Evaluating the Feasibility of Deep Learning for Action Recognition in Small Datasets

Juarez Monteiro*, Roger Granada*, João Paulo Aires*, Rodrigo C. Barros†
School of Technology
Pontifícia Universidade Católica do Rio Grande do Sul
Av. Ipiranga, 6681, 90619-900, Porto Alegre, RS, Brazil
* Email: {juarez.santos, joao.aires.001, roger.granada}@acad.pucrs.br
† Email: rodrigo.barros@pucrs.br

***Abstract*—Action recognition is the computer vision task of identifying what action is happening in a given sequence of frames. Traditional approaches rely on handcrafted features and domain-specific image processing algorithms, often resulting in limited accuracy. The substantial advances in deep learning and the availability of larger datasets have allowed techniques that yield much better performance without domain-specific knowledge to recognize actions being performed based on the raw information from video sequences, a strategy called *representation learning*. However, deep neural networks usually require very large labeled datasets for training and properly generalizing, and due to their high capacity, they often overfit small data, hence providing suboptimal results. This work aims to check the real feasibility of employing deep learning in the context of small-sized action recognition datasets. Our goal is to verify whether deep learning approaches can provide improved performance in cases in which labeled data is not abundant. In order to do so, we perform a thorough empirical analysis in which we investigate distinct network architectures with hyperparameter optimization, as well as different data pre-processing techniques and fusion methods.**

I. Introduction

The success of deep learning models has led to their deployment in most (if not all) areas of computer vision, including image classification [1], [2], object detection [3], and video understanding [4], [5]. One of the main tasks of video understanding is human action recognition, a fundamental and well-studied problem in computer vision. Traditional approaches for action recognition are based on handcrafted features, namely dense trajectories [6], [7] and part-based/structured models [8], [9]. More recently, representation learning (*e.g.*, deep neural networks) has achieved considerable success in that task [10], [11], greatly benefiting from the advances in image-based Convolutional Neural Networks (CNNs) [12], [13]. With the rare exception of some 3D convolutional models [11], [14], most state-of-the-art strategies for action recognition [15]–[17] make use of a 2D-CNN [18] over the frames, often in combination with motion information (*e.g.*, dense optical flow) from the input video.

While using standard deep networks over the full image has shown great promise for the task [17], much is argued that frame-based approaches fail to properly capture spatiotemporal data [11], [19], [20]. Notwithstanding, 3D convolutional models are said to only work in very large datasets, considering that their large capacity leads them to overfit smaller datasets, hurting their generalization ability. Indeed, state-of-the-art approaches for smaller datasets are usually still based on handcrafted features with simpler classification approaches such as SVMs [21], [22].

Hence, in this paper we perform a thorough empirical analysis to verify the feasibility of employing deep learning models for action recognition in small datasets. By small, we mean datasets that range from a few thousand frames to hundreds of thousands [21], [23], [24]. In particular, we evaluate three datasets that present different characteristics:

1) KSCGR [24], which contains images recorded with a static camera;
2) DogCentric [21], which contains dynamic scenes with a lot of camera motion and high inter-class similarity;
3) UCF-11 [23], which contains dynamic scenes with low inter-class similarity.

For verifying the suitability of employing deep neural networks on those three datasets, we make use of several different approaches that have been applied for action recognition in the literature, though often for large datasets:

- Direct classification algorithms (also regarded as Static Models) such as Support Vector Machines and Convolutional Neural Networks: approaches that do not take temporal information into account for classifying actions in videos;
- Temporal state-space models (also regarded as Temporal Models) such as two-stream networks, and Long Short-Term Memory networks: approaches that consider temporal information, either through handcrafted motion information like Optical Flow, via fusion methods, or through recurrent networks.

In our thorough empirical analysis, we test a large variety of approaches and also optimize their hyperparameters, so we can properly verify the hypothesis of whether deep learning works for action recognition in small datasets. Our results show that deep learning approaches can achieve good results when compared with the state-of-the-art methods for the datasets tested in this paper.

II. Setting up the Environment

In this section, we introduce three datasets and the architectures we use to perform experiments and analysis of deep learning algorithms for action recognition using small datasets. We selected three datasets containing different characteristics from the camera motion point of view. Our architectures are based on deep learning algorithms, such as Convolutional Neural Networks (CNN) and Long Short-Term Memory Networks (LSTM).

A. Datasets

In our work we intend to analyze the performance of algorithms for action recognition using deep learning in small datasets. By small, we mean datasets that range from a few thousand videos (clips) to hundreds of thousands [21], [23], [24]. Therefore, we select three datasets with different characteristics. For example, KSCGR [24] contains images recorded with a static camera, while DogCentric [21] and UCF-11 [23] contain dynamic scenes with large variations in camera motion. Unlike DogCentric, which contains egocentric images, *i.e.*, images recorded in first-person perspective, KSCGR and UCF-11 contain a third-person view of the camera. Thus, all three selected datasets comprehend different points of view. Each dataset is detailed as follows.

