Human Action Recognition in Videos using a Robust CNN LSTM Approach

(Reconocimiento de Acciones Humanas en Videos usando una Red Neuronal CNN LSTM Robusta)

Carlos Ismael Orozco,[1] Eduardo Xamena,[2] María Elena Buemi[3] & Julio Jacobo Berlles[4]

Campo temático: Machine Learning.

Abstract

Action recognition in videos is currently a topic of interest in the area of computer vision, due to potential applications such as: multimedia indexing, surveillance in public spaces, among others. In this paper we propose (1) The implementation of a CNN–LSTM architecture. First, a pre-trained VGG16 convolutional neural network extracts the features of the input video. Then, an LSTM classifies the video sequence in a particular class. (2) A study of how the number of LSTM units affects the performance of the system. To carry out the training and test phases, we used the KTH, UCF-11 and HMDB-51 datasets. (3) An evaluation of the performance of our system using accuracy as evaluation metric, given the existing balance of the classes in the datasets. We obtain 93%, 91% and 47% accuracy respectively for each dataset, improving state of the art results for the former two. Besides the results attained, the main contribution of this work lays on the evaluation of different CNN-LSTM architectures for the action recognition task.

Keywords: action recognition; convolutional neural network; long short-term memory.

[1] Universidad Nacional de Salta. ciorozco.unsa@gmail.com
[2] Universidad Nacional de Salta. dreduardoxamena@gmail.com
[3] Universidad de Buenos Aires. mebuemi@dc.uba.ar
[4] Universidad de Buenos Aires. jacobo@dc.uba.ar

Resumen

El reconocimiento de acciones en videos es actualmente un tema de interés en el área de visión por computadora, debido a potenciales aplicaciones como: indexación multimedia, vigilancia en espacios públicos, entre otras. En este artículo proponemos: (1) Implementar una arquitectura CNN–LSTM para esta tarea. Primero, una red neuronal convolucional VGG16 previamente entrenada extrae las características del video de entrada. Luego, una capa LSTM determina la clase particular del video. (2) Estudiar cómo la cantidad de unidades LSTM afecta el rendimiento del sistema. Para llevar a cabo las fases de entrenamiento y prueba, utilizamos los conjuntos de datos KTH, UCF-11 y HMDB-51. (3) Evaluar el rendimiento de nuestro sistema utilizando la precisión como métrica de evaluación, dado el balance existente entre las clases de los conjuntos de datos. Obtenemos un 93%, 91% y 47% de precisión respectivamente para cada conjunto de datos, mejorando los resultados del estado del arte para los primeros dos. Además de los resultados obtenidos, la principal contribución de este trabajo yace en la evaluación de diferentes arquitecturas CNN-LSTM para la tarea de reconocimiento de acciones.

Palabras claves: reconocimiento de acciones; redes neuronales convolucionales; redes neuronales de corta y larga memoria.

1 Introduction

The action recognition problem in videos is of great interest in the area of pattern recognition and computer vision due to its potential applications such as: multimedia indexation, information recovery, patient monitoring and control, automated surveillance in public spaces, human-computer interaction, among others. The objective of the action recognition systems is to classify each video into the class that represents the action that happens in the video. To this end, the interactions between the subjects and/or objects within it, must be taken into account. This problem has been investigated by other works:

(Liu et al., 2013) propose a framework for detecting and recognizing human actions. To achieve a robust estimation of the region of interest, they use a combination of optical flow together with a Harris 3D edge detector to obtain space-time information from video. Then, with the calculation of the local features SIFT and STIP, they train a universal background model (UBM) for the task in question.

(Wang et al. 2011) propose a dense trajectory approach. They take dense points in each frame of the video and track based on the optical flow displacement information.

(Sharma et al. 2015) propose a model based on attention mechanisms for the task of recognizing actions in videos. They use an LSTM neural network contemplating the spatial and temporal aspect of the video.

Attention mechanisms have become a very important concept in deep learning. Its operation tries to imitate the visual capacity of people that allows you to focus your attention on relevant parts of a scene to extract relevant information, resulting in a better generalization. From this approach we can highlight (Bahdanau et al. 2014) as well as a (Li et al., 2020) for temporal attention mechanism.

