Complex Activity Recognition using Deep Convolutional Bidirectional LSTM

Swapnil Sayan Saha
University of California, Los Angeles
Los Angeles, CA
swapnilsayan@g.ucla.edu

Abstract

The goal of the "Cooking Activity Recognition Challenge" is to classify three macro human activities and the constituent isolated sequences of micro-activities (multi-label classification) from inertial and motion capture sensors with varying sampling rates and missing data. In this paper, we benchmark the performance of deep convolutional bidirectional long short-term memory (DCBL) architecture, along with LSTM and convolutional neural network (CNN) for complex activity recognition with missing, misaligned and differently sampled data. For macro-activities, DCBL achieved the highest validation accuracy of 80.1% while for micro-activities, the same architecture achieved 50.3% validation accuracy, outperforming vanilla CNN and LSTM. The low classification accuracy in the later case can be attributed to a large portion of essential event windows lacking sample points and appropriate lengths, as well as the training dataset being too small for multi-label micro-activity training.

1. Introduction

Human activity recognition (HAR) is defined as the process of correlating a sequence of granular human action primitives with similar data from a datastore [1]. The advent of smartphones and wearables has enabled a wide application spectrum for HAR including healthcare monitoring, human computer interaction, fitness tracking and transportation mode recognition [1][2][3]. However, challenges of deploying learning-enabled HAR systems in the wild include dealing with data from multiple noisy sensors with variable sampling rates, misaligned timestamps and missing data [3]. Traditional machine learning approaches are unable to deal with abnormal data on their own, requiring handcrafted features and domain knowledge [2][3]. Furthermore, complex activities may be composed of succession of simple activities, whose spatial context might alter with time [4].

The Cooking Activity Recognition Challenge [5] embraces the aforementioned challenges, with the goal of building a classifier to classify 3 distinct macro and constituent 10 distinct micro-activities (some of which may be common to each macro-activity) as follows:

- Making a sandwich - cut, wash, take, put, other
- Preparing fruitsalad - cut, take, peel, add, mix, put, other
- Preparing cereal - cut, take, pour, peel, put, open, other

The training dataset consists of 30-second windows from 3 subjects in 288 data frames. Sensors include 4 triaxial accelerometers placed at left wrist, right wrist, right arm and left hip and 29 triaxial motion capture (mo-cap) markers placed at random locations of the body, totaling 99 available sensor channels. Each frame corresponds to a single macro-activity, with one-to-multiple possible micro-activities. Mo-cap samples at approximately 100 Hz while the accelerometers sample between 10 - 100Hz at different timestamps. The starting and terminal timestamps for each micro-activity are absent and a particular frame may or may not contain all constituent micro-activities.

For our experiments, we chose to omit the 3 left-wrist channels due to wrongly annonated timestamps and high frequency of missing data in all files. Furthermore, mo-cap supplies absolute position rather than motion signatures unique to each action primitive, providing negligible differential features across action primitives as the data was collected under the same physical setup. Hence, we also omitted the mo-cap data, using 12 accelerometer channels to train our model. Wearable sensors (e.g. accelerometers) are state-of-the-art for human activity recognition [6].

1.1. Convolutional Neural Network

CNNs are able to extract differential patterns in activity windows regardless of precise location by enforcing a degree of local connectivity among adjacent inertial samples and enabling scale invariance (account for noisy feature discrepancy among same activity class), while reducing the number of parameters via weight sharing, making the architecture suitable as a high-dimensional feature extractor and classifier for inertial complex activity recognition. [2][7][8][9][10][11]. Table 4 illustrates the architecture of the implemented CNN. Each 2D convolutional layer contains 128 hidden units, with the kernel size set to (1, 10) with a stride of (1, 1) to ensure maximal overlapping (and correlation) among adjacent inertial samples, ensuring effective feature extraction. In order to smooth the objective functions to account for noisy and unpredictable multimodal data and prevent the the hidden units from being affected by stochastic abnormal deviations in gesture signatures, the convolutional layers are followed by batch normalization layers. 7 convolutional layers are integrated to ensure the network has sufficient capacity to extract salient insights from noisy, misaligned and missing data. The max-pool layers reduce the feature-dimensionalities to feed an abstracted representation of the feature space to subsequent layers. To improve generalization performance across various subjects such that the network ignores fine-grained subject-dependent motion signatures, we include dropout layers in the CNN, preventing overfitting. Finally, the output from convolutional layers are flattened and fed to a fully-connected layer (with 128 hidden units) to map the features to the classes. The activation function is ReLu (state-of-the-art) for convolutional layers, softmax for dense layer during macro-activity classification and ReLu for dense layer during micro-activity classification to account for multiple labels.