1) KSCGR: The Kitchen Scene Context based Gesture Recognition dataset[1] (KSCGR) [24] is a fine-grained kitchen action dataset released as a challenge in ICPR 2012[2]. The dataset contains scenes captured by a Kinect sensor fixed on the top of the kitchen, where 7 different subjects perform five menus for cooking eggs in Japan: *ham and eggs*, *omelet*, *scrambled egg*, *boiled egg*, and *kinshi-tamago*. Each video is 5 to 10 minutes long, containing 9,000 to 18,000 frames with the subject performing multiple actions in sequence. The entire dataset encompasses 227,874 frames in total, where each frame is annotated with one of 8 actions: *breaking*, *mixing*, *baking*, *turning*, *cutting*, *boiling*, *seasoning*, *peeling*, and *none*, where *none* means that there is no action being performed in the current frame or the current action does not fit in any other classification.

2) DogCentric: The DogCentric Activity dataset[3] [21] consists of first-person videos with outdoor scenes of wearable cameras mounted on dogs' back. The dataset contains 209 videos, where each video has a dog performing one of the 10 actions: "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*), "walking" (*walk*). Not all dogs perform all actions and an action can be performed more than once by the same dog. The dataset is unbalanced according to the number of frames for each action and contains 4,920 frames of *Car*, 3,300 frames of *Drink*, 3,795 frames of *Feed*, 1,950 frames of *Left*, 1,380 frames of *Right*, 3,740 frames of *Pet*, 3,545 frames of *Play*, 1,880 frames of *Shake*, 4,960 frames of *Sniff* and 4,175 frames of *Walk*, adding up to 33,645 frames.

3) UCF-11: UCF YouTube Action Dataset[4] [23] 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*. All videos in the dataset were manually collected from YouTube with a fixed resolution of 240 $\times$ 320 pixels. Each video was converted to a frame rate of 29.97 *fps* and annotations were done accordingly, containing a single action associated with the entire video. The entire dataset contains 262,286 frames in total. This dataset is very challenging due to large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination conditions, etc.

B. Deep Learning Algorithms

In this work, we measure and analyze the performance of deep architectures, such as Convolutional Neural Networks (CNNs) and a Long Short-Term Memory (LSTM) for action recognition in small datasets. Recently, deep architectures achieved great success in classifying images containing objects, scenes and complex events [12], [25], [26], and problems in video-based action recognition [10], [11], [27]. As described by Wang *et al.* [17], deep CNNs come with great modeling capacity and are capable of learning discriminative representation from raw visual data with the help of large-scale supervised datasets. In this work, we test different architectures of CNNs as well as an architecture containing an LSTM.

Due to hardware limitations, we could not set up the 3D Convolution Network (C3D) proposed by Tran *et al.* [11]. This architecture uses 3D convolutional layers to extract spatiotemporal features from a sequence of frames. As 3D convolutions increase enormously the amount of memory needed for training, we could not test this model.

1) Convolutional Neural Networks: CNNs have recently enjoyed a great success in large-scale image and video recognition. This success has become possible due to the advance in technology with high-performance computing systems [28] and the availability of large public datasets, such as ImageNet [29]. In fact, since the successful results achieved by AlexNet [25] in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [30] in 2012, the quality of deep architectures significantly improved. The deep architectures we use in this paper are described as follows.

VGG16: The VGG networks [12] developed by the Visual Geometry Group (VGG) from Oxford exhibit a simple yet effective strategy of constructing deep Convolutional Neural Networks: stacking building blocks of the same shape. VGG16 contains 16 weight layers divided in groups of convolutions

[1]http://www.murase.m.is.nagoya-u.ac.jp/KSCGR/

[2]http://www.icpr2012.org/

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

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

using 3×3 filters in each convolutional layer to represent complex features. VGG comes with the idea that multiple 3×3 convolution in sequence can emulate the effect of larger receptive fields, such as the 5×5 and 7×7 used in previous networks. In the whole pipeline, the network receives a sequence of images in the input, passing them through several convolutional layers, pooling layers and fully-connected layers, ending in a *softmax* function that generates the probability of the input image to each class.

The robustness of VGG networks has been proven by various visual recognition tasks [3], [16], [31], [32]. Although VGG networks have the compelling feature of architectural simplicity, they require a lot of computation. When compared with previous networks such as AlexNet [25], the VGG architecture employs three times more parameters. In order to reduce the number of parameters, other deep networks were proposed, such as Inception-V3 [33].

Inception-v3: Google's Inception-V3 [33] (hereafter called V3) is a 48-layers deep Convolutional Neural Network based on *inception* modules. Such modules contain convolutional filters with a number of different dimensionalities running in parallel, covering different clusters of information. The main idea of the *inception* modules is the split-transform-merge strategy, where the input is split into low dimensional embeddings, transformed by a set of different size filters, and merged by concatenation. These modules intent to approach the representational power of large and dense layers, having a lower computational complexity.

V3 is an evolution of the GoogLeNet [26], which was a state of the art network in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [30] in 2014. In this challenge, V3 achieved 4.2% error on the ILSVRC 2012 classification benchmark. The main improvements in the V3 architecture when compared with its previous version rely on the increase in depth, from 22 layers to 48 layers, and a reduction in the size of the filters to a maximum of 3×3. *Inception* architecture of GoogLeNet is designed to perform well with a reduced number of parameters when compared with the previous architectures. For example, GoogLeNet contains about 7 million parameters, which represents a 9 times reduction with respect to AlexNet [25] that contains 60 million parameters.