(Meng et al. 2019) proposes an interpretable and easy-to-connect spatio-temporal attention mechanism. Learn a featured mask to focus on the salient features in the spatial domain and employ a convolutional LSTM-based attention mechanism to identify the most relevant frames in the time domain.

The objective of this work is to implement a video action recognition system. For this we propose the use of a CNN–LSTM architecture. A convolutional neural network extracts the features of the video while an LSTM neural network classifies the video into a certain category. The work is organized as follows: in section 2, the general structure of the system is described; in section 3, the databases used, the evaluation method, the experiments carried out and the results obtained are explained. Finally, in section 4 the conclusions and future work are presented.

2 Robust CNN-LSTM Approach

In this paper we propose the use of a CNN–LSTM architecture. Figure 1 shows a general scheme of the system in its different stages. Input: the video is normalized for a total of 40 frames. CNN: A pre-trained VGG16 extracts the characteristics of the video by obtaining features of size 40 × 25088. LSTM: takes each vector from the previous stage and processes it in n_u LSTM units. Finally, the output stage consists of a dense layer with n_c nodes, one for each class. Section 3 shows in detail the implementation proposed. Also, in the following repository you can find code for the architecture proposed.[5]

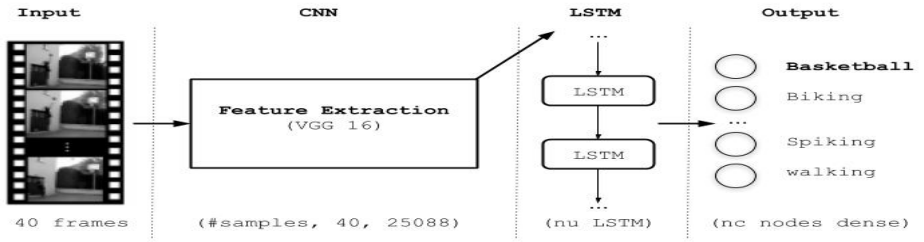

Figure 1: System architecture. The video is normalized to 40 frames. A CNN VGG16 extracts the features. Then an LSTM alongside dense layers of nc classifies a certain class, for example basketball, biking, and others.

2.1 Convolutional Neural Network

CNNs are composed of an ordered set of layers, each of them, in turn, consists of processing units that operate on the output of the previous layer. The layers used are: (a) convolutional layer: has k filters (or kernels) dedicated to produce k feature maps. (b) subsampling layer: each feature map is sub-sampled by means of a max-pooling operation, a process that progressively reduces the spatial size of the representation and the number of parameters to be trained. (c) Dense layer: fully connected layers with the purpose of classifying the input image into one of several classes, according to the training data set.

The convolutional architecture of VGG16 proposed by (Simonyan et al., 2014) obtained very good results in the ImageNet Large-Scale Visual Recognition Challenge - ILSVRC-2014 Classification and location tasks. Figure 2 shows the layers that make up this architecture.

[5] https://gitlab.com/ciorozco/actionrecognition-cnn-lstm

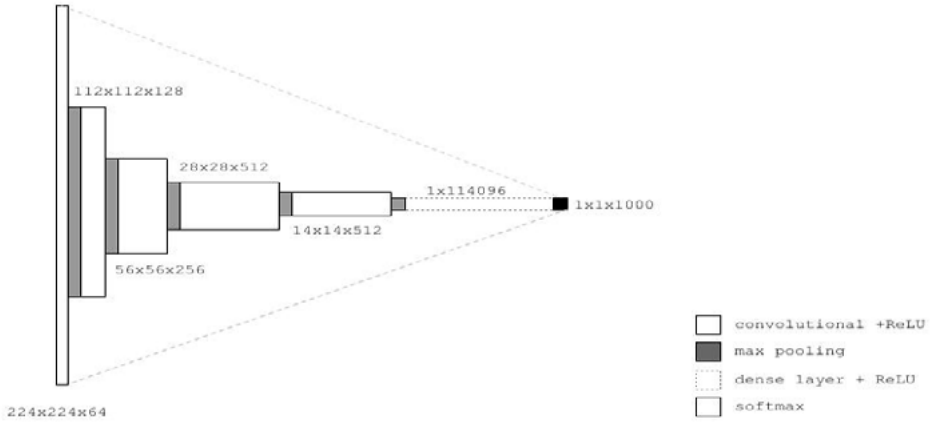

Figure 2: VGG16 pre-trained implemented in the library Keras (Chollet et al. 2015).

