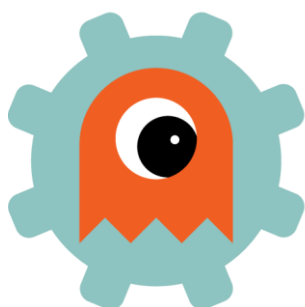


## I1.1 User GUIDE – Image module-methodology

### EUROSTARS E! 9701 - CASPAR 02



Combined Automated Semantic Processing Array v.2



# CASPAR 2

The Combined Automated  
Semantic Processing Array

## THE CONSORTIUM

No.	Partner Name	Logo
1	A Data Pro Ltd	
2	GeoImaging Ltd	

## Contents

The consortium.....	2
1. Introduction .....	4
2. ANNITA subsystem .....	4
2.1 Techniques used: ResNet .....	5
2.2 Training procedure for a new domain set.....	5
2.2.1 New images.....	5
2.2.2 Selection of domains .....	6
2.2.3 Training model.....	6
3. SPAS subsystem .....	7
3.1 Techniques used: Feature Descriptors and Descriptor Matching .....	7
3.2 Modification procedure for a new domain set.....	7
3.2.1 New images.....	7
3.2.2 Pre-process filtering.....	8
3.2.3 Post-process filtering .....	10
4. Discussion .....	12

## 1. INTRODUCTION

This document contains the steps a programmer should follow in order to train ANNITA subsystem to receive and classify a new domain and SPAS subsystem to receive and classify a new logo. ANNITA is currently trained to classify the following 9 classes:

- "Cars",
- "Containers",
- "Pills",
- "Hair",
- "Soft drink - bottles",
- "Soft drink - cans",
- "Snacks - Chocolate bars",
- "Snacks - Chips bars",
- "Cigarettes"

Adding a new class for ANNITA is not an easy task and entails intervening to python code, training again the model including samples from the new class. SPAS is currently able to classify the following 20 logos:

- Cars: "BMW", "Audi", "Suzuki",
- Cosmetics: "L'Oréal", "Schwarzkopf", "Syoss",
- Pharmaceuticals: "Aspirin", "Depon", "Augmentin",
- Soft drinks: "Sprite", "Coca-Cola",
- Chocolate bars: "Bueno", "Mars", "Twix",
- Chips bars: "Ruffles", "Lays", "Cheetos",
- Cigarettes: "Camel", "Marlboro", "Lucky Strike"

Adding a new logo for recognition in SPAS entails reformulation of filters and intuitive pre and post-process techniques

## 2. ANNITA SUBSYSTEM

ANNITA subsystem is about classification among different domains. The current domains are "Cars", "Pharmaceuticals", "Cosmetics", "Soft Drinks", "Snacks" and "Cigarettes". Since the above are just logical distinctions, based on the content, we

needed to split them further to “Containers”, “Hair”, “Pills”, etc. based on their shape in the images, finally forming the classes mentioned in the introduction.

## **2.1 Techniques used: ResNet**

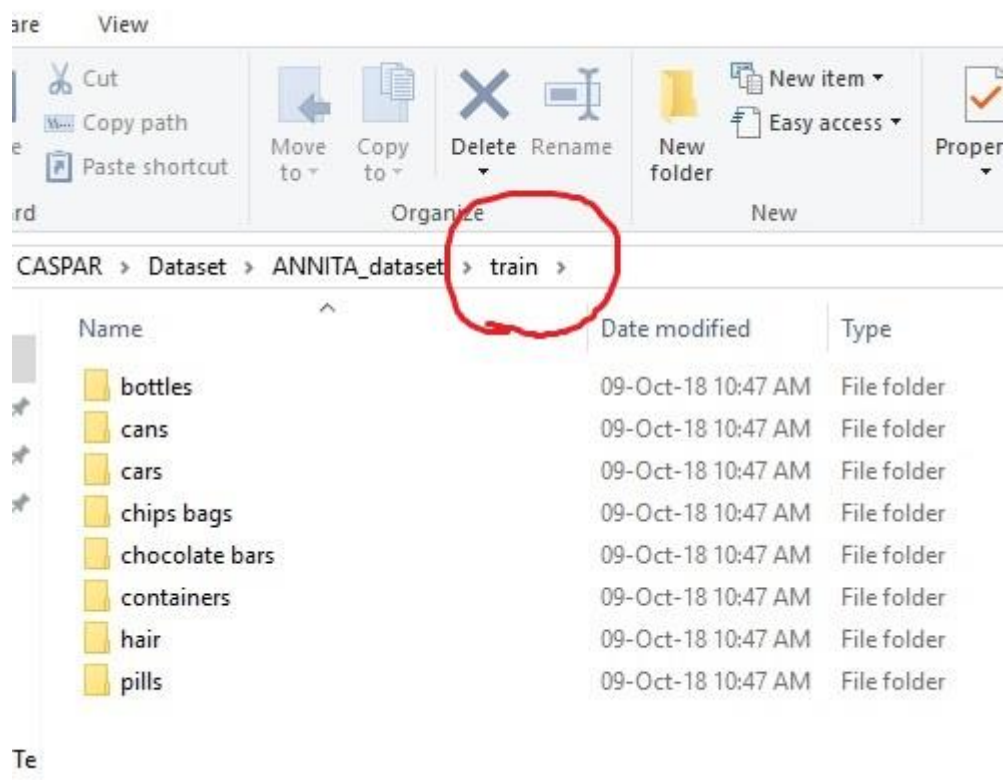
In order to train the classifier we used the Residual Network (ResNet) Architecture Design model that is designed to train very deep networks for Computer Vision. The implementation was made with the programming language Python. More details on ResNet can be found in [1]. For ResNet’s implementation, we used TensorFlow, an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and also used for machine learning applications such as neural networks. Details on Tensorflow can be found in [2] and [3].

