Improving Activity Recognition using Temporal Regions

João Paulo Aires[1] and Juarez Monteiro[1] and Roger Granada[1] and
Felipe Meneguzzi[2] and Rodrigo C. Barros[2]

Faculdade de Informática
Pontifícia Universidade Católica do Rio Grande do Sul
Av. Ipiranga, 6681, 90619-900, Porto Alegre, RS, Brazil
[1] Email: {joao.aires.001, juarez.santos, roger.granada}@acad.pucrs.br
[2] Email: {rodrigo.barros, felipe.meneguzzi}@pucrs.br

Abstract. Recognizing activities in videos is an important task in computer vision area. Automatizing such task can improve the way we monitor activities since it is not necessary to have a human to watch every video. However, the classification of activities in a video is a challenging task since we need to extract temporal features that represent each activity. In this work, we propose an approach to obtain temporal features from videos by dividing the sequence of frames of a video into regions. Frames from these regions are merged in order to identify the temporal aspect that classify activities in a video. Our approach yields better results when compared to a frame-by-frame classification.

Categories and Subject Descriptors: I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding; I.5.4 [**Pattern Recognition**]: Applications

Keywords: Neural Networks, Convolutional Neural Networks, Activity Recognition

1. INTRODUCTION

Recognizing activities in videos is an important task for humans, since it helps the identification of different types of interactions with other agents. An important application of such task refers to the surveillance and assistance of the sick and disabled: such tasks require human monitors to identify any unexpected activity. Here, activity recognition refers to the task of dealing with noisy low-level data directly from sensors [Sukthankar et al. 2014]. The automation of such task is particularly challenging in the real physical world, since it either involves fusing information from a number of sensors or inferring enough information using a single sensor. To perform such task, we need an approach that is able to process the frames of a video and extract enough information in order to determine the activity.

Due to the evolution of graphics processing unit (GPU) and an increasing in computability power, deep learning methods started to be the state-of-the-art of different computer vision tasks [Karpathy et al. 2014; Chen et al. 2017; Redmon et al. 2016]. However, for activity recognition, deep learning algorithms need to consider the temporal aspect of videos since activities tend to occur through the frames. Besides that, this kind of algorithms demands a big volume of data in order to generalize when classifying unseen data.

In this work, we propose an approach to compute the temporal aspect from frames by dividing a video into regions and merging features from different regions in order to recognize an activity. Using Convolutional Neural Networks (CNNs) to extract features from each frame, we train a classifier that receives as input either a concatenation or the mean of features from different regions and outputs a prediction to the corresponding activity. Considering videos as a division of regions, we unify different moments of the video in a single instance, which makes it easier to classify. We perform experiments using two off-the-shelf CNNs in order to verify whether the regions work independently of the architecture. Our results indicate that the use of regions may increase the accuracy about 10% when compared with single networks that classify activities frame-by-frame.

2. METHOD

Recognizing activities from videos often involves an analysis of how objects in frames modify along the time. An approach to recognize activities in videos involves obtaining the temporal information, which may contain descriptions about how objects are moving and the main modifications between frames. Since an activity consists of a sequence of movements, obtaining the temporal information may help systems to better understand which activity is being performed. In this work, we propose an approach to obtain temporal information from a video by dividing its frames into regions. Thus, instead of classifying an activity using only the information from each image frame, we extract and merge the information from several regions of the video in order to obtain its temporal aspect. Figure 1 illustrates the pipeline of our architecture, which is divided into four phases: pre-processing, CNNs, regions division, and classification.

