Runtime Detection of Executional Errors in Robot-Assisted Surgery

Zongyu Li, Kay Hutchinson, and Homa Alemzadeh

Abstract— Despite significant developments in the design of surgical robots and automated techniques for objective evaluation of surgical skills, there are still challenges in ensuring safety in robot-assisted minimally-invasive surgery (RMIS). This paper presents a runtime monitoring system for the detection of executional errors during surgical tasks through the analysis of kinematic data. The proposed system incorporates dual Siamese neural networks and knowledge of surgical context, including surgical tasks and gestures, their distributional similarities, and common error modes, to learn the differences between normal and erroneous surgical trajectories from small training datasets. We evaluate the performance of the error detection using Siamese networks compared to single CNN and LSTM networks trained with different levels of contextual knowledge and training data, using the dry-lab demonstrations of the Suturing and Needle Passing tasks from the JIGSAWS dataset. Our results show that gesture specific task nonspecific Siamese networks obtain micro F1 scores of 0.94 (Siamese-CNN) and 0.95 (Siamese-LSTM), and perform better than single CNN (0.86) and LSTM (0.87) networks. These Siamese networks also outperform gesture nonspecific task specific Siamese-CNN and Siamese-LSTM models for Suturing and Needle Passing.

I. INTRODUCTION

Robot-assisted minimally invasive surgery (RMIS) has become a standard approach across different specialties, including urology, gynecology, and general surgery. Surgical robots translate the surgeon's hand, wrist, and finger movements into precise motions of robotic manipulators and miniaturized surgical instruments. Potential benefits include enhanced visualization, increased dexterity, smaller incisions, and shorter recovery time. However, the safety of RMIS can be compromised due to unintentional human errors [1]–[4] or the potential vulnerabilities of surgical robots to accidental faults and malicious attacks targeting sensors, actuators [5], communication between the surgeon's console and the robot [6], [7], or the robot control software [8], [9].

Developing machine learning (ML) techniques for automated and objective evaluation of surgical skills based on kinematic [10], [11] and video [12] data from procedures has been an active area of research. However, the current ML techniques are mostly developed for *offline* analysis of surgeon's performance. The next generation of surgical robots and simulators can be enhanced with capabilities for *runtime* analysis of surgical tasks and providing data-driven feedback to surgeons during training or actual surgery to improve safety, efficiency, and quality of care [13], [14].

Previous works have proposed the idea of runtime surgical monitoring for detection and prevention of safety-critical events [8], [14], [15], surgeon authentication [16], and virtual coaching [13]. In [8], the authors developed a dynamic model-based anomaly detection technique for mitigation of unsafe events caused by malicious attacks on a surgical robot controller. However, errors can also happen in the operational context as a result of surgeons' sub-optimal performance [15]. In [1], a new rubric was proposed for evaluating executional and procedural errors based on video and kinematic data from dry-lab simulation experiments. This work showed that the type and frequency of errors are dependent on surgical context, characterized by the specific tasks (e.g., Suturing) and gestures (e.g., Pulling suture) performed by the surgeon.

Recently, deep learning (DL) techniques (e.g., CNN and LSTM networks), have been applied to the prediction of potential unsafe events caused by unintentional human errors in simulated surgical training tasks [14], [15] and retinal microsurgery [17]. These works showed preliminary evidence for the importance of incorporating the knowledge of surgical gestures in model training [14], [15] to improve detection performance. However, the performance of DL models is often negatively affected by the small size of training data and the lack of labeled datasets on errors [14].

In this paper we aim to address the challenges in runtime detection of executional errors in robot-assisted surgery by developing models that can achieve reliable performance given small training data. Our goal is to investigate whether learning the differences among erroneous and normal trajectories, and incorporating different levels of surgical context information can improve the error detection accuracy. The main contributions of the paper are as follows:

- Developing a runtime monitoring system based on dual Siamese neural network architectures that learns the differences between normal and erroneous kinematic trajectories and enables reliable error detection performance using small training dataset sizes.
- Demonstrating that incorporating the knowledge of surgical context, including information on the current surgical task and gesture, which captures distributional similarities across gesture and task trajectories and specific types and frequencies of errors in model training, can improve the error detection performance.
- Evaluating the proposed system for runtime detection of executional errors using the publicly-available JIGSAWS dataset collected from dry-lab robot-assisted sur-

Authors are with the Department of Electrical and Computer Engineering (ECE), University of Virginia, Charlottesville, VA 22904, USA. {zl7qw, kch4fk, ha4d}@virginia.edu

The corresponding code for this paper is available at https://github.com/UVA-DSA/ExecErr_Detect_Siamese

gical tasks (Suturing and Needle Passing) performed on a da Vinci robot from Intuitive Surgical Inc. [18]. Our experiments show that dual input Siamese neural networks can achieve on average a better performance than single CNN and LSTM models after augmenting the data with more trajectories (e.g., for gesture nonspecific and task specific models, F1 score for Siamese-CNN vs. CNN: 0.79 vs. 0.72 and Siamese-LSTM vs. LSTM: 0.87 vs. 0.60). We also found that incorporating the knowledge of surgical gestures and specifically training the models with gesture specific trajectory examples helps with improving F1 scores more than training with more gesture nonspecific trajectory examples.

