Shopper Interaction Using Deep Learning

Silvia Cecchini, Luca Rossi, Valentina Sabatini

July 19, 2018

Abstract

The aim of this paper is to get an insight of the interest that customers put on the products placed on a shelf, analyzing the interactions they have with the products themselves in a retail environment. A RGB-D camera is mounted on the ceiling and monitors whether the customers look at specific products, take them or put them back. The core of this project is to recognize whether customers have products in their hands or not, discriminating the case when an operator is refilling the shelf. We also introduce an approach to recognize the type of interaction that happens in a sequence of frames. The methodology described in this paper mainly involves deep learning approaches.

Keywords: customer behavior, retail environment, RGB-D camera, neural networks, deep learning.

1 Introduction

Human behavior analysis has been recently applied in different areas, such as: ambient assisted living [1], OOS (out-of-stock) problems [2], customer satisfaction in retail environment [3], customers path tracking [4], video surveillance [5][6]. In particular there are several studies about intelligent retail environments, with the purpose of investigating how customers behave inside a store.

Most approaches to human behavior analysis involve computer vision techniques, so it's becoming increasingly important the visual analysis of dynamic scenes [7][8] and data fusion from multiple cameras [9]. In this work we focus the attention on analysing and understanding human behaviors in shops, in particular how customers interact with shelves, in order to eventually take better marketing decisions.

There are several ways to use big data and computer vision techniques to achieve this goal, one of them is the analysis of the interest shown in the products by the customers. This can be done by analysing classes of interactions such as looking at products on the shelf, taking them, putting them back. These are all indicators of their interest in the products themselves, so a good marketing strategy should rely on the ability of a company to collect big data and extract useful knowledge from it.

The core of the project presented in this paper is to recognize whether customers have products in their hands or not, discriminating the case when an operator is refilling the shelf. We also introduced an approach to recognize the type of interaction that happens in a sequence of frames.

This paper is structured as following:

- In Section 2 we introduce the problem with its main goal and objectives, so we describe the general approach we decided to use to achieve the goal itself.
- In Section 3 we dig deeper into technical details, such as the types of interactions that have to be recognized, the dataset, some preliminary results used as a baseline for the final ones.
- In Section 4 we describe precisely the 3-step methodology we used to classify images and we introduce a general approach to classify sequences of images.
- In Section 5 we tune the parameters used in this work and described in Section 4, in order to achieve better results, and we show the final results obtained using our methodology.
- In Section 6 we draw our conclusions and introduce some possible future developments of the project.

2 Project goal and objectives

2.1 Problem description

The aim of this project is to analyze the behavior of people in retail shops in order to understand how they interact with the shelves and with the products placed on them (e.g. which products are frequently purchased, which ones are only looked at, which ones are taken and replaced on the shelves...).

RGB-D cameras are mounted on the ceiling and monitor whether the customers look at specific products, take them or put them back, capturing an image every time a person interacts with the shelf. The system is low cost and easy to install, the position of the cameras allows customers privacy, since the captured images don't contain any face.

Our work consists in analysing sequences of these frames and classify them accordingly to the type of interaction, which has to be deduced by analysing the frames themselves individually.

A system to classify the individual frames has already been developed, but in our work we used alternative approaches to improve this system, and finally we classified the frame sequences themselves.

In the next section we will list the types of interactions for both the individual frames and the sequences. Basically by observing a single frame we want to discriminate whether a customer approaches a product or has taken one. By observing a sequence of classified frames we want to understand if a product has just been picked up to be purchased or if the customer puts back a product which has been taken earlier.

Another important challenge is to discriminate the operation of refill, which happens when an operator puts new products on the shelf. Such operation can be identified only by looking at the whole sequence, since individual frames don't always portrait key feature of this operation, such as a box containing products.

2.2 First approach

Our method is mainly based on deep learning, specifically on the use of convolutional neural networks, particularly suitable for image processing. Four different networks have been tested in order to choose the best results, we will refer to them as CNN, CNN2, AlexNet and CaffeNet.

We chose the best network by initially classifying 3 classes: positive, negative and neutral (more on this in the next section). This allowed us to compare the results obtained using our dataset with the ones obtained by the existing method (Shopper Analytics).

Then we used the best network again to classify 4 classes: positive, negative, neutral and refill. The results gave us a baseline for the final ones.

After that, we trained other networks on pair of classes in order to generate models that we used later, these pair of classes are refill/non-refill, neutral/non-neutral, positive/negative.

All these results will be showed in Section 3. In Section 4 we will define a 3-step approach, the order of these steps has been chosen after seeing the 2-classes results obtained in Section 3, since the final results are also influenced by whether the best networks are placed before or after the worse ones in the multi-step process. Each of these steps not uses the trained models, but also applies preprocessing and strategies such voting.

After that we introduced a general approach to identify the type of interaction which happens in a sequence of frames.

