This ICCV workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the version available on IEEE Xplore.

# Batch-Based Activity Recognition from Egocentric Photo-Streams

Alejandro Cartas[1,2] Mariella Dimiccoli[1,2] Petia Radeva[1,2]

[1]University of Barcelona
Mathematics and Computer Science Department
08007 Barcelona
Spain
{alejandro.cartas, petia.ivanova}@ub.edu

[2] Computer Vision Center
Universitat Autónoma de Barcelona
08193 Cerdanyola del Valls
Spain
mariella.dimiccoli@cvc.uab.es

## Abstract

*Activity recognition from long unstructured egocentric photo-streams has several applications in assistive technology such as health monitoring and frailty detection, just to name a few. However, one of its main technical challenges is to deal with the low frame rate of wearable photo-cameras, which causes abrupt appearance changes between consecutive frames. In consequence, important discriminatory low-level features from motion such as optical flow cannot be estimated. In this paper, we present a batch-driven approach for training a deep learning architecture that strongly rely on Long short-term units to tackle this problem. We propose two different implementations of the same approach that process a photo-stream sequence using batches of fixed size with the goal of capturing the temporal evolution of high-level features. The main difference between these implementations is that one explicitly models consecutive batches by overlapping them. Experimental results over a public dataset acquired by three users demonstrate the validity of the proposed architectures to exploit the temporal evolution of convolutional features over time without relying on event boundaries.*

## 1. Introduction

Automatic human behavior understanding has been for a long time one of the main goals of artificial intelligence practitioners [22]. Being a fundamental step towards human behavior understanding and having several application areas like healthcare, ambient intelligence, and video surveillance, activity recognition has become one of the most widely studied problems in computer vision [11, 19, 25, 29, 33].

With the widespread of wearable sensors in recent years [14, 28], there has been growing interest in recognizing activities from images and videos captured by a wearable camera [20]. Since wearable cameras do not require any user intervention, they allow to capture genuine images and videos in a naturalistic setting. Additionally, the first-person point of view is specially well-suited to capture interactions with objects and people; hence tracking the activities of the wearer (see Fig. 1). However, in comparison with third-person videos, the camera free motion, the unconstrained nature of the videos, and the non-visibility of the main actor impose additional challenges to the activity recognition problem.

Figure 1: Examples of egocentric images captured by a chest-mounted OMG Autographer wearable camera from the NTCIR-12 dataset [15].

In a lifelogging scenario, where typically the frame-rate of the camera is very low (2-3 frames per minute), the lack of temporal coherence and the abrupt changes of the field of view, further harden the activity recognition task. As stated in [4], visual lifelogs offer considerable potential to infer behavior patterns through activ-

