MLN (i.e., PriorLayer) shows regularization capability where it improves accuracy in initial LSTM training while avoiding over-fitting early, thus improves the final classification accuracy on unseen data. The code is available at `https://github.com/masud99r/prediction-with-logic-constraints`.**

I. Introduction

Reliable sequential prediction is crucial in developing successful robotic systems in real-world time-sensitive and high-stake applications, such as in surgical, rescue, and recovery robots. Recurrent Neural Networks (RNN) for sequential prediction is an attractive approach [1], [2], [3], [4], [5], [6], [7]. The major advantage of such approaches is that it does not require handcrafted rules and careful feature engineering. RNNs are often trained with multiple epochs, which go over training data in multiple iterations. For a large amount of data, this process might take a long time. However, it is desirable to achieve reasonable prediction results with limited training data and within a quick turn-around (fewer training epochs).

In addition to that, the effectiveness of the learned model heavily depends on the *quantity* and the *quality* of the training data. Lack of annotated instance and lack of quality might heavily hamper models performance. In this case, limited training data [8] in many real-world safety-critical applications (e.g., medical surgery) poses a challenge in applying sequential prediction models in these domains.

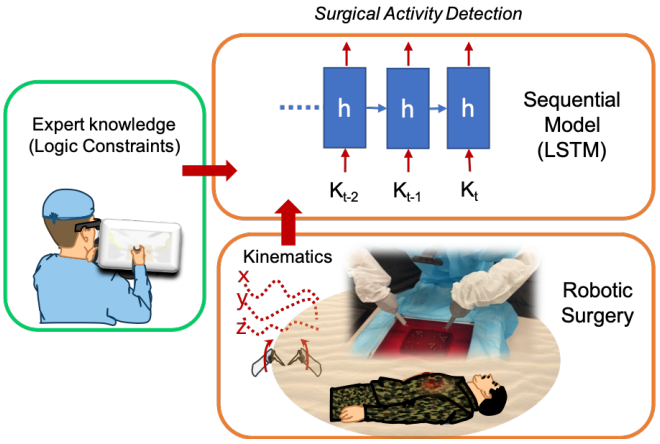


Fig. 1: Framework overview. The data from the austere robotic surgery environment is fed to the sequential model, which also takes input from domain experts in the form of logic constraints. Our approach is to leverage this information and make accurate predictions of the robotic surgical activity in limited data availability.

On the other hand, there is abundant domain knowledge in these real-world safety-critical applications [9], [10]. Human experts are usually trained for a long time in those domains. Consider medical surgery, for example, surgeons are extensively trained for a long time, and over time, they acquire essential knowledge about the critical tasks [11], [12]. Such domain knowledge can be expressed in the form of logic constraints. Unlike hard constraints, domain experts may introduce different importance values, which are often referred to as *soft constraints*. Such constraints/formulas can be expressed in the form of propositional logic. In many cases, assigning a proper weight to these formulae might be difficult for domain experts. In cases where the weights are not available, we propose to leverage Markov Logic Networks (MLN) [13] to learn weights associated with propositional logic formulae from data. Figure 1 shows an overview of our framework.

In this paper, we propose a novel way to integrate domain knowledge, represents as first-order logic (FOL), into RNN-based sequential prediction. We build a Markov Logic Network (MLN)-based classifier which learns FOL formula weights (soft-constraint) automatically from data. Then we

[1] Department of Computer Science, Purdue University, West Lafayette, IN, 47907, USA rahman64, yexiang@purdue.edu
[2]School of Engineering Technology, Purdue University, West Lafayette, IN, 47907, USA rvoyles@purdue.edu
[3]School of Industrial Engineering, Purdue University, West Lafayette, IN, 47907, USA jpwachs@purdue.edu

integrate knowledge into RNN using two methods: (i) PriorLayer, in which the values of the hidden layer of the RNN are combined with weights learned from logic constraints in an additional neural network layer, and (ii) Conflation, in which class probabilities from RNN predictions and constraints weights are combined based on conflation [14] of class probabilities.

In the first method, we introduce an additional linear layer in between hidden states and the final softmax layer. In this layer, we concatenate the constraints weights with learned hidden unit weights and allow the network to train this layer. Intuitively, this layer transfers the past learning (prior/domain/background knowledge, thus named PriorLayer) and history information (hidden state) into prediction.

In the second method, we propose to combine RNN class probabilities with constraints weights. We compute class probabilities using the learned constraints weights (MLN). Then we combine these class probabilities with the RNN generated probabilities with *conflation* of probability distributions [14]. Intuitively, the *conflation* is the distribution determined by the normalized product of the probability densities that minimizes Shannon Information's loss in incorporating the combined information.

We demonstrate the performance of our approaches on robotic activity classification on simulated data (Gym) as well as in real-world robotic surgery tasks (Taurus robot data) [15]. In particular, we focus on learning primitive surgical operations, the so-called surgemes [16], in the sequential demonstrations of basic laparoscopic surgical procedures. We observe that our proposed methods boost the RNN-based networks' performance, which we implemented as Long short-term memory (LSTM) [17].