3 Setup and preliminary results

3.1 Types of interactions

In this paper, we refer to two categories of interactions: frame interactions and sequence interactions. Each category has 4 classes (refill, neutral, positive, negative), which have slightly different meaning in each category, so in this section we will explain the meaning of each one to prevent any ambiguity.

The types of frame interactions are the following:

- Positive: there is a hand holding a product.

Figure 1: Example of a positive frame interaction

- Negative: there is a hand holding nothing.

Figure 2: Example of a negative frame interaction

- Neutral: there is a customer not interacting with the shelf.

Figure 3: Example of a neutral frame interaction

- Refill: there is a refill action, which happens every time a box filled with products is visible. This class has a "priority" over the others (e.g. if there is a hand holding a product and a box with products, the class is "refill" and not "positive"). Some frames can be associated to a refill operation even if there isn't a box but the sequence of frames indicates a refill operation (more on this later), but these images won't be used in the training set.

Figure 4: Example of a refill frame interaction

From this point forward, we will refer to "images labeled as refill/neutral/positive/negative" and "images predicted as refill/neutral/positive/negative" using just the terms "refills", "neutrals", "positives" and "negatives" (the reference to either the labeled or the predicted class will depend on the context).

The types of sequence interactions are the following:

- Positive: a customer, initially not holding any product, picks a product from the shelf and goes away, so the first non-neutral frame is negative and the last non-neutral frame is positive. There aren't any refill frames in the sequence.


Figure 5: Example of a positive sequence interaction

- Negative: a customer, initially holding a product, puts it on the shelf and goes away, so the first non-neutral frame is positive and the last non-neutral frame is negative. There aren't any refill frames in the sequence.


Figure 6: Example of a negative sequence interaction

- Neutral: nothing meaningful happens, so there are either only neutral frames, or the first non-neutral frame and the last non-neutral frame are both positive or both negative. There aren't any refill frames in the sequence.


Figure 7: Example of a neutral sequence interaction

- Refill: there are only refill frames or frames which represent a refill operation but can't be classified as refill frames (because the box is not visible).



Figure 8: Example of a refill sequence interaction

3.2 Dataset

The neural network has been trained using a dataset composed of 6143 RGB and depth frames, manually labeled with the classes previously defined. Since the dataset is unbalanced, we eliminated some images from it to obtain an equal number of refills, neutrals, positives and negatives, for a total of about 4000 images.

In particular, we labeled a dataset used for the training (training set and validation set) and a dataset used for the testing (test set). Before we continue, we need to mention two things about the test set:

- The methodology used to predict the class of images in the test set (giving us the final result) is different from the methodology used to predict the class of images in the validation sets (used in the process of training the networks used in the above mentioned methodology).

While the latter uses the model of the neural network trained on a corresponding training set, the former uses a 3-step process involving preprocessing, voting and sequences identification. This methodology will be discussed in the following section.

- The test set has been labeled differently from the training and validation sets. More precisely, the difference is in which images are refills. The images in the validation set have been labeled as refills only if the images themselves can be seen as refills, that means that there is always a box filled with products, that neural networks and humans can recognize. The test set, however, also contain images that can't be seen as refill themselves by neither neural networks nor humans (i.e. the box is missing), but we know they are refill because the previous and/or the following images in the same sequence can be seen as refills (i.e. contain the box). We didn't put images of this kind in any of the validation sets. More on this in the following section.

3.3 Results

We used 4 different neural networks: AlexNet, CaffeNet, CNN and CNN2, in order to compare results and find the best ones. At the beginning we considered only 3 classes: positive, negative and neutral (Table 1).

3 classes, RGB Frames					
		Train			
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
CNN	0.0042	0.9989	0.9984	0.9984	0.9984
CNN2	0.0015	0.9996	0.9996	0.9996	0.9996
AlexNet	0.0452	0.9844	0.9851	0.9835	0.9843
CaffeNet	0.1072	0.9999	0.9999	0.9999	0.9999
		Validation			
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
CNN	0.5450	0.9026	0.8549	0.8525	0.8537
CNN2	0.8171	0.8525	0.8549	0.8525	0.8537
AlexNet	0.7592	0.8455	0.8507	0.8398	0.8452
CaffeNet	0.8225	0.8553	0.8622	0.8539	0.8580

Table 1: 3 classes results (Positive, Negative, Neutral) on RGB frames.

The metrics obtained are really good, in particular all of them are above 80%. CNN gets the best results: accuracy on validation set reaches 90%, while precision, recall and f1score are equal to 85%.

As previously specified, we applied the same procedure using 4 classes: positive, negative, neutral and refill (Table 2):

4 classes, RGB Frames					
		Train			
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
CNN	0.0131	0.9955	0.9913	0.9907	0.9910
CNN2	0.0016	0.9995	0.9995	0.9995	0.9995
AlexNet	0.3953	0.8449	0.8653	0.8216	0.8425
CaffeNet	0.1418	0.9990	0.9990	0.9990	0.9990
		Validation			
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
CNN	0.3775	0.9186	0.8395	0.8340	0.8367
CNN2	0.7773	0.8611	0.8620	0.8611	0.8616
AlexNet	0.6115	0.7993	0.8164	0.7711	0.7928
CaffeNet	0.7608	0.8731	0.8768	0.8720	0.8743

Table 2: 4 classes results (Positive, Negative, Neutral and Refill) on RGB frames.