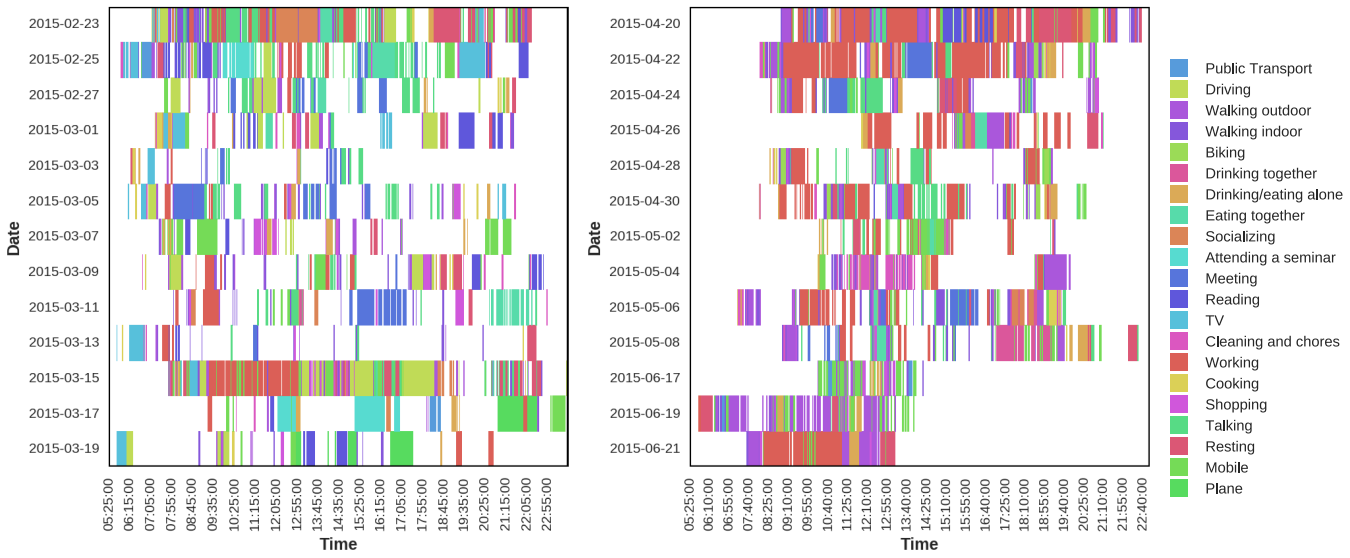



Figure 2: Thirteen different day sequences of annotated pictures from users *u1* and *u2* of the NTCIR-12 dataset.

ity recognition and enable several applications in the field of technology-driven assistive healthcare, such as preventing non-communicable diseases associated with unhealthy trends and risky profiles. Despite that, to the best of our knowledge, research on activity recognition in a lifelogging scenario has received comparatively little attention in the literature [6, 7, 21] with respect to the egocentric video setting [1, 2, 3, 12, 16, 17, 18, 23, 24, 26, 27].

Activity recognition from egocentric videos is mostly focused on recognizing short-term actions such as *take bread* or *put ketchup*, spanning around a few hundreds of frames or more long-term activities: *walking* or *running* that usually last for several minutes. Not surprisingly, one of the most widely exploited features in the video setting was the ego-motion

Given the low frame-rate, activity recognition from egocentric photo-streams has focused on recognizing high-level activities that may last from a few minutes to several hours such as *cooking* or *working* that have proved to be well characterized by ego-motion [24] in the video scenario. Castro et al. [7] showed that it is possible to improve the recognition performance of a fine-tuned CNN by adding a fusion ensemble method that puts together the output of the CNN with time meta-data and contextual features through a random forest. However, the validity of the proposed approach was restricted to data belonging to a single user, since the time meta-data and the contextual features cannot generalize to multiple users. In an attempt to improve the generalization capability of this model, Cartas et al. [6] proposed to use the output of a fully connected layer as additional features to be used in a fusion ensemble model. However, both works operated at image-level even if images were manually labeled in batches. This implies that the annotators used to apply temporal information in order to label certain images, and therefore the labeling of single images could have been different without taking temporal information into account. Oliveria et al. [21] exploited the relationship between objects and activities for activity recognition purpose. However, all the above mentioned approaches treated photo-streams as an unstructured collection of unrelated images.

Encouraged by a previous study [5] showing that, besides drastic changes in appearance, the temporal coherence of concepts is preserved in egocentric photo-streams at event-level, we aim to investigate how to take advantage of temporal information to improve the recognition performance. Our proposed approach is similar to [34], where the activity recognition problem from third-person cameras is cast as a video classification problem, whose goal is to learn a global description of the video while maintaining a low computational cost. In [34], the video is down-sampled to a frame-rate of 1 fpm, and explicit motion information in the form of optical flow images computed over adjacent frames is added to compensate the lost of implicit motion information. A Long Short Term Memory (LSTM) recurrent neural network operating on frame-level CNN activations is used to discover long-range temporal relationships and to learn how to integrate information over time. Contrary to [34], in our lifelogging scenario, image sequences have been originally recorded with a low frame-rate, so that motion information is not available. Furthermore, since a single video for us corresponds to the set of all images of the day, the number of labels may be arbitrarily large and activity boundaries are unknown. Therefore, we propose a