1.2. LSTM Recurrent Neural Network

In order to infer temporal dependencies of a sequence of common integrant micro-activities across several macro-activities (e.g. ”pouring” may arrive after ”taking” in cereal preparation but ”peeling” may arrive after ”taking” in fruit-salad preparation) and translate them differently, it is necessary to store perceived windows, which can be achieved via LSTM RNN [2][7][10]. LSTMs can handle variable delays between timestamps and events commonly encountered in missing data and multimodal data with variable sampling rates. The architecture of implemented LSTM RNN is shown in Table 3, consisting of 2X LSTM-Dropout layers followed by a fully connected layer. The LSTM layers learn dynamics of the time-series data for each sensor channel and utilizes long-term contexual information from previous states to infer the feature space of the current inertial window.

1.3. Deep Convolutional Bidirectional LSTM

A special class of LSTM is bidirectional LSTM, which can ’look into’ both the past and the future [10], increasing the amount of contextual information available to the network. DCBL is an amalgam of CNN and bidirectional LSTM, capable of extracting patterns while learning temporal dynamics of feature activations, offering more tightly bound decision boundaries around analogous activities while exploiting temporal characteristics for classification of macroactivities via constituent microactivities [2][7]. Table 5 illustrates the architecture of proposed DCBL. The initial layers resemble the CNN architecture except at the end, where a bidirectional LSTM layer maps the highly-correlated spatial features extracted by the convolutional layers to the activity classes using recurrent information from both the past and future states. The bidirectional LSTM layer is added at the end instead of the beginning to prevent long-time windows from causing difficulties in learning for the recurrent layer [12], while at the same time, ensuring that the feature map activations are highly correlated across the samples from a large event duration.

2. Results

Table 1 shows the classification performances of the three implemented architectures, with validation accuracy being the performance metric. DCBL, with a window size of 10 seconds and 500 samples per window, achieved 80.1% validation accuracy, outperforming vanilla CNN and LSTM. The learning curve for this classifier is shown in Figure 1.

The same architecture also achieved the highest validation accuracy for multi-label micro-activity classification, achieving classification accuracy of 50.3%. Performance of micro-activity classifiers are shown in Table 2, with validation mean-square error loss being the more appropriate metric to determine the best model.

3. Discussion

For macro-activity classification, vanilla LSTMs perform poorly as they are unable to extract spatially-correlated temporal features over long time-windows. However, since a macro-activity is composed of distinct constituent micro-activities, smaller time-windows may not provide sufficient discriminatory features to distinguish between each activities. This trend is evident for CNNs and DCBLs, where increasing the time-window increases the classification performance. DCBL provides better generalization over CNN (although macro-activity classification can largely be attributed to learning transient features over the latent space) by taking into account temporal insights of constituent microactivities apart from spatial features, which may not be similar for each macro-activity data frame due to distinct constituent micro-activity.

We observed that a large portion of the windows had zero (or negligible) samples, with wrongly annotated timestamps. For a dataset with only 288 frames, this is insufficient to train neural networks to contextualize state-based events, causing poor accuracy for micro-activity classification across all classifiers. Furthermore, the length of constituent micro-activities often totalled to the length of macro-activity, further causing performance drops.

References

[1] S. S. Saha, S. Rahman, M. J. Rasna, A. K. M. Mahfuzul Islam and M. A. Rahman Ahad, *DU-MD: An Open-Source Human Action Dataset for Ubiquitous Wearable Sensors,* 2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Kitakyushu, Japan, 2018, pp. 567-572.

[2] Jeyakumar, Jeya Vikranth, et al. *Deep convolutional bidirectional LSTM based transportation mode recognition.* Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers. 2018.

[3] T. Hossain and S. Inoue, *A Comparative Study on Missing Data Handling Using Machine Learning for Human Activity Recognition,* 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), Spokane, WA, USA, 2019, pp. 124-129.

[4] T. Xing et al., *DeepCEP: Deep Complex Event Processing Using Distributed Multimodal Information,* 2019 IEEE International Conference on Smart Computing (SMARTCOMP), Washington, DC, USA, 2019, pp. 87-92.

[5] *Cooking Activity Recognition Challenge*. 2020, `abc-research.github.io/cook2020/`. Accessed 15 Mar 2020.

[6] Jordao, Artur, et al. *Human activity recognition based on wearable sensor data: A standardization of the state-of-the-art.* arXiv preprint arXiv:1806.05226 (2018).

