Improving Action Recognition using Temporal Regions

Roger Granada[1] and João Paulo Aires[1] and Juarez Monteiro[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: {roger.granada, joao.aires.001, juarez.santos}@acad.pucrs.br
[2] Email: {felipe.meneguzzi,rodrigo.barros}@pucrs.br

Abstract. Recognizing actions in videos is an important task in computer vision area, having important applications such as the surveillance and assistance of the sick and disabled. Automatizing this task can improve the way we monitor actions since it is not necessary to have a human watching a video all the time. However, the classification of actions in a video is challenging since we have to identify temporal features that best represent each action. 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 classifies actions 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: Action Recognition, Convolutional Neural Networks, Neural Networks

1. INTRODUCTION

Recognizing actions 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 action. In this work, action 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 what action is occurring.

Due to the evolution of graphics processing units (GPU) and an increase in computability power, deep learning methods become the state of the art of different computer vision tasks [Karpathy et al. 2014; Redmon et al. 2016; Chen et al. 2017]. Although previous work has considered single frames to classify an action [Monteiro et al. 2017; Monteiro et al. 2017], deep learning algorithms need to consider the temporal aspect of videos since actions tend to occur through the frames. As described by Carreira and Zisserman [2017], actions can be ambiguous in individual frames. For example, using a single frame it is impossible to distinguish if the action is about to happen or if it has just been done. Thus, using the temporal component of videos provides an additional clue for action recognition, since many actions can be recognized by their motion information [Simonyan and Zisserman 2014].

The authors would like to thank CAPES and Motorola Mobility for funding this work. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

In this work, we propose an approach to compute the temporal aspect of a video by dividing it into regions and merging features from different regions in order to recognize an action. Using Convolutional Neural Networks (CNNs) to extract features from each frame, we train a classifier that receives as input either the concatenation or the mean of features from different regions and outputs a prediction to the corresponding action. 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 using regions may increase the accuracy by about 10% when compared with networks that classify actions frame-by-frame.

This paper is organized as follows. Section 2 presents the state of the art approaches to performing action recognition. Section 3 details each step of our approach in order to create the temporal regions using Convolutional Neural Networks. In Section 4, we describe the datasets the baselines and how our approach is trained using deep neural networks, whereas Section 5 presents a thorough experimental analysis for assessing the performance of our proposed approach, as well as a comparison with the baselines and with the state of the art results for each dataset. Finally, we present our conclusions and future work directions in Section 6.

2. RELATED WORK

Advances in hardware and greater availability of data have allowed deep learning algorithms such as Convolutional Neural Networks (CNNs) [LeCun et al. 1998] to consistently improve on the state of the art results when dealing with image-based tasks such as object detection and recognition [Lu et al. 2018], and semantic segmentation [Ge et al. 2018]. Extensions of CNN representation to the action recognition task in videos have been proposed in several recent works [Monteiro et al. 2017; Simonyan and Zisserman 2014; Wang et al. 2017]. For example, Wang et al. [2017] apply a dynamic tracking attention model (DTAM), which is composed of a CNN and a Long-Short Term Memory (LSTM) to perform human action recognition in videos. Their architecture uses the CNN to extract features from images and the LSTM to deal with the sequential information of the actions. DTAM uses local dynamic tracking to identify moving objects, and global dynamic tracking to estimate the motion of the camera and correct the weights of the motion attention model.

Simonyan and Zisserman [2014] propose the two-stream convolutional network architecture, which is composed of two streams running in parallel with a late fusion to merge both streams. The idea behind the two-streams is to mimic the visual cortex, which contains the ventral stream (responsible for object recognition) and the dorsal stream (responsible for recognizing motion) as two separate pathways [Goodale and Milner 1992]. Thus, videos can be decomposed into spatial and temporal components: the spatial one that carries information about scene context, and the temporal one that conveys the motion across frames, indicating the movement of the observer and objects. Simonyan and Zisserman use the raw images in the spatial stream and pre-computed optical flow features in the temporal stream. Using a two-stream architecture with two different CNNs, Monteiro et al. [2017] perform action recognition in a small egocentric dataset. They affirm that a two-stream architecture achieves better results than a single stream because each stream extracts different features from the same image. The extracted features are then merged by a late score fusion using a Support Vector Machine (SVM). The drawback of such approaches is that hand-crafted features such as the optical flow have to be pre-computed before using the network.

Tran et al. [2015] propose a convolutional neural network, called 3D Convolutional Networks (C3D), that is able to extract spatiotemporal features from videos. From a sequence of frames in the input, this architecture uses 3D convolutional and pooling layers to obtain the sense of movement from the sequence. Thus, instead of using 2D convolutions to extract features from single frames, 3D convolutions use a set of sequential frames. They compare the proposed network with the state of the art networks, obtaining the best results in 5 out of 6 different datasets.

Carreira and Zisserman [2017] propose an architecture that achieves the state of the art results in a large set of datasets for action recognition. The architecture is composed of a two-stream Inflated 3D ConvNet (I3D), which is a 2D Convolutional Neural Network containing an inflation technique. This technique converts 2D CNN's weights into 3D CNN's weights, allowing to load pre-trained weights such as from the ImageNet dataset [Deng et al. 2009], turning the spatial model into a temporal model. The conversion is performed by inflating squared filters ($N \times N$) into cubic filters ($N \times N \times N$). This model is able to capture both spatial and temporal aspects, achieving the state of the art results in UCF-101 [Soomro et al. 2012], HMDB [Kuehne et al. 2011], and Kinetics Human Action Video Dataset [Kay et al. 2017] datasets. The drawback of the architectures that use cubic parameters, *i.e.*, 3D convolutions, relies on the number of parameters they use. For example, while the I3D architecture has about 25M parameters, the combination of a single ConvNet+LSTM has just 9M. Such approaches also demand huge processing power, being impractical if one does not have access to GPUs.