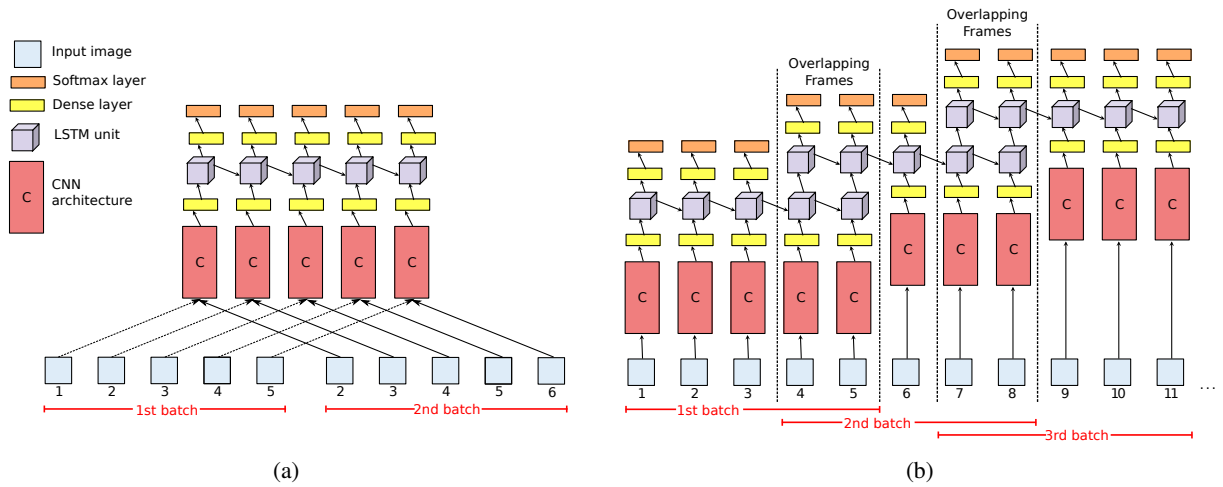



Figure 3: Proposed batch-based activity recognition implementations. (a) CNN+LSTM with a timestep of 5 frames; it has 4 repeated frames between consecutive batches. (b) CNN+Piggyback LSTM with a timestep and overlap of 5 and 2 frames, respectively.

batch-based deep learning training approach aiming to cope with both the lack of knowledge about event boundaries and the not negligible length of photo-streams (up to 2,000 images).

Our contributions can be summarized as follows:

- We propose two end-to-end batch-based implementations that exploit the temporal coherence of concepts in photo-streams and outperform the state of the art end-to-end architectures.
- We demonstrate that it is possible to capture the temporal evolution of features over time in photo-streams even without knowing event boundaries.
- We show that both implementations improve the classification performance, but the first one is slightly better on day sequences that does present clear temporal patterns.

The rest of the paper is organized as follows: section 2 details the proposed approach, whereas the experimental setting, validations and state-of-the-art comparisons are described and discussed in section 3. Concluding remarks are reported in section 4.

## 2. Proposed approach

Our hypothesis is that the variations between successive images in a photo-stream encode additional information which could be useful in making more accurate activity predictions. While this hypothesis has been validated in [34] for conventional videos whose frames share the same label, there is no evidence that it is applicable to the photo-stream scenario. Typically, a photo-stream has several labels corresponding to different activities performed by the user during a day. Due to the sparseness of the observations, adjacent frames may have distinct labels even if in general, when the user is performing a long term activity, several consecutive frames share the same label (see Figure 2). One possible approach would be to first split the photo-stream or video into events manually or by using a state of the art approach as [32, 10], and then to classify each event separately. However, we propose two different CNN-LSTM implementations,based on the idea of batch-based training, able to cope with the whole photo-stream set analysis without the necessity for event segmentation.

In the first implementation (see Fig. 3a), the output from the CNN layer is given as input to a single LSTM layer. While the architecture itself is not new, the way it is trained and what it is supposed to learn differs from [34]. During training, we split the photo-stream into overlapping segments of fixed length and we feed them, together with the corresponding activity label at frame level, to the network. Since each segment has a fixed length, its images may not share the same label. Therefore, we expect the network to learn not only the temporal evolution of features over time within a same event, but also to learn to predict event changes, thus leading to more accurate predictions when event boundaries are unknown.

In the second implementation (see Fig. 3b), we explicitly model the temporal relation between two adjacent overlapping batches. After a CNN architecture, we added a *fully-connected* layer, a LSTM unit, and one last *fully-connected* layer. For frames belonging to the first batch of a sequence (frames 1-5 in Fig. 3b(b)), the input of the LSTM layer is the output of the *fully-connected* layers. For frames belonging to two temporally overlapping batches (frames 4-5

