Recognition of Dynamic Hand Gestures from 3D Motion Data using LSTM and CNN architectures

Chinmaya R. Naguri
School of EECS
Ohio University
Athens, OH 45701
Email: cn262114@ohio.edu

Razvan C. Bunescu
School of EECS
Ohio University
Athens, OH 45701
Email: bunescu@ohio.edu

***Abstract*—Hand gestures provide a natural, non-verbal form of communication that can augment or replace other communication modalities such as speech or writing. Along with voice commands, hand gestures are becoming the primary means of interaction in games, augmented reality, and virtual reality platforms. Recognition accuracy, flexibility, and computational cost are some of the primary factors that can impact the incorporation of hand gestures in these new technologies, as well as their subsequent retrieval from multimodal corpora. In this paper, we present fast and highly accurate gesture recognition systems based on long short-term memory (LSTM) and convolutional neural networks (CNN) that are trained to process input sequences of 3D hand positions and velocities acquired from infrared sensors. When evaluated on real time recognition of six types of hand gestures, the proposed architectures obtain 97% F-measure, demonstrating a significant potential for practical applications in novel human-computer interfaces.**

I. Introduction and Motivation

Gestures over a contact surface are common means of interaction with touch enabled devices, such as laptops and tablets. Touchless gestures have been recently incorporated for basic communication and control in gaming consoles and new technologies such as Augmented Reality and Holographic displays. While the usability of commercial human-computer interfaces (HCI) has improved considerably over the past decades, incorporating gestures in mainstream applications is still a non-trivial task, due to the complexity of the recognition algorithms, their uneven performance, and lack of flexibility in terms of types of gestures. A substantial amount of research has been done in this area using machine learning on traditional video and depth camera input, as in [1], [2], [3], [4].

In this paper, we present a gesture recognition system that processes hand-skeletal information provided by a Leap Motion (LM) controller. Using three Infra-Red (IR) lamps and two IR sensors, the controller can read 3D hand movements with high resolution and precision – over 100 frames/second and a precision of up to 100th of a millimeter. Compared to other types of sensors and modalities, the LM controller has been scarcely explored in research on dynamic gesture recognition. Yuanrong et al [5] use a Support Vector Machine (SVM) classifier to recognize numbers written in air, based on features such as amplitude of the variation of orientation angles. Nigam et al. [6] propose a biometric authentication method that uses SVMs to classify the hand-skeletal information associated with a 3D hand signature pattern. A variant of Histogram of Oriented Optical Flows (HOOF) generates features that track the 3D movements of the hand, whereas Histogram of Oriented Trajectories (HOT) features encode the displacement of the action in 3D space. In the sign language recognition system from [7], measurements from multiple Leap sensors are first aligned and fused using Kalman filtering, and then classified with combinatorial Hidden Markov Models (HMM). The approaches mentioned above use manually engineered features to identify discriminative patterns in the raw 3D motion data. In contrast, a deep learning approach would induce high-level representations from the input sensory data automatically. We follow this paradigm and propose a gesture recognition approach that does not require sophisticated feature engineering. Instead, the time series of 3D positions and velocities are provided as input either directly to a Long Short-Term Memory (LSTM) network, or indirectly to a Convolutional Neural Network (CNN), through filters learned by a denoising auto-encoder.

II. Gesture Recognition Dataset

We created a dataset of hand gestures, represented as multiple time series of frames recorded by a Leap Motion (LM) controller. The controller tracks the user's hand movements in its visible space and computes vectors for the position and velocity for each part of the hand. We used the Leap Motion SDK to access and saved this information in real time. In this work, we only use the 3D positions and velocities of the fingers, which means that a frame contains 5 * (3 + 3) = 30 floating point numbers. We performed multiple (97) recordings of sequences of gestures, where each recording contains on average 6 gestures. To enable a manual segmentation and labeling of the frames with the corresponding gesture labels, we also synchronously recorded the gestures on video using a webcam. During the manual annotation process, the human annotator watches the video recording and marks the timestamps for the start and end of each gesture. These timestamps are automatically mapped to the corresponding Leap Motion frames. All the frames between the start and end frames are marked as *inside* frames and the entire segment is annotated with the corresponding gesture label. We defined a vocabulary of six types of interaction gestures: *Circle, Swipe, Correct,*

TABLE I
DATASET STATISTICS.

