Treball de Fi de Grau

GRAU D'ENGINYERIA INFORMÀTICA

Facultat de Matemàtiques i Informàtica

Universitat de Barcelona

Personal Photo Organization

Stefan Lönn

Directora: Mariella Dimiccoli

Realitzat a: Departament de Matemàtica Aplicada i Anàlisi

Barcelona, 22 de Juny de 2016

Abstract

We live in a society were a large portion of the population has, or has used a smartphone or/and a digital camera. In addition, hard drives and cloud storage are getting cheaper and cheaper, leading to a tremendous growth in stored personal photos. Organizing manually such large collections of photographs would be unfeasible. To solve the need of organizing such a large collection of personal photos automatically, we propose a system that first estimates the user profile, and then classifies the photos in categories related to the estimated profile. To estimate the user profile, we adapt a topic discovery method used in document analysis, called Probabilistic Latent Semantic Analysis (pLSA). Based on the estimated profile, we provide a personalized photo classification, using a set of pretrained Convolutional Neuronal Network (CNN). In this manner, we achieve an efficient, scalable, flexible and more tailored organization of the user's photos.

Summary

INTRODUCTION

With the decrease in price of both hard-drive and cloud storage, accompanied by the ever-growing smartphone sector, we are taking more photos than ever. Organizing such a large collection can be a daunting task, and while the current solutions, such as Google Photos [1] or Eden Photos [2], have partially alleviated the problem, they are generally very generic in their classification. For example, Google Photos provides a classification based on two main topics: Places and Things, and these topics are then divided into a mix of sub-categories. This means we can find sub-categories such as ‘Airplane’ next to ‘Forest’. On the other hand, Eden Photos classifies the user’s photos into 13 broad topics, such as ‘Animals and Pets’. Therefore, photos of cats will appear next to photos of horses, etc. As mentioned, both solutions are quite generic, hence, we propose a method to decrease this generalization, and provide a more fine-tuned classification, tailored to the user. Our proposed system is geared towards smartphone photos, as they usually englobe a large set of different topics with no specific order. Photos made with a professional camera will usually share a topic and will most probably already be organized, as a professional camera is usually used for a purpose, unlike smartphones, which capture more day-to-day events.

As a rule of thumb, everybody has a topic or ‘theme’ he/she particularly enjoy, being it sports, food, nature, etc. With this assumption, we can be certain that the majority of a user’s photos will be of his topic of interest, with its specific sub-topics. For example, an ‘adventurous’ user who likes to take photos while hiking, and will have a large sample of his collection based on landmarks that define nature, such as rivers, valleys or waterfalls. Hence, a system able to first estimate the topic(s) of interest of the user, would be able to propose the most appropriate set of topic-related categories. This would result in a more tailored classification, adapted to each user.

To get an estimation of the user-profile, we will adapt a topic discovery method used in document analysis: Probabilistic Latent Semantic Analysis (from now on, pLSA) [3]. This algorithm discovers latent topics of a document, and generates *K* topics with *N* top-words per topic. Once we have an estimated user-profile, we can proceed with the personal classification of the users’ images. To classify the photos into sub-categories, we will use the deep learning framework *Caffe* [4]. This framework enables us to load and train Convolutional Neuronal Networks (from now on, CNN), and more importantly, it allows us to apply *fine-tuning*, a process where we load a pre-trained CNN and modify the weights through training in order to achieve our goal.

We will classify all the images with their respective topics, even those that are considered outliers in respect to the user-profile. For example, if a user gets an estimation that 70% of his photos are of ‘Nature and Landscape’, and the rest ‘Architecture and Street view’, we will classify both groups with their respective CNN counterpart into the corresponding set of sub-categories. Here lies one of the main advantages of our system: it is flexible. We can increase the number of topic-specific categories that our system can identify, just by fine-tuning the corresponding network of the topic. Instead of using a single CNN that tries to classify a large number of classes, we can use various networks that are specialized in one topic, and offer an improved classification. The same idea can be applied with the user-estimation. We can add more topics, such as Sports and Adventure by training pLSA with more users.

With this approach, we aim to achieve a more precise and flexible organization system by providing more labels of the specific topics that the user likes, instead of trying to apply a generic classification, with no regard to the preference of the user.

To achieve our goal of defining a flexible classification system, we have defined the following main objectives:

- Create a dataset for pLSA training.
- Understand pLSA and find appropriate parameters.
- Automatically assign topics to pLSA output through semantic similarity using the Natural Language Toolkit.
- Understand Caffe Deep Learning Framework.
- Define sub-categories for each topic.
- Create training, validation and test set for each sub-category.
- Find appropriate pre-trained models for each topic.
- Validate our fine-tuning process.
- Define the validation protocol by the user studies.
- Evaluate final system through user studies and quantitative evaluation.

In addition, many subjects during my career have been helpful in the development of this project. For instance, *Computer Vision* introduced various image recognition techniques, and provided vital knowledge to better understand this project. On the other hand, the subject *Image Processing* taught us how to apply filters and transformations, which are used extensively throughout the pre-processing phase of the images in our CNNs. For example, our dataset augmentation was made possible thanks to the lessons taught in this subject.

Technologies used

Throughout this project, we will use an array of different tools and frameworks. We will use **Caffe** to fine-tune and monitor our models. **Python 2.7** will be used for a large part of the project, either for scripts to tag and extract tags from the images, creating and managing our datasets, and creating and running Caffe throught **PyCaffe**. We will use libraries such as **numpy** and **matplotlib** to efficiently create and modify arrays, and visualize plots in an intuitive fashion. Most of our code will be made using **Jupyter Notebook**, which allows us to apply small changes to scripts without the need to run the whole script again, saving valuable time. Occasional **bash scripts** were used to move or delete large amounts of images, as it was the fastest option. To visualize the results, **HTML, Javascript** and **Bootstrap** will be used, to provide a clean and simple interface.

1. PLANIFICATION

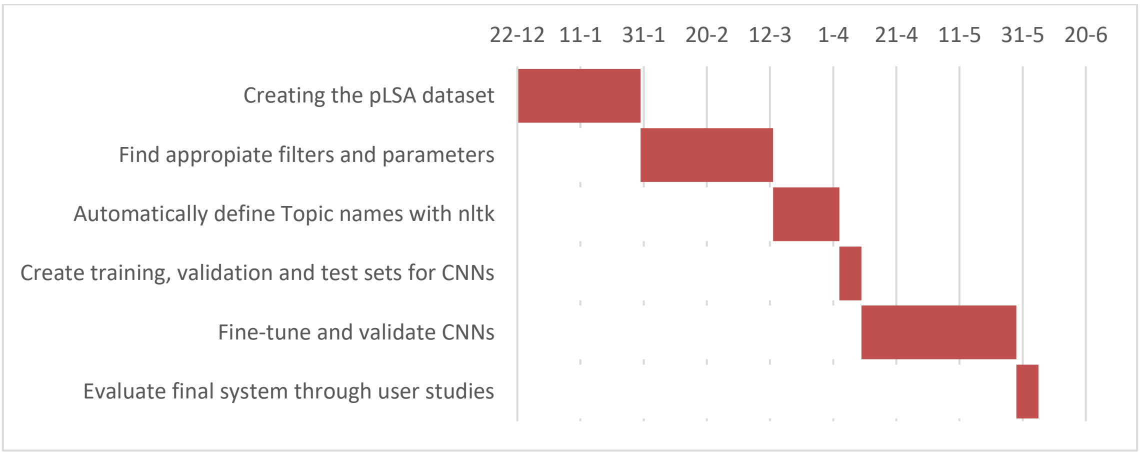


Figure 1. Ideal Gantt Chart

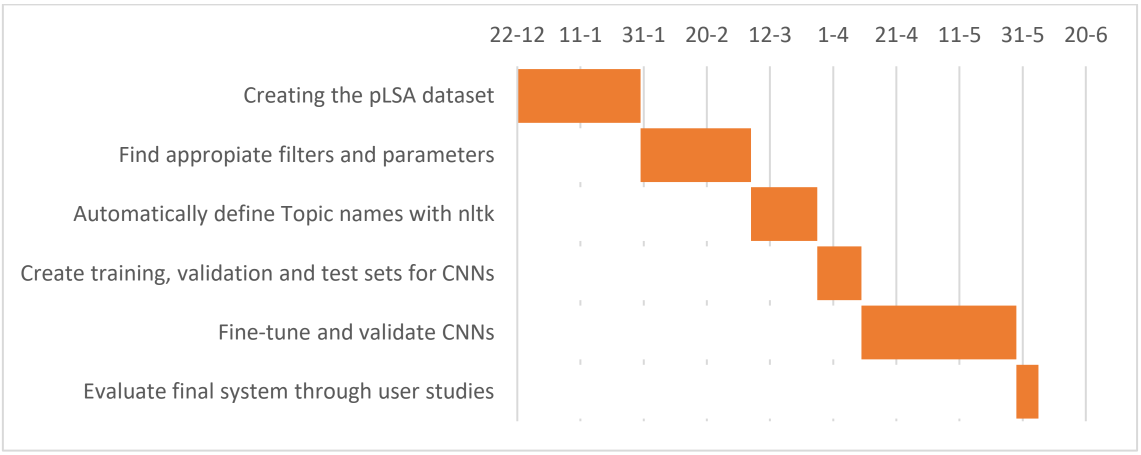


Figure 2. Real Gantt Chart

Figure 1 shows our ideal project planning. As a rule of thumb, the schedule has been followed closely. Problems have arisen throughout the year, however, they were dealt with appropriately and had a minor effect on the overall schedule, as can be seen in Figure 2.

2. STATE OF THE ART

Nowadays, there are several available softwares with the purpose of photo organization. We will take a look at some of the prominent and used ones.

2.1. Google Photos

Google Photos [1] is currently the most relevant product, on one hand, due to its seamless integration in the Google ecosystem, and on the other, due to Googles reputation as an established business in this sector. Using deep learning, their system recognizes 1100 different labels, which are easily recognizable visually by a human, and recognizes both generic visual concepts, such as “kiss” and “dance”, and specific objects like “bear” or “car”. It is a free service provided by Google.

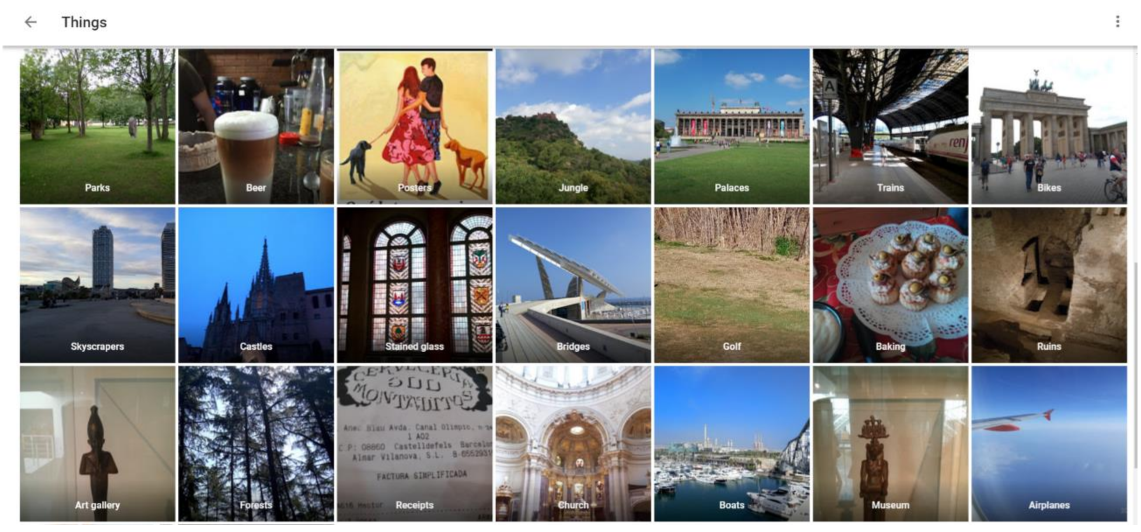


Figure 3. Google Photos ‘Things’ classification

It is based on computer vision and machine learning, using on one hand, neuronal networks to create searchable tags based on the visual content of an image, and on the other, information from other sources like EXIF metadata [5].