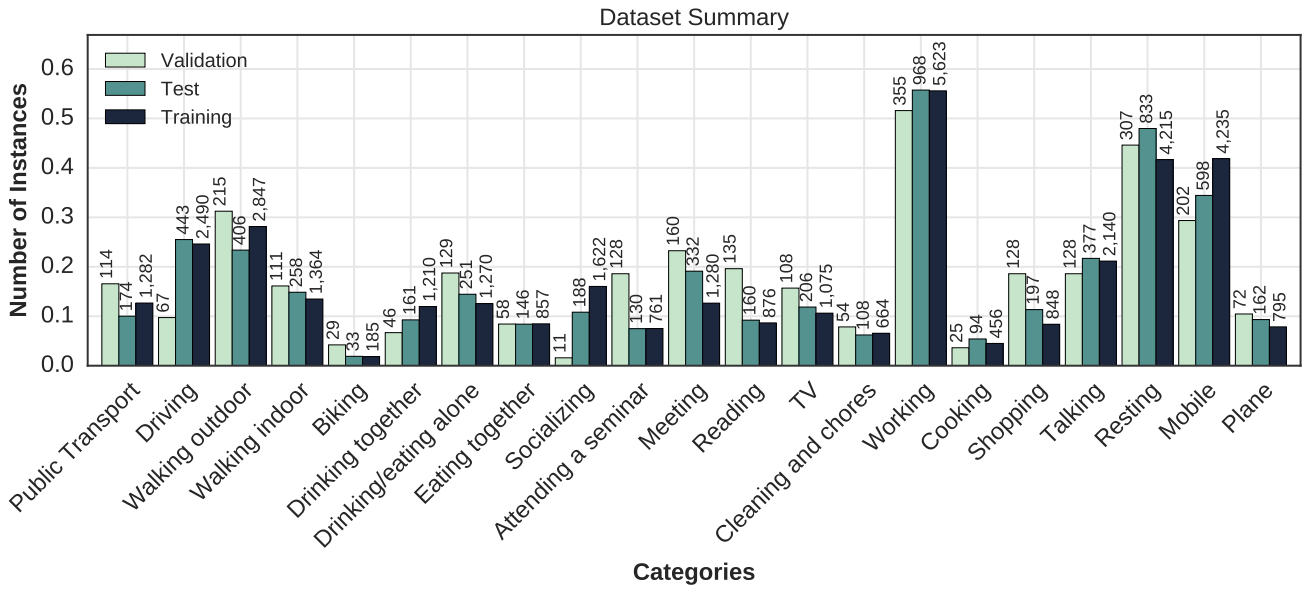



Figure 4: Dataset split summary. All split distributions are normalized, but the corresponding number of instances for each category is shown on top of each split.

in Fig. 3b(b)), the input of the LSTM layer is the output of the LSTM layer from the previous batch. The outputs of the first *fully-connected* layer and the LSTM unit have the same size. In this way, the input of the LSTM layer has the same size of its output, that in the latter is used in subsequent passes. In other words, after the initial feed-foward pass of the first sequence batch, the LSTMs output of the last $n$ frames are stored. In subsequent passes, they are used as the input for the first $n$ LSTMs. For example, Fig. 3b shows this configuration composed of 5 timesteps with an overlapping of 2 output/input. This architecture is supposed to learn more complex long-range temporal dependencies without the need of considering each one-day photo-stream as a single sequence. Indeed, since a photo-stream can be made of up to two thousands frames, even for an LSTM it would be unfeasible to learn such long range dependencies[13]. Using both implementations, we finally obtain the frame-level predictions.

## 3. Experiments

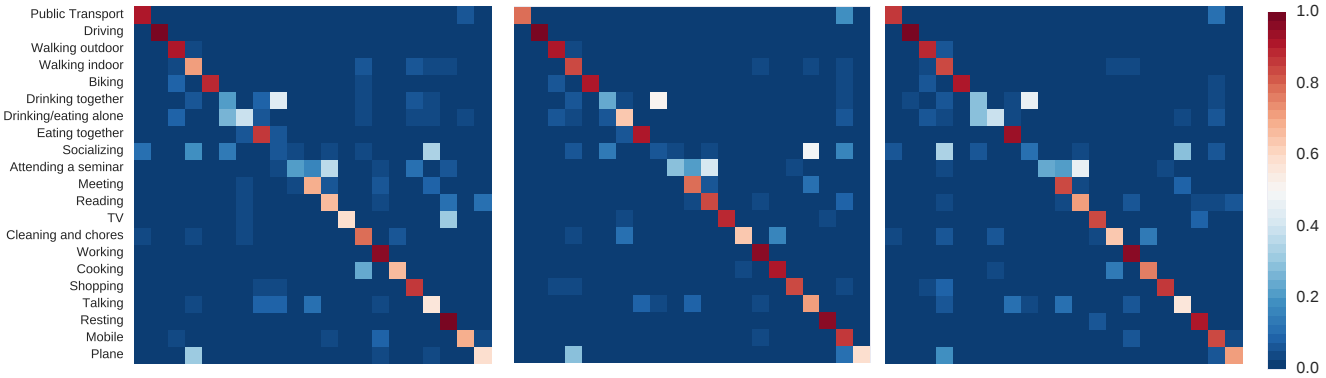
We describe the dataset in section 3.1 and detail the networks training in section 3.2. We then present the experimental results on activity recognition on section 3.3.

