Virtual Guide Dog: An Application to Support Visually-Impaired People through Deep Convolutional Neural Networks

Juarez Monteiro*, João Paulo Aires*, Roger Granada*, Rodrigo C. Barros†, and Felipe Meneguzzi†
Faculdade de Informática
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, felipe.meneguzzi}@pucrs.br

***Abstract*—Activity recognition applications is growing in importance due to two key factors: first there is increased need for more human assistance and surveillance; and second, increased availability of datasets and improved image recognition algorithms have allowed effective recognition of more sophisticated activities. In this paper we develop an activity recognition approach to support visually impaired people that leverages these advances. Specifically, our approach uses a dataset of videos taken from the point of view of a guide-dog to train a convolutional neural-network to recognize the activities taking place around the camera and provide feedback to a visually impaired human user. Our experiments show that our trained models surpass the current state-of-the-art for identifying activities in the doc-centric activity dataset.**

I. Introduction

Animals use many different senses to orientate themselves, but humans use primarily vision for ambient orientation. Part of the process of orientation in humans involve memory used in the construction of what specialists call mental maps [1]. Mental mapping of spaces is essential for the development of efficient orientation and mobility. Blind and visually impaired people lack an important source of information (vision) to build such mental maps and have to adapt themselves to build their mental maps using alternative sources of information. Such alternative sources primarily involve sound and tactile feedback, which are often used in assistive aids such as tactile pavements and audible pedestrian crossing [2]. Blind navigation is a cognitively demanding and often requires conscious moment-to-moment problem solving [3], and, in environments where such aids are lacking or non-existent, blind and visually impaired people have substantial difficulty navigating. One of the earliest mobile navigation aid used by visually impaired people is the guide dog [4], however, although guide dogs help navigate obstacles safely, there are limitations to what a dog can perceive and communicate to its owner. In this paper, we develop an approach to use video data from dog-mounted cameras to provide additional feedback for visually impaired people, with the ultimate aim of providing a richer source of information for the construction of mental maps.

With the growth of video content produced by mobile cameras and surveillance systems, an increasing amount of data is now available and can be used for a variety of applications besides the application responsible for the original data collection. Specifically, we found a video dataset collected from dog-mounted cameras originally collected to recognize the activities being performed by the camera-carrying dog [5]. Our approach uses this data to recognize the activities performed by the guide dog as a proxy for aiding the visually impaired owner and providing audible cues describing the environment surrounding the guide dog. Activity recognition from video is a particularly challenging task and state of the art methods have poor performance with specific data, *e.g.* videos that have dynamic background [6]. Since effective aid relies on accurately recognizing the dog's activity in order to provide correct feedback for the user, we leverage advances on deep learning algorithms in general, and Convolutional Neural Networks (CNNs) [7], [8] in particular, to consistently improve on the state-of-the-art. Our contribution is twofold. First, we develop an aid architecture for visually impaired people that translates recognized activities into audible feedback, enhancing the amount of information a guide dog can provide. Second, we train a deep-learning model for activity recognition using dog-centric data that surpasses the state-of-the-art and obviates the need for manual feature construction. We empirically evaluate the resulting model using the DogCentric Activity dataset [5], and we show that our proposed approach outperforms the current state-of-the-art method [9] for this particular dataset.

This paper is organized as follows. Section II details our deep neural models and the implemented support vector machine for action recognition. Section III describes the experimental set up including the dataset, its preprocessing and how we trained the machine-learning algorithms. Section IV presents a thorough analysis of the experiments assessing the performance of our proposed approach, and Section V focuses on the overall architecture of the aid application we developed using the activity recognition model. Finally, Section VI compares our work to the state-of-the-art and other related work, and Section VII finishes the paper with our conclusions and future work directions.

II. Method

Conventional machine-learning techniques are often limited in their ability to process natural data in their raw form [10]. For decades, constructing machine-learning systems required considerable domain expertise to create an internal representation (feature construction [11]) from which the learning subsystem could detect or classify patterns in the input. Deep learning techniques such as Convolutional Neural Networks mitigate this problem by automatically learning complex representations from unstructured data such as images, videos and audio. In recent years, CNNs have been shown to accurately classify images [12], [13] and videos [14], [15], [16] and a number of architectures have been proposed [12], [13], [17], [18].

Since many architectures have been proposed, we believe that different architectures may identify different features in images, helping to improve the classification task. In this paper, we use two different architectures in parallel and combine their output by late fusion of the different networks. Specifically, we employ the *AlexNet* and *GoogLeNet* networks in the pipeline illustrated by Figure 1. In this pipeline, both networks receive the same pixel data from an input image and each CNN extracts features in parallel independently. The architecture of our approach is similar to two-stream networks [17] but instead of using optical flow to generate features in one of the networks, we let each individual network identify different features, under the assumption that the features generated by the networks will be complementary.