The metrics obtained with 4 classes are comparable to each obtained with 3 classes. Also in

this case CNN gets the best results with 91%, while precision, recall and f1score are equal to 83%. CNN2 and CaffeNet are slightly improved, while AlexNet is the only one who gets lower results compared to the previous case with 3 classes.

Although AlexNet is the network which gave us the worse results for 3 and 4 classes, it is the one who gave us the best results in every 2-class test. For brevity, the following results in this section will only show AlexNet's ones, and in the following section we will use only this network (since every further training will only involve 2 classes at a time.

In this second part we analyze the results obtained with the multi-step approach, starting with the refill class. Refill frames are simple to recognize thanks to the presence of a box on the ground. All the other frames are classified as non-refill. The dataset has been specially balanced for this class and it is composed of 2722 RGB frames. We analyzed the results obtained with AlexNet network and we report them below (Table 3):

2 classes: Refill/Non-Refill					
Train					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.0018	0.9994	0.9994	0.9994	0.9994
Validation					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.2120	0.9598	0.9598	0.9598	0.9598

Table 3: Refill/Non-Refill Results on RGB frames and balanced Dataset.

In this case we can see that the results are excellent.

Similarly, we have analyzed other 2 classes: neutral and non-neutral. Neutral images contain a person who is not interacting with the shelf. Here the results (Table 4):

2 classes: Neutral/Non-Neutral					
Train					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.1955	0.9230	0.9230	0.9230	0.9230
Validation					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.2058	0.9187	0.9187	0.9187	0.9187

Table 4: Neutral/Non-Neutral Results on RGB frames and balanced Dataset.

The results are very good with this class too, even if a little bit lower than the previous class.

Last step is the recognition of positive/negative frames. The difference between these two classes is the presence of a product in the hand (Table 5):

2 classes: Positive/Negative					
Train					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.0181	0.9940	0.9940	0.9940	0.9940
Validation					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.9376	0.8303	0.8303	0.8303	0.8303

Table 5: Positive/Negative Results on RGB frames and balanced Dataset.

This class is the one that obtains the lowest results.

3.4 Conclusions on first results

There are different types of architectures and neural networks but, as we noticed, the results do not improve by changing the architecture of the networks themselves, so the real work should be made on the dataset (preprocessing), as we will see.

We conclude this section with a comparison between 'Shopper Analytics' results [10] and our results previously obtained with 3 and 4 classes (Tables 6-7):

Shopper Analytics - 3 classes			
Test Set			
Networks	**Precision**	**Recall**	**F1Score**
CNN	0.7767	0.7688	0.7727
CNN2	0.7788	0.7788	0.7788
AlexNet	0.8291	0.8291	0.8291
CaffeNet	0.8140	0.8140	0.8140

Our Results - 3 classes			
Validation Set			
Networks	**Precision**	**Recall**	**F1Score**
CNN	0.8549	0.8525	0.8537
CNN2	0.8549	0.8525	0.8537
AlexNet	0.8507	0.8398	0.8452
CaffeNet	0.8622	0.8539	0.8580

Table 6: Comparison between "Shopper Analytics" and our results with 3 classes.

Shopper Analytics - 3 classes			
Test Set			
Networks	**Precision**	**Recall**	**F1Score**
CNN	0.7767	0.7688	0.7727
CNN2	0.7788	0.7788	0.7788
AlexNet	0.8291	0.8291	0.8291
CaffeNet	0.8140	0.8140	0.8140

Our Results - 4 classes			
Validation Set			
Networks	**Precision**	**Recall**	**F1Score**
CNN	0.8395	0.8340	0.8367
CNN2	0.8620	0.8611	0.8616
AlexNet	0.8164	0.7711	0.7928
CaffeNet	0.8768	0.8720	0.8743

Table 7: Comparison between "Shopper Analytics" results with 3 classes and our results with 4 classes.

As we can see from the comparison between the previous tables, the results obtained with 4 classes are comparable (sometimes better) than those with 3 classes related to "Shopper Analytics". The most satisfying thing is to have obtained similar results even working with 4 classes, and these results set a lower bound to the results we want to obtain by the end of this paper.

4 Methodology

4.1 Introduction

In the previous section we gathered some preliminary results from neural networks trained on 3 classes (neutral, positive, negative) to compare our results with Shopper Analytics' ones, on 4 classes (refill, neutral, positive, negative) to define a baseline, and 2 classes (refill and non-refill, neutral and non-neutral, positive and negative) to generate models which have been used later.

In this section we want to define a multi-step approach in which we use the 2-classes models as a pipeline, where each step filters the images associated to a specific class and sends the others to the next step. We also want to describe the methods used to identify and classify sequences.

