CS 484, Fall 2019 Term Project: Object Localization and Recognition

Ahmet Furkan Bıyık
21501084

Zeynep Nur Öztürk
21501472

Ebru Kerem
21501859

***Abstract*— We propose a method for image classification and object localization. The method we proposed using a framework that is similar to the R-CNN (region-based convolutional neural network) model. This method helps us to classify and localize a set of animals based on a pre-trained ResNet-50 on the ImageNet dataset for feature extraction, 2-layer feed-forward neural network classifier to classify the features, and the Selective Search Region method to extract regions of interest. We conclude that this method of object localization is of satisfactory quality and efficiency.**

Keywords—classification, localization, R-CNN, ResNet-50, SGD, region of interests

I. INTRODUCTION

Object localization and recognition are some of the areas of computer vision that are the most fundamental and challenging problems. In this report, we discuss the implementation of the pipeline proposed in the term project specification for the CS484 Image Analysis, Fall 2019. The pipeline provided in the aforementioned specification consists of

- Preprocessing which consists of padding images with zeros until it becomes square image, resizing the image and applying normalization.
- Extract 2048-dimensional features from ResNet-50
- Using the ResNet-50 features and the object labels, train a 2-layer feed-forward neural network classifier
- Use SGD algorithm in trained model
- Selective Search Region method to facilitate the localization of objects in an image with objects at multiple scales.
- Evaluation metric to measure the accuracy of the proposed method

We aim to recognize 10 different classes in the images which are animals such that eagle, dog, cat, tiger, starfish, zebra, bison, antelope, chimpanzee, and elephant. We train our 2-layer feed-forward neural network classifier over 398 labeled test images in a 2048-dimensional feature space, as per the output of ResNet-50 and we use SGD to optimize the overall program. For testing, we use a dataset of 100 photos and attempt to classify these photos by extracting interesting regions via the Selective Search Region algorithm then we classify the bounding boxes using the 2-layer feed-forward neural network classifier. First, the image is over-segmented. Then, a hierarchy of segments is constructed by iteratively merging two neighboring segments that are similar to each other based on color, texture, size, and shape compatibility, 4 until all segments are merged into a single segment that is the whole image. At each iteration, the bounding box of the newly-formed segment is added to the list of region proposals. Lastly, we evaluate the quantitative performance of the model which is done in two stages:

1. Classification accuracy: Compute the confusion matrix, precision and recall for each object type, and the overall accuracy in terms of the percentage of correctly classified test images. Present these results as tables
2. Localization accuracy: Compute the percentage of correctly classified and localized test images. Here, a test image is considered as correctly classified and localized if it is assigned to the correct class, and (if correctly classified) the localization output has a bounding box overlap ratio above 50%

The bounding box overlap ratio between a ground truth box g and a candidate window b is measured by (1).

$$\frac{g \cap b}{g \cup b} \quad (1)$$

II. METHODOLOGY AND APPROACH

A. Preprocessing

An image first loaded into the program and we turn this image into RGB image. We found the maximum value of height and weight and we padded the minimum ones with zeros until it becomes same size with the maximum one. New size of image could be found with (2).

$$\text{Size of image} = (\max(w, h))^2 \quad (2)$$

After pad operation, we resized the images to 224*224 scale. Finally, we applied normalization to the image by making the following assignments (3) and (4) for the color channels, denoted as I_r, I_g, I_b respectively for the red, green, and blue channel.

$$(I_r, I_g, I_b) := \frac{(I_r, I_g, I_b)}{255} - (0.485, 0.456, 0.406) \quad (3)$$

$$(I_r, I_g, I_b) := \left(\frac{I_r}{0.229}, \frac{I_g}{0.224}, \frac{I_b}{0.225}\right) \quad (4)$$

B. Feature Extraction

Features are extracted from the preprocessed image by feeding these images to a pre-trained ResNet-50, distributed alongside the project data and downloaded through PyTorch. Once preprocessing is complete, we append images to an augmented dimension to indicate batch size which is one. Then, we transpose the images to make them suitable for feature extraction function and get images of size [batch_size, 3, im_height, im_weight]. Finally, we transformed images into torch.FloatTensor and feed the model with those images. Results gave us the type of torch.FloatTensor features, we converted them back into Numpy array and saved them in .mat files to use in the training stage. We had 2048-dimensional representations from ResNet-50 in the end.

