Sapienza University of Rome

Vision and Perception
Mask R-CNN and Activity Recognition

Ivan Bergonzani, Michele Cipriano,
Ibis Prevedello, Jean-Pierre Richa

July 10, 2018

1 Introduction

The aim of the project is to train a model based on Mask R-CNN[1] using an extended version of COCO that includes the dataset created on Labelbox and use it to train another network that recognizes activities from videos. The new dataset consists on a bunch of images that shows the Gymnastic activities of ActivityNet. All the images have been downloaded from Google using a Python tool called `google-images-download`. The idea is to have a working model that will be used to classify videos that show Gymnastic activities. This second task is achieved by training a LSTM on the frames taken from the videos themselves.

The project has been developed in Python and it has been tested using Google Compute Engine. The final training of Mask R-CNN has been performed at Alcor lab, while the training of the LSTM has been performed on a Google Compute Engine instance.

2 Mask R-CNN Training

Mask R-CNN has a set of losses that are used to check the performances of the classification, the RPN, the regression on the bounding boxes and the instance segmentation:

- `smooth_l1_loss`: the smooth-L1 loss on the classification of the objects.
- `rpn_class_loss`: the loss on the classification of the object contained in the region proposals, they can either be foreground if there is an object inside or background otherwise.
- `rpn_bbox_loss`: the loss on the bounding box returned by the RPN.
- `mrcnn_class_loss`: the loss for the classifier head of Mask R-CNN.
- `mrcnn_bbox_loss`: the loss for the bounding box refinement at the end of the network.
- `mrcnn_mask_loss`: the binary cross-entropy loss for the masks.

It is possible to see the graphs of these losses using tensorboard once the training is complete. The network has been trained for 35 epochs for the first stage, 30 epochs for the second and 15 epochs for the third stage, being able to recognize most of the objects in the videos. The dataset used has been those created from LabelBox that is summarized in table 2.1. The dataset has been split in 90% for the training set and 10% for the test set. The network, when using `DETECTION_MIN_CONFIDENCE=0.9`, obtained a number of 127 successes out of 169 labels with a success percentage of **0.751**.

3 Activity Recognition

Once Mask R-CNN training is complete it is possible to use the resulting model to extract features and informations from the frames of the videos.

To do this, each video has been preprocessed by removing the first 10% and the last 10% of the frames that likely contain frames that do not contribute to the recognition of the video (e.g. advertisement). From the remaining part of the video, 40 frames have been extracted uniformly and saved. Then, each frame has been fed to the Mask R-CNN model obtained previously. The output masks and the output labels coming from Mask R-CNN have been

Activity	Labels
Doing step aerobics	268
Elliptical trainer	177
Spinning	158
Using parallel bars	241
Using the balance beam	383
Using the pommel horse	248
Using the rowing machine	290
Using uneven bars	253
Total	**2018**

Table 2.1: Number of masks segmented in LabelBox for each activity in Gymnastics.

used to build the tensors to be fed to the LSTM. In order to speed up the training of the activity recognizer, these data are saved in a separate file so that they can be used without going through Mask R-CNN again.

The problem is, hence, reduced to training a LSTM. Multiple experiments have been performed on the features extracted from Mask R-CNN, all summarized in table 3.2. In particular the trainings focused on the number of hidden nodes for the LSTM and the type of features extracted. The number of hidden nodes varied from 64 to 1024. Three types of datasets have been created by extracting the features from the videos contained in the dataset Gymnastics, summarized in table 3.1. The dataset Count contains a vector that counts the number of objects contained in each frame of each video. The dataset Masks contains a tensor with all the masks recognized by Mask R-CNN. The dataset Masks+Count combines the two, with the idea of exploiting both kinds of data. Each dataset has been split in 80% for the training set and 20% for the test set.

The structure of the network is pretty simple. It consists of a LSTM followed by a dropout layer (with probability 0.2) that aims to avoid overfitting. The LSTM considers a timestep of 40 and outputs a softmax over the classes of the videos contained in Gymnastics. Since the LSTM requires a 1D vector as input, the masks are vectorized before being fed to the LSTM itself. This makes the masks to lose the spatiality of the data, making it harder to train the network.