In particular, MLN boosts the accuracy of LSTM from 71% to 84% on the Gym dataset and from 68% to 72% on the Taurus robot dataset. Furthermore, MLN (i.e., PriorLayer) shows regularization capability where it improves accuracy in initial LSTM training while avoiding over-fitting early, thus improves the final classification accuracy on unseen data. We also observe that MLN has a considerable impact on the LSTM prediction where the constraint weights (capturing domain knowledge) are learned properly.

II. Preliminaries

In this section we review machinery used in this paper for the logic constraints.

Markov Networks. A Markov network is a model for the joint distribution of a set of variables $X = (X_1, X_2, ..., X_n)$ [18] which is composed of an undirected graph G and a set of potential functions ϕ. For each variable, the graph has a node, and for each clique in the graph, the model has a potential no-negative real-valued function.

First-order Logic. A formula consists of literals connected by logical connectives (i.e., $\vee$ and $\wedge$). A first-order knowledge base (KB) is a set of sentences or formulas in first-order logic [19]. $A \wedge B \Rightarrow C$ is an example of an implication formula. The left side literal(s) can be called implication antecedents, and the literal on the right side can be called a conclusion. For the classification task, the conclusion usually consists of a class label. Note that many formulae may be typically true in the real world, but they might not always be true. It is a painstaking task to come up with formulae which is always true, and they often capture a fraction of relevant knowledge.

Markov Logic Network (MLN) is a set of first-order formulas and associated weights [13] where each formula represents some constraints. Unlike in pure first-order logic, a constraint may be violated without causing unsatisfiability of the entire system. We can consider a first-order knowledge base as a set of hard constraints on the set of possible worlds: if a world violates even one formula, its probability becomes zero. In contrast, the idea in MLNs is to soften these constraints: if a world violates a formula in the KB, it becomes less probable. A word become more probable if it violates fewer formulas. The probability is given by the associated weights of the formulas where the higher weights indicate strong constraints. These weights are typically learned using a supervised learning approach.

III. RNN for Sequential Prediction

Given a sequence of observations,

$$X_{1:T} = (X_1, X_2, ..., X_T) \tag{1}$$

an RNN factorizes its joint distribution according to the chain rule

$$P(X_{1:T}) = \Pi_{t=1}^{T} P(X_t | X_{1:t-1}) \tag{2}$$

To capture the contextual dependencies, RNN takes the conditional probability as a function of a low dimensional recurrent hidden state as follows,

$$P(X_t | X_{1:t-1}) = P(X_t | h_t) \tag{3}$$

$$h_t = f_W(h_{t-1}, x_t)) \tag{4}$$

where h_t is the current state which is updated by a function (f_W) of previous state h_{t-1} and current input x_t. Depending on the network variety (e.g., RNN, LSTM, GRU) this updating procedure (f_W) can be of various forms. For instance, in RNN we can update the hidden state as in equation 5.

$$h_t = tanh(W_{hh} h_{t-1} + W_{xh} x_t) \tag{5}$$

For a classification problem, the class weight is obtained using a linear transformation on the hidden state h_t as in equation 6.

$$h2c = W_l h_t \tag{6}$$

Then a $softmax$ is applied to generate the class probability (equation 7), and finally, the prediction is obtained by applying $argmax$ (equation 8).

$$P_{class} = softmax(h2c) \tag{7}$$

$$Pred_{class} = argmax(P_{class}) \tag{8}$$

All the states share the same weights W at every time step. Note that, we get a class prediction per input (x_t) and thus a sequence of length m would produce m predictions where in each prediction the network leverages previous context information of given input sequence which is carried over hidden states (h).

Sequential Prediction Task. We consider a sequential prediction task where a RNN network takes low-level data as input and predicts the classes ($C_{1:k}$) in each time-step. The input is a fixed length vector (X_i) where each element contains value of each feature. For a given sequence of observations ($X_{1:T}$) of T time-steps, RNN predict labels $y_{1:T}$ for each time-steps, where $y_i \in C_{1:k}$.

In this paper we consider the task of robotic activity or surgical procedure classification. In the surgical setting, a whole surgical procedure is breakdown into smaller steps which often called "surgeme" [16]. A list of surgemes and their visuals that we used in this paper are given in Figure 5, and 6 (details in experiment).

IV. Logic Constraints from Domain Knowledge

Models often work on the low-level data (sensor readings, robots kinematic data, etc.) which might pose challenges to incorporate domain knowledge. Firstly, the domain experts might find it difficult to express their knowledge in the form of low-level data. Secondly, we need a systematic way to integrate the knowledge into RNN prediction.

A. Map Low-level data to High-level

To address the first challenge, we allow domain experts to express their knowledge in a high-level language. We propose to express constraints in the form of propositional logic which can encode high-level human-understandable language into a logical form. For example, in our task (DESK data), "if two robot arms (left, right) getting closer toward each other and one arm (right) has a surgical tool, the activity is most likely an *exchange* operation". We can easily express this high-level sentence into a propositional logic as $D \wedge L \Rightarrow E$, where D is the binary variable which is *True* if two robot arms are getting closer and L is *True* when left arm holds a surgical tool. Note that, this constraint not necessarily true for all the cases and thus we call this as *soft constraints*. Formulating such constraints required the variables to be in the binary format that is not always the case for input features (X_i). Thus, we binarized all variables (discrete or continuous) with discretization. If the variable is discrete-valued we assign a new name for each value and thus create additional variables in binary format. On the other hand, if x_i is continuous-valued, we first discretize a continuous value with a threshold and then apply the above process to convert them to binary format.