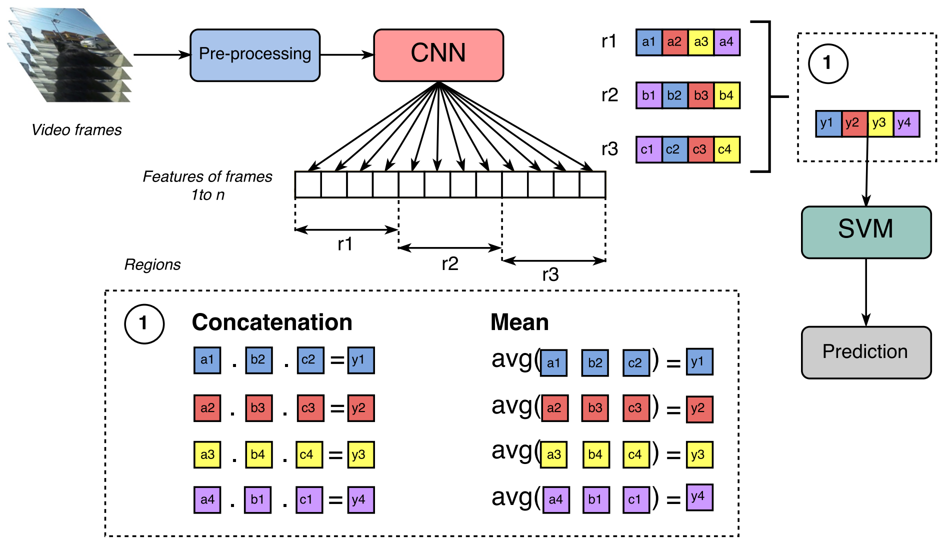

Fig. 1. Pipeline of our architecture for activity recognition using multiple regions.

1) Pre-processing: This phase consists of resizing images to a fixed resolution of 256×256. Resizing is important since it reduces the number of features for each image inside the CNN as well as the processing time.

2) CNNs: In this phase we train two CNNs using the DogCentric Activity dataset [Iwashita et al. 2014] in order to extract features of the activity in each frame. We first train a *GoogLeNet* [Szegedy et al. 2015] due to its reduced number of parameters generated by *inception* modules. *GoogLeNet* is a 22-layer deep network and is based on the *inception* module that contains convolutional filters with different sizes, covering different clusters of information. The second CNN is a slightly modified version of *AlexNet* [Krizhevsky et al. 2012] called *CaffeNet* [Jia et al. 2014] due to its small architecture, which is able to provide a fast training phase. Our version of the *AlexNet* contains 8 weight layers including 5 convolutional layers and 3 fully-connected layers, and 3 max-pooling layers following the first, second and fifth convolutional layers. Both networks receive a sequence of images as input, passing them through several convolutional layers, pooling layers and fully-connected layers (FC), ending in a *softmax* layer that generates the probability of each image to each class. We use CNNs as feature extractors for images of the dataset in order to generate a feature representation of each frame.

3) Regions division: In this phase, we divide each sequence of frames of a video into n regions of the same size, *i.e.*, containing the same number of frames, discarding the remaining ones in case they exist. To make a composition of different parts of the video, we take one frame of each region and either concatenate or take the mean of their features. For example, consider a video divided into three regions and each frame containing ten features, the resulting vector of a concatenation will contain thirty features, while the resulting vector of the mean will contain ten features. In order to avoid repetitions between the regions representation, we merge the features of the frame in the i^{th} position of the first region with the feature of the frame in the $i^{th}+1$ position in the other regions, *e.g.*, $(a_1 \cdot b_2 \cdot c_2)$ in "concatenation" in Figure 1. In case where the features of the frame are in the last position of the first region, the features of the frames in the next regions are extracted from the first position, *e.g.*, $(a_4 \cdot b_1 \cdot c_1)$ in "concatenation" in Figure 1.

4) Classification: In this phase, we apply a Support Vector Machine (SVM) [Crammer and Singer 2001] on the features from the concatenation or mean phase in order to predict the activity.

3. EXPERIMENTS

In this section, we describe the dataset used in our experiments for animal activity recognition and the implementation details used in the CNN models and SVM algorithm.

