Deep learning for Logo Recognition: A Review

Mario Bošnjak
Faculty of Science, Dept. of Mathematics
University of Zagreb
Zagreb, Croatia
mario.bosnjak.007@gmail.com

Magdalena Modrušan
Faculty of Science, Dept. of Mathematics
University of Zagreb
Zagreb, Croatia
unicorn.magdalena@gmail.com

Dorian Stipić
Faculty of Science, Dept. of Mathematics
University of Zagreb
Zagreb, Croatia
dorian.stipic@gmail.com

Fran Domagoj Tomac
Faculty of Science, Dept. of Mathematics
University of Zagreb
Zagreb, Croatia
fd.tomac@hotmail.com

***Abstract*—This seminar work is a review of a deep learning method for logo recognition we employed for receipt logo recognition problem on a student competition 'Mozgalo'[1]. Our recognition pipeline is organized as follows. First, logo region proposals are obtained by the means of Selective Search algorithm and are then given to the Convolutional Neural Network (CNN) pretrained specifically for logo classification task. Experiments are conducted on test sets used in the aforementioned competition collected by 'Microblink'.[2]**

***Index Terms*—Logo recognition, Deep Learning, Convolutional Neural Networks, Selective Search for Object Recognition**

I. Introduction

A. Motivation

Logo recognition is a problem that frequently emerges in a wide range of applications, such as augmented reality, copyright infringement detection, automated computation of brand-related statistics on social media and more. Due to the nature of these applications it is of great importance that logo recognition algorithm allows for fast real-time usage and to attain high accuracy, so that it can replace humans utterly.

Logo recognition is an instance of a problem in the field of computer vision. Historically, this problem was addressed with manually engineered features such as textons or HOGs, but recently the rise of more powerful computing resources led to black-box approaches. State of the art results were obtained by Convolutional Neural Networks so we base our solution on one as well.

II. Problem statement

The training set for the logo recognition problem this paper addresses consists of the total of $45,000$ receipt photographs. Each receipt contains one and only one of the possible 25 store logos and these logos are more or less evenly distributed among the data set; that is to say that there are between $1,445$ and $2,051$ photographs containing any particular logo in the data set. The training set is labeled in the sense that it is made up of 25 folders each containing images for exactly one of the 25 store brands. The objective is to propose an algorithm that would classify a photograph of any receipt into 26 classes; either as belonging to one of the 25 known stores or to the artificial class *Other*. Photographs are colored (RGB images) and they come in variety of sizes, although somewhere in the neighborhood of $1,100 \times 600$ pixels.

Fig. 1. Examples from the training set. These suggest that features such as lighting, amount of background noise and dimension differ from image to image.

III. Method

The basic idea of our approach was to first localize the logo and then classify it accordingly. Localization is no trivial task and we accomplish it via the Selective Search algorithm ([2]) which is also the first part of our recognition pipeline. However, this algorithm does not yield a singular region proposal containing the logo, but many region proposals it assesses likely to contain an object. After obtaining these proposals, for the purposes of training our model, some of them are discarded and others are given to the CNN. Deciding upon which proposals to use is done using ground-truth logo

[1]For more details go to https://www.estudent.hr/category/natjecanja/mozgalo/#radionice-i-tutoriali_tab

[2]All codes are available at https://github.com/PMF-StrojnoUcenje2018/NuzniDovoljni

annotations and the Intersection over Union (IoU) measure. These concepts, together with Selective Search algorithm, preprocessing, data augmentation and CNN architecture used are covered in detail in the following subsections.

Fig. 2. Simplified logo classification pipeline

A. Preprocessing

Prior to running Selective Search algorithm, each image is resized to dimensions of 400×220 pixels. These dimensions were selected by finding the average height and width of images in the training set and the ratio between the two. Images were then scaled with respect to this ratio, while simultaneously aiming to avoid too great of a loss of information and decreasing Selective Search running time to around 0.5 seconds per image.

B. Ground truth logo regions

The method we review here makes use of ground-truth specifying logo position and class for each image. Logo position is represented by a rectangular crop containing the logo in total, but it may also, depending on the perspective of the image, contain a part of the background. As ground-truth logo regions were not given in the training set we acquired them in a semi-automatic fashion. First, from each class, we chose a photograph taken under good conditions to ensure good visibility of the logo which we then cropped as a template or a patch. Next, images were gray-scaled and for every image in that class, template matching was performed to extract the wanted logo regions.