II. PROBLEM STATEMENT

We model surgical procedures as a hierarchy composed of a sequence of tasks (e.g., Suturing, Needle Passing) coming from a library of surgical tasks T, each represented by a grammar graph with steps defined as gestures which are the atomic surgical activities in a procedure [1], [14]. We denote the library of all surgical gestures with G such as in Table I. The change in the surgical context happens in the temporal domain as a result of the change in the position and orientation of the surgical instruments' end-effectors, leading to the corresponding change in the gestures and tasks.

Our previous analysis of data from dry-lab robotic surgery experiments showed that executional and procedural errors can negatively affect surgical performance [1]. Executional errors are defined as the failure of a specific step in a task. Procedural errors are defined as the omission or rearrangement of correctly undertaken steps within a task [4]. Our goal is to detect executional errors at runtime by analyzing the kinematic data collected from the robot.

The input kinematic data to our error detection system is a multi-dimensional time series matrix $\chi_k \in \mathbb{R}^{N_{ch} \times N_s}$, with N_{ch} representing the number of features extracted from the robot's kinematic data and N_s representing the size of the sliding window. We model the data pre-processing step as a function $f(\cdot)$ on each input window χ_k. We assume each χ_k can be mapped to a gesture class $G_i \in G$ using a gesture segmentation and classification method like those proposed in the literature [19]–[27]. We adopt the gesture-specific error rubric presented in [1] to define a set of unique executional error modes E corresponding to each gesture class as shown in Table I. The output of the error detector is an error probability y_k assigned to the input window χ_k, as shown in Figure 1, where G_k and T_k represent the gesture and task classes that the input is mapped to and $Error$ represents the event that one of the error modes in E is observed in this window.

Our goal is to investigate the following research questions by designing and evaluating different neural network architectures for runtime error detection using different levels of contextual information and amounts of training data:

RQ1: Can learning the patterns of differences between erroneous and normal trajectories using a dual input network rather than just learning error patterns improve error

TABLE I: Gesture Definitions (adopted from [28]), Common Gesture-specific Executional Error Modes (adopted from [1])

Gesture	Description	Common Gesture-Specific Errors
G1	Reaching for needle with right hand	Multiple attempts
G2	Positioning needle	Multiple attempts, Out of view
G3	Pushing needle through the tissue	Multiple attempts
G4	Transferring needle from left to right	Multiple attempts, Needle orientation
G6	Pulling suture with left hand	Multiple attempts, Out of view
G8	Orienting needle & Uses tissue/ instrument for stability	Multiple attempts, Needle orientation
G9	Using right hand to help tighten suture	Multiple attempts

detection performance?

RQ2: How does incorporating the knowledge of the tasks and gestures being performed by the surgeon/robot affect the performance of erroneous gesture detection?

We specifically consider the following setups for training different models with different levels of contextual knowledge on the tasks and gestures:

- Gesture specific and task specific models
- Gesture specific and task nonspecific models
- Gesture nonspecific and task specific models
- Gesture nonspecific and task nonspecific model

We hypothesize that the models trained with the data specific to a task or gesture can achieve better performance than the models with no contextual knowledge.

III. METHODS

We develop different single and dual neural network architectures for runtime detection of executional errors, including a CNN network and an LSTM network (inspired by state-of-the-art gesture recognition and error detection models [14], [23]) and their corresponding Siamese networks, respectively built with CNN encoders and LSTM layers (see Figure 1). This section describes the structure of these networks, the data pre-processing done for training and testing the networks, and how the output from the models were utilized to detect erroneous gestures.

A. Data Pre-processing

The input to the error detection networks is a 26-dimensional time-series matrix. To reduce the computational cost and enable faster training convergence, we time-normalized and downsampled the input time-series with a factor of 2. A sliding window of 30 samples (derived from the median length of the shortest gesture in the dataset) with a stride of 20 samples was used to obtain the input data.

B. CNN Network

The CNN network shown in Figure 1 is designed with two sets of blocks composed of a 1d convolution layer, a max-pooling layer, a dropout layer, and a batch normalization
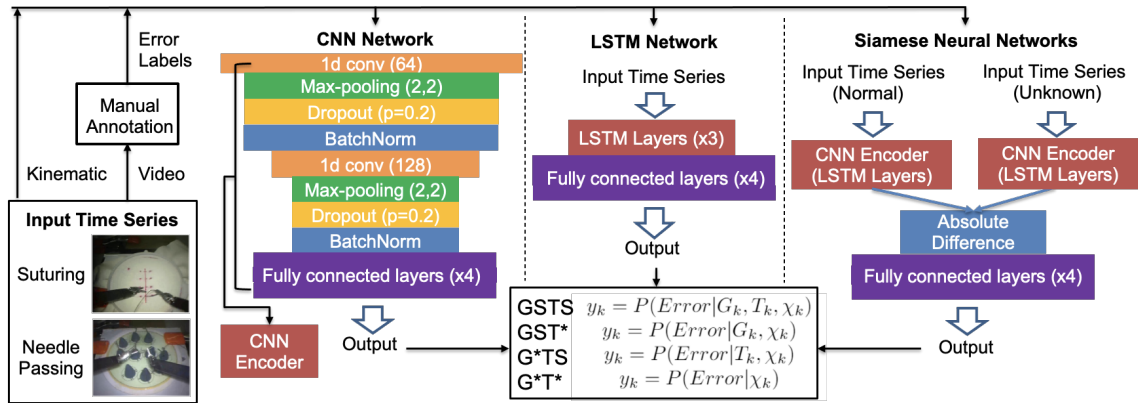
Fig. 1: Different Neural Network Structures with Different Training Setups and Contextual Information

