Visual Computing - Final Project Report
One-shot Object Detection

Duval Alexandre
alexandre.duval@student-cs.fr

El Malih David
david.el-malih@student.ecp.fr

Fillioux Leo
leo.fillioux@student-cs.fr

Maheu Benjamin
benjamin.maheu@student.ecp.fr

Abstract

Given real world query images of cereal shelves taken in grocery stores and a one-shot catalog dataset of stock photos of 62 types of cereals, we use one-shot classification and object detection techniques to retrieve the catalog products appearing in query images. We implement and compare two techniques for object classification on catalog images.

The first approach, based on the NLP concept of bag of words, uses a TF-IDF vectorization for each catalog image and assigns a query image based on how similar its TF-IDF feature vector is to one of the catalog images. The second approach, which is our custom classifier, assigns each query image descriptor to its closest neighbor in the catalog. It then uses a scoring technique to assign the final label.

Concerning object detection, we use descriptors and nearest neighbors matching to construct the existing object's bounding boxes, that we then refine through clustering, non maximum suppression and a clever merger step.

Introduction

Typical deep learning approaches for object detection and object classification often require a large amount of training data. Although this approach is potentially powerful, in the real world it may be impossible or at least very expensive to acquire such quantity of data. In this case, it can therefore be interesting to find alternative approaches, which do not present such requirements.

In this project, we first shed further light on object detection. In particular, we would like to identify cereal boxes in a grocery store shelf. This involves dealing with the various positions of the boxes, the lighting, their orientation, etc... We then focus on predicting the class of the identified cereal box using one-shot learning, meaning using only a single image per catalog class. To do so, we have created a catalog dataset containing one image per category and we try to match our identified box to a category of the catalog.

This project could be adapted to some of the following tasks:

- Perform inventories in supermarkets (our algorithm could be available without requiring any "learning" or "tuning" steps)
- Self Supervised Annotation - In case we have a big database of images to annotate, our tool could be useful as first step to perform - (we hope) - nearly 99% of the amount of work
- Contribute to improving "one-shot learning" research domain. We can then think about our algorithm being applied in fields where the amount of available data is limited such as medical imaging.

1. Problem definition

We would like to perform object detection and classification task in a one-shot context. In other words, having a catalog of images (each image representing one object), we would like to retrieve these objects on bigger images taken in grocery stores.

The detector takes as input a query image which is the whole image of the grocery store shelf and has to find where the relevant cereal boxes are located. It should not detect cereal boxes which are not in our catalog.

The classifier takes as input a crop of the query image, which has been selected by the detector. It assigns to this image a label, by selecting the best matching class ("best matching" is defined below).

Classification can be performed using multiple techniques. We've decided to benchmark 2 of them : Bag of Visual Words with TF-IDF weighting and Raw features matching (described below).

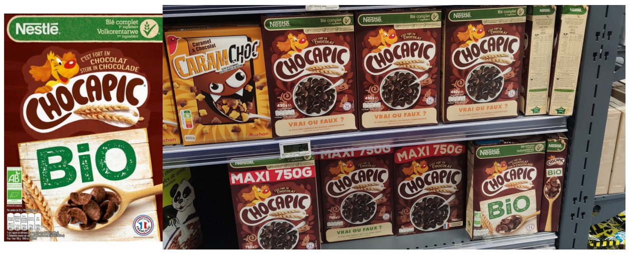


Figure 1. Example of a catalog image (left) and query image (right)

2. Related work

Most of papers we have found on the topic uses deep learning approaches and are therefore not within the scope of our research. Nevertheless, we have spotted one very interesting paper ([2]), which partly inspired us for the object detection task. It is particularly rigorous from a mathematical perspective and derive the bounding box's shape based on a series of optimisation equations, where the scale, label and center coordinates of the object are optimised. Similarly to our work, it first uses SIFT ([3]) on grayscale images to extract descriptors for both query and catalog images. It finds the nearest neighbors for each descriptor of a query image to later predict the bounding bounding box location and its label. Each descriptor is attributed a scale, the center of the object it relates to and a label, based on an average contribution of k-NN (weighted by distance between descriptors, space location and scale difference are penalised). These info are all used in their optimisation scheme to derive the final bounding boxes.

3. Methodology

3.1. Dataset

The dataset used for this problem is split into two parts. The first part is the catalog images of the cereal boxes, with one image per class. These images correspond to very clean, studio images of the boxes and were found on the internet. We only have one catalog image per class.

The second part of the dataset are the query images, meaning pictures (that we took ourselves) of the cereal section inside grocery shops. They contain different lighting, angles, occlusions, etc. The whole point of the project is to be able to identify and associate individual cereal boxes from the query images to catalog images.

