

ID2012 - Ubiquitous Computing

Final project report

# Ubiquitous Shopping



Course Coordinator

Fredrik Kilander

Óttar Guðmundsson

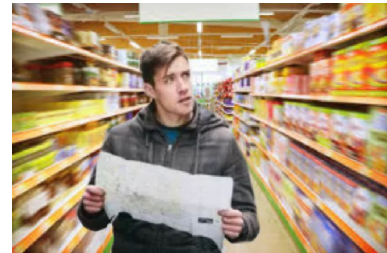
Örn Arnar Karlsson

2018 - 5 - 8

# Idea and motivation

It's Friday night and you are on your way to your friend to watch the game which starts in 10 minutes. On your way your friend asks you to grab snacks, soda, dipping and some frozen pizza in supermarket close to his street. In this scenario, time is of the essence since you don't want to miss the game. Thus you are faced with a few problems

- Where is that supermarket located?
- Where are the products that you are looking for located in that shop?
- What if the store is out of or just doesn't have the products that you are looking for?



We can quickly see that frustration and stress is met as you rush through the unknown supermarket looking for the products needed. What if there was a system that could reduce the time spent in store by providing information about the products close to you in real time?

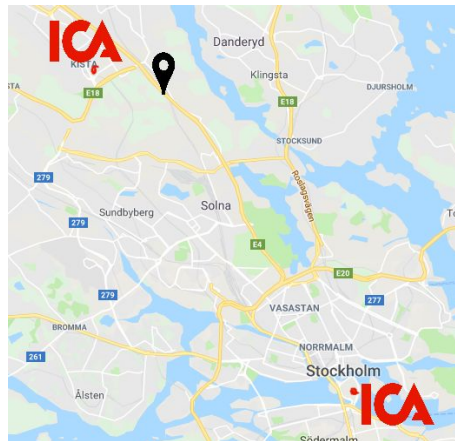
Our project, codenamed "Ubiquitous Shopping" attempts to demonstrate a prototype version of such a software by using geotags for products inside a supermarket. The software focuses on combining modern sensors which most people are equipped with on their daily bases to improve user experience, both for store owners and potential customers. In the context of UC, this project is highly relevant since the techniques used is extremely dependant on mobile sensors such as location and camera that works across multiple devices. They provide the core functionality of the system but is greatly enhanced by real time data provided by other users of the system. For example, customers can report items missing and store owners try to sell extra products on the way out as you pass through the store. The combined data then delivers a generated context that is really useful but only when they act towards each other.

## Combination of data dimensions

For this project, we thought of multiple dimensions that could be combined to generate a 2D map of the shortest path through the closest store. The dimensions of data would be combined with others through a streamed pipeline, where former dimensions would affect those that appeared later in the pipeline. The dimensions used can be classified into the following:

## Location

Location of the user is always used, being either a customer or store owner. It's relevant for the user since we want to bring him the closest store that is available to him and only allow him to update items missing if he is in that particular store. The store owner's location works the same way, he doesn't have to select which store he wants to update since we can automatically determine what store he is currently in.



*Demonstration of available fictional ICA stores in Stockholm*

The original plan was to store the item location based on where the owner was staying but the inaccuracy of the devices could vary up to 2-3 meters so we didn't implement it that way. However, this could be achieved using RFID tags inside the store or bluetooth technology.

