Object Localization and Recognition

Bilkent University Image Analysis (CS 484) course
Term Project Report

Group members

Fuad Aghazada
Email: fuad.aghazada98@gmail.com
ID: 21503691

Eray Şahin
Email: sahin.eray@outlook.com.tr
ID: 21502758

Muhammed Askari Iqbal
Email: askari.iqbal@ug.bilkent.edu.tr
ID: 21504228

Enes Varol
Email: enesvarol77@gmail.com
ID: 21604086

28.05.2019

Abstract

The goal of this project is implementing methods for object localization and recognition for images. The system worked on a subset of ImageNet dataset. Subset contains 400 train and 100 test images with 10 categories.

I. INTRODUCTION

We developed a program that can find animals on images. For this project, we trained on 10 different animals: eagle, dog, cat, tiger, starfish, zebra, bison, antelope, chimpanzee and elephant. We passed 400 images to the system as training images and 100 images as testing images. Details of the approach, implementation steps, and results/discussions will be provided in the rest of the paper.

II. PRE-PROCESSING

A. Data Normalization

Each picture is pre-processed beforehand so that all data inputs are uniform. This uniformity of a picture is achieved with regards to its size and color. Firstly, a tight bound is found in a picture from object boundaries in order to remove background as much as possible - the background does not give us any useful information. Afterward, the image is scaled into a 224x224 pixel image with its RGB channels. This is to satisfy a condition by ResNet that is implemented by PyTorch [1]. It is done by padding the image by adding the least number of zeros in order for the image to become a square - It is then scaled to 224 pixels.

The second step in this process is to normalize the intensities of each pixel by dividing it by 255 so that each pixel value is between [0, 1]. Afterward, each pixels RGB channels are subtracted with the mean values of 0.485, 0.456, 0.406 for red, green and blue respectfully. In the end, each channel is divided with the standard deviation of the channels which correspond to 0.229, 0.224 and 0.225 for red, green and blue channels respectfully.

B. Feature Extraction

Features are extracted by using the PyTorch implementation of the ResNet. By this step, we obtain vectorized data for images. The obtained vector is called the feature vector of the image.

III. TRAINING THE SYSTEM

After extracting the feature vectors for each image, a classifier model should be trained in order to classify the given images. In this case, due to the multiclass dataset, 10 (the number of classes in the dataset) classifiers - Support Vector Machines (SVM) models need to be trained via “one vs all” technique: in this technique, for a given class all samples within this class are labeled as positive whereas the other samples are labeled as negative. By this technique, we are supposed to obtain information for each unique class to determine the performance of that class.

Scikit-learn [2] module of Python has been used to train the classifiers. Due to the possibility of having a linearly inseparable dataset, we used the Radial Basis Function (RBF) kernel for transforming the features to higher dimensions to have linearity in the dataset.

Two hyper-parameters have been used for the SVM models:

- C-regularization factor (0.1) - for regularizing the number of misclassifications (higher the C value, narrower the margin - no much space for misclassification)
- Gamma value (1) - for determining the variance and the bias of the

model (larger the gamma, the higher the bias and lower the variance)

IV. TESTING THE SYSTEM

A. Classification Accuracy

For the classification part of the project, firstly, it was necessary to extract some candidate windows for the image so that we can extract test feature vectors for that image by collecting the feature vectors of the pre-processed and cropped image. These features are necessary for having the predictions from the classifiers. The candidate windows are extracted using a model configuration file via the OpenCV module of Python [3] and limited to 50 windows.

