Face detection with PHOG in the WIDER FACE dataset

Laura Bravo
Universidad de los Andes
Cra 1 #18a-12, Bogotá, Colombia
lm.bravo10@uniandes.edu.co

Juan Pérez
Universidad de los Andes
Cra 1 #18a-12, Bogotá, Colombia
jc.perez13@uniandes.edu.co

Abstract

Object detection is a fundamental task in Computer Vision, which consists of locating all instances of a given object in an image. The usual approach is to extract possible candidates in the image and provide them to a classifier that learned what the object is, so that it decides if the candidate is or is not an instance of such object. The task is, consequently, to define a feature space in which the object has different properties from the rest of possible objects. In the present work, we dealt with the task of face detection in a subset of the WIDER FACE dataset in which all face instances were larger than 80×80 pixels. Our pipeline consisted of Pyramidal Histogram of Oriented Gradients (PHOG) as the descriptor and a subsequent Intersection Kernel Support Vector Machine (IKSVM) as the classification model. We used the open source library VLFeat for the actual implementation of both PHOG and the SVM because of its simplicity and efficiency. Our contribution to the strategy was to take into account the various directions in which the face can be pointing, given that the "average" HOG representation will contain much noise coming from all these directions. We handle this by dividing the data points into subsets in the HOG space through kmeans with $k = 3$, and then we feed the SVM with the data belonging to the cluster that turned to be the one of faces pointing towards the camera (determined by simple visual inspection) as the positive examples. We hypothesized that this method should make it learn a more tightly bounded (restricted) model with a lower recall but probably higher accuracy. The results for the standard PHOG approach gave an AP of 4.657E-6 on the easier detections, while the AP for our strategy (clustering the training images) was 5.954E-6. We think our strategy gave better results when compared to the standard approach because there was a reduction in the false positives and an increase in the accuracy due to the higher selectivity of training images.

1. Introduction

Object detection is the process of finding instances of real-world objects in images or videos [1, 2]. It is a fundamental problem in computer vision, and it has been widely studied by research groups. Object recognition is closely related to this task, since recognition requires to search for candidates for classification in an image and then classify them. Therefore, either detection could be regarded as a problem of recognizing an object in an image with the restriction that there are only two classes: the object the system knows and everything else; or recognition could be regarded as a detection problem, where detection should be run as many times as possible classes, so that all objects in the image of every class are found.

There have been several models developed to tackle this problem, for instance: feature-based object detection, Viola-Jones object detection, *Support Vector Machine* (*SVM*) classification with histograms of oriented gradients (HOG) features, through blob analysis, gradient-based, derivative-based, and template matching approaches [1].

For classification tasks, in which objects from a given class must be differentiated from objects of all other classes, the usual approach is to define a descriptor of the image in which there is, possibly, an object and classify the image according to this. Naturally, the features that characterize the object must not only correctly describe it, but they should also tell it apart from the other classes. One of the most famous descriptors is HOG. Basically, it takes an image, divides it into cells, computes the orientation distribution in each cell and keeps counts of it with a histogram; hence, the image can be described by these histograms.

This strategy can be combined with spatial pyramids for images, so that the window of evaluation, that is, the window that is being extracted for examination, can always be the same size, while the image is the one that changes. The result of this is that the whole strategy, a multiscale HOG, is able to characterize objects that are in several scales with a descriptor that is very similar (given the constraints of rescaling the image in the pyramids). A very important advantage of this is that only one classifier needs to be trained, instead of several classifiers each responding to a given scale [3].

The multiscale HOG strategy requires a not so trivial implementation, since it includes gradient computation through filtering, binning the resulting orientations and a special form of normalization that was well studied by Dalal and Triggs [3]. Of course, extremely efficient implementations of this strategy exist, and are already included in libraries such as the open source library *VLFeat*, which implements many popular computer vision algorithms specializing in image understanding and local features extraction and matching, like HOG [4].

Among the many targets for object detection, faces appear to be of great interest, probably due to the fixation humans have with themselves, giving way to phenomena like pareidolia. Also, there is the fact that pictures containing humans account for a large part of the visual content produced everyday, so it seems reasonable to study and develop algorithms that are able to process information regarding humans, like face detection. Detection of faces in an image permits other information to be extracted, like estimation of the amount of people in the image or actual identification of them.