In the next section we will show the results obtained by applying this methodology on a test set, comparing them with the results obtained using the initial network trained on 4 classes.

Our goal here is to get better results than the network trained on 4 classes, so each step of the methodology uses the 2-classes models previously created, plus some improvements (e.g. preprocessing) that we will describe soon.

In order to predict which class each frame of a test set belongs to and eventually predict the type of interaction in a sequence, given a dataset of RGB-D images with their timestamps, we follow a 3-step approach to classify those images:

1. Refills identification
2. Neutrals identification
3. Positives and negatives identification

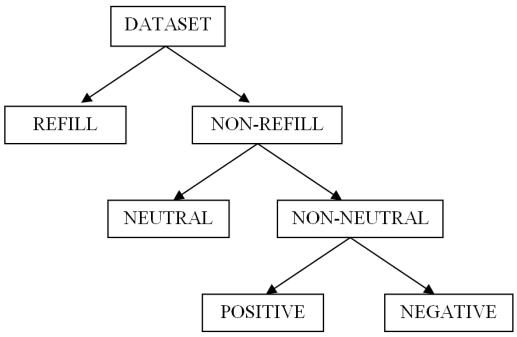

Figure 9: 3-step dataset classification

Each step excludes the predicted images from the next steps. This means that, for example, step 1 splits the original dataset in a sub-dataset with only refill images and a sub-dataset with only non-refill images, so step 2 only works with non-refill images, and so on.

Furthermore, the order in which those steps are performed is crucial. Refill images are the most ambiguous because they can be mistaken for neutrals, positives and negatives (there is always a person interacting with the shelf in a positive/negative/neutral way, but a box with products is usually visible too). So the best thing to do is to recognize those images first, so they can be excluded from further analysis. The refill class is also the one which gave the best results in the previous 2-classes test, so even if it wasn't ambiguous, it would have given the best overall results anyway if predicted first.

Predicting neutrals before positives and negatives is also important, firstly because of the better results it gave us in the 2-classes test, secondly because, as we will see later, it allows us to do some preprocessing for the last step, eventually improving the results of the 2-classes test for positives and negatives.

4.2 Dataset splitting

The sequence identification method we will introduce in this section involves the definition of two methods:

1. How to split the dataset in sequences
2. How to classify each sequence

Before we start talking about the 3-step methodology, we need to introduce how the dataset is split in sequence, because as we will see soon, step 1 will need them in order to identify all the refills in the dataset. We will talk about how to classify each sequence later, when every image of the dataset will have been associated to a class.

The method we will present in this section to split the dataset and classify the sequences is not a tested method (at least, not on a sufficient amount of data) because of the difficulty of evaluating the method itself (in order to compute some kind of metrics we would need to manually label a lot of sequences, letting alone that the sequences themselves have to be well defined, since we can't know priorly which frames belongs to a given sequence). So our goal here is to provide an introduction to a method, a framework where future works can be based on.

We identified several potential criteria to split the dataset in sequences (e.g. long sequences of neutrals, similar frames, frames not containing people...), but since we have a small dataset and we haven't defined metrics to evaluate our results, we kept it simple and used the timestamp as the only criteria to recognize whenever a sequence ends and a new sequence begins.

Using some parameters that have to be tuned, such has the maximum difference between the timestamps of the last and the first frame of a sequence, and the maximum difference between the timestamps of two consecutive frames, we defined the following function:

```
def split_dataset(dataset, sequences, max_timestamp_diff, max_timestamp_frame_diff):
        sequence = []
        timestamp = 0
```

```
prev_timestamp = 0
length = 0
for item in dataset:
        if timestamp == 0:
                sequence.append(item)
                timestamp = item.timestamp
                prev_timestamp = item.timestamp
                length = 1
        elif (item.timestamp - timestamp <= max_timestamp_diff or \
        item.timestamp - prev_timestamp <= max_timestamp_frame_diff) and \
        length < max_length:
                sequence.append(item)
                prev_timestamp = item.timestamp
                length += 1
        else:
                sequences.append((sequence, None))
                sequence = []
                sequence.append(item)
                timestamp = item.timestamp
                prev_timestamp = item.timestamp
                length = 1
```

Now that we have our list of sequences, we can describe the 3-step methodology, before describing how the sequences themselves are classified.

4.3 Step 1: refills identification

The refill class is the one which gave us the best metrics in the 2-classes test, but this only works with the images containing a visible box: the high results obtained in the 2-classes test are not enough for our purposes and we need to use a more sophisticated approach. So before we continue we have to consider two more things:

- As we said before, the test set contains also those refill images which can't be seen as refills (i.e. the box is missing) but the context (i.e. the sequence of frames) makes it clear that they should be classified as refills. In a given sequence, according to the definitions we gave in Section 3, either every image is a refill or every image is a non-refill.
- Knowing that every sequence is either refill or non-refill, we can still improve (or worsen, if we don't tune the parameters correctly) the results, for example changing a sequence of predictions like R-R-N-R-R to R-R-R-R-R (where R refers to refill and N to non-refill).