### 3.1. Dataset

We employed the NTCIR-12 dataset [15] on our experiments. This dataset contains 89,593 egocentric images collected in 79 days by three different persons. The data collection was done in a period of almost a month per person. During this time, each user worn a chest-mounted camera that took two pictures per minute. Continuing previous work [6], we used an extended subset of 44,902 images from the NTCIR-12 dataset, around 15,000 images per person. These images were annotated using 21 activity categories and correspond to all three users and 78 days at different times. The annotation process was done in batches of consecutive frames, meaning that the context of a continuous activity across frames was implicitly taken into account by the annotators.

We split the annotated images subset in training, validation, and test sets. These splits contain full day sequences and maintain the inherit class imbalance, as illustrated on Fig. 4. We accomplished this by doing the following. First, the day sequences were grouped in bins of similar number of images by using the first-fit decreasing algorithm. Second, all possible combinations of test splits from the bins were calculated by using the Twiddle algorithm [8]. Then, two category distributions were computed for each test split combination and its remaining bins. Then, for each pair of distributions the sum of the Bhattacharya distances between them and the whole dataset category distribution was obtained. Finally, the best test split is the one with the shortest distance. The validation and training split was calculated on the remaining bins by doing the same steps.

For the recurrent proposed models, the annotated pictures of a day were considered as one sequence. In total, the training and validation set consisted of 59 and 7 sequences,




(a) VGG-16

(b) VGG-16+LSTM (timestep of 15)

(c) VGG-16+Piggyback LSTM (timestep of 10 and overlap of 3)

Figure 5: Normalized confusion matrices of all models for the best trained configuration.

respectively. For example, thirteen day sequences from two users are shown in Fig. 2.

### 3.2. Training

All the models were implemented using the Keras framework [9]. In addition, all the models have the same data augmentation process at the frame level. Namely, we randomly applied horizontal flips, translation and rotation shifts, and zoom operations. To avoid overfitting, we added dropout layers [31] and weight normalization to all models. The VGG CNN architecture is used to process individual images. To aggregate photostream-level information, we leverage an LSTM to consider sequences of CNN activations. The training and configuration details for all models are described as follows.

**VGG-16 CNN**. We fine-tuned a VGG-16 network [30] as our base model. Only the last fully-connected layer was changed to have a 21 category output. The fine-tuning was done in two phases using the splits described in the previous section.The goal of the first phase was to initialize the weights at the top layers, since only the bottom layers of the CNN were initialized using the weights of the ImageNet classification task. Therefore, during the first phase, only the fully-convolutional layers were backpropagated. The optimization method used was the Stochastic Gradient Descent (SGD) for 10 epochs, a learning rate $\alpha = 1 \times 10^{-5}$, a batch size of 1, a momentum $\mu = 0.9$, and a weight decay equal to $5 \times 10{-}6$. In the second phase, the last two convolutional layers were also fine-tuned and the initial weights were obtained from the best epoch of the first phase. Moreover, the SGD ran for another 10 epochs and set with the same parameters except the learning rate $\alpha = 4 \times 10^{-5}$. The best validation results were obtained at the eight epoch.

**VGG-16 CNN + LSTM**. For this architecture, we added a LSTM layer of 256 units followed by a fully-connected layer after the first fully-connected layer of the VGG-16 network. Furthermore, in our experiments, we used three LSTM configurations with a time step of five, ten, and fifteen frames. In order to train all the configurations, we froze the weights of the first four blocks of the convolutional layers. During training, we used SGD as optimization algorithm. For the timestep 5 configuration, we trained it for 5 epochs with a learning rate $\alpha = 2.5 \times 10^{-5}$, a momentum $\mu = 0.9$, and a weight decay equal to $5 \times 10{-}6$. For the timestep 10 configuration, we trained it for 4 epochs with a learning rate $\alpha = 1 \times 10^{-4}$, a momentum $\mu = 0.9$, and a weight decay equal to $5 \times 10{-}6$. The timestep 15 configuration was trained for 2 epochs with a learning rate $\alpha = 1 \times 10^{-4}$, a momentum $\mu = 0.9$, and a weight decay equal to $5 \times 10{-}6$. The initial weights of the optimization process were the ones obtained for the base model. The training was performed in batches of 5, 10, and 15 frames, respectively. These batches were sampled using a sliding window of frames from each sequence. For instance, Fig.