C. Training

Using the ResNet-50 features and the object labels, we train a 2-layer feed-forward neural network classifier. The initial dimension we give feed-forward network is 2048 which is the length of the ResNet-50 feature specified in the term project specification for the CS484 Image Analysis. Then, we decrease this dimension to 10 which is the number of classes with the set of operations. This operation size to build an effective classifier defined by hidden size of the network. We determined this hidden size as 256 to get an effective classifier. At first, we tried 100 as hidden size. However, it turned out to be too small for a properly working neural network. After we define the model, we train it using the optimization algorithm Stochastic Gradient Descent (SGD). Gradient descent is an iterative algorithm that starts from a random point on a function and travels down its slope in steps until it reaches the lowest point of that function. SGD randomly picks one data point from the whole data set at each iteration to reduce the computations enormously [1]. SGD has a learning rate element that affects the learning of the training program. However, training must be stopped at the point where loss starts increasing to prevent remembering rather than learning. We loop over the algorithm until the epoch is in range. Epoch size affects the accuracy of the results. As much as train data loops over the network, the result will be better. We determine the range of epoch as 200 for better results and speed. The final results of training are the "probabilities" that a feature belongs to a particular class.

D. Extracting Candidate Windows

Selective Search Region method was used to create bounding boxes that represent potential localizations of an object. First, the image is oversegmented. Then, a hierarchy of segments that are similar to each other based on color, texture, size and shape compatibility, until all segments are merged into a single segment that is the whole image. At each iteration, the bounding box of the newly-formed segment is added to the list. Since the selective search algorithm has a very high recall, we restrict the output at 400.

E. Classification and Localization

The images that comes from the test folder were used to extract features from the pre-trained ResNet-50 network. These features were then normalized in the same way the training image features were normalized. Normalized features extracted from the proposed object regions were classified using the previously trained 2-layer feed-forward classifier. Classification outcomes for each bounding box were then compared according to their assignment scores. Bounding box with the highest score was chosen to be the representative bounding box of the object that was aimed to be detected.

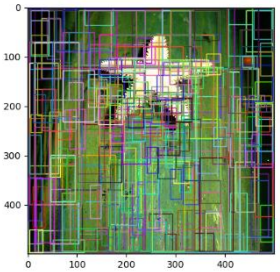
F. Evaluation

We performed evaluation in two steps. We use classification accuracy and localization accuracy. We compute the confusion matrix (see table 1) by using mlxtend.evaluate package. We compare target and predicted results in confusion matrix and compute recall precision and overall accuracy (see table 2 and table 3). Then, we compute localization accuracy by using the bounding box overlap ratio between a ground truth box g and a candidate window b is measured by (5). If this ratio is bigger than 0.5, we count localization as correct.

$$\frac{g \cap b \text{ (intersection area)}}{g \cup b \text{ (union area)}} \quad (5)$$

III. Results

In the below, you can observe test results of selective search region bounding boxes. We use a total of 400 boxes per image. We achieved overall 0.990 accuracy and in the localization, we achieved 32/100 correct ratio. Images in figure 1 show all 400 candidate windows for each category. Images in figure 2 show two windows. Red window is actual bounding box and blue window is the highest rated candidate bounding box by network.



