Object Localization and Recognition

CS484, Fall 2019 Term Project

1st Bartu Atabek
21602229
Bilkent University
Computer Engineering Department

2nd Utku Görkem Ertürk
21502497
Bilkent University
Computer Engineering Department

3rd Cagla Sozen
21602547
Bilkent University
Computer Engineering Department

***Abstract*—The goal of this project was to develop a method for image classification and object localizations on 500 images provided from the ImageNet dataset, 398 being train images and 102 being test images. Our approach was to use a pre-trained ResNet-50 using PyTorch for feature extraction and use the extracted features on a custom 2-layer feed-forward neural network classifier to obtain image classification with 10 classes and to localize objects using candidate windows extracted by the Selective Search algorithm for the objects in the image.**

***Index Terms*—Object recognition, localization, feature extraction, transfer learning, neural networks, ImageNet, Res-Net50, selective search**

I. Introduction

Aim of this project was to localize objects in images correctly with accurate bounding boxes and classify into one of the 10 categories. Images used in this project, as well as the classes, were taken from ImageNet [1] database. Classes that the images were categorized into were: eagle, dog, cat, tiger, starfish, zebra, bison, antelope, chimpanzee and elephant.

Images used for training purposes were already preprocessed such that thy would only contain an object from the corresponding class. Samples from the training dataset is shown below in "Fig. 1" for better understanding the already preprocessed version of the training images.

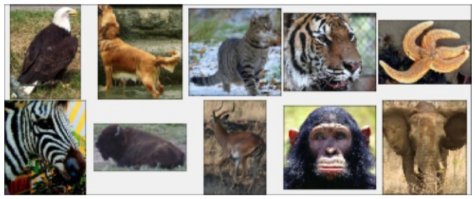

Fig. 1: Example images from the training set.

In this context, we've used an architecture similar to Region-Based Convolutional Neural Network model, where we first extract features using transfer learning with a pre-trained ResNet-50 model for extracting the features of manually pre-processed images. Methods used for pre-processing will be explained in the latter parts of this report. Then, we've defined a custom 2-layer feed forward neural network with number of hidden units being a parameter. We've used this neural network for both classification and for object localization by first extracting candidate windows with Selective Search algorithm [2] for extracting candidate windows and using the custom neural network for the classification of the candidate windows. Finally, we've chosen the top-scoring candidate window among all other windows as the object localization result and the image to the class with the highest score.

II. Approach

In this part, the methodologies used for each counter-part of the project will be discussed. Namely, implementation of *Preprocessing*, *Feature Extraction*, *Training*, *Testing* and *Evaluation* parts will be discussed in detail. The complete project implementation was made in Python using the PyTorch framework.

A. Pre-processing

There exists no standard aspect ratio and size for the ImageNet images. To standardize the images for feature extraction should not be dependent on the aspect size and ratio, we start with a pre-processing step where we first convert all the images into RGB and then pad the image with zeros until the image becomes square. Then we resize the image such that it will be 224x224 since it is the minimum required size for the pre-trained models of PyTorch . Finally, we normalize the images as required by all pre-trained models in PyTorch [3]. We first load the image into the [0,1] range by dividing all three channels by 255 and then normalize using *mean = [0.485, 0.456, 0.406]* and *std = [0.229, 0.224, 0.225]* .

An example of the pre-processed images can be found below in and"Fig. 2 "Fig. 3". Implementation of the pre-processing step was made using only Numpy and Math packages of Python.

B. Feature Extraction

After pre-processing, 224x224 normalized representations of all images are obtained, which are compatible with ResNet-50 model of PyTorch. For the feature extraction, a pretrained ResNet-50 Model is loaded and it's last layer, before applying softmax in the output layer, is cut out for obtaining the features.