[1] https://photos.google.com/

This app groups images for you, dividing your photos into 2 basic concepts:

- Places
- Things

And provides a more refined classification based on these three topics. One of its main disadvantages is this pooling of different themes into one generic ‘Things’ class. As seen in Figure 3, finding topics such as ‘Receipts’ or ‘Church’ next to topics such as ‘Airplanes’ or ‘Forest’ is not an intuitive nor organised way to find a topic quickly.

Our system aims to improve this, providing a more specific classification based on a larger spectrum of topics and sub-topics, which will result in a more intuitive way of finding the desired topic.

2.2. Eden Photos

An ‘Apple’ alternative, Eden [2] creates albums based on the visual topic of the photos and organizes the photos accordingly. This app uses image recognition and artificial intelligence to classify the photos.

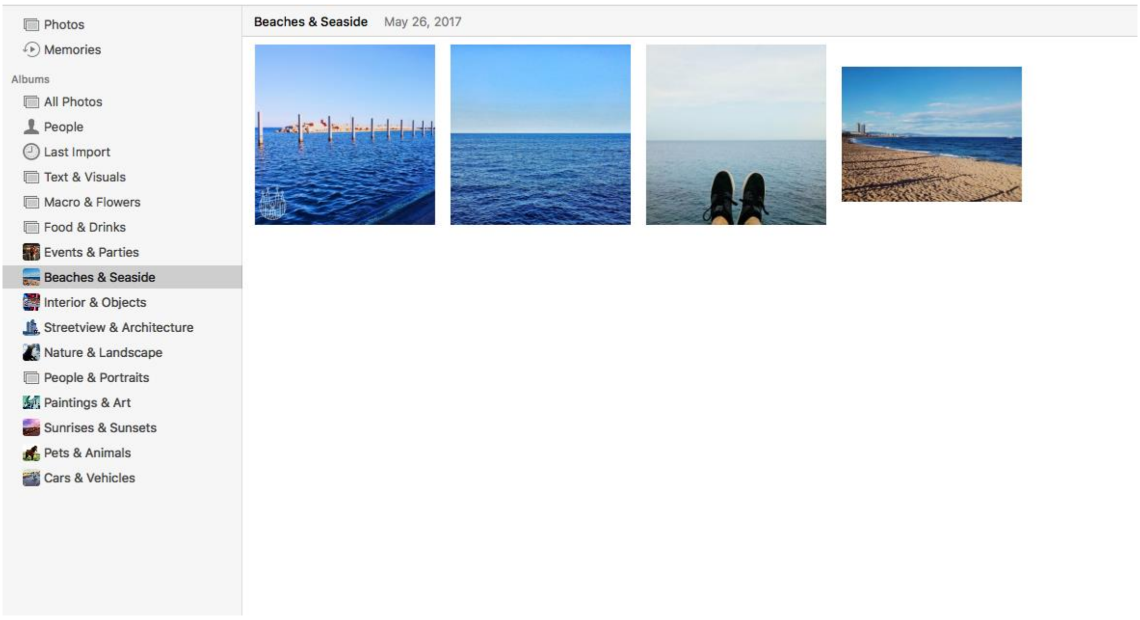


Figure 4. Eden Photos ‘Beaches and Seaside’ classification

[2] http://edenphotos.io

Eden classifies photos in the following 13 topics:

- Text and Visuals
- Macro and Flowers
- Food and Drinks
- Events and Parties
- Beaches and Seaside
- Interior and Objects
- Street view and Architecture
- Nature and Landscape
- People and Portraits
- Paintings and Art
- Sunrises and Sunsets
- Pets and Animals
- Cars and Vehicles

One of its main disadvantages is the lack of specific categories, as most of its topics have a rather broad definition, like 'Nature and Landscape'. Another disadvantage is its price, which is currently $14.99 in the App Store.

2.3. Others

The list of photo organizing software is endless. Many paid alternatives, such as *ACDSee 20*, *Zoner Photo Studio X* or *PaintShop Pro X9* offer a long list of features, even going as far as adding tools to edit and share your photos. There are, however, some main characteristics of the best-selling software in this category. Even though they all tag, categorize and organize your images, the most popular ones balance the efficiency when finding a photo and the usability of the interface. Many also offer a backup feature, together with the conversion of outdated media into viewable formats.

There has also been many attempts to tackle automatic classification. Methods such as *event recognition* [6] or *multiscale timeline* [7] are just a few of the different approaches available to tackle a common problem: summarize, classify or increase the efficiency when browsing and searching large photo albums. However, these systems don't provide a personalized classification, focusing more on the efficiency when viewing and searching photos. Our system hopes to add a viable alternative to these systems, by providing a scalable, flexible and personalized method to intuitively classify and organize a user's photos.

3. PROPOSED APPROACH

Our system consists of two sub-systems, each performing one of the following tasks:

1. Estimating the user profile
2. Photo classification based on the user-profile

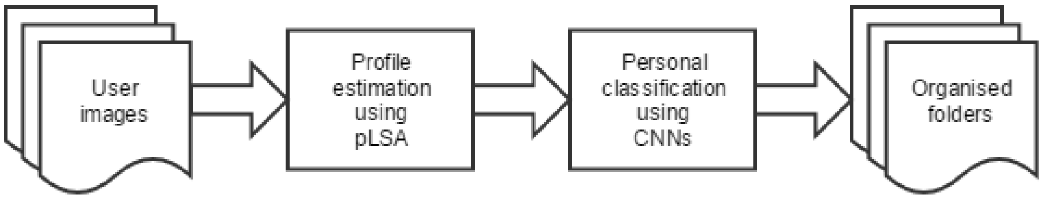

User images
Profile estimation using pLSA
Personal classification using CNNs
Organised folders

Figure 5. Our system overview

As mentioned before, we will adapt a topic discovery method used in document analysis, pLSA, to estimate a user-profile, while we will fine-tune pretrained CNNs to provide a more specific classification based on this estimated profile.

Figure 5 shows an overview of our system. The system receives as an input the user photos. These photos are tagged the extracted tags form the input of the first part: profile estimation with pLSA. The output profiles of the pLSA is the input of the second part: personal classification with CNN. Here, each CNN receives the photos of its corresponding topic, which was determined in the first part of the system. Once classified, the output is a folder, organized in folders by topic and sub-topic.

3.1. Estimating the user-profile

3.1.1. Probabilistic Latent Semantic Analysis

Given a corpus of N documents containing words from a vocabulary of size M, we organize them in K topics. We summarize the corpus of documents with a $M \times N$ co-occurrence matrix, where each element $X(w_i,d_j)$ stores the number of occurrence of the word w_i in document $d_{j.}$. Each occurrence of a word in a document has a corresponding latent variable z_k associated, representing the topic.

	Topic					
Word	**0**	**1**	**2**	**3**	**4**	**5**
Surf	**0.747**	0.000	0.000	0.253	0.000	0.000
Sunrise	**0.826**	0.000	0.000	0.174	0.000	0.000
Sandbar	**0.949**	0.000	0.000	0.051	0.000	0.000
Seaside	**0.947**	0.000	0.000	0.003	0.000	0.000
Shoreline	**0.991**	0.000	0.000	0.009	0.000	0.000
Romance	0.077	**0.923**	0.000	0.000	0.000	0.000

Table 1. Example of z_k for document 0

Table 1 shows an example of the latent variable z_k for 6 words of a document. The values have been normalized to help visualize the distribution. Here we can see that this document has a clear motif: Nature, and the tags are a good representation of the document. It is worth noting that the tag *Romance* is considered to be topic 1, but the overall output of this document will be topic 0, as it's the dominant topic.

PLSA's aim is to find the topic specific word distribution *P(w|z)* and the corresponding document specific mixing proportions *P(z|dj)* which make up the document specific word distribution *P(w|dj)*.

$$P(w|d) = \sum_{k=1}^{K} P(z_k|d)P(w|z_k)$$

Once we have *P(w|z)*, we can can compute de topic distribution of new images, and assign the dominant topic to the collection. As each image is assigned a probability value for each topic, we consider the sum of these probabilities and assign to the sequence the topic with the largest sum.

$$argmax_k \sum_{i=1}^{N} P\left(x_k \mid d_{test}^{i}\right), k = 1, \dots, K$$

Where N is the number of test images, and K the number of topics modelled. The output should be a number between 1 and K, which gives to dominant topic.

To adapt this to our problem, we consider each image a document and each visual tag as a word. A tag is a word or concept that defines the image, with each image having multiple tags. Each tag has an associated confidence level, meaning that a high confidence tag will probably be correct.

3.1.2. Tagging

To get the tags of each image, we use the Imagga API [3], which, given an image, outputs a set of tags, each with its own confidence level. Imagga uses a combination of deep learning and expansion to give a reliable output. They have a corpus of 75,000 images, tagged by both neuronal networks and people. The tags defined by real people are usually more abstract concepts such as *'time'*, *'kiss'*, *'dance'*, while the neuronal networks define specific topics such as '*car'*, *'person'*, *'street'*. Imagga applies deep learning first, giving a set of tags with clearly defined topics. They then find which abstract tags usually appear with these tags, and gives a combined output to the user, containing tags which are both abstract concepts and specific topics.

3.1.3. Choosing the topics

When choosing the topics, our aim was to build a set of generic categories that characterize a user profile. In our case, we opted for the following six topics:

- Interior and Objects
- Nature and Landscape
- Food and Drinks
- Pets and Animals
- Street view and Architecture
- People and Portraits

We chose these topics because they provide a good generalization of the different types of user profiles. This approach provides a great flexibility, as the list of topics can be expanded to include less common topics not covered by the current selection, for instance *Sports and Adventures*.