Template matching is executed by sliding the center of the patch along the search image and calculating normed correlation-coefficient between the template (patch) and current part (subimage) of the search image. The sought for region is thus obtained by taking the part with the highest correlation-coefficient with the template. Precisely put, let T and I denote the template and the search image respectively. Furthermore, let w and h stand for width and height of the template and $T(x,y), I(x,y)$ for (x,y)-pixel color value in template and search image. Template matching method yields matrix R such that $R_{x,y}$ holds the normed correlation-coefficient between template centered at (x,y) pixel and appropriate part of the search image. Matrix R is obtained as follows.

$$R_{x,y} = \frac{\sum_{x',y'} \left((T'(x',y') I'(x+x', y+y') \right)}{\sqrt{\sum_{x',y'} T'(x',y')^2 \sum_{x',y'} I'(x+x', y+y')^2}}$$

where

$$T'(x',y') = T(x',y') - 1/(wh) \sum_{x'',y''} T(x'',y'') \ ,$$

and

$$I'(x+x', y+y') = I(x+x', y+y') - 1/(wh) \sum_{x'',y''} I(x+x'', y+y'') \ .$$

Template matching, however, at times generated wrong logo region proposals and faulty images had to be removed from the training set manually. After these interventions, the training set consisted of about $33,000$ images with precise labels and logo regions annotations. What is more, not every class contained the same number of images, because some classes imposed more of a challenge for template matching than others.

C. Selective Search

Even though it was not always treated as such, image segmentation problem is hierarchical in nature due to spatial relation of objects in an image. Furthermore, as there are many conflicting reasons why certain regions of an image should be grouped together, for instance, on the basis of texture or color, there may not be a generic solution for a segmentation problem at all. This calls for an algorithm that would increase the diversity of grouping approaches. Moreover, objects tend to appear in any scale in an image which also imposes a demand on an algorithm. Also, as it is only a part of object recognition framework, the segmentation algorithm should also be fast to compute not to impede upon overall performance of the object recognition algorithm. The Selective Search algorithm design takes care of all of these considerations: hierarchical nature of the problem, capturing all object scales, diversification and fast computation.

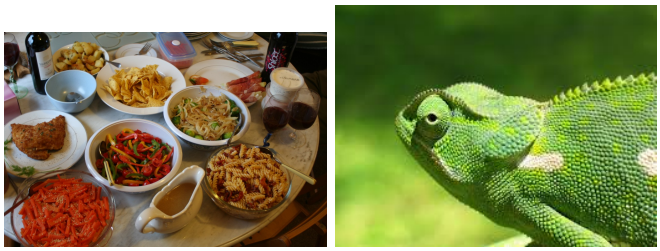


Fig. 3. In the first image the hierarchical nature of the problem is evident: salad and spoons, for instance, are *inside* the bowl which stands *on* the table. Chameleon image shows there are plentiful of reasons image region can form an object: chameleon can be distinguished from its surroundings by texture, but not by color, so depending on the context and discrimination criterion used we may end up with different object recognitions.

Selective Search algorithm starts with a set of small regions which, ideally, do not span multiple objects and is obtained via the fast method of Felzenszwalb and Huttenlocher [3]. The hierarchical grouping procedure then continues as follows. First, the similarities between all neighboring regions are calculated. The two most similar regions are then grouped together, and new similarities are calculated between the resulting region and its neighbors. The process continues in iterative fashion until the whole image becomes a single region. Below is the algorithm of the general procedure. We didn't implement it

directly but it has such an important role in our approach, therefore we decided to describe it precisely.[3]

Algorithm 1 Hierarchical Grouping Algorithm

Require: color image
Ensure: Set of object location hypotheses L
  Obtain initial regions $R = \{r_1, \ldots, r_n\}$ using [3]
  Initialize similarity set $S = \emptyset$
  for all Neighboring region pairs (r_i, r_j) **do**
    Calculate similarity $s(r_i, r_j)$
    $S = S \cup s(r_i, r_j)$
  end for
  while $S \neq \emptyset$ **do**
    Get the highest similarity $s(r_i, r_j) = \max(S)$
    Merge corresponding regions $r_t = r_i \cup r_j$
    Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$
    Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$
    Calculate similarity set S_t between r_t and its neighbors
    $S = S \cup S_t$
    $R = R \cup r_t$
  end while
  Extract object location boxes L from all regions in R

To ensure that diversification demand is met, several methods are employed: using variety of color spaces with different invariance properties, using different similarity measures s_{ij}, and varying the starting regions.