| Method | Accuracy | Macro Precision | Macro Recall | Macro F1-score |
|---|---|---|---|---|
| VGG-16 | 75.97 | 68.50 | 67.49 | 66.80 |
| VGG-16+LSTM timestep 5 | 79.68 | 72.96 | 71.36 | 70.87 |
| VGG-16+LSTM timestep 10 | 80.39 | 75.25 | 71.86 | 71.97 |
| VGG-16+LSTM timestep 15 | **81.73** | **76.68** | **74.04** | **74.16** |
| VGG-16+Piggyback LSTM timestep 5 overlap 2 | 75.97 | 69.74 | 62.98 | 63.24 |
| VGG-16+Piggyback LSTM timestep 10 overlap 3 | 79.04 | 72.98 | 71.88 | 71.06 |
| VGG-16+Piggyback LSTM timestep 15 overlap 4 | 78.51 | 73.00 | 69.52 | 69.88 |

Table 1: Performance summary of all methods with different number of timesteps and overlapping frames.

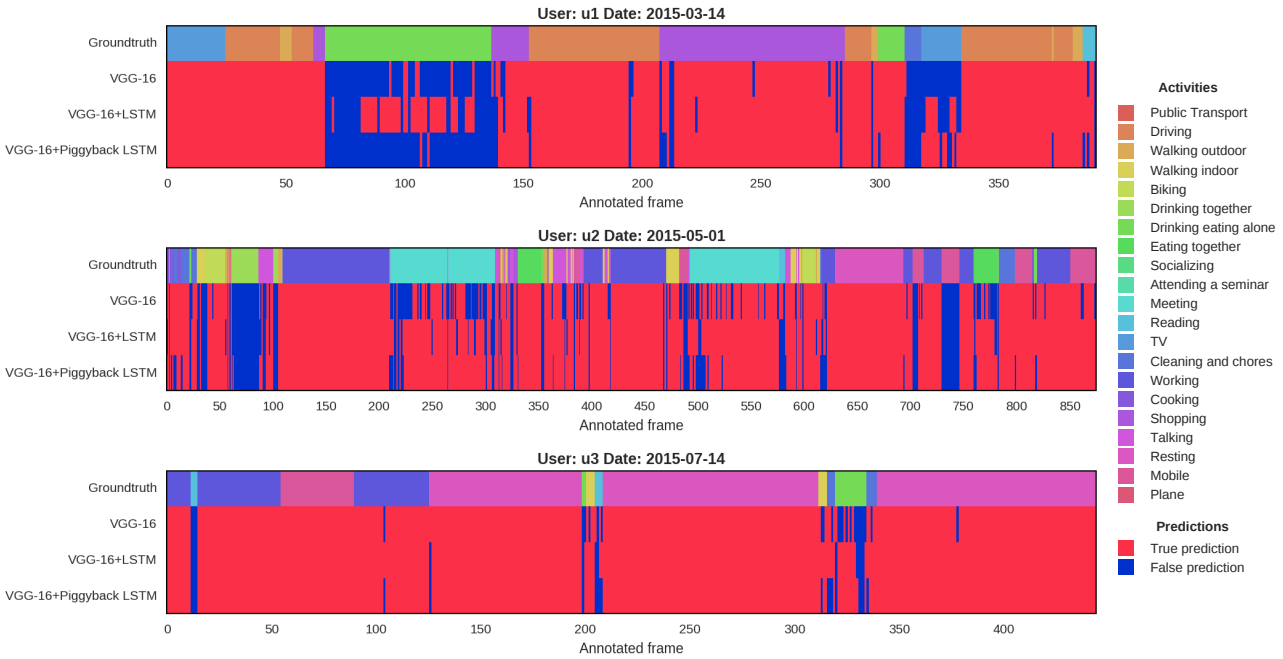



Figure 6: Prediction comparison of all models for three test day sequences from all users in the dataset.

3a shows an unrolled version of this model and two consecutive batches of 5 frames. Besides being a crucial characteristic of the proposed approach, this form of training can be understood as a kind of data augmentation over the sequence.