layer. The output from the last batch normalization layer is flattened and fed into four fully connected layers, including three layers with ReLU activation functions [29] and one last layer with a Sigmoid activation function that outputs the probability of the gesture being erroneous. To train the network, $[f(\chi_k), Y_k]$ is fed into the network, where $f(\chi_k)$ is the data after pre-processing, and Y_k is the ground truth indicating whether the gesture is erroneous ($Y_k = 1$) or normal ($Y_k = 0$). The binary cross entropy loss function [30] is used to train this network.

C. LSTM Network

The LSTM network contains 3 LSTM layers. The hidden states from the last LSTM layer are flattened and fed into four fully connected layers with dropout layers and with ReLU activation functions for the first three layers. The last fully connected layer is connected to a Sigmoid activation function that outputs the probability of a gesture being normal or erroneous. The training process is similar to the CNN network.

D. Dual Siamese Neural Networks

Siamese neural networks are a class of neural network architectures with two or more identical structures that focus on learning a similarity function for indicating whether the inputs are from the same or different classes. Because of their ability to achieve better prediction accuracy than single networks for small data sizes, they have been applied to applications such as signature verification, face recognition [31], and EEG-based brain-computer interface signal classification [32].

As shown in Figure 1, we designed two different Siamese networks, each containing two identical neural networks, with the same structure as the single input CNN encoder or the LSTM layers used for the CNN and LSTM networks described above. For binary classification, each Siamese network takes in two gesture windows and processes them using its two identical structures in parallel. Then, the absolute value of the difference between outputs from the two structures is calculated. We then use three fully connected layers with ReLU activation, and the fourth layer is connected to a Sigmoid activation function that outputs the probability that one of the two windows belongs to a different class.

To train the Siamese neural network, the tuple $[f(\chi_k), f(\chi_j), Y_{kj}]$ is fed into the network, where $f(\chi_k), f(\chi_j)$ are pairs of the gesture windows after pre-processing and Y_{kj} denotes the label. Y_{kj} is assigned a value of 0 when χ_k, χ_j are both normal gesture windows and a value of 1 when one of the windows is erroneous and the other is normal. The binary cross entropy loss function is used to train the network.

During the evaluation phase, each input window is paired with all the existing normal windows in the training dataset. The network outputs a value of 0 when the input window is indicated to be in the same class as a normal window and a value of 1, meaning that the unknown gesture window is not normal, and thus, erroneous. We then use majority voting to fuse the results from all the pairs to generate a final normal or erroneous predicted label for each input window.

IV. EXPERIMENTAL EVALUATION

All experiments were conducted on a 64-bit PC with an Intel Core i9 CPU @ 3.70GHz and 32GB RAM running Linux Ubuntu 20.04 and an NVIDIA RTX2080 Ti 11GB GPU. The neural network models were implemented using PyTorch [33] version 1.8.0.

A. Datasets

The JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) [28] is a publicly available dataset, collected using the da Vinci surgical robot [18] from eight surgeons with different levels of skills and expertise during three dry-lab surgical tasks (Suturing, Knot Tying, and Needle Passing) on a bench-top model. The JIGSAWS dataset includes synchronized kinematic and video data recorded at 30Hz along with manual annotations indicating the gestures for each trajectory and scores quantifying the surgeons' performance.

We used the kinematic data from the robot's patient-side manipulators (PSM) for the Suturing and Needle Passing tasks and the gesture labels from the JIGSAWS dataset. The kinematic variables included tool tip position, orientation, linear velocity, rotational velocity, and gripper angle for the PSMs of both the left and right hands. Tool tip orientation was described using Euler angles calculated from the left and right hand rotational matrices to reduce the dimensionality of the inputs. We used the executional errors labels for