Complementary Similarity Measures. Four complementary, fast to compute, measures are defined. All of these measures are in range $[0, 1]$.

- *Color similarity measure*
  For each region one-dimensional color histograms are obtained for each color channel using 25 bins which leads to a color histogram $C_i = \{c_i^1, \ldots, c_i^n\}$ for each region r_i, with $n = 75$. The color histograms are then normalized using the L_1 norm. Finally, similarity is measured using the histogram intersection

$$s_{color}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i, c_j^k).$$

  The color histograms are efficiently propagated through the hierarchy by

$$C_l = \frac{\text{size}(r_i) \times C_i + \text{size}(r_j) \times C_j}{\text{size}(r_i) + \text{size}(r_j)}$$

  where size of the resulting region is given with:

$$\text{size}(r_l) = \text{size}(r_i) + \text{size}(r_j)$$

- *Texture similarity measure*
  Texture is represented using SIFT[4]-like measurements. Gaussian derivatives in eight orientations are taken using $\sigma = 1$ for each color channel. For each orientation for each color channel a histogram is then extracted using a bin size of 10. This leads to a texture histogram $T_i = \{t_i^1, \ldots, t_i^n\}$ for each region with dimensionality $n = 240$. Again, texture histograms are normalised using the L_1 norm, and similarity is calculated using histogram intersection:

$$s_{texture}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i, c_j^k)$$

  Also, texture histograms are propagated through the hierarchy in exactly the same way as the color histograms.

- *Size similarity measure*
  This measure ensures regions in S, that is to say, ones that have not yet been merged, are of similar sizes throughout the algorithm. This means that small regions are encouraged to merge early. Size measure is defined in the following way:

$$s_{size}(r_i, r_j) = 1 - \frac{\text{size}(r_i) + \text{size}(r_j)}{\text{size}(im)}$$

  where size(im) denotes dimension of the image.

- *Goodness of fit measure*
  This measure assesses exactly how well regions r_i and r_j fit into each other. The idea is to fill the gaps: if r_i is contained in r_j it is only logical to merge them first to avoid any holes. On the other hand, if r_i and r_j are hardly touching each other it is not likely they form a meaningful region. Let us define BB_{ij} to be the tight bounding box around r_i and r_j. Now we can define the wanted measure as the fraction of the image contained in BB_{ij} which is not covered by r_i and r_j:

$$s_{fill}(r_i, r_j) = 1 - \frac{\text{size}(BB_{ij}) - \text{size}(r_i) - \text{size}(r_j)}{\text{size}(im)}$$

Final measure is a combination of the above four measures:

$$s(r_i, r_j) = a_1 s_{color}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j) \ ,$$

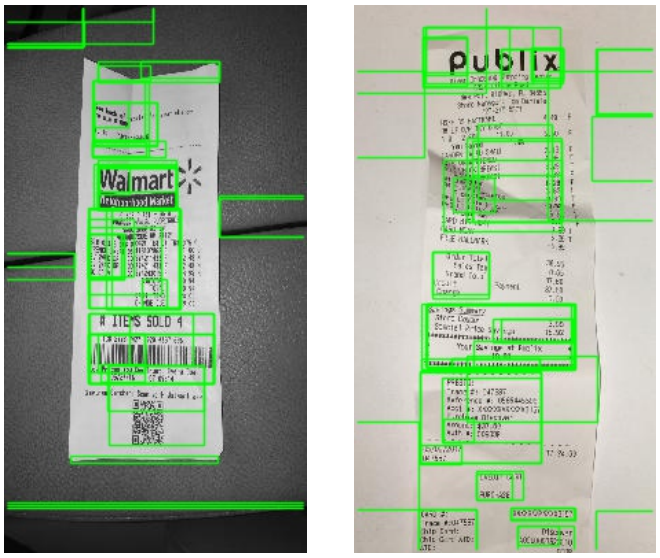
where $a_i \in \{0, 1\}$.