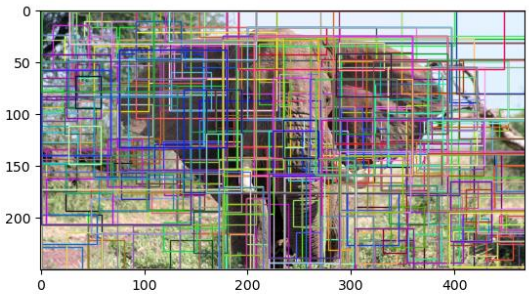
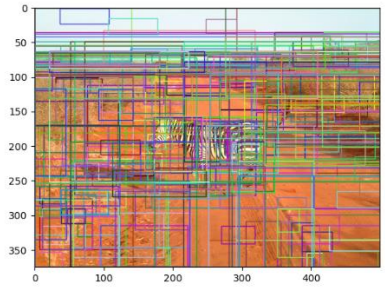
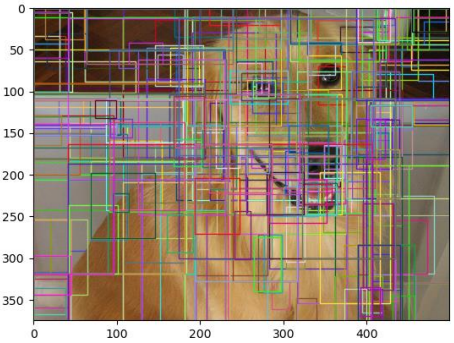
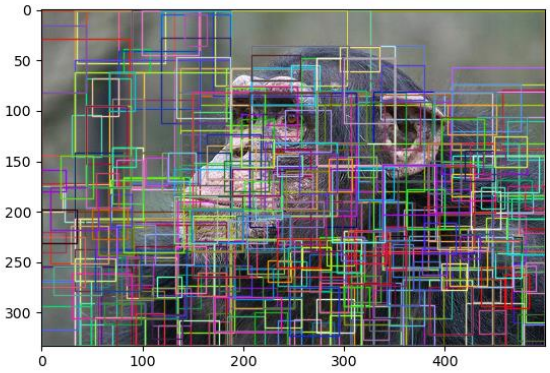


Figure 1: Selective Search Region method for bounding boxes







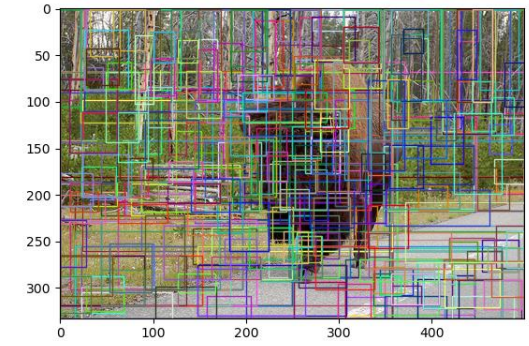
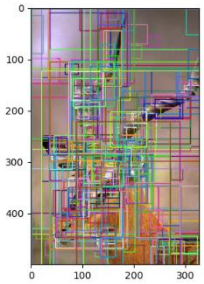
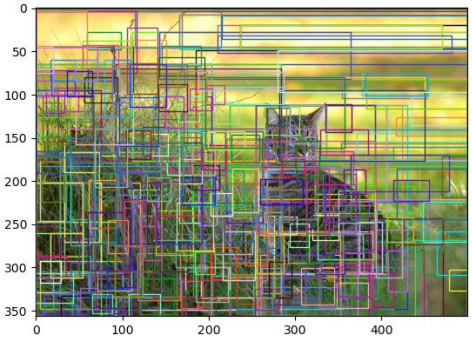
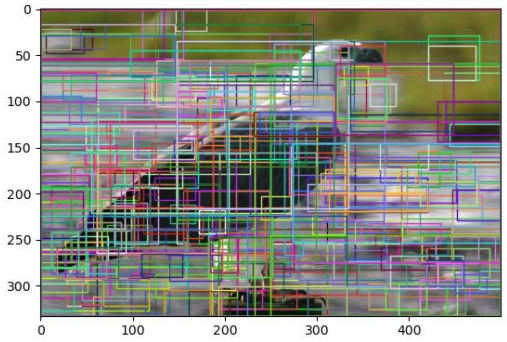