Circle	Swipe	Correct	Wrong	Pull	Push	Trans
97	100	99	98	98	98	415

Wrong, Pull, and *Push*. Additionally, we also annotated *Transition* gestures, which capture the quick movement of the hand in between two interaction gestures. For example, after doing a *Swipe* from left to right, most users tend to move their hand back to a center position in space which will be more comfortable for making a subsequent *Correct* (tick or check mark) gesture. Correspondingly, the movement of the hand between the two gestures is marked as a transition gesture, named so because it is not meant to trigger an actual interaction with the computer.

Table I shows, for each gesture type, the total number of gestures that were recorded and manually annotated. These gestures were the result of 97 recording sessions, where each session contained on average 1,812 frames, accounting for a sequence of 6 gestures on average.

III. System Architecture

As shown in Figure 1, the Gesture Recognition (GR) task is modeled as a pipeline of two major modules: Gesture Detection (GD) and Gesture Classification (GC). The hand movements are sensed by the Leap Motion controller, which generates a time series of frames containing 3D positions and velocities of the five fingers. The Gesture Detection module (Section III-A) determines whether a frame appears *inside* a gesture (marked with yellow) or *outside* a gesture. The sequence of labeled frames is then passed to a post-processing module (Section III-A1) that joins groups of inside frames that are sufficiently close to each other into larger groups of inside frames, further extended with a constant number of frames to the left and right. The resulting candidate gesture segments are passed to the Gesture Classification module (Section III-B), which classifies each of them either as one of a set of predefined gesture types or as an unknown gesture.

A. Gesture Detection

The Gesture Detection (GD) module, shown in Figure 2, processes time-series of frames from the LM controller in order to identify subsequences of frames that correspond to one of the predefined types of gestures. This is achieved by casting gesture detection as a sequence tagging problem, where frames inside a gesture are tagged as *Inside*, while frames outside a gesture are tagged as *Outside*. At each time step, a frame is provided as input to a Recurrent Neural Network (RNN), implemented using Long Short-Term Memory (LSTM) units [8] in order to avoid the vanishing gradient problem of traditional RNNs [9]. For each frame, the LSTM cell takes three types of inputs: the current frame (finger positions and velocities), the previous LSTM cell output and the content of the memory cell from the previous time step. The output of the

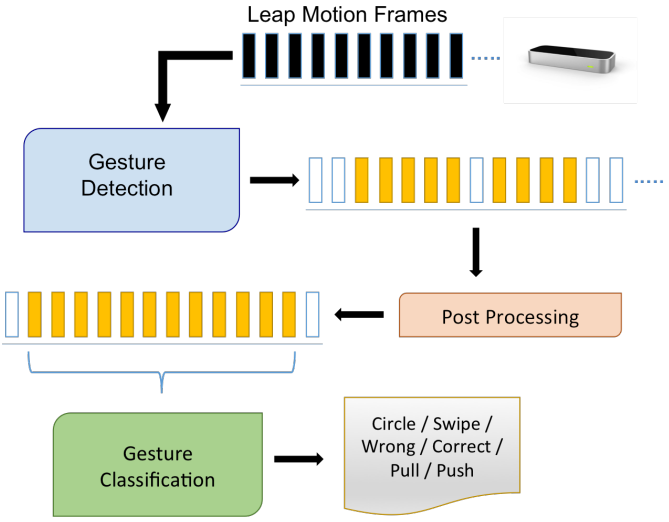

Fig. 1. High level system architecture.