3.1 Dataset

The DogCentric Activity dataset[1] [Iwashita et al. 2014] consists of 209 videos containing 10 different activities performed by 4 dogs. The dataset contains first-person videos in 320×240 resolution (recorded at 48 frames per second) taken from an egocentric animal viewpoint, *i.e.*, a wearable camera mounted on dogs' back records outdoor and indoor scenes. Not all dogs perform all activities and an activity can be performed more than once by the same dog. The 10 target activities in the dataset include: "waiting for a car to pass by" (hereafter named *Car*), "drinking water" (*Drink*), "feeding" (*Feed*), "turning dog's head to the left" (*Left*), "turning dog's head to the right" (*Right*), "petting" (*Pet*), "playing with a ball" (*Play*), "shaking dog's body by himself" (*Shake*), "sniffing" (*Sniff*), and "walking" (*Walk*). It is important to mention that these egocentric (first-person) videos are very challenging due to their strong camera motion. In the left side of Figure 2, we show the dataset distribution according to its classes. As we can see, classes such as 'Left' and 'Right' have less than 10% of presence in the dataset, whereas 'Car' and 'Sniff' classes have 15% indicating that the dataset is unbalanced. In the right side of Figure 2, we illustrate examples for each class in the dataset.

In order to perform experiments, we divided the dataset into training, validation, and test sets. We use the validation set to obtain the model configuration that best fits the training data, *i.e.*, the configuration with the highest accuracy, and the test set to assess the accuracy of the selected model in unseen data. We use a method to divide the dataset that is similar to the one proposed by Iwashita et al. [2014], and consists of randomly selecting half of the videos of each activity as test set. In case where the number of videos (N) is an odd number, we separate $\frac{(N+1)}{2}$ videos for test set. We divide the rest of the videos into training and validation sets. Validation set contains 20% of the videos and the rest is separated to training set. Thus, our division contains 105 videos (17,400 frames) in testing set, 20 videos (3,205 frames) in validation set and 84 videos (13,040 frames) in training set.

[1] `http://robotics.ait.kyushu-u.ac.jp/~yumi/db/first_dog.html`

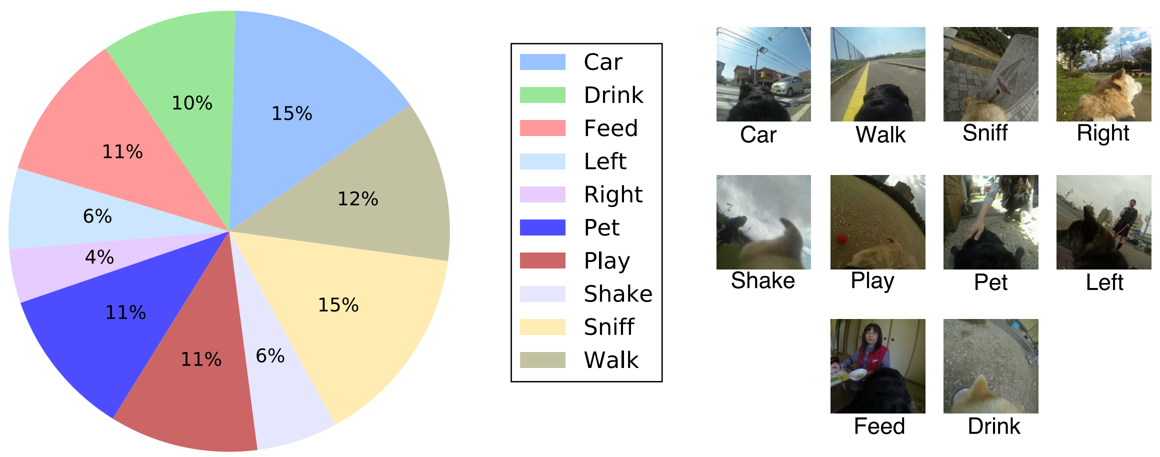

Fig. 2. Frames from each class of the dataset and the distribution of frames in classes.

3.2 Model Training

We implemented each part of our architecture using different toolkits: for CNNs, we use the *Caffe*[2] framework [Jia et al. 2014], and for the SVM, we use the Crammer and Singer [2001] implementation from *scikit-learn*[3] [Pedregosa et al. 2011]. Each CNN is trained by fine-tuning a network pre-trained in ImageNet [Russakovsky et al. 2015]. In training phase, both networks (*GoogLeNet* and *AlexNet*) use mini-batch stochastic gradient with momentum (0.9). For each image, we apply a random crop, *i.e.*, a crop in a random part of the image, generating a sub-image of 224×224. We subtract all pixels from each image by the mean pixel of all training images. The activation of each convolution (including those within the "inception" modules in *GoogLeNet*) use a Rectified Linear Unit (ReLU).