It is interesting to study how the accuracy and the loss varies with the

Num.	Activity	Videos
1	Doing crunches	62
2	Doing step aerobics	78
3	Elliptical trainer	79
4	Kneeling	92
5	Rope skipping	101
6	Running a marathon	81
7	Spinning	78
8	Tumbling	61
9	Using parallel bars	100
10	Using the balance beam	105
11	Using the pommel horse	64
12	Using the rowing machine	68
13	Using uneven bars	63
14	Zumba	61
	Total	**1093**

Table 3.1: Number of videos for each activity contained in Gymnastics.

epochs changing the type of data that is used to train the network and the number of hidden nodes in the LSTM. More in detail, each neural network is trained for two stages that differs in the value of the learning rate and the number of epochs. All the experiments used a learning rate of 10^{-3} for the first stage and 10^{-4} for the second stage. The number of epochs have been chosen so that there is no overfitting in the test set.

Figures 3.1 and 3.2 show how the accuracy changes over the epochs when using only the masks as input of the network. The networks are able to learn the activities represented by the videos, achieving a higher accuracy more quickly on the training set when using a larger number of hidden nodes. This is somehow reflected also on the test set, even if the difference is not large as in the training set. The same thing applies when using both masks and the count of the objects in each frame (figures 3.3 and 3.4). For this second experiment the LSTM manages to obtain slightly better results in the test set. When using only the count of the objects (figures 3.5 and 3.6) the networks manage to perform better in the test set with respect to the previous two experiments. Nevertheless it is not able to achieve a high accuracy in the training set.

Figures 3.7 and 3.8 shows the confusion matrices obtained when using an LSTM of 512 hidden layers with only the masks of the objects. Regarding the training set, the main problem of the network is on the recognition of the activity "Using parallel bars", that is often confused with "Using the pommel horse". On the test set, the network never manages to classify a video as "Tumbling". The reasons for this behaviour is due in particular to the extraction of the masks when using Mask R-CNN. Hence, an improvement of Mask R-CNN would improve the quality of the dataset that would, in turn, improve the accuracy of the LSTM. Moreover, the network does not guarantee that it is able to extract at least one mask for every frame of the video, hence, using more frames, or selecting better frames for each video would improve the accuracy of the LSTM, increasing the accuracy of the activity classification. This, however, is very expensive from a computational point of view, since it would require going through Mask R-CNN more often.

The structure of the network could be improved by considering the spatial relations of the frames. In fact, since the LSTM requires an input a 1D vector, all the frames have been vectorized. One possibility is to replace the LSTM with a Grid LSTM[2], that accept as input a tensor of any dimension, like in the case of the frames.

4 Implementation

The project has been implemented in Python using Tensorflow and matterport/Mask-R_CNN. Below, the list of Python files implemented for the project with a brief description of their behaviour.

- **activity.py:** training and evaluation of Mask R-CNN on an extended version of COCO dataset that considers the dataset created on LabelBox during the semester.
- **common.py:** common information of the dataset that are used by multiple programs inside the project.
- **count_zeros.py:** a tool to count the number of frames extracted from the video that do not have masks when fed to Mask R-CNN.
- **eval_videos.py:** saves the confusion matrix using a dataset of videos and a pretrained model.

Network	Dataset	Stage 1	Stage 2	Train Acc.	Test. Acc.
LSTM-64	Count	250	300	0.499	0.516
LSTM-128	Count	250	300	0.505	0.512
LSTM-256	Count	250	300	0.525	**0.532**
LSTM-512	Count	250	300	**0.619**	0.488
LSTM-1024	Count	250	300	0.616	0.500
LSTM-64	Masks	15	50	0.735	0.391
LSTM-128	Masks	15	50	0.869	0.435
LSTM-256	Masks	15	50	0.914	0.419
LSTM-512	Masks	15	50	**0.928**	**0.452**
LSTM-1024	Masks	15	50	0.924	0.427
LSTM-64	Masks+Count	15	50	0.706	0.419
LSTM-128	Masks+Count	15	50	0.865	0.456
LSTM-256	Masks+Count	15	50	0.913	**0.492**
LSTM-512	Masks+Count	15	50	0.921	0.423
LSTM-1024	Masks+Count	15	50	**0.928**	0.435

Table 3.2: The table shows the results of the experiments. Different number of hidden nodes have been tried for the LSTM. All the datasets contain different informations from the frames of the videos: Count represents each frame as a vector containing the number of objects in the frame itself, Masks represents each frame as a semantic map, Masks+Count combines the two. Note that since the LSTM requires a 1D vector as input, all the datasets have been vectorized before being fed to the LSTM. Note that even if the number epochs is higher for the datase Count, it takes less time to train w.r.t. the other two datasets. For example, when using LSTM-512 it takes **4m59s** on Count and **16m25s** on Masks on an instance of Google Compute Engine with a Tesla K80.