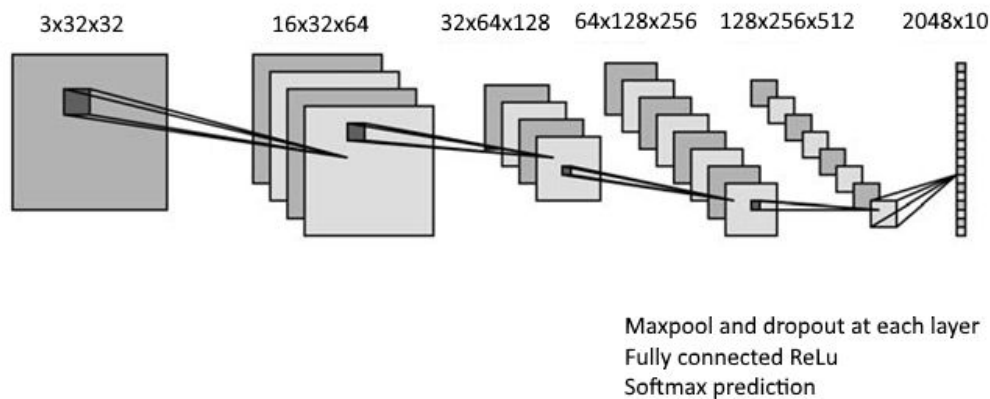


*RFID tags could be placed within the store to get more precise location.*

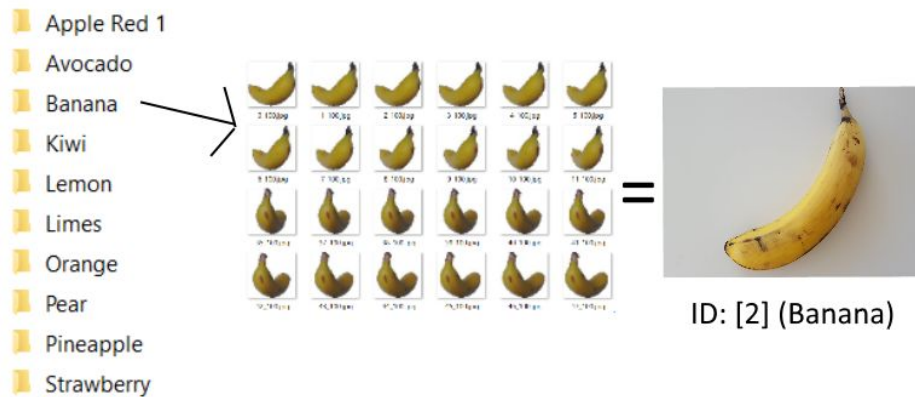
## Image classification

A Convolutional Neural Network written in Tensorflow was used to recognize 10 different products that were available within the system. The dataset used had a 360° view of different fruits with around 500 images per fruit. Each image was rescaled to 32x32 to reduce the computational power needed to classify an image. The network itself was

trained with 0.001 adjusted learning rate for 5000 epochs and batch normalization. Using early stopping via cross validation the network achieved 95.3% accuracy on these 10 classes. The network was pretty standard, using dropout and max pooling at every convolutional layer and finally fully connected ReLu activation layer at the end with softmax activation for the correct prediction. All weights were initialized with Xavier and used biases as well.



*The network architecture*

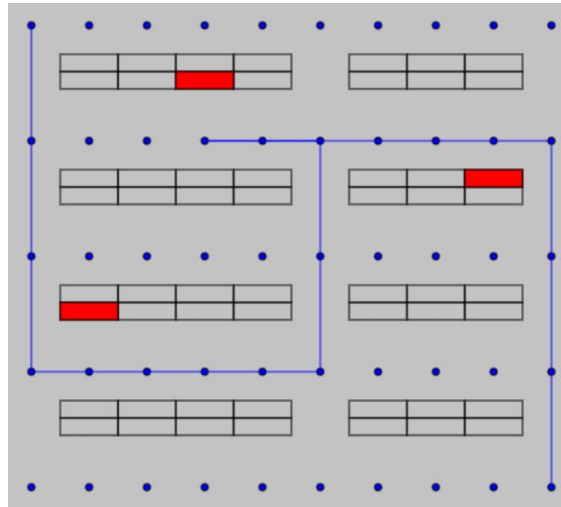


*Demonstration of the dataset used and the correct prediction of an image taken on our mobile phone.*

When the network had finished training the weights were then saved as an ckpt. file that could then be loaded by a web service to predict a newly saved image to a specific folder. On a side note, barcodes of products could also have been used here to determine the product. We wanted to implement a more general and user friendly approach to determine the object so the store owner wouldn't need to find the barcode and scan it.