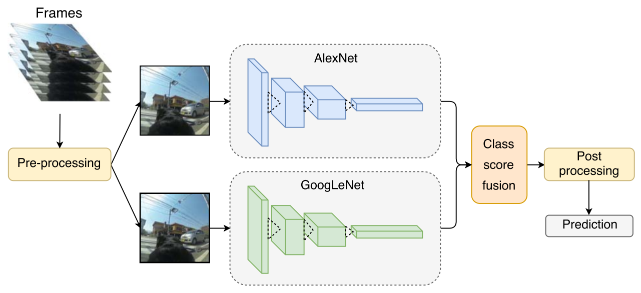

Fig. 1. Pipeline of our architecture for activity recognition.

Pre-processing: The pre-processing step consists of resizing images to a fixed resolution of 256×256. Resizing is important because it reduces the number of features for each image inside the CNN as well as the processing time.

CNNs: Our architecture has two CNNs that run in parallel. First is *GoogLeNet* [18] due to its reduced number of parameters generated by *inception* modules. And the second is a slightly modified version of *AlexNet* [12] called *CaffeNet* [19] due to its small architecture, which is able to provide a fast training phase. Although the AlexNet model provided by *Caffe* has some small differences from the AlexNet by Krizhevsky [12], we do not believe our results would be changed by small architectural and optimization differences. 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. *GoogLeNet* is a 22-layer deep network and is based on the *inception* module that contains convolutional filters with different sizes, covering different clusters of information. 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.

Class score fusion: A late fusion combines vectors of probabilities generated by each CNN by stacking the *softmax* output of each CNN. Although other architectures employ fusion methods such as averaging the probabilities of both vectors [17], in this work we train a multi-class Support Vector Machine (SVM) using the stacked L2-normalized *softmax* scores as features.

Post-processing: This step intends to reduce noisy predicted classes and consists of a smoothing pass on the sequence of the classes. We post-process the output of the SVM by smoothing this output 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. We chose the window size after several tests using the predicted classes from the validation dataset and varying the window size from 10 to 50 in 10 frame increments, ultimately choosing a 10-frame window.

III. Experiments

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

A. Dataset

The DogCentric Activity dataset[1] [5] consists of first-person videos with outdoor scenes of wearable cameras mounted on dogs' back. These kind of videos (first-person videos - also known as egocentric videos) are very challenging due to their strong camera motion. The DogCentric Activity dataset contains dog activities videos in 320×240 resolution (recorded at 48 frames per second) taken from a egocentric animal viewpoint. The dataset is divided into 209 videos containing 10 different activities performed by 4 dogs. Not all dogs perform all activities and an activity can be performed more than once by the same dog. These activities include movements performed by the dog himself (*e.g.*, running, shaking, *etc.*), interactions with people (*e.g.*, petting, feeding, *etc.*) and interaction with objects (*e.g.*, waiting for a car to pass by). Although different dogs do the same activity, their background varies from residential areas to a sand beach. The 10 target activities in the dataset include: "waiting for a car to pass by" (hereafter named *Car*), "drinking water" (*Drink*), "feeding" (*Feed*), "turning dog's head to the left" (*Left*), "turning dog's head to the right" (*Right*), "petting" (*Pet*), "playing with a ball"

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

(*Play*), "shaking dog's body by himself" (*Shake*), "sniffing" (*Sniff*), "walking" (*Walk*). The dataset is unbalanced according to the number of frames for each activity 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. Figure 2 illustrates a frame of each activity and the percentage of each activity in the dataset.

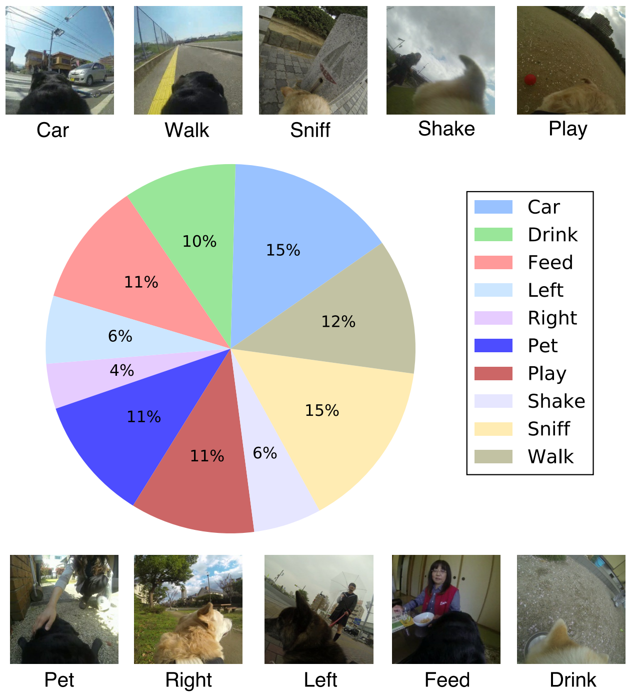

Fig. 2. Example frames of each activity and the percentage of frames for each activity available in DogCentric Activity dataset.