Training	Task	Network	G1	G2	G3	G4	G6	micro F1	micro F1	Training
Gesture specific Task specific (GSTS)	Suturing	CNN	**0.66**	**0.48**	**0.73**	0.72	0.90	**0.78**	0.72	Gesture nonspecific Task specific (G*TS)
		Siamese -CNN	0.58	0.29	0.70	**0.75**	**0.91**	0.58	**0.79**	
		LSTM	0.52	**0.48**	0.73	**0.84**	**0.94**	0.52	0.60	
		Siamese -LSTM	**0.66**	0.33	**0.74**	0.75	0.90	**0.66**	**0.87**	
	Data	Total Trials	29	166	164	119	163	641	641	
		Erroneous Trials	8 (28%)	22 (13%)	82 (51%)	71 (60%)	121 (74%)	304 (47%)	304 (47%)	
	Needle Passing	CNN	**0.57**	**0.61**	0.36	**0.30**	0.43	0.43	0.56	
		Siamese -CNN	0.51	0.55	**0.43**	0.20	**0.46**	0.43	**0.74**	
		LSTM	0.52	**0.82**	**0.75**	**0.42**	0.51	**0.75**	0.56	
		Siamese -LSTM	**0.60**	0.64	0.48	0.24	**0.54**	0.47	**0.81**	
	Data	Total Trials	30	117	111	83	112	453	453	
		Erroneous Trials	11 (37%)	55 (47%)	17 (15%)	23 (28%)	46 (41%)	152 (34%)	152 (34%)	
Gesture specific Task nonspecific (GST*)	Both Tasks	CNN	0.74	0.73	0.82	0.86	0.86	0.86		
		Siamese -CNN	**0.89**	**0.87**	**0.92**	**0.87**	**0.94**	**0.94**		
		LSTM	0.72	0.62	0.83	0.73	0.87	0.87	0.67	**Baseline G*T***
		Siamese -LSTM	**0.89**	**0.87**	**0.93**	**0.90**	**0.95**	**0.95**	0.68	
	Data	Total Trials	59	283	275	202	275	1094		
		Erroneous Trials	19 (32%)	77 (27%)	99 (36%)	94 (47%)	167 (61%)	456 (43%)		

TABLE II: Performance of CNN, Siamese-CNN, LSTM, and Siamese-LSTM Models with different Training Setups. Results for gestures G8 and G9 are not shown because of very small data available for these gestures (e.g., less than 3 erroneous trials in Needle Passing). We obtained the baseline **G*T*** models with our top performing network pairs (Siamese-LSTM/LSTM).

JIGSAWS presented in [1]. Each label for a gesture trial is either normal (0) or erroneous (1). Table I summarizes the gesture definitions and their dominant error modes.

B. Experimental Setup

We experimented with the following training setups for the network structures described in the previous section. For each training setup, we performed cross validation with the Leave-One-SuperTrial-Out (LOSO) setup of the JIGSAWS dataset [28] (training on 4 super trials and leaving one super trial out for evaluation) and calculated the mean F1 scores across all folds. For each setup, we tuned the network parameters, including learning rate for the Adam Optimizer [34], batch size and epoch number with nested cross validation.

1) Gesture Specific and Task Specific training (GSTS): In this setup separate models were trained for each gesture from each task, resulting in 10 different models.

2) Gesture Specific and Task Nonspecific training (GST):* This setup includes five separate models for different gesture classes (G1, G2, G3, G4, G6) by combining gesture-specific data from both tasks.

*3) Gesture Nonspecific and Task Specific training (G*TS):* In this setup two models were trained by combining all different gesture data for each task.

*4) Gesture Nonspecific and Task Nonspecific training (G*T*):* This setup only included one model as the baseline by training on all the gesture data from both tasks.

Although the total size of the dataset is fixed, the amount of data (both normal and erroneous examples) used for training the models in each setup is different. For example, the G*T* setup includes one model with the maximum training data size while the GSTS models are trained with smallest amount of data relevant to each specific gesture in each task. Table II shows the total number of trials or gesture examples along with the number of erroneous gesture examples used for training in each setup. For evaluating the performance of different network architectures and comparing different training setups, we use the same testing setup and calculate the mean F1 score with cross validation.

C. Gesture Specific and Task Specific (GSTS) Models

In this experiment, we evaluated the performance of the CNN and LSTM networks versus the corresponding Siamese networks when trained with **GSTS** training. The results for different **GSTS** models are shown in the "Suturing" and "Needle Passing" rows in Table II. Each cell in these two rows provides the mean F1 score for each **GSTS** model.

Observation 1: There are no apparent advantages of Siamese networks with GSTS training. The Siamese network performance compared with that of its corresponding CNN or LSTM network varies across different gestures. For example, in G1 in task Suturing, CNN performs better than Siamese-CNN, and Siamese-LSTM performs better than LSTM. However, in G4 in task Suturing, Siamese-CNN performs better than CNN, and LSTM performs better than Siamese-LSTM. Hence, among **GSTS** models, there is no observed advantage of using Siamese networks over CNN and LSTM networks. This is likely due to model variance resulting from training with very small datasets.

Observation 2: There is a positive correlation between error percentages and F1 scores for GSTS models. Figure 2a shows the percentage of erroneous trials in each dataset used for training each **GSTS** model versus the F1 score achieved in both the Suturing and Needle Passing tasks and for the four different network structures. We see a positive correlation between the F1 scores and error percentages across all the network structures. The same observation about the impact of error sample sizes on detection accuracy was made in [14].

> **Insight 1:** Having a higher percentage of error samples helps improve the performance of **GSTS** models.

