Term Project: Object Localization and Recognition

Deniz Şen
Computer Science
Bilkent University

Şehmuz Ali Subay
Mechanical Engineering
Bilkent University

Arkın Yılmaz
Computer Science
Bilkent University

***Abstract*—Image classification and object recognition is a big matter in computer vision and finding a suitable model for this purpose is a rather expensive process. This project aims to define the usability and reliability of an image analysis and machine learning model, particularly a model which uses a region-based convolutional neural network(RCNN) for feature extraction and multiple binary support vector machine(SVM) for classification. At the end of the project, it can be seen that this model is in fact surely reliable, on the other hand time-consuming.**

Keywords—deep neural network, support vector machine, image classification, object localization

I. Preparation of the environment

The program is separated into 4 subprograms where 2 of them are Python3 and 2 are MATLAB scripts.

A. Python Environment

The Python scripts require many libraries to work, thus their dependencies as well. Therefore it is strongly recommended that during the preparation of the environment, Anaconda Navigator or Anaconda Prompt is used to install and get the libraries ready to run [1]. On the Anaconda Prompt, an environment should be created using the command "`conda create --name image`" where "image" is the name of the environment. Then the environment should be activated using "`activate image`" command. Then the commands "`conda install -c soumith torchvision`", "`conda install pillow`", "`conda install scipy`" and "`conda install tqdm`" should be run in order to install torchvision, pillow, scipy and tqdm along with their dependent libraries [2, 3, 4, 5]. After installing the libraries, the content of the ZIP file of the project should be extracted to a directory.

B. MATLAB Environment

The MATLAB scripts of the program require MATLAB Image Processing Toolbox and MATLAB Statistics and Machine Learning Toolbox before any other inclusion of libraries [6, 7]. Further, it is required that Piotr Dollar's Structured Edge Detection Toolbox is gathered from GitHub [8]. After downloading the repository of this toolbox, unzip the content of repository inside a folder called "edges-master" inside the directory in which the content of he project ZIP file were put. Please note that this toolbox also requires Piotr's MATLAB Toolbox which should also be taken from GitHub [9]. This repository should be unzipped inside a folder called "toolbox-master" in the same directory as the project files. Then, on the MATLAB console the commands "`addpath(genpath('edges-master'));`" and "`addpath(genpath('toolbox-master'));`" should be run. Lastly, the training images should be put inside a folder called "train" where each image belonging to a class is put inside its own folder, just as we were given. Similarly, the test images should be put inside a folder called "test/images". Inside the "test" directory, there should be a text file called "bounding_box.txt" The project should now be ready to run.

II. Running the project

The project is constituted by 4 scripts that will be explained later. The first script to be run is called "trainFeature.py" which is a Python script. Running this program takes 10-15 minutes. When the program finishes the execution, the second script which is called "trainingWindow.m" which is a MATLAB script. Execution of this program takes 1-2 minutes. Please note that the data pipeline between different programs is done by saving the data structures as ".mat" files inside the directory, therefore deletion of any of these files will cause the program to not work. When the second program is finished, the third program "windowRead.py" should be run and this is a Python script. This program takes around 100-120 minutes as it processes many images. Finally, when the third program finishes the execution, the MATLAB program "testProcess.m" whose execution takes 30-60 seconds, should be run. The accuracy of classification and localization can be seen as the contents of variables "classificationAccuracy" and "localizationAccuracy" respectively. Note that neither of the scripts cleans the workspace or saved matrices from the project directory.

III. IMPLEMENTATION DETAILS

As described earlier, the project is constituted by 4 different scripts which pipeline data between each other and process them by themselves. These scripts have multiple kinds of operations. The program has the following main operations to be done:

- Preprocessing of the train images
- Model training
- Preprocessing of the test images
- Model testing

A. *Preprocessing of the Train Images*

Every image in the dataset has different dimensions which makes the feature extraction not possible with a fixed sized filters. Therefore we need to process each training image such that every image will have the same dimension. In this case, the images are processed to be 224x224. For that, an empty 224x224 image is created in every iteration of images. Then, the original image is resized to have one of its dimensions as 224. So the resized image can be fitted in the middle of the empty image perfectly with some black parts remained, depending on the balance of the ratio. Then, since the program uses PyTorch, the content of the resized image is normalized by first dividing each color channel with 255. Furthermore, from red, green and blue channels 0.485, 0.456 and 0,406 are subtracted and divided by 0.229, 0.224 and 0.225 respectively. Figure 1 shows a processed image.