Figure 1: Selective Search Region method for bounding boxes







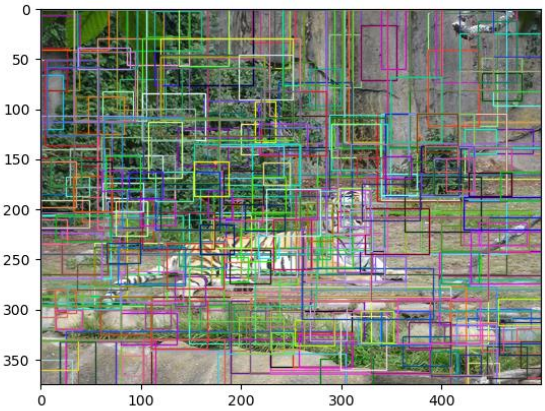
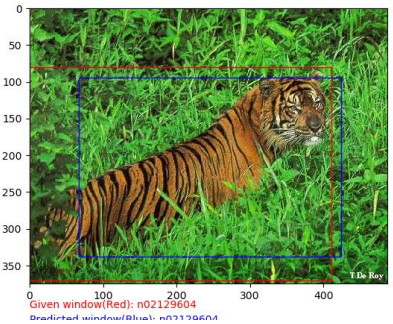
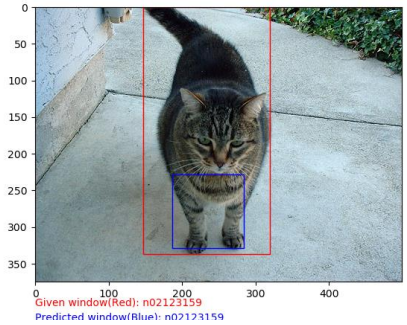
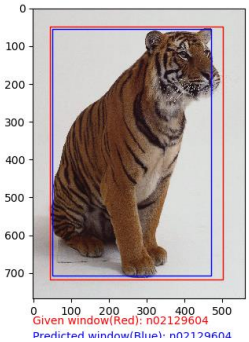
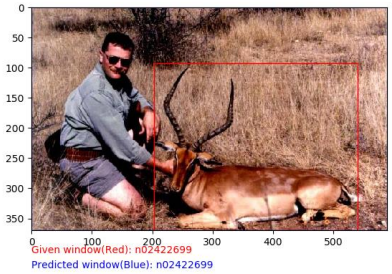


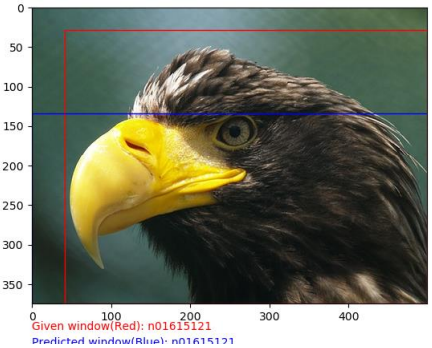
Figure 1: Selective Search Region method for bounding boxes