In order to perform experiments, we divided the dataset into training, validation, and test sets. We use the validation set to obtain the model configuration that best fits the training data, *i.e.*, the configuration with the highest accuracy, and the test set to assess the accuracy of the selected model in unseen data. Our method to divide the dataset is similar to the one proposed by Iwashita *et al.* [5] and consists of randomly selecting half of the videos of each activity as test set. In case where the number of videos (N) is odd number, we separate $\frac{(N+1)}{2}$ videos for 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 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.

B. Model Training

We implemented each part of our architecture using different toolkits: we implemented the neural networks using the *Caffe*[2] framework [19], whereas we used the Crammer and Singer [20] implementation of the SVM from *scikit-learn*[3][21]. We trained each model within our architecture using different strategies. Each CNN was trained using two different training strategies: fully-training the network from scratch, and fine-tuning a pre-trained network. We trained the SVM using each *softmax* output from train and validation set of AlexNet$_{\text{[Fine-tuned]}}$ and GoogLeNet$_{\text{[Fine-Tuned]}}$ networks as well as a fusion of both.

Fully-trained networks: The fully-trained networks learn all weights from scratch using the DogCentric Activity dataset [5]. For both networks *GoogLeNet* and *AlexNet* the training phase uses 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. All images have their pixels subtracted by the mean pixel of all training images. The activation of each convolution (including those within the "inception" modules in *GoogLeNet*) use a rectified linear unit (ReLU).

In *GoogLeNet*, each iteration uses a mini-batch with 128 images. Regarding the weight initialization in *GoogLeNet*, we employ the *Xavier* [22] algorithm that automatically determines the value of initialization based on the number of input neurons. To reduce the chances of overfitting, we apply dropout on the fully-connected layers with a probability of 80%. The learning rate is set to 3×10^{-4} and we drop it to 2×10^{-4} every epoch, stopping the training after 2.04k iterations (20 epochs).

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

Fine-tuned networks: Our fine-tuned networks (*AlexNet*$_{\textit{[Fine-tuned]}}$ and *GoogLeNet*$_{\textit{[Fine-tuned]}}$) are based on a fine-tuning process that uses weights from a network pre-trained on the 1.2-million-image ILSVRC 2012 ImageNet dataset [23]. More specifically, we use the already-trained models provided by the *Caffe* software package [19].

During training phase the network loads the pre-trained models and updates their weights via backpropagation for all images of the training set. The network updates all pre-trained layers with different learning rates, allowing the network to learn more the specific features of our dataset in the last layers than the basic features in the first layers.

The configuration of both networks are similar to the configuration of their fully-trained versions. However in *AlexNet*$_{\textit{[Fine-tuned]}}$ we decrease the global learning rate in 10% (we set to 10^{-5}) and in the last layer we increase the learning rate of the weights from 1 to 10 the learning rate of the bias from 2 to 20. In *GoogLeNet*$_{\textit{[Fine-tuned]}}$ we change the dropout to

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

70%, decrease the global learning rate to 3×10^{-5} and increase the learning rate of the weights from 1 to 10 the learning rate of the bias from 2 to 20. This configuration allows all layers to learn, but giving the final layer the capability to learn faster than the other layers.

SVM: We train the multi-class Support Vector Machine using the off-the-shelf implementation from `scikit-learn` toolbox. We use the linear kernel and default *scikit-learn* regularization parameter $C = 1$ with the square of the *hinge loss* as loss function.

IV. Results

In order to evaluate our approach, we compare the predictions of each model using the test set. We use the classification generated by each model for each class and the accuracy over all classes to check which models perform better the classification task. Table I shows the accuracy scores for each class individually (*Car*, *Drink*, *Feed*, *Left*, *Right*, *Pet*, *Play*, *Shake*, *Sniff*, *Walk*) and the global accuracy (*All*) that considers all classes at once. The *Fused* network consists in the application of late fusion in both $AlexNet_{[Fine\text{-}tuned]}$ and $GoogLeNet_{[Fine\text{-}tuned]}$ networks using SVM to generate the final scores. We combine $AlexNet_{[Fine\text{-}tuned]}$ and $GoogLeNet_{[Fine\text{-}tuned]}$ because they achieved a better accuracy when compared with the fully-trained models (*AlexNet* and *GoogLeNet*).

As Table I shows, our *Fused+SVM+PP* approach (see Figure 1) obtains the best results for most classes, achieving the global accuracy (*All*) of 75.6%. This approach is the result of a post-processing smoothing over the output of the *Fused+SVM* model. For *Fused* models, we train the SVM using the stacked *softmax* outputs from training and validation sets of the fined-tuned models ($AlexNet_{[Fine\text{-}tuned]}$ and $GoogLeNet_{[Fine\text{-}tuned]}$). We perform the testing using the stacked *softmax* outputs from test set of the same models. For *Fine-tuned* models, we train the SVM with the *softmax* output from training and validation sets of each model.

Classes containing small number of frames such as *Left* (6% of total frames), *Right* (4% of total frames) and *Shake* (6% of total frames) achieved low accuracy scores for all models. Another reasonable explanation is that the frames of these classes are too similar to frames of other classes. For example, in classes *Right* and *Left* there are some frames before the dog turn the head to the right that are very similar to frames of the class *Walk*.