3.2. Classification

The classification task can be formulated as follows : given a part of a query image, which corresponds to a single cereal box that exists in the catalog, find the corresponding catalog image. We have tried two techniques that we will benchmark : Bag of Visual Words with TF-IDF weighting and Features Matching (ours).

3.2.1 Bag of visual words

Given a set of catalog images, each depicting a single object of interest, we first resize it to a fixed size of 256 x 256. We than compute SURF ([1]) descriptors from regions extracted from sliding windows with these parameters : stride = 8px, sizes = (12, 24, 32, 48, 56, 64). Hence, we get a fixed number of descriptors per image (6144 precisely). We then have a database of catalog descriptors.

The next step is to clusterize this database into N clusters (N being our vocabulary size) using KMeans clustering algorithm. Each cluster center corresponds to a word in our vocabulary.

Now that we have a well defined vocabulary with the clusters, we want to describe our catalog images with these "words". We will build the IDF vector of these clusters. For each cluster, the associated IDF value is $log(\frac{\text{total number of images}}{\text{number of images containing a word in the cluster}})$.

For each catalog image, we evaluate the TF vector which is defined by the softmax of the similarity score to each cluster center, which we divide by the total number of descriptors. This gives us a score which indicates how present a certain "word" is in the image. Each catalog image will then be associated to a TF-IDF vector (by element wise multiplication of both vectors).

To classify a query image, we will evaluate its TF-IDF vector in the same way and will classify the image to the label of the TF-IDF which is to which it is the closest.

3.2.2 Features Matching (ours)

In our custom approach, the first step is the same as in the bag of visual words : resizing to 256 x 256 followed by feature extraction using SURF descriptors and sliding windows with these parameters : stride = 8px, sizes = (12, 24, 32, 48, 56, 64).

Once we have all key points and descriptors in the catalog images. We repeat the process for the query image. For each of these key points, we search for the nearest neighbor among the descriptors of the catalog images. Since we have a lot of descriptors to match, classical kNN would be very slow. So we perform an indexing step on catalog descriptors (using LSH or kd-trees) and the query descriptors are searched within the indexed structure.

Once it is found, we store the label as well as the confidence level of the match of a descriptor, meaning its quality. This means that for each query image, we will potentially get a few different labels. We then attribute the label based on a majority rule, weighted by the confidence level of each match.

More precisely, the confidence level of a match m is defined by $score(m) = e^{-(d/\sigma)^2}$, where d is the distance between both the descriptor and its nearest neighbor and $\sigma = 0.2$; the bigger the distance, the lower the score.

The score associated to a label l will be the sum of the matches corresponding to this label $\text{score}(l) = \sum_{m \in \{m.\text{label}=l\}} \text{score}(m)$. We then chose to associate to an image, the label which has the highest score:

$$\text{label} = \arg\max_{l}(\text{score}(l))$$

3.3. Detection

The framework for this task is similar to the Classification part. We start by creating features using SURF descriptor for both the catalog and query images. These features' location is specified manually at equivalent intervals on the image, and is not automatically determined by the algorithm because we would like an identical and representative number of descriptors per image. Once done, we focus on a single query image as the process is identical for all.

For each of query image descriptors, we find its nearest neighbor among all catalog features based on the euclidean distance. We extract the associated information, meaning the label C and scale S of the catalog image it belongs to, as well of the vector going from descriptor location and pointing towards the center of the object X. We also retrieve the confidence score cs of the matching and add all this data to the selected query image descriptor.

As first filtering step, we only keep the descriptors having a cs greater than 0.9. We then use all its data to predict the bounding box coordinates for this descriptor, which is supposed to approximate the location of the object it belongs to. We can think of this box prediction as a function $f(X, S, C)$, which is in fact simply determined by the true bounding box of the corresponding catalog image, that we manually created. In other words, we are able to adapt the bounding box created for the catalog image to which the nearest neighbor of the selected query image belongs to, by using the scale and the distance of the descriptor to the center of the cereal box.

In addition to producing too many bounding boxes, this approach is still quite inaccurate, which is why we perform a DBSCAN. This algorithm allows us to cluster the boxes that might refer to the same object. In short, it takes the center of the object to which the descriptor belongs to (found from X), and sort of expands clusters from core samples of high density. The main advantages are that we do not have to specify the number of clusters that we desire, and that it identifies noise data (e.g background - not cereal boxes). More precisely, for each bounding boxes, we define a feature vector $\text{features}(bbox) = (x_c, y_c, w, h)$, which is composed of the coordinates of the center of the bounding box and its width and height. We apply DBSCAN on these feature vectors. Hence we get clusters of bounding boxes, each cluster will then produce on unique "prototype" bounding box (which is more accurate).

