Recognizing Surgical Activities with Recurrent Neural Networks

Robert DiPietro[1(✉)], Colin Lea[1], Anand Malpani[1], Narges Ahmidi[1], S. Swaroop Vedula[1], Gyusung I. Lee[2], Mija R. Lee[2], and Gregory D. Hager[1]

[1] Department of Computer Science, Johns Hopkins University, Baltimore, MD, USA
rdipietro@gmail.com

[2] Department of Surgery, Johns Hopkins University, Baltimore, MD, USA

Abstract. We apply recurrent neural networks to the task of recognizing surgical activities from robot kinematics. Prior work in this area focuses on recognizing short, low-level activities, or *gestures*, and has been based on variants of hidden Markov models and conditional random fields. In contrast, we work on recognizing both gestures and longer, higher-level activites, or *maneuvers*, and we model the mapping from kinematics to gestures/maneuvers with recurrent neural networks. To our knowledge, we are the first to apply recurrent neural networks to this task. Using a single model and a single set of hyperparameters, we match state-of-the-art performance for gesture recognition and advance state-of-the-art performance for maneuver recognition, in terms of both accuracy and edit distance. Code is available at https://github.com/rdipietro/miccai-2016-surgical-activity-rec.

1 Introduction

Automated surgical-activity recognition is a valuable precursor for higher-level goals such as objective surgical-skill assessment and for providing targeted feedback to trainees. Previous research on automated surgical-activity recognition has focused on gestures within a surgical task [9,10,13,15]. Gestures are atomic segments of activity that typically last for a few seconds, such as grasping a needle. In contrast, maneuvers are composed of a sequence of gestures and represent higher-level segments of activity, such as tying a knot. We believe that targeted feedback for maneuvers is meaningful and consistent with the subjective feedback that faculty surgeons currently provide to trainees.

Here we focus on jointly segmenting and classifying surgical activities. Other work in this area has focused on variants of hidden Markov models (HMMs) and conditional random fields (CRFs) [9,10,13,15]. HMM and CRF based methods often define unary (label-input) and pairwise (label-label) energy terms, and during inference find a global label configuration that minimizes overall energy. Here we put emphasis on the unary terms and note that defining unaries that are both general and meaningful is a difficult task. For example, of the works above, the unaries of [10] are perhaps most general: they are computed using

S. Ourselin et al. (Eds.): MICCAI 2016, Part I, LNCS 9900, pp. 551–558, 2016.
DOI: 10.1007/978-3-319-46720-7_64

Fig. 1. Example images from the JIGSAWS and MISTIC datasets.

learned convolutional filters. However, we note that even these unaries depend only on inputs from fairly local neighborhoods in time.

In this work, we use recurrent neural networks (RNNs), and in particular long short-term memory (LSTM), to map kinematics to labels. Rather than operating only on local neighborhoods in time, LSTM maintains a memory cell and *learns* when to write to memory, when to reset memory, and when to read from memory, forming unaries that in principle depend on *all* inputs. In fact, we will rely *only* on these unary terms, or in other words assume that labels are independent given the sequence of kinematics. Despite this, we will see that predicted labels are smooth over time with no post-processing. Further, using a single model and a single set of hyperparameters, we match state-of-the-art performance for gesture recognition and improve over state-of-the-art performance for maneuver recognition, in terms of both accuracy and edit distance.

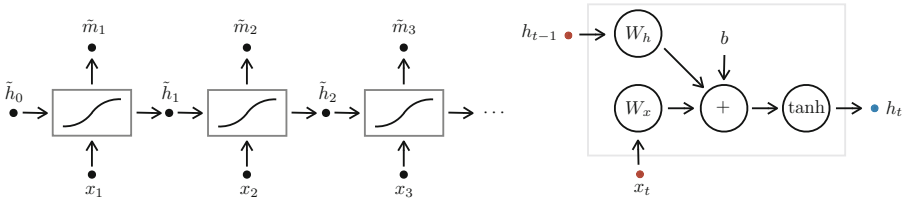
2 Methods

The goal of this work is to use n_x kinematic signals over time to label every time step with one of n_y surgical activities. An individual sequence of length T is composed of kinematic inputs $\{x_t\}$, with each $x_t \in \mathbb{R}^{n_x}$, and a collection of one-hot encoded activity labels $\{y_t\}$, with each $y_t \in \{0,1\}^{n_y}$. (For example, if we have classes 1, 2, and 3, then the one-hot encoding of label 2 is $(0,1,0)^T$.) We aim to learn a mapping from $\{x_t\}$ to $\{y_t\}$ in a supervised fashion that generalizes to users that were absent from the training set. In this work, we use recurrent neural networks to discriminatively model $p(y_t|x_1, x_2, \ldots, x_t)$ for all t when operating online and $p(y_t|x_1, x_2, \ldots, x_T)$ for all t when operating offline.