## 2D graph of nodes

To be able to give the user a map that he could navigate through, each store had to be saved in a database. All stores had information on how the racks were positioned and how big they were as well as a navigational map that held information of connected nodes. Each item was then paired to a selected rack so by selecting products, a short path could be generated throughout the store that visited all the nodes starting from the entrance and leading to it's exit.



*A map of a store, the racks, the location of must visit nodes and all possible nodes.*

## User input

The map that is going to be generated depends on what the user needs. Thus the navigation will always depend on the user preferences. Also, if a user notices some of the products that he is looking for isn't there anymore, he can notify the system that it is missing. Then the next user that will be looking for the same product will be notified that someone did report it missing so he might want to skip the risk of going there. The store owner will also be notified, so he can then either refill on the supplies and remove the missing tag of the product or he can confirm the item is indeed missing, so upcoming consumers will be notified.

### Missing items?

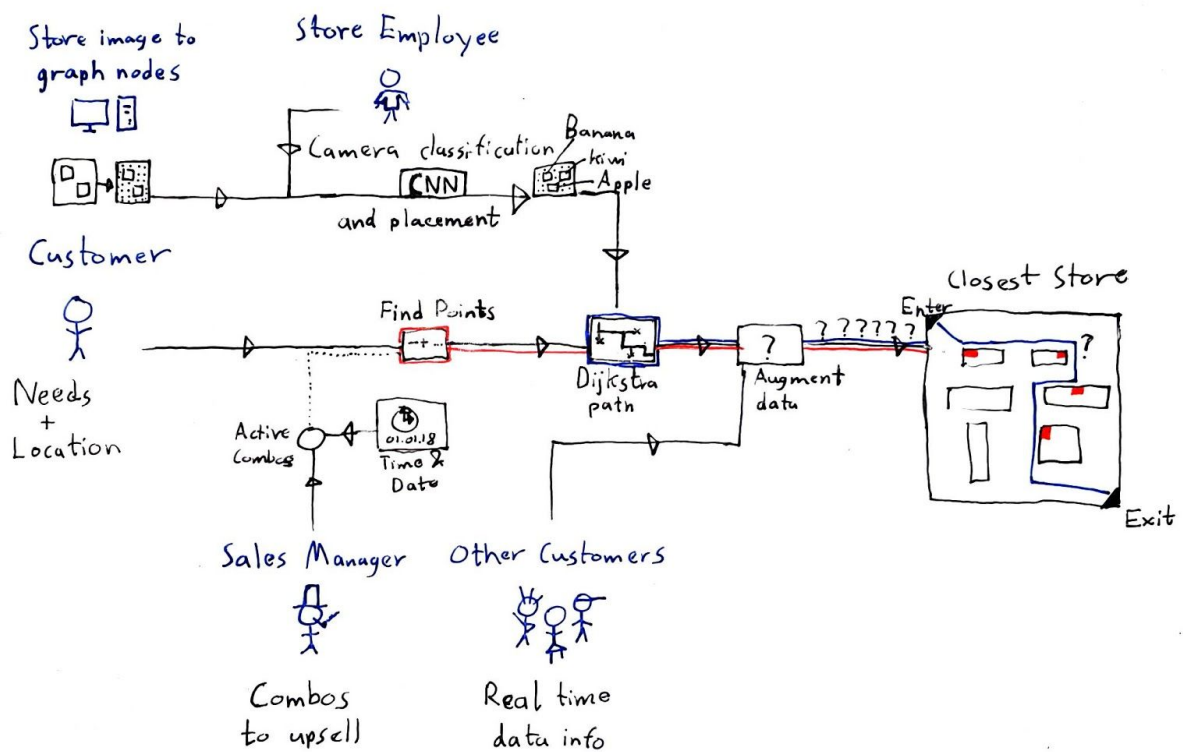
Tell us which item is missing and our staff will check it out as soon as possible

*At the bottom left users could notify the store they were in that items were missing*

## Time and date

Finally, time and date would also affect the results of the generated context. Shops could have different opening times and upsell combos could be activated by different times and dates. For example, if a store is open late at night on weekends and the customer is looking for soda, snack and pizza, maybe the store owner would like to sell him dipping as well. But if the customer is up early and is buying salad and carrots on a Monday, why not promote some avocados too?