The input of the VGG16 is a fixed size image of 224×224×3. This input image is passed through a stack of convolutional layers (boxes that use ReLus as activation functions). The convolutional layers are usually accompanied by max-pooling layers (max pooling boxes) then two dense layers (boxes fully connected and ReLu as activation function) of 4096 nodes each. Finally, a dense layer (boxes with Softmax activation function) of 1000 nodes, yields the output of this CNN.

2.2 Long Short-Term Memory

The neural networks LSTMs (Hochreiter et al., 1997) are a special type of recurrent neural network (RNN) that are formulated in such a way that remembering information for long periods of time is their natural behavior. The entries for time step t are: input xt, previous output h_{t-1} and previous memory c_{t-1}. The outputs are: current output ht and current memory c_t.

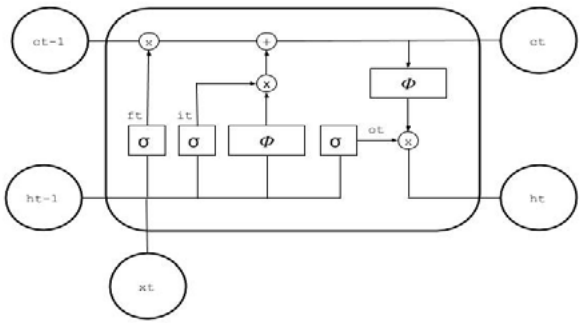

Figure 3: LSTM unit.

To calculate ht and ct the unit LSTM is modulated by three types of gates, following the internal structure shown in Figure 3. They are calculated as:

1. Input gate i: Controls whether the current entry xt is considered.

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right)$$

2. Forget gate f: Allows the LSTM to forget the previous memory c_{t-1}.

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

3. Output gate o: Decides how much memory will be transferred to the hidden state h_t.

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

Finally:

$$c_t = f_t \otimes c_{t-1} \oplus i_t \otimes \varphi(W_{xc}x_t + W_{hc}h_{t-1} + b_c)$$
$$h_t = o_t \otimes \varphi(c_t)$$

Where: σ is the sigmoidal function, φ is the hyperbolic tangent, □ represents the product with the value of the gate and the weights of the matrix denoted by W_{ij}.

3 Experiments and Results

Our system was implemented in Python using the library Theano (Bastien et al., 2012), (Bergstra et al., 2010) on an Intel CORE i7-6700HQ computer with 16GB DDR3 memory and Ubuntu Operating System 16.04. The experiments were carried out on an NVIDIA Titan Xp GPU mounted on a server with a similar configuration.

Algorithm 1 shows the implementation of the features extraction. All the videos in the database were re-sized to 224 × 224 shape to feed the VGG16. We set a maximum number of 40 frames per video. When the number of frames is less than 40 we complete the process by replicating the last frame. We carried out a vectorization of the max pooling of layer 10 obtaining a vector of dimension 7 × 7 × 512 = 25088 to feed the neural network LSTM.

Algorithm 1 Features extraction

```
Require: videos, labels, VGG16, #frames=40
Ensure: X, y
X = [ ], y = [ ]
for (video, label) in (videos, labels) do
    sample = [ ]
    for frame in video (not exceed #frames) do
      frame.resize((224, 224))
      feature = VGG16(frame, layer=10)
      sample.append(feature.flatten())
      end for
      X.append(sample)
   y.append(label)
 end for
```

Algorithm 2 shows the LSTM model implemented together with the supervised training configuration. As input parameter it receives: X train of size #samples × 40 × 25088, and y train of size #samples × nc. The network parameters are optimized by minimizing the cross-entropy loss function using stochastic gradient descent with the update rule RMSProp (Dauphin et al., 2015).