2.1 Recurrent Neural Networks

Though not yet as ubiquitous as their feedforward counterparts, RNNs have been applied successfully to many diverse sequence-modeling tasks, from text-to-handwriting generation [6] to machine translation [14].

A generic RNN is shown in Fig. 2a. An RNN maintains a hidden state $\tilde{h}_t$, and at each time step t, the nonlinear block uses the previous hidden state $\tilde{h}_{t-1}$ and the current input x_t to produce a new hidden state $\tilde{h}_t$ and an output $\tilde{m}_t$.



(a) A recurrent neural network. (b) A vanilla RNN block.

Fig. 2. A recurrent neural network.

If we use the nonlinear block shown in Fig. 2b, we end up with a specific and simple model: a vanilla RNN with one hidden layer. The recursive equation for a vanilla RNN, which can be read off precisely from Fig. 2b, is

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b) \tag{1}$$

Here, W_x, W_h, and b are free parameters that are shared over time. For the vanilla RNN, we have $\tilde{m}_t = \tilde{h}_t = h_t$. The height of h_t is a hyperparameter and is referred to as the number of hidden units.

In the case of multiclass classification, we use a linear layer to transform $\tilde{m}_t$ to appropriate size n_y and apply a softmax to obtain a vector of class probabilities:

$$\hat{y}_t = \text{softmax}(W_{ym}\tilde{m}_t + b_y) \tag{2}$$

$$p(y_{tk} = 1 \mid x_1, x_2, \ldots, x_t) = \hat{y}_{tk} \tag{3}$$

where $\text{softmax}(x) = \exp(x)/\sum_i \exp(x_i)$.

RNNs traditionally propagate information forward in time, forming predictions using only past and present inputs. Bidirectional RNNs [12] can improve performance when operating offline by using future inputs as well. This essentially consists of running one RNN in the forward direction and one RNN in the backward direction, concatenating hidden states, and computing outputs jointly.

2.2 Long Short-Term Memory

Vanilla RNNs are very difficult to train because of what is known as the *vanishing gradient problem* [1]. LSTM [8] was specifically designed to overcome this problem and has since become one of the most widely-used RNN architectures. The recursive equations for the LSTM block used in this work are

$$\tilde{x}_t = \tanh(W_{\tilde{x}x} x_t + W_{\tilde{x}m} m_{t-1} + b_{\tilde{x}}) \tag{4}$$

$$i_t = \sigma(W_{ix} x_t + W_{im} m_{t-1} + W_{ic} c_{t-1} + b_i) \tag{5}$$

$$f_t = \sigma(W_{fx} x_t + W_{fm} m_{t-1} + W_{fc} c_{t-1} + b_f) \tag{6}$$

$$c_t = i_t \odot \tilde{x}_t + f_t \odot c_{t-1} \tag{7}$$

$$o_t = \sigma(W_{ox} x_t + W_{om} m_{t-1} + W_{oc} c_t + b_o) \tag{8}$$

$$m_t = o_t \odot \tanh(c_t) \tag{9}$$

where $\odot$ represents element-wise multiplication and $\sigma(x) = 1/(1+\exp(-x))$. All matrices W and all biases b are free parameters that are shared across time.

LSTM maintains a memory over time and *learns* when to write to memory, when to reset memory, and when to read from memory [5]. In the context of the generic RNN, $\tilde{m}_t = m_t$, and $\tilde{h}_t$ is the concatenation of c_t and m_t. c_t is the *memory cell* and is updated at each time step to be a linear combination of $\tilde{x}_t$ and c_{t-1}, with proportions governed by the *input gate* i_t and the *forget gate* f_t. m_t, the output, is a nonlinear version of c_t that is filtered by the output gate o_t. Note that all elements of the gates i_t, f_t, and o_t lie between 0 and 1.