Our networks trained from scratch (*AlexNet* and *GoogLeNet*) achieved the lowest accuracy scores for most of the results when compared to their fine-tuned versions. These networks seems to be somehow complementary since *AlexNet* achieved better results than *GoogLeNet* for 4 out of 9 (both networks achieved the same accuracy for *Shake*) and *GoogLeNet* achieved better results than *AlexNet* for 5 out of 9. These split results indicate that the networks may identify different features of each class. Fine tuning our networks improved results for most classes. Using SVM after the fine-tuned network in order to classify frames increased even more the accuracy scores.

Observing that our networks may identify different features in the same image, we decide to fuse both networks using a late fusion method. As using SVM usually achieves better results than the fine-tuned versions without SVM, we perform the late fusion stacking the *softmax* scores of both networks and classifying them using SVM. This process (*Fuse+SVM*) obtained better results for most classes (*Right* has the best accuracy using the fully-trained *AlexNet* network) and the best accuracy over all classes when compared with the fully-trained and fine-tuned versions. A reason that the *Right* class do not achieved a good result for the *Fused* network is that both fine-tuned versions achieved low scores for the class.

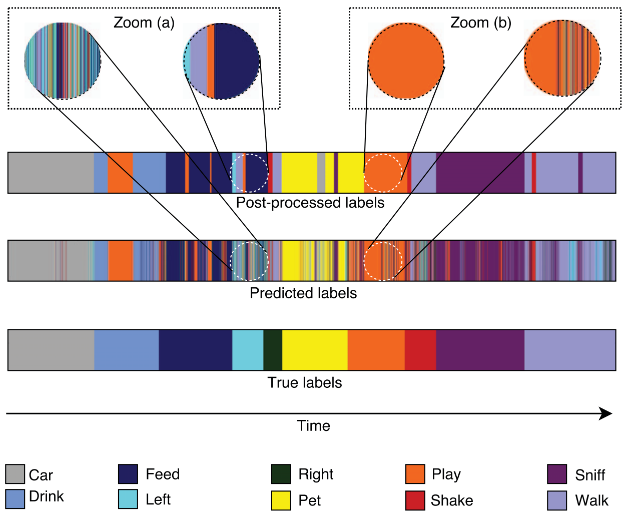

Fig. 3. Temporal representation of the frame sequence for all classes in DogCentric Activity dataset.

Since the process of identifying actions occurs frame by frame instead of the entire video, sometimes the misclassification of a small number of frames of an activity may occur. Observing the output of the *Fused* method for each class (Figure 3 – *Predicted labels* bar) we can see a lot of noisy predicted labels, *i.e.*, few labels that are classified by the network in the middle of a group of frames with another label. As an activity would not occur in a single frame or in a small number of frames, we believe that a frame in the middle of a sequence of 10 frames that contains a different class suggests that the frame was misclassified. Thus, we apply a post processing on the predicted labels in order to smooth labels removing the misclassified ones. Figure 3 illustrates the sequence of frames in a timeline with their corresponding true label (lowest bar) predicted label (middle bar) and predicted label after the smoothing process (highest bar). In each bar, classes are represented by colored vertical lines in a temporal sequence that are described in the legend. In *Zoom (a)* and *Zoom (b)* we can see in the same part of the frame sequence how the post processing eliminates noisy labels. Observing the results in Table I we can see that after the post-processing the accuracy increased for most classes and for the overall

TABLE I

Per-activity accuracy in the DogCentric Activity dataset for all baselines and fusion methods.

Network	**Car**	**Drink**	**Feed**	**Left**	**Right**	**Pet**	**Play**	**Shake**	**Sniff**	**Walk**	**All**
AlexNet	0.289	0.212	0.070	0.070	**0.044**	0.179	0.004	0.000	0.221	0.021	0.267
GoogLeNet	0.401	0.170	0.218	0.002	0.042	0.143	0.142	0.000	0.239	0.033	0.320
AlexNet$_{\text{[Fine-tuned]}}$	0.858	0.404	0.315	0.012	0.000	0.395	0.294	0.161	0.453	0.224	0.526
AlexNet$_{\text{[Fine-tuned]}}$ + SVM	0.876	0.497	0.419	0.052	0.000	0.539	0.327	**0.188**	0.565	0.415	0.618
GoogLeNet$_{\text{[Fine-tuned]}}$	0.756	0.496	0.462	0.144	0.022	0.542	0.364	0.053	0.507	0.207	0.571
GoogLeNet$_{\text{[Fine-tuned]}}$ + SVM	0.799	0.524	0.486	0.202	0.002	0.573	0.395	0.097	0.596	0.379	0.634
Fused + SVM	0.895	0.541	0.517	**0.254**	0.032	0.634	0.412	0.179	0.629	0.474	0.682
Fused + SVM + PP	**0.914**	**0.545**	**0.636**	0.114	0.000	**0.653**	**0.436**	0.099	**0.911**	**0.625**	**0.756**