The resulting image is converted from Numpy array to PyTorch array which allows it to be processed by ResNet50 which is a pretrained deep neural network. It takes an input 224x224 torch array gives a feature vector of 2048 dimensions. These vectors will be the representations of the images in the training steps. The obtained feature vector for each image is recorded and saved into a MATLAB array using scipy.io.

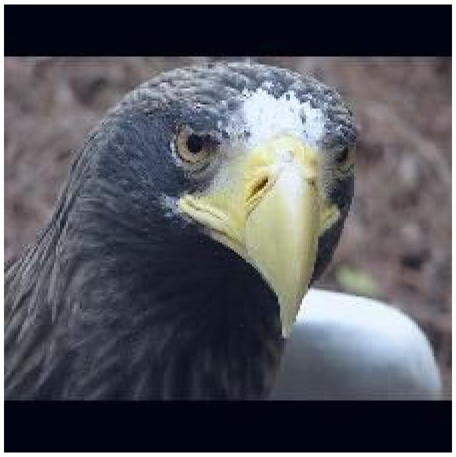


Figure 1

B. *Model Training*

The training is done in the MATLAB side since the classification model is be a SVM implementation and the program uses MATLAB's built-in SVM computation function called "fitcsvm". This function takes 2 main parameters along with some settings. The first parameter is the feature array and the second one is the label array of these features. According to these inputs, the function makes a binary SVM and returns the model. In this implementation, since there are 10 classes, one binary SVM would not be enough for our classification, instead, the whole model has 10 SVM models. For instance, for an image of monkey, the model will ask the question of whether this image is a tiger or not. If it is not, it will be labeled as "not tiger" or "tiger". At the end of this process, 10 SVM models are obtained and saved into disk for future trials.

C. *Preprocessing of the Test Images*

After computing SVM classifiers, the process starts the testing step. classification of images are done by looking at candidate bounding boxes that are found inside the test images. Because we are trying to classify objects inside particular images, we have to look for the bounding boxes of these objects. Note that the train images are set to contain the animals almost perfectly. To compute the bounding boxes of the objects, te program uses "edgeboxes" function of Piotr Dollar's Structured Edge Detection Toolbox. This function takes an RGB image as input and returns every candidate bounding box with their upper left corner coordinates and their widths and heights. ALso it gives a certainty score to each bounding box. Since the function returns around 2000 candidate bounding boxes, the program only takes the most certain 50 boxes into account. These boxes are then sent to Python to be preprocessed and to be ready to extract ResNet50 features from. The same preprocessing steps are done on these candidate windows. In the test data, there are 100 images and each image gives 50 bounding boxes which makes 5000 windows to be processed.

D. Model Testing

While processing the windows, the resulting new image is changed into a torch array as well and given input to ResNet50 deep neural network. Again, the network returns the feature vector representation of the image. Then, the program continues in MATLAB and uses the SVM classifiers to classify each feature vector as either it is an animal or not. For the prediction, it uses "predict" function which takes an SVM classifier and an array of features. It labels each feature, hence candidate window and gives a classification score. Here, the program takes the highest scored window of each test image and predict the image's class as such. Figures 2-21 show the classifications of the images, along with their highest scored bounding box.

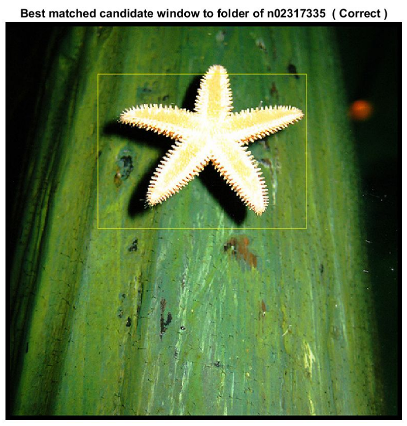

Best matched candidate window to folder of n02317335 (Correct)

Figure 2

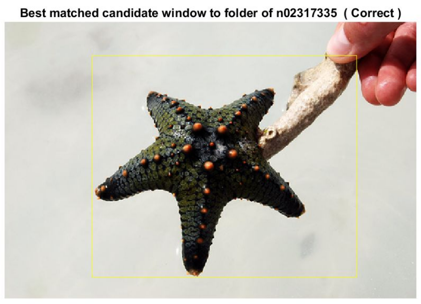

Best matched candidate window to folder of n02317335 (Correct)

