Master in Artificial Intelligence

UNIVERSITAT POLITÈCNICA DE CATALUNYA (UPC) – BarcelonaTech
UNIVERSITAT DE BARCELONA (UB)
UNIVERSITAT ROVIRA i VIRGILI (URV)

Master Thesis

First-person activity recognition: how to generalize to unseen users?

Director: Mariella Dimiccoli
Co-director: Petia Radeva
FACULTAT DE MATEMÀTIQUES I INFORMÀTICA (UB)

Abstract

Recent advances in wearable technology, accompanied by the decreasing cost of data storage and increase of data availability have made possible to take pictures everywhere at every time. Wearable cameras are nowadays among the most popular wearable devices. Besides leisure, wearable cameras are attracting a lot of attention for the improvement of working conditions, productivity and safety monitoring. Since the collected data can be potentially used for memory training and extracting lifestyle patterns useful to prevent noncommunicable diseases as obesity, they are being investigated in the context of Preventive Medicine. Most of these applications require to automatically recognize the ability performed by the user. This work aims to make a step forwards towards activity recognition from photo-streams captured by a wearable camera by developing a method that allows to label new images with minial effort from the user and generalize well for unseen users.

Contents

1 Introduction

Devices and technologies have taken huge part of our lives and will be used even more in the future. We try to automate and improve as many things as possible, and exploit them to improve our quality of life. One thing that can help in this are lifelogging camera devices, so the growing interest for them is not a surprise. With them we can easily record everything through the day from a first-person perspective. A lifelogging wearable camera typically makes a few pictures per minute without requiring any action from the user, so it can generate huge amount of data.

Analysing automatically this data and being able to automatically understand what are the activities being performed in the pictures, has a lot of useful applications. The Activities of Daily Living (IADL) include, but are not limited to the activities performed on a daily basis for living at home or in a community [7]. The monitoring of IADL can be used to detect frailty in elderly people [8], or to understand and improve our habits, since it can give us insights about what to change. In addition, observing the activities of the user over a long period of time has a lot of applications in Preventive Medicine, because it would allow to estimate the habits of the user that are associate to many noncommunicable diseases [9].

An useful approach to the problem can be activity recognition through image classification. The goal of activity recognition is to recognize common human activities, performed on daily basis. The problem is quite challenging firstly because there is huge inter- and intra-class variability in human activities performed by different individuals. What makes the recognition a much more difficult task, is the case of images, captured by a wearable camera, because the images are taken from first-person (or ego-centric) point of view. Compared to images, taken by a third-view camera, in egocentric images the main actor is not visible and what he is doing has to be inferred by the objects he is manipulating, the persons he is interacting with etc.

Additionally, due to the free motion of the camera objects often appear blurred or partially occluded, and since they are being manipulated, their appearance may undergo huge variations. Such pictures can be seen in Table 1.1 and Table 1.2.

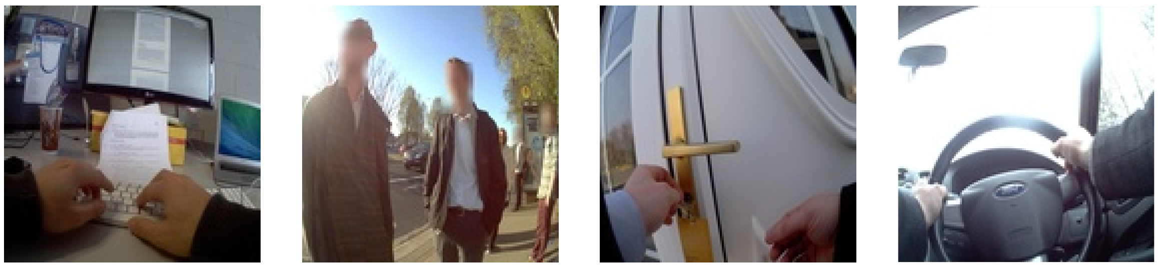


Table 1.1: First-person (or ego-centric) images, taken with wearable camera

Table 1.2: Blurred and occluded images, taken with wearable camera

Activity recognition can be done both on video or photo streams. When working with video (35fps), a lot of contextual information is also available - for example spatio and temporal features and optical flow sequences can be extracted. If this information is used in the proper way, it can help a lot and improve the results.

Activity classification from egocentric photo streams is even more difficult problem than from video, since they provide less contextual action information. We chose to work with the second kind of sequences and focus on cameras with low temporal resolution (2 fpm), because they allow to capture the full day and therefore are suited to collect data over long period of time. However, this imposes additional challenges to the activity recognition problem with respect to conventional videos - the frequent sudden changes in the field of view for example. An example for this is showed in Table 1.3. Motion cannot be used to enhance activity recognition since optical flow cannot be reliably estimated when temporally adjacent frames undergo abrupt changes. Observations are very sparse so that there is much less contextual information to infer the activities of the wearer.

Another important aspect of lifelogging cameras is that, since they are worn all the day during a long period of time, they are typically worn on the chest (Fig. 1.1) and not on the head for social reasons. Consequently, head movement and attention cannot be used as additional features for activity recognition.

Table 1.3: Sudden changes in the field of view in images, taken with wearable camera

Table 1.4: Different types of cameras: the one on the left is worn on the head and the one on the right is worn on the chest. Images taken from

1.1 Objectives

Deep learning approaches for egocentric activity recognition often achieve very high accuracy predictions [11,16] but remains unclear whether these performances hold also for *unseen users*. By unseen users we refer to users whose images have not been feed into the training set. Since the performance of deep learning methods strongly relies on the employed training dataset, more than on more generalization capability, the purpose of this work is to create a system for automatic activity recognition from egocentric photo-streams, that could potentially be used in real applications and by a lot of people with different lifestyles. We compare and analyze different algorithms for automatic activity recognition and we propose a general method for activity recognition from egocentric images, that does not require a cumbersome annotation effort to generalise to unseen users.

Using wearable cameras leads to a lot of data - if the camera takes for example two pictures per minute, it will create almost 2000 pictures every day. State of the art algorithms for image classification and activity recognition are based on deep learning approaches that are supervised and require a huge annotation effort. The problem is that this should be done manually and doing it for thousands of images is a very time consuming task. Also often there are no good tools for annotating the pictures, so the labeling becomes even harder. Some example images and the activities on them can be seen in Table 1.1.1.

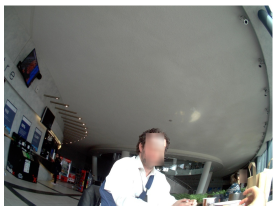
Mobile	Driving	Shopping	Having drinks/meal
			
Public transport	Walking outdoors	Cleaning	Talking

Table 1.1.1: Sample images, labeled with activities

The task of supervised algorithms is to learn how to map input data to output data and their purpose is to assign one of the already given labels to unseen input. So they require previously defined categories and each of the images should be labeled with one of these categories. One of the advantages of this is that we can easily measure if and how well the algorithm is performing. What is not so good is that we need to predefine all possible categories, which is not always easy, especially in the case of ego-centric activity recognition since we cannot predict all the activities a person can perform during the day.

We can think of some activities that most of the people do - like socializing, eating, working and so on, but how many activities should we have and should they be more general or specified? Also every person has a different lifestyle and every person does different activities through the day - one can practise some kind of sport, while other can play a musical instrument, one can spend time with his/her dog, but other may not have a pet. Finding the right number of categories and the right categories and then labeling all the pictures can sometimes really be a problem and a difficult task, but it is beneficial to the final results.

That being said, it is logical to try with the opposite - unsupervised learning. In this approach we do not annotate the input data - the model finds features from the input on its own and divides the data into several groups. The results should fit better the structure of the data. There are also some unsupervised algorithms which do not require exact number of clusters and find the optimal number on their own. It is a really big advantage that there is no need to annotate all the data. With the wearable cameras we can collect millions of images almost without effort, but they are of no use for the supervised algorithms it they are not annotated. Using unsupervised approach allows us to benefit from all the available data without losing time and effort to prepare it for training. This advantage however is also a drawback - it is difficult to measure the quality of the clusters and how well is the algorithm performing.

Most of the models need to be trained with data which is very similar to the test data so that they can perform well. Because of this the data is usually split in two subsets (or three in the case of deep learning approaches) - one for training and one for test (and one for validation in the case of DL approaches). But the approach, proposed by this work should work well with new and unseen data without the need the model to be trained again or fine-tuned. Here comes another advantage of the unsupervised models for performing activity recognition. If our model is trained on images from several people and then we want to recognise the activities in the images of another person - there is a possibility that his/her activities are different from the ones of the others. So an unsupervised approach can find that for this person the clusters should be a bit different and should be able to split the data, based on his/her activities and not on predefined labels which may not fit his/her lifestyle and activities.

1.2 Related Work

Understanding human activities from videos has been a well-studied topic in computer vision. As the field of egocentric vision is quite new and challenging, there has been growing interest in the last several years in recognizing activities from egocentric data and it has became an active area of research. However, most works have focused on activity recognition from videos [17], while activity recognition from photo-streams has been little explored.