Complementary starting regions. This is another diversification technique, and is achieved indirectly by varying the color spaces.

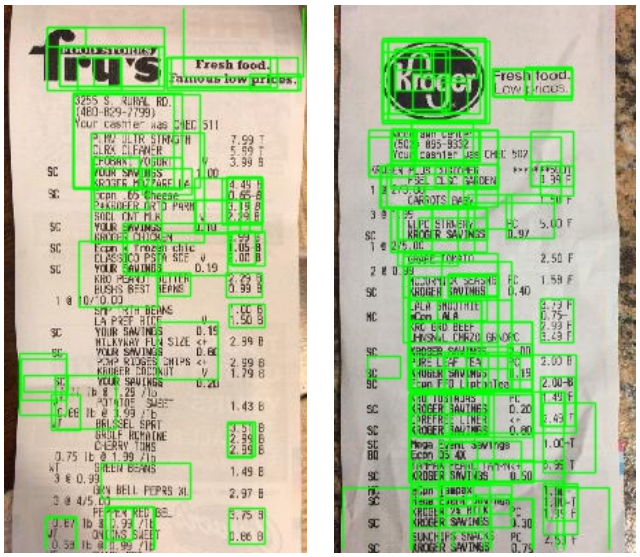
Combining locations. Ideally, one would like object hypotheses to be ordered in such a way that the locations which are most likely to be an object come first. In [3] authors chose to order the combined object hypotheses set based on the order in which the hypotheses were formed with certain randomness included to make it more computationally feasible.

[3] More can be found reading [2]

[4] SIFT stands for "Scale Invariant Feature Transform" and is an algorithm for detecting and describing local features in images. More can be found on Wikipedia: https://en.wikipedia.org/wiki/Scale-invariant_feature_transform



(a) (b)



(c) (d)

Fig. 4. Object hypotheses Selective Search makes on some images from the train set; the algorithm proposes handful of regions that contain at least a part of the logo for every image.

D. Logo/No-Logo annotations

Selective Search variant we used makes around 300 object hypotheses for a RGB given image resized as described in A. The idea now was to discriminate between regions which contain at least a part of the logo, and those that do not. This is achieved by calculating Intersection over Union (IoU) measure. Figure 5 describes how IoU measure is obtained for any two given regions. Amongst the regions Selective Search provides, those having IoU with ground-truth logo region greater than or equal to 0.4 [5] are labeled as containing a logo and others are labeled as background (i.e. not containing a logo). The hypothesis is that adding a 'Background' class will make the CNN more precise in discriminating between logo and background object proposals, i.e. between useful and not useful information. Selective Search labeled on average $20,000$ logo regions for each class[6] ($10-20$ regions out of 300 per image) we hypothesized that in order to prevent a classification bias of the CNN class balancing[7] is important so each class that had less than $15,000$ logo regions was augmented. The 'Background' class being particularly numerous[8] was reduced to approximately $400,000$ regions. Below average ratios for each class are shown[9].

[5]It is important to notice that this method selects as containing a logo, regions where the logo doesn't have to be localized precisely, the method is thus naturally more robust.

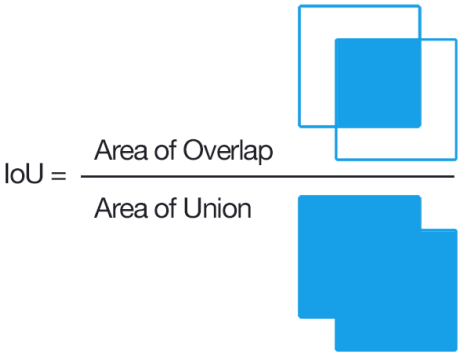


Fig. 5. Intersection over Union measure for any two given regions is easily calculated.