Two-Stream Network: Proposed by Simonyan and Zisserman [10], the network computes two separated recognition streams (a spatial containing still images and a temporal containing the motion performed in a sequence of images). Both networks are merged by late fusion, where the prediction scores are combined by concatenation or by the mean. This network intends to fuse the features extracted from static images by a CNN with the features extracted by passing the optical flow of images through another CNN. In our work, we use the VGG-16 network in both streams, receiving still frames from videos in the static stream and images generated by optical flow in the temporal stream.

2) Recurrent Neural Network: Recurrent Neural Network (RNN) models are designed to deal with sequential information, *i.e.*, recognize patterns in data using not only the current input, but also the information they have perceived previously in time. Such networks are called recurrent because they perform the same task for each element in a sequence with the output being dependent on the previous information. The recurrent unit can be understood as a memory about what the network has processed so far. Although RNN models are suitable to deal with temporal data such as video sequences, it has a significant limitation known as the "vanishing gradient", which make it difficult to backpropagate an error signal through a long-range temporal interval, and consequently, being difficult to train them to learn long-term dynamic. In order to deal with this problem, Hochreiter and Schmidhuber [34] propose a type of recurrent neural network called Long Short-Term Memory (LSTM), which enables long-range learning by incorporating memory units that allow the network to learn, to forget and to update hidden states given new information.

LRCN: In this work, we use an adaptation of the Long-term Recurrent Convolutional Network (LRCN) [15], which combines a deep hierarchical visual feature extractor (CNN) with a model that can learn to recognize and synthesize temporal dynamics for tasks involving sequential data (LSTM). The main idea of LRCN is to extract features from images using a CNN and then use the LSTM to learn patterns from the sequence of images that represent an action. Our LRCN differs from the original since we do not train it end-to-end, using the CNN as a feature extractor to the recurrent network.

III. Experiments

In this section, we describe the main implementation details applied in our experiments. We describe how each dataset is divided into train, validation and test in order to perform the experiments, pre-processing steps, and the hyperparameters applied in each deep architecture.

A. Data pre-processing

Pre-processing consists of two steps: image resizing and optical flow generation. Resizing is important since it reduces the multidimensional space required by the CNNs to learn suitable features for image classification, as well as the total processing time. In this step, all images of the datasets are resized to a fixed resolution of 256×256. The second step generates the dense optical flow representation [35] of adjacent frames. In a nutshell, optical flow represents the 2D displacement of pixels between frames generating vectors corresponding to the movement of points from the first frame to the second. Dense optical flow generates the displacement vectors for both horizontal and vertical displacements, regarding all points within frames. In order to generate an image containing the optical flow from the sequence of frames, we combine the 2-channel optical flow vectors (horizontal and vertical), associating color to their magnitude and hue value to their direction. The output of the data pre-processing step consists of two datasets (a) RGB images containing 256×256 images, and (b) optical flow images containing 256×256 encoding the motion across frames.

B. Dataset Division

In order to perform experiments, we divided all datasets used in this paper into training, validation, and test sets. The division of each dataset is performed as follows:

DogCentric: Our method to divide the dataset is similar to the one proposed by Iwashita *et al.* [21] and consists of randomly selecting half of the videos of each activity as test set. 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. In case where the number of videos (N) is an odd number, we separate $\frac{(N+1)}{2}$ videos for the test set. The rest of the videos are divided into training and validation sets. Validation set contains 20% of the videos and the rest is separated to the training set. The complete division contains 105 videos (17,400 frames) in testing set, 20 videos (3,205 frames) in validation set and 84 videos (13,040 frames) in training set.

KSCGR: For this dataset we follow the same division performed in ICPR 2012[5] challenge and consists of a test set containing 2 subjects, each performing 5 recipes, *i.e.*, 10 videos with 55,781 frames in total. The rest of the dataset is divided into train and validation sets, where the training set contains 4 subjects, each of them performing 5 recipes, *i.e.*, 20 videos and 139,196 frames in total, and the validation set contains 1 subject performing 5 recipes with 32,897 frames in total.

UCF-11: In this dataset, each class contains 25 folders with at least 4 action videos in it. We divided the videos into train (folders 1-13), validation (folders 14-16) and test (folders 17-25). The training, validation and test set contain all the 11 classes available in the dataset and respectively 824, 189 and 589 videos. The final division contains 133,551 frames for the training set, 30,699 frames for the validation set and 98,036 frames for the test set.