1.2.1 Activity recognition from egocentric videos

Fathi et al. [1] present a hierarchical method to analyze daily activities using video from an egocentric camera. They use joint modeling of activities, actions, and objects and introduce a novel representation of actions based on object-hand interactions. Their dataset contained 7 kinds of daily activities, performed by 4 people and 16 kinds of objects used in these activities.

Pirsiavash and Ramanan [2] present a new dataset of 1 million frames, annotated with activities, object tracks, hand positions, and interaction events. They used 18 different actions and 42 different objects and had videos of dozens of people performing unscripted, everyday activities. Their model involves long-scale temporal structure and complex object interactions. Their representations include temporal pyramids and composite object models and show that the objects with which is interacted are most useful for the activity recognition.

Another multi-task clustering framework for activity analysis of daily living is suggested by Yan et al. [4]. They sue the fact that everyday activities of multiple individuals are related, since typically people perform the same actions in similar environments. For each person, a set of samples is available and they should be segmented corresponding to the user into parts and the resulting partitions should be consistent with each other, because people perform about the same activities. Two clustering approaches are used - Earth Mover's Distance Multi-Task Clustering and Convex Multi-Task Clustering. They used two datasets. The first consists of over two hours of data, showing five common activities in an office environment, performed by five subjects. The second one contains videos recorded by 20 different users, performing 18 non-scripted daily activities in the house, like brushing teeth, washing dishes, or making tea. It also has annotations about the presence of 42 relevant objects and about temporal segmentation.

1.2.2 Activity recognition from egocentric images

To the best of our knowledge, so far there have been only two attempt to recognize egocentric activities from photo-streams.

D. Castro et al. presented a method to analyze images taken from a passive egocentric wearable camera along with the contextual information, such as time and day of week. They used Convolutional Neural Network (CNN) with a classification method they introduced, called a late fusion ensemble, which incorporates relevant contextual information and increases the classification accuracy. The proposed approach was tested on a dataset of more than 40 000 images over a 6 month period with 19 activity classes and an overall accuracy of 83.07% was achieved. However, the dataset was acquired only by a single person and since the user had a routinary life, these performances are not surprising.

Later, Cartas et al. [11] tried to generalize this framework to multiple users. What is novel in it is that instead of using time information as contextual information, since this does not make sense when the data belongs to multiple users, they used the features of the fully connected layer. They proved that the classification accuracy of the CNNl argely improves when its output is combined, through a random decision forest, with contextual information from a fully connected layer. The used dataset consists of 18,674 images acquired by three people, and an overall accuracy of 86% is achieved in the recognition on 21 classes.

1.2.3 Active learning

Supervised approaches assume a fixed set of labeled data, which is not necessarily true in real-world applications. For example, in the case of activity recognition we assume a fixed number of activities, but a unseen user may perform activities not included in this set. So we may want to labels his data to improve the results of the supervised model. Getting labeled data is usually expensive and time consuming. Active learning aims at achieving the best learning result with a limited labeled data set, i.e., choosing the most appropriate unlabeled data to get labeled.

Dasgupta and Hsu [18] presented an active learning appoach that exploits cluster structure in data. Their method starts with a hierarchical clustering of the unlabeled points and discover any informative pruning of the cluster tree. By doing this they get fairly pure clusters at the leaves of the tree, so it is enough for the user to label only one item for each leave in order to achieve estimate of the labels of the entire dataset. This is extremely useful, as it reduces a lot the effort and time needed for labeling. With this approach they labeled 10000 training examples with only 400 labels with small error.

One very interesting work about image clustering was provided by Yang, Parikh and Batra [6]. They propose a recurrent framework for Joint Unsupervised Learning of deep representations and image clusters. In their framerwork, successive operations in a clustering algorithm are expressed as steps in a recurrent process, stacked on top of representations output by a Convolutional Neural Network (CNN). During training, image clusters and representations are updated jointly: image clustering in the forward pass, and representation learning in the backward pass. They combine two processes into a single model using weighted triplet loss function. Their approach is focused on image clustering of very simple images, but their ideas can be very useful to generate high purity clusters in an active learning framework. The experiments on different databases of images showed that this approach outperforms unsupervised state-of-the-art algorithms on image clustering and it finds better representations of the images which generalize well when transferred to other tasks.

2 Methodology

Our first and main objective was to find an approach able to generalize well and gives satisfactory results even for such unseen users. To achieve this goal, we have tried to use and combine different methodologies - supervised and unsupervised that are detailed in this section.

Our second goal was to define a method to easily get labels for an unseen user so that these could be used for re-train the system and improve the performances of the algorithm.

We used several different subsets of the NTCIR egocentric dataset egocentric dataset [10]. The dataset contains images from three lifeloggers who collected them by using a wearable camera that takes a picture every 30 seconds for a period of about one month each.

First we split all the data we have in train, test and validation sets. Then we split the images by users, who have taken them. We trained the models with the images for two of the users and then tested with the images from the third user, using a cross validation strategy. The reason we did this is because we wanted to see how much the performances drop when the test images belong to an unseen user - a user whose images are seen for the first time and not similar images have been used for training.

2.1 Supervised CNN-based methodologies

In recent researches regarding images, highest results are obtained using approaches based on deep learning and convolutional neural networks [19]. These networks assume that the inputs are images and are specially designed to be much more efficient than normal convolutional networks for this kind of input. They consist of layers and receptive fields, which process small regions of the image (Fig 2.1.1). There are several types of layers - Convolutional Layer, Pooling Layer, and Fully-Connected Layer. The layers can be stacked in different combinations in order to obtain the best architecture for the specific classification task. In our case we used GoogLeNet (Fig. 2.1.2), because it is suitable for the images we have and is deep enough to find meaningful features.

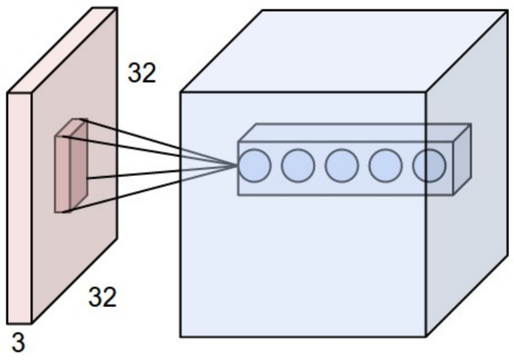


Fig. 2.1.1 Input volume (image) in red and volume of neurons in blue
http://cs231n.github.io/assets/cnn/depthcol.jpeg

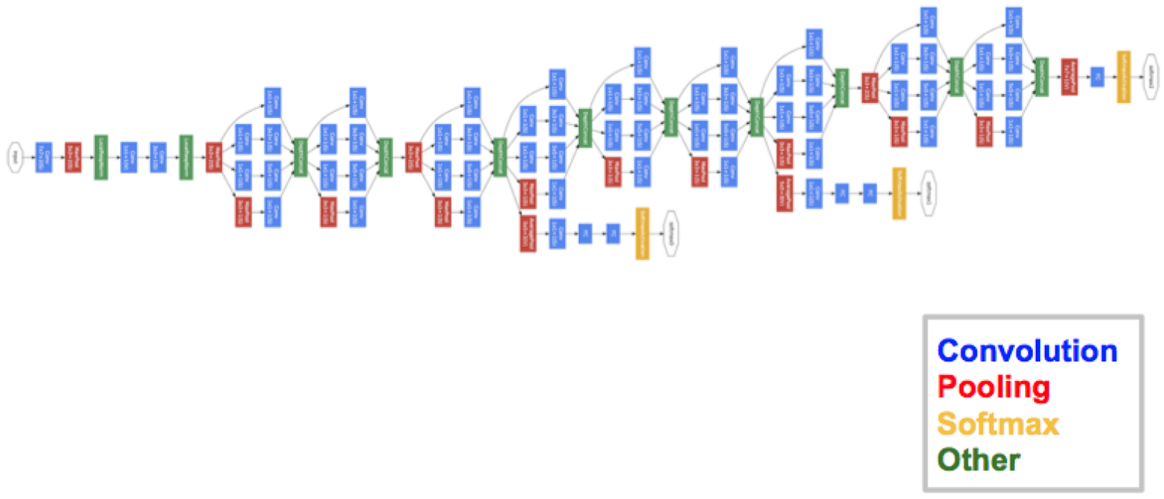


Fig. 2.1.2 Architecture of GoogLeNet
http://www.csc.kth.se/~roelof/deepdream/googlenet2.png

We limited our analysis to state of the art supervised approaches based on CNN. It is well known that large CNN networks may allow for more expressive power, however it is also prone to over fitting due to the large number of parameters. Additionally, uniform increased network size increases computational resources. Since the images captured by a wearable camera are real world images, and recognizing activities require lot of expressive power, we used GoogleLeNet architecture. This CNN architecture is characterized by the presence of an inception module that by approximating a sparse structure with spatially repeated dense components and using dimension reduction, keep the computational complexity bounded.