With these two things in mind, we can divide the first step in two sub-steps:

1.1 Identify refill images using the neural network trained before

1.2 Identify refill sequences in function of the number and the ratio of refill images (identified in step 1.1) in each sequence

The first step is performed through the following lines of Python code:

```
predict_refill(dataset, NN_REFILL)
split_dataset(dataset, sequences, MAX_TIMESTAMP_DIFF, \
MAX_TIMESTAMP_FRAMES_DIFF, MAX_LENGTH)
for sequence, prediction in sequences:
        prediction = predict_refill_sequence(sequence, MIN_REFILL,\
        MIN_REFILL_RATIO, MIN_REFILL_LENGTH)
reduced_dataset = reduce_dataset(dataset)
```

The function predict_refill() performs the step 1.1 by using the previously trained neural network model (on the two classes refill and non-refill) to classify the images of the dataset (which is

a list of objects where each object contains the RGB image, the depth image, the timestamp, the label and the predicted class).

We already saw the function split_dataset(). The function predict_refill_sequence() perform the step 1.2, which is also the first step of sequences classification, associating a given sequence (and all the images in it) to either the refill or non-refill class, in function of the number and the ratio of refill images in the whole sequence:

```
def predict_refill_sequence(sequence, min_refill, min_refill_ratio, \
min_refill_length):
        refill = False
        num_refill = 0
        if len(sequence) >= min_refill_length:
                for item in sequence:
                        if item.predicted_class == 'refill':
                                num_refill += 1
                if num_refill / len(sequence) >= min_refill_ratio and \
                num_refill >= min_refill:
                        refill = True
        for item in sequence:
                item.predicted_class = 'refill' if refill else None
        return 'refill' if refill else None
```

After these two sub-steps, the images which have been identified as refill are removed from the dataset, and the second step is performed.

4.4 Step 2: neutrals identification

At this point, if there aren't too many unidentified refills in the dataset (so the results are at least as high as those of the 2-classes test), identifying neutrals should also be easy, since the 2-classes test gave high results.

However, we can't just be satisfied with those results, because since the errors of each step propagate to the next steps, we risk to get relatively low results in the last step, so we need to study methods to get as accurate as possible in predicting neutrals.

We observed that, when separating neutrals from positives and negatives (non-neutrals), the depth images could give us useful information too. In fact, the main characteristic of non-neutrals is the presence of a stretched arm which, apart from some exceptions, is usually situated in a specific height range. We initially tried to use depth images to better identify refills too, but the differences found between the depth maps of refills and non-refills are far lower that the differences found between neutrals and non-neutrals.

Anyway, we created different datasets preprocessed accordingly, and on each one of them we trained a neural network to produce a different model (the metrics are showed in the following section). The datasets we used are:

- Only RGB images (as in step 1)
- Only depth images
- Depth images with high and low threshold. The values of these thresholds will be tuned in the next section to isolate the arm from the background in as many images as possible. With this preprocessing, theoretically, the precision should improve because the background (that is essentially noise) is removed, and we will verify this in the results section. On the other hand, no matter how finely tuned the chosen range is, in some images the arms will be cut off too (but they could be still recognized using the whole depth image), so the recall would be lower.
- Depth images with only low threshold. Analyzing the dataset, we found that statistically, for some reason, people tend to pick products placed on higher shelves more frequently than those on lower ones, so less arms will be cut off in the low thresholding process. This is a fair compromise between the previous two options.

- RGB images with mask. Essentially we isolated the arm from the background in the RGB images using images from the third option (depth with high and low threshold) as a mask. As for the third option, we expect this model to have a higher precision than the first one, but a lower recall.

As we can see, each option is different depending on whether we want a higher precision or a higher recall. Which one should we choose?

Maybe (depending on the results) we don't need to compromise. The idea is to use all the 5 models in the step 2 of the testing process and to implement a (weighted) voting system. In the process of parameters tuning (which we will see in the following section) we decided that instead of using a majority system, it's better to use a quota system, so if the weighted sum of the models which identified a given image as neutral is equal or higher than a given quota, the image will eventually be classified as neutral, otherwise as non-neutral. The values of both the weights and the quota will be tuned in the next section.

The second step is performed through the following lines of code:

```
prepare_for_preprocessing(reduced_dataset)
pred_rgb = predict_neutral_from_rgb(reduced_dataset, NN_NEUTRAL_RGB)
pred_depth = predict_neutral_from_depth(reduced_dataset, NN_NEUTRAL_DEPTH)
preprocess_thr_low(reduced_dataset)
pred_low = predict_neutral_from_prep(reduced_dataset, NN_NEUTRAL_DEPTH_LOW)
preprocess_thr(reduced_dataset)
pred_thr = predict_neutral_from_prep(reduced_dataset, NN_NEUTRAL_DEPTH_THR)
preprocess_mask(reduced_dataset)
pred_mask = predict_neutral_from_prep(reduced_dataset, NN_NEUTRAL_DEPTH_MASK)
predictions = [pred_rgb, pred_depth, pred_low, pred_thr, pred_mask]
vote(reduced_dataset, predictions, prediction_weights, MIN_NEUTRALS_FOR_VOTING)
reduced_dataset = reduce_dataset(dataset)
```