accuracy. *Zoom (b)* in Figure 3 illustrates when a few noisy incorrectly classified labels are replaced by the correct class (*Play*). On the other hand, the smoothing process eliminates few correctly predicted classes when they are in the middle of other classes. For example, *Zoom (a)* in Figure 3 shows that few frames are correctly classified as *Left* activity. When applying the post-processing, the smoothing process replaces these few correctly classified frames to other activities such as *Drink*, *Play* or *Feed*. As observed before, classes with the lowest number of frames tend to be misclassified, and thus, the post processing fails when smoothing these classes. For example, the class *Right* contains 4% of the frames in the dataset and our *Fused* network identified a small portion of the correct frames. This small portion of the correct frames is sparse and the post-processing replaces all these correctly predicted labels by incorrect but most frequent labels.

Using the output of the fused networks (*Fused+SVM*) we perform an analysis to see how predicted classes are classified in relation to the true classes. We decide to use the output of the *Fused+SVM* networks instead of the *Fused+SVM+PP* because it is more interesting to observe the classification predicted by the network instead of its smoothed results. In order to perform this analysis we generate a normalized confusion matrix, depicted in Figure 4 that shows the effect of the fused networks, 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. The confusion matrix shows normalized values, predicted values are divided by the total number of true values for each cell. Observing the confusion matrix we can approximate the precision and recall value of each class.

Observing Figure 4, we can see that the network predicts few frames to the class *Right*. Other classes predicted as *Right* include *Shake* and *Walk*, indicating that some frames of these classes may be similar to the frames of the class *Right*. Usually the network classify wrongly the true class *Right* by the class *Walk*, indicating that features of these classes in particular may be very similar. The class *Car* is very dissimilar to the other classes showing a dark shade of blue in the matrix, having almost all frames correctly classified, which indicates a high precision score. The high classification score of the class *Car* is reasonable since its scenario is usually different from the scenario of the other classes. On the other hand, some

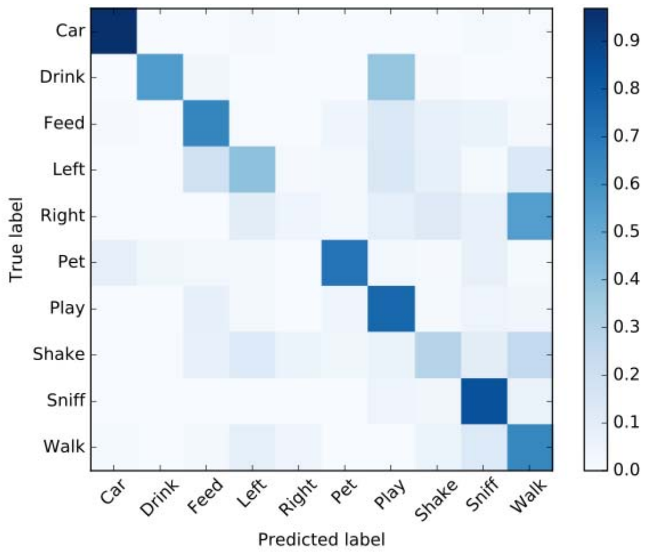

Fig. 4. Normalized confusion matrix for our *Fused+SVM* network.

frames of the true class *Pet* are misclassified as *Car*, which decreases its recall. Other classes, such as *Shake* and *Play* are misclassified by other classes such as *Feed*, *Left* and *Right*. Our network sometimes wrongly classify the class *Drink* as the class *Play*. This error may be reasonable because in many frames from the class *Play* the dog is playing with the ball and the camera is recording the floor, as it occurs when the dog is drinking water.

Since classification accuracy takes into account only the proportion of correct results that a classifier achieves, it is not suitable for unbalanced datasets as it may be biased towards classes with larger number of examples. By analyzing the DogCentric Activity dataset in Figure 2, we note that it is indeed unbalanced, *i.e.*, classes are not equally distributed over frames. For dealing with the unbalanced nature of the dataset, we measure the performance of the fusion method based on Precision (P), Recall (R) and F-Measure (F) scores. Calculating such scores for all classes, our network achieves the highest precision scores for *Drink*, *Car* and *Pet*, the highest recall scores for *Car*, *Sniff* and *Play*, and the highest F-measure are obtained by *Car*, *Pet* and *Sniff* classes. Table II shows Precision, Recall and F-measure scores for all classes and the overall (*All*) scores. As illustrated in Figure 4, the *Right* class has the lowest precision, recall and F-measure scores.

TABLE II
PRECISION, RECALL AND F-MEASURE FOR EACH ACTIVITY OF THE DOGCENTRIC ACTIVITY DATASET USING OUR FUSED NETWORK (*Fused+SVM*).