We finetuned a GoogLeNet CNN and used it as a fixed feature extractor. We tried different approaches. One was to use ensemble of classifiers - CNN + Random Forest. The other one was to apply unsupervised clustering algorithm on the extracted features from the network. We tried applying kMeans and Spectral Clustering.

2.2 Semi-unsupervised methodologies

We also tried to use unsupervised clustering algorithms and active labeling in order to improve the results. We first split the new, unseen images in big clusters using agglomerative clustering. We actually cluster not the images, but the features obtained for each image from the CNN network. After this, using some metrcis and thresholds we obtain the clusters which contain images from only one class and give them to the user to put labels (he/she has to put only one label for each cluster). After this we finetune the network with the images labeled by the user. We can then repeat this procedure until we get pure clusters. When it is not possible to form good clusters anymore, the user can label all the remaining images (the remaning images should be a small subset of all unseen images).

3 Experiments

3.1 Dataset

The proposed method was tested on a subset of the NTCIR-12 egocentric dataset [10]. This dataset consists of data from three lifeloggers who collected pictures using a Looxcie wearable camera for a period of about one month each. The data consists of a large collection of images - taken with a frame rate of two pictures per minute, which makes 88 124 images in total, summing up to 18.18 GB.

There are some XML descriptions of the data like semantic locations and the physical activities, but at a granularity of one minute. The problem is that we need a label for every picture and we need a predefined common set of categories for all the users, so that we can calculate how accurate is our approach. We also need the labeled pictures to be equally distributed between the three users, so that we have equal number of images for each of the lifeloggers.

Luckily, a large number of pictures (about 18,000) from the NTCIR dataset were annotated for the work of Marin et al. - Recognizing Activities of Daily Living from Egocentric Images [5]. The problem was that they were not enough and they were mostly from the first user (about 11, 000) and much less for the second and third. So we needed more labeled images for all of the users and had to do this manually.

The labeling work was performed using a special annotation tool [11], developed for the work of Juan Marin. The annotation tool [11] was specially designed for labeling big datasets of images and it made things much easier - labeling a group of consecutive images, related to the same category is done with one click, a nice preview with what is already annotated is provided and so on. The process of labeling can be seen in Figure 3.1.1 and Figure 3.1.2. However the manual labeling was still a hard task. It is really time-consuming and it is very easy to make mistakes, so the work should be also checked in the end, which again takes a lot of time.

Another thing to take into account was the list of possible activities from which to choose from when annotating. We already had more than 23 000 labeled images, so to maintain full compatibility with them, the activities (categories) for the labeling were kept the same as in the other work [5]. These activities are a word or short description of what is visualized in the picture. Example activities are for instance driving, cooking, having drinks with somebody and so on. The final list containing all the different activities is shown in Table 3.1.1.

Public transport
Driving
Walking outdoors
Walking indoors
Biking
Having drinks with somebody
Having drinks\meal alone
Having meal with somebody
Socializing
Attending a seminar
Meeting
Reading
TV
Cleaning and chores
Working
Cooking
Shopping
Talking
Resting
Mobile
Plane

Table 3.1.1: Final list of activities

Having to do all this manual work again shows how better it is to have unsupervised approach in which there is no need to annotate so many images. It also shows how difficult it is to find the right number of activities and to choose what they to be. The final categories are not the best, because some of them are very similar and even for a human it is very difficult to distinguish between them - like for example socializing and talking. Also there are some activities which are pretty common, but are not included in the final list - like doing a sport or playing a musical instrument. Some not so common activities can be seen, too. One example is praying or going to the church. This shows that an approach in which we do not need to know the number of clusters in advance can be good, because it will find and model all these differences between the different lifestyles.

However, the activities were kept the same, because of the compatibility with what is already annotated and because it is not an easy task to come up with the perfect categories - a new and different list can have both advantages and disadvantages.

Finally, using the annotation tool [11], more than 21 000 of images were labeled. Together with the images, labeled for the other work - Recognizing Activities of Daily Living from Egocentric Images, this totals to 45 000 labeled images in total, 15 000 for each of the three lifeloggers. The concrete number of annotated images for each activity and for each user can be seen in Table 3.1.2. Some histograms with the images for each of the lifeloggers are presented in Figure 3.1.3 - 3.1.5. The histograms were made using Pygal [1].

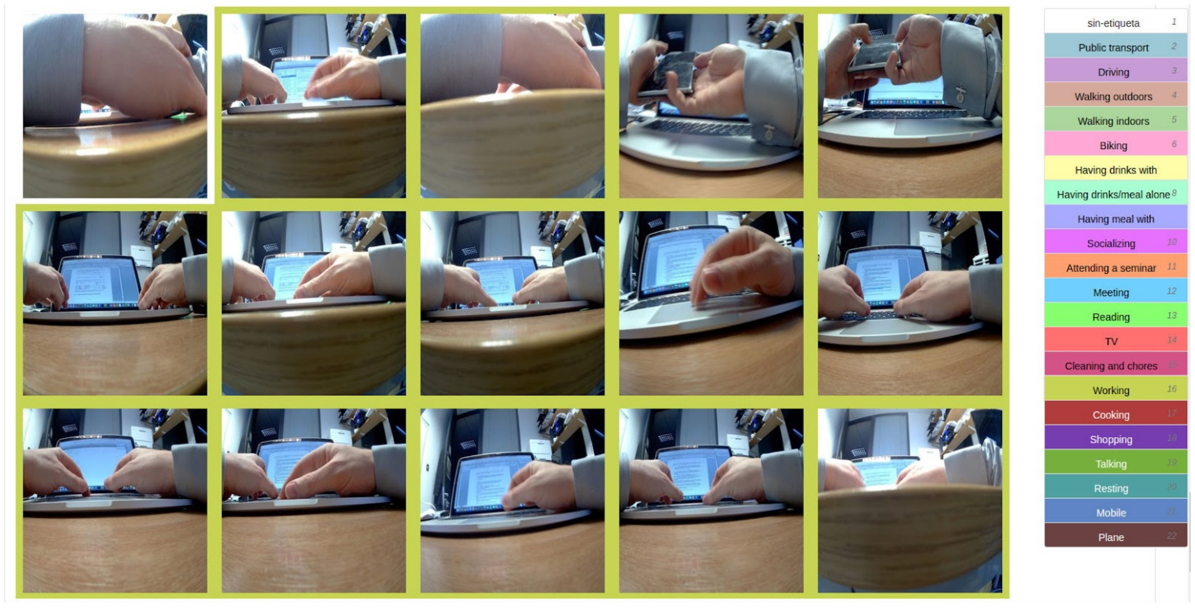


Figure 3.1.1: Labeling images with the activity “Working” using the annotation tool [11]

	User 1	User 2	User 3	Total
Public transport	183	237	1154	1574
Driving	2159	26	816	3001
Walking outdoors	688	1743	1045	3476
Walking indoors	610	827	304	1741
Biking	0	247	0	247
Having drinks with somebody	167	420	831	1418
Having drinks\meal alone	510	496	646	1652
Having meal with somebody	429	406	230	1065
Socializing	273	495	1057	1825
Attending a seminar	508	515	0	1023
Meeting	1025	751	0	1776
Reading	880	276	18	1174
TV	726	2	661	1389
Cleaning and chores	247	388	192	827
Working	1463	4326	1182	6971
Cooking	192	176	207	575
Shopping	595	250	330	1175
Talking	1660	682	309	2651
Resting	1250	1127	2988	5365
Mobile	1001	1342	2702	5045
Plane	434	268	328	1030
Total	15000	15000	15000	45000

Table 3.1.2: Number of annotated images for each user and activity

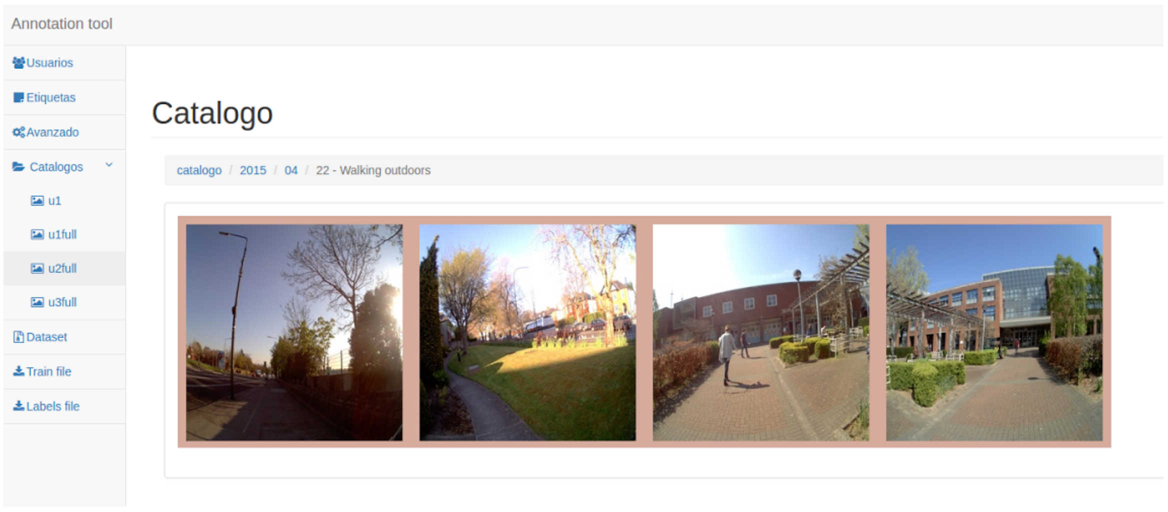


Figure 3.1.2: Some of the images, labeled with the activity "Walking outdoors" presented in the annotation tool [11]