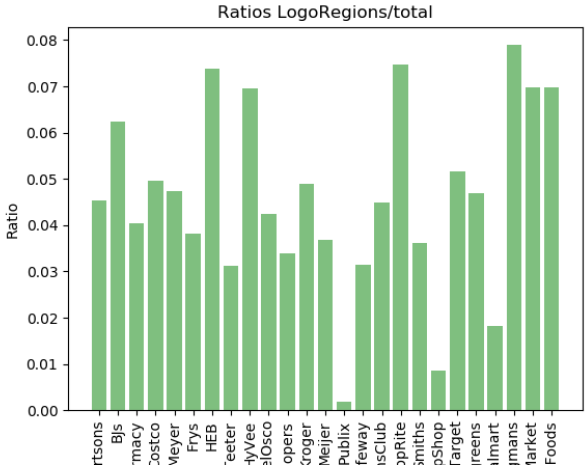


Fig. 6. Average Ratios between number of regions intersecting with ground truth logo and total number of region proposals, it can be seen clearly in which class Selective Search struggles the most.

[6]On three classes Selective Search performed poorly and they had beetwen $1,000-5,000$ regions: Publix, Stopshop, Walmart

[7]No class is more than twice as big as any other class

[8]Around $280 \times 33,000 \sim 9$million

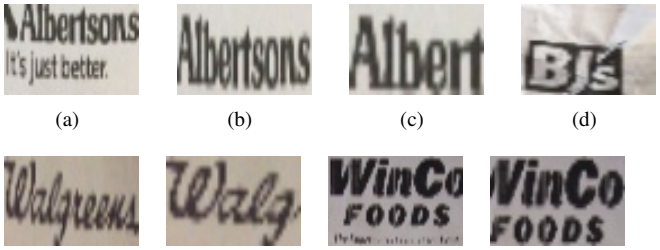
[9]It will be seen in Results how this is correlated with the overall performance of the method for each class

E. Data Augmentation and feeding the CNN

After the region proposals were obtained via the Selective Search, resizing, gray-scaling, data augmentation and, finally, CNN training followed.

At first the region proposals fed to the CNN were resized to 32×32 pixels, and then we also tried resizing to 40×60 pixels for training the network. The latter choice was made due to the notion that logos are mostly of rectangular shape larger in width than in height.

In order to mend the dis-balance between classes, some training examples were augmented. The main idea was to replicate possible real life situations and alterations that can occur (to logo regions) in a photograph: random distortions, skews, rotations (up to 5°) and zooms were added as spatial transformations and random brightness and contrast as lighting. Finally, the regions were gray-scaled prior to being fed to CNN.[10]



(a) (b) (c) (d)

(e) (f) (g) (h)

Fig. 7. Some examples of object proposals made by Selective Search which had IoU greater or equal to 0.4 with ground-truth, it is clear that some proposals contain only parts of the actual logo, some other contain larger regions than the logo itself.

F. Convolutional Neural Network

It appeared to us that CNN used by [1], and shown in table I, had a bottleneck (which is demonstrated in [4] not to be ideal for learning) between the third pooling layer and the first fully connected one (layers 9 and 10) caused by too great of a difference in number of units. We tried to improve upon this issue by adding more fully connected layers in order to make the architecture of the Net more "smooth". The training set was of sufficient size to avoid over-fitting and so the proposed CNN used by [1] was made overall larger. Our final CNN architecture is shown in table II. We trained the CNN for 15 epochs and learning rate of 0.001 we then started to reduce the learning rate by factors of 10 and trained for as many epochs as it was needed, until convergence. The Net stabilized its accuracy around 99% which was good enough for our purposes.[11]

[10]The logos are completely or partially black

[11]We didn't look for a perfect accuracy of 99.9%, although we could, because the evaluation process was probabilistic and didn't need a perfect Net

TABLE I
CNN ARCHITECTURE USED IN [1]

	Layers
1	Conv 32 filters of 5×5
2	ReLu activation
3	Max pool with stride of 2
4	Conv 32 filters of 5×5
5	ReLu activation
6	Average pool with stride of 2
7	Conv 64 filters of 5×5
8	ReLu activation
9	Average pool with stride of 2
10	Fully connected of size 64
11	Fully connected of size 33
12	Softmax

TABLE II
TYPE 1 OF CNN ARCHITECTURE WE USED

	Layers
1	Conv 50 filters of 5×5
2	ReLu activation
3	Max pool with stride of 2
4	Conv 50 filters of 5×5
5	ReLu activation
6	Average pool with stride of 2
7	Conv 100 filters of 5×5
8	ReLu activation
9	Average pool with stride of 2
10	Fully connected of size 1000
11	Fully connected of size 1000
12	Fully connected of size 200
13	Fully connected of size 26
14	Softmax

Our first try was CNN architecture shown in figure III whith regions of size 32×32 pixels as inputs. Using 60×40 as input CNN in table II was used.