The main difference between our approach and the described approaches is the way we extract the spatiotemporal features. While the current approaches use 3D convolutions with a large number of parameters, we divide a video into a set of regions to obtain the movement (spatiotemporal features) through different regions.

3. METHOD

Recognizing actions from videos often involves an analysis of how objects in frames modify along the time. An approach for action recognition in videos involves obtaining the temporal information, which may contain descriptions about how objects are moving and the main modifications between frames. Since an action consists of a sequence of movements, obtaining the temporal information may help systems to better understand which action 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 action using only the information of each 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, region division, and classification.

1) Pre-processing: The pre-processing step intends to adapt raw images from input to fit into our

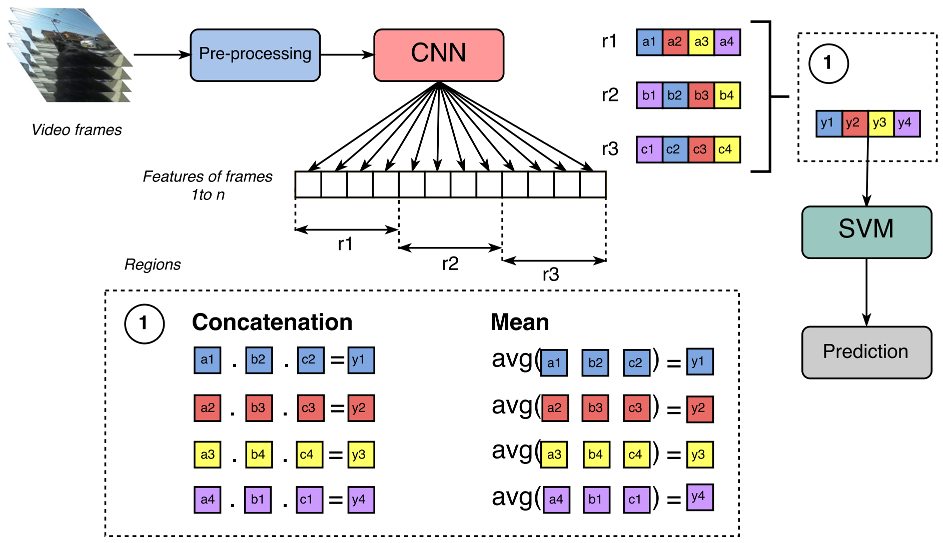

Fig. 1. Pipeline of our architecture for action recognition using multiple regions.

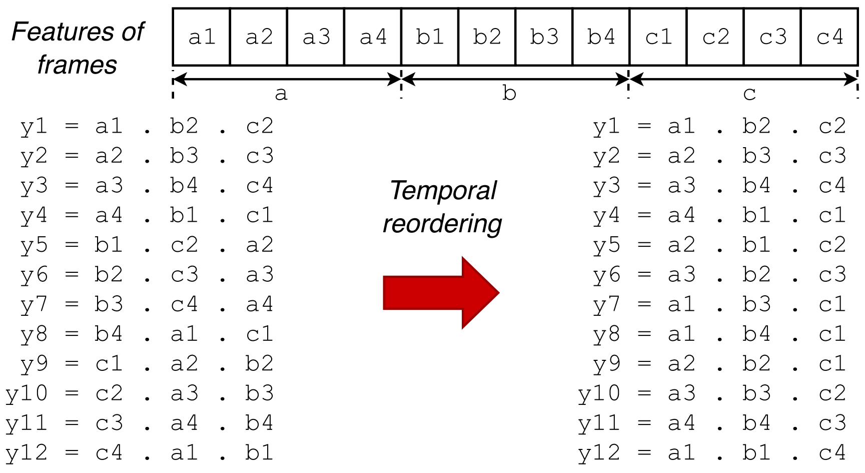

Fig. 2. Extraction temporal regions from a sequence of frames.

deep learning architectures (AlexNet [Krizhevsky et al. 2012] and GoogLeNet [Szegedy et al. 2015]). In order to meet the network requirements, we first transform the input image into a square by cropping the largest dimension and maintaining the lowest dimension, *e.g.*, an input image of 460×640 is cropped to 460×460. The cropped image is then resized to 256×256, which corresponds to the input image of both CNNs. 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 in order to extract features of the action in each frame. We 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 based on the *inception* module that contains convolutional filters in different sizes, covering different clusters of information. The second CNN is a slightly modified version of AlexNet [Krizhevsky et al. 2012] as proposed by Jia et al. [2014]. This network contains a 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. Thus, we use CNNs as feature extractors for images of the dataset in order to generate a feature representation of each frame. Such features are the *softmax* result of a forward pass in the trained network, resulting in a vector with the size equals to the number of classes of the dataset. This vector contains the probability of the input image belonging to each class in classification.

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. In order to avoid repetitions between the region's 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. 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.

Figure 2 illustrates the extraction of temporal regions from a sequence of 12 frames divided into 3 regions, where we slide a target frame (highlighted in bold) from a_1 to c_4, extracting the temporal regions based on the target frame. As our target frame slides through regions, we have to apply a temporal reordering in the extracted frames to keep the original temporal sequence. For example, when our target frame is b_3, we extract from other regions the $i^{th}+1$ position a_4 and c_4, resulting

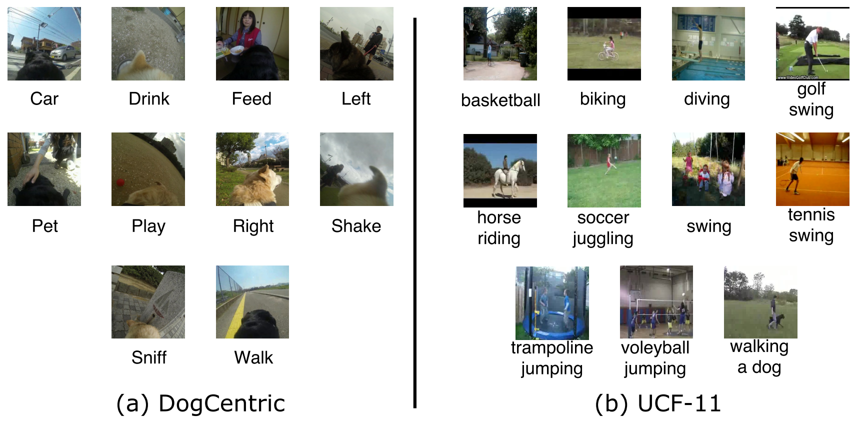

Fig. 3. Examples of frames from each class of DogCentric (a) and UCF-11 (b) datasets.