C. Model Training

In order to identify our models, each model's name is composed by the name of the CNN (V3 for Inception-V3 and VGG16 for a VGG network containing 16 weight layers) followed by the type of input (RGB for raw images and OF for optical flow), or by the type of classifier (LRCN for Long-term Recurrent Convolutional Network and SVM for Support Vector Machine) or the method of fusion used in the two-stream network (2STREAM(CONCAT) when concatenating the extracted features and 2STREAM(MEAN) when calculating the mean of each feature). When the model uses the extracted features, we present the name of the layer inside brackets (*e.g.*, V3[CONV] or VGG16[CONV] when extracting features from the last convolutional layer and V3[FC] or VGG16[FC] when extracting features from the fully connected layer. Thus, we may have a model named V3-RGB that classifies raw images using the default architecture of the Inception-V3 network. A model named VGG16[CONV]-SVM classifies the features extracted from the last convolutional layer of the VGG-16 network using a Support Vector Machine. A model named V3[FC]-LRCN uses the Long-term Recurrent Convolutional Network to classify a sequence of features extracted from the fully connected layer of the Inception-V3 network. Finally, a model named VGG16-2STREAM(MEAN) performs the classification of the mean of the features extracted from two VGG-16 networks containing one steams with raw images and another stream with optical flow images.

Hyperparameter optimization: In this work we use grid search in order to find the best hyperparameters for our models. Grid search is simple and naive, however it is a good approach when models do not take so long to train. Basically, the grid search tests all the combination between all the hyperparameters trying to find the model that best fit into the problem. We focus the grid search in *dropout* and *learning rate* hyperparemeters, since they are commonly changed when trying to learn a deep model. For dropout, we set the values 0.5, 0.7, 0.9 and 0.95, and for learning rate, we set $5e^{-3}$, $1e^{-3}$, $5e^{-4}$, $1e^{-4}$, $5e^{-5}$ and $1e^{-5}$. As mentioned before, the grid search method will test all the possible combinations, thus, using 4 different values for dropout and 6 values for learning rate we have in a total 24 different combinations to perform. As we apply the grid search in 8 models (V3-RGB, V3-OF, V3[FC]-LRCN, V3[CONV]-LRCN, VGG16-RGB, VGG16-OF, VGG16[FC]-LRCN, and VGG16[CONV]-LRCN), we obtain 192 trained models plus 8 models (V3[FC]-SVM, V3[CONV]-SVM, V3-2STREAM(MEAN), V3-2STREAM(CONCAT), VGG16[FC]-SVM, VGG16[CONV]-SVM, VGG16-2STREAM(MEAN), and VGG16-2STREAM(CONCAT)) that do not use grid search for each dataset. Summing up, we have a total of 600 trained models for all datasets used in this paper.

Architecture configuration: All deep models used in this work are implemented using *Keras*[6] and *TensorFlow*[7] frameworks. As highly recommended when using small datasets, we pre-trained all the CNNs using the ImageNet dataset with weights being directly loaded from Keras core library. For all CNNs the training phase uses mini-batch stochastic gradient with momentum (0.9). The activation of each convolution (including those within the "inception" modules in V3) uses a rectified linear unit (ReLU). In each iteration we use a mini-batch with 128 images. To reduce the chances of overfitting, we apply dropout on the fully-connected layers with the range previously mentioned. Since we are working with small datasets we do not need too many epochs to converge the model, with that in mind we set the maximum limit of epochs to 30. In fact, we implemented an early stopping and most of our experiments take no longer than 15 epochs to finish. For the SVM model, we use the Crammer and Singer [36] implementation of the SVM is from *scikit-learn*[8] [37] with

[5]http://www.icpr2012.org/

[6]https://keras.io/

[7]https://www.tensorflow.org/

[8]http://scikit-learn.org/

the default parameters from *scikit-learn*.

IV. Results and Analysis

In order to evaluate deep learning models, we compare the output of each network using the test set. Table I shows the accuracy values for all our developed architectures in each dataset individually. As we tested a total of 600 models, Table I only presents the scores based on the best combination of learning rate and drop out for each architecture. The whole content of our experiments as well as the results for all models can be found in the project's website[9].

TABLE I: Accuracy, Precision, Recall and F-Score for all models using three datasets.