TABLE III
TYPE 2 OF CNN ARCHITECTURE WE USED

	Layers
1	Conv 32 filters of 5×5
2	ReLu activation
3	Max pool with stride of 2
4	Conv 32 filters of 5×5
5	ReLu activation
6	Average pool with stride of 2
7	Conv 64 filters of 5×5
8	ReLu activation
9	Average pool with stride of 2
10	Fully connected of size 256
11	Fully connected of size 52
12	Fully connected of size 26
13	Softmax

IV. EVALUATION

The test set we evaluated our method on consisted of $10,000$ receipt images around $5,000$ of which belonged to the 'Other' class, and the other 25 classes were uniformly distributed in the other half of the test set (~ 200 images for every class in the other half). Testing pipeline is organized as follows.
The test image is first resized to dimensions of 400×220

pixels, as was the case for training images. Then, selective search is ran on the image and object region hypotheses are obtained which are then gray-scaled and resized to 40×60 pixels (or 32×32 pixels depending on the CNN). These processed region proposals are then fed to the trained CNN which, for each region, yields a 26×1 stochastic vector. Here, a hyper-parameter is introduced; a threshold[12] on basis of which final verdict is made from the stochastic vectors. One here can try different approaches and we indeed tried many:

- **Average-Probability:** From the prediction vectors at disposal, mean confidence (probability) for each class is extracted and maximum average confidence is considered; if it is greater than a given threshold, prediction is made. Otherwise, 'Other' class is predicted.
- **Most sure Class:** From the prediction vectors at disposal, one can calculate the maximum probability out of all classes and all region proposals i.e. the region for which the CNN is the most sure it comes from a particular class ('Background' excluded); again if it is greater than a given threshold, prediction is made. Otherwise, 'Other' class is predicted.
- **Number of proposals:** From the prediction vectors at disposal, one can calculate the ratio between the most frequent predicted class for the regions [13] and the total number of regions and make a decision confronting it with the mean ratio extracted for each class from the training set or similar methods using the number of proposals for each class.
- **The more the better** A combination of the methods above.

V. Results

On the test set described in IV given by competition organisers using region proposals re-sized to 32×32 pixels we achieved results showed in VI. Training was performed on the train set augmented as described in previous subsections, and CNN architecture shown in Table III. All the thresholds are found using this small process: observing the data and manually setting "good" guesses, local grid-searching around the guessed values. Exhaustive grid-search wasn't possible because the parameters to choose where too many, and we didn't have enough computing power to do so.

TABLE IV
32x32 Pixels proposals with Background Class

Accuracy	0.455
F1 Score Macro	0.554
F1 Score Micro	0.455
F1 Score Weighted	0.443

[12]Can be set manually or automatically with a grid search among reasonably good choices.

[13]After the 'Background' class, which is expected to be the most frequent always

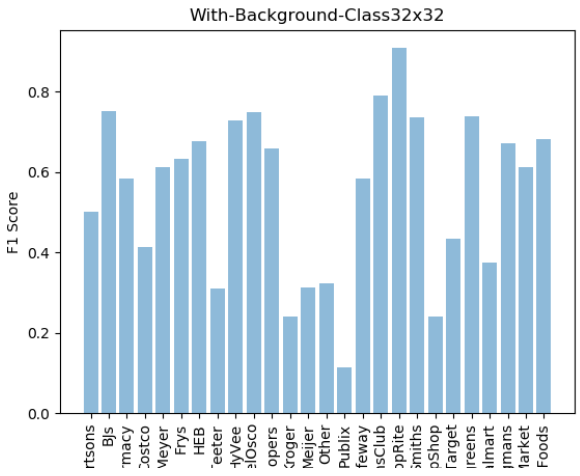


Fig. 8. F1 scores and accuracy for each class. 32×32 pixels regions

Results achieved by re-sizing the regions to 40×60 are far better as expected and are shown in Table V.