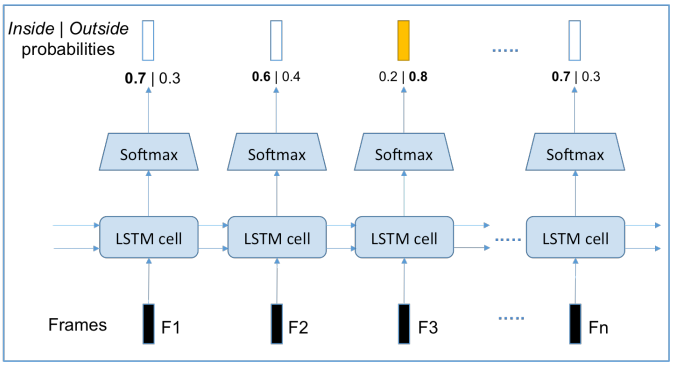

Fig. 2. Gesture detection architecture.

LSTM cell at each time step is fed into a logistic regression model that classifies the frame as either *inside* or *outside* a gesture. An important aspect to mention here is the use of *lookahead* frames. The frames at the beginning and the end of a gesture would be hard to classify, even for a human, without knowing the next few frames. Consequently, we shifted the labels of the frames l positions into the future, which means that whenever the GD module outputs a label at time step k, this corresponds to the frame l time steps into the past, i.e. at position $k - l$. For our experiments, we chose $l = 15$ frames, which is a quarter of the length of the fastest gesture in the dataset (a *Swipe*). If used in realtime interactions, this would correspond to a delay of approximately 15 * 10 = 125 milliseconds, which is negligible in terms of lag perception.

1) Post-processing: The trained GD model often labels a small number of frames in the middle of a gesture as *outside* the gesture, as shown in the second gesture from the graph on the left in Figure 3. Such gaps lead to fragmentation of the gestures, which is detrimental to the overall gesture recognition performance. Figure 3 also shows that, despite the use of *lookahead* frames, there are still cases where frames at the beginning or the end of a gesture are not detected. Since

recall is more important than precision at this stage of the recognition pipeline, we applied a sequence of three automatic post-processing steps: (1) The first step joins any two groups of frames marked as inside frames if they are separated by at most 10 frames, with the value 10 chosen by inspecting the system output on a development set; (2) The second step marks any inside regions that are less than 20 frames in length as outside (i.e. ignore); (3) The third step adds 10 frames on either side of each segment extracted by the GD module. The graph on the right in Figure 3 shows the result of post-processing the original graph on the left.

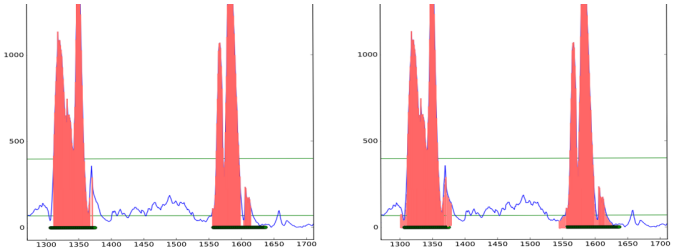

Fig. 3. Post-processing example. The hand speed (blue) vs. frame graph on the left shows the frames detected as *inside* by the GD module (red). The graph on the right shows the result of consolidating regions in post-processing. The black lines at the bottom show the extent of the true gestures.

B. Gesture Classification

The gesture segments identified and post-processed by the GD module are classified by the Gesture Classification (GC) module into one of the predefined types of gestures, using a Softmax model. Since the input segments have a variable number of frames and the Softmax requires a fixed size input, we use a mean-pooling layer on top of the outputs of either an LSTM network (Figure 4) or a Convolutional Neural Network (CNN) filter layer (Figure 5). For the LSTM version, we used the same configuration as in the GD module. However, instead of classifying each LSTM output vector separately, the outputs are mean pooled and fed into one Softmax model. For the CNN version, the filters were trained separately using a denoising autoencoder [10]. Furthermore, the input sequence of frames was partitioned into 4 equally sized regions and the CNN filter outputs from each region were mean-pooled before being sent to the Softmax model.