Each iteration in *GoogLeNet* uses a mini-batch with 128 images. For the weight initialization, we employ the *Xavier* [Glorot and Bengio 2010] algorithm, which automatically determines the value of initialization based on the number of input neurons. To avoid overfitting, we apply a dropout with a probability of 80% on the fully-connected layers. The learning rate is set to 3×10^{-4} and we drop it to 2×10^{-4} every epoch, stopping the training after 2.04k iterations (20 epochs).

In *AlexNet*, each iteration contains a mini-batch with 64 images. We initialize the weights using the *Gaussian* distribution with a standard deviation of 0.01. Similar to *GoogLeNet*, we avoid overfitting by applying a dropout with 90% of probability to prone nodes of fully-connected layers. The learning rate is set to 10^{-4} and we drop it 5×10^{-4} every epoch, stopping the training after 4.08k iterations (20 epochs).

4. RESULTS

In order to evaluate our approach, we calculate the accuracy ($\mathcal{A}$), precision ($\mathcal{P}$), recall ($\mathcal{R}$) and F-measure ($\mathcal{F}$). Since classification accuracy takes into account only the proportion of correct results that a classifier achieves, it is not suitable for unbalanced datasets because it may be biased towards classes with larger number of examples. Although other factors may change results, classes with a larger number of examples tend to achieve better results since the network has more examples to learn the variability of the features. By analyzing the DogCentric dataset, we notice that it is indeed unbalanced, *i.e.*, classes are not equally distributed over frames. Thus, we decided to calculate

[2]http://caffe.berkeleyvision.org/
[3]http://scikit-learn.org/

precision, recall, and f-measure to make a better evaluation over the unbalanced dataset. We calculate scores considering all classes presented in the test set as explained in Section 3.1. Table I shows the results achieved by *AlexNet* and *GoogLeNet* in our experiments when splitting the dataset into regions. The best results for concatenation and mean are in bold. It is important to note that Regions=1 is equivalent to compute single frames individually without splitting the video, *i.e.*, the traditional approach that do not encode the temporal aspect of an activity.

1) Overall Performance: As presented in Table I, when splitting the video into temporal regions we increase the accuracy of the network. For *AlexNet*, we achieve the best accuracy in two situations: (a) by using 16 regions and the mean of the feature vectors (59% of accuracy); and (b) using 32 regions with the concatenation of the vectors (59% accuracy). In terms of precision, recall, and f-measure, *AlexNet* achieves the best results when using 32 regions and the concatenation of vectors. For *GoogLeNet*, we obtain the best accuracy for both 8 and 16 regions, achieving 62% of accuracy using concatenated vectors, while the highest precision is achieved by the concatenation of vectors using 16 regions (72% of precision). In general, *AlexNet* achieves the best scores when increasing the number of regions to 32, while *GoogLeNet* achieves the highest scores using 8 regions. Overall, *GoogLeNet* obtains better results for all metrics. This occurs due to the set of layers in *GoogLeNet*, such as inception that allows the extraction of more complex features from data.

2) Mean vs. Concatenation: In general, the concatenation and the mean of the regions achieve close results for most regions. However, when increasing the number of regions the concatenation surpasses the mean and achieves the best results. On the other hand, the concatenation has the drawback that the more the number of regions, the more memory the system needs, while the mean keeps the memory constant.