TABLE V
40x60 Pixels proposals with Background Class

Accuracy	0.639
F1 Score Macro	0.714
F1 Score Micro	0.639
F1 Score Weighted	0.657

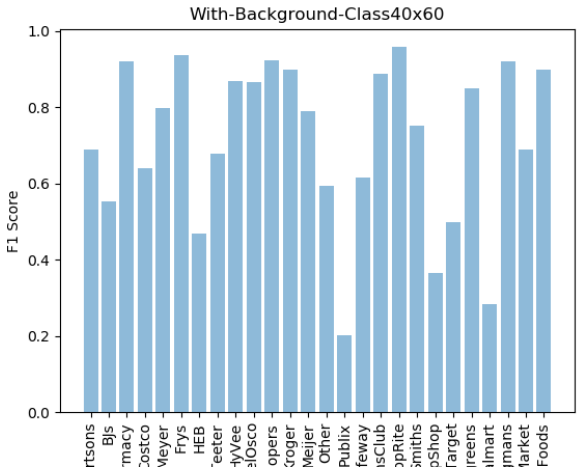


Fig. 9. F1 scores and accuracy for each class. 40×60 pixels regions

We also tested the importance of the extra background class, and trained a CNN without it. The results obtained are quite poor, the reason is: the CNN is trained solely on logo regions but in testing time the Net has to evaluate many different regions and only about 5% of them are logos, therefore the Net basically encounters a lot of regions it doesn't know to which class they belong. For this reason choosing the various thresholds is also extremely difficult and complex.

TABLE VI
40X60 PIXELS PROPOSALS WITHOUT BACKGROUND CLASS

Accuracy	0.449
F1 Score Macro	0.456
F1 Score Micro	0.449
F1 Score Weighted	0.452

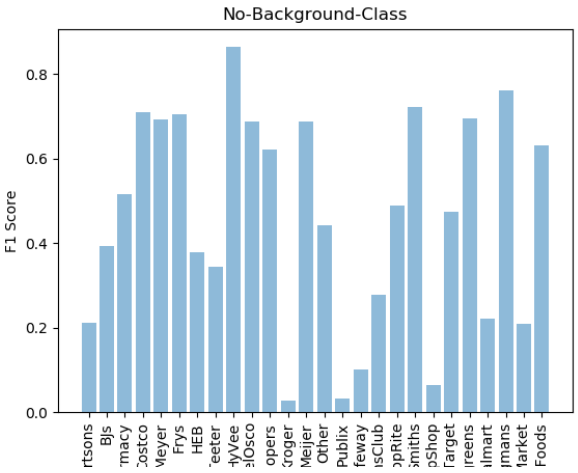


Fig. 10. F1 scores and accuracy for each class. 40×60 pixels regions, Without background class

VI. OTHER POSSIBLE METHODS

Great advances have been made in the field of computer vision and one has many methods at disposal. One approach proven to give great results during the competition was to use end-to-end pipeline. It consists of using state of the art, very deep and complex, CNNs such as the one described in [5] or [6]. However one weakness of such approach is that it fails to generalize the problem i.e. to recognize more than one logo in an image, whereas the method we used can be naturally implemented also in those situations, as well.

VII. FURTHER IMPROVEMENT OF THE METHOD

The method we used is quite complex and there are many possibilities of improvement. To start with, a different CNN architecture might give better results. Also a different and more refined data preprocessing can be done such as contrast normalization obtained by subtracting the mean and dividing by the standard deviation extracted for each image or from the whole dataset. Also as described in [7] it is possible to train the CNN with the so called hard or adversarial examples, which exploit intrinsic weaknesses of neural networks thus giving overall better performance. It is important to notice that the results we obtained can be a lot better, using exactly the same CNN and method the reason is we didn't purposely spend a lot of time for finding the best possible thresholds because it wasn't the purpose of this paper. We showed that the method is flexible and potentially applicable in an industrial environment.