This version of LSTM, unlike the original, has forget gates and *peephole connections*, which let the input, forget, and output gates depend on the memory cell. Forget gates are a standard part of modern LSTM [7], and we include peephole connections because they have been found to improve performance when precise timing is required [4]. All weight matrices are full except the peephole matrices W_{ic}, W_{fc}, and W_{oc}, which by convention are restricted to be diagonal.

Loss. Because we assume every y_t is independent of all other $y_{t'}$ given $x_1, \ldots, x_t$, maximizing the log likelihood of our data is equivalent to minimizing the overall cross entropy between the true labels $\{y_t\}$ and the predicted labels $\{\hat{y}_t\}$. The global loss for an individual sequence is therefore

$$l_{\text{seq}}(\{y_t\}, \{\hat{y}_t\}) = \sum_t l_t(y_t, \hat{y}_t) \qquad \text{with} \qquad l_t(y_t, \hat{y}_t) = -\sum_k y_{tk} \log \hat{y}_{tk}$$

Training. All experiments in this paper use standard stochastic gradient descent to minimize loss. Although the loss is non-convex, it has repeatedly been observed empirically that ending up in a poor local optimum is unlikely. Gradients can be obtained efficiently using backpropagation [11]. In practice, one can build a computation graph out of fundamental operations, each with known local gradients, and then apply the chain rule to compute overall gradients with respect to all free parameters. Frameworks such as Theano and Google TensorFlow let the user specify these computation graphs symbolically and alleviate the user from computing overall gradients manually.

Once gradients are obtained for a particular free parameter p, we take a small step in the direction opposite to that of the gradient: with η being the learning rate,

$$p' = p - \eta \frac{\partial l_{\text{seq}}}{\partial p} \qquad \text{with} \qquad \frac{\partial l_{\text{seq}}}{\partial p} = \sum_t \frac{\partial l_t}{\partial p}$$

3 Experiments

3.1 Datasets

The JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS) [2] is a public benchmark surgical activity dataset recorded using the *da Vinci*. JIGSAWS contains synchronized video and kinematic data from a standard 4-throw

suturing task performed by eight subjects with varying skill levels. All subjects performed about 5 trials, resulting in a total of 39 trials. We use the same measurements and activity labels as the current state-of-the-art method [10]. Measurements are position (x, y, z), velocity (v_x, v_y, v_z), and gripper angle (θ) for each of the left and right slave manipulators, and the surgical activity at each time step is one of ten different gestures.

The Minimally Invasive Surgical Training and Innovation Center - Science of Learning (MISTIC-SL) dataset, also recorded using the *da Vinci*, includes 49 right-handed trials performed by 15 surgeons with varying skill levels. We follow [3] and use a subset of 39 right-handed trials for all experiments. All trials consist of a suture throw followed by a surgeon's knot, eight more suture throws, and another surgeon's knot. We used the same kinematic measurements as for JIGSAWS, and the surgical activity at each time step is one of 4 maneuvers: suture throw (ST), knot tying (KT), grasp pull run suture (GPRS), and inter-maneuver segment (IMS). It is not possible for us to release this dataset at this time, though we hope we will be able to release it in the future.

3.2 Experimental Setup

JIGSAWS has a standardized leave-one-user-out evaluation setup: for the i-th run, train using all users except i and test on user i. All results in this paper are averaged over the 8 runs, one per user. We follow the same strategy for MISTIC-SL, averaging over 11 runs, one for each user that does not appear in the validation set, as explained below.

We include accuracy and edit distance (Levenshtein distance) as performance metrics. Accuracy is the percentage of correctly-classified frames, measuring performance without taking temporal consistency into account. In contrast, edit distance is the number of operations needed to transform predicted segment-level labels into ground-truth segment-level labels, here normalized for each dataset using the maximum number (over all sequences) of segment-level labels.

3.3 Hyperparameter Selection and Training

Here we include the most relevant details regarding hyperparameter selection and training; other details are fully specified in code, available at https://github.com/rdipietro/miccai-2016-surgical-activity-rec.