3) Single vs. Multiple Regions: When comparing the architectures using single frames or multiple regions, we can observe that even when using two regions, the performance increase in both *AlexNet* and *GoogLeNet*. Although some activities can be identified in a single frame, it usually takes several frames to identify an activity. Thus, it seems reasonable that using more than a single frame will lead to better results. For a better understanding of how networks are classifying each class using a single frame and a set of regions, we illustrate in Figure 3 the confusion matrices generated for single frames and for the concatenation of 32 regions (the best results) for *AlexNet*, and the confusion matrices generated by single frames and the concatenation of 8 regions for *GoogLeNet*. As we can see in Figure 3, there is an increase in precision for bot *Left* and *Shake* classes when splitting the dataset into regions to predict the class of the video using *AlexNet*. Comparing the results from *GoogLeNet*, there is an increase in precision for both *Right* and *Shake* classes when splitting the dataset into eight regions. Other classes are also positively affected by the region division as it works as a reinforcement to the network classification. Regarding the number of regions, we notice that our results start to obtain significant increase with 8 regions or more. There is a difference between *GoogLeNet* and *AlexNet*

Table I. Accuracy ($\mathcal{A}$), Precision ($\mathcal{P}$), Recall ($\mathcal{R}$) and F-measure ($\mathcal{F}$) achieved for *AlexNet* and *GoogLeNet* when using regions in the DogCentric Activity dataset.

Regions	**Fusion**	**AlexNet**				**GoogLeNet**			
		$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$
1	–	0.51	0.57	0.52	0.52	0.56	0.62	0.56	0.56
2	Concat	0.55	0.59	0.55	0.55	0.58	0.64	0.58	0.58
	Mean	0.55	0.59	0.55	0.55	0.58	0.64	0.58	0.58
4	Concat	0.56	0.61	0.57	0.56	0.60	0.67	0.60	0.60
	Mean	0.57	0.61	0.57	0.57	0.60	0.67	0.60	0.60
8	Concat	0.56	0.61	0.57	0.56	**0.62**	0.71	**0.63**	**0.63**
	Mean	0.58	0.62	0.58	0.57	0.59	0.67	0.59	0.59
16	Concat	0.57	0.62	0.57	0.57	**0.62**	**0.72**	0.62	0.61
	Mean	**0.59**	0.63	**0.59**	0.58	0.58	0.67	0.59	0.59
32	Concat	**0.59**	**0.66**	**0.59**	**0.59**	0.60	0.71	0.61	0.60
	Mean	0.58	0.63	**0.59**	0.58	0.58	0.69	0.58	0.60

number of regions that yield their best results. While *GoogLeNet* obtains its best results with 8 and 16 regions, *AlexNet* obtains with 32 regions. We attribute this to the fact that *AlexNet* needs more features (regions) from different parts of the video to obtain good results, while *GoogLeNet* does not need that much since it has a more complex structure.

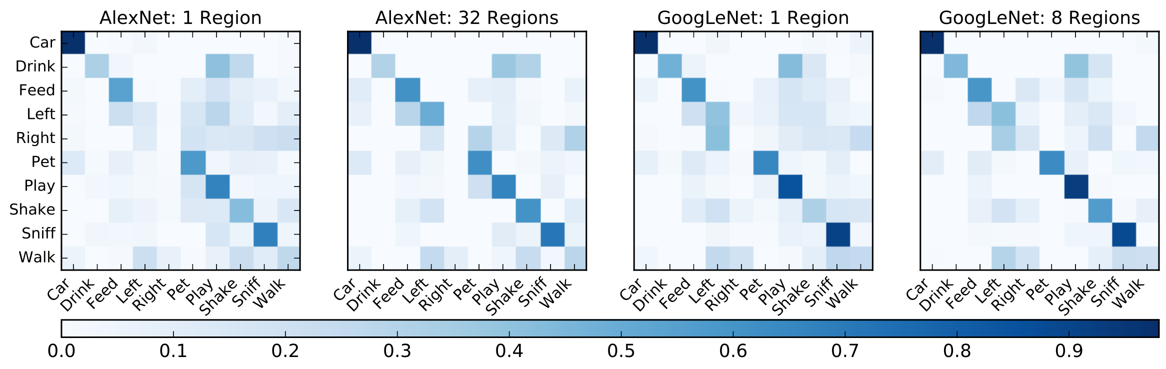

Fig. 3. Confusion matrix for *AlexNet* using 1 and 32 regions and *GoogLeNet* using 1 and 8 regions.