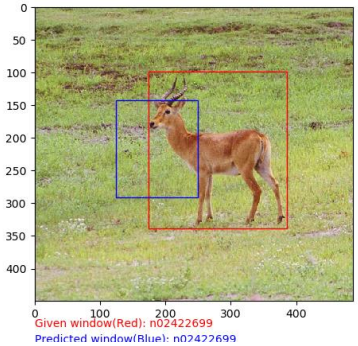


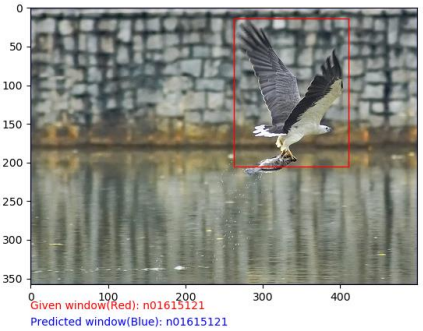


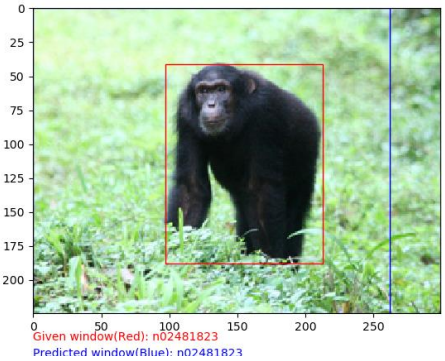


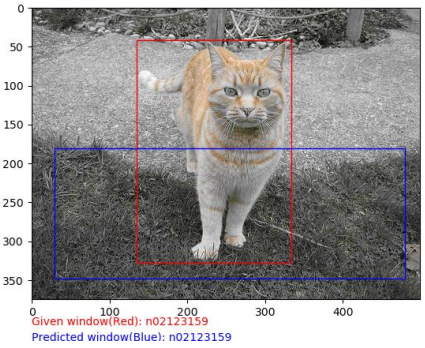












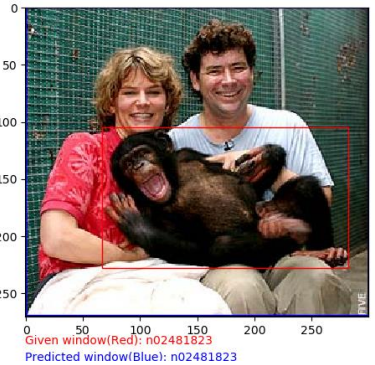
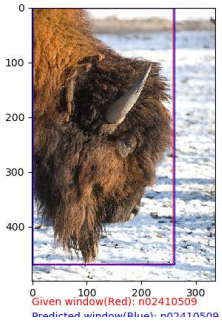
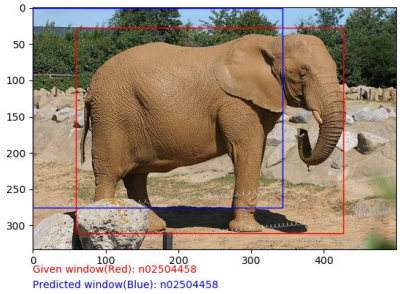
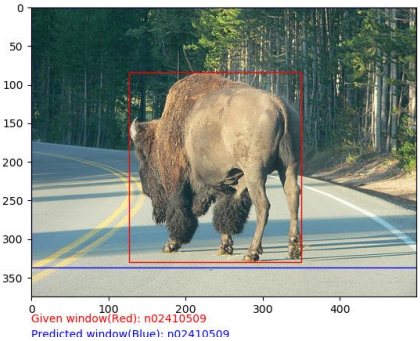


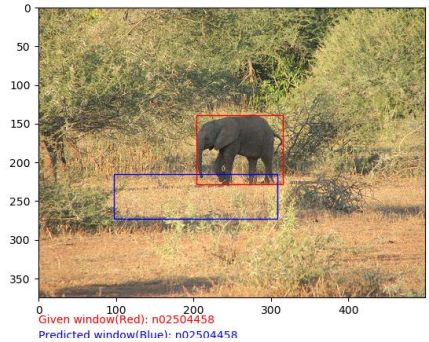
Figure 2: Classification and localization

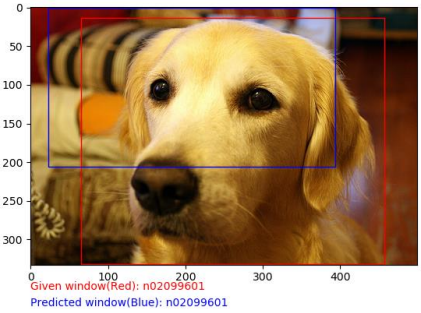
Figure 2: Classification and localization