When all of this comes together, we can see a diagram of the data dimensions and how they act towards each other to create the final context.



*A classification model diagram that displays the final context*

# Parts implemented

We set ourselves the goal of providing a real time working solution in the presentation of the project. Even though the core functionality of this project was using camera, classification and location, other software components such as framework, databases and interface were required to establish a working solution. The following list displays what was implemented

## Software used

- **ASP.Net C# Website**<sup>1</sup> Used Visual Studio to develop a simple website in C# with two pages. One of them created a simple interface for a customer to look for products while the other one had an interface for a store owner.
- **IIS Manager 7.5**<sup>2</sup> Hosted the website on our computers so external devices such as mobile phones and other computers could run and test the software.
- **SQL Database**<sup>3</sup> Managed all of the data used by creating multiple tables that had some sort of relation to each other. Information like where stores were located, the drawings of each store, the locations of items and so on. The database was hosted on GearHost to separate logic and data.
- **Telerik ASP.Net Ajax**<sup>4</sup> Library used for creating lightweight controls for the interface to deliver smooth user experience via ajax mechanics for the website.
- **FLASK server for web services**<sup>5</sup> Used the simple FLASK library for creating a single web service that could host the trained Tensorflow model.
- **Tensorflow**<sup>6</sup> The Google machine library was used to develop a fully connected dropout Convolutional Neural Network for image classification.
- **Fruit 360-dataset**<sup>7</sup> A dataset from Kaggle that holds around 500 images for 30 different classes of fruits, where each image represents an angle of a fruit.
- **Google Maps API**<sup>8</sup> Got an API key to access Google Maps services to get the location of each device to determine the store that was going to be used in each context.

---

<sup>1</sup> <https://www.asp.net>

<sup>2</sup> <https://msdn.microsoft.com/en-us/library/bb763170.aspx>

<sup>3</sup> <https://www.microsoft.com/en-us/sql-server/>

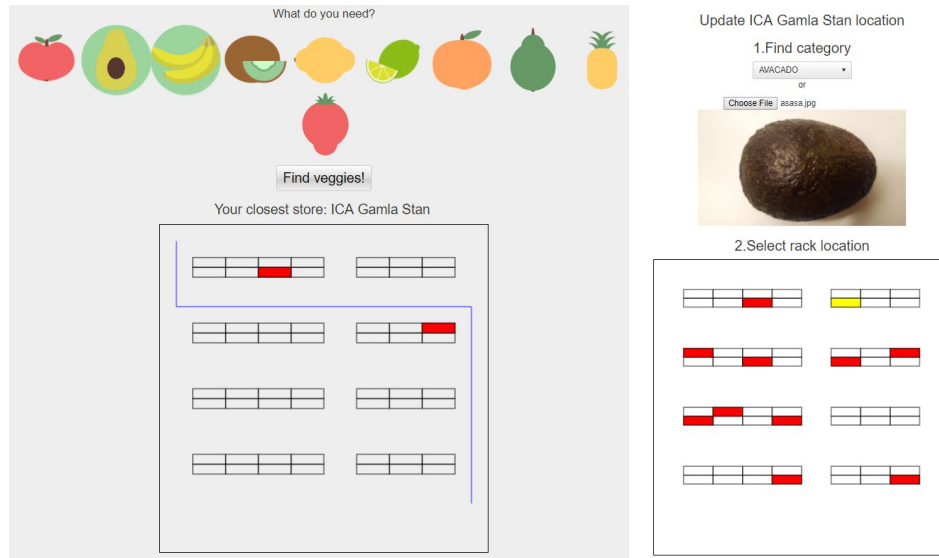
<sup>4</sup> <https://demos.telerik.com/aspnet-ajax/>