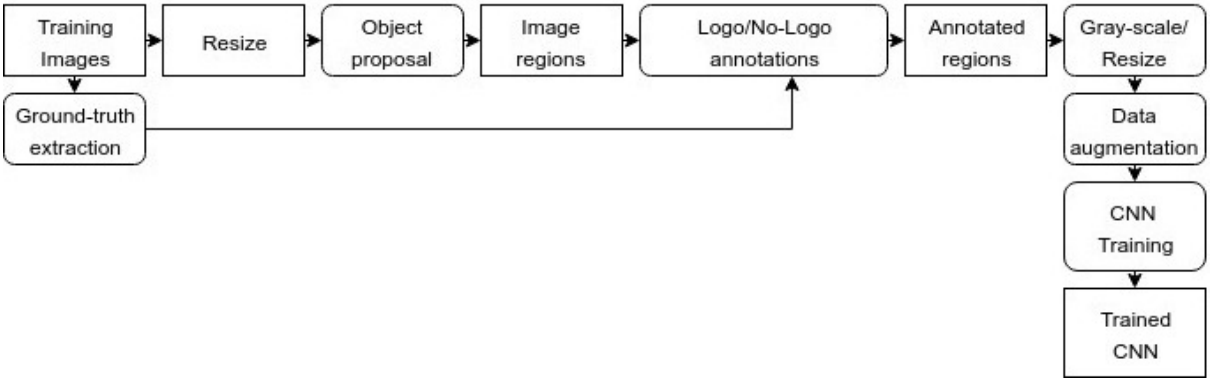


Fig. 11. Detailed training pipeline

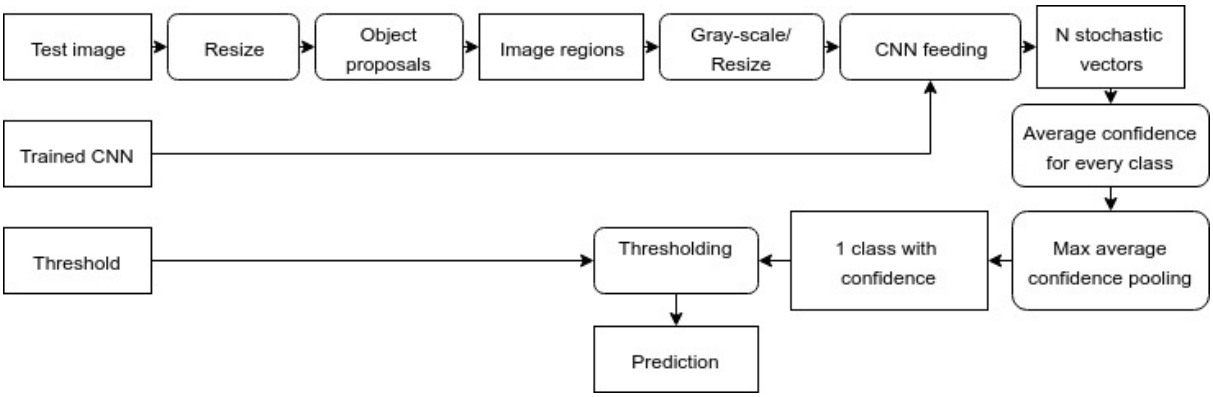


Fig. 12. Detailed testing pipeline

REFERENCES

[1] Simone Bianco, Marco Buzzelli, Davide Mazzini, Raimondo Schettini, "Deep Learning for Logo Recognition", DISCo-Univesitá degli Studi di Milano-Bicocca, 2017

[2] J.R.R Uijlings, K.E.A. van de Sande, T. Gevers, and A.W.M. Smeulders, "Selective Search for Object Recognition" University of Trento, Italy, University of Amsterdam, The Netherlands, Technical Report, 2012

[3] P.F. Felzenszwalb and D.P. Huttenlocher, "Efficient Graph-Based Image Segmentation", *IJCV*, 59:167-181, 2004. 1, 3, 4, 5, 7

[4] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, "Rethinking the Inception Architecture for Computer Vision".

[5] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, "Densely Connected Convolutional Networks".

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun ,"Deep Residual Learning for Image Recognition".

[7] Uri Shaham, Yutaro Yamada, Sahand Negahban, "Understanding Adversarial Training: Increasing Local Stability of Neural Nets through Robust Optimization".