Face detection can be studied just like any other detection problem, just that, depending on the dataset to be studied, some difficulties can be found. For instance, skin contrast with the background, the direction in which the face points, the orientation with respect to the camera's plane, variability in scale and occlusion [5]. The approach that is taken to deal with these difficulties will generate modifications in the model, that can, for example, increase the dimension of the feature space by adding information, and/or increase the number of machine learning models that are trained so that each one of them can be specialized in handling some sort of extra difficulty.

In this work we applied multiscale HOG to the problem of face detection to a subset of the WIDER FACE dataset [5]. We used HOG's implementation from the *VLFeat* library to represent all the train images (cropped images of faces in this case), clustered them in this space to state a division between the different things it might be confusing with a face, chose the "face cluster" by visual inspection, and, finally, trained a *SVM* with this as the positive examples to learn to detect faces in this subset.

2. Materials and methods

2.1. Our multiscale HOG strategy

As previously mentioned, we use the *VLFeat* library, which does not only have implementation of the algorithm to compute HOG, but also a very efficient implementation of *SVMs* with a wide variety of kernels to handle non-linearities. The hyperparameters of this strategy that we could control in this implementation were: (i) size of the cells in which a given image (or patch of image) is divided, (ii) number of bins in the histogram, which is the number of orientations in which the information will be discretized, and (iii) the number of scales the spatial pyramid takes into account.

Given that in this strategy the sliding window is of constant size, the first two hyperparameter define the number of cells in such window and are related to the way in which the space is approximated/discretized. It accounts for the resolution in which the features are computed at a given scale, and will affect the response of the method to finer or coarser features, that is, it will define what objects are "visible", HOG-wise, at a given scale and which are not. The third parameter's effect is rather obvious. It will immediately determine which object can and cannot be found in an image, given the size of the sliding window. If the number of scales of the pyramid is low, the strategy will be restricted to only finding a somewhat small amount of faces that do not differ much in size from those of the pyramid, which could be stated as low recall; if, otherwise, the number of scales is too high, it will make the strategy look for faces at scales in which they cannot exist and, unfortunately, it will incorrectly find some; this can also be stated as increasing the recall but probably reducing the precision.

These parameters are to be optimized based on the model of face that the algorithm learns, and the dataset in which it has to search for faces. Our modifications to the traditional strategy are based on observations about the task of face detection in particular. We think that a simple visual inspection by a human can improve the overall HOG model of a face, since we humans are very precise at this task. Therefore, we propose that, after all the HOG models for faces from the training data are computed, a simple unsupervised clustering performed by *kmeans*, with $k = 3$, can divide these data into three subsets, from which we can select the one that looks more like a human face, and then feed a *SVM* with only these examples as the positive data. We believe that this strategy acts as a filter for these data by letting a human provide some insight of what a human actually does when detecting faces that are looking straight at the camera. This is also because we believe that there is some noise in the final model due to the averaging performed over not so many examples in this training data.

Naturally, this will lower the recall for faces that are not looking directly towards the camera, but, for a rather simple strategy as this one, higher results than the ones with the traditional approach were expected.

2.2. Hard negative mining

A very important algorithm that has shown to provide improvements in detection tasks is hard negative mining. This procedure arises because the *SVM* that is being trained for detection of a specific object requires not only positive examples of the class it is looking for, but also negative examples of the other class, which is anything but its class. The everything-else class is well defined, but its examples are not, and are usually just random images of the same size as those of positive examples that *do not* containing faces. The division of the higher-dimensional space that the *SVM* is trying to do, will probably be quite trivial, with positive examples being far from negative examples in many directions, but that proposes a new issue: it will be easy for the *SVM* to confuse things that are not faces but look like faces in this in this space, since it has not seen many of those and just pays a penalty in the algorithm for letting a few of those enter in the ”positive” side of the division. For instance, in this work, the *SVM* could easily mistake a face with a basketball, since they present shapes that are much alike. These examples are known as hard negatives, in consideration of the difficulty they present to be correctly classified.