$y_7 = b_3.c_4.a_4$ in Figure 2, which does not correspond to the correct sequence of frames. Thus, we sort this sequence in the correct sequence, resulting in $y_7 = a_4.b_3.c_4$. Finally, 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, as illustrated in Figure 1.

4) Classification: In this phase, we apply a Support Vector Machine (SVM) [Crammer and Singer 2001] to the features from the concatenation or mean phase in order to predict the action. As we use the *softmax* result as features to the SVM, the maximum size of our vector of features using temporal regions contains $n \times \#classes$ features for Concatenation method and $\#classes$ for Mean method, where n is the number of regions and $\#classes$ is the number of classes of the dataset.

4. EXPERIMENTS

In this section, we describe the datasets used in our experiments for action recognition and the implementation details used in the baseline, CNN models and the SVM algorithm. All experiments were performed using an Intel(R) Xeon(R) CPU E5-2620@2.00GHz, 24 Cores and NVIDIA Titan Xp GPU.

4.1 Dataset

In this work, we perform experiments using two datasets containing different characteristics: a dataset containing an egocentric viewpoint of actions performed by dogs and a dataset containing third-person viewpoint of actions performed by humans. A common point of both datasets is the fact that the two datasets contain a single action for each video. We detail each dataset as follows.

DogCentric Activity dataset[1] [Iwashita et al. 2014] consists of 209 videos containing 10 different actions 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 the dogs' backs records outdoor and indoor scenes. Not all dogs perform all actions and an action can be performed more than once by the same dog. The 10 target actions 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"

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

(*Walk*). It is important to mention that these egocentric (first-person) videos are very challenging due to their strong camera motion. Figure 3 (a) contains frames from each action performed by a dog.

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 action as a test set. In the 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. The validation set contains 20% of the videos and the rest is separated to the training set. Thus, our division contains 105 videos (17,400 frames) in the testing set, 20 videos (3,205 frames) in the validation set and 84 videos (13,040 frames) in the training set.

UCF YouTube Action dataset[2] (hereafter called UCF-11) [Liu et al. 2009] consists of 1,600 videos extracted from YouTube containing 11 actions: *basketball shooting*, *biking/cycling*, *diving*, *golf swinging*, *horse back riding*, *soccer juggling*, *swinging*, *tennis swinging*, *trampoline jumping*, *volleyball spiking*, and *walking with a dog.* Each category is subdivided into 25 groups, where each group contains at least 4 videos that share some common features, such as the same subject performing the action, similar background, similar viewpoint *etc.*. For example, the first group of the *basketball shooting* category contains seven videos with the same people playing basketball. Each video has 320 × 240 resolution and was converted to a frame rate of 29.97 fps and annotations were done accordingly, containing a single action associated with the entire video. As performed to DogCentric dataset, the UCF-11 dataset is divided into train, validation and test sets. Figure 3 (b) contains a frame for each action performed in the dataset.

4.2 Handcrafted Baselines

As deep learning approaches have become the state of the art of different computer vision tasks [Karpathy et al. 2014; Redmon et al. 2016; Chen et al. 2017], we developed as a baseline a pre-deep learning approach that uses hand-crafted features. Thus, we can demonstrate that our approach surpasses methods based on hand-crafted features as well as the frame-by-frame classification using deep learning. Proposed by Wang et al. [2011], Dense Trajectories approach groups different handcrafted features (descriptors) to classify an action. It is a highly relevant approach in the literature, being one of the most cited works that use handcrafted features for action recognition. The approach is an inspiration for new models that intend to capture the temporal aspect, such as deep two-stream networks [Simonyan and Zisserman 2014; Feichtenhofer et al. 2016].

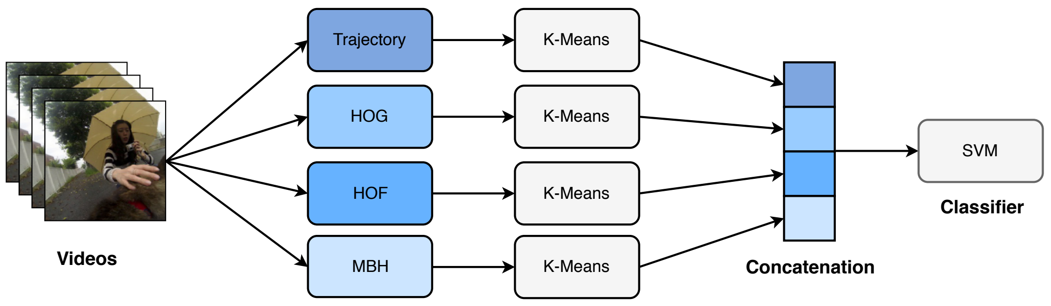

Fig. 4. Pipeline of the Dense Trajectories approach

[2]http://crcv.ucf.edu/data/UCF_YouTube_Action.php

In our baseline, we follow the configuration proposed by Wang et al. [2011] and extract the following descriptors: Trajectory, Histogram of Gradients (HOG), Histogram of Optical Flow (HOF), and Motion Boundary History (MBH), where MBH can be divided into MBHx and MBHy containing the motion boundary in the x and y axes respectively, as illustrated in Figure 4. For each descriptor we create a codebook separately, fixing the number of visual words per descriptor to 4000. We cluster a subset of 100,000 random features in training using k-means algorithm. Algorithm 1 describes the process of generating clusters for descriptors of each video.