<sup>5</sup> <http://flask.pocoo.org>

<sup>6</sup> <https://www.tensorflow.org>

<sup>7</sup> <https://www.kaggle.com/moltean/fruits>

<sup>8</sup> <https://developers.google.com/maps/documentation/javascript/>

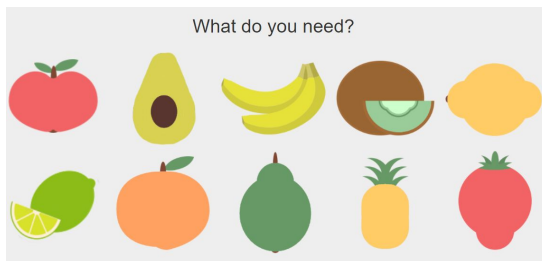


*Left - Interface for the customer*  
*Right - Interface for the store owner*

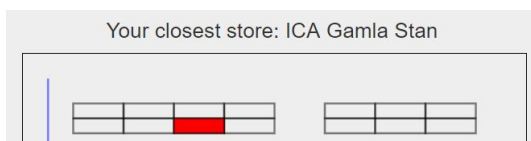
## Parts implemented

### ***For the customer***

- *Gives you a list of available groceries*  
 Our solution allowed the selection of 10 basic fruits that can be

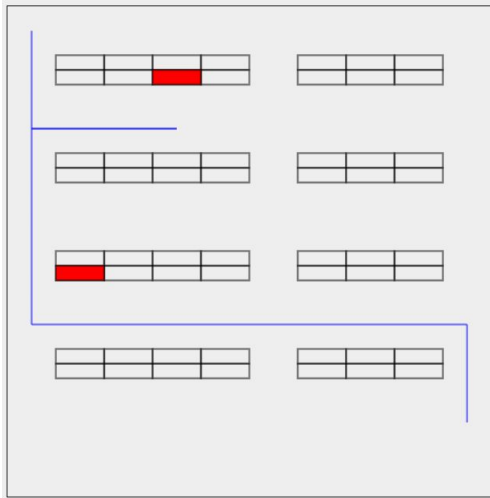


- *Takes you to the nearest shop*  
 When the user has selected the products he needs, the system will locate the nearest store.



- *Shows you the location of groceries, giving the 'shortest' shopping path*  
 The selected products are displayed in a red squares, resembling a rack where a product is located. The Dijkstra algorithm then generates the shortest path that visits all of those racks starting from the entrance to the exit. Here we can see the location of Banana and Kiwi.





- *Offers another store if the nearest one is missing any items*

If an item is reported missing a warning label will be displayed on top of the map. A button will also appear that allows the customer to locate the closest store that surely has all of those items.

Warning - someone marked KIWI missing so maybe it's not available

Show me the closest store that has all items

- *Report an item missing*

If a customer notices some items missing from the store, he can notify the system to warn future customers about it and let the manager know.

## Missing items?

Tell us which item is missing and our staff will check it out as soon as possible

Kiwi

Let us know

### For the store manager

- *Automatically gives you your shop based on your location*

When a store manager opens up the interface to update a shop, he will automatically have the closest store selected with the items that are available populated for selection



- *Update product location*

The store employee can update the location of products inside the store by either selecting a product from a list or taking a picture of it which will be classified. After doing so he can select a new rack which will be colored yellow and finally press the update button that saves a new location for that product.


1.Find category

AVACADO ▼

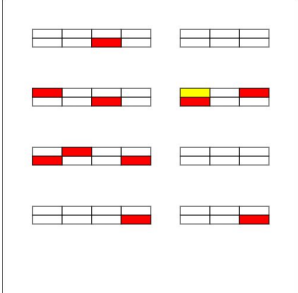
or

Choose File

asasa.jpg



2.Select rack location



### 3.Update the store

AVACADO location successfully updated in ICA Gamla Stan

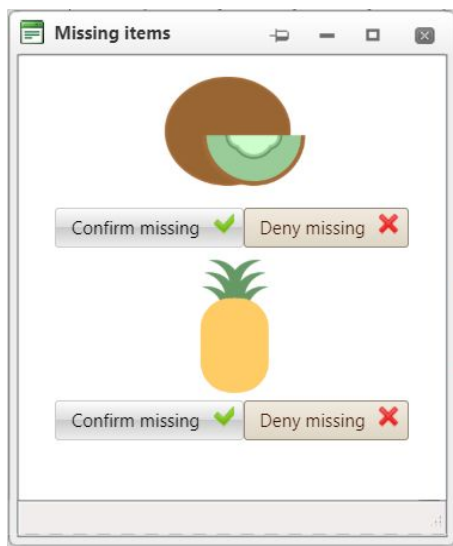
Update Store

- *List of reported missing products*

If some items have been reported missing in the store that the manager is currently active in, a button in the bottom left corner will appear