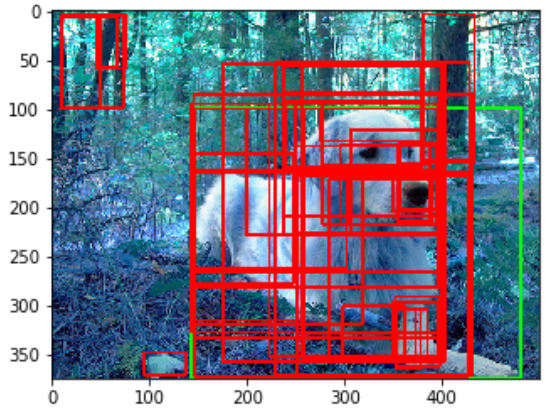

Figure 1: Candidate windows for an image

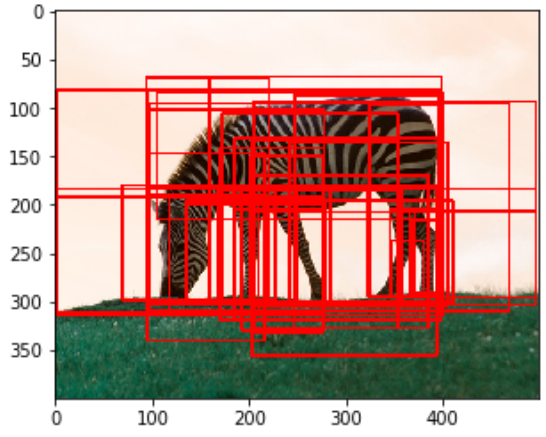

Figure 2: Candidate windows for another image

After vectorization of the test images, we pass the test feature vectors to each SVM classifier that has been trained in the previous steps. The predictions of each classifier are collected for each image. Afterward, the best prediction has been taken from all the predictions. A prediction contains information of the predicted label and the index of the best-suited candidate window for the image.

B. Localization Accuracy

Having extracted the candidate windows, we obtain possible information about the localization of the animals in the image. In addition, after the predictions of classifiers, we can also obtain the best-suited edge box for an image. This box that is calculated is compared with one that is given for the test images - by dividing the intersection of the ground truth box and the predicted box to their union. It helps us to observe how well does the system function to accurately identify an object. If the resulting box does not coincide with the given box for less than 50 % localization accuracy, the result is dropped and is considered a misclassification.

V. RESULTS

The results are displayed in two forms. First in the form of a table that shows the classification accuracy of our model for each of the label. It gives a good estimate of how the test pictures were classified from the 400 pictures it trained from.

TABLE 1. Confusion Matrices

Object	Confusion Matrix
Eagle	TP: 10, FP: 1, TN: 89, FN: 0
Dog	TP: 9, FP: 0, TN: 90, FN: 1
Cat	TP: 9, FP: 0, TN: 90, FN:: 1
Tiger	TP: 10, FP: 0, TN: 90, FN: 0

Star	TP: 10, FP: 1, TN: 89, FN: 0
Zebra	TP: 10, FP: 0, TN: 90, FN: 0
Bison	TP: 9, FP: 1, TN: 89, FN: 1
Antelope	TP: 10, FP: 0, TN: 90, FN: 0
Chimpanzee	TP: 10, FP: 3, TN: 87, FN: 0
Elephant	TP: 7, FP: 0, TN: 90, FN: 3

TABLE 2. Classification Statistics

Object	Accuracy	Precision	Recall	F-Score
Starfish	0.99	0.909	0.999	0.952
Elephant	0.97	0.999	0.699	0.824
Zebra	1.0	0.999	0.899	0.999
Dog	0.99	0.999	0.899	0.947
Bison	0.98	0.899	0.899	0.899
Chimpanzee	0.97	0.999	0.999	0.999
Antilope	1.0	0.999	0.999	0.999
Cat	0.99	0.999	0.899	0.947
Eagle	0.99	0.909	0.999	0.952
Tiger	1.00	0.999	0.999	0.999

We achieved a 94% overall classification accuracy. As stated in the assignment, this metric considers only the percentage of correctly classified images. In other words, it does *not* consider how localization was performed while classifying those images (unlike localization accuracy).

The second is the results of the localization accuracy and how the bordering box was obtained for each picture versus the bordering box that was expected from the pictures. These give the most real-life value of our system as it shows how accurately our model can predict the position of an object in an image.

Table 3 (next) shows the overall statistics for all test data instances with their localization accuracies. In the table, entries marked by yellow denote the images for which localization accuracy was below the 50% threshold. On the other hand, the instances colored by red are the misclassified test instances. Overall (averaged) localization accuracy we achieved was 63% (with the threshold of 50 % localization accuracy).