D. Gesture Specific and Task Nonspecific (GST*) Models

In this experiment, our goal is to understand whether training **GST*** models that incorporate gesture data from different tasks can achieve better error detection performance than the **GSTS** models. This is motivated by the observation that the same gestures from different tasks share similar trajectories

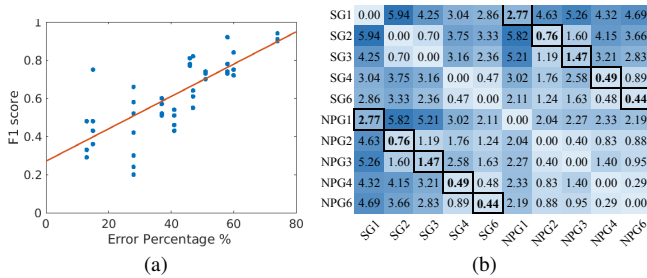|	SG1	SG2	SG3	SG4	SG6	NPG1	NPG2	NPG3	NPG4	NPG6
SG1	0.00	5.94	4.25	3.04	2.86	**2.77**	4.63	5.26	4.32	4.69
SG2	5.94	0.00	0.70	3.75	3.33	5.82	**0.76**	1.60	4.15	3.66
SG3	4.25	0.70	0.00	3.16	2.36	5.21	1.19	**1.47**	3.21	2.83
SG4	3.04	3.75	3.16	0.00	0.47	3.02	1.76	2.58	**0.49**	0.89
SG6	2.86	3.33	2.36	0.47	0.00	2.11	1.24	1.63	0.48	**0.44**
NPG1	**2.77**	5.82	5.21	3.02	2.11	0.00	2.04	2.27	2.33	2.19
NPG2	4.63	**0.76**	1.19	1.76	1.24	2.04	0.00	0.40	0.83	0.88
NPG3	5.26	1.60	**1.47**	2.58	1.63	2.27	0.40	0.00	1.40	0.95
NPG4	4.32	4.15	3.21	**0.49**	0.48	2.33	0.83	1.40	0.00	0.29
NPG6	4.69	3.66	2.83	0.89	**0.44**	2.19	0.88	0.95	0.29	0.00

(a)	(b)

Fig. 2: (a) F1 Scores vs. Error Percentages in Training Datasets, (b) Average KL Divergence Among Distribution of Normal Gestures

and error modes [1] and can provide more examples of errors of the same type for training. Thus, we combined the gesture data from the Suturing and Needle Passing tasks. We also compared the Siamese networks' performance with that of the CNN and LSTM networks. The row named "Both Tasks" in Table II shows the results for the **GST*** models.

Observation 1: GST* models achieve comparable or better performance than GSTS models for most cases in the four different network structures. For the CNN network, G1, G2, G3 and G4's F1 scores as shown in "Both Tasks" are greater than those from "Suturing" and "Needle Passing." For G6, the **GST*** model achieves comparable results to "Suturing" (0.86 vs. 0.90), but better results than "Needle Passing" (0.86 vs. 0.43). For the LSTM network, the F1 scores in "Both Tasks" are comparable or greater than those for G1, G2, and G3 in Suturing and for G1, G3, G4, and G6 in Needle Passing. The F1 scores of the Siamese-CNN and Siamese-LSTM in "Both Tasks" are greater than those in "Suturing" and "Needle Passing" for all gestures.

Observation 2: The baseline G*T* model can achieve better performance than GSTS models, but GST* models achieve better performance than the baseline. The performance of the baseline classifier trained with all the data without the knowledge of gestures and tasks is shown in Figures 3a and 3b as black lines. For the LSTM network, we observe that the baseline has better performance than the **GSTS** models for G1 and G2 in the Suturing task, and G1, G4, and G6 in the Needle Passing task. For the Siamese-LSTM network, the baseline has better performance in G1 and G2 for the Suturing task, and for all gestures in the Needle Passing task. This increase in performance is likely due to an increase in the number of training samples. Because the network sees more trajectory examples, it learns to better differentiate normal and erroneous trajectories. From Figures 3a and 3b, we also note that the performance of the baseline is better than the **G*TS-S** and **G*TS-N** models shown as dashed lines for the LSTM network, but for the Siamese-LSTM network the baseline performance is worse. This could be because training with more varied data confuses the Siamese network, since it calculates dual input differences and performs voting to determine the final result. We will focus on the comparison between **GSTS**, **GST*** and **G*TS** in Section IV-E. However, the **GST*** models (shown in Figures 3a and 3b with the green bars) still perform better than the baseline or the **G*TS** models in most cases. When we

combine gestures from both tasks, we increase the number of gesture specific training samples. In order to investigate the similarity of normal gestures between tasks, we used KL divergence to quantify the difference in the distributions of the kinematic variables before data pre-processing. In Figure 2b, the outlined black squares show the pairwise KL-divergence between distributions of the same gestures from two different tasks. These values are the smallest in the rows for G1, G4, and G6. Also in G2 and G3, 0.76 and 1.47 are smaller than most other elements in the row. A similar pattern holds when comparing Needle Passing to Suturing, but Needle Passing gestures are more similar to other Needle Passing gestures of different types. This shows that the gestures from Suturing are more similar to the same gestures in Needle passing than gestures of different types in the two tasks.