From all the confusion matrices, we can observe that the classes *Car* and *Sniff* have the highest scores in classification. The highest scores indicate that the classes are very different of the other classes and the features can be well separated from others. In fact, the class *Car*, for example, is the only class that contains a car passing by the dog. Besides, both classes have most number of samples in the dataset, which facilitates the network to learn their features.

4) Unbalanced classes: Analyzing the dataset we observe that it is indeed unbalanced, with the classes *Car* and *Sniff* having the largest number of frames (each containing $\approx 15\%$ of the frames of the dataset). On the other hand, *Left* class is only $\approx 4\%$ of the total frames in the dataset and *Right* class is $\approx 6\%$ of the total frames. This unbalanced nature of the dataset seems to impact the results, since the lower the number of frames the class has, the lower the accuracy score it achieves.

5. RELATED WORK

The increasing availability of wearable devices such as cameras, Google Glass[4], Microsoft SenseCam[5], *etc.* facilitates the low-cost acquisition of rich activity data. In recent years, this increase of available egocentric data have attracted a lot of attention in the computer vision community. The first-person point-of-view (or egocentric) is very popular in the study of daily activities [Ryoo et al. 2015; Hsieh et al. 2016], and many work have been proposed. For example, Ma et al. [2016] propose a method that uses twin-stream networks, one to capture the appearance of the objects such as hand regions and objects attributes, and the other to capture motion such as local hand movements and global head motion using stacked optical flow fields as input. A late fusion method joins both networks. Their experiments use GTEA and Gaze [Fathi et al. 2012] egocentric datasets.

Using first-person images and Inertial Measurement Units (IMU) sensors, Spriggs et al. [2009] propose an approach to make action segmentation and classification from brownie recipe videos. They use the Carnegie Mellon University Multimodal Activity dataset and manually annotate 29 actions in the videos. In order to segment actions, the authors use the data from IMU sensors as input for unsupervised algorithms that divide the video according to the actions. To classify actions, they test two supervised algorithms: Hidden Markov Models and K-Nearest Neighbours.

[4]https://www.google.com/glass/start/
[5]http://research.microsoft.com/en-us/um/cambridge/projects/sensecam/

Pirsiavash and Ramanan [2012] propose a two-step approach to identify actions from first-person videos. First, they identify the objects in the frames and classify those been used by the person as active. Then, they use the information from the objects to classify the activity been performed. The authors use a manually annotated dataset containing 1 million frames with activities of daily living recorded from an "egocentric" point of view. They annotated 18 action classes and 42 identified objects. To identify activities, they use models considering the presence of objects and when they are active.

Ryoo and Matthies [2013] introduce an approach to recognize interaction-level human activities from an egocentric point of view. First, they extract features from first-person videos, such as global and local motion descriptors. Then, they cluster similar descriptors and create a multi-channel kernels to compute video similarities. Using such representations, they train an SVM to classify the videos according to the correct activity.

Iwashita et al. [2014] focus on recognizing activities not only using wearable cameras attached to a person but also to animals, from DogCentric Activity dataset. They present several approaches to extract features, such as dense optical flow, binary pattern, cuboid feature detector and STIP detector. Using these hand-crafted features combined with a SVM they achieve the highest classification accuracy of 60.5%. Unlike Iwashita *et al.* , we do not use hand-crafted features to identify the activity, but instead, we train a end-to-end CNN.

Ryoo et al. [2015] develop a new feature representation named pooled time series (PoT), which intends to capture motion information in first-person videos by applying time series pooling of feature descriptors, detecting short-term/long-term changes in each descriptor element. Activity recognition is performed by training and testing a Support Vector Machine (SVM) classifier using these features, achieving 74% of accuracy when combined with INRIA's improved trajectory feature (ITF) [Wang and Schmid 2013] in the DogCentric Activity dataset.

6. CONCLUSIONS AND FUTURE WORK

In this work, we use features from single frames generated by a CNN to create temporal regions. Using such regions, we induce the temporal aspect of videos in order to recognize actions. Our model architecture is composed by a CNN (*AlexNet* or *GoogLeNet*), a method to separate and compute regions, and a classifier (SVM). Using images from the DogCentric Activity dataset, we perform experiments showing that our approach can improve the activity recognition task. We test our approach using two networks *AlexNet* and *GoogLeNet*, increasing up to 10% of Precision when using regions to classify activities.