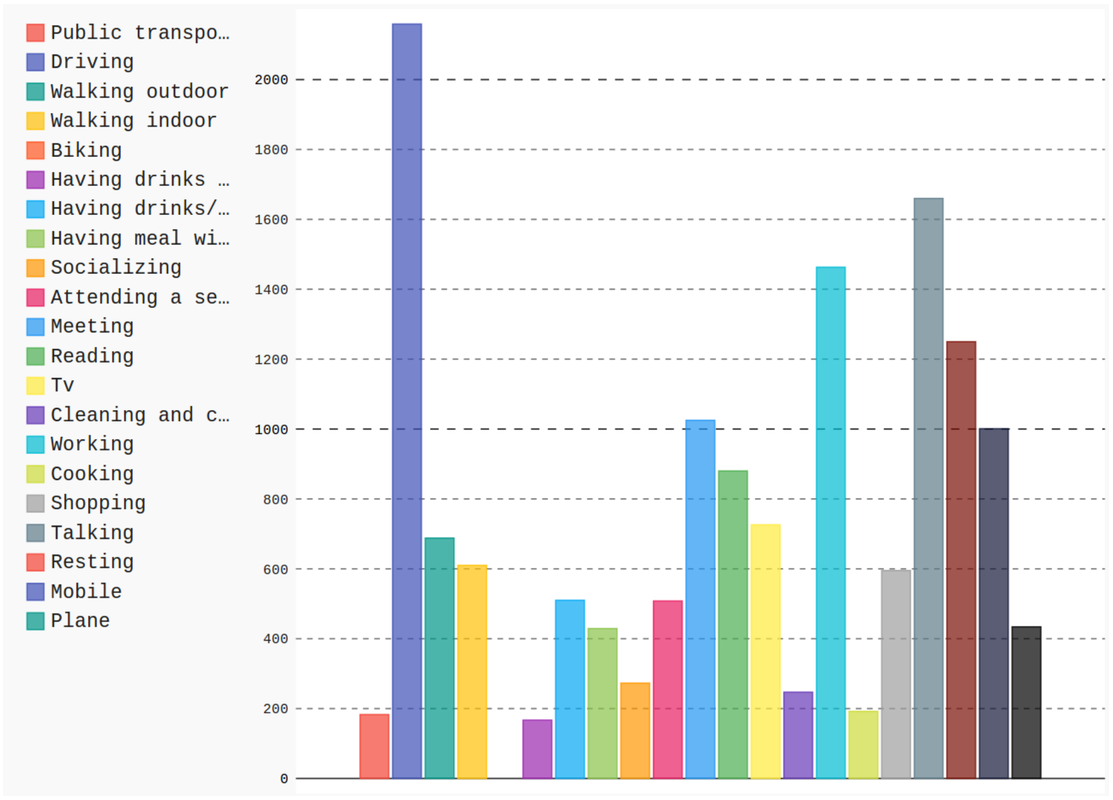


Figure 3.1.3: Histogram of the number of images for each activity for user (lifelogger) 1. We can see that he drives a lot and also likes talking.

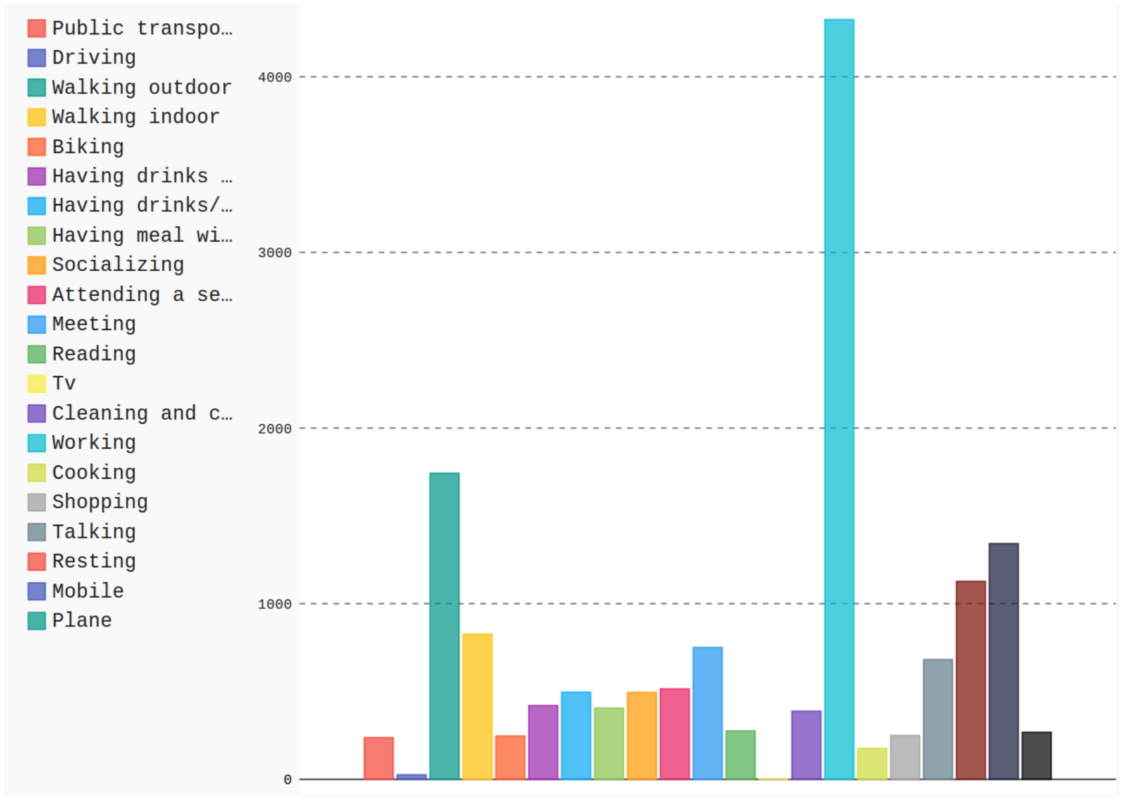


Figure 3.1.4: Histogram of the number of images for each activity for user (lifelogger) 2. He is really dedicated to his work.

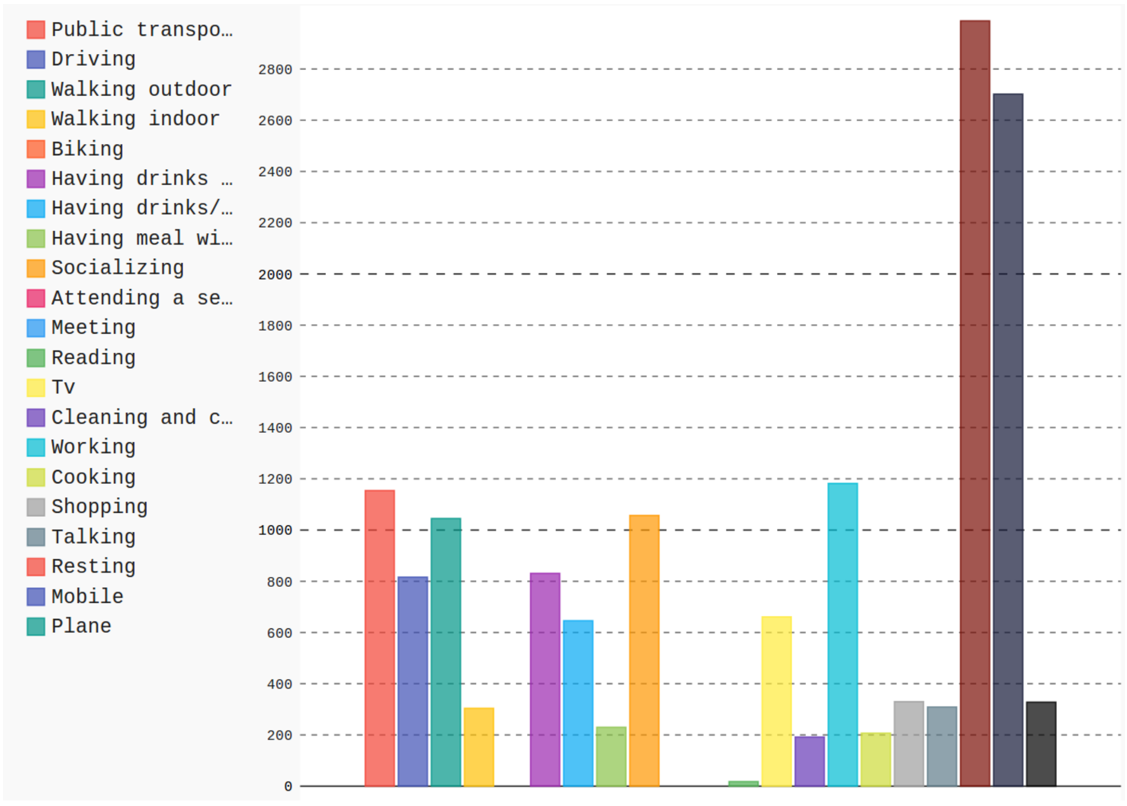


Figure 3.1.5: Histogram of the number of images for each activity for user (lifelogger) 3. He likes to have rest and is using his mobile phone very often.