Algorithm 1: Generating cluster for each descriptor

```
   Input: Videos ($V$)
   Output: Clustering model for each descriptor
 1 begin
 2     $D \leftarrow$ [Trajectory, HOG, HOF, MBHx, MBHy]
 3     for $v$ in $V$ do
 4         for $d$ in $D$ do
 5             $\text{feature}_d \leftarrow \text{extract\_features}(d, v)$
 6             $\text{normalized\_feature}_d \leftarrow \text{normalize\_features}(\text{feature}_d)$
 7             $\text{cluster}_d \leftarrow \text{generate\_cluster}(\text{normalized\_feature}_d)$
 8         end
 9     end
10     return cluster
11 end
```

Thus, for each video, the descriptors are assigned to their closest vocabulary word using Euclidean distance, being the resulting histograms of visual words used as descriptor of the video. For classification, a Support Vector Machine (SVM) is trained using the concatenation of the extracted features.

Algorithm 2 describes the process of training the SVM with the extracted descriptors of a video having the clusters, and Algorithm 3 describes the classification of a new video using the generated clusters. In order to optimize parameters in the SVM classifier, we follow Hsu et al. [2003] and perform a grid search for SVM using RBF kernel and another grid search for SVM with a linear kernel. For both models we range the values of C from 2^{-5} to 2^{15}, increasing by 2^2 each step, and for the model with the RBF kernel, it is also advisable to vary the hyperparameter γ: 2^{-15}, 2^{-13}, ..., 2^3. Given this combination, for each descriptor, we generate 11 models for linear SVM and 110 models for the RBF kernel.

An important limitation of these approaches is that they only work for videos that have a single action in the entire sequence of frames. Thus, the approach cannot generate clusters when the dataset has more than one action in a clip. In order to obtain a larger range of baseline results, we train SVMs with all features separately. Therefore, our baselines contain a total of 6 approaches for the dataset, being an SVM trained with trajectory (*Trajectory*), an SVM trained with Histogram of Gradients (*HOG*), an SVM trained with Histogram of Optical Flow (*HOF*), an SVM trained with the Motion Boundary History of the x axis (*MBHx*), an SVM trained with the Motion Boundary History of the y axis (*MBHy*), and an SVM trained with dense trajectories (*Dense*).

4.3 Model Training

We implemented each part of our architecture using different toolkits: for CNNs, we use the Caffe[3] framework [Jia et al. 2014], and for the SVM, we use the Crammer and Singer [2001] implementation

[3]`http://caffe.berkeleyvision.org/`

Algorithm 2: Train the SVM classifier using features extracted from videos

```
Input: Clusters ($C$), Videos ($V$)
Output: SVM Classifier
1  begin
2      $D \leftarrow$ [Trajectory, HOG, HOF, MBHx, MBHy]
3      for $v$ in $V$ do
4          for $d$ in $D$ do
5              feature$_d \leftarrow$ extract_features($d$, $v$)
6              normalized_feature$_d \leftarrow$ normalize_features(feature$_d$)
7              for $c_d$ in $C$ do
8                  histogram$_d \leftarrow$ generate_hist($c_d$, normalized_feature$_d$)
9                  hist_norm$_d \leftarrow$ normalize_histogram(histogram$_d$)
10                 hist_concat $\leftarrow$ concatenate_histogram(hist_norm$_d$)
11             end
12         end
13     end
14     svm_model $\leftarrow$ train_svm(hist_concat)
15     return svm_model
16 end
```

Algorithm 3: Classifying a video using the trained SVM

```
Input: Clusters ($C$), SVM model (svm_model), video ($v$)
Output: Video classification
1  begin
2      $D \leftarrow [Trajectory, HOG, HOF, MBHx, MBHy]$
3      for $d$ in $D$ do
4          feature$_d \leftarrow$ extract_features($d$, $v$)
5          normalized_feature$_d \leftarrow$ normalize_features(feature$_d$)
6          for $c_d$ in $C$ do
7              histogram$_d \leftarrow$ generate_hist($c_d$, normalized_feature$_d$)
8              hist_norm$_d \leftarrow$ normalize_histogram(histogram$_d$)
9              hist_concat $\leftarrow$ concatenate_histogram(hist_norm$_d$)
10         end
11     end
12     classification $\leftarrow$ classify_video(hist_concat, svm_model)
13     return classification
14 end
```

from *scikit-learn*[4] [Pedregosa et al. 2011]. Each CNN is trained by fine-tuning a network pre-trained in ImageNet [Russakovsky et al. 2015] dataset. In the 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) uses 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

[4]`http://scikit-learn.org/`

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 20 epochs.

5. RESULTS

In order to evaluate our approach, we calculate the accuracy ($\mathcal{A}$), precision ($\mathcal{P}$), recall ($\mathcal{R}$) and F-measure ($\mathcal{F}$) that each model achieves. Since classification accuracy takes into account only the proportion of correct results that a classifier achieves, it is not suitable for unbalanced datasets since it may be biased towards classes with a 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 both datasets, we notice that they are indeed unbalanced, *i.e.*, classes are not equally distributed over frames. Hence, we decided to calculate precision, recall, and F-measure to make a better evaluation of the unbalanced dataset. We calculate scores considering all classes presented in the test set as explained in Section 4.1.

Table I shows the results achieved by AlexNet and GoogLeNet in our experiments when splitting both datasets into regions. The best results for concatenation and mean in each dataset are highlighted in bold. It is important to note that Regions=1 is equivalent to computing single frames individually without splitting the video, *i.e.*, the traditional approach that does not encode the temporal aspect of an action.

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 and UCF-11 datasets.