Figure 3

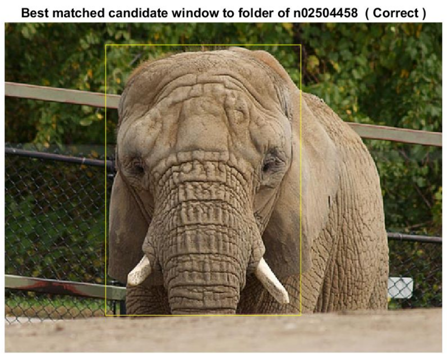

Best matched candidate window to folder of n02504458 (Correct)

Figure 4

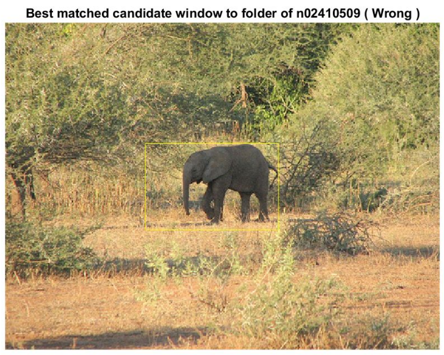

Best matched candidate window to folder of n02410509 (Wrong)

Figure 5

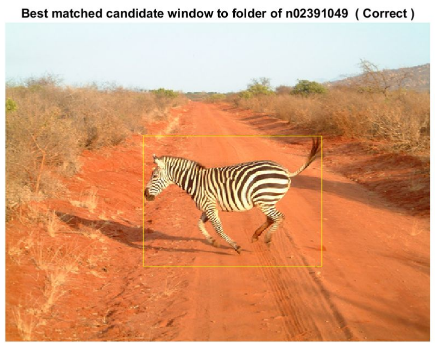

Best matched candidate window to folder of n02391049 (Correct)

Figure 6

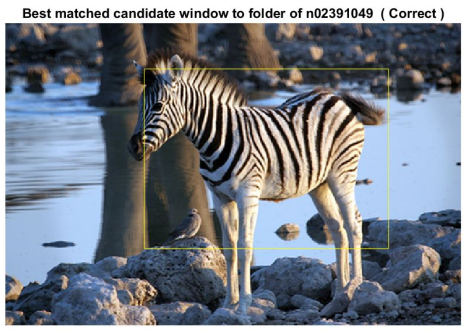

Best matched candidate window to folder of n02391049 (Correct)

Figure 7

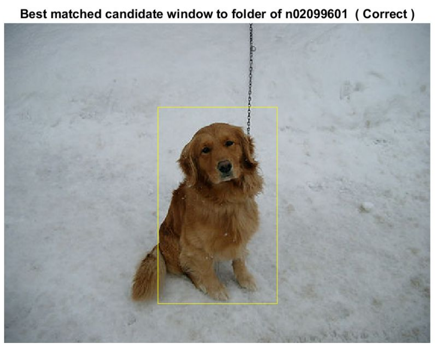

Best matched candidate window to folder of n02099601 (Correct)

Figure 8

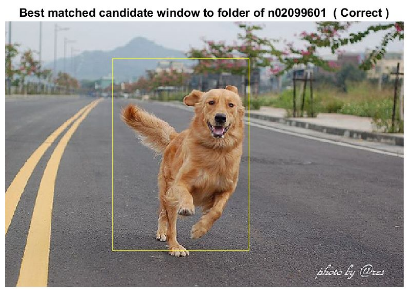

Best matched candidate window to folder of n02099601 (Correct)
photo by @zes

Figure 9

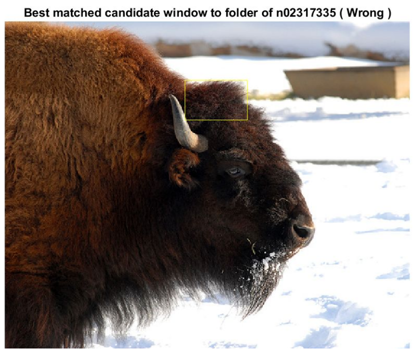

Best matched candidate window to folder of n02317335 (Wrong)

Figure 10

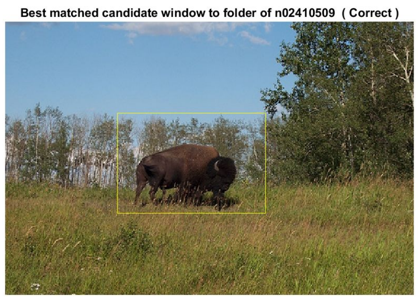

Best matched candidate window to folder of n02410509 (Correct)