> **Insight 2:** Increasing the amount of training data by adding *more relevant data* (e.g., data from the same gesture class) improves the performance more than adding more irrelevant data.

Observation 3: GST* models can achieve better results than GSTS models and the baseline G*T* model despite having lower error percentages. From the insights of the **GSTS** training in the previous section, we noted that higher error percentage correlated with better performance. However, we observe that the **GST*** LSTM obtains a higher F1 score for G1 (0.72) with an error percentage of 32% compared to the **GSTS** LSTM models for Suturing (0.52, 28%) and Needle Passing (0.52, 37%). This is also true for G3 where the **GST*** LSTM obtains an F1 score of 0.83 with an error percentage of 36% and outperforms the **GSTS** LSTM models for Suturing (0.73, 51%) and Needle Passing (0.75, 15%). In Figure 3b the Siamese-LSTM network also has higher F1 scores and lower error percentages for the **GST*** models for all gestures. We also calculated the error percentages for the **G*T*** model (baseline), which is 43% for the black lines in both figures. We observe that **GST*** models (green bars in Figure 3a and Figure 3b) sometimes achieve better performance than the baseline despite being trained on smaller datasets with lower error percentages (e.g., G1 and G3 in Figure 3a ; G1, G2 and G3 in Figure 3b).

> **Insight 3:** Increasing the amount of relevant training data from the same gesture class improves the performance more than adding more error examples.

Observation 4: Dual input Siamese networks perform better than their corresponding single input network structures with GST* training. We compared the performance among the four networks with **GST*** training. In Figure 3c, we observe that the average F1 scores for the Siamese networks are higher than their corresponding single networks. Siamese networks may perform better because of their double input structure that enables comparing and learning the difference between the two inputs. In addition, since the Siamese networks use pairs of inputs, the number of combinations of samples used for training is much greater than that of the single CNN or LSTM networks.
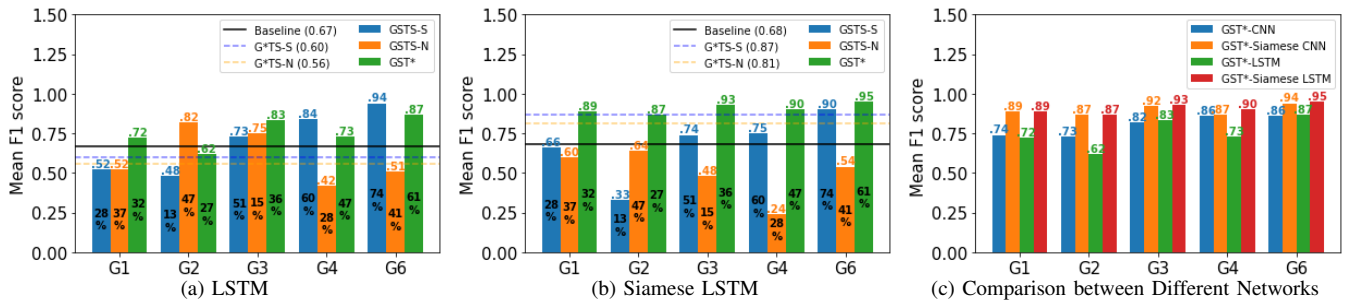
Fig. 3: Performance Comparison Across Different Networks and Training Setups. The values on top of each bar represent the F1 scores, and the percentage values at the middle of each bar represent the percentage of errors in the training data.

E. Gesture Nonspecific and Task Specific **G*TS** Models

In this experiment, we evaluated the performance of all the network architectures when trained using the gesture nonspecific and task specific (**G*TS**) setup and compared them to the **GSTS** and **GST*** models. To compare the overall performance of different training setups, we evaluated the error detection performance by calculating the micro average F1 scores which provide a general performance metric and do not consider the individual performance for specific gestures.

Observation 1: G*TS training improves performance compared to GSTS training. However, GST* models have better performance than G*TS and GSTS models even though the training dataset for G*TS training is larger. In Table II, **G*TS** training has a higher micro F1 score than **GSTS** training for the Suturing task except for the CNN network. Similarly, for the Needle Passing task, **G*TS** training has a higher micro F1 score than **GSTS** except for the LSTM network. This is most likely due to an increase in the number of training samples when all gestures are combined to train a single model for each task. However, **GST*** training leads to the best micro F1 score compared to **G*TS** and **GSTS** training for all networks. In this case, the amount of data for **GST*** training is smaller than the **G*TS** training as shown in the "Data" row in Table II. This could be because increasing the relevant data from the same gesture class helps improve the network performance in comparison with just increasing the amount of training data (similar to **Insight 2** from the previous experiment).