3.2. Personalized photo classification

3.2.1. Convolutional Neuronal Network

Once we have the user-profile, we can proceed with the personal classification. We will be using CNNs, which are networks made up of a cascade of layers of neurons, with weights connecting them. The main difference between normal neuronal networks and convolutional neuronal networks is that the latter receive images as input, and therefore has been optimize with image processing in mind, increasing the performance. CNNs are usually a combination of 3 main layers:

[3] https://docs.imagga.com/

- Convolution layer
- Pooling layer
- Fully connected layer

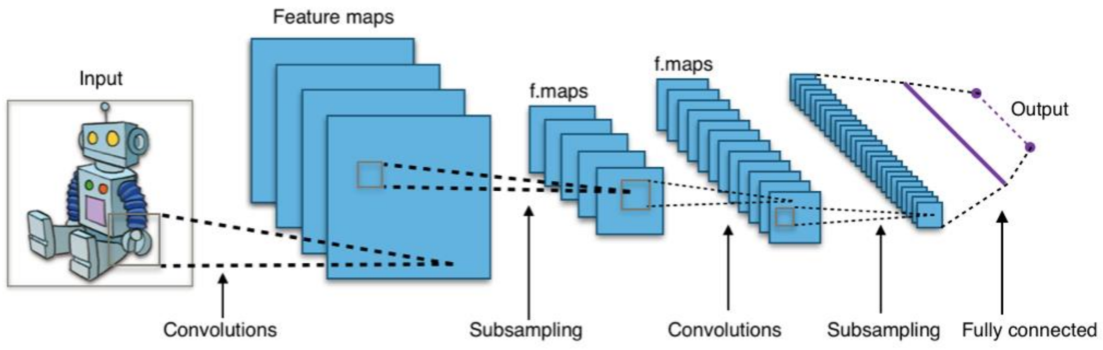


Figure 6. Typical CNN structure

In addition to these layers, all CNN architectures contain ReLU (Rectified Linear Units) layers and Loss layers. The first layer of all CNNs is the **input layer**, which contains the original image pixel values, and will therefore have the same dimensions as the image.

The **convolution layer** is the defining property of CNNs, and is what differentiates them from normal neuronal networks. The parameters of this layer are a set of filters (or kernels), which are connected to local regions of the input volume. These filters are convolved with the input volume, computing a dot product between their weights and their connected region. This means that the output volume of this layer will be defined by the number of filters used. For example, with an input volume of 32x32x3 and 12 filters, the output volume would be 32x32x12.

The **ReLU** layer, or **Rectified Linear Units** applies an activation function to all the elements of the input volume, in order to speed up the training process, as well as introducing the non-linearity. Some common activation functions are:

- **Max:** f(x) = max(0,x)
- **Sigmoid:** $f(x) = (1 + e^{-x})^{-1}$
- **Tanh:** f(x) = tanh(x)

However, only the max function is used since the others lead to the vanishing gradient problem, where, after a certain amount of training, the gradient becomes 0. This means that this section of the network stop learning.

The **pooling** layer performs a non-linear down-sampling, to reduce the amount of computation in the network. *Max pooling* is the most common type of pooling. It partitions the input volume into rectangles, and outputs the max of each sub-region. The idea is that the specific feature location is not as important as its relative location to the other features. This means that the filters of the following layer will see a larger part of the image, allowing the network to learn features of growing complexity, going from low level features such as edges and curves, to high level features like hands or ears. It also helps avoid over-fitting, as it provides a more abstract representation of the input volume.

The most common form is pooling the layer with 2x2 filters, and a stride of 2. This means that the input volume is split in 2x2 sub-regions, and outputs the max of 4 numbers for each sub-region. The depth of the volume does not change. In this fashion, we reduce the spatial dimension drastically.

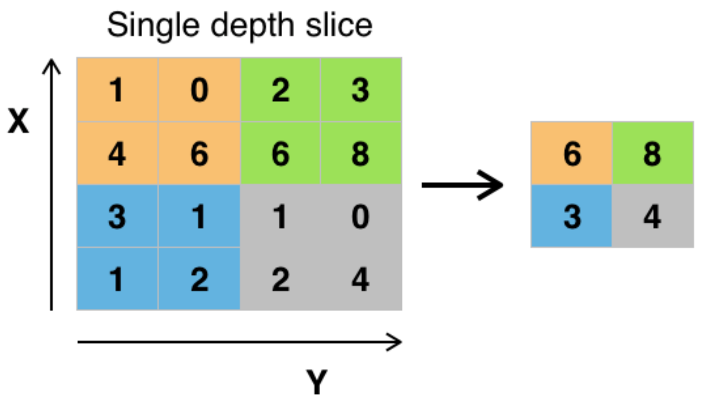


Figure 7. Max pooling with a stride of 2

One of the most important layers is the **fully connected layer**, which connects all the neurons to the activations of the previous layer. The output is a volume of size 1x1xN, where N is the number of classes.

The last layer is usually the **loss layer**, which defines the penalization when the predicted label deviates from the ground truth. There are various loss functions that can be used here, depending on the task at hand. We will use the *SoftMax* loss function, which computes the class score out of K mutually exclusive classes. The output is a vector of normalized class probabilities.

CNNs requires a lot of resources and time, as the training process consumes a lot of time and computational power. The **training process** is the stage where we feed the network labelled data, so that it adapts its weights and biases to learn and recognize our data. While training, one of the most common problems is **over-fitting**. Over-fitting is the result of the network becoming too specialized on our dataset, and therefore only classifies correctly images from the training dataset, while failing to classify images that were not part of the training dataset. Over-fitting usually occurs when the dataset is small compared to the number of iterations. To avoid this problem, some common solutions are to increase the dataset size, or decrease the training iterations, however, tracking the loss during the training process is crucial, as it shows if the network is learning adequately or if it has stagnated.

Another method to reduce over-fitting is using *dropout* [8]. Dropout is an efficient yet simple alternative, where a neuron is kept active with a probability of p, where p is usually 0.5. Otherwise, the neuron is 'dropped out', setting it to zero. The removed nodes are reinserted (with their original weights) in the next forward pass. By applying this method, we avoid over-fitting as it leads to a more robust learning, generalizing new data better. It's worth mentioning that dropout is only applied in the training phase, not in the test phase. Figure 8 shows an example of a network before and after dropout.

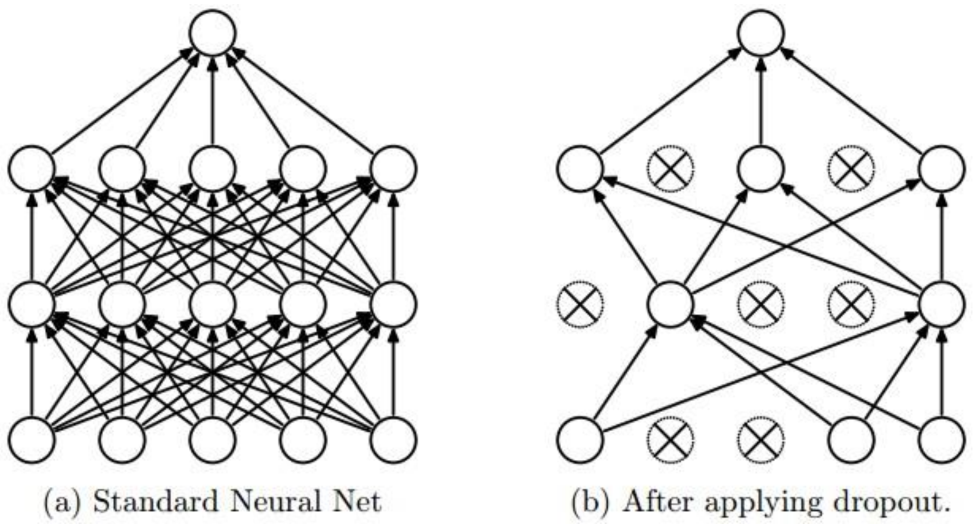


Figure 8. Before and after Dropout

With our system, we will use pretrained CNNs, and adapt these pretrained networks to our needs. The reason we use pretrained networks (instead of training from scratch), is that it saves us a lot of time, as training a network from scratch requires a huge number labelled images, while **fine-tuning** a pre-existing network usually requires a much smaller dataset. Fine-tuning is the process of adapting an already trained network. It is important that the topic of the trained network is similar to our purpose, as fine-tuning a network to learn the difference between cats and dogs when it has been trained on landscapes will produce no results unless a huge amount of data is used for fine-tuning.

We will be using different pre-trained networks, one being Places2, a CNN trained on a dataset with the same name, made up of indoors and outdoors photos of places. This database contains over 10 million images, with more than 400 unique scene categories.

It features from 5000 to 30000 images per class. We will use this net to classify two of the six profiles: ‘Nature and Landscape’ and ‘Street view and Architecture’. To train the remaining four categories, we used the caffe reference model, a slight variation of AlexNet, trained on ImageNet. To train ‘Food and Beverage’, we used images from a dataset named Food101, which contains 101 food categories, with 101,000 images: 750 training images and 250 manually reviewed test images. We chose 15 of the most common categories for this project, however, the categories are flexible and adding the 101 food classes would only require using the whole dataset to fine-tune. We tried to use the labels already pre-trained on ImageNet. For example, Food and Beverage uses 6 new labels, with the other 11 already pre-trained on ImageNet, while Interior and Object uses 2 new labels.

3.2.1.1. AlexNet

With a similar architecture to LeNet, it is deeper and bigger, with convolutional layers stacked one after the other. It has 5 convolutional layers and 3 fully-connected layers [9]. It was submitted to the 2012 ILSVRC, and improved the performance significantly, by as much as 10% [10].

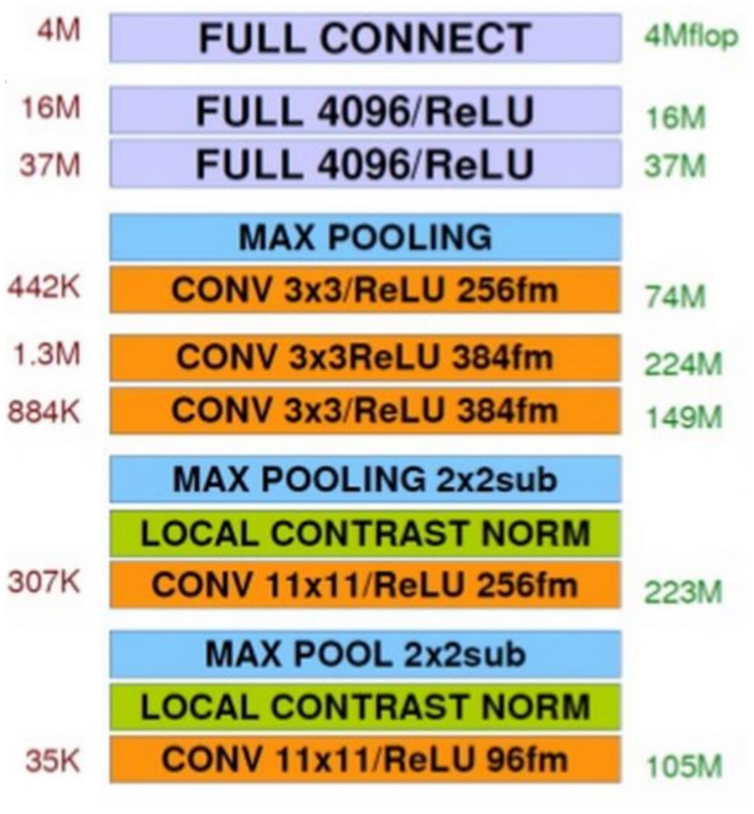


Figure 9. AlexNet architecture

3.2.1.2. VGGNet

This architecture came in second in the ILSVRC in 2014, just after GoogLeNet. It has 16 layers, compared to the 8 layers of AlexNet. These 16 layers are split into 13 convolutional layers and 3 fully connected layers. Its main feature is its depth, however, this means that it is time consuming and uses a lot more memory, as it has 140 million parameters, compared to the 60 million of AlexNet. Nonetheless, modified version have appeared, reducing drastically the number of parameters without affecting the performance of the network [11].