Figure 11

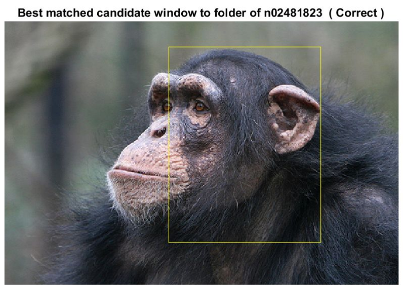

Best matched candidate window to folder of n02481823 (Correct)

Figure 12

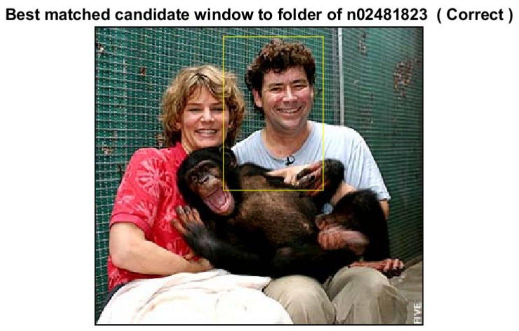

Best matched candidate window to folder of n02481823 (Correct)
FIVE

Figure 13

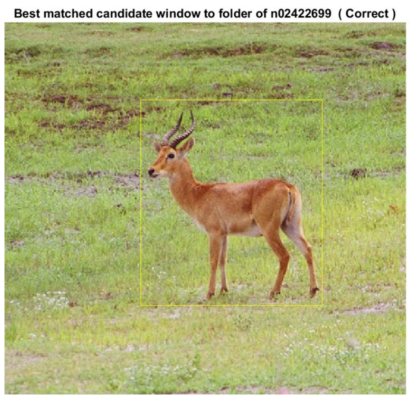

Best matched candidate window to folder of n02422699 (Correct)

Figure 14

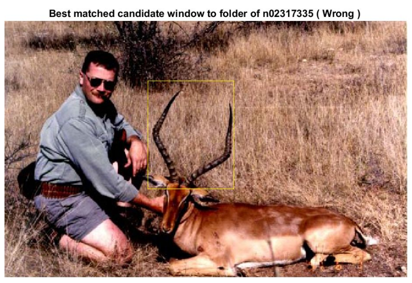

Best matched candidate window to folder of n02317335 (Wrong)

Figure 15

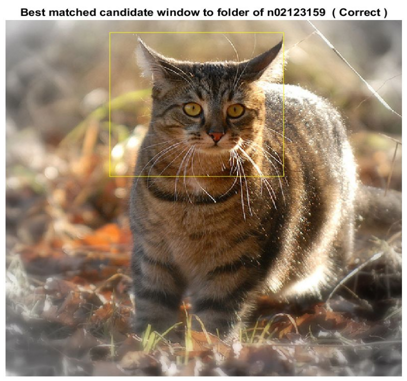

Best matched candidate window to folder of n02123159 (Correct)

Figure 16

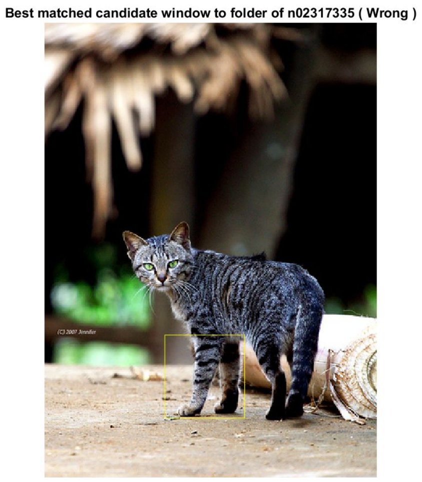

Best matched candidate window to folder of n02317335 (Wrong)

Figure 17

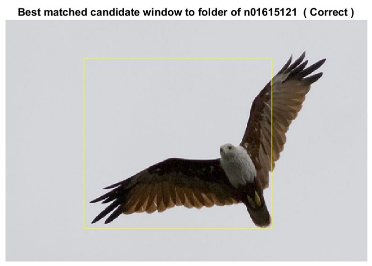

Best matched candidate window to folder of n01615121 (Correct)

Figure 18

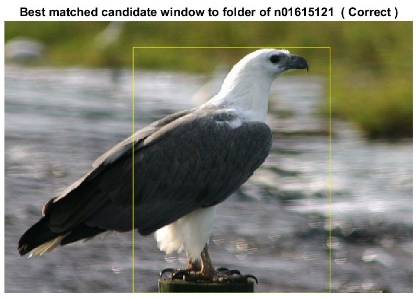

Best matched candidate window to folder of n01615121 (Correct)