Activity	Precision	Recall	F-measure
Car	0.92	**0.97**	**0.94**
Drink	**0.93**	0.57	0.70
Feed	0.72	0.65	0.68
Left	0.41	0.40	0.41
Right	0.11	0.04	0.06
Pet	0.85	0.72	0.78
Play	0.47	0.77	0.58
Shake	0.32	0.29	0.30
Sniff	0.71	0.84	0.77
Walk	0.64	0.64	0.64
All	0.69	0.68	0.69

We calculate the values of Precision, Recall, F-measure, and Accuracy for all networks developed in our work and compare them with the existing work. In order to compare our results with the current state-of-the-art in the DogCentric Activity dataset, we add in Table III the results achieved by Iwashita *et al.* [5] and by Ryoo *et al.* [9]. We compare our results with 4 approaches performed by Iwashita *et al.* [5] namely *Linear kernel*, *RBF kernel*, *Histogram intersection* and *Multi-channel*, and with 3 approaches performed by Ryoo *et al.* [9] namely *PoT+STIP+Cuboid*, *PoT+ITF+STIP+Cuboid* and *PoT+ITF*, where *RBF Kernel* denotes the Radial Basis Function Kernel, *PoT* stands for Pooled Times series, *STIP* means Space Time Interest Point, and *ITF* stands for Improved Trajectory Features. We only compare our networks in terms of Accuracy with the state-of-the-art because the other authors do not provide Precision, Recall and F-measure results. Our fused model without post processing (*Fused+SVM*) achieves higher accuracy (63.4%) than models proposed by Iwashita *et al.* [5]. Their highest score is obtained by a multi-channel model with a global accuracy of 60.5%. Approaches proposed by Ryoo *et al.* [9] overcome our fused model, with an accuracy higher than 70%. On the other hand, our fused model with post-processing (*Fused+SVM+PP*) overcomes the current state-of-the-art approach that uses a combination of two feature representations, namely Pooled Time series and Improved Trajectory Features and achieves an accuracy of 74.5% [9].

V. APPLICATION

In this work we develop an application that translates recognized activities into audible feedback in order to help visually impaired people. This application intends to enhance the amount of information a guide dog can provide as feedback, helping the construction of mental maps in visually impaired people. Our application receives a sequence of frames from a camera and translates the identified activities into sound feedback. Figure 5 illustrates the architecture of our application, where *AlexNet* and *GoogLeNet* are our trained networks that are fused in *SVM*. *Audio generation* receives the predicted class for the current frame and generates its corresponding phrase in output. The user receives a new sound (a new phrase) only when the *SVM* yields a different predicted class, avoiding the continuous reproduction of the same phrase.

TABLE III
PRECISION, RECALL, F-MEASURE, AND ACCURACY FOR ALL BASELINES, FUSION METHODS AND THE CURRENT STATE-OF-THE-ART APPROACH FOR THE DOGCENTRIC ACTIVITY DATASET.

Approach	Precision	Recall	F-measure	Accuracy
Linear kernel [5]	-	-	-	0.526
RBF kernel [5]	-	-	-	0.542
Histogram intersection [5]	-	-	-	0.573
Multi-channel [5]	-	-	-	0.605
PoT+STIP+Cuboid [9]	-	-	-	0.731
PoT+ITF+STIP+Cuboid [9]	-	-	-	0.741
PoT+ITF [9]	-	-	-	0.745
AlexNet	0.260	0.270	0.265	0.267
GoogLeNet	0.340	0.320	0.330	0.320
AlexNet$_{\text{[Fine-tuned]}}$	0.550	0.530	0.540	0.526
AlexNet$_{\text{[Fine-tuned]}}$ + SVM	0.620	0.620	0.620	0.618
GoogLeNet$_{\text{[Fine-tuned]}}$	0.600	0.570	0.585	0.571
GoogLeNet$_{\text{[Fine-tuned]}}$ + SVM	0.640	0.630	0.635	0.634
Fused+SVM	0.690	0.680	0.685	0.682
Fused+SVM+PP	**0.740**	**0.760**	**0.750**	**0.756**

We test our application using the DogCentric Activity dataset [5] since it contains video data from dog-mounted camera. As some activities of DogCentric Activity dataset may not interest the user, such as "The dog is looking left" or "The dog is shaking", we decide to create phrases for 5 out of the 10 classes. We record audio feedback for each class of interest (*Car*, *Drink Feed*, *Pet* and *Sniff*), and thus, the user receives an audio feedback every time a different class is predicted in the video. Below we set classes in bold and their corresponding sentences we created to describe the activity.

- **Car**: *There is a car on your way.*
- **Drink**: *The dog is drinking.*
- **Feed**: *The dog is being fed.*
- **Pet**: *The dog is being petted.*
- **Sniff**: *The dog is now sniffing something.*

Figure 6 depicts two screens of our application when running to detect the classes *Pet* (a) and *Car*. In order to

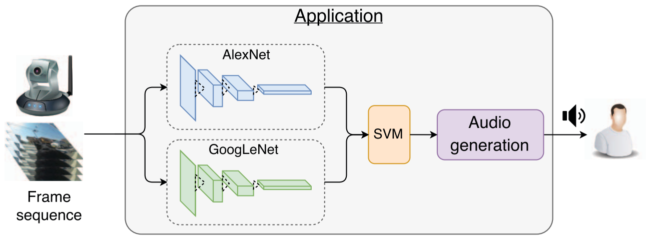

Fig. 5. Architecture of the application that translated activities identified in frame sequences into audible feedback.