3.2 Experimental setting

We wanted to start our experiments by trying something simple and fast, so that we can have a baseline and we can be sure that we are moving in the right direction and improving the results as much as possible.

Our idea was to use a deep learning framework and with its help to fine-tune already pretrained convolutional neural network. To do this we chose Caffe [1] - a deep learning framework, allowing easily to perform fine-tuning over networks trained with ImageNet. The neural networks that we used for the fine-tuning process was GoogLeNet. The base weights were extracted from the Caffe Github Repository [14]. For writing the code and running of the experiments, iPython notebooks were used [15].

We first tried this approach using all the data we have for all the users. The reason we did this is because we wanted to improve the intra-class and between class generalization capability of the CNN and to see how it performs.

Next, we wanted to simulate the case when we already have finetuned network with images from a lot of different users and we use it to recognise the activities of a new (unseen) user. As we have data only from three users, we split the dataset and used for traning and validation only the images from two users. The third user is left to be new (or unseen). We wanted to see if and how the performace drops in this case, because actually this is the purpose of the system - it should work fine with new users and our method should allow the customization of the system to any kind of user with any kind of activities.

Since we have the data of three users, we performed the fine-tuning process four times. At first we used all the images for all the users. We split the data into three subsets - one for training (about 75% of the data or around 33 750 images), one for validation (about 10% or 4500 images) and the last one for testing (about 15% or 6750 images). We did this in order to see what performance we will get and how it will drop when we use the images for only two of the users for training and validation and test with the images for the third user, which will be unseen data for the model. After that we used the data for two of the users for training and validation and then tested the results on the data for the third user, so we fine-tuned three more times - once for each of the users.

The number of images used in each case for train, validation and test can be seen in Table 3.2.1. The batch size for GoogLeNet was reduced to 10, so that it can be run on a system with GPU with 2GB VRAM and the learning rate was set to 0.000067. All the parameters for the solver are presented in Table 3.2.2.

In order to evaluate the model, we used the metric accuracy. When we tested the trained networks we measured the accuracy for each class (activity) and then the global accuracy. Since here it is interesting to compare the results for the four trained networks, all the results are summarized in Table 3.2.3.

	Test on images from user 1, train and validate on images from user 2 and 3	**Test on images from user 2, train and validate on images from user 1 and 3**	**Test on images from user 3, train and validate on images from user 1 and 2**	**Using images from all users**
Training	27000 (90% of the images from user 2 and 3)	27000 (90% of the images from user 1 and 3)	27000 (90% of the images from user 1 and 2)	33 750 (75% of all images)
Validation	3000 (10% of the images from user 2 and 3)	3000 (10% of the images from user 1 and 3)	3000 (10% of the images from user 1 and 2)	4500 (10% of all images)
Test	15000 (100% of the images from user 1)	15000 (100% of the images from user 2)	15000 (100% of the images from user 3)	6750 (15% of all images)
Total	45000	45000	45000	45000

Table 3.2.1: Number of images, used for train, validation and test sets in each case

	Test on user 1, train on user 2 and 3	**Test on user 2, train on user 1 and 3**	**Test on user 3, train on user 1 and 2**	**Using images from all users**
batch size	10	10	10	10
test iterations	59	59	59	89
test interval	673	674	672	841
base lr	0.000067	0.000067	0.000067	0.000067

display	168	168	168	210
max iterations	26920	26980	26900	33670
lr policy	"step"	"step"	"step"	"step"
gamma	0.10000	0.10000	0.10000	0.10000
momentum	0.9	0.9	0.9	0.9
weight decay	0.005	0.005	0.005	0.005
stepsize	13460	13490	13450	16835
snapshot	13460	13490	13450	16835
solver mode	GPU	GPU	GPU	GPU
solver type	SGD	SGD	SGD	SGD

Table 3.2.2: Solver definition for the fine-tuning

Accuracy	**Test on user 1, train on user 2 and 3**	**Test on user 2, train on user 1 and 3**	**Test on user 3, train on user 1 and 2**	**Using images from all users**
Public transport	36%	39%	29%	**89%**
Driving	96%	92%	57%	**98%**
Walking outdoors	81%	93%	**96%**	92%
Walking indoors	65%	57%	53%	**69%**
Biking	-	-	-	82%
Having drinks with somebody	62%	39%	21%	**81%**
Having drinks\meal alone	47%	64%	70%	**80%**

Having meal with somebody	60%	57%	43%	**82%**
Socializing	9%	33%	21%	**77%**
Attending a seminar	6%	4%	-	**84%**
Meeting	42%	47%	-	**79%**
Reading	64%	63%	83%	**87%**
TV	58%	50%	21%	**85%**
Cleaning and chores	30%	27%	**73%**	63%
Working	54%	85%	88%	**93%**
Cooking	14%	65%	38%	**65%**
Shopping	56%	**82%**	70%	78%
Talking	45%	48%	42%	**83%**
Resting	79%	57%	63%	**92%**
Mobile	70%	64%	71%	**86%**
Plane	6%	73%	1%	**87%**
Total Accuracy	**59%**	**66%**	**57%**	**87%**

Table 3.2.3: Obtained accuracies after fine-tuning

As we can see from Table 3.2.3 and as expected, the accuracy dropped with between 21% and 30% when we trained on one dataset and then tested on another dataset, which is not similar to the images, used for training and is new for the model.

What is interesting here is to explore what is the accuracy for each of the activities and each of the users and why.

If we concentrate on the accuracies in Table 3.2.3 and on Table 3.1.2 where we have the number of pictures for each user and category, we can notice some dependencies. For example as we can see in Table 3.2.4. for the activity Having drinks with somebody we have a lot of images for user 3, but much less for user 1 and 2. So when the data for user 3 is unseen, we are training this activity with 587 images and we are testing with 831 images, so probably this is the reason we get accuracy of only 21% in this case.

We can see another interesting example in Table 3.2.5. At all we have 1023 images for the activity Attending a seminar, so we get good accuracy when we train and test with known data. The problem comes with unseen data, because one of the users do not have images in this category. So when we treat the images of user 1 as unseen data and test on them, we actually have trained the model only on the images from user 2, which makes it really difficult, because the images from the activity Attending a seminar can look quite different for different people - they depend on the room, if there is a screen on the pictures, if there are other people and so on. This is why we get accuracy of 6% and 4%.

Having drinks with somebody	**User 1**	**User 2**	**User 3**	**Total**
Number of images	167	420	831	1418
Accuracy	62%	39%	21%	81%

Table 3.2.4: Number of images and accuracies for the activity Having drinks with somebody

Attending a seminar	**User 1**	**User 2**	**User 3**	**Total**
Number of images	508	515	0	1023
Accuracy	6%	4%	-	84%

Table 3.2.5: Number of images and accuracies for the activity Attending a seminar

3.3 Results

The first approach we tried was to use GoogLeNet in an ensemble with another classifier. First, we fine-tuned GoogLeNet, using the Caffe framework [13] with batch size of 10 images and a learning rate $\alpha = 6.7\text{x}10^{-5}$. We did this four times with the different combinations of subsets. We then extracted features from different final output layers of the CNN - first the softmax probability layer, giving a vector of 21 features and second, a combination of the pool5/7x7 fully connected layer and the softmax probability layer, giving a vector of 1045 features. After this, we trained several models by combining the different final output layers of the CNN with different clustering and classification algorithms. We tried the following ensembles:

- GoogLeNet+RandomForest - for this model we trained a random forest of 500 trees. We did this twice - first only on the softmax probability layer, giving a vector of 21 features and then on the pool5/7x7 fully connected layer and the softmax layer, giving a vector of 1045 features. This is a fully supervised approach as we need all the labels for both the CNN and the Random Forest algorithm. This corresponded to test the generalization performance of the algorithm proposed by Juan Marin [5] to unseen users. The results can be seen in Table 3.3.1 and Table 3.3.2.
- GoogLeNet+kMeans - we applied kMeans twice - on the softmax probability layer (21 features) and second time on the pool5/7x7 fully connected layer and the softmax probability layer (1045 features). Here we needed to specify the number of classes. Results are in Table 3.3.3 and Tables 3.3.4.
- GoogLeNet+SpectralClustering - again applied the clustering algorithm twice - on the softmax probability layer and on the pool5/7x7 fully connected layer and the softmax probability layer. Results are in Table 3.3.5 and Tables 3.3.6.