TABLE 3. Overall and Localization Statistics

Test data	Actual label	Predicted label	Localization accuracy (%)
1	star	star	76.61
2	star	star	44.68
3	star	star	63.12
4	star	star	97.28
5	star	star	85.52
6	star	star	75.41
7	star	star	68.53
8	star	star	51.66
9	star	star	54.28
10	star	star	71.85
11	elephant	elephant	75.3
12	elephant	elephant	88.13
13	elephant	elephant	70.56
14	elephant	elephant	78.67
15	elephant	chimpanzee	21.92
16	elephant	elephant	98.17
17	elephant	star	0
18	elephant	elephant	35.57
19	elephant	bison	81
20	elephant	elephant	64.78
21	zebra	zebra	52.4
22	zebra	zebra	45.41

23	zebra	zebra	56.84
24	zebra	zebra	88.47
25	zebra	zebra	77.25
26	zebra	zebra	76.56
27	zebra	zebra	69.68
28	zebra	zebra	63.85
29	zebra	zebra	76.76
30	zebra	zebra	50.28
31	dog	dog	76.06
32	dog	dog	44.83
33	dog	chimpanzee	17.03
34	dog	dog	43.5
35	dog	dog	70
36	dog	dog	79.82
37	dog	dog	43.28
38	dog	dog	79.4
39	dog	dog	71.02
40	dog	dog	73.22
41	bison	bison	38.62
42	bison	bison	84.68
43	bison	bison	78.3
44	bison	chimpanzee	13.13
45	bison	bison	77.39
46	bison	bison	45.74
47	bison	bison	77.58
48	bison	bison	51.48
49	bison	bison	32.34
50	bison	bison	58.15
51	chimpanzee	chimpanzee	96.37
52	chimpanzee	chimpanzee	69.27
53	chimpanzee	chimpanzee	35.28
54	chimpanzee	chimpanzee	34.41
55	chimpanzee	chimpanzee	54.48

56	chimpanzee	chimpanzee	45.64
57	chimpanzee	chimpanzee	38.04
58	chimpanzee	chimpanzee	96.46
59	chimpanzee	chimpanzee	32.58
60	chimpanzee	chimpanzee	65.62
61	antelope	antelope	96.95
62	antelope	antelope	37.19
63	antelope	antelope	64.96
64	antelope	antelope	92.18
65	antelope	antelope	57.09
66	antelope	antelope	32.09
67	antelope	antelope	89.12
68	antelope	antelope	49.6
69	antelope	antelope	93.84
70	antelope	antelope	38.34
71	cat	cat	56.28
72	cat	eagle	9.15
73	cat	cat	43.17
74	cat	cat	47.01
75	cat	cat	33.97
76	cat	cat	59.51
77	cat	cat	18.97
78	cat	cat	31.13
79	cat	cat	96.52
80	cat	cat	42.93
81	eagle	eagle	97.5
82	eagle	eagle	78.88
83	eagle	eagle	36.44
84	eagle	eagle	26.65
85	eagle	eagle	67.6
86	eagle	eagle	48.41
87	eagle	eagle	66.59
88	eagle	eagle	44.86

89	eagle	eagle	96.85
90	eagle	eagle	92.54
91	tiger	tiger	31.95
92	tiger	tiger	17.28
93	tiger	tiger	57.83
94	tiger	tiger	18.51
95	tiger	tiger	77.14
96	tiger	tiger	84
97	tiger	tiger	58.41
98	tiger	tiger	73.54
99	tiger	tiger	67.44
100	tiger	tiger	30.24

You can find the average localization rates (per class) in the following table. After analyzing the actual and predicted bounding boxes, we have concluded that localization accuracy depends on object proposals. In some cases, object proposals focus mostly on the head of the animal or tend to be concentrated around nearby objects. In those cases, the two regions (the one we predict and the ground truth box) tend not to overlap much. This might be the part of the problem with low localization accuracies. Additionally, since we feed each box into our models, the choice of hyperparameters also affect the localization accuracy (by choosing the most suitable box).

TABLE 4. Average Localization Accuracy for Each Object Type

Object type	Avg. Localization Accuracy (%)
star	68.894
elephant	61.41
zebra	65.75
dog	59.816
bison	55.741
chimpanzee	56.815
antelope	65.136
cat	43.864
eagle	65.632
tiger	51.634

In the following figures, green boxes represent ground truth boxes, and red boxes represent the best-predicted object proposal.