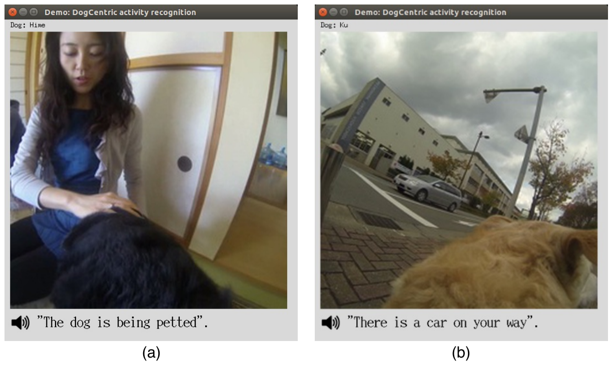
Demo: DogCentric activity recognition
Dog: Hime
"The dog is being petted".
(a)
Demo: DogCentric activity recognition
Dog: Ku
"There is a car on your way".
(b)

Fig. 6. Demo of the application for *Pet* and *Car* activities.

provide a visual feedback we print the sound phrase on the screen while yielding the feedback. Thus, when the application detect someone petting the dog, it yields the phrase "The dog is being petted" (a) and when a car appears on the video, the application yields the phrase "There is a car on your way".

VI. Related Work

The increasing availability of wearable devices such as cameras, Google Glass[4], Microsoft SenseCam[5], *etc.* facilitates the low-cost acquisition of rich activity data. In recent years, this increase of available egocentric data have attracted a lot of attention in the computer vision community. The first-person point-of-view is very popular in the study of daily activities [9], [24]. For instance, Hsieh *et al.* [24] performs activity recognition using multiple representations such as objects manipulated by a user, the motion pattern of the hands, background contexts, *etc.*. In order to perform activity recognition they use the ADL+ dataset that contains video and wrist-worn accelerometer data of people performing activities of daily living. A model mixing object features representing *what* is in the scene, scene features describing *where* the subject is, and motion features indicating *how* the subjects interact with the objects is developed.

Ma *et al.* [25] propose a method that uses twin-stream networks, one to capture the appearance of the objects such as hand regions and objects attributes, and the other to capture motion such as local hand movements and global head motion using stacked optical flow fields as input. The appearance-based stream is trained using the hand segmentation, focusing on certain regions of the image near the hand, and with object images cropped based on hand location to identify objects of manipulation. The stream intends to encode features such as hand-object configurations and object attributes. The motion-based stream is trained to differentiate between action labels using as input a stack of optical-flow motion fields. A late fusion method [17] joins both networks. Their experiments use GTEA [26] and Gaze [27] datasets.

Other work focus on recognizing activities not only using wearable cameras attached to a person, but also to an animals, also called first-person animal activity recognition. Iwashita *et al.* [5] perform the activity recognition extracting global and local features from the DogCentric Activity dataset. More specifically, they extract global features from dense optical flow [28] and binary pattern [29], and local features from a cuboid feature detector [30] and STIP detector [31]. Using these hand-crafted features they achieve the highest classification accuracy of 60.5%.

Ryoo *et al.* [9] develop a new feature representation named pooled time series (PoT). PoT intends to capture motion information in first-person videos applying time series pooling of feature descriptors, detecting short-term/long-term changes in each descriptor element. Their approach first extracts appearance/motion descriptors from a sequence of frames and represents them as a set of time series. A time series represents how each element of a descriptor is changing over time. Temporal pooling is applied on each time series in order to summarize the information represented in a sequence of video frames. The system applies multiple pooling operations and the results are concatenated to form the final feature vector that better represents the video. Finally, activity recognition is performed by training and testing a Support Vector Machine (SVM) classifier using these vectors. This approach achieves 74% of accuracy when combined with INRIA's improved trajectory feature (ITF) [32] using the DogCentric Activity dataset [5].

VII. Conclusions and Future Work

In this work, we developed a novel architecture for dog-centric activity recognition based on two convolutional neural networks and a late fusion method. The pipeline of the architecture includes training two deep convolutional neural networks in parallel to extract features from images and by a late fusion process classify unseen images. Using frame sequences from the DogCentric Activity dataset, we perform experiments showing that the convolutional networks can indeed learn high-level relevant features for the activity recognition task. Experiments show that our approach that employs the late fusion of two CNNs achieves better results when compared with the current state-of-the-art.

In addition to the developed models and fusion methods, we suggest an application that could support visually impaired people in their orientation and mobility skills. Our approach shows that it is possible to use the CNNs classification in applications to support people in their tasks.