Before applying the model on the images, we require another conversion for obtaining the feature representation we need. For this, an extra dimension is appended to the image for representation of the batch size. Then the transpose of the

Fig. 2: Sample Non-Preprocessed Image from ImageNet Dataset

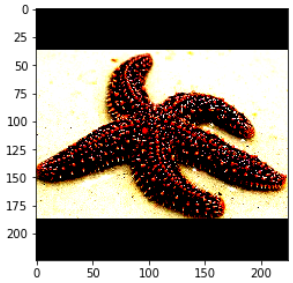

Fig. 3: Sample Pre-processed Image

image is taken such that it would be compatible with the input format of the model. Finally, once again for compatibility, the Numpy image is converted into *torch.FloatTensor* and then the model is applied to the image and features. Extracted features are in the form of a 2048 dimension vector.

C. Classification Model

For classification of the images and candidate windows, a custom 2-layer feed forward neural network was implemented with the following properties.

- **Input Layer— Input Size:** 2048 **Output Size:** 1375
- **Output Layer— Input Size:**1375 **Output Size:** 10
- **Activation Function:** ReLU

Input size of the input layer was determined to be 2048 for compatibility with the feature vectors and the output size pf the output layer was determined to be 10 for representing the 10 classes. The hidden unit size between layers and activation function were determined empirically.

D. Training the System

For training the model described above, parameters that fit best to our problem were selected initially and experiments were made to find the optimizer, loss function and learning rate yielding the best results. For instance, Cross Entropy loss was used because of it's compatibility for multi-class classification tasks. Furthermore, maximum number of epochs were determined according to the convergence of the loss at the end of each epoch. Hence, decisions for the training were made both and empirically.

- **Maximum Number of Epochs:** 10
- **Loss Function:** Cross Entropy Loss
- **Optimizer:** Adam
- **Learning Rate:** 0.001

For comparing the predictions with the ground truth values, the maximum of the 10 dimensional output vector from the model was chosen and compared.

E. Testing the System

Testing the system was performed in two stages since the target objects appear in natural scenes with a background in the test images. In the first stage, we've extracted candidate windows using Selective Search Algorithm [2] and then the model was applied to these candidate windows after they are pre-processed to find the window with the highest scores to make the classification and object localization.

1) Extracting Candidate Windows: For predicting the bounding boxes of objects in the images, first region proposals were obtained using Selective Search Algorithm [2] which outputs a list of proposals for the regions in the images. Each of these proposals were pre-processed as described in *Section II-A* for standardization and compatibility with the model.

2) Classification and Localization : After the pre-processing is complete for the candidate windows, their feature vectors are obtained as described in *Section II-B*. Then, the feature representations were given to the model. Finally, the class with the highest score was assign to the image among all windows and the top-scoring candidate window was chosen as the object localization result. These results were evaluated individually according to accuracy, which will be discussed in the proceeding section.

F. Evaluation

For the classification accuracy, the confusion matrix was computed, precision and recall for each class was calculated. For localization accuracy, the percentage of correctly classified images were calculated when the ground truth values of the bounding boxes and the predicted bounding boxes overlapped over a ratio over 50%. The evaluation was made with all 102 images in the test set. Detailed discussion of the performance metrics used for evaluation can be found in *Section III* .

III. Performance Metrics

Precision and recall was used as the performance metrics for the classification tasks. These metrics were calculated individually for each class. Also, the overall accuracy among all classes was used to measure the overall performance. This was also calculated during loss calculation to better observe the performance of the model during training.

For calculating the metrics, precision and recall, the confusion matrix was computed and the resulting matrix is shown in "Fig. 5"

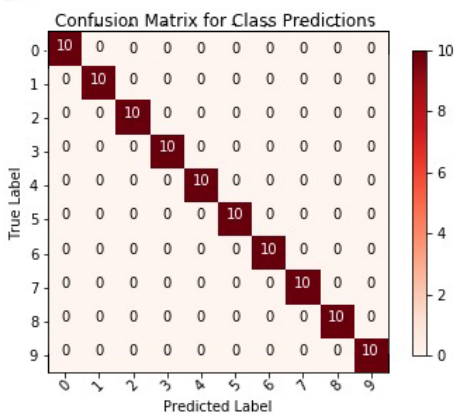

Fig. 4: Confusion Matrix for Classification.