	Regions	**Fusion**	**AlexNet**				**GoogLeNet**			
			$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$
DogCentric	1	–	0.51	0.57	0.52	0.52	0.56	0.62	0.56	0.56
	2	Concatenation	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	Concatenation	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	Concatenation	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	Concatenation	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	Concatenation	**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
UCF-11	1	–	0.73	0.75	0.74	0.74	0.73	0.75	0.73	0.73
	2	Concatenation	0.76	0.78	0.77	0.76	0.76	0.78	0.76	0.76
		Mean	0.76	0.78	0.77	0.77	0.76	0.78	0.76	0.76
	4	Concatenation	**0.77**	**0.79**	**0.78**	**0.78**	0.74	0.76	0.75	0.74
		Mean	**0.77**	**0.79**	**0.78**	**0.78**	0.75	0.77	0.75	0.74
	8	Concatenation	**0.77**	**0.79**	0.77	0.77	0.74	0.77	0.75	0.74
		Mean	**0.77**	**0.79**	**0.78**	**0.78**	0.74	0.75	0.74	0.73
	16	Concatenation	**0.77**	**0.79**	0.77	0.77	0.76	0.79	**0.77**	0.76
		Mean	**0.77**	0.78	0.77	0.77	0.76	0.78	0.76	0.76
	32	Concatenation	**0.77**	0.78	0.77	0.77	**0.77**	**0.80**	**0.77**	**0.77**
		Mean	0.76	0.78	0.77	0.76	0.75	0.78	0.76	0.75

1) Overall Performance: We separated the analysis of the overall performance by dataset.

DogCentric: 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 seems to classify actions better than AlexNet, surpassing the latter in all metrics. This may occur due to the set of layers in GoogLeNet, such as *inception* that allows the extraction of more complex features from data.

UCF-11: As shown in Table I, the best results occur in different configurations for each network. Using AlexNet, we can see that the results are very close to each other, concentrating in the middle of the number of regions (4 and 8 regions), having higher and lower achieved lower results. The accuracy for each region and feature division has no much difference being around 76% and 77% for each case, except when we apply 1 region only, which obtained 73% of accuracy. On the other hand, precision, recall, and F-measure have their best results centralized on 4 and 8 regions for both concatenation and mean. We believe that the results are very close to each other because UCF-11 contains short videos with about 100 frames each video. Thus, increasing the number of regions does not change the main features to identify an action, *i.e.*, the network identifies the same features in different regions. Using GoogLeNet, the best results are predominant when using 16 and 32 regions. More specifically, the combination of 32 regions and concatenation obtains the best results for all metrics.

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, since it generates a vector of size $n \times \#classes$ for encoding the temporal aspect, where n is the number of regions and $\#classes$ is the number of classes of the dataset, 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 increases in both AlexNet and GoogLeNet. Although some actions can be identified in a single frame, it usually takes several frames to identify an action. 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 5 the confusion matrices generated using the DogCentric dataset with 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.

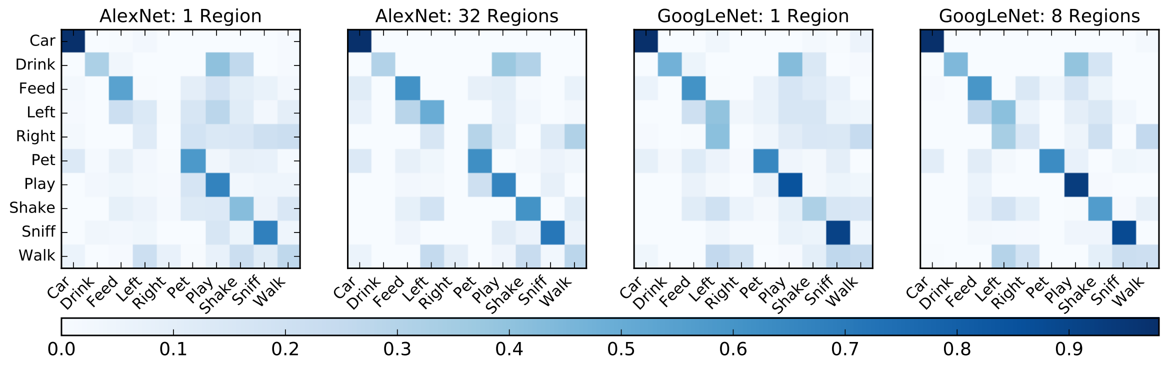

Fig. 5. Confusion matrix for AlexNet using 1 and 32 regions and GoogLeNet using 1 and 8 regions for the DogCentric dataset.

As we can see in Figure 5, there is an increase in precision for both *Left* and *Shake* classes when splitting the dataset into regions 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 a significant increase with 8 regions or more. There is a difference between GoogLeNet and AlexNet number of regions that yields their best results, such difference occurs for both datasets. While GoogLeNet applied to the DogCentric dataset obtains its best results with 8 and 16 regions, AlexNet obtains with 32 regions. The opposite occurs when we apply such networks in the UCF-11 dataset, GoogLeNet obtains its best results with 16 and 32 regions, whereas AlexNet obtains its best results using 4 and 8 regions. We can conclude that using a number of regions higher than 1 obtains better results for most cases and each network achieves the best results using different combinations of the number of regions and fusion methods depending on the dataset characteristics.

From all the confusion matrices, we can observe that the classes *Car* and *Sniff* have the highest scores in classification, indicating that these classes are very different from 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 the largest number of samples in the dataset, which facilitates the network to learn their features.

4) Unbalanced classes: Analyzing both datasets we observe that they are unbalanced, *i.e.*, the number of frames across classes varies. Figure 6 illustrates the distribution of frames in both DogCentric and UCF-11 datasets.