[7] Ordóñez, Francisco Javier, and Daniel Roggen. *Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition.* Sensors 16.1 (2016): 115.

[8] S. Ha, J. Yun and S. Choi, *Multi-modal Convolutional Neural Networks for Activity Recognition,* 2015 IEEE International Conference on Systems, Man, and Cybernetics, Kowloon, 2015, pp. 3017-3022.

[9] M. Panwar et al., *CNN based approach for activity recognition using a wrist-worn accelerometer,* 2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Seogwipo, 2017, pp. 2438-2441.

[10] Nils Y. Hammerla, Shane Halloran, and Thomas Plötz. 2016. *Deep, convolutional, and recurrent models for human activity recognition using wearables.* In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16). AAAI Press, 1533–1540.

[11] M. Zeng et al., *Convolutional Neural Networks for human activity recognition using mobile sensors,* 6th International Conference on Mobile Computing, Applications and Services, Austin, TX, 2014, pp. 197-205.

[12] Graves, Alex, and Jurgen Schmidhuber. *Framewise phoneme classification with bidirectional LSTM and other neural network architectures.* Neural networks 18.5-6 (2005): 602-610.

A. Performance Summary

Table 1: Macro-activity training and validation accuracies for implemented architectures.

Classifier	Window Size	Samples/Window	Validation Accuracy	Training Accuracy
DCBL	3 sec	150	0.74	0.95
	6 sec (norm. enabled)	300	0.77	0.95
	6 sec	300	0.78	0.96
	10 sec	**500**	**0.80**	**0.96**
CNN	3 sec	150	0.71	0.85
	6 sec	300	0.72	0.95
	10 sec	500	0.71	0.95
LSTM	6 sec	300	0.65	0.83
	10 sec	500	0.63	0.71

Table 2: Micro-activity classification metrics for implemented architectures

Classifier	Window Size	Samples/Window	Validation Accuracy	Training Accuracy	Validation MSE	Training MSE
DCBL	3 sec	150	0.47	0.59	0.1211	0.0157
	6 sec	300	0.50	0.57	0.1029	0.0081
	10 sec	**500**	**0.50**	**0.57**	**0.0945**	**0.0065**
CNN	3 sec	150	0.49	0.59	0.1245	0.0453
	6 sec	300	0.41	0.51	0.1178	0.0805
	10 sec	500	0.47	0.58	0.1195	0.0496
LSTM	6 sec	300	0.38	0.46	0.1446	0.1110
	10 sec	500	0.29	0.39	0.1417	0.1040

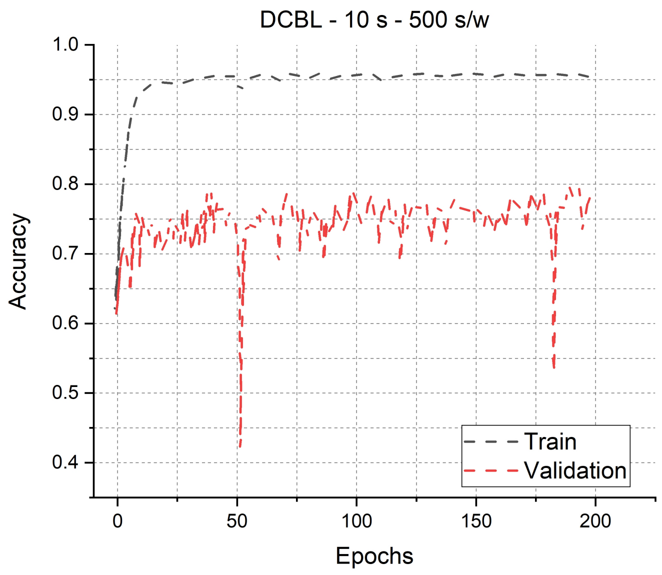

Figure 1: Learning curve for best macro-activity classifier.

B. Architectures and Methods

B.1. Implementation

The models were implemented in Jupyter notebook (Python), using Keras and Sklearn via a Tensorflow backend, with MATLAB being used for minor errands. All models were trained on a GPU-machine with 128 GB RAM, 2x 12 GB Nvidia GeForce GTX 1080 Ti and 3.4GHz AMD Ryzen Threadripper 1950X 16-core CPU.