Algorithm 2 Training of LSTM Neural Network model.

```
Require: X train, y train, nu, epochs=50, batch size=25
Ensure: model
nc = y train.get classes()
model = new neural network()
model.add(LSTM(nu, input shape=(40, 25088)))
model.add(Dense(nc, activation = 'sigmoide'))
model.compile(optimizer='RMSProp', metrics=['accuracy']))
model.fit(X train, y train, epochs, batch size)
```

The experiments carried out in this work use the KTH dataset proposed by (Schuldt et al., 2004), UCF-11 database proposed by (Liu et al., 2009) and HMDB-51 proposed by (Kuehne et al., 2011).

3.1 KTH

KTH (Schuldt et al., 2004) has 599 videos that belong to one of the following nc = 6 classes: walking, jogging, running, boxing, waving and clapping. For each class, several videos of 25 people are captured in four different scenarios (indoors, outdoors, outdoors with scale variation and outdoors with different clothes). To report results we follow the original configuration (Schuldt et al., 2004) using as training set 16 people and as test set 9 people. To determine the number of LSTM units we carry

out a variation of these quantities and evaluate its accuracy. In Figure 4, an ROC curve (Receiver Operating Characteristic) is shown for the system output accuracy, varying the number of LSTM units. For n_u = 360 LSTM units the precision obtained was 93.86% (best result). Table 1 summarizes the results obtained by our system compared with other systems included in the bibliography. Our result is better than those proposed by (Baccouche et al., 2011), (Jones et al., 2012) and (Liu et al., 2013).

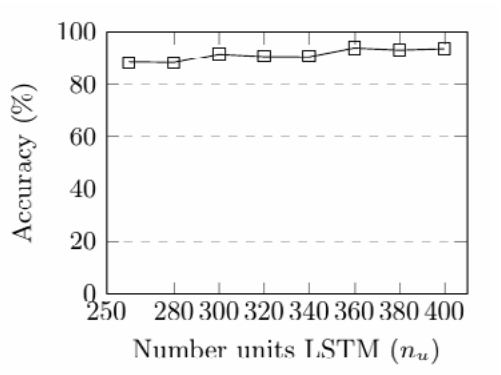

Figure 4: ROC graphic for the system with varying number of LSTM units against Accuracy.

Table 1: Results of the video classification (with nu = 360) using the database KTH (Schuldt et al., 2004).

Approach	Acc (%)
(Baccouche et al., 2011)	91.04%
(Liu et al., 2013)	93.67%
(Jones et al., 2012)	93.20%
CNN–LSTM (Our proposal)	93.86%

3.2 UCF-11

UCF-11 (Liu et al., 2009) has 1600 videos that belong to one of the following nc = 11 classes: basketball, biking, diving, golf swing, horse riding, soccer juggling, swing, tennis swing, trampoline jumping, volleyball spiking and walking. Performance measure is calculated by average accuracy over all classes, using configuration as the original (Liu et al., 2009), and we use leave one out cross validation (LOOCV) for a predefined set of 25 folds. Figure 5 shows the relationship between number of units and final precision. For n_u = 320 LSTM units the obtained precision was 91.94% (best result). Table 2 summarizes the results obtained by our system compared to the systems mentioned in the bibliography. Our result is better than the one proposed by (Wang et al. 2011), (Sharma et al. 2015), (Liu et al., 2009), (Liu et al., 2013) and (Cho et al., 2014).

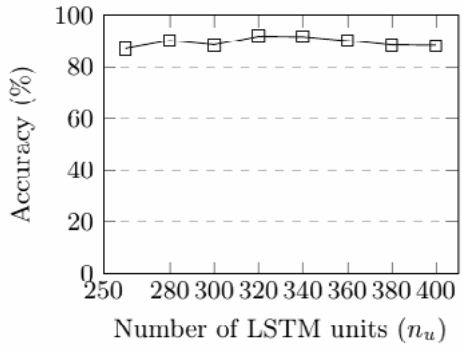

Figure 5: ROC graphic for the system with varying number of LSTM units against Accuracy.

Table 2: Results of the video classification (with nu = 320) using the database UCF-11 (Liu et al., 2009).