B. Formulate Logic Constraints

The importance of the constraints might vary depending on the constraints type and the underline problem. Thus a natural way to quantify the importance is to assign a weight to each such constraint. If domain experts have sufficient knowledge about the problem, they can provide a weight along with formulating a constraint. In many cases, assigning such weight is very challenging. Thus, we propose to learn these weights from training data. We leverage the Markov Logic Network (MLN) to learn formula weights from the data.

We now discuss two types of constraints that we consider in this paper.

C1. *Static Input Feature Constraints*: Let's assume $X_i^t = (x_1, x_2, ...x_m)$ an input instance of length m at time-step t, where x_i is the value of feature i. Propositional logic expect each feature variable to be in binary format. For any arbitrary value variable x_i, we first convert it to binary format (details in previous section).

We can impose constraints of a single variable or combining multiple variables. In the case of classification, we formulate the constraints in the form of implication as $A \wedge B \Rightarrow C$, where A and B are two feature variables and C is the class label. Intuitively, the weight of this formula will say how likely the prediction would be class C given both A and B is true. Note that, there is no penalty of formulating a large number of constraints from domain knowledge as we can learn the weight from the training data.

C2. *Sequential Input Feature Constraints*: In sequential prediction task, input also comes in sequence and thus a feature might exhibit interesting properties over time-steps. To capture such properties and leverage in the prediction process we formulate this type of constraints. As an example, *gripper state* of the robotic arms might provide useful information about what the robot is doing. This feature value often ranges between two numbers, let say, 0 to 100 where a value of 0 means completely closed, 100 means completely *open*, and any value (d) in between indicates how much the gripper is *close/open*. Thus over the time if we see the value of d is decreasing we can assume that the robot is trying to grab the objects.

Formally, let's consider a feature variable whose value at time-step t is x_i^t and at $t-1$ is x_i^{t-1}. The difference of value between these two is calculated using equation 9.

$$\Delta_i^t = x_i^t - x_i^{t-1} \tag{9}$$

Depending on the value of Δ_i^t we generate three boolean variables for x_i: (i) x_i^{static} which is *True* if $\Delta_i^t = 0$; (ii) $x_i^{increase}$ which is *True* if $\Delta_i^t > 0$, and *False* otherwise; and (iii) $x_i^{decrease}$ which is *True* if $\Delta_i^t < 0$, and *False* otherwise. Note that, we can replace the zero (0) with a threshold value (usually very small) which will ignore very tiny changes. All the features are now augmented to *binary format* and we can formulate constraints as of MLN requirements (propositional logic).

C. Logic Learning Module (LLM)

An overview of the logic learning module is given in Figure 2. Here, we discuss the process of weights and class probability learning in details.

Learning MLN Formula Weights. The input to this process is the formulas and annotated data (sample from correspond-

ing domain). This process assigns a weight for each formula corresponding to each class label. The probability of the world x specified by a grounded Markov network is given by equation 10 [13].

$$P(X_{worlds} = x_{world}) = \frac{1}{Z} exp(\sum_i w_i n_i(x_{world})) \\ = \frac{1}{Z} \prod_i \phi_i(x_{world}^{\{i\}})^{n_i(x_{world})} \quad (10)$$

where $n_i(x_{world})$ is the number of true groundings of formula F_i in x_{world}, $x_{world}^{\{i\}}$ is the truth values (state) of the atoms appearing in F_i, and $\phi_i(x_{world}^{\{i\}}) = e^{w_i}$.

As an illustration, let's consider a variable gripper state (GS) with two possible groundings (open = *gsopen* and close = *gsclose*), possible classes are $C = \{C1, C2, C3, C4, C5, C6, C7\}$, and the formula $Has(GS) => Topic(C)$. In the formula, if all variables get a value (i.e., grounding) we call that a *world*. For example, $Has(gsclose) => Topic(C3)$ is referred to as x_{world}. We get these variables and groundings from annotated data instance pair (X_t, y_t) and learn weight using equation 10. While combining with RNN, we first get the groundings of variables (e.g., $Has(GS = gsclose)$) from input (X_t) at timestamp *t*, then consider to get weights of possible groundings. In this specific example, we generate a total of 7 groundings and their corresponding weights (Equation 11)

$$\langle Has(GS = gsclose) => Topic(C = C1), w1 \rangle \\ \langle Has(GS = gsclose) => Topic(C = C2), w2 \rangle \\ ... \\ \langle Has(GS = gsclose) => Topic(C = C7), w7 \rangle \quad (11)$$

For the weight learning, we used a pseudo-log-likelihood method [13]. Example of learned formula weights (on our dataset) can be found in Table II.

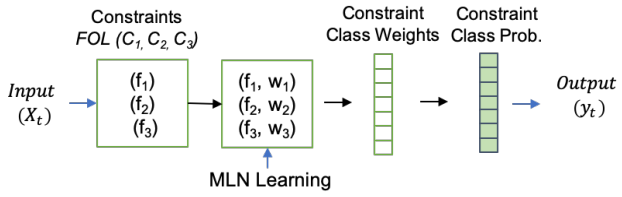


Fig. 2: Logic Learning Module. Constraints formulated in the first-order logic form (FOL) from feature variables whose weights (i.e., w_i) can be learned from data using MLN approach.