The voting system is performed by the vote() function:

```
def vote(dataset, predictions, weights, min_neutral_for_voting):
        n = len(predictions)
        i = 0
        for item in dataset:
                num_neutrals = 0
                for index, prediction in enumerate(predictions):
                        if prediction[i] == 'neutral':
                                num_neutrals += weights[index]
                if num_neutrals >= min_neutral_for_voting:
                        item.predicted_class = 'neutral'
                i += 1
```

At this point, only images with arms should be present in the dataset. The only difference is whether the hand contains a product or not.

4.5 Step 3: positives and negatives identification

The 2-classes test for positives and negatives provided the lowest results. Considering that at this point in the test process the overall results for these two classes will be even lower because of the errors propagated from the previous steps, it's critical to use more accurate methods other than the model of the trained neural network.

Theoretically, if we train a neural network using the dataset of RGB images with mask obtained in the previous step, we should get more accurate results, but we also find the same problem of step 2: images where the arms are cut off.

In this step it doesn't make much sense to use a voting system because we can't get a lot of different datasets like in the previous step. The depth maps don't provide useful information, since the depth of the arm is similar to all the images, and even if it wasn't, it couldn't give any information about whether the hand is holding a product or not.

The approach we used (its effectiveness will be validated in the next section) is the following: we split the dataset of masked RGBs in two. One contains "dark" images and the other "light" ones. On the light images we applied the model trained on the masked RGBs, meanwhile the dark ones have been replaced by the original RGBs, on which we applied the original model.

A dark image is an image where the arm has been cut off in the thresholding process, so the image is (almost) completely black. In order to identify dark images, we check if they contain at least a given percentage of dark pixels. A dark pixel is a pixel which intensity is lower than a given number. We don't just average the intensity because the masked images usually have a dark background, so a very little arm could keep the average intensity low and the image will be considered dark. Counting the dark pixels, however, allows us to tell if an image is actually black and "light" elements (like an arm) have been cut off.

This approach will be validated in the next section, where the parameters used to identify dark images (minimum percentage and maximum intensity) will be tuned too.

The second step is performed through the following lines of code:

```
preprocess_mask(reduced_dataset)
dark_dataset, light_dataset = split_dark_light(reduced_dataset, \
DARK_THRESHOLD, DARK_RATIO)
predict_posneg(dark_dataset, NN_POSNEG)
predict_posneg_mask(light_dataset, NN_POSNEG_MASK)
```

The split_dark_light() function performs the splitting between dark and light images:

```
def split_dark_light(dataset, dark_threshold, dark_ratio):
        dark = []
        light = []
        for item in dataset:
            img = item.get_masked_img()
                height, width = img.shape
                i = 0
                for row in range(height):
                        for col in range(width):
                                color = int(img[row, col])
                                if color <= dark_threshold:
                                        i += 1
                x = (height * width) * (dark_ratio / 100)
                if i >= x:
                        light.append(item)
                else:
                        dark.append(item)
        return dark, light
```

4.6 Sequence splitting and classification

Now that a class has been associated to every frame, we can use such predictions to identify each interaction as a whole, associating a class to a sequence of frames in the ways we saw earlier.

We have already split the dataset in sequences, now we can classify them through the following code (where sequences is a list of tuples containing the sequence itself and the predicted class, initially set to None):

```
for sequence, prediction in sequences:
        prediction = predict_sequence(sequence)
```

The function predict_sequence() is defined as following, classifying each sequence accordingly to the types of interactions as described in Section 3:

```
def predict_sequence(sequence):
        first = None
        last = None
        for item in sequence:
```

```
            if item.predicted_class == 'refill':
                    return 'refill'
            if item.predicted_class == 'positive':
                    if first == None:
                            first = 'positive'
                    last = 'positive'
            elif item.predicted_class == 'negative':
                    if first == None:
                            first = 'negative'
                    last = 'negative'
    if first == last:
            return 'neutral'
    return last
```

We can notice that this is an "exact" method that doesn't uses learning approaches or estimations of any kind (it doesn't involve free parameters to be tuned), because it only relies on the definitions we gave in Section 3. That means that we don't need to define metrics to evaluate the classification, but only the dataset splitting. Unless, of course, if the definition itself of a refill/neutral/positive/negative sequence is changed, because the one we gave before doesn't cover every possible case (if a customer begins an interaction with an empty hand and ends it with a full hand, it doesn't always mean that he purchased a product).

5 Parameters tuning and final results

5.1 Intermediate Results

In the previous section we said that we trained new 2-classes networks to improve the results of step 2 and step 3, here we will show the results these networks gave us.