**VGG-16 CNN + Piggyback LSTM**. For this architecture, we added a fully-connected and LSTM layers after the convolutional architecture of the VGG-16 network. Both layers have an output vector length of 256 and are followed by a dense layer with a softmax activation. The feedback from overlapping frames between batches was implemented using a filter layer. Therefore, the network had two additional inputs: one input for the previous batch and another one used as a mask. We used three different configurations in the reported experiments. The first configuration had a batch size of 5 frames and 2 overlapping frames, the second one had a batch size of 10 frames and 3 overlapping frames, and the last one had a batch size of 15 frames and 4 overlapping frames. Our results were achieved by dividing the training into two phases. The purpose of the first training phase was to learn the high-level features from adjacent frames, while the purpose of the second one was to learn temporal patterns from them throughout the sequence. During the first phase, the day sequences were considered as consecutive batches without overlapping. Accordingly, this phase followed the same training procedure as the previous described architecture. In the second phase, we froze all the convolutional layers and the first fully-connected layer. The data augmentation process for the day sequences consisted in the following. Given a batch size $n$ and a overlapping number of frames $m$, the sequential training batches with $m$ overlapping frames were created from the day sequences starting at the first $n$ frames. This created more training examples than the previous architecture and considered all the frames of a sequence. The first training step was to randomly shuffle the day sequences at each epoch. Then, all the day sequences were processed one by one. The ordered training batches from a day sequence were consecutively feedforwarded to the network. The SGD algorithm was also used for this second phase. The learning rates for the configurations were $\alpha = 2.5 \times 10^{-5}$ , $\alpha = 1 \times 10^{-4}$, and $\alpha = 1 \times 10^{-4}$, correspondingly. Moreover, all configurations shared the same momentum $\mu = 0.9$ and weight decay equal to $5 \times 10{-}6$. Early stopping criteria was used for both training phases.

### 3.3. Results and Discussion

The evaluation of the models and their configurations was done over 12 day sequences from all users, i.e. 6,225 images. In contrast with previous works [11, 34], we are interested in a many-to-many sequence classification and we did not apply any kind of average over a processed batch. Since the dataset is unbalanced, we used other metrics besides accuracy to measure the classification performance. Table 1 shows the obtained results for the activity recognition task. These results demonstrated that process-

ing batches of sequential frames improves accuracy performances with respect to the pure CNN baseline for most of the cases. A larger time-step is generally preferred since it allows to better capture the temporal evolution of features over time. By using the first proposed architecture, we achieve an improvement of more than $4\%$ with respect to the VGG-16 baseline even considering a very small batch size. The results of the second proposed architecture also improved all the performance metrics with respect to the pure CNN baseline, but only for the largest timestep configuration. In comparison with the former architecture, the overall performance improvement was less. This might be explained as a consequence of not having clear temporal activity patterns throughout the whole day sequences, as shown in Fig. 2. Moreover, a comparison of the models over three day sequences is illustrated in Fig. 6. Although, the proposed architectures improved the overall accuracy, they still failed at classifying categories highly correlated like *Eating together* and *Driking together* as seen on the confusion matrices on Fig. 5.

## 4. Conclusions

We presented a batch-based learning approach for activity recognition from egocentric photo-stream sequences. In order to learn temporal activity patterns between frames, both proposed implementations of this approach uses a LSTM unit on top of a convolutional neural network to process a day sequence of frames using windows of fixed size. Specifically, our first implementation uses a sliding window of consecutive frames to generate training batches. Moreover, our second implementation is able to handle information of previous batches from a sequence by reprocessing a fixed number of overlapping frames.

Although this paper has demonstrated that it is possible to exploit temporal coherence of concepts without knowing event boundaries, we consider that clustering a day sequence into different scene subsequences could further improve the activity recognition. Additionally, we think that the second proposed implementation could improve the activity recognition performance on video data. Both ideas will be addressed in future work.

## Acknowledgments

A.C. was supported by a doctoral fellowship from the Mexican Council of Science and Technology (CONACYT) (grant-no. 366596). This work was partially founded by TIN2015-66951-C2, SGR 1219, CERCA, *ICREA Academia'2014* and 20141510 (Marató TV3). The funders had no role in the study design, data collection, analysis, and preparation of the manuscript. M.D. is grateful to the NVIDIA donation program for its support with GPU card.

## References

[1] G. Abebe, A. Cavallaro, and X. Parra. Robust multi-dimensional motion features for first-person vision activity recognition. *Computer Vision and Image Understanding*, 149:229–248, 2016. 2

[2] S. Bambach, S. Lee, D. J. Crandall, and C. Yu. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1949–1957, 2015. 2

[3] A. Behera, D. C. Hogg, and A. G. Cohn. Egocentric activity monitoring and recovery. In *Asian Conference on Computer Vision*, pages 519–532. Springer, 2012. 2

[4] M. Bolanos, M. Dimiccoli, and P. Radeva. Toward storytelling from visual lifelogging: An overview. *IEEE Transactions on Human-Machine Systems*, 47(1):77–90, 2017. 1

[5] D. Byrne, A. R. Doherty, C. G. Snoek, G. J. Jones, and A. F. Smeaton. Everyday concept detection in visual lifelogs: validation, relationships and trends. *Multimedia Tools and Applications*, 49(1):119–144, 2010. 2