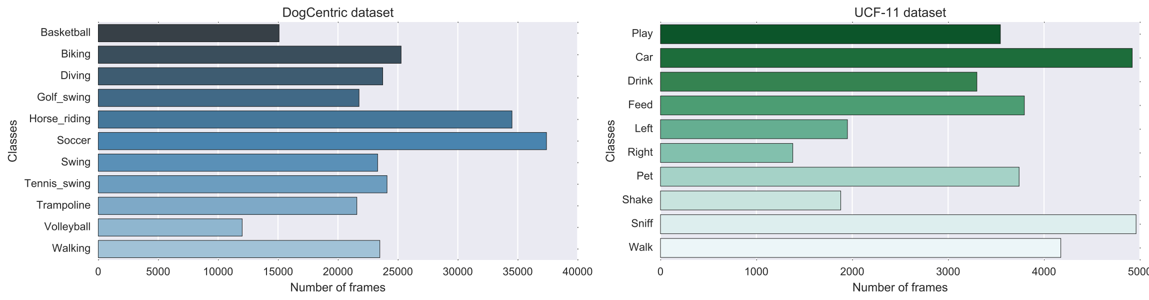

Fig. 6. Number of frames for each class in DogCentric and UCF-11 datasets.

Analyzing the DogCentric 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, the *Left* class is only $\approx 4\%$ of the total frames in the dataset and the *Right* class is $\approx 6\%$ of the total frames. The unbalanced nature of the dataset seems to impact on the results since the lower the number of frames the class has, the lower the accuracy score it achieves. In UCF-11 dataset most classes have about 9% of the total number of frames. Exceptions are *Basketball* with 5%, *Volleyball_spiking*, *Horse_riding* with 12%, and *Soccer_juggling* with 13%. Since we evaluate using precision, recall, and F-measure, we are able to deal with such differences between class samples.

5) Comparing with baselines: In order to compare our work with the existing baselines, we perform a series of experiments using handcrafted feature models as proposed by Wang et al. [2011]. Table II shows the results from each baseline model compared to the deep approaches (AlexNet and GoogLeNet) using a single region (which means the off-the-shelf way) and the best results obtained by both deep approaches using temporal regions. As detailed in Section 4.2, we perform a grid search in the training set changing the hyperparameters C and γ to the baselines: Trajectory, HOG, HOF, MBHx, MBHy, and Dense trajectories. We check the combination of hyperparameters using the validation set and the best combination of C and γ are used in the test set. Due to space constraints, we do not add to the paper the results achieved by the combination of hyperparameters in the validation

Table II. Results achieved by our approach and existing baselines in DogCentric and UCF-11 datasets.

		Approach	**Kernel**	C	γ	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$	$p-\mathcal{W}$	$p-\mathcal{N}$
DogCentric	CNN Single Region	AlexNet	Linear	2^0	–	0.51	0.57	0.52	0.52	–	–
		GoogLeNet	Linear	2^0	–	0.56	0.62	0.56	0.56	–	–
	CNN Multiple Regions	AlexNet	Linear	2^0	–	0.59	0.66	0.59	0.59	–	–
		GoogLeNet	Linear	2^0	–	**0.62**	**0.71**	**0.63**	**0.63**	–	–
	Baselines	Trajectory	RBF	2^7	2^1	0.28	0.33	0.29	0.27	0.009	0.000
		HOG	RBF	2^7	2^3	0.51	0.54	0.51	0.50	0.332	0.700
		HOF	RBF	2^5	2^3	0.48	0.51	0.49	0.46	0.092	0.150
		MBHx	RBF	2^7	2^3	0.51	0.59	0.51	0.52	0.241	0.210
		MBHy	RBF	2^7	2^3	0.54	0.56	0.54	0.52	0.386	0.690
		Dense	RBF	2^5	2^1	0.49	0.51	0.50	0.46	0.168	0.250
UCF-11	CNN Single Region	AlexNet	Linear	2^0	–	0.73	0.75	0.74	0.74	–	–
		GoogLeNet	Linear	2^0	–	0.73	0.75	0.73	0.73	–	–
	CNN Multiple Regions	AlexNet	Linear	2^0	–	**0.77**	0.79	**0.78**	**0.78**	–	–
		GoogLeNet	Linear	2^0	–	**0.77**	0.80	**0.77**	**0.77**	–	–
	Baselines	Trajectory	Linear	2^{11}	–	0.39	0.44	0.39	0.40	0.003	0.000
		HOG	RBF	2^7	2^3	0.74	0.79	0.74	0.76	0.020	0.013
		HOF	RBF	2^7	2^3	0.68	0.72	0.68	0.68	0.016	0.168
		MBHx	RBF	2^{13}	2^{-1}	0.70	0.78	0.70	0.70	0.075	0.087
		MBHy	RBF	2^9	2^1	0.76	**0.83**	0.76	**0.78**	0.050	0.026
		Dense	RBF	2^{13}	2^{-5}	0.76	0.78	0.76	0.75	0.154	0.000

set, but the complete table containing all results of the grid search can be found on the project's website[5]. Thus, Table II shows the results achieved by our approaches using a single and multiple regions, and the baselines using the configuration of hyperparameters (*Kernel*, C and γ) that achieve the highest results in validation set. As the deep learning methods use the default *scikit-learn* parameter for SVM, we add its default value in the column C.

As we can see, GoogLeNet with temporal regions obtains the best result in all cases in the DogCentric dataset and almost all cases in the UCF-11 dataset, when compared with all other approaches. In the DogCentric dataset, GoogLeNet network obtains 62% of accuracy, 72% of precision, 63% of recall, and 63% of F-measure. The results compared here consider the number of regions that achieved the best results in both GoogLeNet and AlexNet when using temporal regions (see Table I). Among the baselines, we can see that both HOG and MBHy obtain similar results in all metrics when compared to the other baseline approaches. However, MBHy surpasses HOG obtaining 54% of accuracy, 56% of precision, 54% of recall, and 52% of F-measure. Directly comparing the best approaches of both cases, we can see that using deep learning with temporal regions has on average an improvement of 11% in the DogCentric dataset. It is interesting to note that both deep learning approaches without temporal regions obtain close results to the ones achieved by the baseline approaches.