Figure 19

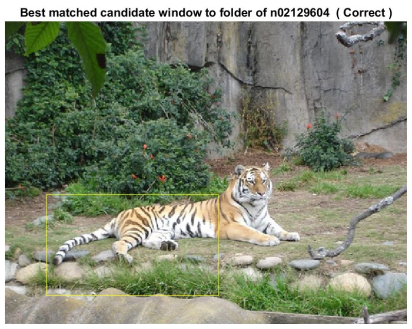

Best matched candidate window to folder of n02129604 (Correct)

Figure 20

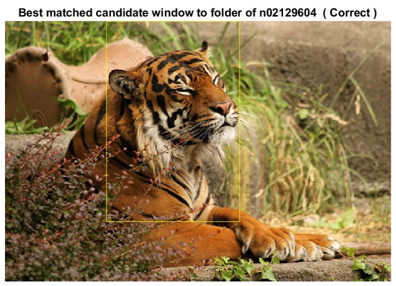

Best matched candidate window to folder of n02129604 (Correct)

Figure 21

IV. RESULT AND DISCUSSION

The results are taken into account in 2 ways, the classification accuracy and the localization accuracy. The overall classification accuracy is 91% and localization accuracy is 30% [Appendix A]. It is clear that the classification accuracy is indeed very good but at first glance, the localization accuracy does not seem to be very reliable. Our hit criterion for localization was to take the intersection of the candidate bounding box and the ground truth bounding box and divide it with their union. If the fraction is greater than 50%, we took it as a hit, otherwise it is taken as a miss. In our case, hit is rather hard to get a hit since 50% is a high percentage for collapsing 2 boxes.

In terms of individual results, itis possible to see that the images of 6 classes were classified 100% correctly which is perfection. On the other hand there are 2 classes whose images were classified with 91% success rate. There are 2 classes whose images were classified with a lower rate which are around 70%. These object type are buffalos and sea stars. Most remarkably, the buffalos are confused only with elephants. However, when we look at the images inside the dataset of the buffaloes and elephants, we can see that there are color related and texture related similarities. Therefore the mistakes are understandable.

Since the implementation of the bounding box detection is not done us, it would not be reliable to argue about the difficulty of detecting the boxes. However we can look at the bounding boxes of some of the images as reference. For instance Figure 14 shows a gazelle image with its bounding box almost perfectly covering the animal. However, there are some other images whose top scoring bounding box is not very reliable and its result is not correct, such as the cat on the Figure 17. The bounding box takes only the leg of the cat therefore the classification is not correct. On the other hand, there are some interesting results as well, for instance on the Figure 13, we can see that the bounding box has taken the human face yet it classified the image as a monkey image. Perhaps this is caused by the small class number, since the model could not fit the human face into any other animal and classified it as a monkey.

Overall, the model that was used in this project seems to be very reliable in terms of classification accuracy. 91% is arguably one of the best classification results possible. However, it should not be deceiving because as mentioned earlier, the number of classes is rather small for real life classification problems. But for these kinds of small number of classes, usage of the pretrained deep neural network ResNet50, along with binary SVM can in fact be useful. This model is proven to be useful for another purpose and that is the fact that even though the dataset was rather small the classification was successful. This might be the result of the usage of bounding boxes of candidate local objects since the model goes through the images in much more details.

On the other hand this causes a big problem, which is the inefficiency of the running time. Overall, execution of the program for 400 images take around 2 hours which makes the model only to be used for long-term studies. However, the running time could be shortened by using the GPU for the matrix operations since they are designed to efficiently do matrix operations.

Overall, the results of this project are indeed very satisfactory and they showcased the importance of detailed feature extraction and usages of machine learning and classification scores. It is possible to say that this quiet basic model can be used for small datasets if the aim of the operation is to increase the accuracy.

References

[1] "Downloads - Anaconda", Anaconda, 2019. [Online]. Available: https://www.anaconda.com/download/. [Accessed: 10- Jan- 2019].

[2] "Torchvision :: Anaconda Cloud", Anaconda.org, 2019. [Online]. Available: https://anaconda.org/soumith/torchvision. [Accessed: 10- Jan- 2019].

[3] "Pillow :: Anaconda Cloud", Anaconda.org, 2019. [Online]. Available: https://anaconda.org/anaconda/pillow. [Accessed: 10- Jan- 2019].