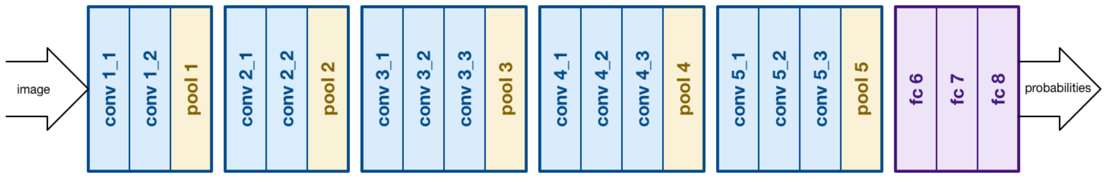


Figure 10. VGGNet architecture [12]

3.2.2. Choosing the sub-topics

In order to decide which sub-topics to define, we asked real users what type of topics they would like to see defined, and which topics they currently believe are missing from the current apps. We decided to define from 15 to 20 different sub-topics per profile.

Nature and Landscape: mountain, beach, coast, valley, glacier, cliff, lake, sky, waterfall, river, snow, sea, trail, sunset, forest, flower, grass

Architecture and Streetview: car, alley, railroad track, park, cathedral, street, bridge, highway, house, crosswalk, fountain, pier, tower, lawn, harbor

Animals and Pets: cat, gorilla, horse, turtle, elephant, bird, fish, dog, shark, rabbit, frog, lizard, butterfly, snake, bear

Food and Beverage: strawberry, banana, cocktail, coffee, ice cream, beer, hot dog, spaghetti, hamburger, cake, french fries, salad, soup, pizza, bread

Objects and Interiors: bookcase, television, laptop, bed, table, window, stairs, lamp, couch, chair, painting, shoe, guitar, watch, frying pan

People and Family: selfie, portrait, family, child, adult, group

4. IMPLEMENTATION

4.1. Estimating the user-profile

4.1.1. Preparing the dataset

To prepare the dataset, we start by ‘tagging’ de fotos of 8 users, to get a tag vector per image, showing the prevalent subjects of the fotos. To tag the images, we use the Imagga API.

User	Number of images
1	527
2	3551
3	1000
4	1000
5	1000
6	729
7	628
8	1964
9	823
10	502
11	827

Table 2. pLSA dataset distribution

The Imagga API, given an image, outputs several tags with its corresponding confidence level. Using the ‘verbose’ parameter, we can identify the origin of the tag: recognition or additional. Additional tags are user-identified, while recognition tags proceed from computer vision.

In addition, we only want to consider tags that appear at least 10 times throughout all the images. By filtering the uncommon tags, we get a more precise output, as we’re not interested in tags that the user seldom uses. We also split words composed of more than word. For example, ‘mechanical vehicle’ becomes two separate words: ‘mechanical’ and ‘vehicle’, both with the confidence level of ‘mechanical vehicle’. We did this to increase the vocabulary size, and found that it improved the results.

The tags that fulfil our criteria form our vocabulary. Each image has its corresponding tag vector, containing the tags that are part of the vocabulary. This matrix of vectors, also known as matrix of co-occurrence, is then passed to the algorithm in the form of an array of dictionaries, where each position in the array corresponds to an image.

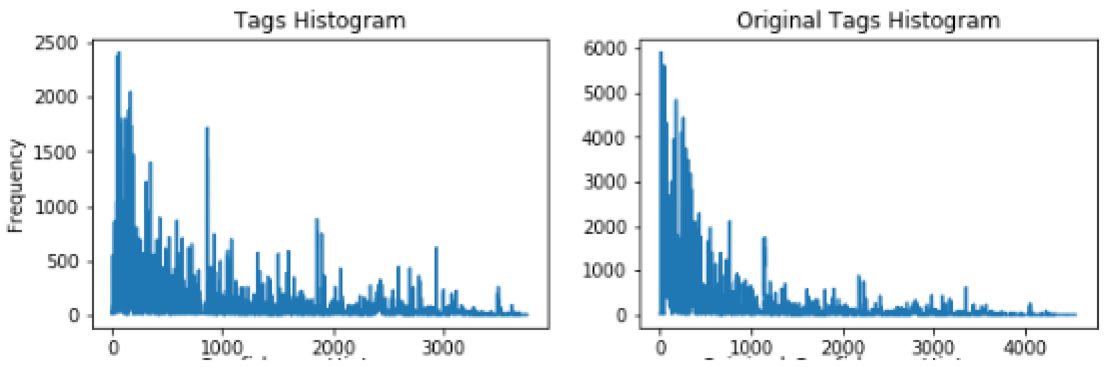


Figure 11. Filtered Tags (Left) vs Original Tags (Right)

Figure 11 shows the amount of tags before and after filtering. Here we can see that, after filtering the words that appear less than 10 times and those that appear in all the users, we reduce the number of tags by approximately 1000 words.

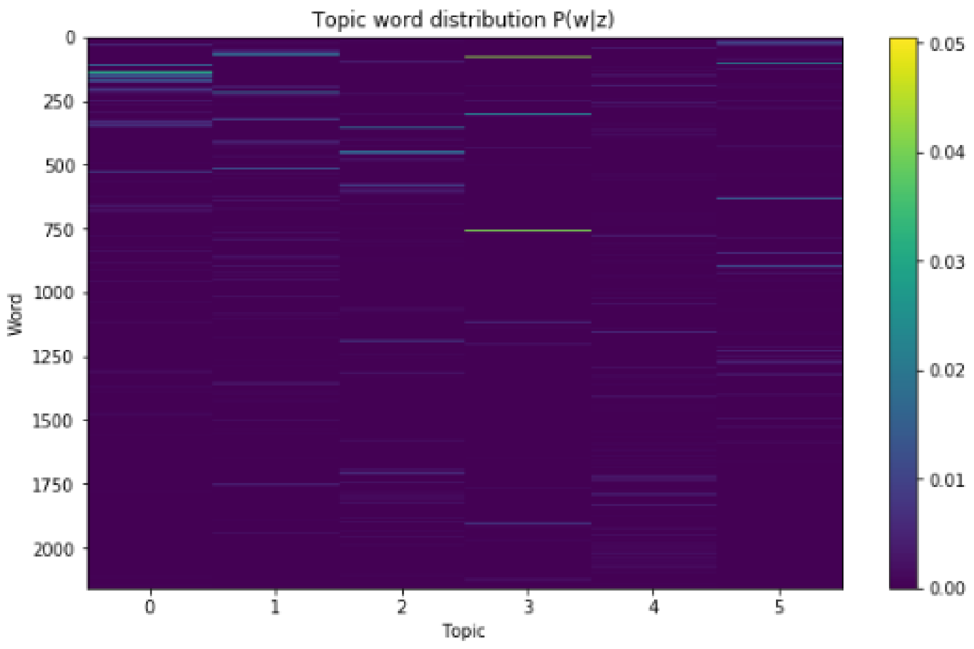


Figure 12. Topic-word distribution of pLSA output

Figure 12 shows the distribution of tags across the 6 topics. As shown, there are clear differences between each topic. Our final topic definitions are:

People and Family: boy, child, youth, kid, girls, together, clothing, healthy, handsome, children

Streetview and Architecture: tower, town, famous, exterior, religion, monument, church, brick, historical, buildings

Food and Beverage: fresh, healthy, meal, eating, plate, diet, lunch, delicious, tasty, gourmet

Interior and Object: estate, apartment, living, door, residential, comfortable, real, inside, contemporary, carpet

Animal and Pet: animal, dog, canine, mammal, pet, hunting, fur, cat, animals, pets

Nature and Landscape: weather, mountains, rocks, shoreline, barrier, sunrise, surf, seaside, country, cloudy

With these topics defined, we save the PLSA object so we can load it in our system and apply inference with every new user.

4.1.2. Probabilistic Latent Semantic Analysis

For this part, we will be using the code found at Github by the user *isnowfy* [4], as it was one of the few current implementations to include an inference function.

Using the array of word vectors as input, the algorithm outputs are stored in internal attributes to the class 'PLSA'. Here we can access the document-topic distribution and the word-topic distribution.

Parsing this information, and with 6 number of topics, showing the top 10 words per topic, we get 6 defined topics. To give each topic a name, we use the semantic similarity between the top 10 words and the categories defined at the start. To calculate this similarity, we will use the *Natural Language Toolkit* (NLTK) [13], where we can find the similarity between two words based on different criteria, such as the shortest path in the hypernym taxonomy.

NLTK provides an interface to more than 50 corpora and lexical resources such as WordNet. In addition, it also provides text processing libraries for classification, stemming, parsing and many more functions. We will be using it to access the WordNet interface. WordNet on the other hand is a lexical database of English were words are grouped into sets of synonyms called synsets and it provides usage examples and definitions, as well as recording de lexical relations among the synsets [14]. This results in a network of related words and concepts, and we will use this feature to find the semantic similarity between words.

[4] https://github.com/isnowfy/plsa

By accessing WordNet through the NLTK interface, we can use one of the many similarity functions found in the toolkit. Some of these similarity functions include the *Resnik Simlarity, Jiang-Conrath Similarity*, or the *Lin Similarity* [15]. We opted for the Lin function, as it produced the best results. This function returns a score showing the similarity between two word senses, *S1* and *S2*, based on the Least Common Subsumer (LCS). The LCS is the most specific ancestor node. The formula used is the following:

$$\frac{2 * LCS}{S1 + S2}$$

This will return a value between 0 and 1, where 0 means no similarity, while a value closer to 1 indicates a strong similarity.

We use this similarity for all the words in each topic, and compare them to the two words in the topic name. The topic that has the highest probability is the one that will get assigned to the nameless topic.

In order to use nltk, we first download the entire corpus collection using nltk.download(). Once we have the corpus, we can choose which to use, which in our case is the *semcor* corpus, as it gave us the best results.

4.2. Personalized photo classification

4.2.1. Defining the datasets

In order to train, and validate the training process, we need to define train and validation datasets for each topic, as well as a test dataset to check the result of the process. Our goal is to get approximately 1000 to 2000 images, and once we have the complete dataset, split it in the following fashion:

- 70% - Training set
- 15% - Validation set
- 15% - Test set

To find the images, we used a variety of sources, such as the Places2 dataset [16], the Food101 dataset [17] , ImageNet [18] or the Gallagher Dataset [19]. However, as our goal is to classify photos done mainly with a smartphone, we tried to fetch images that were made by a smartphone. Hence, we decided to scrape Instagram and Flickr, were the photos represent a more day-to-day motif, rather than a perfect photo made with optimum light conditions, etc.

Once we have all the datasets defined, we created a file for each topic, in the format:

- File_path, label_number

And another file with the labels. Once these files are defined, we can proceed with the training of the CNNs.