5) Statistical Test: We performed two non-parametric significance tests: (a) Wilcoxon Signed-Ranks [Wilcoxon 1945] and (b) McNemar's test [McNemar 1947] to answer the question whether the performances of two methods are equivalent. We compute p-values using the Wilcoxon signed rank and McNemar's test since they are non-parametric tests, *i.e.*, they do not rely on assumptions that data belong to any particular distribution. We perform the Wilcoxon test because it is based on individual comparison by ranking methods, while McNemar's test because it uses contingency tables. In order to perform the significance tests, we create two hypothesis, namely:

—H_0: The performance of our approach in the action recognition task is equal to the performance of the baselines.

—H_1: The performance of our approach in the action recognition task is different from the performance of the baselines.

[5]`https://github.com/jrzmnt/TemporalRegions`

In particular, we apply the significance tests to check the null hypothesis (H_0) against the alternative one (H_1). We add a p-value column for the Wilcoxon test ($p-\mathcal{W}$) and a p-value column for McNemar's test ($p-\mathcal{N}$) in Table II, that compare the accuracy per class obtained by our best model to the accuracy per class obtained by the baselines. For the DogCentric dataset, we perform the significance test using GoogLeNet with multiple regions. Using a significance level of $<$0.05, we can see that all baselines confirm the null hypothesis (H_0) but the Trajectory approach (with a p-value of 0.009 for the Wilcoxon test and near zero for McNemar's test).

For the UCF-11 dataset, we compare the results using AlexNet with multiple regions. For the Wilcoxon test, all baselines but MBHx and Dense reject the null hypothesis (confirm H_1), while for McNemar's test, all baselines but HOF (p-value=0.168) and MBHx (p-value=0.087) reject the null hypothesis. When comparing both significance tests, we see that they differ for HOF and Dense. In general, most of the p-values in both tests show a similar trend.

6) Comparing with related work: Many work have been proposed for action recognition in recent years. Here, we describe some of these work that perform experiments using DogCentric and UCF-11 datasets. We explain their main approaches and Table III summarizes their results of Accuracy ($\mathcal{A}$), Precision ($\mathcal{P}$), Recall ($\mathcal{R}$) and F-measure ($\mathcal{F}$) when they exist. At the bottom of each dataset we add the results of our approaches using the number of regions that achieves the best results.

Iwashita et al. [2014] perform the activity recognition extracting hand-crafted global features from dense optical flow and local features from a cuboid feature detector [Dollár et al. 2005] and STIP detector [Laptev 2005]. All features are clustered using the concept of visual words and integrated using a linear kernel and three non-linear kernels (RBF kernel, multi-channel χ^2 kernel, and multi-channel histogram intersection kernel) in order to recognize first-person animal activities. Using hand-crafted features they achieve the highest accuracy of 60.5%.

Ryoo et al. [2015] develop a feature representation named pooled time series (PoT) that applies time series pooling of feature descriptors, detecting short-term/long-term changes in each descriptor element. Their approach extracts appearance/motion descriptors from a sequence of frames and represents them as a set of time series. Temporal pooling is applied to each time series in order to summarize the information represented in a sequence of video frames that are classified using a Support Vector Machine (SVM) classifier. This approach achieves 74% accuracy when combined with INRIA's improved trajectory feature (ITF) [Wang and Schmid 2013] in the DogCentric Activity dataset.

Recently, Monteiro et al. [2017] have created an architecture containing two convolutional neural networks running in parallel with a late fusion to merge both networks. They apply post-processing on the predicted labels, which consists of smoothing the output of the fusing method by using a sliding window of fixed size through the predicted classes assigning to the target frame the majority voting of all frames within the window. This smoothing process aims to eliminate few correctly predicted classes when they are in the middle of other classes. In their experiments, this architecture with the post-processing step achieves 75.6% of accuracy, 74% of precision, 76% of recall and 75% of F-measure when using the DogCentric dataset.

Wang et al. [2017] proposes a dynamic tracking attention model (DTAM), which applies a motion attention mechanism composed of a convolutional neural network (CNN) and long short-term memory (LSTM) in order to perform human action recognition in videos. In that work, the CNN is used as a feature extractor, while the LSTM deals with the temporal aspect of the information about actions in the video. DTAM uses local dynamic tracking to identify moving objects, and global dynamic tracking to estimate the motion of the camera and correct the weights of the motion attention model. Using a merge of visual attention [Mnih et al. 2014] with their proposed DTAM, they achieve 91% of accuracy. Values of precision, recall, and F-score are not reported in their work.

Table III. Accuracy ($\mathcal{A}$), Precision ($\mathcal{P}$), Recall ($\mathcal{R}$) and F-measure ($\mathcal{F}$) achieved by our method and by related work using DogCentric and UCF-11 datasets.

Dataset	**Architecture**	$\mathcal{A}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{F}$
DogCentric	Linear kernel [Iwashita et al. 2014]	0.53	-	-	-
	RBF kernel [Iwashita et al. 2014]	0.54	-	-	-
	Histogram intersection [Iwashita et al. 2014]	0.57	-	-	-
	Multi-channel [Iwashita et al. 2014]	0.61	-	-	-
	PoT+STIP+Cuboid [Ryoo et al. 2015]	0.73	-	-	-
	PoT+ITF+STIP+Cuboid [Ryoo et al. 2015]	0.74	-	-	-
	PoT+ITF [Ryoo et al. 2015]	0.75	-	-	-
	2 CNNs-SVM [Monteiro et al. 2017]	0.68	0.69	0.68	0.69
	2 CNNs-SVM-PP [Monteiro et al. 2017]	**0.76**	**0.74**	**0.76**	**0.75**
	AlexNet (32 regions, concatenation)	0.59	0.66	0.59	0.59
	GoogLeNet (8 regions, concatenation)	0.62	0.71	0.63	0.63
UCF-11	DTAM [Wang et al. 2017]	0.90	-	-	-
	Visual-DTAM [Wang et al. 2017]	**0.91**	-	-	-
	AlexNet (4 regions, concatenation)	0.77	0.79	0.78	0.78
	GoogLeNet (32 regions, concatenation)	0.77	0.80	0.77	0.77

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 architecture is composed of a CNN (AlexNet or GoogLeNet), a method to separate and compute regions, and a final classifier (SVM). Using images from the DogCentric and UCF-11 datasets, we perform experiments showing that our approach can improve the action recognition task. We test our approach using two networks AlexNet and GoogLeNet, increasing precision by up to 10% when using regions to classify actions. Our proposed architecture also achieves better results when compared to a baseline composed of handcrafted methods.