Dataset	Architecture	*A*	*P*	*R*	*F*
DogCentric	V3-RGB	0.571	0.680	0.570	0.590
	V3-OF	0.128	0.030	0.130	0.040
	V3[FC]-LRCN	0.594	0.670	0.590	0.600
	V3[CONV]-LRCN	0.581	0.680	0.580	0.590
	V3[FC]-SVM	0.558	0.660	0.560	0.560
	V3[CONV]-SVM	0.550	0.650	0.550	0.550
	V3-2STREAM(CONCAT)	0.556	0.690	0.560	0.570
	V3-2STREAM(MEAN)	0.541	0.670	0.540	0.550
	VGG16-RGB	0.540	0.588	0.541	0.538
	VGG16-OF	0.118	0.110	0.120	0.090
	VGG16[FC]-LRCN	0.556	0.590	0.560	0.550
	VGG16[CONV]-LRCN	**0.600**	0.640	**0.600**	**0.600**
	VGG16[FC]-SVM	0.508	0.620	0.510	0.500
	VGG16[CONV]-SVM	0.301	**0.760**	0.300	0.340
	VGG16-2STREAM(CONCAT)	0.535	0.620	0.540	0.540
	VGG16-2STREAM(MEAN)	0.529	0.610	0.540	0.540
KSCGR	V3-RGB	0.615	**0.773**	0.615	0.660
	V3-OF	0.630	0.650	0.630	0.600
	V3[FC]-LRCN	**0.733**	0.750	**0.730**	**0.710**
	V3[CONV]-LRCN	0.729	0.740	**0.730**	0.700
	V3[FC]-SVM	0.723	0.740	0.720	0.680
	V3[CONV]-SVM	0.713	0.730	0.710	0.670
	V3-2STREAM(CONCAT)	0.722	0.750	0.720	0.680
	V3-2STREAM(MEAN)	0.721	0.740	0.720	0.690
	VGG16-RGB	0.592	0.590	0.590	0.550
	VGG16-OF	0.415	0.460	0.420	0.370
	VGG16[FC]-LRCN	0.571	0.560	0.570	0.550
	VGG16[CONV]-LRCN	0.560	0.600	0.560	0.510
	VGG16[FC]-SVM	0.563	0.600	0.560	0.480
	VGG16[CONV]-SVM	0.499	0.290	0.500	0.360
	VGG16-2STREAM(CONCAT)	0.604	0.620	0.600	0.570
	VGG16-2STREAM(MEAN)	0.598	0.650	0.600	0.560
UCF-11	V3-RGB	0.744	0.760	0.740	0.740
	V3-OF	0.668	0.680	0.670	0.660
	V3[FC]-LRCN	0.752	0.770	0.750	0.750
	V3[CONV]-LRCN	0.752	0.750	0.750	0.740
	V3[FC]-SVM	0.750	0.780	0.750	0.750
	V3[CONV]-SVM	0.757	0.790	0.760	0.760
	V3-2STREAM(CONCAT)	0.755	0.770	0.760	0.760
	V3-2STREAM(MEAN)	**0.786**	**0.800**	**0.790**	**0.780**
	VGG16-RGB	0.708	0.700	0.710	0.700
	VGG16-OF	0.480	0.470	0.480	0.470
	VGG16[FC]-LRCN	0.711	0.710	0.710	0.700
	VGG16[CONV]-LRCN	0.752	0.750	0.750	0.740
	VGG16[FC]-SVM	0.696	0.730	0.700	0.700
	VGG16[CONV]-SVM	0.397	0.770	0.400	0.410
	VGG16-2STREAM(CONCAT)	0.710	0.730	0.710	0.710
	VGG16-2STREAM(MEAN)	0.720	0.720	0.720	0.720

[9]https://github.com/jrzmnt/ActionRecognitionSmallDatasets

A. *Overall Performance*

Observing Table I, in general, architectures that take into account the temporal aspect (LRCN, 2STREAM) achieved better results when compared to approaches that use only static frames. For example, the best results for DogCentric and KSCGR are achieved by approaches using LRCN, while for UCF-11 it is achieved by the two-stream network. We faced some relevant correlations obtained in the validation phase in aspect of hyperparameters. Due to space constraints, we do not add the values of learning rate and drop out in Table I. However, we note that the best results were achieved using high values of dropout and low values of learning rate parameter for all datasets used in this research.

As it is interesting to have not only a global view of the accuracy of the models but also a better visualization about how well the model performs on different actions, we generate a confusion matrix as illustrated in Figure 1. The normalized confusion matrix shows the performance of the best models for each dataset used in this paper, where rows represent the true classes and columns the predicted classes. Shades of blue represent the value in each cell, going chromatically from a darker blue for higher values to a lighter blue for lower values. We can observe that actions are easier to classify in UCF-11 than the other datasets, probably due to the low inter-class similarity. Actions in datasets with high inter-class similarity like DogCentric are harder to classify, having many misclassified classes. For example, the class "Right" in DogCentric is usually misclassified as "Left", "Play" and "Walk", as well as the class "Peeling" in KSCGR is misclassified as "Breaking".

B. *Static Networks*

By static networks, we consider the networks that do not take into account the temporal aspect, *e.g.*, V3-RGB, V3[*]-SVM, VGG16-RGB, VGG16[*]-SVM. In general, it is a challenge task for a static approach to classify actions since it is not using the temporal aspect. Despite such challenge, the static networks obtained good results when compared with approaches that include the temporal aspect.

Looking at the results obtained using the DogCentric dataset, the best result is achieved using the Inception-V3 with RGB images, which obtained the follow results, 57% of accuracy, 68% of precision, 57% of recall and 59% of F-score. For the KSCGR dataset, the best static model obtained 72.3%, 74%, 72% and 68% of respectively accuracy, precision, recall, and F-score, using the features extracted from the *fc* layer of the Inception-V3 network and then passing these features to the SVM. The best results for UCF-11 is using an Inception-V3 architecture with extracted features from the *conv* layer and then passing these features to the SVM. This model achieved 75.7% of accuracy, 79% of precision, 76% of recall and 76% of F-score. In general, we can see that the architecture that stands out over the others is the Inception-V3 using the SVM as a classifier, which obtained the best results for the static approaches for the three datasets used.