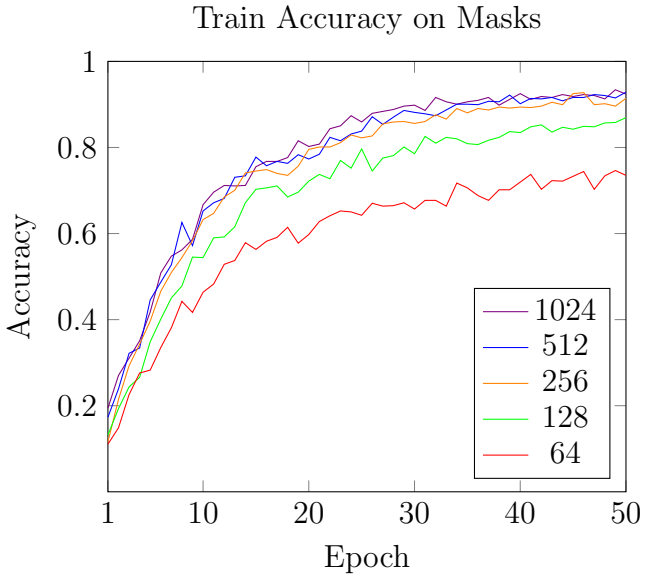

Figure 3.1: The plot shows the how the train accuracy varies for the dataset Masks changing the number of hidden nodes in the LSTM.

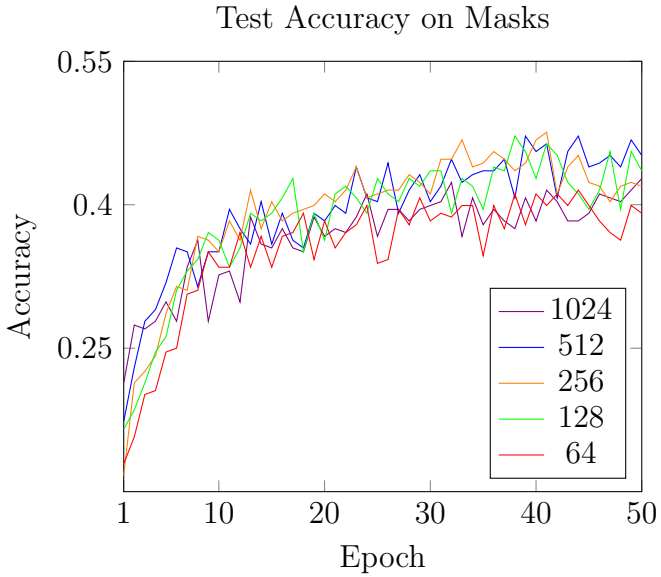

Figure 3.2: The plot shows the how the test accuracy varies for the dataset Masks changing the number of hidden nodes in the LSTM.

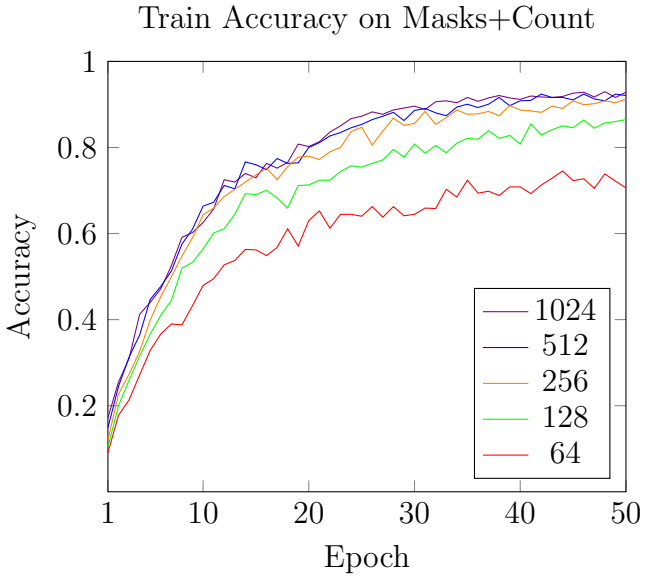

Figure 3.3: The plot shows the how the train accuracy varies for the dataset Masks+Count changing the number of hidden nodes in the LSTM.