## **2.2 Training procedure for a new domain set**

ANNITA subsystem is currently able to classify the categories mentioned in the introduction. For each new category added, the subsystem needs to be trained anew.

### **2.2.1 New images**

The first thing that needs to be done upon adding a new set of images is to add the new images to the system. To do that, the user needs to have a pool of at least 300 different images and split those in two folders. 80% of the images must be for training and the rest 20% will be for validation (Figure 1). Training images should be put in one folder and validation images in separate folder



**Figure 1: Splitting the new domain in training and validation dataset**

### 2.2.2 Selection of domains

Our methodology treats classification as a multidimensional problem, therefore we have to choose the domains which the classifier needs to use. We can select from two, up to all the domains (currently 8). As it is clearly understood, the more domains (classes), the more difficult the problem is. If we need to train the classifier for all the domains, we leave the dataset folder as it is in Figure 1. Otherwise we have to delete the subfolders of the categories we don't need.

### 2.2.3 Training model

There are several parameters that can be modified during training, it is advisable however not to be changed, as they have the values that after several tests seem to work better for the current 9 classes. After adding a 10<sup>th</sup>, 11<sup>th</sup> etc. class, they should also work just fine. If the initial domains change we will have a totally new problem. In this case, the programmer-user can freely experiment on finding the new optimal parameters. The model is trained for 10 epochs and each epoch contains 100 steps.

The training procedure output comprises the training and testing accuracy plot, as well as the training and testing loss functions

### **3. SPAS SUBSYSTEM**

SPAS subsystem is currently able to classify the categories mentioned in the introduction. For each new category added, the subsystem needs to be modified anew. We do not use the term "train" for SPAS, since it's not about neural networks or classifiers. SPAS is a feature matching platform based on pattern recognition methodology.

#### **3.1 Techniques used: Feature Descriptors and Descriptor Matching**

In *D4.2 Development of image clustering and classification tool*, we analysed the techniques used for SPAS. A brief summary is presented here for purposes of easier reference. Feature Descriptors are image features calculated for pixel neighborhoods that are used to describe certain attributes of the image. The matching compares the descriptors calculated in the target image with the ones calculated in a prototype image (source image) and signifies the matches.

In order to apply Feature Matching, we need first to extract the proper features and then use the right algorithms to combine them in order to have the desired results. Among all feature descriptors and descriptor matching techniques that were described in D4.1, we chose SIFT features and the FLANN based matcher because of the wider literature and experience, as well as the better results of the aforementioned methods.

#### **3.2 Modification procedure for a new domain set**

##### **3.2.1 New images**

In order to begin the matching procedure, we need to carefully choose a representative prototype image, the "source image" (Figure 2) with which the matcher will compare all test images. It has to be an image that best represents most of the logos found in the internet. Therefore it has to be a clean image that is not folded, washed out, or altered in any other way. Since most brands change their logo every few years, it's best to keep the most updated logo, keeping in mind that results will not be equally successful in case of older logos.



(a)



(b)



(c)