Constraint Weights to Class Probability. This step works with the learned formula weights. For a data point (frame or activity unit) p, the weights of each class is calculated as in Equation 12

$$w_i = \frac{1}{|S|} \sum_{f \in S} Weight(f, C_i) \quad (12)$$

where S is the set of True formulas in the data point p and $Weight(f, C_i)$ is the classweight of formula f for class C_i.

We convert these weights into probability by normalizing and taking log as follows

$$c_i = \log(\frac{1 + w_i}{\sum_{j=0}^{m}(1 + w_j)}) \quad (13)$$

where m is total number of classes. Thus the class probability vector is defined as $C = [c_0, c_1, ..., c_m]$. We use this probability for classification and combined with machine learning based classification methods. The predicted class for the data point p is identified by taking $argmax$ across all the classes as in Equation 14

$$classlabel = argmax([c_0, c_1, ..., c_m]) \quad (14)$$

V. RNN WITH LOGIC CONSTRAINTS

In this section we discuss how we regularize RNN models with logic constraint weights using two proposed methods.

PriorLayer. This method takes into account the constraint weights during the RNN training. An overview of the process is depicted in Figure 3. Equation 6 represents a fully connected linear layer which map RNN hidden state into class weights of size $|C|$. Intuitively, this layer transfers the previous learning and history information into prediction. On the other hand, domain knowledge can be thought of as already learned information which can be captured using constraints weights learning by MLN. Thus the formula weights (i.e., *Constraints Class Weight*) can be injected into the class prediction by adding a linear layer after the layer represented by Equation 6. We first concatenate the added layer's equation given in 15.

$$h2c' = W_{c_1} * h2c + W_{c_2} * c2c, \quad (15)$$

where $c2c$ is the weights of classes given by the constraints from the input data instance. The dimension of $h2c$ and $c2c$ are equal to the number of class labels. We learn the weights W_{c_1} and W_{c_2} along with other RNN parameters *during training*.

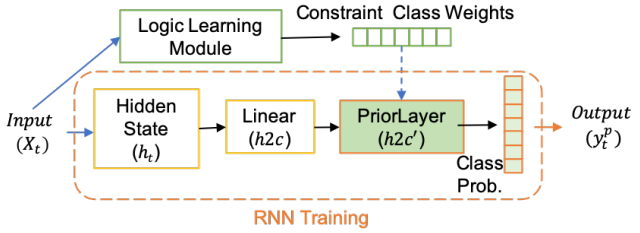


Fig. 3: PriorLayer. The input X_t at timestamp t goes through Logic Leaning Module which generate class weights. The class weight then added (element-wise) into the RNN Linear ($h2c$) layer to form PriorLayer($h2c'$). After that $softmax$ is used to generate final class probability. Then $argmax$ is applied on the class probabilities to generate class label prediction (y_t^p). The parameters weights of PriorLayer layer are learned along with other RNN parameters *during training* process.

Conflation. An overview of this method is depicted in Figure 4. RNN and the Logic Learning Module (LLM) can be be trained in isolation. For each testing input instance (X_t

at timestamp t) a trained-RNN produces class probabilities (P_L) and LLM produces class probabilities (P_K) (equation 13). We propose to combine these class probabilities by *conflation of probability distributions* [14] as in the equation 16.

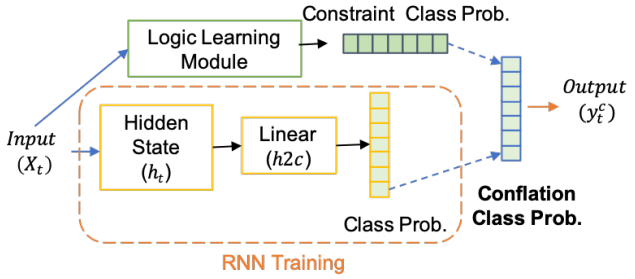


Fig. 4: Conflation. The RNN unit trained in isolation. Each testing input (X_t at timestamp t) goes through both trained-RNN and Logic Learning Module which produces two sets of class probabilities. These probabilities is then combined by *conflation of probability distributions* [14] as in the equation 16. Then $argmax$ is applied on the Conflation Class Probabilities to generate class label prediction (y_t^c).

$$P_C = \frac{P_L \times P_K}{P_L \times P_K + (1 - P_L) \times (1 - P_K)} \tag{16}$$

where P_C is the combined class probability of that class. Intuitively, the conflation is the distribution determined by the normalized product of the probability densities which is shown to be the unique probability distribution that minimizes the loss of Shannon Information in incorporating the combined information from P_L, and P_K into a single distribution P_C [14]. Note that, this approach does not require joined training with the RNN and thus can be applied during testing time.

Note that, our proposed methods PriorLayer, and Conflation are agnostic to the hidden state update procedure (applied after the hidden state is computed) and thus can be integrated to network variation which applied different state update techniques (e.g., LSTM, GRU). On the other hand, the combined models performance depends on the quality of formula weights learning that is domain knowledge. If the weights capture more prior knowledge it can contributed to the accuracy more efficiently. In contrast, a less accurate weight estimation (manual or learning) might have less impact on final performance of the combined models.