[6] A. Cartas, J. Marín, P. Radeva, and M. Dimiccoli. Recognizing activities of daily living from egocentric images. In *Proceedings of the Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, pages 87–95, Cham, June 2017. Springer. 2, 4

[7] D. Castro, S. Hickson, V. Bettadapura, E. Thomaz, G. Abowd, H. Christensen, and I. Essa. Predicting daily activities from egocentric images using deep learning. In *Proceedings of the 2015 ACM International symposium on Wearable Computers*, pages 75–82. ACM, 2015. 2

[8] P. J. Chase. Algorithm 382: Combinations of m out of n objects [g6]. *Commun. ACM*, 13(6):368–, June 1970. 4

[9] F. Chollet et al. Keras. https://github.com/fchollet/keras, 2015. 5

[10] M. Dimiccoli, M. Bolaños, E. Talavera, M. Aghaei, S. G. Nikolov, and P. Radeva. Sr-clustering: Semantic regularized clustering for egocentric photo streams segmentation. *Computer Vision and Image Understanding*, 155:55–69, 2016. 3

[11] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2625–2634, June 2015. 1, 6

[12] A. Fathi, A. Farhadi, and J. M. Rehg. Understanding egocentric activities. In *2011 International Conference on Computer Vision*, pages 407–414. IEEE, 2011. 2

[13] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471, 2000. 4

[14] S. D. Guler, M. Gannon, and K. Sicchio. A brief history of wearables. In *Crafting Wearables: Blending Technology with Fashion*, pages 3–10. Apress, Berkeley, CA, 2016. 1

[15] C. Gurrin, H. Joho, F. Hopfgartner, L. Zhou, and R. Albatal. Overview of ntcir-12 lifelog task. In *Proceedings of the 12th NTCIR Conference on Evaluation of Information Access Technologies*, pages 354–360, 2016. 1, 4

[16] Y. Li, Z. Ye, and J. M. Rehg. Delving into egocentric actions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 287–295, 2015. 2

[17] M. Ma, H. Fan, and K. M. Kitani. Going deeper into first-person activity recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1894–1903, June 2016. 2

[18] T. McCandless and K. Grauman. Object-centric spatio-temporal pyramids for egocentric activity recognition. In *BMVC*, volume 2, page 3, 2013. 2

[19] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4694–4702, 2015. 1

[20] T.-H.-C. Nguyen, J.-C. Nebel, and F. Florez-Revuelta. Recognition of activities of daily living with egocentric vision: A review. *Sensors (Basel)*, 16(1):72, Jan 2016. sensors-16-00072[PII]. 1

[21] G. Oliveira-Barra, M. Dimiccoli, and P. Radeva. Leveraging activity indexing for egocentric image retrieval. In *Proceedings of the Iberian Conference on Pattern Recognition and Image Analysis (IbPRIA)*, pages 295–303, Cham, June 2017. Springer. 2

[22] M. Pantic, A. Pentland, A. Nijholt, and T. Huang. Human computing and machine understanding of human behavior: A survey. In *Proceedings of the 8th international conference on Multimodal interfaces*, pages 239–248. ACM, 2006. 1

[23] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2847–2854. IEEE, 2012. 2

[24] Y. Poleg, A. Ephrat, S. Peleg, and C. Arora. Compact cnn for indexing egocentric videos. In *Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on*, pages 1–9. IEEE, 2016. 2

[25] R. Poppe. A survey on vision-based human action recognition. *Image and vision computing*, 28(6):976–990, 2010. 1

[26] M. Ryoo and L. Matthies. First-person activity recognition: Feature, temporal structure, and prediction. *International Journal of Computer Vision*, 119(3):307–328, 2016. 2

[27] M. S. Ryoo and L. Matthies. First-person activity recognition: What are they doing to me? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2730–2737, 2013. 2

[28] E. Sazonov and M. R. Neuman. *Wearable Sensors: Fundamentals, implementation and applications*. Elsevier, 2014. 1

[29] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014. 1

[30] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 5

[31] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014. 5

[32] E. Talavera, M. Dimiccoli, M. Bolanos, M. Aghaei, and P. Radeva. R-clustering for egocentric video segmentation. In *Iberian Conference on Pattern Recognition and Image Analysis*, pages 327–336. Springer, 2015. 3

[33] D. Weinland, R. Ronfard, and E. Boyer. A survey of vision-based methods for action representation, segmentation and recognition. *Computer vision and image understanding*, 115(2):224–241, 2011. 1

[34] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4694–4702, 2015. 2, 3, 6