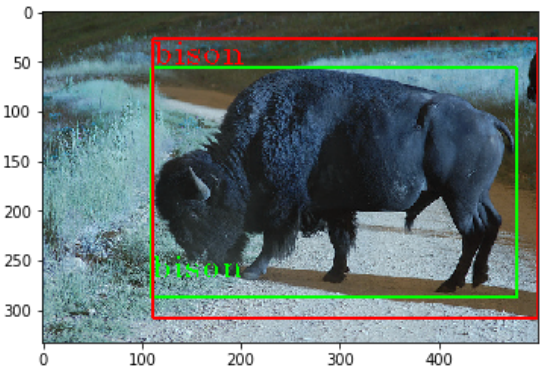

Figure 3: Predicted box vs ground truth box (localization accuracy = 77%)

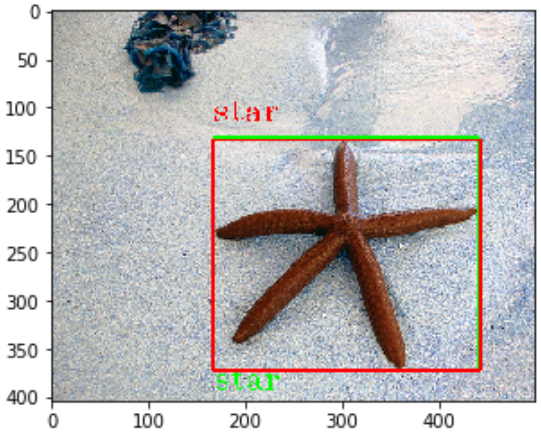

Figure 4: Predicted box vs ground truth box (localization accuracy = 97%)

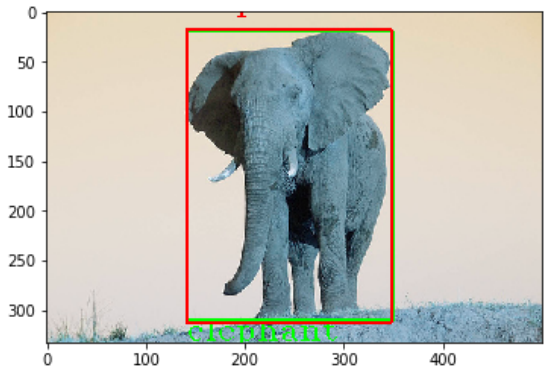

Figure 5: Predicted box vs ground truth box (localization accuracy = 98%)

Sometimes the system can misclassify and poorly localize some images:

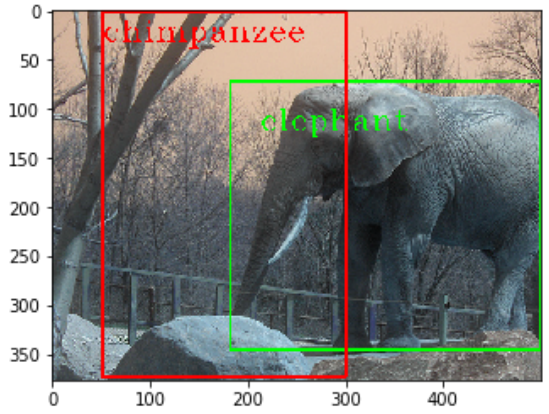

Figure 6: A misclassified image (predicted as a chimpanzee) with localization accuracy 22%

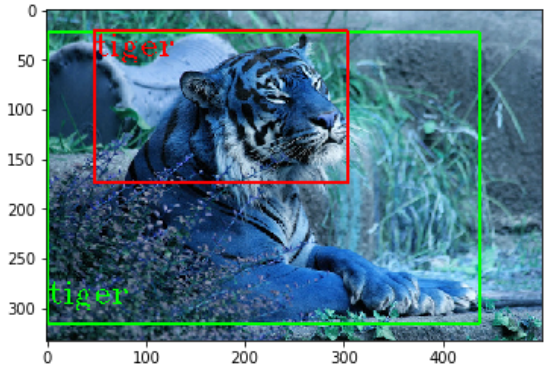

Figure 7: A correctly class classified image with localization accuracy 30%

Below, examples for candidate windows (object proposals) from each class are given.