CNN (soft-max probabilities) + RandomForest

	Test on user 1 (train on user 2 and user 3)	**Test on user 2 (train on user 1 and user 3)**	**Test on user 3 (train on user 1 and user 2)**
Accuracy	62%	67.11%	58.89%
CNN	59%	66%	57%

Table 3.3.1 Results from applying CNN (soft-max probabilites) + RandomForst

CNN (pool5-7x7_s1_probs) + RandomForest

	Test on user 1 (train on user 2 and user 3)	**Test on user 2 (train on user 1 and user 3)**	**Test on user 3 (train on user 1 and user 2)**
Accuracy	65%	70%	62%
CNN	59%	66%	57%

Table 3.3.2 Results from applying CNN (pool5-7x7_s1_probs) + RandomForst

As expected from [5] we can see that using the network to obtain features and then combine them with another classifier (RandomForest in our case) improves the results. When using only the soft-max probabilites we see an improvement in the accuracy with 1 to 3 percents. This means the network performs well as fixed feature extractor. What makes it perform worse is the fact that is uses a very simple classifier - softmax to predict the final label. When we replace it with a better and more complex classifier like Random Forest, we get better results. Also as expected, using pool5-7x7_s1_probs insead of softmax probabilities gives bettwe results.

CNN (soft-max probabilities) + kMeans

	Test on user 1 (train on user 2 and user 3)	**Test on user 2 (train on user 1 and user 3)**	**Test on user 3 (train on user 1 and user 2)**
NMI	50.15%	51.09%	49.36%
Purity	67.17%	67.64%	69.93%
Accuracy	59.29%	65.12%	63.62%
CNN	59%	66%	57%

Table 3.3.3 Results from applying CNN (soft-max probabilites) + kMeans

CNN (pool5-7x7_s1_probs) + kMeans (with default parameters)

	Test on user 1 (train on user 2 and user 3)	**Test on user 2 (train on user 1 and user 3)**	**Test on user 3 (train on user 1 and user 2)**
NMI	56.38%	50.33%	58.74%
Purity	71.68%	71%	76.49%
Accuracy	65.58%	62.33%	71.58%
CNN	59%	66%	57%

Table 3.3.4 Results from applying CNN (pool5-7x7_s1_pr obs) + kMeans

The results for CNN combined with kMeans are not very consecutive. It’s not very surprising to see that when we use soft-max probabilities as features, the results are almost the same - this is a very simple feature and it looks like there are not any well-formed clusters. However, when we use pool5-7x7_s1_probs layers, in two of the cases, the results boost quite a lot - with 6 to 14%. This is actually the higest accuracy obtained when we test as user 3 is unseen.

CNN (soft-max probabilities) + Spectral Clustering

	Test on user 1 (train on user 2 and user 3)	**Test on user 2 (train on user 1 and user 3)**	**Test on user 3 (train on user 1 and user 2)**
NMI	51.1%	51.33%	49.87%
Purity	67.81%	68.08%	68.96%
Accuracy	60.01%	65.73%	64.42%
CNN	59%	66%	57%

Table 3.3.5 Results from applying CNN (soft-max probabilities) + SpectralClustering

CNN (pool5-7x7_s1_probs) + Spectral Clustering

	Test on user 1 (train on user 2 and user 3)	**Test on user 2 (train on user 1 and user 3)**	**Test on user 3 (train on user 1 and user 2)**
NMI	0.402079467548	0.6%	0.4%
Purity	0.918953201148	100%	100%
Accuracy	0.350757727485	28.86%	25%
CNN	59%	66%	57%

Table 3.3.6 Results from applying CNN (pool5-7x7_s1_pr obs) + SpectralClustering

The results for Spectral Clustering are not good in all the cases, which means this approach is not suitable for all datasets we use.

Semi-supervised approach

The other approach we tried uses again features obtained from the network - from pool5-7x7_s1_probs layers. They are 1045-dimensional vectors. This time we apply agglomerative clustering to the features in order to split the images in groups, which contain similar images where the same activity is performed. As it is impossible all clusters to be perfect, we should find a way to realize which of the clusters are pure (containing image with only one activity) and which are not. The idea is to ask the user to put only one label for all the images in the pure clutsers (because all the images are with the same activity being performed).

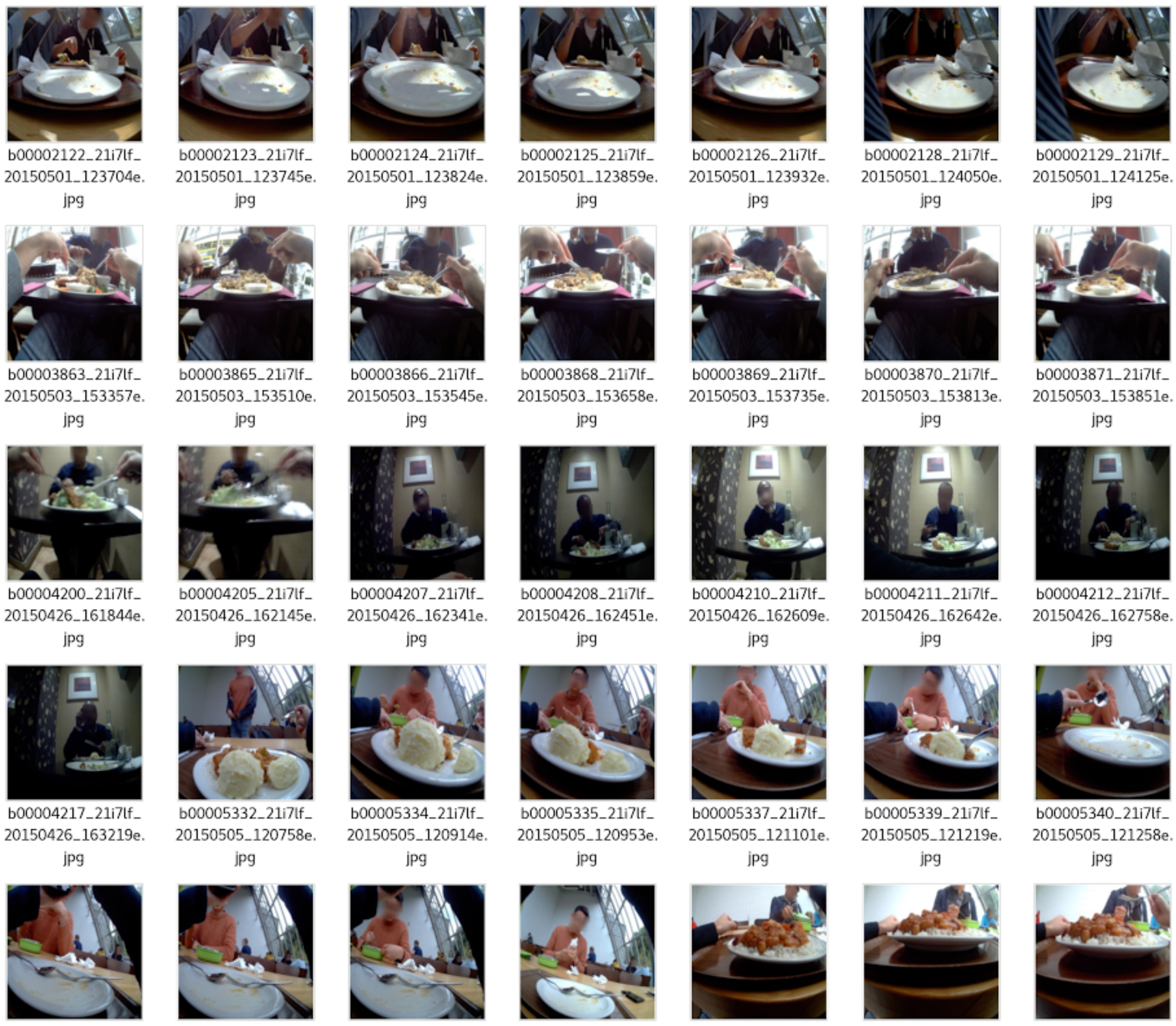
The first problem here is to choose in how many clusters to split the data. Unfortunaltely, there is no good metric saying this. It is nice to have as little clusters as we can, because we will have to label each cluster and more clutsers will mean more data to label, but less clusters means more images in each cluster. When we have a lot of images in the clusters there is less chance that they will be pure.