VI. Experiments

We conduct experiments on two robotic environments: (i) OpenAI Gym[1] (simulation), and (ii) DESK - A Robotic Activity Dataset for Dexterous Surgical Skill [15] (real Taurus robot used for surgical task). A summary of the datasets used in this paper is given in Table I. A fraction of data (126 frames for Gym, and 1405 frames for Taurus) was used for training MLN (formulae weight learning), and LSTM and rest of data was used for testing.

[1]https://gym.openai.com/

TABLE I: Robotic activity dataset stats

Robot	classes	frames	segments	frames/segment
Gym	4	20,188	1,612	12.5
Taurus	7	18,734	644	29.1

(i) **Gym - Pick and Place**. We collect demonstrations from the gym environment for the task of *pick and place*. The task is to place a box (black color) to a target location (red color) shown in Figure 5. For each demonstration, the box randomly placed on the table and the target set to any location in the three-dimensional space (x, y, z). We assign the four class labels to break down the total activity of placing a box on the target as shown in Figure 5. We collected robot kinematics data which was used for the experiments. Note that, we generate those demonstrations in the simulation along with collecting the activity annotation (class label).

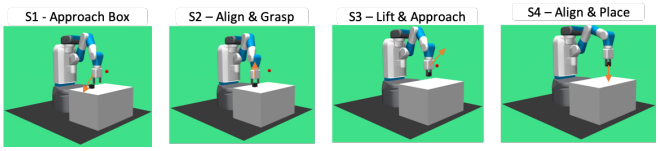


Fig. 5: Activity classes for the Pick and Place task for Gym *FetchPickAndPlace-v1*.

(ii) **DESK** dataset contains a library of surgical motions for the *peg transfer task*, one of the four basic laparoscopic surgical procedures, using three robots. This dataset has been used for robotic action recognition tasks [15], [20]. We conduct experiments on Taurus Dexterous Robot dataset. The peg transfer procedure is present in the fundamentals of laparoscopic surgery [21] which is often used to train surgeons [22], [23]. Figure 6 shows snippets of seven activities of Taurus II robot from the dataset. We use the kinematic data as features for the activity recognition task.

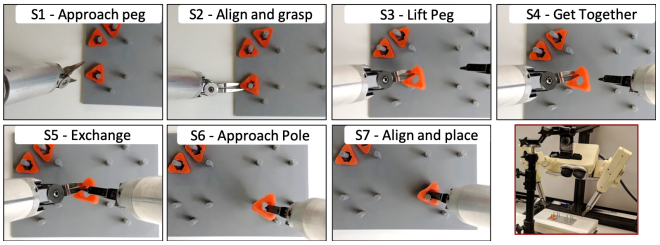


Fig. 6: Surgemes in the peg transfer task for the Taurus II robot. The image on the lower right is the Taurus robot [15]

Data Processing: We leverage kinematic feature data for the sequential classification task. We feed the kinematic data in each time-step to an LSTM network as input which predicts the class level corresponding to the input data. We processed the data and convert the feature variables into the binary format as described in section IV for constraints formulation. We leveraged these binary variables to incorporate domain/prior knowledge in the form of propositional logic. Note that these formulations of features help to easily

TABLE II: Sample from learned formula weights using MLN on Taurus robot data (in Figure 6). The weight of formula no. 5 is much higher for *exchange* class (S5) compared to other classes which matches with the intuition.

No.	Formula	Weight
1	$leftGsClosed \wedge rightGsClosed \Rightarrow S1$	2.37
2	$leftGsClosed \wedge rightGsClosed \Rightarrow S2$	1.26
3	$leftGsClosed \wedge rightGsClosed \Rightarrow S3$	-0.70
4	$leftGsClosed \wedge rightGsClosed \Rightarrow S4$	-1.21
5	$leftGsClosed \wedge rightGsClosed \Rightarrow S5$	30.81
6	$leftGsClosed \wedge rightGsClosed \Rightarrow S6$	-0.95
7	$leftGsClosed \wedge rightGsClosed \Rightarrow S7$	-1.00

embed human-intuitive constraints. For example, if two arms of a robot are getting close to each other its most likely "exchange" surgeme.

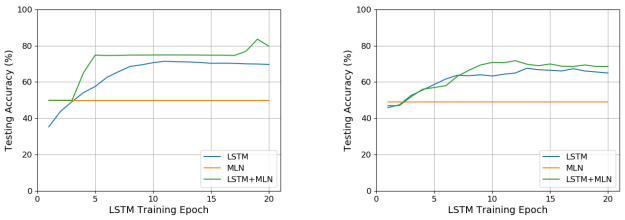
Constraints Formulation. To demonstrate the effectiveness of our approach we generate several constraints on both Gym and DESK dataset. We first used domain knowledge to come up with the formula that we want to learn from the data. Note that our approach allows domain expert to give importance (weight) of constraints. In case the expert unsure of the weights we leverage Markov logic network to learn the importance of the formula constraints based on data. This weighting process allows users to formulate as many formulae as possible without worrying about specifying importance.

Formula Weight Learning. After we formulate constraints formula, we leverage the MLN to learn its weights. Table II shows few sample constraints along with their learned weights. For the classification task, we aim to learn the weights of these constraint formulas corresponds to each class type.