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], C3D [Tran et al. 2015] and 3D CNN [Ji et al. 2013].

ACKNOWLEDGMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) and Fundação de Amparo á Pesquisa do Estado do Rio Grande do Sul (FAPERGS) agreement (DOCFIX 04/2018). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

REFERENCES

Carreira, J. and Zisserman, A. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 4724–4733, 2017.

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* 2 (Dec): 265–292, 2001.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. pp. 248–255, 2009.

Dollár, P., Rabaud, V., Cottrell, G., and Belongie, S. Behavior recognition via sparse spatio-temporal features. In *Proceedings of the 2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*. pp. 65–72, 2005.

Feichtenhofer, C., Pinz, A., and Zisserman, A. Convolutional two-stream network fusion for video action recognition. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1933–1941, 2016.

Ge, W., Yang, S., and Yu, Y. Multi-evidence filtering and fusion for multi-label classification, object detection and semantic segmentation based on weakly supervised learning. In *Proceedings of CVPR'18*. pp. 1277–1286, 2018.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. pp. 249–256, 2010.

Goodale, M. A. and Milner, A. D. Separate visual pathways for perception and action. *Trends in Neurosciences* 15 (1): 20–25, 1992.

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

Hsu, C.-W., Chang, C.-C., and Lin, C.-J. A practical guide to support vector classification. Tech. rep., National Taiwan University. July, 2003.

Iwashita, Y., Takamine, A., Kurazume, R., and Ryoo, M. S. First-person animal activity recognition from egocentric videos. In *Proceedings of the 22nd International Conference on Pattern Recognition*. pp. 4310–4315, 2014.

Ji, S., Xu, W., Yang, M., and Yu, K. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 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 *Proceedings of the 22nd ACM International Conference on Multimedia*. 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*. pp. 1725–1732, 2014.

Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., and Zisserman, A. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950, May 2017*, 2017.

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*. pp. 1097–1105, 2012.

Kuehne, H., Jhuang, H., Garrote, E., Poggio, T. A., and Serre, T. Hmdb: A large video database for human motion recognition. In *Proceedings of the 2011 International Conference on Computer Vision*. ICCV'11. IEEE, Washington, DC, USA, pp. 2556–2563, 2011.

Laptev, I. On space-time interest points. *International Journal of Computer Vision* 64 (2-3): 107–123, 2005.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86 (11): 2278–2324, 1998.

Liu, J., Luo, J., and Shah, M. Recognizing realistic actions from videos "in the wild". In *Proceedings of the 2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp. 1996–2003, 2009.

Lu, C., Su, H., Li, Y., Lu, Y., Yi, L., Tang, C.-K., and Guibas, L. J. Beyond holistic object recognition: Enriching image understanding with part states. In *Proceedings of CVPR'18*. pp. 6955–6963, 2018.

McNemar, Q. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 12 (2): 153–157, 1947.

Mnih, V., Heess, N., Graves, A., and Kavukcuoglu, K. Recurrent models of visual attention. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. pp. 2204–2212, 2014.

Monteiro, J., Aires, J. P., Granada, R., Barros, R. C., and Meneguzzi, F. Virtual guide dog: An application to support visually-impaired people through deep convolutional neural networks. In *Proceedings of the 2017 International Joint Conference on Neural Networks*. pp. 2267–2274, 2017.

Monteiro, J., Granada, R., Barros, R. C., and Meneguzzi, F. Deep neural networks for kitchen activity recognition. In *Proceedings of the 2017 International Joint Conference on Neural Networks*. pp. 2048–2055, 2017.

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* 12 (Oct): 2825–2830, 2011.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 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. *International Journal of Computer Vision* 115 (3): 211–252, 2015.

Ryoo, M. S., Rothrock, B., and Matthies, L. Pooled motion features for first-person videos. In *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 896–904, 2015.

Simonyan, K. and Zisserman, A. Two-stream convolutional networks for action recognition in videos. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, 2014.

Soomro, K., Zamir, A. R., and Shah, M. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

Sukthankar, G., Goldman, R. P., Geib, C., Pynadath, D. V., and Bui, H. H. *Plan, Activity, and Intent Recognition: Theory and Practice*. Morgan Kaufmann Publishers Inc., 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 *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*. pp. 1–9, 2015.

Tran, D., Bourdev, L., Fergus, R., Torresani, L., and Paluri, M. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV 2015)*. pp. 4489–4497, 2015.

Wang, C.-Y., Chiang, C.-C., Ding, J.-J., and Wang, J.-C. Dynamic tracking attention model for action recognition. In *Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing*. pp. 1617–1621, 2017.

Wang, H., Kläser, A., Schmid, C., and Liu, C.-L. Action recognition by dense trajectories. In *Proceedings of the 2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. pp. 3169–3176, 2011.

Wang, H. and Schmid, C. Action recognition with improved trajectories. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*. pp. 3551–3558, 2013.

Wilcoxon, F. Individual comparisons by ranking methods. *Biometrics Bulletin* 1 (6): 80–83, 1945.