We tried several experiments, the results from which can be seen in Table 3.3.7. We used user 2 as unseen user, because only he has images for activity, which the others two do not have. Only user 2 rides a bike. We split his 15000 images into two subset - one 12750 images (75%) and one 2250 images (15%). The last one is used only in the final stage for testing. So in the table the results are from clusteting of 12750 vectors, which are 1045-dimensional and are obtained from the GoogLeNet, finetuned for user 1 and user 3.

Average images in cluster (number of clusters)	30 (425)	50 (255)	80 (159)	100 (127)
silhouette	0.0554	0.0442	0.0393	0.0370
calinski harabaz	60.2473	85.3951	118.6345	139.0057
NMI	0.5533	0.5537	0.5572	0.5600
purity	0.8572	0.8264	0.8044	0.7947
accuracy	0.8571	0.8262	0.8043	0.79458

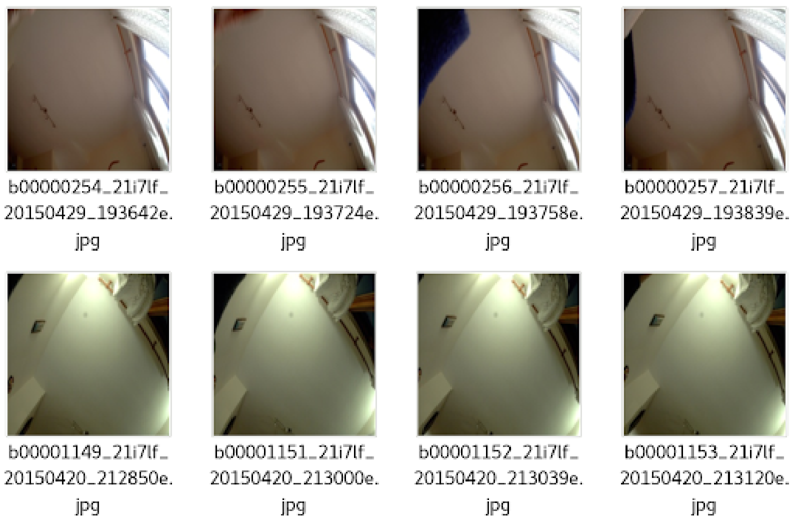
Table 3.3.7 Clustering with different number of clusters, when user 2 is the unseen user

Looking at the imags in the clusters we obtained and at the table above, we decided to try with around 100 images in cluster. As we have 12750 images, we got 127 clusers at all. The clusters contain different numbers of images, but after looking at the results, most of them are pure and have collected images with the same activity even though the images theyself are quite different. Some nice examples can be seen in Fig 3.3.1. Some of the images are almost identical, so they logically go to a cluster together. Such example is in Fig. 3.3.2



b00002122_21i7lf_20150501_123704e.jpg
b00002123_21i7lf_20150501_123745e.jpg
b00002124_21i7lf_20150501_123824e.jpg
b00002125_21i7lf_20150501_123859e.jpg
b00002126_21i7lf_20150501_123932e.jpg
b00002128_21i7lf_20150501_124050e.jpg
b00002129_21i7lf_20150501_124125e.jpg
b00003863_21i7lf_20150503_153357e.jpg
b00003865_21i7lf_20150503_153510e.jpg
b00003866_21i7lf_20150503_153545e.jpg
b00003868_21i7lf_20150503_153658e.jpg
b00003869_21i7lf_20150503_153735e.jpg
b00003870_21i7lf_20150503_153813e.jpg
b00003871_21i7lf_20150503_153851e.jpg
b00004200_21i7lf_20150426_161844e.jpg
b00004205_21i7lf_20150426_162145e.jpg
b00004207_21i7lf_20150426_162341e.jpg
b00004208_21i7lf_20150426_162451e.jpg
b00004210_21i7lf_20150426_162609e.jpg
b00004211_21i7lf_20150426_162642e.jpg
b00004212_21i7lf_20150426_162758e.jpg
b00004217_21i7lf_20150426_163219e.jpg
b00005332_21i7lf_20150505_120758e.jpg
b00005334_21i7lf_20150505_120914e.jpg
b00005335_21i7lf_20150505_120953e.jpg
b00005337_21i7lf_20150505_121101e.jpg
b00005339_21i7lf_20150505_121219e.jpg
b00005340_21i7lf_20150505_121258e.jpg

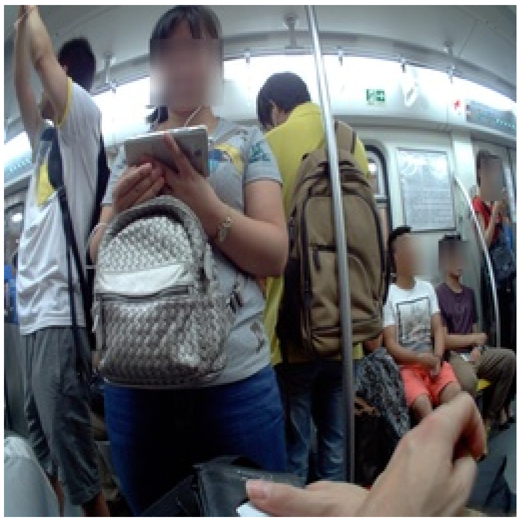
Fig 3.3.1 Some images from one cluster - all with the same activity - Having meal with someone



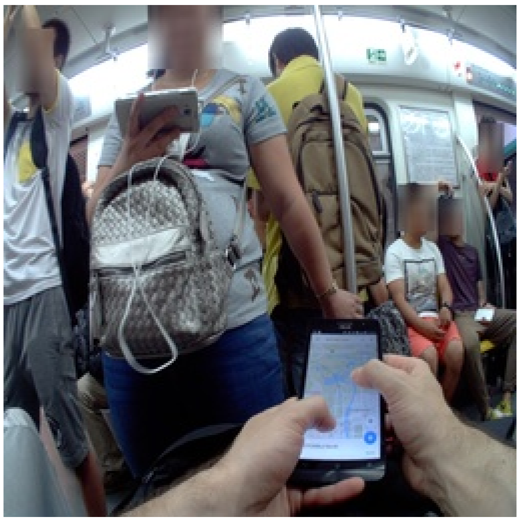
b00000254_21i7lf_20150429_193642e.jpg
b00000255_21i7lf_20150429_193724e.jpg
b00000256_21i7lf_20150429_193758e.jpg
b00000257_21i7lf_20150429_193839e.jpg
b00001149_21i7lf_20150420_212850e.jpg
b00001151_21i7lf_20150420_213000e.jpg
b00001152_21i7lf_20150420_213039e.jpg
b00001153_21i7lf_20150420_213120e.jpg

Fig 3.3.2 Almost identical images

One of the bigest problems is that a big amount of the images have two activities in them - for example working and eating, walking and mobile and so on, so it’s very difficult to distinguish them, bacause they look very similar, but the small rectangle in the bottom of the screen can change the right activity from walking to mobile. Such example is shown in Table 3.3.8.



Public transport



Mobile

Table 3.3.8 Almost identical images with different activities

Another clusters just contain very diffent images. Our aim is to find the one which are not pure and to query the user only for the pure ones. Unfortunately, here the metrics from above (silhouette index, calinski harabaz index) did not give any useful information about the purity of the clusters.

We tried to combine several other metrics to obtain better information about how good are the clusters. Again using clustering we split each cluster into two smaller clusters. We then found the centroids of the two subclusters and the distance between them. If the two centroid are close - this means that maybe the cluster is pure. If they are far away, maybe we have different activities in the cluster. Another thing we measured is the distance from the centroid to the point with is most far away. We sorted the clusters by this metrics and put a sensitive threshold which of them are pure and which not. Another thing we did to find good clusters was to obtain the predicted label from the network for each images in the cluster. We then measured how many percents of the images in the cluster are predicted to be from the most frequent activity - if over 85% percents of the images are from the same activity, probably the cluster is pure.

The reason why the user has to label manually some images is that when we have new activity, which is unseen for the network, there is no way to guess it automatically.

Using the approach above we selected 73 good clusters for the user to label out the of 127 clusters and labeled all the images in the cluster with the same label. This means that the user have to choose 73 labels for the 73 clusters. We labeled the clusters and when applied this labels to all the images in the cluster, we actually labeled 8622 images out ot 12750 by choosing only 73 lables. The accuracy of the labels put in this way is 85.14%. We then finetuned the network we had with the new images. The global accuracy of the network improved from 64% to 66%.