One could propose a process by which a *SVM* that has been trained with random negatives, is run over images whose annotations are known, and the false positives produced by the strategy (known because the annotations define them to be false) are saved as hard negatives; this is known as hard negative mining [3, 6]. The idea is to retrain the *SVM* with the same positive examples as before, but now the negative examples will not only include the random ones but also the hard negatives. This produces a feature space with data that is much richer in information regarding the things that the model confused before; therefore, the division of this space, as performed by the *SVM*, will be much more restricted because of *tighter* boundaries, which in our terms translates to a more well-stated definition of a face.

2.3. Definition of initial negatives

Initially, in order to determine the effect of the negatives selected for the hard negative mining, two sets of negative images were extracted. The first was composed of 2000 64x64 random crops from the WIDER FACE training set and, therefore, some of them actually contained human faces, which is undesirable. The second set consisted of 2000 crops with resolutions from 100×100 to 200×200 that were taken from a selected classes of the ImageNet training dataset that did not contain nor had relationship with human faces (for instance, dog images should not contain human faces, but, as a matter of fact, they do because humans tend to pose around dogs).

2.4. Dataset

All the algorithms were developed and performed over WIDER FACE [5], which is a publicly available dataset developed by the Chinese University of Hong Kong. It has tens of thousands of images and hundredths of thousands of labeled faces in a wide variety of scenes (see Fig. 1 for examples). The dataset itself is a remarkable attempt to represent the general case of faces in realistic scenes, in which they present a high degree of variability in many of the features that a face has, giving way to a lot of issues when trying to detect them.

Figure 1. Examples of images in the *parade* scenes of WIDER FACE.

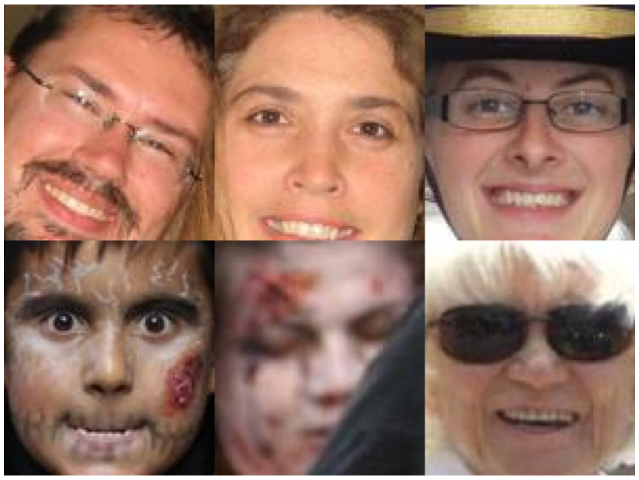
Figure 2. Examples of faces in the *parade* scenes of WIDER FACE.

So that reasonable results could be achieved with a rather simple implementation of this strategy, we used a subset of WIDER FACE, in which all face instances are larger than 80×80 pixels. Even in this restricted subset, many of the issues previously mentioned remain.

Annotations for training are not the same as those for other computer vision tasks. The goal here is that the model gathers experience about what a face looks like, hence, annotations are just faces, and a general model of a face can be learned by computing HOG and providing these features to a *SVM*. Examples of these cropped images of faces can be seen in Fig. 2. Evaluation annotations are different too. Since the objective is to detect (possibly) various instances of faces in an image, annotations must be a bounding boxes, each of a single face.

The evaluation metric is based on that of the PASCAL VOC Challenge. It is performed based on a bounding box provided by the algorithm together with a confidence level for it. The confidence level allows results to be ranked such that the trade-off between false positives and false negatives can be evaluated, without defining arbitrary costs on each type of classification error. Detections are assigned to ground truth objects and judged to be true or false positives by measuring bounding box overlap. To be considered a correct detection, the Jaccard Index must exceed 50%. Detections output by a method are assigned to ground truth object annotations satisfying the overlap criterion in order ranked by the (decreasing) confidence output. Ground truth objects with no matching detection are false negatives, while multiple detections of the same object in an image are considered false detections [7].

2.5. Methods that were evaluated

In this work we did not only evaluate our strategy, but also the baseline strategy, which is the one that has only one *SVM* instead of three, so that we could tell if our modifications provided an enhancement to the overall strategy. Needless to say, these two strategies were evaluated in the same way for a fair comparison. Additionally, the functions included in the *VLFeat* library, by default, compare the method the user applies with other well known methods, so we also present results comparing our strategy to these.