In addition, we also decided to apply data augmentation to the datasets that had a lower amount of images or that gave bad results. We applied two techniques: Blurring and Darkening. Blurring was applied using a Gaussian filter with a sigma of 3, while the darkening was applied by subtracting a constant from the image. In this fashion, we were able to replicate low light conditions or blurry photos, which are a staple of many smartphone photos.

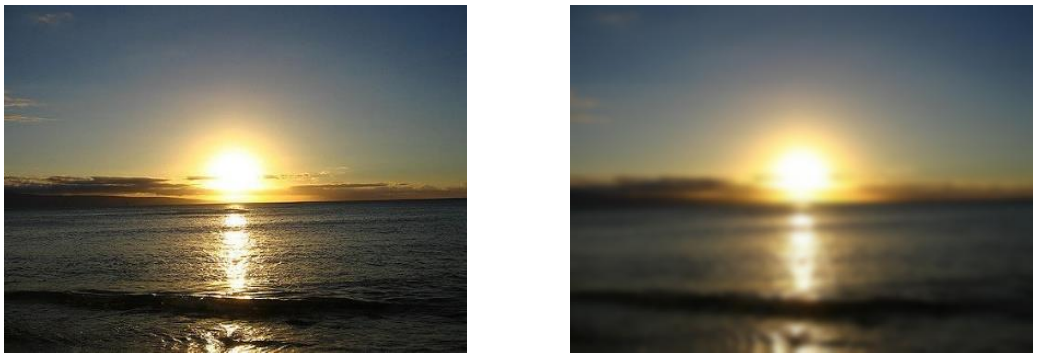


Figure 13. Data augmentation using Gaussian filter with sigma 3.
Original (Left), Blur (Right)

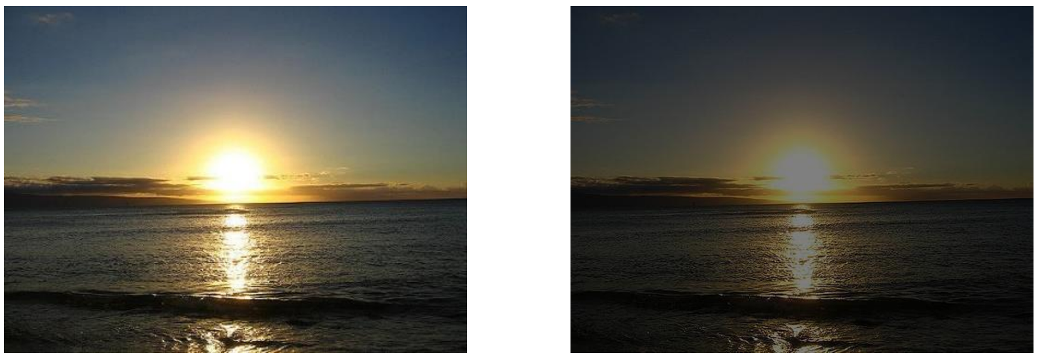


Figure 14. Data augmentation using Darken.
Original (Left), Darken (Right)

4.2.2. Training the models

To implement CNNs, we chose the deep learning framework ‘Caffe’ [4]. It allows us to define, train and test CNNs. We opted for PyCaffe, Caffes python wrapper, to interact with the networks. Caffe was developed by Berkeley Vision and Learning Center, and is widely used. Other options were Torch [5] or TensorFlow [6].

Using PyCaffe is straightforward, and while its CLI is easier to use, PyCaffe offers more tools to analyse and monitor the networks progression. The general workflow used during the fine-tuning process of the 6 different CNNs:

- **Define external parameters**
  - Training and Validation set size
  - Number of Epochs
- **Load CNN**
  - Load architecture
  - Load weights
  - Load mode (TEST/TRAIN)
- **Define and save solver**
- **Execute training**

Our solvers use a base learning rate of 0.001, decreasing the rate every 10 epochs. An **epoch** is defined as a forward and backward pass to the whole dataset, while an iteration is a forward and backward pass, using the amount of images specified by the batch size. For example, if we have 1000 images in the training set, with a batch size of 250, then we will need 4 iterations to complete 1 epoch. In the solver we also define our snapshots, saving the solver state every 5 epochs, and we train each model for 25 epochs.

Once the training ends, the results are saved in a log file, which is parsed and a plot is made, showing the accuracy and loss of both the training and the validation set.

[5] http://torch.ch/
[6] https://www.tensorflow.org/

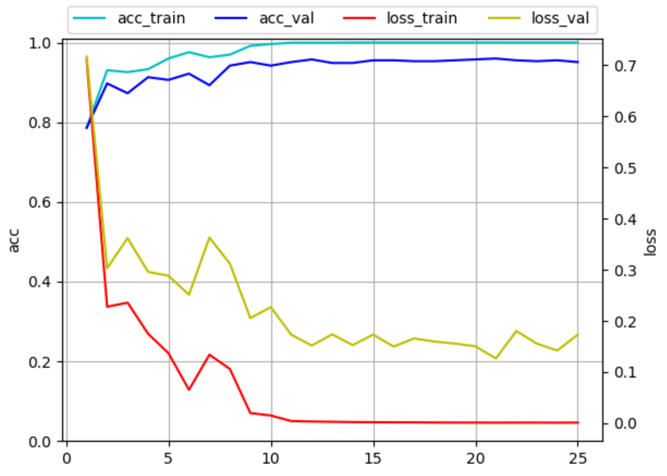


Figure 15. Monitoring accuracy and loss of CNN

Figure 15 shows the plot of the training and validation accuracy, together with its loss. The similar trends indicate that there is very little over-fitting, and the constant decrease in loss indicates a good learning rate.

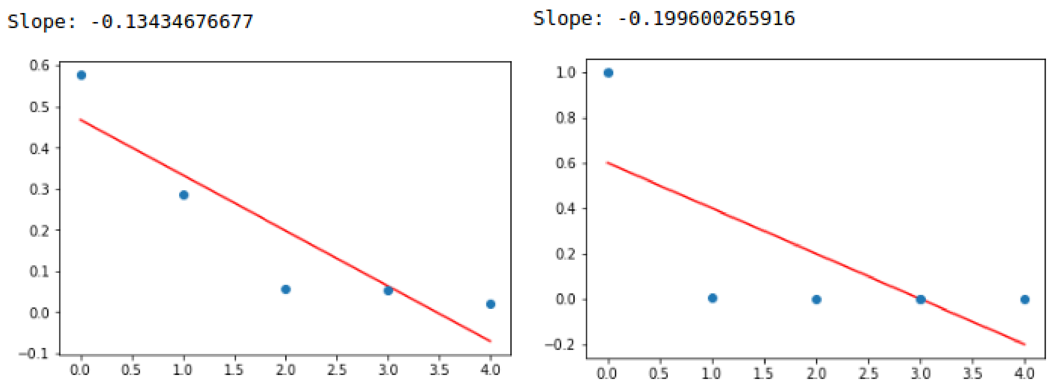
4.3. Our system

Our system is defined by the user-estimation followed by the personal photo classification. The input will be the set of images of the user, and the output will be a folder, organised by topics and sub-topics.

In order to increase the reliability and accuracy of our system, we have applied two thresholds, depending on the stage. The first threshold is applied when estimating the user-profile. We only classify images that have a topic distribution high enough that we are sure it is a valid topic. This way we avoid classifying images that does not have a well-defined topic or a topic not included in the ones for which we trained the system.

The other threshold is applied when classifying the images with a well-defined topic. This threshold removes images that have been wrongly classified by the pLSA inference. For example, if an image has low confidence tags, its estimated topic will probably be wrong. In this case, this threshold will make sure that it does not classify the image, as the predicted class would most probably be wrong.

These two thresholds have the disadvantage that not all the images are classified. We found that usually 65% to 75% of the user photos are classified, leaving roughly 30% of the photos without class. Instead of not classifying these images, we classify these images using the 6 different models, and save the linear regression of the top 5 results of each model. The top prediction of the model that has the flattest slope will be the classified class of the image. We use the slope as an indication of the disparity of the predictions.



```
Slope: -0.13434676677
```

```
Slope: -0.199600265916
```

Figure 16. Top 5 predictions of an image of a valley.
People and Family (Left) vs Nature and Landscape (Right)

Figure 16 shows a comparison between the top 5 predictions of a photo of a valley. The predictions to the left were made using the People and Family model, while the right predictions were made using the Nature and Landscape model. As we can see, there is a clear distinction: Nature and Landscape has a top prediction with a high confidence, while the other 4 are rather low. This creates a flat slope, as the variation between the last 4 predictions is low, with the only outlier being the top prediction. On the other hand, People and Family does not have a clear winner, instead it has a more diverse distribution, creating a steeper slow, as the variation is larger.

4.4. Visualizing the results

In order to show the end result, we opted for an HTML implementation, showing the 6 topics, with each sub-topic appearing after clicking the respective topic. In order to accomplish a visually pleasing organisation, we used Bootstrap [20], together with Javascript to give a more personal touch. The end result is a simple HTML page, where the user chooses the output folder of our system, and the page will show the 6 topics, with a cover photo representing its content.

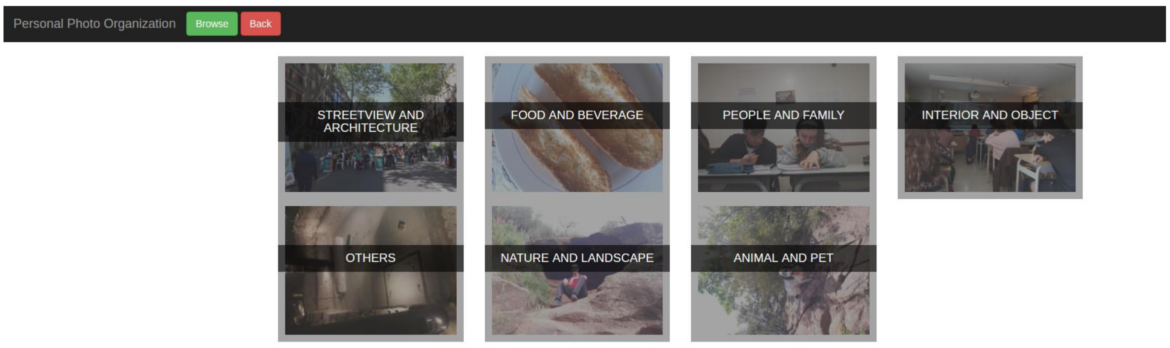


Figure 17. An example of our visualization

5. EVALUATION AND RESULTS

5.1. Probabilistic Latent Semantic Analysis

5.1.1. Cross-validation

To validate the training of pLSA, we use a 3-fold cross-validation. We take the first 3 users, and train with the remaining 8 and infer the topic of the 3 users. We then take the next 3 users, and repeat the process, till we have used all the users as validation.