[4] "Scipy :: Anaconda Cloud", Anaconda.org, 2019. [Online]. Available: https://anaconda.org/anaconda/scipy. [Accessed: 10- Jan- 2019].

[5] "Tqdm :: Anaconda Cloud", Anaconda.org, 2019. [Online]. Available: https://anaconda.org/conda-forge/tqdm. [Accessed: 10- Jan- 2019].

[6] "Image Processing Toolbox", Mathworks.com, 2019. [Online]. Available: https://www.mathworks.com/products/image.html. [Accessed: 10- Jan- 2019].

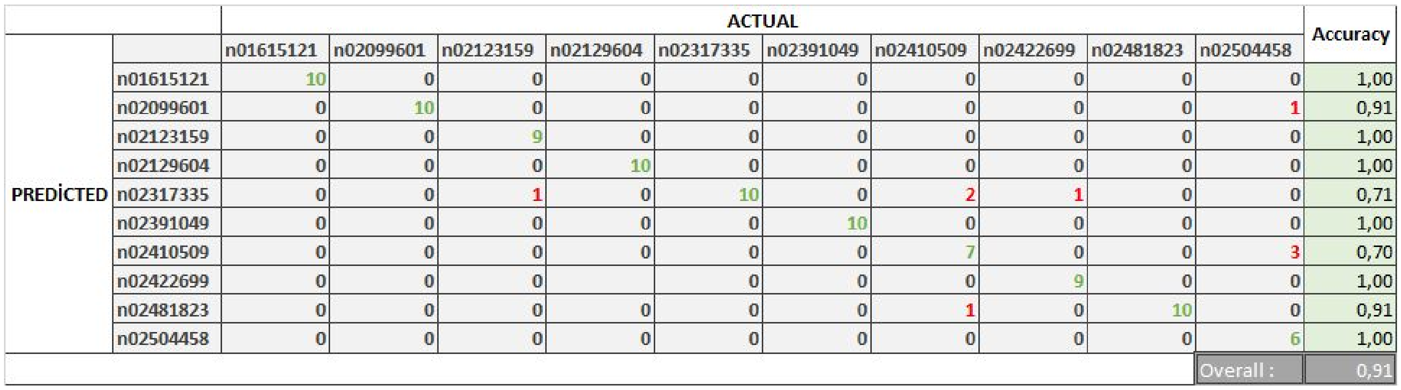
[7] "Statistics and Machine Learning Toolbox", Mathworks.com, 2019. [Online]. Available: https://www.mathworks.com/products/statistics.html?s_tid=srchtitle. [Accessed: 10- Jan- 2019].

[8] "pdollar/edges", GitHub, 2019. [Online]. Available: https://github.com/pdollar/edges. [Accessed: 10- Jan- 2019].

[9] "Piotr's Matlab Toolbox", Pdollar.github.io, 2019. [Online]. Available: https://pdollar.github.io/toolbox/. [Accessed: 10- Jan- 2019].

APPENDIX A

Table 1- Test image classification results



		ACTUAL										Accuracy
		n01615121	n02099601	n02123159	n02129604	n02317335	n02391049	n02410509	n02422699	n02481823	n02504458	
	n01615121	10	0	0	0	0	0	0	0	0	0	1,00
	n02099601	0	10	0	0	0	0	0	0	0	1	0,91
	n02123159	0	0	9	0	0	0	0	0	0	0	1,00
	n02129604	0	0	0	10	0	0	0	0	0	0	1,00
PREDİCTED	n02317335	0	0	1	0	10	0	2	1	0	0	0,71
	n02391049	0	0	0	0	0	10	0	0	0	0	1,00
	n02410509	0	0	0	0	0	0	7	0	0	3	0,70
	n02422699	0	0	0		0	0	0	9	0	0	1,00
	n02481823	0	0	0	0	0	0	1	0	10	0	0,91
	n02504458	0	0	0	0	0	0	0	0	0	6	1,00
											Overall :	0,91

APPENDIX B

Şehmuz Ali Subay: worked on code to pad the images and resize the images. Feature extraction code was written. Classification of test image up to the features of trained images was done. Images with best matched candidate window were prepared. Confusion matrix, accuracy table was drawn.

Arkın Yılmaz: worked on preprocessing the input images and feature extraction. Mainly worked on training the binary SVM classifiers. Helped to localize objects. Computed the localization accuracy by using overlap ratio.

Deniz Şen: worked on binary SVM classifiers, worked on the bounding box extraction, candidate window preprocessing, computation of the suitable windows and classification and elimination of the candidate edge boxes in the test step.