3. Results

The traditional multiscale HOG method, with scales from **0.5 to 8** with 8 orientations and a window size of **size** was evaluated on the validation set of the WIDER FACE dataset. The results were grouped by the difficulty of the faces to be detected into easy, medium and hard. The Figures 3 4 5 contain the PR curves with our method using negatives extracted from the WIDER FACE dataset (OurMethod1) and negatives from a subset of the ImageNet dataset (OurMethod2). As well as a clustering approximation to reduce the multiple orientations that a face can be in and that can ultimately add noise to the model (OurMethod3). As we expected, the best results were obtained for all the methods with the easy to detect images and the worst with the hard faces.

Nonetheless, after comparing the results obtained with both methods 1 and 2, the AP improved when extracting the negative instances directly from the WIDER FACE dataset. Despite the fact that these images contained complete and incomplete faces, that is, they were not negative examples. This can be explained because even though there was noise in the negatives, by being extracted directly from the same dataset that the evaluation images, they were much more useful in determining irrelevant information, than a images from a completely different dataset. Moreover, the size of the negative images also contributed to the effectiveness of the native negatives. The images were small, and thus the patches extracted from them in the hard negative mining phase were even smaller reducing the possibility of the

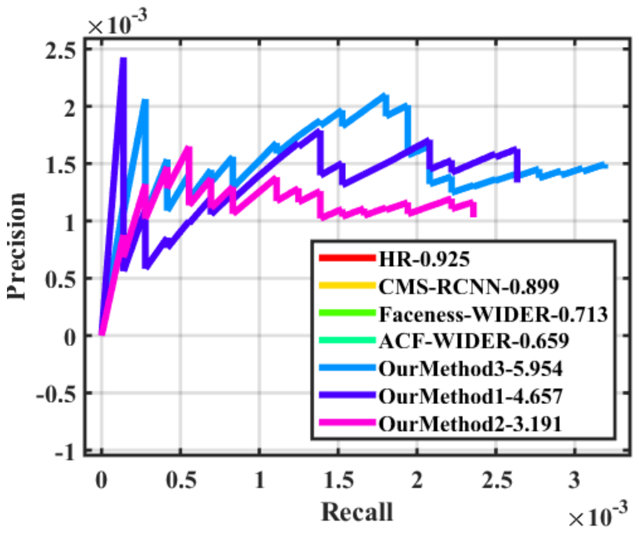

Figure 3. Result of detection on easy subset of WIDER FACE using negatives from the same dataset (OurMethod1) with an AP of 4.657E-6 and with the negatives from ImageNet (OurMethod2) the AP was 3.191E-6. The result of clusterizing the training images (OurMethod3) was 5.954E-6.

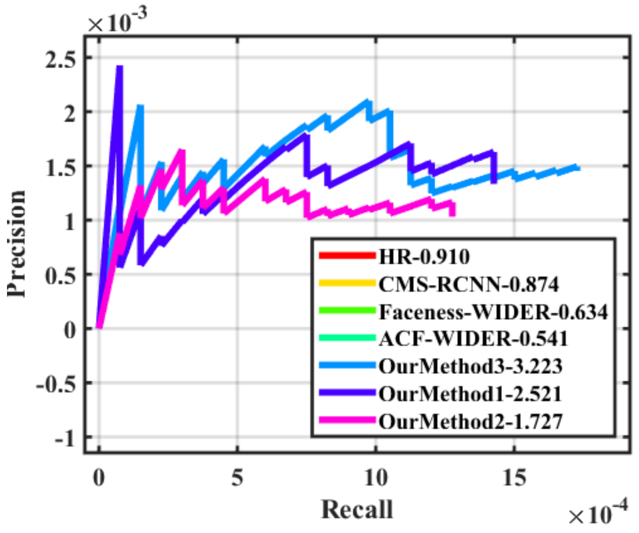

Figure 4. Result of detection on medium difficulty subset of WIDER FACE using negatives from the same dataset (OurMethod1) with an AP of 2.521E-6 and with the negatives from ImageNet (OurMethod2) the AP was 1.727E-6.The result of clusterizing the training images (OurMethod3) was 3.223E-6.

model learning a complete face as a negative example.