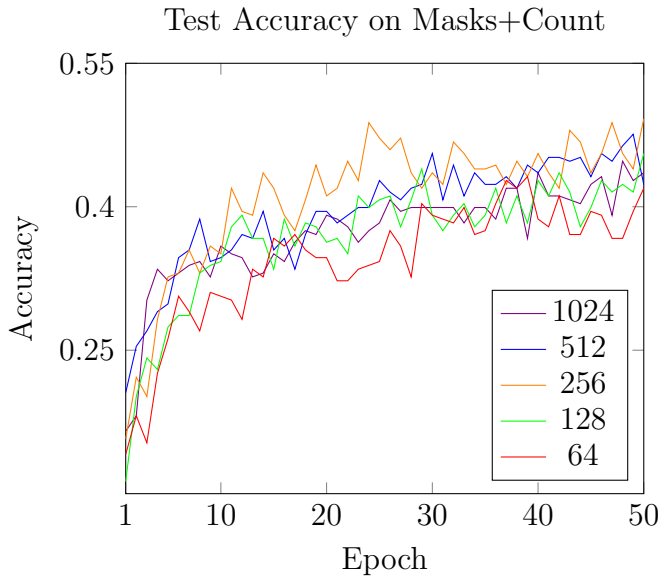

Figure 3.4: The plot shows the how the test accuracy varies for the dataset Masks+Count changing the number of hidden nodes in the LSTM.

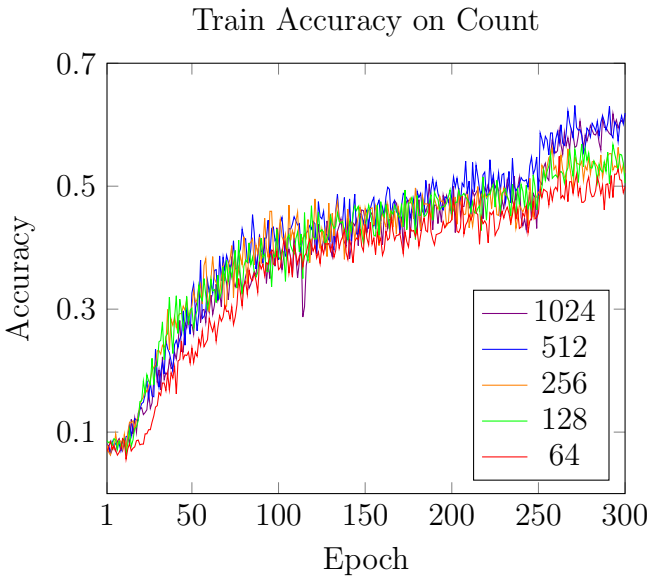

Figure 3.5: The plot shows the how the train accuracy varies for the dataset Count changing the number of hidden nodes in the LSTM.

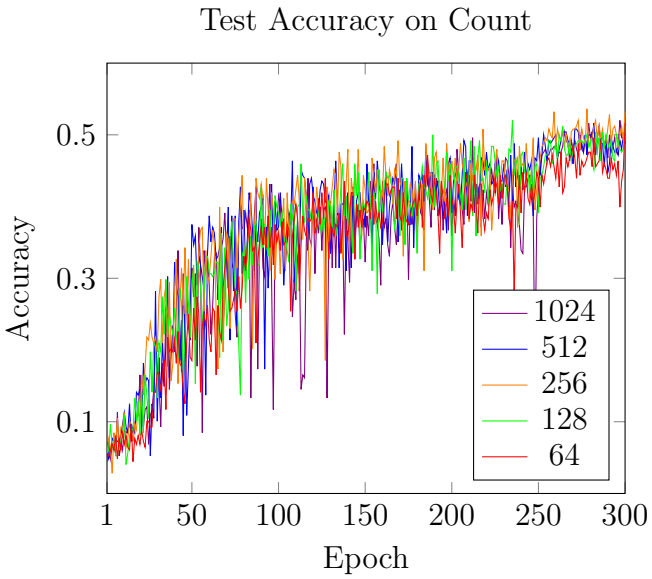

Figure 3.6: The plot shows the how the test accuracy varies for the dataset Count changing the number of hidden nodes in the LSTM.