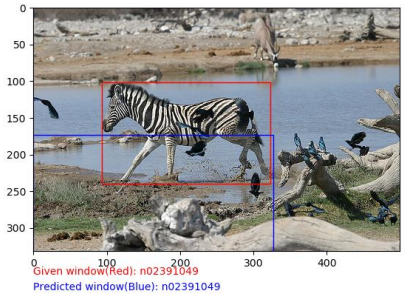


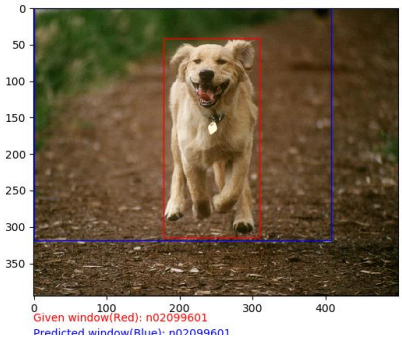


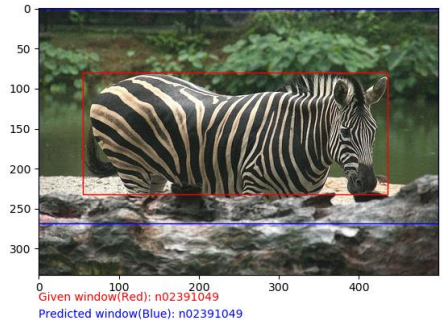












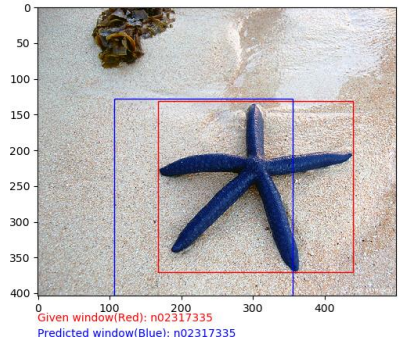


Figure 2: Classification and localization

Figure 2: Classification and localization

		Predicted Class									
		n01615121	n02099601	n02123159	n02129604	n02317335	n02391049	n02410509	n02422699	n02481823	n02504458
Actual Class	n01615121	10	0	0	0	0	0	0	0	0	0
	n02099601	0	10	0	0	0	0	0	0	0	0
	n02123159	0	0	10	0	0	0	0	0	0	0
	n02129604	0	0	0	9	0	1	0	0	0	0
	n02317335	0	0	0	0	10	0	0	0	0	0
	n02391049	0	0	0	0	0	10	0	0	0	0
	n02410509	0	0	0	0	0	0	10	0	0	0
	n02422699	0	0	0	0	0	0	0	10	0	0
	n02481823	0	0	0	0	0	0	0	0	10	0
	n02504458	0	0	0	0	0	0	0	0	0	10

Table 1: Confusion Matrix

Category	Recall	Precision
n01615121	1.000	1.000
n02099601	1.000	1.000
n02123159	1.000	1.000
n02129604	0.900	0.909
n02317335	1.000	1.000
n02391049	1.000	1.000
n02410509	1.000	1.000
n02422699	1.000	1.000
n02481823	1.000	1.000
n02504458	1.000	1.000

Table 2: Recall and Precision for each category

Overall Recall	0.990
Overall Precision	0.991
Overall Accuracy	0.990
Localization Accuracy	0.32

Table 3: Overall results

In the selective region search algorithm, it is expected to get output more than a thousand proposals. A large number of proposals creates a bottleneck in our system. Thus, we restrict it by discarding boxes that are too small to contain an object using a threshold. We find the maximum classification score among all candidate windows and classes for a test image, we assign the image to the class with the highest score among all windows, and we use the top-scoring candidate window as the object localization result. If the number of windows is higher, then the result becomes more accurate. However, since efficiency of the code depends on the number of window we restrict the number of windows to 400. Although we decrease numbers, the program still works in 12 hours because it preprocesses every image for 400 times for 400 windows.

Regarding the training process, we experimented with the training with different batch sizes, epochs, learning rates and also hidden sizes. We used hidden size in the neural network part. To decrease 2048-dimension layer to 10, we experimented with different hidden size layers. We started with 32-dimension and we incremented the size by multiplication of 2 until 512-dimension. We got better results in size 256 and we decided to use that. For the learning rate, we realized that each different algorithm needs different learning rate. We tried different values for learning rate and we observed 0.0005 is the best one. We observed that if training does not stop at the point where loss starts increasing, it remembers rather than learning. Batch size defines the number of training examples using in one iteration. For batch size, we train our model with 4, 8, 16, 32 and 64 size. We observed that 32 is the most effective one of all of them. 4, 8, 16 and 64 gave us incompetent results. And lastly, as the number of epochs increases, the number of times the weight is changed in the neural network and the curve goes from underfitting to optimal to overfitting curve. We tried several epoch numbers but we have seen that the best one we can handle is 800 we fixed our epoch size to 800.

IV. DIVISION OF LABOR

We divided the work equally amongst ourselves.

V. REFERENCES

[1] "Stochastic Gradient Descent" . https://towardsdatascience.com/stochastic-gradient-descent-clearly-explained-53d239905d31. Accessed [9.1.2020]