The third method (OurMethod3) was overall the best one with an AP of 5.954E-6 on the easy detections, 3.223E-6 in the medium difficulty ones and 1.343E-6 in the hard ones. The AP improved because of the increase in both precision and recall. This is could be due to the decrease in false positives given by creating a HOG template model that considered only a specific cluster of training images, that we inter-

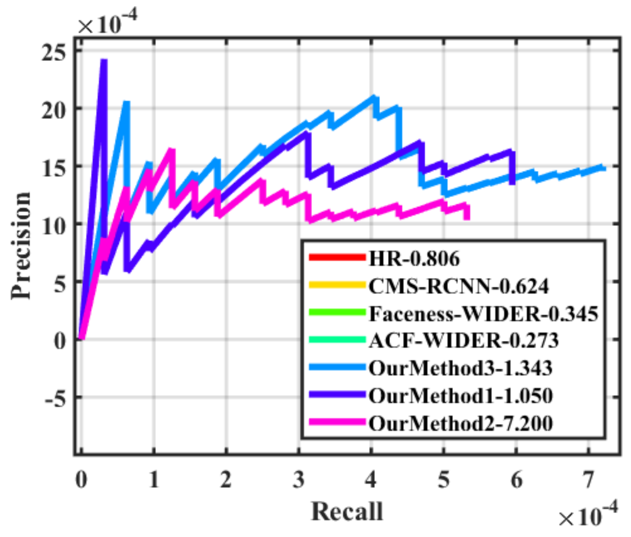

Figure 5. Result of detection on hard subset of WIDER FACE using negatives from the same dataset (OurMethod1) with an AP of 1.050E-6 and with the negatives from ImageNet (OurMethod2) the AP was 7.200E-7.The result of clustering the training images (OurMethod3) was 1.343E-6.

pret as a specific orientation. Figure 6 displays the average image of the training instances from the selected cluster, the result is an image with similar characteristics to a face an oval shape, dark spaces at eye level, a mouth and a long structure at the center of the image akin to a nose. Consequently, it is understandable that by selecting this cluster, an appropriate model was created based on the subset of training images. Nevertheless, in general none of the methods has a sufficient recall nor precision, but this was improved by implementing OurMethod3. This suggests that it is more convenient to attempt to separate the training images when creating the model, so as to decrease the false positives (less noise) and also creating a model with a higher selectivity (better recall).

4. Conclusions

In general, the limitations come from the fact that HOG describes shape but does not consider color and/or texture characteristics. Hence, the *SVMs*, even with hard negative mining, will still tend to confuse things that have a face-like shape. Perhaps, a basketball may be understood as a face because its representation by HOG is pretty much the same as that for a face.

For our strategy in particular, we realize that we are only taking into account the faces that are directly pointing towards the camera and, therefore, leaving out the ones that are not and. These faces do not share many HOG features between them, so the selection we make will produce a more selective *SVM* that will probably despise faces pointing to other directions.

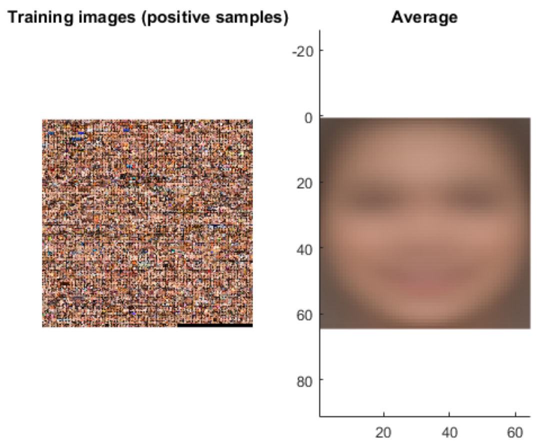

Figure 6. Mosaic of training images (left panel) and average response of positive images (right panel).

The HOG strategy itself was defined to be invariant to color but, for this particular case, color could provide very important information. Human skin color is rather restricted to a very narrow color spectra, so even if HOG itself does not take color into account, two things could be done: (i) concatenate some information color to the descriptor so that the resulting feature space is responsive to this, or (ii) do not feed every sliding window to the strategy, HOG and *SVM*, but filter them first according to their color content. The color descriptor could be, for instance, some color histogram.