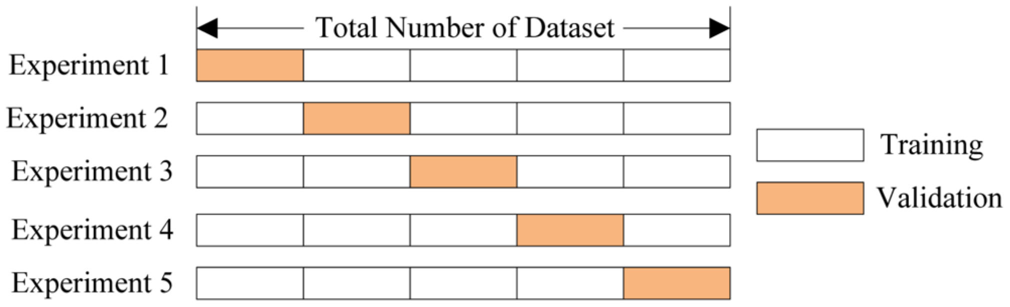


Figure 18. An example of 2-fold cross-validation [21]

5.1.2. Topic Coherence

To evaluate the topics defined by pLSA, we calculate the observed topic coherence. It measures how well a topic model assigns topics to models. Defined by Newman et al in 2010 [22], there are various implementations, however, we will be using the Normalised Pointwise Mutual Information (NPMI) variant:

$$\sum_{j=2}^{N}\sum_{i=1}^{j-1}\frac{\log\frac{P(w_j,w_i)}{P(w_i)P(w_j)}}{-\log P(w_i,w_j)}$$

Where *$P(w_i)$* and *$P(w_j)$* represents the probability of seeing w_i and w_j respectively in a random document, and *$P(w_i,w_j)$* the probability of seeing w_i and w_j in a random document. The output value will be a number between -1 and 1, where 1 indicates a strong coherence, while -1 indicates a lack of coherence.

The algorithm receives as input the topic file, were each line represents a topic, and each line contains top-N words that define the topic, which is the output of pLSA. The output shows the individual topic coherence of each topic, together with and average and median topic coherence.

In our case, we got the following output:

- **People and Family**: 0.36
- **Street-view and Architecture**: 0.47
- **Food and Beverage**: 0.39
- **Interior and Object**: 0.59
- **Animal and Pet**: 0.44
- **Nature and Landscape**: 0.38

- **Average Topic Coherence** = 0.438
- **Median Topic Coherence** = 0.412

5.2. CNN

To evaluate the training process, we monitored the training and validation loss, together with their accuracy. In order to avoid our main problem, over-fitting, we made sure that the training and validation accuracy followed a similar trend. If the validation accuracy was much lower than that of the training accuracy, we would have over-fitting, resulting in a poor real-world performance of the model.

CNN	Accuracy
Nature and Landscape	89,50%
Street view and Architecture	90,25%
Food and Beverage	83,50%
Animal and Pet	84,00%
Interior and Object	84,25%
People and Family	88,00%

Table 3. CNN accuracies

Table 3 shows the accuracy of each model. To calculate the accuracy, we use the 'test' mode in Caffe, where we calculate the average accuracy of the model when classifying the test set over 100 iterations with a batch size of 4.

5.3. User study to evaluate the system

Since the accuracy of our system depends on how the user perceives the classification, we perform user studies, such as those used in the paper *Predicting Important Objects for Egocentric Video Summarization* [23]. We recruit 5 users, with each user providing 500 to 1000 photos. In addition, we chose 5 famous 'YouTube Vloggers', downloaded 1000 images from their Instagram, and classified them. We chose social media celebrities who are used to filming their daily life, as their Instagram photos usually represent a typical smartphone photo from their day-to-day.

Each user will be asked to evaluate their photo classification, and that of the other users to better evaluate the overall accuracy of the system. We will sit the user at a desk with two screens, one screen showing our system classification, the other showing the system we are comparing it to, which will be either Google Photos or Eden Photos. The users will not be aware of which screen shows which system. The first phase will show the users photos classified by the two systems, and we then proceed to ask the first question. Once we have recorded the answer, we show the other user's photos. Next, we ask the second question, changing the user´s photos until we have shown all the users. By the end of the evaluation, a single user will have done 10 evaluations, 1 on its own photos and 9 on other user's photos.

5.3.1. Evaluation of own photos

To evaluate our system, we classify the user photos in three different apps: Google Photo, Eden and our system. We the proceed to show the user the photos, classified using our system and that of Google Photos. Next, we ask 2 questions: (1) *Which system has the best categories?* and (2) *Which system classifies better?* The first question specifies which type of classification is best suited for the user; a general topic organization, or a more hierarchical classification, with topics and sub-topics. The second question addresses the perceived accuracy of the system. Once we have compared our system to Google Photos, we repeat the same process with Eden.

	Much better	*Better*	*Similar*	*Worse*	*Much worse*
Sub-category classification	0%	**60%**	40%	0%	0%
Accuracy of sub-categories	0%	0%	40%	**60%**	0%

Table 4. User evaluation of Google vs Our system on users own photos

Table 4 shows our results compared to Google Photos. As we can see. 3 out of 5 users preferred our topic classification to that of Googles. However, the consensus is that our accuracy is worse, with 60% of the users evaluating Google Photos as the more accurate system.

	Much better	*Better*	*Similar*	*Worse*	*Much worse*
Sub-category classification	20%	**40%**	40%	0%	0%
Accuracy of sub-categories	0%	20%	**60%**	20%	0%

Table 5. User evaluation of Eden vs Our system on users own photos

On the other hand, Table 5 shows the evaluation results when compared to Eden. Yet again we can see that our topic classification is preferred, with 3 users out of 5 choosing our system over Eden Photos. In terms of accuracy, we can see we got similar results, with 60% of the users finding our systems equal in accuracy.

The results show that our topic definition is better than both Google Photos and Eden Photos, however, in terms of accuracy our system performs worse than Google Photos in terms of sub-categories classification, while outperforming Eden Photos in terms of topic classification. These results are understandable, as our system has used much less images for training. As an example, our topics trained on VGG-Places (Nature and Landscape , Street-view and Architecture) perform much better than those trained on CaffeNet, such as Animals and Pets. This is due to the architectures used, as CaffeNet performs worse than VGG, and that VGG-Places was already pre-trained on a large quantity of images of both ‘Nature and Landscape’ and ‘Street-view and Architecture’ before we started fine-tuning, while CaffeNet was trained on ImageNet, which was not specifically pretrained on one topic. Moving from CaffeNet to VGG and using a larger dataset should increase the performance of our system in terms of accuracy.

5.3.2. Evaluation on other photos

The next step is to measure the accuracy of our system on an absolute scale. Therefore, we show the users the classification of other users, and ask them to evaluate the accuracy. As before, we compare our system, first with Google Photos, then with Eden, asking the same 2 questions.

	Much better	*Better*	*Similar*	*Worse*	*Much worse*
Sub-category classification	4.44%	24.44%	**57.78%**	11.11%	2.22%
Accuracy of sub-categories	0%	6.67%	**40.00%**	40.00%	13.33%

Table 6. User evaluation of Google vs Our system on other photos

Table 6 shows our performance against Google Photos. Out of 45 evaluations, 9 per user, we found that our topic organisation was better than Google Photos 28.88% of the time, while we obtained similar results 57.78% of the time. On the other hand, our system loses in terms of accuracy 53.33% of the time. We observed that our users found or topic organization equal, and even better than that of Google Photos. They preferred our more detailed organisation, with topics and subtopics, while praising Googles 'Places' organisation, where they classify the photos geographically.

	Much better	*Better*	*Similar*	*Worse*	*Much worse*
Sub-category classification	**44.44%**	35.56%	15.56%	4.44%	0.00%
Accuracy of sub-categories	0.00%	17.78%	**53.33%**	22.22%	6.67%

Table 7. User evaluation of Eden vs Our system on other photos

Table 7 shows the results against Eden Photos. In this case, there was a clear trend: the users found Eden's organisation too shallow, or too abstract. This is clearly shown in the table, where 80% of the users preferred our topic organisation. In terms of accuracy, 52% of the users found the accuracy to be similar, with mistakes common to both systems.

As a side-note, all our users mentioned that our system tends to classify animals poorly, while categories like nature tend to produce reliable results. This could be a result of slight over-fitting, or show a need to improve the animals network. In addition, there was a clear difference in performance depending on the quality of the photo. Users that use high quality cameras usually obtain a higher accuracy in their classification, while blurry or low-light photos tend to be all over the place. However, in these cases, all systems worked poorly, with Google having an edge over our system and that of Eden. Many users also mentioned our accuracy when classifying people, specially being able to differentiate between selfies, portraits and groups. Many users were impressed with this classification, and appreciated the specific sub-topics.

We also observed that Instagram filters that distort the colours tend to decrease the tag accuracy returned by the Imagga API. This results in a lower, or mistaken topic prediction, and decreases the overall performance of the system, even with the thresholds aforementioned.

In general, the users were satisfied with our topic classification, finding it represented their photos better than that of the other systems, albeit with worse accuracy depending on the user.