Algorithm 1: DBSCAN

```
Result: A set of clusters
Input: $N$ centers of objects, cluster distance $\epsilon$, $M$:
  minimum number of points per cluster;
while Not all points are marked do
    Select an arbitrary unmarked point $P$;
    Extract neighborhood of $P$ using $\epsilon$ ;
    if neighborhood is greater than $M$ then
        mark as core point;
        form cluster;
    else
        mark as weak
    end
end
```

Looking at each cluster one at a time, we use a custom version of non maximal suppression to create refined bounding boxes. We choose the descriptor of the cluster presenting the highest confidence level cs (quality of matching for nearest neighbor), meaning that we are quite certain about its corresponding bounding box. We select all boxes of the cluster whose intersection over union metric with the first box is above a certain threshold: $iou > 0.5$. Say there are n of them. And we define the new bounding box (for this object) as the average coordinates $(x_i^{\min}, y_i^{\min}, x_i^{\max}, y_i^{\max})$ weighted by their confidence levels cs_i. Defining by $(x^{\min}, y^{\min}, x^{\max}, y^{\max})$ a box's coordinates,

$$\begin{pmatrix} x^{\min} \\ y^{\min} \\ x^{\max} \\ y^{\max} \end{pmatrix} = \frac{1}{\sum_i cs_i} \sum_i cs_i \begin{pmatrix} x_i^{\min} \\ y_i^{\min} \\ x_i^{\max} \\ y_i^{\max} \end{pmatrix}$$

Note that we only keep this last box and discard others.

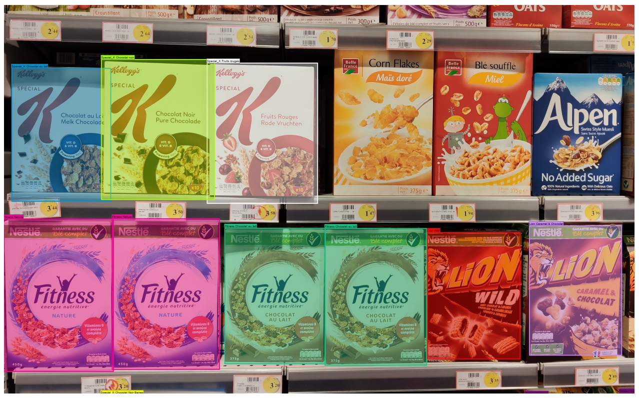


Figure 2. Example of results of detection

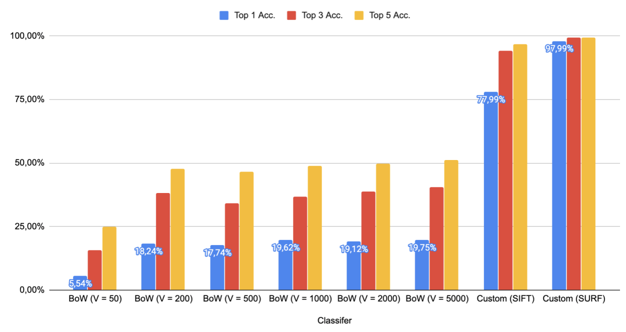


Figure 3. Top-k accuracy for multiple classifiers

As a result, we end up with a new single bounding box per cluster, that is per cereal box detected.

We define its label similarly to the classification task detailed above, using a majority rule weighted by the confidence level associated with each box. Note that the boxes below the threshold are simply discarded and all boxes are deleted at the end of this step, except of course the one newly created.

Finally, a last step consists in merging two bounding boxes predicting the same label if they overlap enough, because we consider that they in fact refer to the same object. This last step yields the final bounding boxes and draw them on the original query image, with a different color for each label (3.3).

4. Evaluation

4.1. Classification

The metrics that we used to evaluate the classification task are the top-1, top-3 and top-5 accuracy. For each query image, we will predict an ordered list of 5 possible labels. For top-k accuracy, we consider that the predicted label is correct if the true label is predicted in the list of the k first labels and we compute the accuracy by counting the number of "corrects" divided by the total number of query images.

We will compare the bag of visual words methods for multiple vocabulary sizes and our custom method (4.1).

The first comment we can make is that the accuracy increases with the vocabulary size, which makes sense as the more descriptors we can have for an image, the more accurately we will be able to describe it. However, the accuracy quickly reach their maximum value (we can barely see an increase from a 2000 word vocabulary to a 5000 word vocabulary). This best accuracy is relatively low, it reaches a 20% top-1 accuracy. Which is already quite an improvement, when we know that there is a total of 62 classes. (It is 12 times better than a random classifier).