For each run we train for a total of approximately 80 epochs, maintaining a learning rate of 1.0 for the first 40 epochs and then halving the learning rate every 5 epochs for the rest of training. Using a small batch size is important; we found that otherwise the lack of stochasticity let us converge to bad local optima. We use a batch size of 5 sequences for all experiments.

Because JIGSAWS has a fixed leave-one-user-out test setup, with all users appearing in the test set exactly once, it is not possible to use JIGSAWS for hyperparameter selection without inadvertently training on the test set. We therefore choose all hyperparameters using a small MISTIC-SL validation set consisting of 4 users (those with only one trial each), and we use the resulting hyperparameters for both JIGSAWS experiments and MISTIC-SL experiments.

We performed a grid search over the number of RNN hidden layers (1 or 2), the number of hidden units per layer (64, 128, 256, 512, or 1024), and whether dropout [16] is used (with $p = 0.5$). 1 hidden layer of 1024 units, with dropout, resulted in the lowest edit distance and simultaneously yielded high accuracy. These hyperparameters were used for all experiments.

Using a modern GPU, training takes about 1 h for any particular JIGSAWS run and about 10 h for any particular MISTIC-SL run (MISTIC-SL sequences are approximately 10x longer than JIGSAWS sequences). We note, however, that RNN inference is fast, with a running time that scales linearly with sequence length. At test time, it took the bidirectional RNN approximately 1 s of compute time per minute of sequence (300 time steps).

3.4 Results

Table 1 shows results for both JIGSAWS (gesture recognition) and MISTIC-SL (maneuver recognition). A forward LSTM and a bidirectional LSTM are compared to the Markov/semi-Markov conditional random field (MsM-CRF), Shared Discriminative Sparse Dictionary Learning (SDSDL), Skip-Chain CRF (SC-CRF), and Latent-Convolutional Skip-Chain CRF (LC-SC-CRF). We note that the LC-SC-CRF results were computed by the original author, using the same MISTIC-SL validation set for hyperparameter selection.

We include standard deviations where possible, though we note that they largely describe the user-to-user variations in the datasets. (Some users are exceptionally challenging, regardless of the method.) We also carried out statistical-significance testing using a paired-sample permutation test (p-value of 0.05). This test suggests that the accuracy and edit-distance differences between the bidirectional LSTM and LC-SC-CRF are insignificant in the case of JIGSAWS but are significant in the case of MISTIC-SL. We also remark that even the forward LSTM is competitive here, despite being the only algorithm that can run online.

Qualitative results are shown in Fig. 3 for the trials with highest, median, and lowest accuracies for each dataset. We note that the predicted label sequences are smooth, despite the fact that we assumed that labels are independent given the sequence of kinematics.

Table 1. Quantitative results and comparisons to prior work.

	JIGSAWS		MISTIC-SL	
	Accuracy (%)	Edit dist. (%)	Accuracy (%)	Edit dist. (%)
MsM-CRF [15]	72.6	—	—	—
SDSDL [13]	78.7	—	—	—
SC-CRF [9]	80.3	—	—	—
LC-SC-CRF [10]	82.5 ± 5.4	14.8 ± 9.4	81.7 ± 6.2	29.7 ± 6.8
Forward LSTM	80.5 ± 6.2	19.8 ± 8.7	87.8 ± 3.7	33.9 ± 13.3
Bidir. LSTM	**83.3 ± 5.7**	**14.6 ± 9.6**	**89.5 ± 4.0**	**19.5 ± 5.2**