Approach	Acc (%)
(Liu et al., 2009)	71.02%
(Liu et al., 2013)	76.1%
(Wang et al. 2011)	84.2%
(Sharma et al. 2015)	85.0%
(Cho et al., 2014)	88.0%
CNN–LSTM (Our proposal)	91.94%

3.3 HMDB-51

HMDB-51 Human Motion dataset (Kuehne et al., 2011) has 6766 videos that belong to one of the following $n_c = 51$ classes: clap, drink, hug, jump, somersault and many others. Also provides three train-test splits, each consisting of 5100 videos, 3570 videos for training and 1530 videos for test, i.e a ratio of 70/30 by class. We evaluate the average accuracy over the 3 splits. Figure 6 shows the ROC curve between the number of LSTM units and final precision. For $n_u = 400$ LSTM units, the precision obtained was 47.36% (best result). Table 3 summarizes the results obtained by our system compared to other results in the bibliography. Our result is better than the ones obtained by the systems proposed by (Sharma et al. 2015), (Kuehne et al., 2011), (Jiang et al. 2012) and (Kliper et al., 2012).

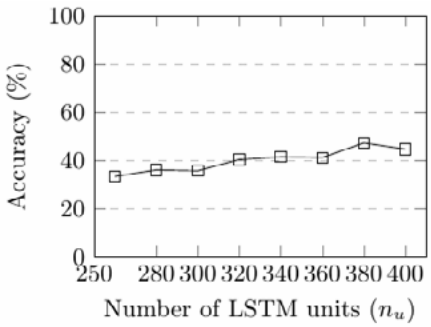

Figure 6: ROC graphic for the system with varying number of LSTM units against Accuracy.

Table 3: Results of the video classification (with nu = 320) using the database HMDB-51 (Kuehne et al., 2011).

Approach	Acc (%)
(Kuehne et al., 2011)	23.0%
(Kliper-Gross et al., 2012)	29.2%
(Jiang et al. 2012)	40.7%
(Sharma et al. 2015)	41.3%
CNN–LSTM (Our proposal)	47.36%

Figure 7 shows output examples of our system for the datasets KTH (left), UCF-11 (middle) and HMDB-51 (right) respectively. Each section shows a set of video frames (upper part) and a top-3 of the classes with the highest score (lower part). Our system yields the class with the highest score, this is the most probable class, as output.