B.2. Splitting, Windowing and Hyperparameters

The training set was split intelligently before windowing (along with random permutation) to ensure that the macro-labels are balanced in the training (and validation) set. Holdout ratio was 70:30 (train:validation). To account for missing data and misalignment, rather than attempting to interpolate at exact sample points manually, the individual overlapping windows were aligned based on initial and terminal timestamps of each window, appending zeros at the end of the window if number of samples were less than desired. Hyperparameters in this step include the window size (3, 6 and 10 seconds), number of samples per overlapping window (150, 300 and 500) and step size for overlapping window (1 second was chosen for at least $\simeq$ 60% overlapping). For the classifiers, the hyperparameters were batch size (32, 64), dropout (set to 0.5 to prevent overfitting) and batch normalization (set to 1). 128 units were chosen in the hidden layers for convolutional, LSTM and MLP layers. All models were trained for 200 epochs. Adam optimizer was used during the training process.

To manually account for missing data, pre-processing techniques such as bandpass filtering (using 20 Hz butterworth low-pass filter), normalization (z-normalization, univariance and min-max scaling) and interpolation (linear, spline) were applied to the training and validation samples but negligible or negative performance improvement was observed over our windowing approach. One example attempt (out of multiple) is repored for DCBL in Table 1. We also initially attempted a naive implementation of DCBL for all 99 sensor channels and the mo-cap channels separately, achieving only around 30-40% validation accuracy for macro-activity classification alone.

B.3. LSTM Architecture

Table 3: Sample architecture of implemented LSTM.

Layer (type)	Param #	Size	Stride	Activation	Output Shape
LSTM 1	70656			tanh	(300, 128)
Dropout 1					(300,128)
LSTM 2	131584			tanh	(128)
Dropout 2					(128)
Dense 2	387			Softmax*	(3)

B.4. CNN Architecture

Table 4: Sample architecture of implemented CNN.

Layer (type)	Param #	Size	Stride	Activation	Output Shape
Conv 1	1408	(1, 10)	(1, 1)	ReLu	(9, 300, 128)
Conv 2	163968	(1, 10)	(1, 1)	ReLu	(9, 300, 128)
B. Norm 1	512				(9,300,128)
Conv 3	163968	(1, 10)	(1, 1)	ReLu	(9, 300, 128)
Conv 4	163968	(1, 10)	(1, 1)	ReLu	(9, 300, 128)
B. Norm 2	512				(9, 300, 128)
Max. pool 1		(1,3)	(1,1)		(9, 100, 128)
Conv 5	163968	(1, 10)	(1, 1)	ReLu	(9, 100, 128)
Conv 6	163968	(1, 10)	(1, 1)	ReLu	(9, 100, 128)
B. Norm 3	512				(9, 100, 128)
Max. pool 2		(1,2)	(1,1)		(9, 50, 128)
Dropout 1					(9, 50, 128)
Conv 7	163968	(1, 10)	(1, 1)	ReLu	(9, 50, 128)
B. Norm 4	512				(9, 50, 128)
Max. pool 3		(1,2)	(1,1)		(9, 25, 128)
Dropout 2					(9, 25, 128)
Flatten 1					(28800)
Dense 1	1843264			ReLu	(64)
B. Norm 4	256				(64)
Dropout 3					(64)
Dense 2	195			Softmax*	(3)

B.5. DCBL Architecture

Table 5: Sample architecture of implemented DCBL.

Layer (type)	Param #	Size	Stride	Activation	Output Shape
Conv 1	1408	(1, 10)	(1, 1)	ReLu	(9, 500, 128)
Conv 2	163968	(1, 10)	(1, 1)	ReLu	(9, 500, 128)
B. Norm 1	512				(9,500,128)
Conv 3	163968	(1, 10)	(1, 1)	ReLu	(9, 500, 128)
Conv 4	163968	(1, 10)	(1, 1)	ReLu	(9, 500, 128)
B. Norm 2	512				(9, 500, 128)
Max. pool 1		(1,3)	(1,1)		(9, 166, 128)
Conv 5	163968	(1, 10)	(1, 1)	ReLu	(9, 166, 128)
Conv 6	163968	(1, 10)	(1, 1)	ReLu	(9, 166, 128)
B. Norm 3	512				(9, 166, 128)
Max. pool 2		(1,2)	(1,1)		(9, 83, 128)
Dropout 1					(9, 83, 128)
Conv 7	163968	(1, 10)	(1, 1)	ReLu	(9, 83, 128)
B. Norm 4	512				(9, 83, 128)
Max. pool 3		(1,2)	(1,1)		(9, 41, 128)
Dropout 2					(9, 41, 128)
Permute 1					(41,9,128)
Reshape 1					(41,1152)
Bi. LSTM	1311744				(256)
Dense	771			Softmax*	(3)

* for micro-activity classification, activation is ReLu.