When this button is clicked, a small dialog opens where he can confirm that items are indeed missing or not. The context will then be updated for future customers.



- *Create upsell combos*

The manager can edit the upsell combos by clicking a button centered at the bottom which opens up a popup window. There a combo can be created and activated in a given time period and older combos can also be deleted.



## What we didn't implement (only theory)

- *Image of store converted to a graph with nodes.*

Even though we found some solutions that could create a 2D map of a graph network we didn't implement it since the demonstration only required two stores. For that reason, each store with its racks and graphs was manually drawn and inserted into a database.

- *Perfect latitude and longitude coordinates of each item.*

The original concept of the project was to use the current latitude and longitude of the mobile device to register the location of the sensed product. After a considerable amount of time spent developing we learned that the accuracy of the location is not as great as one can hope for. So instead we used the location to determine the active store that was being used and gave the store owner a map where he could pinpoint the rack where the product was placed. This can however be done by using RFID tags and bluetooth devices inside of the store and writing a small software that would pinpoint the location of the device used by calculating the distance to each RFID.

## Datatables

A small view of the datatables used for this project. STORES held information about the active stores in the system where the RACKS and NODES had the unique id of a store as a foreign key. Items were then joined on both the RACKS and the STORE id while the upsell combos were only matched by items. Upsell combos could have been linked to individual stores but since the project focused on one company in the beginning, we decided that combos applied for all stores simultaneously.

### STORES

ID	int
X	varchar(50)
Y	varchar(50)
NAME	nvarchar(50)
ADDRESS	nvarchar(50)
COUNTRY	nvarchar(MAX)

### ITEMS

ID	int
RACK_ID	int
STORE_ID	int
NAME	nvarchar(50)
PRICE	money
STATUS	nvarchar(50)

### RACKS

ID	int
STORE_ID	int
NODE_ID	int
X	int
Y	int
WIDTH	int
HEIGHT	int

### UPSELL

COMBO_ID	int
PROD_NAME	nvarchar(100)
DATE_TO	date
DATE_FROM	date
TIME_FROM	time(7)
TIME_TO	time(7)
ACTIVE	int
UPSELL_PRICE	float

### NODES

ID	int
STORE_ID	int
X	decimal(18, 4)
Y	decimal(18, 4)
ADJACENCIES	nvarchar(50)

## Future work for improvement

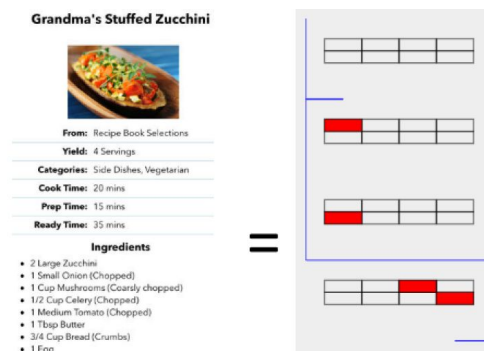
We discussed further work on the project that might improve its final generated context. Most of these parts could have been implemented but would require a lot more time and we decided that since all of them were simply an improvement, we classified them only as nice to have rather than vital components.

### Accurate location sensing indoors

As said before, RFID tags and Bluetooth technology placed inside the store could pinpoint the accurate location of the mobile user to give relative location based on a 2D graph map of the store. The cost of setting up such a system for each store would require funding and be more complex operation. This adds undoubtedly to the relation of the project to Ubiquitous Computing since the user wouldn't even have to think about where to place the location of the products he wants to sell.