Interestingly, the formula $leftGsClosed \wedge rightGsClosed \Rightarrow S5$ weights much higher for class S5 (exchange) compared to remaining classes. Intuitively, when both arms *gripper* is closed it is most likely an *exchange* activity in the DESK dataset and the learned weights also indicate the same. Later, we use these weights to classify activities and embed constraint in the machine learning-based classification.

Settings. We leveraged a Python library *pracmln*[2] to train the MLN weights. We used pseudo-log-likelihood (fast conjunction grounding) algorithm (BPLL_CG) as the learning method. We used the same training data as used for LSTM training for weight learning. For the experiments, we used a PyTorch[3] implementation of the LSTM network with a hidden layer dimension of 32. We used NLLLoss and SGD for optimizer with a learning rate of 0.1. We used the same network architecture for all the experiments for a fair comparison between methods. We recognized a surgeme segment using half of its frames and report overall accuracy (%) performance. We used the following machine configuration to run our experiments: 20 core-CPU with 256 GB of RAM, CPU Model: Intel(R) Xeon(R) Silver 4114 CPU @ 2.20GHz.

[2]http://www.pracmln.org/
[3]https://pytorch.org/



(a) Gym Robot (b) Taurus Robot

Fig. 7: Impact of MLN as LSTM training progresses. MLN performance is not dependent on epoch thus remain fixed. Overall, the combined method LSTM+MLN performs better at different epochs.

A. Results

Impact of MLN on LSTM Performance. Table III shows the best performance of different models on Gym and DESK testing data. We see that LSTM perform better compared to MLN. However, when combined with LSTM, MLN improves the performance of LSTM. Both the methods, PriorLayer and Conflation, improve the performance while the combined approach performs the best. Figure 7 shows the performance over the different training epoch. We see that at different training epochs MLN consistently improves the overall performance.

These results show that the domain knowledge incorporated through MLN is helping to improve overall sequential classification. However, the constraints formulae might not capture domain knowledge for all the classes (surgemes). Thus, the impact might vary accross class labels. Table IV shows the performance comparison of different models per surgeme class. Results vary across surgeme class labels, in some cases, MLN performs better than LSTM.

Note that MLN performance heavily depends on the learned constraint weights. Thus, where the constraints formula learned some meaningful weights, the association between formula and class label, from the domain knowledge it performs better. MLN might complement LSTM learning in some cases and thus improve combined classification accuracy. Overall, the combined model LSTM+MLN performs better which combines the best of both LSTM and MLN models.

MLN as Regularizer. Figure 8 shows training and testing accuracy for LSTM and LSTM+PriorLayer approach on Gym data. In this experiment, we used only PriorLayer modification and did not use Conflation.

In the initial stage of the training (i.e., early epoch) the improvement is higher. Also, we see that the LSTM *training* accuracy is higher than LSTM+PriorLayer training accuracy. However, the testing accuracy of the LSTM+PriorLayer is higher than LSTM only. This shows that without PriorLayer LSTM tends to overfit the data. After around 3 epoch the LSTM training accuracy reaches to 100% while testing accuracy remains the same. On the other hand, LSTM+PriorLayer training accuracy shows a less over-fitting trend while producing higher testing accuracy.

TABLE III: Test Accuracy (%) Comparison.

Dataset	LSTM	MLN	LSTM+PriorLayer	LSTM+Conflation	LSTM+MLN (Combined)
Gym	71	50	83	77	**84**
Taurus	68	49	71	69	**72**

TABLE IV: Per class Test Accuracy (%) Comparison

Dataset	Class	LSTM	MLN	LSTM+MLN
Gym	S1	98	100	99
	S2	99	100	100
	S3	63	0	76
	S4	35	0	33
Taurus	S1	87	91	91
	S2	80	30	78
	S3	87	0	88
	S4	58	83	76
	S5	100	84	94
	S6	48	35	54
	S7	63	20	62

Note that, we tested regularization capability only for the PriorLayer method. The Conflation is applied only at the testing time and the training is done in the same way as the regular LSTM in this scenario. Thus we did not report the training accuracy of the Conflation.

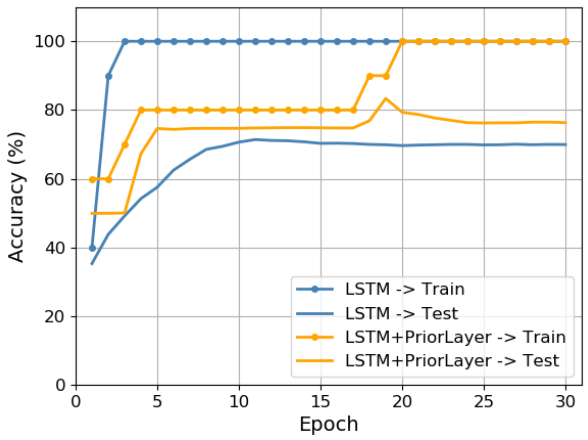


Fig. 8: Training and testing accuracy for LSTM with and without PriorLayer approach on Gym dataset. The LSTM model overfit early (epoch=3) while LSTM+PriorLayer model overfit later (epoch=20) thus achieves higher testing accuracy.