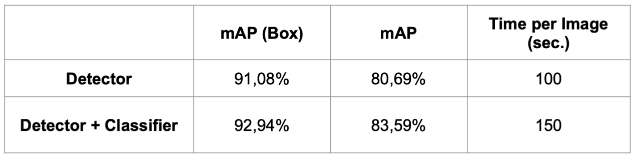
As we can see, our custom classifier scored near perfect detection top-1 detection (98%), when using the SURF feature descriptor. This is quite an impressive result and shows that one-shot classification can achieve more than satisfying results when the algorithm is well tuned. This also shows that this dataset is well fit to one-shot detection. Indeed, the cereal boxes in one class are all identical and the only thing that changes is the position, lighting and orientation in the query picture, this is very well handled by the feature descriptors that we chose.

4.2. Detection

For each predicted bounding box, we have the coordinates, the label as well as the score of the bounding box. For each of the bounding boxes, we will find its corresponding ground truth bounding box, and classify the prediction as either being a false positive or a true positive. The prediction is considered as a false positive if :

- it does not have an IOU greater than 0.5 with any ground truth bounding box
- the label is not correct
- the ground truth to which it is the closest is already taken by another prediction

From this information, we can evaluate the average precision, which is the area under the curve in the precision-recall graph. The mean average precision is the mean of all the average precisions for all images. This corresponds to the "mAP" column in our table. The "mAP (Box)" column corresponds to the exact same metric, where we did not take the label into account. This evaluates the performance of our model when it comes to detecting cereal boxes,



	mAP (Box)	mAP	Time per Image (sec.)
Detector	91,08%	80,69%	100
Detector + Classifier	92,94%	83,59%	150

Figure 4. mAP when using the detector and the detector together with the classifier

disregard of the class it belongs to. The ”mAP (Box)” score is therefore always greater than the ”mAP”. The graph 4.2 shows what the mean average precision graph looks like, with an area under the curve of 0.8359.

The second row corresponds to using the detector to find the correct bounding boxes and using the classifier to classify these found bounding boxes among the catalog classes. The first row uses the detector for both tasks, this means that the classification is done using a majority rule of the descriptors' labels on an image. We can see that even though the results are quite similar, our custom classifier performs better than just using the information of the labels in the descriptors.

Looking at the running time per image, both are quite long, but still satisfying if we want to use our algorithm as an automatic annotator to annotate data and then train some faster algorithms (such as YOLO or Faster R-CNN).

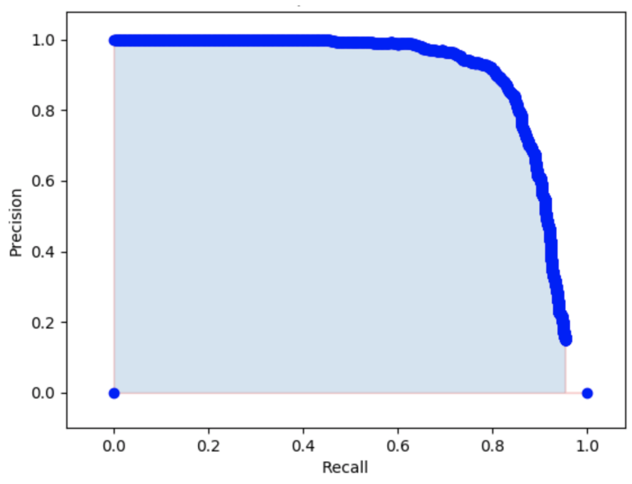


Figure 5. Mean average precision of 83.59%

Conclusion

One-shot learning is a quite promising field of research since the annotation of huge amount of data requires a lot of time and money.

Under some hypothesis such as a low intra-class variance (such as grocery stores products), classical computer vision algorithms (SIFT/SURF descriptors) have proven to be very effective in classification and detection tasks. In a one-shot database context, it is quite tough to undertake a ”learning” approach. Hence, these algorithms are well adapted to these kinds of tasks.

For cereal boxes dataset, we are able to reach 98% accuracy for classification and 92% mAP for detection but with a quite high computational cost. Nevertheless, our algorithms are a good way to perform automatic annotation on big datasets, make some small corrections by humans (which is less time consuming than annotating from scratch) and use the big dataset - which is now annotated - to train a Deep Object Detection algorithm.

References

[1] H. Bay. Surf: Speeded up robust features. Jun. 2008.
[2] L. Karlinsky. Fine-grained recognition of thousands of object categories with single-example training. May. 2018.
[3] D. G. Lowe. Distinctive image features from scale-invariant keypoints. Jan. 2004.