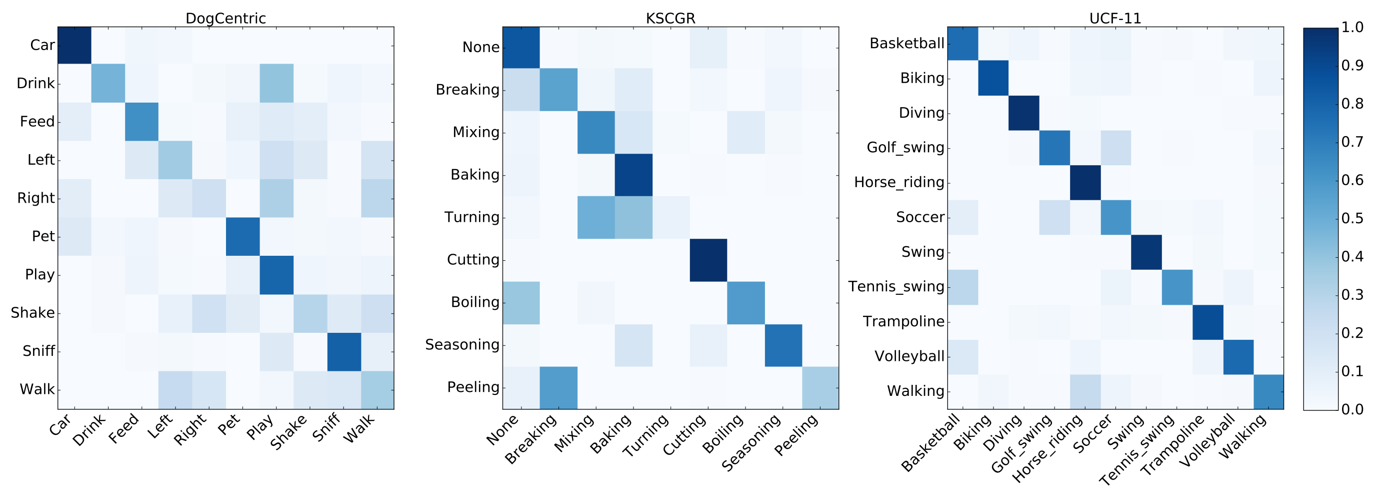

Fig. 1: Normalized confusion matrix for the best models in respectively DogCentric, KSCGR and UCF-11 datasets.

C. *Static Networks vs. Two-Stream Networks*

For the most models which used the two-stream architecture, they could not surpass the results obtained by static networks. As we can see, the best results for a static network using the DogCentric dataset is 57% of accuracy, 68% of precision, 57% of recall and 59% of F-score. Comparing these results with the best two-stream combination for DogCentric that obtained 55.6%, 69%, 56%, 57% of respectively accuracy, precision, recall, and F-score, we can see that the two-stream model only surpass in 1% of precision the static model. This difference occurs because the network using optical flow achieved low results in this dataset, pushing down the score of two-stream networks.

Experiments using the other datasets show that the two-stream models surpass the static networks. The most relevant difference between the results using static and two-stream approaches are presented when using the UCF-11 dataset, where the best results are achieved by the mean of the streams using the Inception-V3 network. This model achieved a difference of $\approx 3\%$ of accuracy, $\approx 1\%$ of precision, $\approx 3\%$ of recall, $\approx 2\%$ of F-score, when compared with the best static model in UCF-11 dataset. We believe that the two-stream results were not so significant because this kind of approach requires more data to learn how to capture the context of the action through time. Even if the UCF-11 is considered a small dataset, we know that this dataset is larger than the other two used in this work, increasing the results to the two-stream approach when compared with other datasets.

D. *Static Networks vs. Recurrent Networks*

In general, recurrent neural networks performed slightly better than static networks. The best configuration extracts features using the last convolutional layer of the CNN and feed the LSTM to classification. It seems better to use the Inception-V3 CNN to extract features instead of VGG-16 before passing the features to the LSTM. In fact, VGG16[*]-LRCN and V3[*]-LRCN achieve similar results in most datasets but KSCGR where V3[FC]-LRCN obtains 73% of accuracy while VGG16[FC]-LRCN obtains only 57%.

When comparing static networks with recurrent network approaches, the difference between the best accuracies is $\approx 3\%$ for DogCentric dataset, $\approx 1\%$ for KSCGR dataset and $\approx 1\%$ for UCF-11 dataset. It is important to note that for the UCF-11 dataset the static model which uses *conv* features from an Inception-V3 architecture surpass the results obtained for the recurrent models. We suspect that the small difference between the results using static and recurrent model in UCF-11 dataset ($\approx 1\%$) is related to the fact that the dataset contains short videos, making it difficult to the recurrent model to learn temporal features. Finally, it would be worth to choose static models over recurrent models, since the difference between the results of both models is not so significant, and because static models are faster to train and to classify than recurrent models.

E. *Recurrent Networks vs. Two-Stream Networks*

Comparing the temporal approaches, we can see that recurrent networks and two-stream achieve the best results for all the datasets we use in this work. The difference between the results is not higher than 3.5% of accuracy, 5% of precision and 4% of recall. The main issue on using two-stream is its dependency of having two trained models to perform the final classification. We train one model with optical flow images and the other with raw RGB images. If the optical flow images are not so representative, the neural network model will not generate good results, which leads the two-stream to limited results, as can be seen when using the DogCentric dataset. On the other hand, recurrent networks that do not have the dependency of optical flow as two-stream models may have some issues when trying to learn features from short videos.