As future work, we intend to augment our dataset using hand-crafted features such as histogram of optical flow (HOF), histogram of gradients (HOG), motion boundary history (MBH) and dense trajectories to extract more features from unbalanced data. Along with the current images, all these features feed the deep convolutional neural networks intending to reduce errors due to the small size of our dataset. We plan to employ deep learning architectures that take into account the temporal dimension, such as Long-Short Term Memory networks (LSTM) [33] and 3D CNNs [14] considering that they are capable of encoding temporal features.

[4]https://www.google.com/glass/start/

[5]http://research.microsoft.com/en-us/um/cambridge/projects/sensecam/

Future work includes the creation of a more realistic dataset where each video may contain a sequence of activities instead of a single activity. Finally, we intend to test our application with visually impaired users and relate their feedback about the effectiveness of our approach.

Acknowledgement

This paper was achieved in cooperation with HP Brasil Indústria e Comércio de Equipamentos Eletrônicos LTDA. using incentives of Brazilian Informatics Law (Law nº 8.2.48 of 1991). The authors would also like to thank FAPERGS, CAPES, and CNPq for funding this research.

References

[1] O. Lahav and D. Mioduser, “Blind persons’ acquisition of spatial cognitive mapping and orientation skills supported by virtual environment,” *International Journal on Disability and Human Development*, vol. 4, no. 3, pp. 231–238, 2005.

[2] M. A. Heller, “Tactile picture perception in sighted and blind people,” *Behavioural Brain Research*, vol. 135, no. 12, pp. 65 – 68, 2002. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166432802001560

[3] N. A. Giudice and G. E. Legge, “Blind navigation and the role of technology,” *Engineering handbook of smart technology for aging, disability, and independence*, pp. 479–500, 2008.

[4] S. Tachi and K. Komoriya, “Guide dog robot,” *Autonomous Mobile Robots: Control, Planning, and Architecture*, pp. 360–367, 1984.

[5] Y. Iwashita, A. Takamine, R. Kurazume, and M. S. Ryoo, “First-person animal activity recognition from egocentric videos,” in *International Conference on Pattern Recognition (ICPR)*, Stockholm, Sweden, August 2014.

[6] R. Poppe, “A survey on vision-based human action recognition,” *Image and Vision Computing*, vol. 28, no. 6, pp. 976–990, Jun. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.imavis.2009.11.014

[7] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, “Handwritten digit recognition with a back-propagation network,” in *Advances in Neural Information Processing Systems 2*, D. S. Touretzky, Ed. Morgan-Kaufmann, 1990, pp. 396–404. [Online]. Available: http://papers.nips.cc/paper/293-handwritten-digit-recognition-with-a-back-propagation-network.pdf

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[9] M. S. Ryoo, B. Rothrock, and L. Matthies, “Pooled motion features for first-person videos,” in *Proceedings of the 2015 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’ 15)*, June 2015, pp. 896–904.

[10] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[11] P. Sondhi, “Feature construction methods: a survey,” *sifaka.cs.uiuc.edu*, vol. 69, pp. 70–71, 2009.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[13] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *Computing Research Repository (CoRR)*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556

[14] 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, 2013.

[15] 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 the 2014 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’ 14)*. Washington, DC, USA: IEEE Computer Society, 2014, pp. 1725–1732. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2014.223

[16] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3D convolutional networks,” in *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2015, pp. 4489–4497.

[17] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Advances in Neural Information Processing Systems*, 2014, pp. 568–576.

[18] 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 the 2015 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’ 15)*, June 2015.

[19] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the 22nd ACM international conference on Multimedia*. ACM, 2014, pp. 675–678.

[20] 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.

[21] 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, pp. 2825–2830, 2011.

[22] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks.” in *In Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS10). Society for Artificial Intelligence and Statistics*, vol. 9, 2010, pp. 249–256.

[23] 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 (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[24] P.-J. Hsieh, Y.-L. Lin, Y.-H. Chen, and W. Hsu, “Egocentric activity recognition by leveraging multiple mid-level representations,” in *Multimedia and Expo (ICME), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1–6.

[25] M. Ma, H. Fan, and K. Kitani M, “Going deeper into first-person activity recognition,” in *Proceedings of the 2016 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’ 16)*, 2016.

[26] A. Fathi, X. Ren, and J. M. Rehg, “Learning to recognize objects in egocentric activities,” in *Proceedings of the 2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’ 11)*. IEEE, 2011, pp. 3281–3288.

[27] A. Fathi, Y. Li, and J. M. Rehg, “Learning to recognize daily actions using gaze,” in *European Conference on Computer Vision*. Springer, 2012, pp. 314–327.

[28] M. S. Ryoo and L. Matthies, “First-person activity recognition: What are they doing to me?” in *Proceedings of the 2013 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’ 13)*, 2013, pp. 2730–2737.

[29] T. Ojala, M. Pietikainen, and T. Maenpaa, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 7, pp. 971–987, 2002.

[30] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie, “Behavior recognition via sparse spatio-temporal features,” in *2005 IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*. IEEE, 2005, pp. 65–72.

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

[32] H. Wang and C. Schmid, “Action recognition with improved trajectories,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 3551–3558.

[33] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.