VII. Related Work

Incorporating problem structures and domain knowledge in machine learning approaches have been studied and applied extensively in literature [24], [25], [26], [27], [28], [29] because of its potential to improve learning and generalization while improving interpretability. Posterior regularization and related frameworks [24] have been successfully used to incorporate structured constraints (logic rules) on probabilistic models. In recent time, constraints have been applied through extended posterior regularization on deep generative models [26], and through decision diagrams on generative adversarial network (GAN) [30] in various context. A most common form of these logic constraints is the first-order logic (FOL). The constraints weights can be set manually [31], [32], [33] from domain expert or can be learned [13], [26] from data in more practical settings. In contrast to these works, our approach learns the weight of the constraints in isolation to the RNN module and applied to RNN during training and testing. While our approach is adaptable to any weight learning mechanism, in this paper we leverage Markov Logic Network (MLN) [13]. Different forms of regularization have been proposed by modifying network architectures [34], [35], [36], [37]. In contrast, we leverage domain knowledge (logic constraints) as additional information to regularize RNN.

Our proposed method is agnostic to task design and dataset. Thus our approach can be applied to other similar surgical classification datasets, such as JIGSAWS [38]. Moreover, it can be integrated into surgical systems where the surgeme classification is used as sub-modules such as in SARTRES [39], and DESERTS [40].

VIII. Conclusion

We propose a systematic approach to regularize RNN-based sequential prediction by incorporating domain knowledge with logic constraints. We apply two methods - adding a layer after the hidden unit, and combining class probabilities. We evaluate these methods for robotic activity classification on simulation (Gym) and real-world robotic (DESK - Taurus) dataset. We observe that the logic constraint-based model helps to improve LSTM performance. Furthermore, MLN shows regularization capability where it improves accuracy in initial LSTM training while avoiding over-fitting early and thus improves the final classification accuracy on unseen data. Additionally, we observe that MLN has a considerable impact on the LSTM prediction where the constraint weights (capturing domain knowledge) are learned properly.

Acknowledgment

This work was supported by the following funding agencies: The NSF FMitF program under award No. CCF-1918327 and the CR-II program under Award No. IIS-1850243, the Office of the Assistant Secretary of Defense for Health Affairs under Award No. W81XWH-18-1-0769, the NSF Center for Robots and Sensors for the Human Well-Being under Award No. CNS-1439717. The Computational infrastructure was partially supported by Microsoft AI for Earth program. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by these institutions.

REFERENCES

[1] J. L. Elman, "Finding structure in time," *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[2] B. A. Pearlmutter, "Gradient calculations for dynamic recurrent neural networks: A survey," *IEEE Transactions on Neural networks*, vol. 6, no. 5, pp. 1212–1228, 1995.

[3] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 5, pp. 855–868, 2008.

[4] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[5] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[6] R. DiPietro, C. Lea, A. Malpani, N. Ahmidi, S. S. Vedula, G. I. Lee, M. R. Lee, and G. D. Hager, "Recognizing surgical activities with recurrent neural networks," in *International conference on medical image computing and computer-assisted intervention*. Springer, 2016, pp. 551–558.

[7] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun, "Doctor ai: Predicting clinical events via recurrent neural networks," in *Machine Learning for Healthcare Conference*, 2016, pp. 301–318.

[8] S. Mark, G. Michael, N. Marshall, L. Anthony, and B. Suresh, "Machine learning in health care: A critical appraisal of challenges and opportunities," *EGEMS (Washington, DC)*, vol. 7, p. 1, 2019.

[9] T. Windle, J. C. McClay, and J. R. Windle, "The impact of domain knowledge on structured data collection and templated note design," *Applied clinical informatics*, vol. 4, no. 03, pp. 317–330, 2013.

[10] M. V. Schaverien, "Development of expertise in surgical training," *Journal of surgical education*, vol. 67, no. 1, pp. 37–43, 2010.

[11] H. Sadideen, A. Alvand, M. Saadeddin, and R. Kneebone, "Surgical experts: Born or made?" *International Journal of Surgery*, vol. 11, no. 9, pp. 773 – 778, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S174391911301008X

[12] R. K. Reznick and H. MacRae, "Teaching surgical skills—changes in the wind," *New England Journal of Medicine*, vol. 355, no. 25, pp. 2664–2669, 2006.

[13] M. Richardson and P. Domingos, "Markov logic networks," *Machine learning*, vol. 62, no. 1-2, pp. 107–136, 2006.

[14] T. Hill, "Conflations of probability distributions," *Transactions of the American Mathematical Society*, vol. 363, no. 6, pp. 3351–3372, 2011.

[15] N. Madapana, M. M. Rahman, N. Sanchez-Tamayo, M. V. Balakuntala, G. Gonzalez, J. P. Bindu, L. N. V. Venkatesh, X. Zhang, J. B. Noguera, T. Low, R. Voyles, Y. Xue, and J. Wachs, "Desk: A robotic activity dataset for dexterous surgical skills transfer to medical robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-2019)*, 2019.

[16] H. C. Lin, I. Shafran, D. Yuh, and G. D. Hager, "Towards automatic skill evaluation: Detection and segmentation of robot-assisted surgical motions," *Computer Aided Surgery*, vol. 11, no. 5, pp. 220–230, 2006.

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

[18] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.

[19] M. R. Genesereth and N. J. Nilsson, *Logical Foundations of Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987.