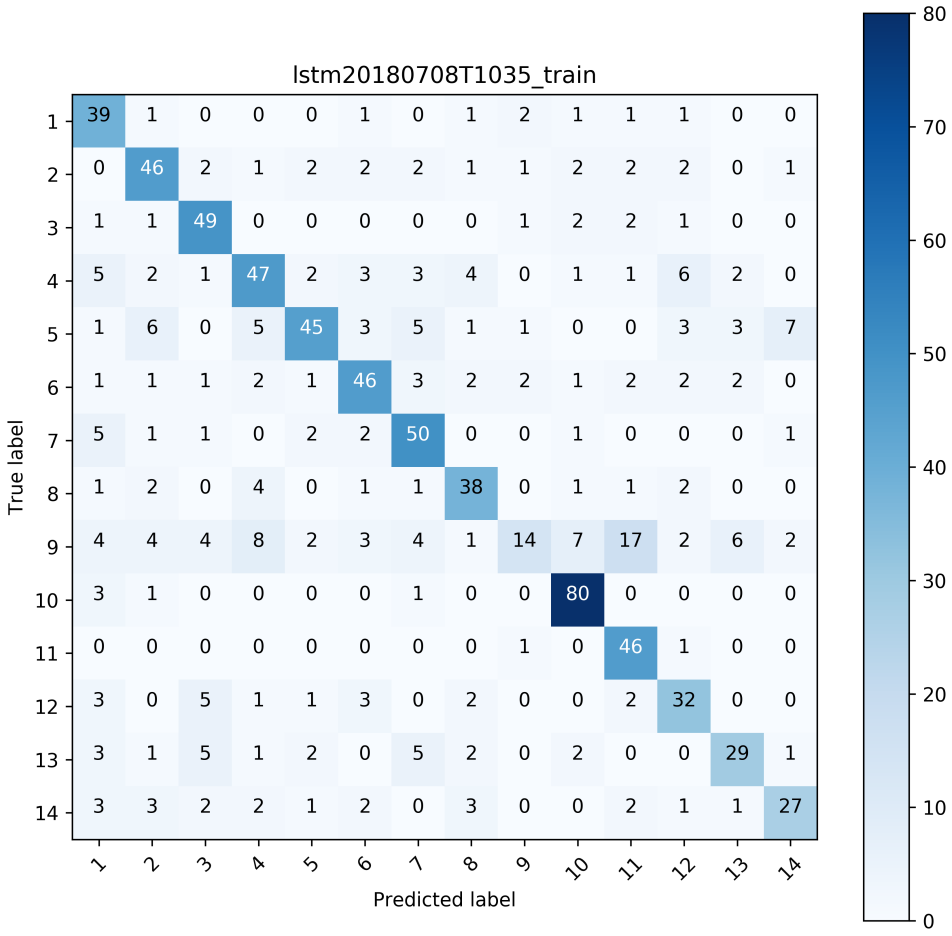

Figure 3.7: Confusion matrix on the training set Masks obtained using an LSTM of 512 hidden nodes. See table 3.1 for the association between the number and the activity.