IV. Experimental Evaluation

The Gesture Detection (GD) and Gesture Classification (GC) modules are trained independently, using a k-fold cross-validation scheme. The dataset of 97 recordings is shuffled and split into 10 folds. One fold is set aside for validation, while the remaining 9 folds are used for training and testing in a 9-fold cross-validation scheme in which one fold is used for testing and the remaining 8 folds are used for training. This is repeated 9 times, so that each fold gets to be used as test data. To compute overall evaluation measures, the test results are pooled together over the 9 folds. We first evaluate the GD and GC models independently, using the same training and test folds for both – this ensures that the test data is not seen

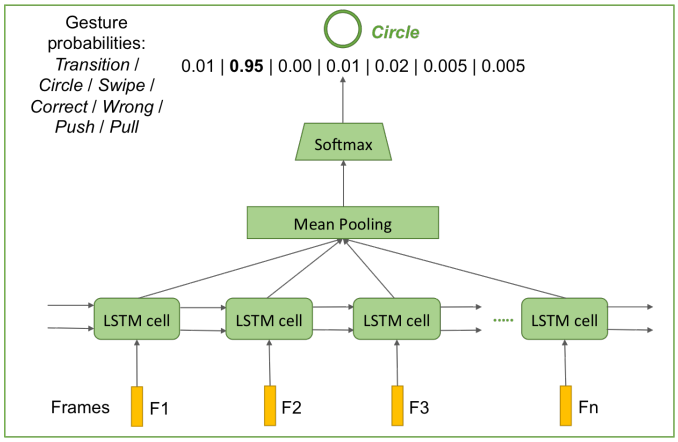

Fig. 4. Gesture classification architecture, using LSTM.

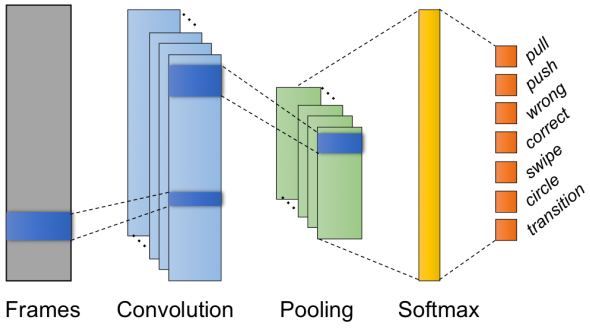

Fig. 5. Gesture classification architecture, using CNN.

during training by either GD or GC. When training and testing the GC module on its own, we use the true gesture boundaries.

Both GD and GC models are trained to minimize the cross-entropy cost function using Adadelta [11], with one gradient update per frame (GD) or gesture (GC). The models are trained until an early stopping criterion is satisfied, but not more than a maximum 600 epochs. The generalization performance is tracked on the validation data and early stopping is done when this performance is seen to degrade.

Because recordings are independent of each other, there is no continuation of hand movement between two recording. Consequently, at the start of each recording, we reset the previous cell state and previous LSTM output for the first LSTM in the sequence. The dimension of memory cells in the LSTM is set to be same as the input frame vector (30). Orthogonal vectors, derived from random numbers using singular value decomposition, are used to initialize the parameters for the LSTM input, forget and output gates. Saxe et al. [12] showed that this initialization technique leads to faster convergence during training. The parameters for the Softmax layer are initialized with random values from a univariate standard Gaussian distribution and L2 regularized using a weight decay of 1e-4. For the CNN version of GC, the filters were trained for a maximum of 1,000 epochs on 10,000 unlabeled segments of

20 frames, sampled at random from the LM controller output, using a denoising autoencoder with 0.2 noise probability.

The GD results are shown in Table II, before and after post-processing. Precision, recall, and F_1-measure are computed with respect to the *inside* frames, which account for 26% of the total frames. As expected, the post-processing step improves the recall substantially, with only a small decrease in precision. Table III shows the GC accuracy for both the LSTM and CNN implementations. Since the LSTM version performed better, we use it in the final gesture recognition pipeline. The bottom section of Table III also shows the performance of the LSTM model for each gesture type.

TABLE II
GESTURE DETECTION (GD) RESULTS (%), WITH AND WITHOUT POST-PROCESSING (PP).

	Precision	Recall	F_1-measure	Accuracy
GD	84.8	67.9	75.4	88.2
GD+PP	81.1	86.8	83.8	91.5

TABLE III
GESTURE CLASSIFICATION (GC) RESULTS (%): PRECISION (P), RECALL (R), F_1-MEASURE (F_1), AND ACCURACY.