The problem with this is that, although not common, it happens that people paint their face with some unnatural color, so this extra information could be misguiding. This specific issue occurs in WIDER FACE, so, if this is implemented with the algorithm, it could provide a reduction in its recall (but an increase in precision, too). We consider that this should not actually be the official task for simple algorithms, in view of the fact that this is challenging even for humans. Consider face camouflage, for instance, which has the precise objective of hindering detection, and actually accomplishes it in war scenarios.

Additionally, regarding the correct handling of the fact that faces can point to directions other than just towards the camera, we think that adding other *SVMs* as a way to model the various directions in which a face can be pointing (same as the number of clusters for *kmeans*), could readily improve the method's performance. Nevertheless, that would introduce the problem of combining the response of all these *SVMs* that actually point towards the same class (face). This is difficult because it is a problem similar to the that of multiclass *SVMs*, where the way to interpret the whole system's output is not uniquely defined, requiring the programmer to make a decision and, as a result, giving way to some error. Nevertheless, experiments with this approach should be made, so as to define the method for combining these responses that provides the best performance for this task.

References

[1] ”Object Detection - MATLAB & Simulink”, Mathworks.com, 2017. [Online]. Available: `https://www.mathworks.com/discovery/object-detection.html`. [Accessed: 25- Apr- 2017].

[2] ”UC Berkeley Computer Vision Group - Recognition”, `Www2.eecs.berkeley.edu`, 2017. [Online]. Available: `https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/shape/`. [Accessed: 25- Apr- 2017].

[3] N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05).

[4] A. Vedaldi and B. Fulkerson, ”VLFeat - Home”, Vlfeat.org, 2017. [Online]. Available: `http://www.vlfeat.org`. [Accessed: 26- Apr- 2017].

[5] S. Yang, P. Luo, C. C. Loy, and X. Tang, “WIDER FACE: A Face Detection Benchmark,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[6] A. Shrivastava, A. Gupta, and R. Girshick, “Training Region-Based Object Detectors with Online Hard Example Mining,” 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[7] M. Everingham, S. Eslami, L. Van Gool, C. Williams, J. Winn and A. Zisserman, ”The Pascal Visual Object Classes Challenge: A Retrospective”, International Journal of Computer Vision, vol. 111, no. 1, pp. 98-136, 2014.

[8] ”Does HOG capture color?”, Web.mit.edu, 2017. [Online]. Available: `http://web.mit.edu/vondrick/ihog/color/`. [Accessed: 27- Apr- 2017].

[9] ”Object Detection in a Cluttered Scene Using Point Feature Matching - MATLAB & Simulink Example - MathWorks United Kingdom”, Mathworks.com, 2017. [Online]. Available: `https://www.mathworks.com/help/vision/examples/object-detection-in-a-cluttered-scene-using-point-feature-matching.html#btt5qyu`. [Accessed: 28- Apr- 2017].

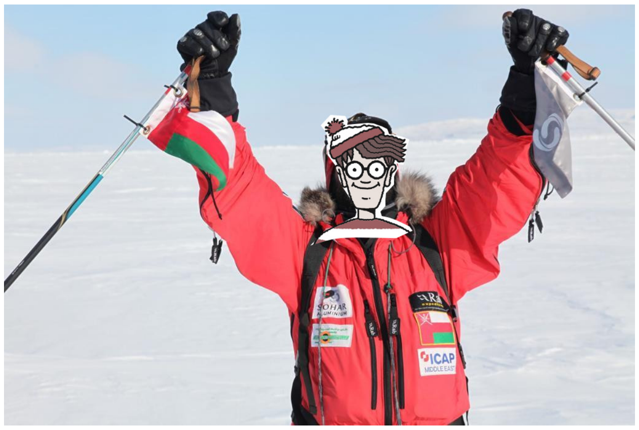
Figure 7. Image in which Waldo was found!

Extra credits

Finding Waldo

With the help of one of MATLAB’s tutorial [9] we implemented an object detection algorithm based on matches of feature keypoints in the original Waldo image, and all the images in the Test set. Waldo was found to be in the image called `13_Interview_Interview_On_Location_13_559`, which can be seen in Fig. 7.