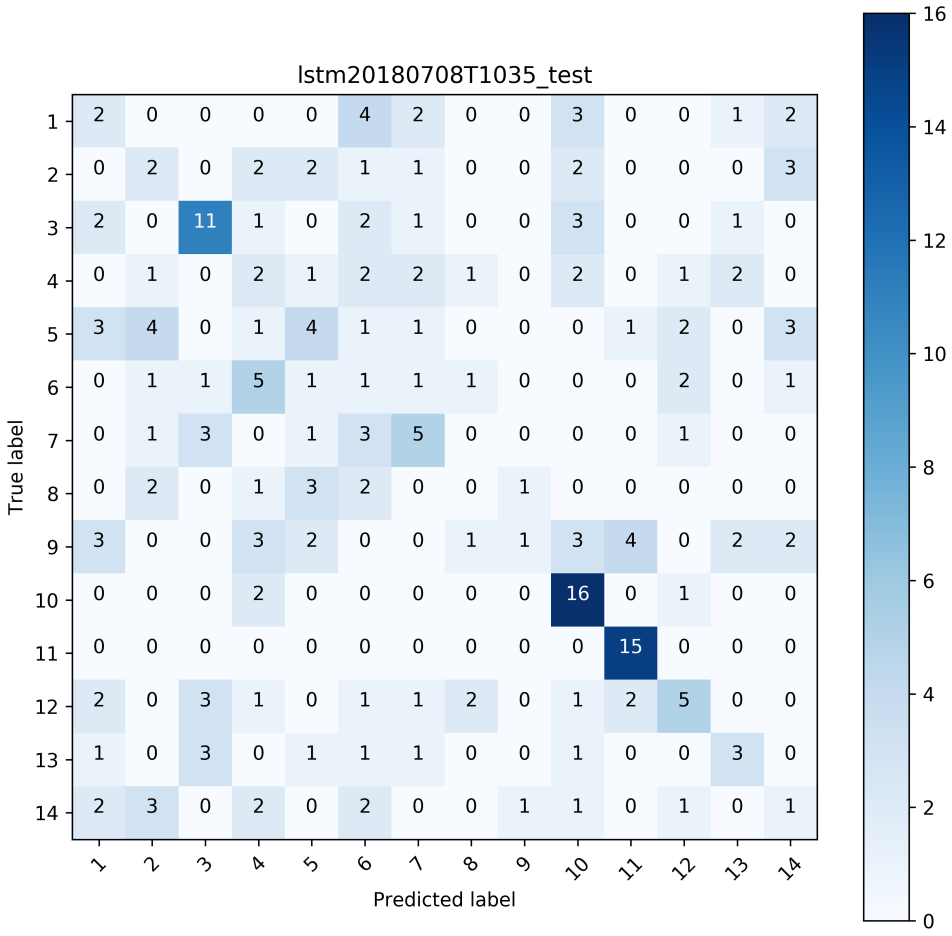

Figure 3.8: Confusion matrix on the test set Masks obtained using an LSTM of 512 hidden nodes. See table 3.1 for the association between the number and the activity.

- **generate_dataset.py:** a tool to generate a unique `.json` file that is later used when training Mask R-CNN.
- **generate_npz.py:** generates the dataset to be fed to the LSTM for the activity recognition on the videos.
- **LSTM/extractFrames.py:** a tool to extract a specified number of frames from a folder containing videos organized in categories.
- **LSTM/splitDataset.py:** groups the videos extracted with **LSTM/extractFrames.py** in a training and a test set.
- **mask_video.py:** generates a video with the masks of the objects recognized by Mask R-CNN.
- **MaskExam.py:** a collection of functions used by **mask_video.py**.
- **merge_npz.py:** a tool to merge two consistent `.npz` files, this has been used to create the dataset Masks+Count.
- **resize_npz.py:** a tool to resize the size of the frames contained in a `.npz` file.
- **split_data.py:** generates a training and a test set from the `.json` file created by **generate_dataset.py**.
- **tbevent2list.py:** a tool to get the data from Tensorboard in LaTeX format to be used in a `tikzpicture`.
- **train_videos.py:** train the LSTM specifying the hyperparameters and the datasets to be used.
- **validation.py:** validate the behaviour of a Mask R-CNN model obtained from **activity.py**.
- **VideoClassifier.py:** implementation of the network containing the LSTM that it is used to classify videos.
- **videos_to_npz.py:** a tool to create `.npz` files containing the videos stored as a numpy array.

5 Conclusion

The final implementation of the LSTM managed to correctly classify the videos by looking at their frames, achieving a maximum accuracy on the training set of **0.928** on Masks when using LSTM-512 and on Masks+Count when using LSTM-1024. The highest accuracy achieved on the test set is **0.532** on Count when using LSTM-256. Studying the plots of the accuracy in all kind of data used, it is possible to notice that using more than 256 hidden nodes for the LSTM does not improve much the performances of the network. The reason is due to the size of the dataset and the amount of masks contained in it.

As previously said, the activity recognition can be improved in multiple ways, working on both Mask R-CNN and the extraction of the features to be used in the LSTM. In particular, Mask R-CNN could be improved by improving the dataset and retraining the network, while the LSTM could be improved by feeding the data in a more accurate way. The data could be extracted also considering the length of the videos, that influence the temporal relation between the frames, and the amount of pixels that compose the masks in each frame. Finally, since the LSTM do not consider the spatiality of the frames, it could be replaced by a Grid LSTM, that should be better at exploiting this kind of information.

References

[1] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[2] N. Kalchbrenner, I. Danihelka, and A. Graves, "Grid long short-term memory," *CoRR*, vol. abs/1507.01526, 2015.