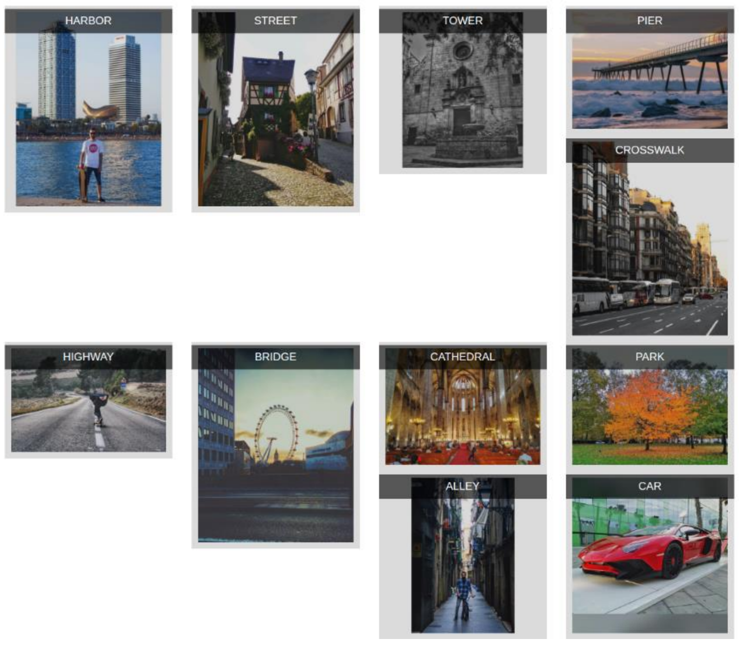


Figure 19. Street view and Architecture classification

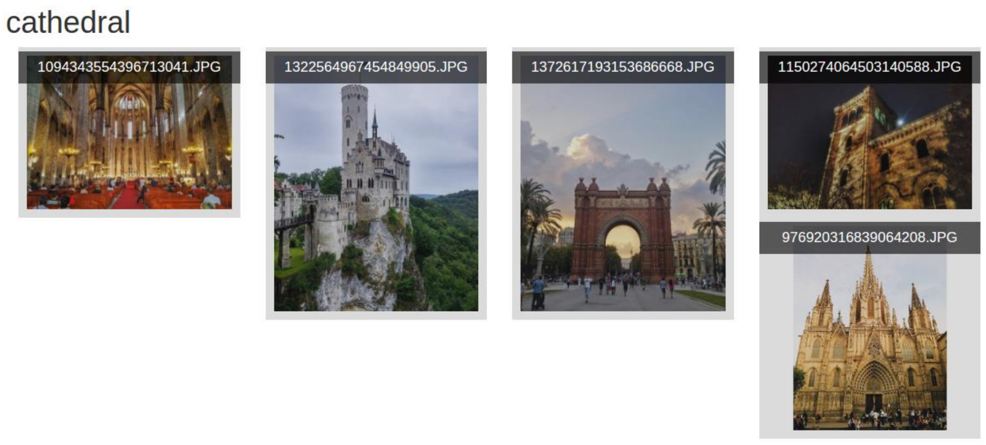


Figure 20. Cathedral classification

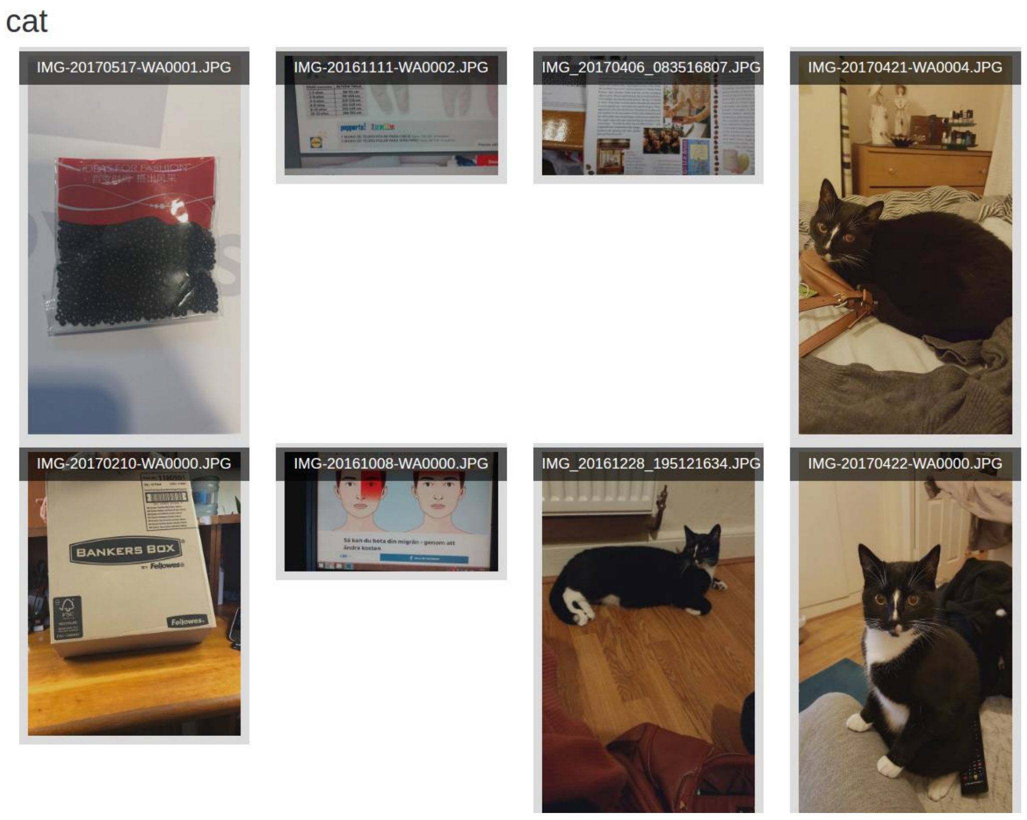


Figure 21. Cat classification

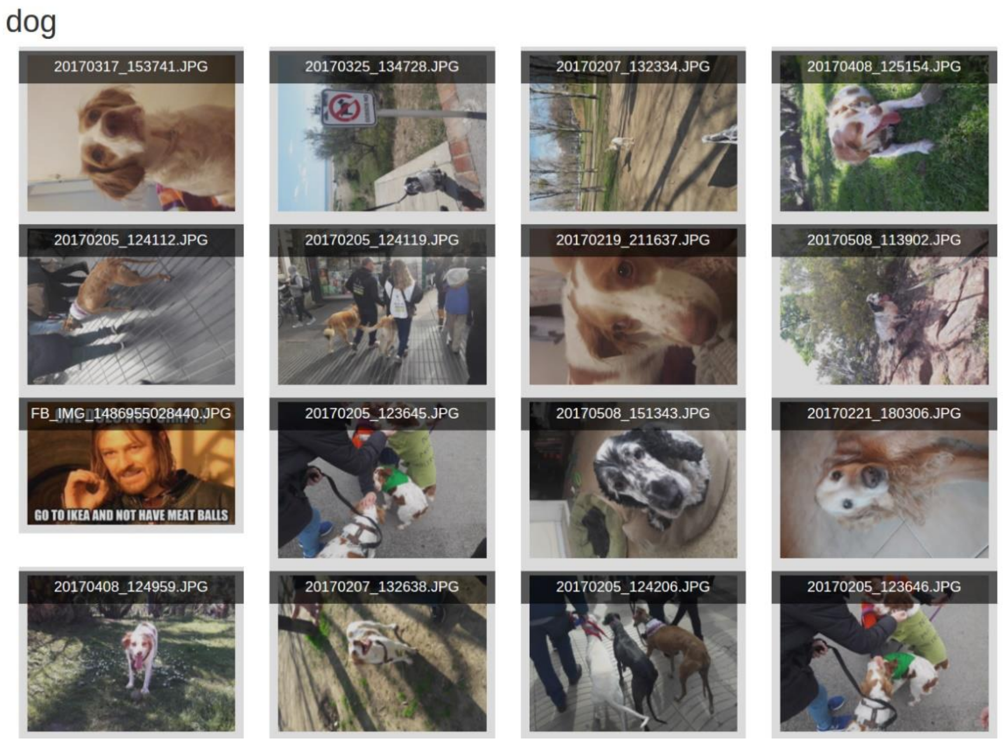


Figure 22. Dog classification

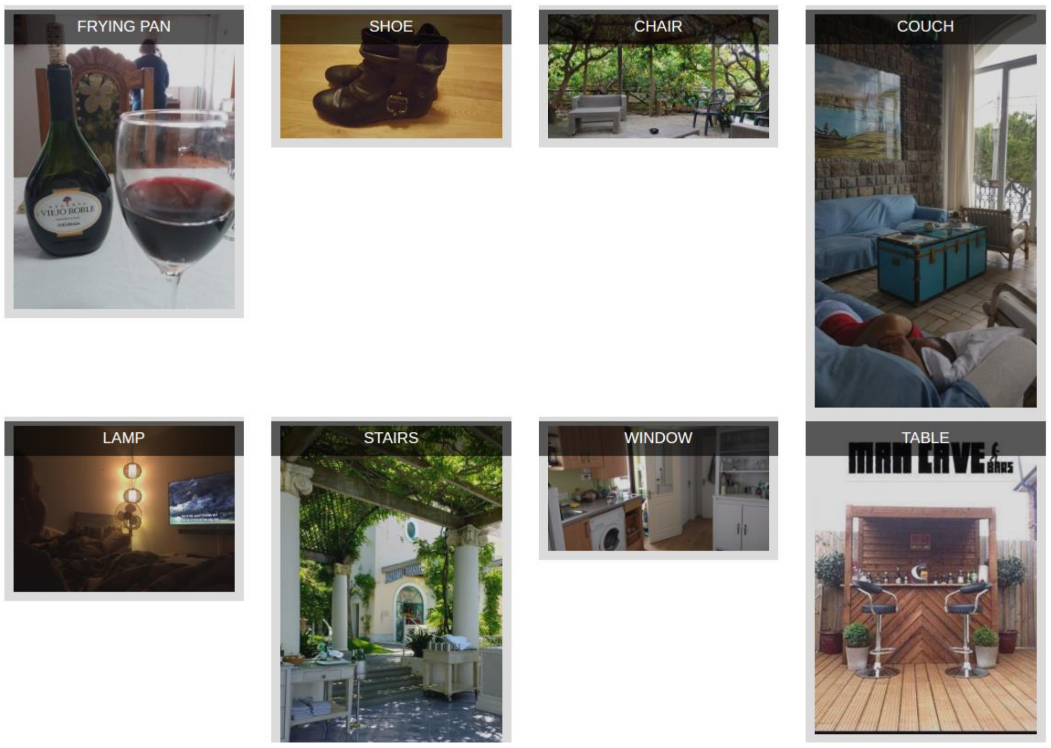


Figure 23. Interior and Object classification

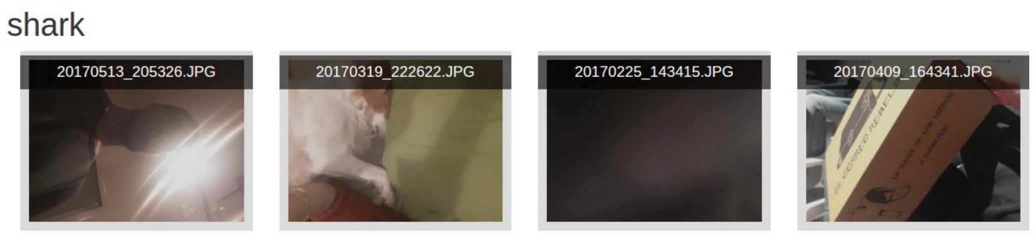


Figure 24. Shark Classification

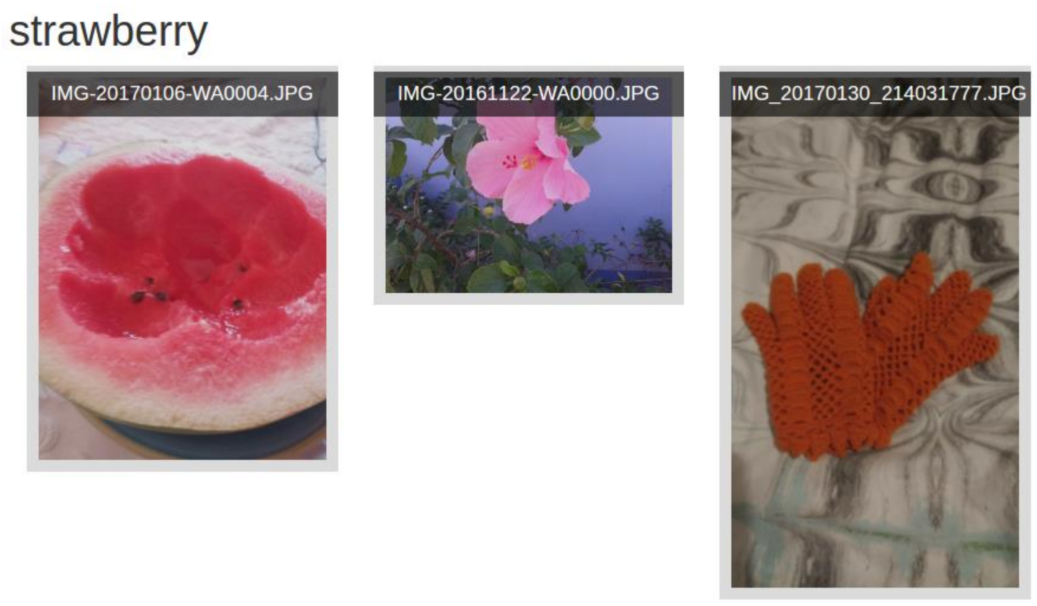


Figure 25. Strawberry classification

Here we can appreciate a clear difference in accuracy between Street-view and Architecture and Animal and Pet, as observed in our user study. Figure 21 shows that the first part of our system, the profile estimation, can be improved as it is currently classifying objects such as boxes as animals. This is due to a lack of confidence in the tags returned by Imagga, however, increasing the number of sub-topics, such as adding a category 'Box' to the topic Interior and Object, should improve the accuracy. We can also see in Figure 24 that the category 'Shark' receives images which are not supposed to be classified as animals, reinforcing the idea that the first part of our system can be improved. On the other hand, Figures 19 and 20 show a high degree of accuracy when classifying Street-view and Architecture.