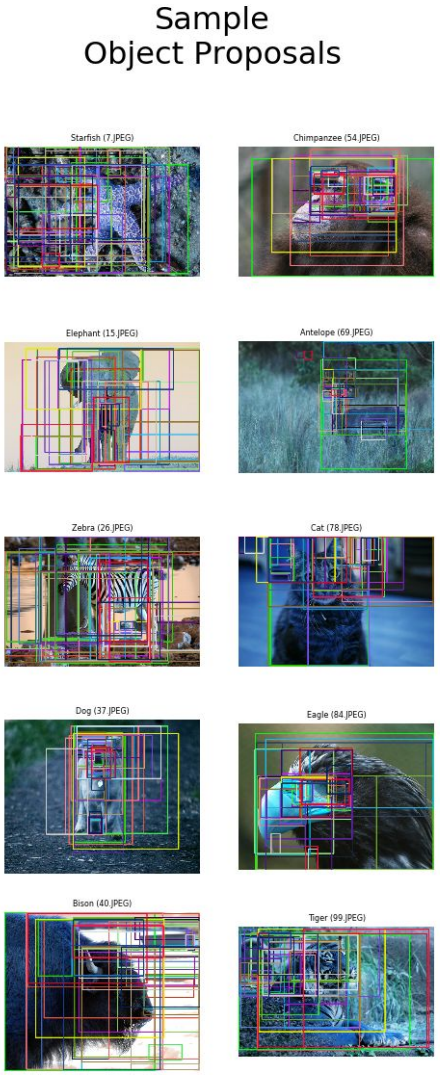

Figure 8: Object proposals illustrated on samples from each class

On the next page, localization results for two test samples (from each object type) are given. The red window and red label denote the predicted object (while the green ones illustrate the ground truth label and window).

Figure 9: Localization results (colored with red) and ground truth locations of objects (shown in green)

VI. SUGGESTIONS FOR IMPROVEMENTS

As seen by the results, our system was trained well to identify and label an image correctly. However, identifying a localized area where the object would lie inside of the system is not done properly. There are a few suggestions for improving this problem.

The Edge box method that we use is a very computationally heavy method and uses a lot of computational power that our laptops did not have to run smoothly. The method uses a hyperparameter for the number of rectangles created for an image. To be able to run the program on our computers, we kept the hyperparameter of how many rectangles can be drawn as 50 but this is not always enough to accurately recognize the localized boundary. To counter this problem, we can use cloud computing with Graphic cards that will allow us to be able to increase this hyperparameter while being able to run the project.

Another improvement that we could make was to use a wider range of SVM parameters to train our data. We used a few different values of the SVM parameters but then settled on the one that gave us a good classification accuracy. By testing with these parameters more we might have found an even better result.

Lastly, the feature vector that we had obtained from the ResNet was used directly into learning. The feature vector was not normalized properly and that could have resulted in problems with learning as well. A good approach would have been to gather all the feature vectors and made a big corpus like a bag of words. This corpus could then be used to make a new feature vector for each image that would be encoded one hot for each feature that it corresponded to in the new feature vector. Naive Bayes could then be used to have dimensionality reduction on this new feature vector to only focus on features that are useful to all the training data. With this new feature vector, the trained model would be less biased

towards images that have more of those features that occur less in the rest of the images and thus increase accuracy.

VII. HOW TO RUN THE SYSTEM

The program has been coded in Python3 programming language with some necessary external library/modules, so Python should be installed on the machine. The project folder contains a text file with the name of 'requirements.txt' which contains all the used Python3 modules with the specified version. To install these modules, the *pip* package manager (should be installed on the machine too) can be used with the following command: `pip install -r requirements.txt`

After having all the necessary modules installed, the program can be run by executing the following command: `python3 main.py`

Alternatively, the project folder also contains a Python Notebook file using which you can view the results of the needed parts of the project by executing the specific cells.

The source code and models for the project can be found on our GitHub repository: https://github.com/fuadaghazada/Object-Localization-and-Recognition/tree/master/src

VIII. REFERENCES

[1] Pytorch, "pytorch/vision," *GitHub*. [Online]. Available: https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py. [Accessed: 28-May- 2019].

[2] scikit-learn, "RBF SVM parameters - scikit-learn 0.21.2 documentation," *scikit-learn*. [Online]. Available: https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html. [Accessed: 28-May- 2019].

[3] opencv-contrib, "edge detection and edgeboxes".[Online].Available: https://github.com/opencv/opencv_contrib/blob/96ea9a0d8a2dee4ec97ebdec8f79f3c8a24de3b0/modules/ximgproc/samples/edgeboxes_demo.py [Accessed: 29-May- 2019]

APPENDIX

Contribution of Group Members

The project workload divided into group members uniformly and all members completed their responsibilities.