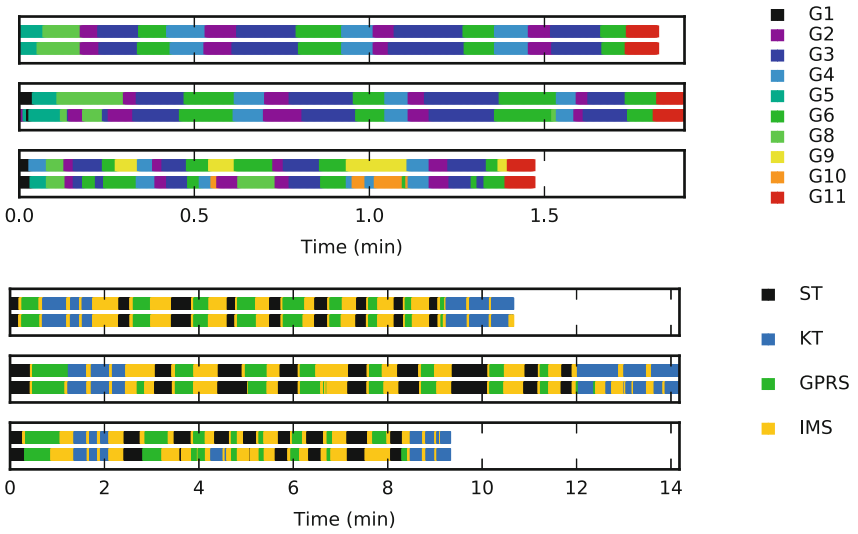


Fig. 3. Qualitative results for JIGSAWS (top) and MISTIC-SL (bottom) using a bidirectional LSTM. For each dataset, we show results from the trials with highest accuracy (top), median accuracy (middle), and lowest accuracy (bottom). In all cases, ground truth is displayed above predictions.

4 Summary

In this work we performed joint segmentation and classification of surgical activities from robot kinematics. Unlike prior work, we focused on high-level maneuver prediction in addition to low-level gesture prediction, and we modeled the mapping from inputs to labels with recurrent neural networks instead of with HMM or CRF based methods. Using a single model and a single set of hyperparameters, we matched state-of-the-art performance for JIGSAWS (gesture recognition) and advanced state-of-the-art performance for MISTIC-SL (maneuver recognition), in the latter case increasing accuracy from 81.7 % to 89.5 % and decreasing normalized edit distance from 29.7 % to 19.5 %.

References

1. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**(2), 157–166 (1994)
2. Gao, Y., Vedula, S.S., Reiley, C.E., Ahmidi, N., Varadarajan, B., Lin, H.C., Tao, L., Zappella, L., Bejar, B., Yuh, D.D., Chen, C.C.G., Vidal, R., Khudanpur, S., Hager, G.D.: Language of surgery: a surgical gesture dataset for human motion modeling. In: Modeling and Monitoring of Computer Assisted Interventions (M2CAI) 2014. Springer, Boston, USA (2014)
3. Gao, Y., Vedula, S., Lee, G.I., Lee, M.R., Khudanpur, S., Hager, G.D.: Unsupervised surgical data alignment with application to automatic activity annotation. In: 2016 IEEE International Conference on Robotics and Automation (ICRA) (2016)

4. Gers, F.A., Schmidhuber, J.: Recurrent nets that time and count. In: IEEE Conference on Neural Networks, vol. 3 (2000)
5. Graves, A.: Supervised Sequence Labelling. Springer, Heidelberg (2012)
6. Graves, A.: Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 (2013)
7. Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: LSTM: A search space odyssey. arXiv preprint arXiv:1503.04069 (2015)
8. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
9. Lea, C., Hager, G.D., Vidal, R.: An improved model for segmentation and recognition of fine-grained activities with application to surgical training tasks. In: 2015 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1123–1129. IEEE (2015)
10. Lea, C., Vidal, R., Hager, G.D.: Learning convolutional action primitives for fine-grained action recognition. In: 2016 IEEE International Conference on Robotics and Automation (ICRA) (2016)
11. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Cogn. Model. **5**(3), 1 (1988)
12. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Trans. Sig. Process. **45**(11), 2673–2681 (1997)
13. Sefati, S., Cowan, N.J., Vidal, R.: Learning shared, discriminative dictionaries for surgical gesture segmentation and classification. In: Modeling and Monitoring of Computer Assisted Interventions (M2CAI) 2015. Springer, Heidelberg (2015)
14. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems (2014)
15. Tao, L., Zappella, L., Hager, G.D., Vidal, R.: Surgical gesture segmentation and recognition. In: Mori, K., Sakuma, I., Sato, Y., Barillot, C., Navab, N. (eds.) MICCAI 2013, Part III. LNCS, vol. 8151, pp. 339–346. Springer, Heidelberg (2013)
16. Zaremba, W., Sutskever, I., Vinyals, O.: Recurrent neural network regularization. arXiv preprint arXiv:1409.2329 (2014)