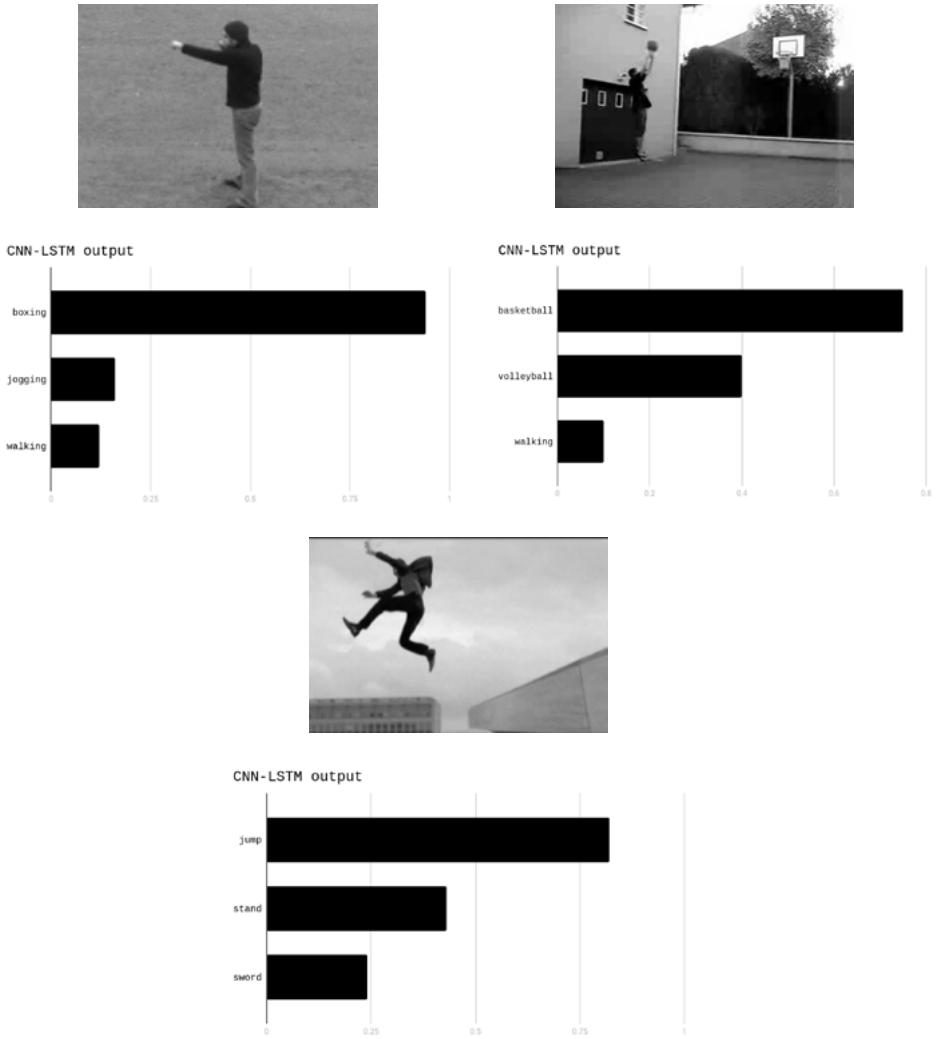

Figure 7: Top-Left: Example for KTH: This video shows a person boxing. Our system classifies it into the class "boxing", as it reaches the score 94% (maximum). Button-Middle: Example for UCF-11: This video shows a young man throwing a ball to the basket. Our system classifies it into the class "basketball" (score 75%). Top-Right: Example for HMDB-51: This video shows a person jumping. Our system classifies it into the class "jump" (score 82%).

4 Conclusions and Future Work

In this work we implement a video action recognition system, using a CNN–LSTM neural network. First, a VGG16 extracts the characteristics of the video. Then an LSTM neural network classifies the scene into the class it belongs to.

The framework was implemented in Python using the library Theano (Bastien et al., 2012) and (Bergstra et al., 2010), trained and tested using the databases (Schuldt et al., 2004), (Liu et al., 2009) and (Kuehne et al., 2011), and performed on an NVIDIA Titan Xp GPU. The influence of the number of LSTM units over the system performance was studied. The ROC curve has been presented for each dataset that we have proposed. The best precision value was obtained for dataset KTH (Schuldt et al., 2004) with 93% for $n_u = 360$ units. A closer value was the result of the experiments carried out on the dataset UCF-11 (Liu et al., 2009) with a precision of 91% for $n_u = 320$ units. Finally, for $n_u = 400$ units, the dataset HMDB-51 (Kuehne et al., 2011) has achieved a precision of 47%. Defining the number of LSTM units allowed us to better adjust our model, also allowing us to avoid overfitting it in its training stage.

Regarding the explained performance results, the most remarkable contribution of our work is the considerable improvement on the action recognition task on the KTH and UCF-11 datasets, and the evaluation of different numbers of LSTM units. KTH and UCF-11 datasets have lower sizes than HMDB-51, and perhaps neural models simpler than very deep or intricated architectures as transformers can provide good results, without the need of compute-intensive and time-consuming training and testing cycles.

As future work the use of other databases will be considered, such as Hollywood2 (Marszalek et al., 2009), UCF-50 (Reddy et al., 2013) and UCF-101 (Soomro et al., 2012) to make the system more robust. Another goal is to implement the attention mechanisms proposed by (Bahdanau et al., 2014), (Laokulrat et al., 2016) and temporal attention mechanism proposed by (Li et al., 2020). as well as the transformer approach to the problem in question (Girdhar et al., 2019) and we are going to delve into techniques to avoid overfitting.

Acknowledgement

The authors thank NVIDIA for the donation of a TITAN Xp GPU for Departamento de Informática. Facultad de Ciencias Exactas. Universidad Nacional de Salta, Argentina.

References

Liu D., Shyu M., and Zhao G. (2013). Spatial-temporal motion information integration for action detection and recognition in non-static background. In 2013 IEEE 14th International Conference on Information Reuse Integration (IRI), pages 626–633, Aug 2013.