CNN Overall Accuracy = 97.2% LSTM Overall Accuracy = 98.4%							
LSTM	Circle	Swipe	Correct	Wrong	Pull	Push	Trans
P	100	98.9	98.9	100	98.8	98.9	97.3
R	98.9	96.7	98.9	98.8	95.5	97.7	99.5
F_1	99.4	97.8	98.9	99.4	97.1	98.3	98.4

We evaluate the entire Gesture Recognition (GR) pipeline by running the trained GC module on the output of the GD module on each test fold as follows: the GD module is trained on the 8 training folds; the GC module is trained on the same 8 folds using the true gesture boundaries; the GD module is run on the test fold; the detected gestures are post-processed and used as test input for the GC module. A gesture recognized by the system is considered correct only if the following two conditions are satisfied: 1) at least half of the frames from the system gesture are contained in a true gesture; and 2) the label assigned by the system is the same as the label of the true gesture. The overall gesture recognition results are shown in Table IV. The system is fast enough to be used in real time, without any noticeable lag between the end of gesture and the corresponding classification decision. The dataset, the code, and a demo will be made publicly available[1].

TABLE IV
GESTURE RECOGNITION (GR) PIPELINE RESULTS (%).

Precision	Recall	F_1-measure
95.9	98.1	97.0

V. CONCLUSIONS AND FUTURE WORK

We presented a gesture recognition system based on long short-term memory (LSTM) networks. The architecture is trained on raw input sequences of 3D hand positions and velocities computed by a Leap Motion controller from infrared sensing data, which obviates the need for manual feature engineering. The system is highly accurate (97% F-measure) and fast enough to do real time recognition without noticeable lags. As such, it holds the promise for practical applications in novel HCI platforms.

In future work, we plan to evaluate the system on a larger set of gesture types, collected from multiple users. Such a dataset will also enable quantifying how well a system trained for one user transfers to another user. A practical deployment of our approach would also benefit from allowing the user to label a misclassified gesture and provide it as an online training example, in a never-ending learning version of the system.

[1] http://xxx.xx.xxxx.xxx

REFERENCES

[1] B. Ionescu, D. Coquin, P. Lambert, and V. Buzuloiu, "Dynamic hand gesture recognition using the skeleton of the hand," *EURASIP Journal on Advances in Signal Processing*, vol. 2005, no. 13, pp. 2101–2109, 2005.

[2] H. Francke, J. Ruiz-del Solar, and R. Verschae, "Real-time hand gesture detection and recognition using boosted classifiers and active learning," in *Pacific-Rim Symposium on Image and Video Technology*, Santiago, Chile, 2007, pp. 533–547.

[3] N. H. A.-Q. Dardas, "Real-time hand gesture detection and recognition for human computer interaction," Ph.D. dissertation, Université d'Ottawa/University of Ottawa, 2012.

[4] Y. Yin, "Real-time continuous gesture recognition for natural multimodal interaction," Ph.D. dissertation, MIT, 2014.

[5] Y. Xu, Q. Wang, X. Bai, Y.-L. Chen, and X. Wu, "A novel feature extracting method for dynamic gesture recognition based on Support Vector Machine," in *IEEE International Conference on Image Processing*. Paris, France: IEEE, October 2014, pp. 437–441.

[6] I. Nigam, M. Vatsa, and R. Singh, "Leap signature recognition using hoof and hot features," in *IEEE International Conference on Image Processing*. Paris, France: IEEE, October 2014, pp. 5012–5016.

[7] K.-Y. Fok, N. Ganganath, C.-T. Cheng, and C. K. Tse, "A real-time asl recognition system using leap motion sensors," in *Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. Xi'an, China: IEEE, September 2015, pp. 411–414.

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

[9] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[10] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *International Conference on Machine Learning*. New York, NY, USA: ACM, 2008, pp. 1096–1103.

[11] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012. [Online]. Available: http://arxiv.org/abs/1212.5701

[12] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *CoRR*, vol. abs/1312.6120, 2013. [Online]. Available: https://arxiv.org/abs/1312.6120