**Figure 2: Example source images (a) Bueno source image, (b) Twix source image, (c) Mars source image**

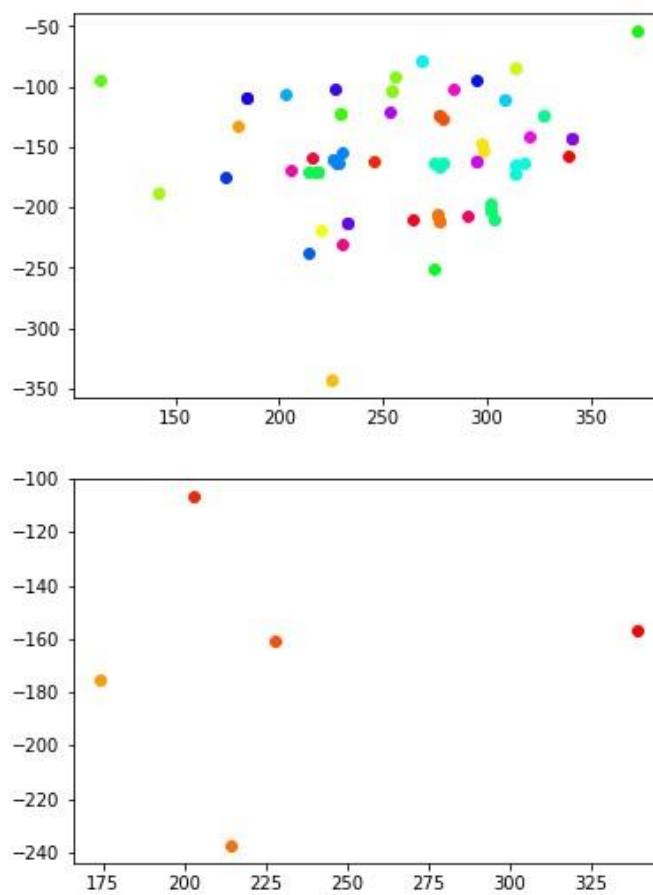
### 3.2.2 Pre-process filtering

Brand logos can be found anywhere in an image. They can be in a very small part of the image, or they can pretty much cover most of the image, they can contain letters, or have shapes that look like other things too. For example the BMW logo looks like a wheel and is often mistaken as wheel, or, even worse, car wheels can be conceived as BMW logos. Therefore, it is not possible to find a technique that successfully solves all



problems, leading us to confront each problem separately with filters. A pre-process filter is designed to limit the matcher's possible "successful" samples, clustering the possible samples. In Figure 3, we see the initial possible matches. As we see, there are many possible matches; some of them are obviously out of the scope. Excluding those from the picture and keeping the samples that cluster, we only keep the few cluster centers in the second graph of Figure 3. That way, the matcher can focus only to the important areas that are possible to comprise part of the target logo. The pre-process method in the code is "preprocess\_keypoint\_clusters (clusters, keep\_clusters, min\_elements, max\_elements)". The methods arguments are explained below:

- **clusters**: The number of clusters that the samples are going to form.
- **keep\_clusters**: The number of clusters that are going to be kept. In Figure 3, the "keep\_clusters" parameter was set to 5.
- **min\_elements**: The minimum number of samples each cluster will contain. For example you can have a cluster with just three samples, if min\_elements=3, or with more if the user raises this threshold
- **max\_elements**: The maximum number of samples each cluster will contain. For example you can have a cluster with 15 samples, if max\_elements=15.



**Figure 3: Samples before and after filtering**

### 3.2.3 Post-process filtering

Contrary to the pre-process filter that can be applied to all cases, the post-process filters are depended on each case and must be properly developed or modify to fit each logo. In order to create a good post-process filter, the user needs to run the matcher many times without any filters, in order to observe the results and decide accordingly what filters could be developed and applied.

### 3.3 Image-based identification technique

As mentioned before, it is more than often that a company changes its logo from time to time for marketing or other reasons. Despite the fact that the main idea of the logo usually remains the same (Figure 4), the aforementioned filtering techniques may become obsolete, leading to algorithm's inability to correctly identify the logo.



**Figure 4:** Quite similar logo change for Ruffles

For that reason we developed the option for the user to directly upload the logo he needs to have identified in the internet images. With this technique, the user is firstly asked to upload the desired logo and then the Feature Descriptors and Descriptor Matching methodologies do the job without any filterings.

Advantages:

- **Better generalization:** ANY logo uploaded by the user may be identified, not just the ones mentioned before (Suzuki, BMW, Ruffles, Mars etc)
- No need for filters and pre or post-process techniques
- **Better results:** while the filters work good, it's always a lot better to have the exact logo you are looking for as source. Otherwise the results are based on a possible obsolete logo and the filters are trying to catch the main edges and features of that logo

Disadvantages:

- **Extra effort needed by the user:** the user previously needed just to pick the desired logo from a (limited) drop-down menu. Now he needs to upload the desired logo himself
- User may upload a logo that is not properly isolated from background, therefore detailed instructions and an example must be provided

In the case of this image-based identification technique, the programmer does not need to make any change to the code, he just needs to disable all extra filtering pre and post-process techniques.

## 4. DISCUSSION

ANNITA takes some time to train, since we are talking about a neural network, however it doesn't need to have any parameters altered from domain to domain. Concerning SPAS, while the pre-process filter is universal and could be used in almost all cases, changing the parameters properly, the post-process filter should be implemented for each new domain. However, even without the post-process filter the matcher can produce good results.

## 5. REFERENCES

- [1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian SunDeep; "Residual Learning for Image Recognition", Microsoft Research, arXiv:1512.03385v1 [cs.CV], 2015
- [2] Dean, Jeff; Monga, Rajat; et al. "TensorFlow: Large-scale machine learning on heterogeneous systems" . TensorFlow.org. Google Research, 2015
- [3] <https://www.tensorflow.org/>