Observation 2: Dual input Siamese networks perform better than single networks with G*TS training. As shown in Table II, the Siamese-CNN has a greater micro F1 score than the CNN network for Suturing (0.79 vs. 0.72) and Needle Passing (0.74 vs. 0.56). The Siamese-LSTM also has a greater micro F1 score than the LSTM for Suturing (0.87 vs. 0.60) and Needle Passing (0.81 vs. 0.56).

> **Insight 4**: Dual input Siamese networks achieve better performance likely due to their dual input structure that enables comparing normal/erroneous and normal/normal pairs of trajectories and learning models that better differentiate erroneous trajectories

V. DISCUSSION AND CONCLUSION

This paper evaluated the performance of dual input Siamese networks versus single CNN and LSTM network architectures for the task of erroneous gesture detection. We compared different training setups to assess how various levels of contextual knowledge on gestures and tasks affect the performance of each network. Our results show that the dual Siamese networks achieve higher accuracy than their corresponding single CNN and LSTM networks in most cases using the **GST*** and **G*TS** training setups.

On the other hand, Siamese networks suffer from higher computational complexity than single networks. The single CNN networks take less computation time than LSTM networks across different training setups and gesture classes (avg. 0.50 ms vs. 1.26 ms). For both the Siamese-CNN and Siamese-LSTM networks, the computation time per window was proportional to the size of training data. This is because the Siamese networks pair each unknown window with all the known windows from the training data for voting and, thus, process more data at runtime. Also, Siamese-LSTM had much larger avg. computation time compared to Siamese-CNN across training setups and gestures (avg. 14.95 ms vs. 1.86 ms). This is because of hundreds of hidden states within the three LSTM layers that each must be computed sequentially. However, for the Siamese-CNN, outputs from the convolutional layers can be calculated in parallel on the GPU. Thus, for runtime error detection, using Siamese-CNN networks might be preferred and the parallel implementation of the Siamese models on GPU accelerators is required.

Our results also show that despite having fewer training samples, gesture specific models achieve better accuracy than generalized gesture nonspecific models. This further motivates the importance of developing models and labeled datasets that focus on gestures as the building blocks of procedures, their distributional similarities, and their unique error modes [1]. However, for runtime error detection using a gesture specific model, a gesture classifier is needed to first determine the gesture class for the input window [35]. The extra computation time needed for gesture classification might negatively impact the performance and timeliness of error detection [14]. In addition, the gesture classifier might misclassify a given gesture, leading to the execution of an error detector that does not correspond to the gesture.

Future work will focus on a more in-depth evaluation of the performance of the end-to-end gesture-specific error detection models, including the integration of the gesture classification and erroneous gesture detection models and analyzing the trade-offs between timeliness and accuracy.

REFERENCES

[1] K. Hutchinson, Z. Li, L. A. Cantrell, N. S. Schenkman, and H. Alemzadeh, "Analysis of executional and procedural errors in dry-lab robotic surgery experiments," *The International Journal of Medical Robotics and Computer Assisted Surgery*, vol. n/a, no. n/a, p. e2375, 2022. [Online]. Available: https://github.com/UVA-DSA/ExecProc_Error_Analysis

[2] O. Elhage, B. Challacombe, A. Shortland, and P. Dasgupta, "An assessment of the physical impact of complex surgical tasks on surgeon errors and discomfort: a comparison between robot-assisted, laparoscopic and open approaches," *BJU international*, vol. 115, no. 2, pp. 274–281, 2015.

[3] T. R. Eubanks, R. H. Clements, D. Pohl, N. Williams, D. C. Schaad, S. Horgan, and C. Pellegrini, "An objective scoring system for laparoscopic cholecystectomy," *Journal of the American College of Surgeons*, vol. 189, no. 6, pp. 566–574, 1999.

[4] P. Joice, G. Hanna, and A. Cuschieri, "Errors enacted during endoscopic surgery—a human reliability analysis," *Applied ergonomics*, vol. 29, no. 6, pp. 409–414, 1998.

[5] H. Alemzadeh, J. Raman, N. Leveson, Z. Kalbarczyk, and R. K. Iyer, "Adverse events in robotic surgery: a retrospective study of 14 years of fda data," *PloS one*, vol. 11, no. 4, p. e0151470, 2016.

[6] T. Bonaci, J. Herron, T. Yusuf, J. Yan, T. Kohno, and H. J. Chizeck, "To make a robot secure: An experimental analysis of cyber security threats against teleoperated surgical robots," *arXiv preprint arXiv:1504.04339*, 2015.

[7] T. Bonaci, J. Yan, J. Herron, T. Kohno, and H. J. Chizeck, "Experimental analysis of denial-of-service attacks on teleoperated robotic systems," in *Proceedings of the ACM/IEEE sixth international conference on cyber-physical systems*, 2015, pp. 11–20.

[8] H. Alemzadeh, D. Chen, X. Li, T. Kesavadas, Z. T. Kalbarczyk, and R. K. Iyer, "Targeted attacks on teleoperated surgical robots: Dynamic model-based detection and mitigation," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2016, pp. 395–406.