As it is quite difficult to find the pure clusters and no metric is very reliable, another possible approach is to show all the clusters to the user and ask him/her to label only the ones which contain images of the same activity. We tried this approach and selected 85 clusters out of 127 for pure. When labeling them (putting 85 labels) we actually labeled 8376 images with accuracy 87.89%. After finetuning the network with the new images, we obtained much better results. The improvements are shown below. On the left are the numbers before fintuning, on the right are after finetuning. Accuracy has increased with 12%. This shows that is it very important to have the new activities labeled corectly - like biking for example. There were no such labels before, so the network couldn't guess this activity, but when we finetuned it with some biking images the results incresed from 0 to 80% for this caregory.

public transport: 47 -> 43
driving: 100
walking outdoor: 89 -> 84
walking indoor: 48 -> 44
biking: 0 -> 80
having drinks with somebody: 33 -> 76
having drinks/meal alone: 71 -> 63
having meal with somebody: 40 -> 53
socializing: 51 -> 20
attending a seminar: 3 -> 53
meeting: 31 -> 53
reading: 60 -> 73
tv: 0
cleaning and chores: 32 -> 56
working: 87 -> 96
cooking: 78 -> 5
shopping: 70 -> 48
talking: 40 -> 75
resting: 46 -> 77
mobile: 66 -> 85
plane: 91

Overall accuracy: 64% -> 76%

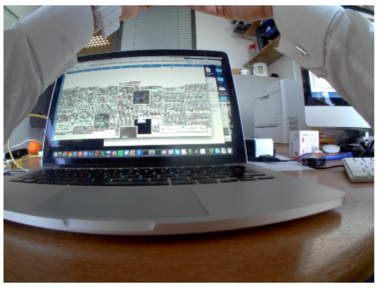
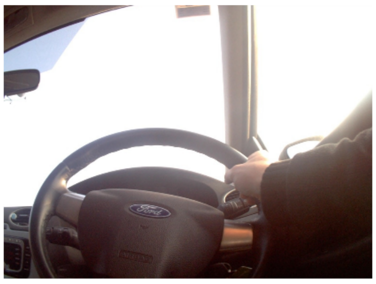
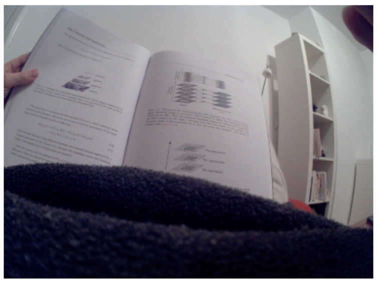
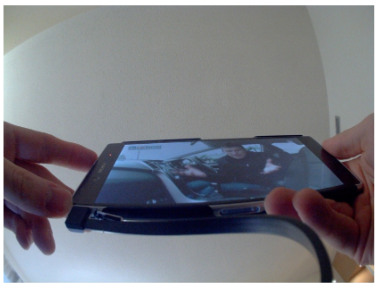
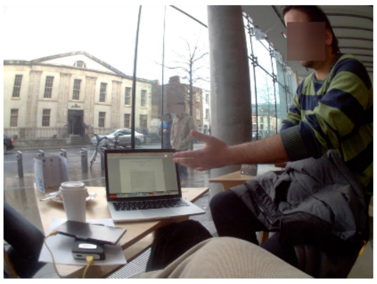
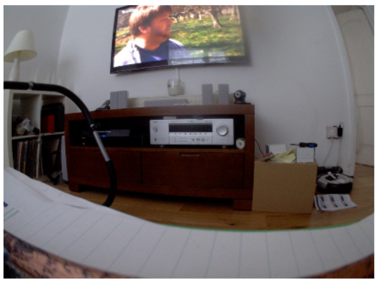
JULE

We also tried using the JULE approach for getting nice clusters. Unfortunately the images they used in their experiments are more simple than our and we need a deeper network to make it work and give nice results. We got a little improvement over kMeans, but it is not enough for good results and further investigation is needed.

kMeans: NMI: 0.227110318957 Purity: 0.452733333333 Accuracy: 0.337	**JULE:** NMI: 0.270105200465 Purity: 0.547866666667 Accuracy: 0.352866666667

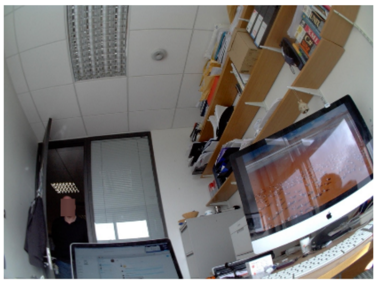
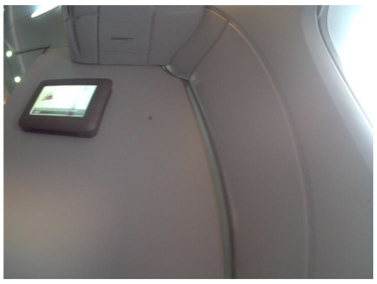
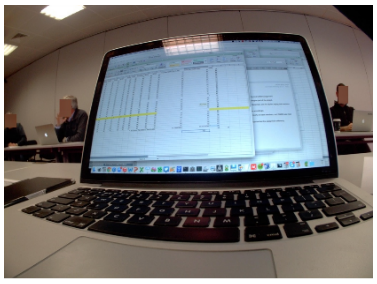
3.4 Discussion

In the table below are some correctly classified activities, but more interesting are the mistakes in the next table.

		
Working	Driving	Reading
		

Mobile	Talking	TV
Resting	Walking outdoor	Reading

Table 3.4.1: Correctly classified images

		
Talking	Cleaning and chores	Attending a seminar
Working	TV	Plane
		
Having meal with somebody	Plane	Having drinks with smbd
Socializing	Walking indoor	Having meal with somebody
		

Meeting	Driving	Talking
Working	Public transport	Meeting

Table 3.4.2: Some mistakes

As we can see from the mistakes above, most of them are because we actually have two activities in the same time. A possible solution for this is to use the top 2 labels instead of top 1.

Trying this really boost a lot the results. In the normal case, before finetuning with the labeled images from the clusters, for user 2 we had 64% accuracy. When using top 2 results, this went to 76%. In the case wih the finetunig from 76% the accuracy went to 86%.

Another thing is that a lof of the labels are confusing even for us - like talking and socializing for example.

4 Conclusions and Future Work

For the future we should address the problem with several activities on one images and detect all performed activities.

We also worked on appling the recurrent framework for Joint Unsupervised LEarning (JULE) of deep representations and image clusters [6] in the domain of egocentric images and activity recognition. What we did was to use the framework to train a model and obtain clusters of images representing the same activity, instead of the agglomerative clustering we used. The approach seems suitable for our work, but it should be adapted to work with a more deep network as we need for our task.

Maybe a better selection of labels can also help improve the results.

5 References

1. A. Fathi, A. Farhadi, and J. M. Rehg. Understanding egocentric activities. In 2011 International Conference on Computer Vision, pages 407–414. IEEE, 2011.
2. H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 2847–2854. IEEE, 2012.
3. Xu Sun, Hisashi Kashima, Ryota Tomioka, Naonori Ueda, and Ping Li. A New Multi-Task Learning Method for Personalized Activity Recognition. In Data Mining (ICDM), 2011 IEEE 11th International Conference on
4. Yan Yan, Elisa Ricci, Gaowen Liu, and Nicu Sebe. Egocentric Daily Activity Recognition via Multitask Clustering
5. Alejandro Cartas, Juan Marin, Petia Radeva, and Mariella Dimiccoli. Recognizing Activities of Daily Living from Egocentric Images
6. Jianwei Yang, Devi Parikh, Dhruv Batra. Joint Unsupervised Learning of Deep Representations and Image Clusters
7. I. Martin-Lesende, K. Vrotsou, I. Vergara, A. Bueno, A. Diez, et al. Design and validation of the vida questionnaire, for assessing instrumental activities of daily living in elderly people. J Gerontol Geriat Res, 4(214):2, 2015.
8. Dent, E., Kowal, P., & Hoogendijk, E. O. (2016). Frailty measurement in research and clinical practice: a review. European journal of internal medicine, 31, 3-10.
9. S. M. Sch¨ussler-Fiorenza Rose, M. G. Stineman, Q. Pan, H. Bogner, J. E. Kurichi, J. E. Streim, and D. Xie. Potentially avoidable hospitalizations among people at different activity of daily living limitation stages. Health services research, 2016.
10. http://research.nii.ac.jp/ntcir/permission/ntcir-12/perm-en-Lifelog.html
11. https://github.com/hermetico/TFG/tree/master/annotation-tool
12. http://pygal.org/en/stable/
13. http://caffe.berkeleyvision.org/
14. https://github.com/BVLC/caffe
15. https://ipython.org/notebook.html
16. D. Castro, S. Hickson, V. Bettadapura, E. Thomaz, G. Abowd, H. Christensen, and I. Essa. Predicting daily activities from egocentric images using deep learning. In proceedings of the 2015 ACM International symposium on Wearable Computers, pages 75–82. ACM, 2015.
17. Nguyen THC, Nebel JC, Florez-Revuelta F, et al., Recognition of activities of daily living with egocentric vision: A review. Sensors 2016; 16(1):72
18. Sanjoy Dasgupta, Daniel Hsu. Hierarchical sampling for active learning. ICML '08 Proceedings of the 25th international conference on Machine learning, pages 208-215
19. http://cs231n.github.io/convolutional-networks/