V. Comparison with the State of the Art

In this section, we compare the results we achieved with the results achieved by the state-of-the-art algorithms in each dataset. Table II presents the precision, recall, and F-scores for our best models and for the state-of-the-art algorithms. Missing values in the table means that the work does not present any measure in the paper, *e.g.*, the work of Iwashita *et al.* [21] performs action recognition using Linear Kernel in the DogCentric dataset, reporting only values of accuracy, and thus, the columns P, R, F in the table do not contain any value. Below we explain each work described in Table II.

A. DogCentric dataset

Iwashita *et al.* [21] perform the activity recognition extracting hand-crafted global features from dense optical flow and local features from a cuboid feature detector [38] and STIP detector [39]. 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.* [40] 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% of accuracy when combined with INRIA's improved trajectory feature (ITF) [7].

Monteiro *et al.* [5] apply a two-stream network containing two CNNs with different configuration running in parallel in the DocCentric dataset. After a pre-processing step to resize images, both streams extract different features that are merged by a late score fusion using SVM. A post-processing step is applied in order to reduce noisy predicted classes, and consist of using a sliding window with a fixed size that assures to the target frame the majority voting of all frames within the window. Their approach achieved 74% of precision, 76% or recall, 75% of F-score and 76% of accuracy.

B. KSCGR dataset

Bansal *et al.* [22] perform daily life cooking activity recognition based on hand-crafted features for hand movements and objects use. Their method uses hand regions as well as objects in use to feed a dynamic Support Vector Machine (SVM) and Hidden Markov Model (HMM) hybrid model to combine the structural and temporal information to jointly infer the activity. Their method achieves 64% of overall accuracy, 62% of precision, 63% of recall and 60% of F-score. A post-processing on the predicted actions is performed by creating a context grammar to select the most likely guess for misclassified frames. Using the post-processing step, they increased accuracy in $\approx$ 7%, achieving a final accuracy of 72%, 68% of precision, 68% of recall and 72% of F-score.

TABLE II: State of the art results to each dataset.

Dataset	Architecture	A	P	R	F
DOG	Linear kernel [21]	0.53	-	-	-
	RBF kernel [21]	0.54	-	-	-
	Histogram intersection [21]	0.57	-	-	-
	Multi-channel [21]	0.61	-	-	-
	PoT+STIP+Cuboid [40]	0.73	-	-	-
	PoT+ITF+STIP+Cuboid [40]	0.74	-	-	-
	PoT+ITF [40]	0.75	-	-	-
	2 CNNs-SVM [5]	0.68	0.69	0.68	0.69
	2 CNNs-SVM-PP [5]	**0.76**	**0.74**	**0.76**	**0.75**
	VGG16[CONV]-LRCN	0.60	0.64	0.60	0.60
KSCGR	HCF [22]	0.64	0.62	0.63	0.61
	HCF-PP [22]	0.68	0.68	0.68	0.72
	3 CNNs-LSTM [4]	0.78	0.78	0.78	0.78
	3 CNNs-LSTM-PP [4]	**0.79**	**0.80**	**0.78**	**0.79**
	V3[FC]-LRCN	0.73	0.75	0.73	0.71
UCF-11	DTAM [41]	0.90	-	-	-
	Visual-DTAM [41]	**0.91**	-	-	-
	V3-2STREAM(MEAN)	0.79	**0.80**	**0.79**	**0.78**

Monteiro *et al.* [4] merge three different architectures of CNNs (AlexNet [25], GoogLeNet [26] and SqueezeNet [42]) using a Long-Short Term Memory (LSTM) [34]. All the architectures are pre-trained on the 1.3-million-image ILSVRC 2012 ImageNet dataset [30] and CNNs are used as feature extractor to the LSTM classification. Using this approach, they achieve 78% in all measures. As developed in Monteiro *et al.* [5], a post-processing represented by a sliding window is applied on the predicted classes, increasing the accuracy to 79%, the precision to 80%, and the F-score to 79%.

C. UCF-11 dataset

Wang *et al.* [41] uses a dynamic tracking attention model (DTAM) composed by a CNN and an LSTM to recognize human action in video sequences. While the CNN performs feature extraction, the LSTM is applied to handle sequential information about actions that are extracted from videos. 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 [43] with their proposed DTAM, they achieve 91% of accuracy. Values of precision, recall, and F-score are not reported in their work.

VI. Conclusions and Future Work

In this work, we presented an analysis using deep learning algorithms in small datasets for action recognition. We trained a total of 600 models using different deep learning approaches. We tested different deep architectures using different hyper-parameters in order to have a large number of results for each dataset. We achieved relevant results with deep learning models when compared with the state-of-the-art results for the datasets used in this work, which leads us to believe that deep learning approaches do work properly for small datasets. We perform a discussion about the results and compare the approaches in order to shed a light on the approaches when using small datasets. Finally, we discuss the viability of using the best models generated.

As future work, we intend to construct deep learning approaches capable of suppressing the limitations of the small datasets, *i.e.*, overfitting. Avoid overfitting is not an easy task, thus, we intend to build an architecture that focuses on regularization and that does not have an abundance of parameters.

Acknowledgement

We would like to thank Motorola Mobility for funding this work.