Li J., Liu X., Zhang W., Zhang M., Song J. and Sebe N. (2020). Spatio-Temporal Attention Networks for Action Recognition and Detection, in IEEE Transactions on Multimedia, doi: 10.1109/TMM.2020.2965434.

Wang H., Klser A., Schmid C., and Liu C. (2011). Action recognition by dense trajectories. In CVPR 2011, pages 3169–3176.

Girdhar R., et al. Video action transformer network. En Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019. p. 244-253.

Sharma S., Kiros R., and Salakhutdinov R. (2015). Action recognition using visual attention. CoRR, abs/1511.04119.

Simonyan K. and Zisserman A. (2014). Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556.

Chollet F. et al. Keras. https://keras.io, 2015.

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

Bastien F., Lamblin P., Pascanu R., Bergstra J., Goodfellow I., Bergeron A., Bouchard N., and Bengio B. (2012). Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop.

Bergstra J., Breuleux O., Bastien F., Lamblin P., Pascanu R., Desjardins G., Turian J., Warde-Farley D., and Bengio Y. (2010). Theano: a CPU and GPU math expression compiler. In Proceedings of the Python for Scientific Computing Conference (SciPy).

Dauphin Y., Harm de Vries, and Bengio Y. (2015). Rmsprop and equilibrated adaptive learning rates for non-convex optimization. In NIPS. corr abs/1502.04390.

Schuldt C., Laptev I. and Caputo B. (2004). Recognizing human actions: a local svm approach. In Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004., volume 3, pages 32–36 Vol.3.

Liu J., Luo J., and Shah M. (2009). Recognizing realistic actions from videos in the wild. In 2009 IEEE Conference on Computer Vision and Pattern Recognition (pp. 1996-2003). IEEE.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., & Serre, T. (2011). HMDB: A large video database for human motion recognition. In 2011 International Conference on Computer Vision (pp. 2556–2563). IEEE. https://doi.org/10.1109/ICCV.2011.6126543

Baccouche M., Mamalet F., Wolf C., Garcia C., and Baskurt A. (2011). Sequential Deep Learning for Human Action Recognition. In B. Lepri A.A. Salah, editor, 2nd International Workshop on Human Behavior Understanding (HBU), pages 29–39, Amsterdam, Netherlands, Springer.

Cho J., Lee M., Chang H, and Oh S. (2014). Robust action recognition using local motion and group sparsity. Pattern Recognition, 47(5):1813 – 1825.

Jiang Y., Dai Q., Xue X., Liu W., and Ngo C. (2012). Trajectory-based modeling of human actions with motion reference points. In European Conference on Computer Vision, pages 425–438.

Jones S., Shao L., Zhang J., and Liu Y. (2012). Relevance feedback for real-world human action retrieval. Pattern Recognition Letters, 33(4):446 – 452, 2012. Intelligent Multimedia Interactivity.

Kliper-Gross O., Gurovich Y., Hassner T., and Wolf L. (2012). Motion interchange patterns for action recognition in unconstrained videos. In European Conference on Computer Vision (ECCV).

Soomro K., Zamir A., and Shah M. (2012). UCF101: A dataset of 101 human actions classes from videos in the wild. CoRR, abs/1212.0402, 2012.

Marszalek M., Laptev I., and Schmid C. (2009). Actions in context. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 2929–2936.

Meng L., Zhao B., Chang B., Huang G., Sun W., Tung F. and Sigal L. Interpretable spatio-temporal attention for video action recognition. InProceedings of the IEEE international conference on computer vision workshops, pages 0–0, 2019.

Reddy K. and Shah M. (2013). Recognizing 50 human action categories of web videos. Mach. Vision Appl., 24(5):971–981, July 2013.

Laokulrat N., Phan S., Nishida N., Shu R., Ehara Y., Okazaki N., Miyao Y., and Nakayama H. (2016). Generating video description using sequence-to-sequence model with temporal attention. In COLING 2016, 26th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, December 11-16, Osaka, Japan, pages 44–52.

Bahdanau D., Cho K., and Bengio Y. (2014). Neural machine translation by jointly learning to align and translate. CoRR, abs/1409.0473, 2014.