For future work, we plan to test our approach in other datasets to verify its performance in different domains, as well as use other CNN architectures to ensure that our temporal regions can improve other models as well. Finally, we intend to compare our approach with other methods that consider the temporal aspect of the video, such as Long-Short Term Memory (LSTM) [Hochreiter and Schmidhuber 1997] and 3D CNN [Ji et al. 2013].

ACKNOWLEDGMENT

This paper was achieved in cooperation with HP Brasil Indústria e Comércio de Equipamentos Eletrônicos LTDA. using incentives of Brazilian Informatics Law (Law nº 8.2.48 of 1991).

REFERENCES

Chen, H., Chen, J., Hu, R., Chen, C., and Wang, Z. Action recognition with temporal scale-invariant deep learning framework. *China Communications* 14 (2): 163–172, 2017.

Crammer, K. and Singer, Y. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of machine learning research* vol. 2, pp. 265–292, 2001.

Fathi, A., Li, Y., and Rehg, J. M. Learning to recognize daily actions using gaze. In *European Conference on Computer Vision*. Springer-Verlag, Berlin, Heidelberg, pp. 314–327, 2012.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS'10*. Vol. 9. PMLR, Chia Laguna Resort, Sardinia, Italy, pp. 249–256, 2010.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation* 9 (8): 1735–1780, 1997.

Hsieh, P.-J., Lin, Y.-L., Chen, Y.-H., and Hsu, W. Egocentric activity recognition by leveraging multiple mid-level representations. In *2016 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, Seattle, WA, USA, pp. 1–6, 2016.

Iwashita, Y., Takamine, A., Kurazume, R., and Ryoo, M. S. First-person animal activity recognition from egocentric videos. In *2014 22nd International Conference on Pattern Recognition (ICPR)*. IEEE, Stockholm, Sweden, pp. 4310–4315, 2014.

Ji, S., Xu, W., Yang, M., and Yu, K. 3d convolutional neural networks for human action recognition. *TPAMI* 35 (1): 221–231, 2013.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., and Darrell, T. Caffe: Convolutional architecture for fast feature embedding. In *22nd ACM international conference on Multimedia*. ACM, New York, NY, USA, pp. 675–678, 2014.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. Large-scale video classification with convolutional neural networks. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*. CVPR '14. IEEE Computer Society, Washington, DC, USA, pp. 1725–1732, 2014.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*. NIPS'12. Curran Associates Inc., USA, pp. 1097–1105, 2012.

Ma, M., Fan, H., and Kitani, K. M. Going deeper into first-person activity recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Las Vegas, Nevada, USA, 2016.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* vol. 12, pp. 2825–2830, 2011.

Pirsiavash, H. and Ramanan, D. Detecting activities of daily living in first-person camera views. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, Providence, RI, USA, pp. 2847–2854, 2012.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Las Vegas, Nevada, USA, pp. 779–788, 2016.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. ImageNet Large Scale Visual Recognition Challenge. *IJCV* 115 (3): 211–252, 2015.

Ryoo, M. S. and Matthies, L. First-person activity recognition: What are they doing to me? In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

Ryoo, M. S., Rothrock, B., and Matthies, L. Pooled motion features for first-person videos. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Boston, MA, USA, pp. 896–904, 2015.

Spriggs, E. H., Torre, F. D. L., and Hebert, M. Temporal segmentation and activity classification from first-person sensing. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, Miami, FL, USA, pp. 17–24, 2009.

Sukthankar, G., , Geib, C., , Bui, H. H., , Pynadath, D. V., , and Goldman, R. P. *Plan, Activity, and Intent Recognition*. Morgan Kaufmann, Boston, 2014.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, Boston, MA, USA, 2015.

Wang, H. and Schmid, C. Action recognition with improved trajectories. In *2013 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Sydney, NSW, Australia, pp. 3551–3558, 2013.