To try to improve the results obtained with neutral/non-neutral, we tried out AlexNet network also with depth frames (Table 8) and we obtained:

2 classes: Neutral/Non-Neutral					
Train					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.3378	0.8519	0.8519	0.8519	0.8519
Validation					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.2766	0.8994	0.8994	0.8994	0.8994

Table 8: Neutral/Non-Neutral Results on Depth frames and balanced Dataset.

Other results are about adding threshold (Table 9) and low threshold (Table 10). With threshold we tried to blacken the shelf while with low threshold also the lower part of the image, like the floor.

2 classes: Neutral/Non-Neutral					
Train					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.0204	0.9926	0.9926	0.9926	0.9926
Validation					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.5730	0.9013	0.9013	0.9013	0.9013

Table 9: Neutral/Non-Neutral Results on Depth frames, threshold and balanced Dataset.

2 classes: Neutral/Non-Neutral					
Train					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.0136	0.9955	0.9955	0.9955	0.9955
Validation					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.6356	0.8974	0.8974	0.8974	0.8974

Table 10: Neutral/Non-Neutral Results on Depth frames, low threshold and balanced Dataset.

Another experimentation on neutral/non-neutral was done using masks. In particular the customer figure in each frame has been cut and used as a mask to be applied to the relative RGB frame (Table 11):

2 classes: Neutral/Non-Neutral					
Train					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.0039	0.9984	0.9984	0.9984	0.9984
Validation					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.5666	0.9087	0.9087	0.9087	0.9087

Table 11: Neutral/Non-Neutral Results with Masked frames and balanced Dataset (Without Preprocessing).

Also with the positive/negative class we tried to apply the mask to the frames, considering that in the previous case (neutral/non-neutral) it turned out to be one of the best methods (Table 12):

2 classes: Positive/Negative					
Train					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.0053	0.9980	0.9980	0.9980	0.9980
Validation					
Networks	**Loss**	**Accuracy**	**Precision**	**Recall**	**F1Score**
AlexNet	0.8479	0.8814	0.8814	0.8814	0.8814

Table 12: Positive/Negative Results on Masked frames and balanced Dataset.

In this case (Positive/Negative) the masked frames improve the results previously obtained, bringing the accuracy on validation set from 83% to 88%.

5.2 Parameters tuning

In the previous section we named some free parameters to tune in order to get the best possible results. In order to choose the best parameters, we should have a bigger test set (or multiple ones). Since our dataset is not very large, and a big part of it has been used for the training, we decided that parameters tuning could be one of the future development of this work, but we still tried to improve the results as much as we could by tuning the parameters as we will show soon.

```
MAX_TIMESTAMP_DIFF = 60000          #average time of refill operation
MAX_TIMESTAMP_FRAMES_DIFF = 3000    #not many things happen in 3 seconds...
MIN_REFILL_LENGTH = 5
MIN_REFILL_RATIO = 0.2              #relatively low because of refill images
                                    #without a box

MIN_NEUTRALS_FOR_VOTING = 2
PREDICTION_WEIGHTS = {'rgb': 1,
                      'depth': 0,   # worse precision on the test set
                      'low': 1,
                      'thr': 1,
                      'mask': 2}    # best precision on the test set
                                    #(but low recall)
DARK_THRESHOLD = 40
DARK_RATIO = 98
```

This tuning of the parameters allowed us to get the following results.

5.3 Final results

The following are the results of Step 1 alone, we won't compare it to the 2-classes refill test because, as we repeatedly said, it has been labeled differently.

Refill				
	Test			
Class	**Accuracy**	**Precision**	**Recall**	**F1Score**
Refill	0.9744	0.9710	0.9710	0.9710

Table 13: Step 1: Refill.

The following are the results of Step 2 alone, compared to the 2-classes neutral test. We can see that the voting system improved the precision.

Neutral				
	Test			
Class	**Accuracy**	**Precision**	**Recall**	**F1Score**
Neutral (Step2)	0.9611	0.9301	0.9774	0.9531
Neutral (2-classes RGB)	0.9269	0.8536	0.9887	0.9162

Table 14: Neutral comparison.

The following are the results of Step 3 alone, compared to the 2-classes positive/negative test. We can see that the use of the masked images didn't improve the 2-classes results.

Positive/Negative				
	Test			
Class	**Accuracy**	**Precision**	**Recall**	**F1Score**
Positive (Step3)	0.8965	0.8559	0.9224	0.8879
Negative (Step3)	0.8965	0.9338	0.8758	0.9039
Positive (2-classes RGB)	0.9003	0.8629	0.9224	0.8916
Negative (2-classes RGB)	0.9003	0.9343	0.8827	0.9077

Table 15: Positive/Negative Comparison.

Finally, the following are the overall results of the 3-step methodology:

Final Results				
	Test			
Class	**Accuracy**	**Precision**	**Recall**	**F1Score**
Refill	0.9744	0.9710	0.9710	0.9710
Neutral	0.9579	0.8956	0.9209	0.9080
Positive	0.9489	0.8064	0.8620	0.8333
Negative	0.9579	0.9242	0.8413	0.8808

Table 16: Final Results.