### Recipe classification

Delicious recipes are often the key to an impressive dinner party but they most often incorporate new materials and products that the user doesn't know. A shopping plugin for recipes on websites could work in such a way that one can simply find a recipe online and click on a "Ubiquitous button" that would read the contents of the recipe. Then it would map those recipes to the closest store and give him the shortest path through the store.



*Clicking on a recipe button could give you a navigation of the required materials in the closest store to the user.*

### Project Tango

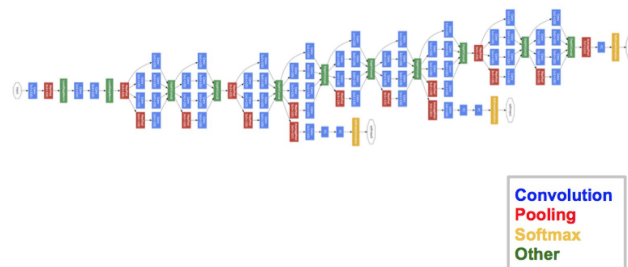
Google is has been working on a mobile device called Project Tango<sup>9</sup> that can sense and map a 3D environment to a 1:1 scale model. Using such a device could really easy the mapping of the store for new store owners to create an accurate 3D model of their store. With that, customers could even download the model on their way to the store to get an

<sup>9</sup> <https://www.youtube.com/watch?v=Qe10ExwzCqk>

exact navigation in a virtual environment. As of 1st March 2018, the project has been abandoned to focus more on its predecessor ARCore.

### Better CNN

Our Convolutional Neural Network was really basic and only classified only 10 out of 30 products available. The average store has so much more to sell that the net would need to be improved by a landslide. The solution to this problem would be to have a way better model that required more data. An example of this would be something like Google's GoogLeNet<sup>10</sup> that achieved around 96% accuracy on the CIFAR-1000 dataset.



*The architecture of the GoogLeNet*

### Collaborative filtering on sales data

Instead of letting store owners create combos by themselves, a history of shopping lists could be studied to see what products are being bought together and make sure that users does at least see them on their way out. If we had the data of user shopping history (from ICA membership cards for example) then these preferences could be cross matched with users with similar taste buds!

## Characteristics for HCI evaluation

When all of these techniques and architecture is put together, not only do we see a working product but a highly relevant course project. To further argue the strong connection of the project to the main aspect of this course, we can compare them to the bold characteristics of the HCI evaluation.

In terms of context awareness the system can easily adapt to any store. It knows which store it is currently updating and can classify a product from an image of the environment independent of its user. Thus, it adapts to the user accessing the software and updates the context based on its user.

---

<sup>10</sup> <https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf>

In terms of mobility the system works as long as it can access the location of the device. If a user is walking around with either its laptop, mobile or a tablet each device will automatically adjust itself based on the closest store.

In terms of transparency its not strong since the software focuses on user interacting with a device in all cases.

In terms of attention it's not really for the customer since he has to look at the device to know where he is going. This could be more relevant if the software talked back to the person like Google Maps does by telling the person to turn left or right based on its indoor position. But for the store manager itself its really relevant since he focuses on simply taking pictures of his products rather than finding their names inside a database.

In terms of calmness the software interacts with the user by only showing him relevant information and the info that he wants to see. He is not being bombarded with ads or irrelevant information, but instead guided smartly unknowingly passing targeted selling points.

## Conclusion

Since we ended up creating and delivering a working solution for a real time problem that both of us encounter quite frequently, we agreed that this project was a success. Even though it was a little bit of letdown how inaccurate the location of GPS indoors was we managed to adapt to the problem by thinking of how this could be implemented with more funding. But we managed to find an alternative solution to our problem that fulfilled the aspect of the course, that is, applying multiple sensors of data without using such an accurate sensing of the location dimension. This project also gave us the idea of problems that might rise when working with real time data and how multiple dimensions have an effect of the different versions of the generated context. All in all, we are happy with the final outcome of the product.



---

Óttar Guðmundsson

---

Örn Arnar Karlsson