[20] M. M. Rahman, N. Sanchez-Tamayo, G. Gonzalez, M. Agarwal, V. Aggarwal, R. M. Voyles, Y. Xue, and J. Wachs, "Transferring dexterous surgical skill knowledge between robots for semi-autonomous teleoperation," in *28th IEEE International Conference on Robot and Human Interactive Communication (Ro-Man-2019)*, 2019.

[21] E. M. Ritter and D. J. Scott, "Design of a Proficiency-Based Skills Training Curriculum for the Fundamentals of Laparoscopic Surgery," *Surgical Innovation*, vol. 14, no. 2, pp. 107–112, Jun. 2007. [Online]. Available: https://doi.org/10.1177/1553350607302329

[22] N. A. Arain, G. Dulan, D. C. Hogg, R. V. Rege, C. E. Powers, S. T. Tesfay, L. S. Hynan, and D. J. Scott, "Comprehensive proficiency-based inanimate training for robotic surgery: reliability, feasibility, and educational benefit," *Surgical Endoscopy*, vol. 26, no. 10, pp. 2740–2745, Oct. 2012. [Online]. Available: https://doi.org/10.1007/s00464-012-2264-x

[23] R. A. Joseph, A. C. Goh, S. P. Cuevas, M. A. Donovan, M. G. Kauffman, N. A. Salas, B. Miles, B. L. Bass, and B. J. Dunkin, ""Chopstick" surgery: a novel technique improves surgeon performance and eliminates arm collision in robotic single-incision laparoscopic surgery," *Surgical Endoscopy*, vol. 24, no. 6, pp. 1331–1335, Jun. 2010. [Online]. Available: https://doi.org/10.1007/s00464-009-0769-8

[24] K. Ganchev, J. Gillenwater, B. Taskar *et al.*, "Posterior regularization for structured latent variable models," *Journal of Machine Learning Research*, vol. 11, no. Jul, pp. 2001–2049, 2010.

[25] Z. Hu, X. Ma, Z. Liu, E. Hovy, and E. Xing, "Harnessing deep neural networks with logic rules," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 2410–2420.

[26] Z. Hu, Z. Yang, R. R. Salakhutdinov, L. Qin, X. Liang, H. Dong, and E. P. Xing, "Deep generative models with learnable knowledge constraints," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 501–10 512.

[27] A. Li and P. Beek, "Bayesian network structure learning with side constraints," in *International Conference on Probabilistic Graphical Models*, 2018, pp. 225–236.

[28] N. Takeishi and Y. Kawahara, "Knowledge-based regularization in generative modeling," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 2390–2396, main track. [Online]. Available: https://doi.org/10.24963/ijcai.2020/331

[29] L. Yao, Y. S. Zhang, B. Wei, Z. Jin, R. Zhang, Y. Zhang, and Q. Chen, "Incorporating knowledge graph embeddings into topic modeling," in *AAAI*, 2017.

[30] Y. Xue and W.-J. van Hoeve, "Embedding decision diagrams into generative adversarial networks," in *International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research*. Springer, 2019, pp. 616–632.

[31] M. Fischer, M. Balunovic, D. Drachsler-Cohen, T. Gehr, C. Zhang, and M. Vechev, "Dl2: Training and querying neural networks with logic," in *International Conference on Machine Learning*, 2019, pp. 1931–1941.

[32] J. Xu, Z. Zhang, T. Friedman, Y. Liang, and G. Van den Broeck, "A semantic loss function for deep learning with symbolic knowledge," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. Stockholmsmässan, Stockholm Sweden: PMLR, 10–15 Jul 2018, pp. 5502–5511. [Online]. Available: http://proceedings.mlr.press/v80/xu18h.html

[33] D. Andrzejewski, X. Zhu, M. Craven, and B. Recht, "A framework for incorporating general domain knowledge into latent dirichlet allocation using first-order logic," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[34] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[35] S. Merity, B. McCann, and R. Socher, "Revisiting activation regularization for language rnns," *arXiv preprint arXiv:1708.01009*, 2017.

[36] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[37] A. B. Dieng, R. Ranganath, J. Altosaar, and D. M. Blei, "Noisin: Unbiased regularization for recurrent neural networks," *arXiv preprint arXiv:1805.01500*, 2018.

[38] Y. Gao, S. S. Vedula, C. E. Reiley, N. Ahmidi, B. Varadarajan, H. C. Lin, L. Tao, L. Zappella, B. Béjar, and D. D. Yuh, "JHU-ISI gesture and skill assessment working set (JIGSAWS): A surgical activity dataset for human motion modeling," in *MICCAI Workshop: M2CAI*, vol. 3, 2014, p. 3.

[39] M. M. Rahman, M. V. Balakuntala, G. Gonzalez, M. Agarwal, U. Kaur, V. L. Venkatesh, N. Sanchez-Tamayo, Y. Xue, R. M. Voyles, V. Aggarwal *et al.*, "Sartres: a semi-autonomous robot teleoperation environment for surgery," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, pp. 1–8, 2020.

[40] G. Gonzalez, M. Agarwal, M. V. B. S. Mur, M. M. Rahman, U. Kaur, R. Voyles, V. Aggarwal, Y. Xue, and J. Wachs, "Deserts: Delay-tolerant semi-autonomous robot teleoperation for surgery," in *In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA 2021)*, 2021.