[9] H. Alemzadeh, D. Chen, A. Lewis, Z. Kalbarczyk, J. Raman, N. Leveson, and R. Iyer, "Systems-theoretic safety assessment of robotic telesurgical systems," in *International conference on computer safety, reliability, and security*. Springer, 2014, pp. 213–227.

[10] Z. Wang and A. Majewicz Fey, "Deep learning with convolutional neural network for objective skill evaluation in robot-assisted surgery." *International journal of computer assisted radiology and surgery*, vol. 13, no. 12, p. 1959–1970, Dec 2018.

[11] A. Zia and I. Essa, "Automated surgical skill assessment in rmis training." *International journal of computer assisted radiology and surgery*, vol. 13, no. 5, p. 731–739, May 2018.

[12] I. Funke, S. T. Mees, J. Weitz, and S. Speidel, "Video-based surgical skill assessment using 3d convolutional neural networks." *International journal of computer assisted radiology and surgery*, vol. 14, no. 7, p. 1217–1225, Jul 2019.

[13] L. Maier-Hein, S. S. Vedula, S. Speidel, N. Navab, R. Kikinis, A. Park, M. Eisenmann, H. Feussner, G. Forestier, S. Giannarou *et al.*, "Surgical data science for next-generation interventions," *Nature Biomedical Engineering*, vol. 1, no. 9, p. 691, 2017.

[14] M. S. Yasar and H. Alemzadeh, "Real-time context-aware detection of unsafe events in robot-assisted surgery," *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 385–397, 2020.

[15] M. S. Yasar, D. Evans, and H. Alemzadeh, "Context-aware monitoring in robotic surgery," *2019 International Symposium on Medical Robotics (ISMR)*, pp. 1–7, 2019.

[16] J. Yan, K. Huang, K. Lindgren, T. Bonaci, and H. Chizeck, "Continuous operator authentication for teleoperated systems using hidden markov models," *ArXiv*, vol. abs/2010.14006, 2020.

[17] C. He, N. Patel, I. Iordachita, and M. Kobilarov, "Enabling technology for safe robot-assisted retinal surgery: Early warning for unsafe scleral force," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3889–3894.

[18] Intuitive Surgical Inc., "The da vinci surgical system."

[19] X. Gao, Y. Jin, Q. Dou, and P.-A. Heng, "Automatic gesture recognition in robot-assisted surgery with reinforcement learning and tree search," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 8440–8446.

[20] B. van Amsterdam, M. Clarkson, and D. Stoyanov, "Multi-task recurrent neural network for surgical gesture recognition and progress prediction," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1380–1386, 2020.

[21] B. van Amsterdam, H. Nakawala, E. Momi, and D. Stoyanov, "Weakly supervised recognition of surgical gestures," *2019 International Conference on Robotics and Automation (ICRA)*, pp. 9565–9571, 2019.

[22] Y. Qin, S. A. Pedram, S. Feyzabadi, M. Allan, A. J. McLeod, J. W. Burdick, and M. Azizian, "Temporal segmentation of surgical subtasks through deep learning with multiple data sources," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 371–377.

[23] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, *Temporal convolutional networks: A unified approach to action segmentation*, ser. Lecture notes in computer science. Springer International Publishing, 2016, vol. 9915, p. 47–54.

[24] 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*, ser. Lecture notes in computer science. Springer International Publishing, 2016, vol. 9900, p. 551–558.

[25] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning," *The International journal of robotics research*, vol. 36, no. 13–14, p. 1595–1618, Dec 2017.

[26] L. Tao, E. Elhamifar, S. Khudanpur, G. D. Hager, and R. Vidal, *Sparse hidden markov models for surgical gesture classification and skill evaluation*, ser. Lecture notes in computer science. Springer Berlin Heidelberg, 2012, vol. 7330, p. 167–177.

[27] B. Varadarajan, C. Reiley, H. Lin, S. Khudanpur, and G. Hager, "Data-derived models for segmentation with application to surgical assessment and training." *Medical Image Computing and Computer-Assisted Intervention*, vol. 12, no. Pt 1, p. 426–434, 2009.

[28] Y. Gao, S. S. Vedula, C. E. Reiley, N. Ahmidi, B. Varadarajan, H. C. Lin, L. Tao, L. Zappella, B. Béjar, D. D. Yuh *et al.*, "Jhu-isi gesture and skill assessment working set (jigsaws): A surgical activity dataset for human motion modeling," vol. 3, p. 3, 2014.

[29] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[30] I. J. Good, "Rational decisions," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 14, no. 1, pp. 107–114, 1952. [Online]. Available: http://www.jstor.org/stable/2984087

[31] G. R. Koch, "Siamese neural networks for one-shot image recognition," *undefined*, 2015.

[32] S. Shahtalebi, A. Asif, and A. Mohammadi, "Siamese neural networks for eeg-based brain-computer interfaces." *Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual International Conference*, vol. 2020, p. 442–446, 2020.

[33] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, pp. 8026–8037, 2019.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] B. van Amsterdam, M. Clarkson, and D. Stoyanov, "Gesture recognition in robotic surgery: a review," *IEEE Transactions on Biomedical Engineering*, 2021.