Final Comparison				
	Test			
Methods	**Accuracy**	**Precision**	**Recall**	**F1Score**
Shopper Analytics (best NN)	0.7931	0.8291	0.8291	0.8291
4 classes (best NN)	0.8731	0.8768	0.8720	0.8743
Final Results (classes average)	0.9597	0.8993	0.8988	0.8982

Table 17: Final Results.

These results are higher than the Shopper Analytics' ones, even considering one more class. But we need to make a quick note on Step 3, that we left as it is, without replacing it with the

2-classes model which gave better results (so the overall results would be even higher). We did this because in most images of the dataset there is a time shift between the RGB images and the corresponding depth images due to some defects of the cameras, which didn't allow us to use the mask correctly. The networks trained on the masks, however, gave higher results on the validation set, because the images in it didn't have this problem. Of course we couldn't use the images in the validation set for the test set, so the methodology should be tested again on a correct test set to effectively compare Step 3 with the 2-classes results.

6 Conclusion and further developments

In this work, by using a 3-step approach, preprocessing, voting and sequences identification, we managed to improve the previous results. We also introduced a simple method to identify sequences, based on the timestamp of the frames, and to classify them.

Further developments of this project include a better approach to split and classify sequences, and the definition of a method to evaluate them. Defining metrics can be difficult because the sequences are not defined a priori, but during the testing process itself. Anyway, a first simple idea to define metrics could be the following:

1. Manually split the sequences in the test set and label them.
2. Add a second label to each image of the dataset, which will be the same as the corresponding sequence, so for example each image of a positive sequence will have the second label set to positive.
3. Do the same for the predicted sequences, i.e. every image of a predicted sequence will have a second predicted class which will be the same as the sequence.
4. Calculate the metrics as before on the images, this time using the second label and the second predicted class.

There should be also a better definition of the types of interactions, because as we said earlier, for example, if a customers begins an interaction with an empty hand and ends it with a full one, it doesn't always mean that he purchased a product.

Talking about sequences, the splitting operation (which for now uses only the timestamps) can be improved, for example using long streaks of neutrals as separators, such as images without people in them, or very similar images in a row.

Future works should include a better tuning of the parameters defined in this paper, but in order to do that a larger dataset should be used, both for the training and the testing. The new dataset should also contain images without the time shift between RGBs and depth maps, as we mentioned earlier. Then the testing process should be tested again and the Step 3 results should be compared with the 2-classes ones.

References

[1] E. Frontoni, A. Mancini and P. Zingaretti, "RGBD Sensors for human activity detection in AAL environments", Living Italian Forum 2013 Longhi, S., Siciliano, P., Germani, M., Monteriu, A. (Eds.), 300 p. 50 illus. Available Formats: eBook ISBN 978-3-319-01118-9, Due: July 31, 2014.

[2] Emanuele Frontoni, Adriano Mancini, Primo Zingaretti, Valerio Placidi, "Information management for intelligent retail environment: the shelf detector system", Information, vol. 5, no. 2, pp. 255-271, 2014.

[3] Hafedh Ibrahim, Faouzi Najjar, "Assessing the effects of self-congruity, attitudes and customer satisfaction on customer behavioural intentions in retail environment", Marketing Intelligence & Planning, Vol. 26 Issue: 2, pp.207-227, 2008.

[4] M. Sturari et al., Robust and affordable retail customer profiling by vision and radio beacon sensor fusion, Pattern Recognition Letters (2016).

[5] T. Ko, Co. Raytheon and Va. Arlington, "A survey on behavior analysis in video surveillance for homeland security applications", Applied Imagery Pattern Recognition Workshop, 2008. AIPR '08. 37th IEEE, pp. 1-8, 15-17 Oct. 2008.

[6] M. Cristani, R. Raghavendra, A. Del Bue, and V. Murino, "Human behavior analysis in video surveillance: A Social Signal Processing perspective", Neurocomputing vol.100 Special issue: Behaviours in video, pp. 86-97, 16 January 2013.

[7] A. Ascani, E. Frontoni, A. Mancini and P. Zingaretti, "Feature group matching for appearance-based localization", IEEE/RSJ 2008 International Conference on Intelligent RObots and Systems IROS 2008, Nice, 2008.

[8] C. Desai, D. Ramanan, and C. Fowlkes, "Discriminative models for static human-object interactions", Computer Vision and Pattern Recognition Work-shops (IEEE/CVPRW), pp. 9-16, 2010.

[9] T. Brox, L. Bourdev, S. Maji and J. Malik, "Object segmentation by alignment of poselet activations to image contours", International Conference on Computer Vision and Pattern recognition IEEE/CVPR, pp. 2225-2232, 2011.

[10] Liciotti, D., Contigiani, M., Frontoni, E., Mancini, A., Zingaretti, P., & Placidi, V. (2014, August). Shopper analytics: A customer activity recognition system using a distributed rgb-d camera network. In International workshop on video analytics for audience measurement in retail and digital signage (pp. 146-157). Springer, Cham.