References

[1] J. Wehrmann and R. C. Barros, "Convolutions through time for multi-label movie genre classification," in *Proceedings of SAC'17*, 2017, pp. 114–119.

[2] G. S. Simões, J. Wehrmann, R. C. Barros, and D. D. Ruiz, "Movie genre classification with convolutional neural networks," in *Proceedings of IJCNN'16*, 2016, pp. 259–266.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[4] J. Monteiro, R. Granada, R. C. Barros, and F. Meneguzzi, "Deep neural networks for kitchen activity recognition," in *Proceedings of IJCNN'17*, 2017, pp. 2048–2055.

[5] J. Monteiro, J. P. Aires, R. Granada, R. C. Barros, and F. Meneguzzi, "Virtual guide dog: An application to support visually-impaired people through deep convolutional neural networks," in *Proceedings of IJCNN'17*, 2017, pp. 2267–2274.

[6] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Action recognition by dense trajectories," in *Proceedings of CVPR'11*, 2011, pp. 3169–3176.

[7] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *Proceedings of ICCV 2013*, 2013, pp. 3551–3558.

[8] V. Delaitre, I. Laptev, and J. Sivic, "Recognizing human actions in still images: A study of bag-of-features and part-based representations," in *Proceedings of BMVA 2010*, 2010, pp. 97.1–97.11.

[9] B. Yao and L. Fei-Fei, "Grouplet: A structured image representation for recognizing human and object interactions," in *Proceedings of CVPR'10*, 2010, pp. 9–16.

[10] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Proceedings of NIPS'14*, 2014, pp. 568–576.

[11] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of ICCV'15*, 2015, pp. 4489–4497.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of ICLR'15*, 2015.

[13] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Proceedings of AAAI 2017*, 2017, pp. 4278–4284.

[14] A. Piergiovanni, C. Fan, and M. S. Ryoo, "Title learning latent subevents in activity videos using temporal attention filters," in *Proceedings of AAAI'17*, 2017, pp. 4247–4254.

[15] J. Donahue, L. A. Hendricks, M. Rohrbach, S. Venugopalan, S. Guadarrama, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 677–691, 2017.

[16] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," in *Proceedings of CVPR'16*, 2016, pp. 1933–1941.

[17] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. V. Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *Proceedings of ECCV'16*, 2016, pp. 20–36.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of ICML'15*, 2015, pp. 448–456.

[19] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, January 2013.

[20] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proceedings of CVPR'17*, 2017, pp. 4724–4733.

[21] Y. Iwashita, A. Takamine, R. Kurazume, and M. S. Ryoo, "First-person animal activity recognition from egocentric videos," in *Proceedings of ICPR'14*, 2014, pp. 4310–4315.

[22] S. Bansal, S. Khandelwal, S. Gupta, and D. Goyal, "Kitchen activity recognition based on scene context," in *Proceedings of ICIP 2013*, 2013, pp. 3461–3465.

[23] J. Liu, J. Luo, and M. Shah, "Recognizing realistic actions from videos "in the wild"," in *Proceedings of CVPR'09*, 2009, pp. 1996–2003.

[24] A. Shimada, K. Kondo, D. Deguchi, G. Morin, and H. Stern, "Kitchen scene context based gesture recognition: A contest in icpr2012," in *Proceedings of WDIA'12*, 2013, pp. 168–185.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of NIPS'12*, 2012, pp. 1097–1105.

[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of CVPR'15*, 2015, pp. 1–9.

[27] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of CVPR'14*, 2014, pp. 1725–1732.

[28] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *Proceedings of NIPS'12*, 2012, pp. 1223–1231.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of CVPR'09*, 2009, pp. 248–255.

[30] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[31] R. Girshick, "Fast r-cnn," in *Proceedings of ICCV '15*, 2015, pp. 1440–1448.

[32] G. Cheron, I. Laptev, and C. Schmid, "P-cnn: Pose-based cnn features for action recognition," in *Proceedings of ICCV 2015*, 2015, pp. 3218–3226.

[33] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of CVPR'16*, 2016, pp. 2818–2826.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[35] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Proceedings of SCIA'03*, 2003, pp. 363–370.

[36] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *Journal of Machine Learning Research*, vol. 2, no. Dec, pp. 265–292, 2001.

[37] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *Journal of Machine Learning Research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[38] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, "Behavior recognition via sparse spatio-temporal features," in *Proceedings of VSPETS 2005*, 2005, pp. 65–72.

[39] I. Laptev, "On space-time interest points," *International Journal of Computer Vision*, vol. 64, no. 2-3, pp. 107–123, 2005.

[40] M. S. Ryoo, B. Rothrock, and L. Matthies, "Pooled motion features for first-person videos," in *Proceedings of CVPR 2015*, 2015, pp. 896–904.

[41] C.-Y. Wang, C.-C. Chiang, J.-J. Ding, and J.-C. Wang, "Dynamic tracking attention model for action recognition," in *Proceedings of ICASSP'17*, 2017, pp. 1617–1621.

[42] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *arXiv:1602.07360*, 2016.

[43] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu, "Recurrent models of visual attention," in *Proceedings of NIPS'14*, 2014, pp. 2204–2212.