5.3.3. Quantitative Evaluation

To better show this difference in accuracy, we performed a quantitative test, where we show the success rate of our system in the different topics. As the three systems (Google Photos, Eden Photos and our system) have such different topic organisations, we only performed this test on our system.

First off, we will be using 3 users:

	User 1	**User 2**	**User 3**
Street-view and Architecture	66	64	88
People and Family	32	68	34
Interior and Object	10	62	43
Animal and Pet	10	223	46
Nature and Landscape	93	86	137
Food and Beverage	0	23	43
Total:	211	526	369

Figure 26. User photo distribution

Figure 26 shows the users photo distribution, giving a rough estimate of each users preference. As can be observed, User 1 and 3 prefer urban and landscape photos, while User 2 has a strong bias towards animals.

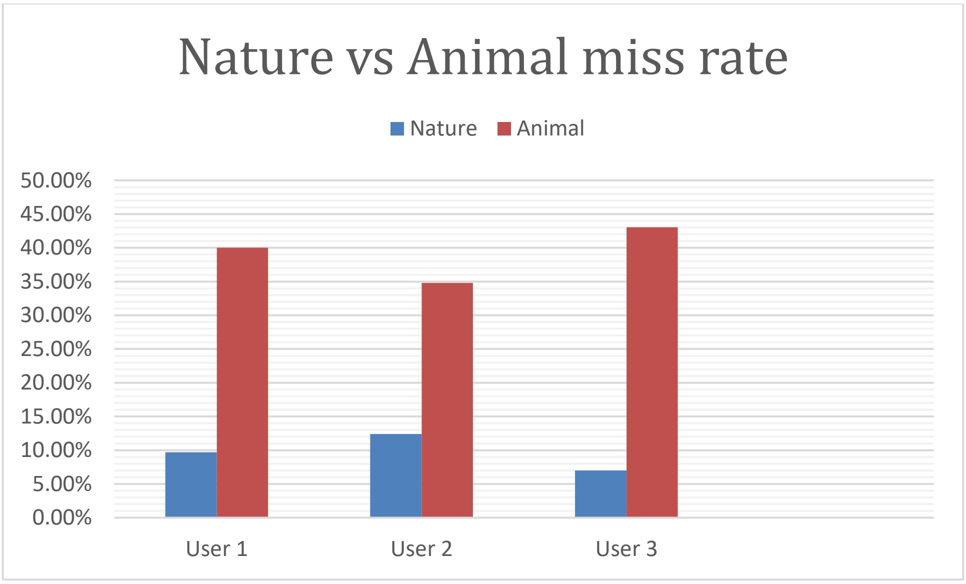


Figure 27. Nature vs Animal accuracy

Figure 27 shows the results of 3 users. The graph shows the percentage of photos the system failed to classify correctly. It is worth noting that the test considered an image to be wrong when it was placed in a wrong topic, such as a cat being considered Nature and Landscape. We did not consider an image wrong if its subtopic was wrong, such as a street being considered an alley.

As shown, Nature and Landscape has a considerable advantage over Animal and Pet, with roughly a 30% change in accuracy. This supports the user study observations, where many users commented on the lack of accuracy of Animal and Pet compared to the rest.

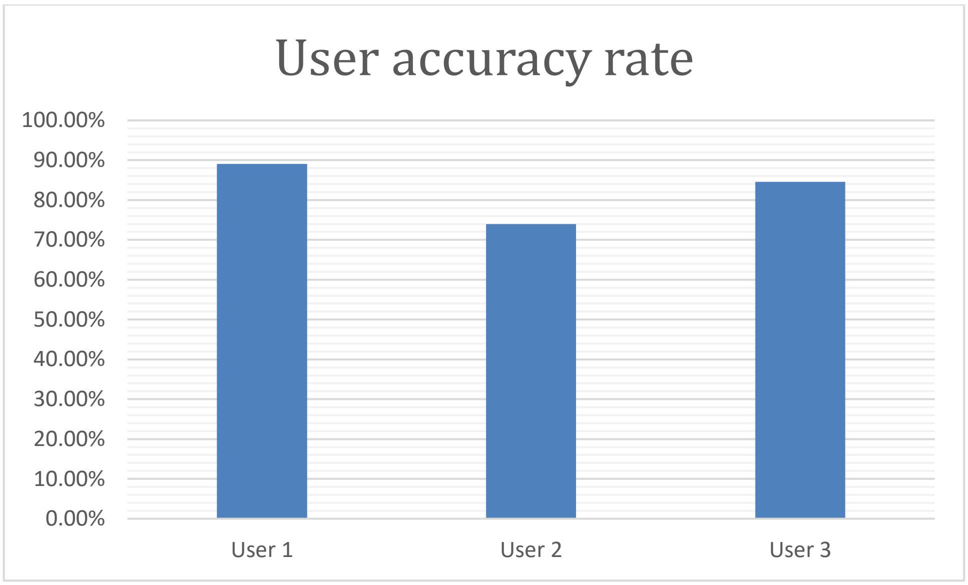


Figure 28. User accuracy rate

Figure 28 shows the results of the overall accuracy of the topics, making the same consideration as before: an image is wrong if its topic is wrong, not its sub-topic. Here we can observe that User 2, the user that has a large collection of animal photos also has the lowest overall accuracy, due to predominant motif being animals. On the other hand, the other two users, who show a more varied motif, have a better overall accuracy. For example, User 1 has an interest in hiking, hence the many photos of nature, while User 3 travels a lot, and therefore has many pictures of different cities and landscapes.

6. CONCLUSIONS AND FUTURE WORK

When we started this project, we defined many objectives, which have all been accomplished. Through extensive trial and error, proper parameters and thresholds have been applied to pLSA to achieve proper topic definitions. The Natural Language Toolkit has been a helpful tool to automatically assign the topics, providing the necessary functions to calculate the semantic similarity between the top N words of each topic and our defined topics.

Caffe, and more specifically **PyCaffe,** has been an essential tool throughout the fine-tuning process, providing a fast way to interact with the different networks, and understanding the framework has been essential to fulfil our goal. Appropriate datasets were built, thanks to publicly available datasets such as Places365, Food101 or ImageNet, as well as using scripts to scrape social media such as Instagram to fetch images with a more casual motif, showing day-to-day activities. In addition, thanks to these public datasets, pre-trained models of ImageNet and Places365 were used.

Thanks to the validation datasets we created, we were able to monitor our training process, making sure we avoided over-fitting. Once we had all our models trained, we could evaluate our system through user-studies, providing an external evaluation of real scenarios. These evaluations showed that users preferred our more specific classification, however, through observations we concluded that certain topics performed better than others. This could be due over-fitting, or a small dataset for this topic. Therefore, as future work, the training process could be expanded on, either by using a larger dataset, longer training periods or using other networks such as GoogleNet or ResNet.

The number of sub-topics could be expanded upon, in order to improve the accuracy of the system. One of our systems great advantages is the flexibility when adding either topics or sub-topics, it is easy to do, as we would only need to find or expand the dataset to incorporate these topics. Adding a topic such as 'Sport and Adventures' would be an easy task, as expanding the initial dataset with a user with these images would allow us to train pLSA with this topic. The same goes for our sub-topics, as adding a sub-topic to any model would just be a matter of adding the label and an appropriate amount of images to the dataset. For example, in the user study we observed that some users have photos of text or posters, and as we don't have a topic defined for these, they got classified as Animals. Adding a topic such as 'Text and Visuals', with its corresponding CNN, would solve this problem.

FIGURE AND TABLE INDEX

BIBLIOGRAPHY

[1] "Google Photos," [Online]. Available: https://photos.google.com/.

[2] "Eden Photos," [Online]. Available: http://edenphotos.io/?utm_source=BetaList.

[3] T. Hofmann, "Probabilistic Latent Semantic Analysis," 1999.

[4] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, L. a. Jonathan, R. Girshick, S. Guadarrama and T. Darrell, "Caffe: Convolutional Architecture for Fast Feature Embedding," 2014.

[5] C. Rosenberg, "Improving Photo Search: A Step Across the Semantic Gap," 12 June 2013. [Online]. Available: https://research.googleblog.com/2013/06/improving-photo-search-step-across.html.

[6] C. Guo and X. Tian, "Event Recognition in Personal Photo Collections".

[7] K. Karlsson, W. Jiang and D.-Q. Zhang, "Mobile Photo Album Management with Multiscale Timeline," 2014.

[8] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from," *Journal of Machine Learning Research 15*, 2014.

[9] A. Krizhevsky, I. Sutskever and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks".

[10] Stanford University, "Convolutional Neural Networks for Visual Recognition," [Online]. Available: http://cs231n.github.io/convolutional-networks/#case.

[11] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks For Large-Scale Image Recognition," 2015.

[12] M. Hollemans, "Convolutional neural networks on the iPhone with VGGNet," 2016. [Online]. Available: http://machinethink.net/blog/convolutional-neural-networks-on-the-iphone-with-vggnet/.

[13] Bird, Steven, E. Loper and E. Klein, Natural Language Processing with Python, 2009.

[14] Princeton University, "About WordNet," Princeton University, [Online]. Available: http://wordnet.princeton.edu.

[15] "WordNet Interface," [Online]. Available: http://www.nltk.org/howto/wordnet.html.

[16] B. Zhou, A. Khosla, A. Lapedriza, A. Torralba and A. Oliva, "Places: An Image Database for Deep Scene Understanding," 2016.

[17] L. Bossard, M. Guillaumin and L. Van Gool, "Food-101: Mining Discriminative Components with Random Forests," 2014.

[18] "ImageNet," 2016. [Online]. Available: http://www.image-net.org/.

[19] T. C. Andrew Gallagher, "Understanding Groups of Images of People," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[20] Bootstrap Core Team, "Bootstrap," [Online]. Available: http://getbootstrap.com/.

[21] S. Sawtelle, "Learning Curves and Validation Curves in Scikit-Learn," 2016. [Online]. Available: http://sdsawtelle.github.io/blog/output/week6-andrew-ng-machine-learning-with-python.html.

[22] J. H. Lau, D. Newman and T. Baldwin, "Machine Reading Tea Leaves: Automatically Evaluating Topic Coherence and Topic Model Quality," 2014.

[23] Y. J. Lee and K. Grauman, "Predicting Important Objects for Egocentric Video Summarization," 2015.