After the calculation of the confusion matrix, its content was used to calculate the Precision and Recall for each class. Classification statistics for each class in terms of precision and recall can be found in "Table. I"

TABLE I: Classification Statistics

Class	Precision	Recall
Starfish	1.0	1.0
Elephant	1.0	1.0
Zebra	1.0	1.0
Dog	1.0	1.0
Bison	1.0	1.0
Monkey	1.0	1.0
Antelope	1.0	1.0
Cat	1.0	1.0
Eagle	1.0	1.0
Tiger	1.0	1.0

Overall accuracy among all classes was found to be **100%**.

For localization accuracy, bounding box overlap ratio was calculated and the results can be interpreted from "Fig. 5" for all 100 images.

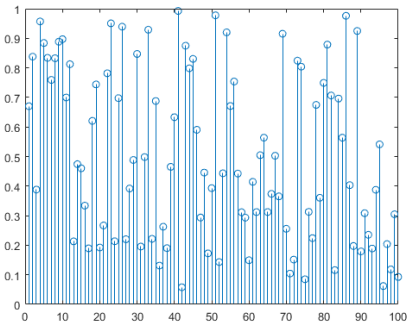

Fig. 5: Localization Accuracies.

IV. Results

A. *Object Proposals*

For each image there are approximately 600 box proposals. Thus, for the visualization of the box proposals on images, minimum width and height is chosen as 100 px.

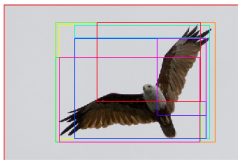

Fig. 6: Object Proposal Sample for Class 0

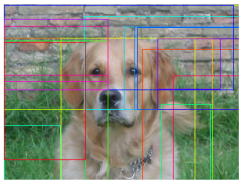

Fig. 7: Object Proposal Sample for Class Eagle

Fig. 8: Object Proposal Sample for Class Dog

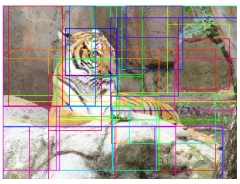

Fig. 9: Object Proposal Sample for Class Cat

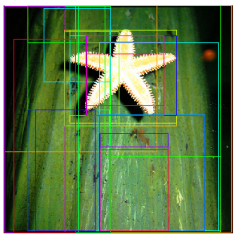

Fig. 10: Object Proposal Sample for Class Tiger

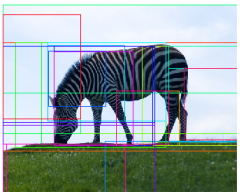

Fig. 11: Object Proposal Sample for Class Starfish

Fig. 12: Object Proposal Sample for Class Zebra

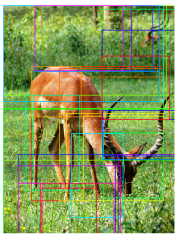

Fig. 13: Object Proposal Sample for Class Bison

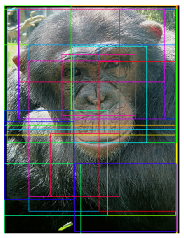

Fig. 14: Object Proposal Sample for Class Antelope

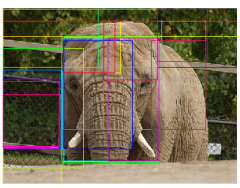

Fig. 15: Object Proposal Sample for Class Elephant

B. Localization

For object localization, one correct and one incorrect object localization image was included for each class.

1) Antelope Class: Examples of correct and false localization results for the Antelope Class can be found in “Fig. 16”and“Fig. 17”

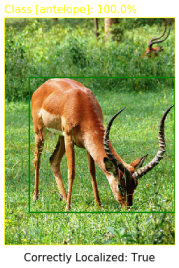

Fig. 16: Correct Localization Result for Antelope Class

Fig. 17: Incorrect Localization Result for Antelope Class

2) Bison Class: Examples of correct and false localization results for the Bison Class can be found in “Fig. 18”and“Fig. 19”

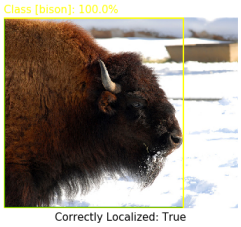

Fig. 18: Correct Localization Result for Bison Class

Fig. 19: Incorrect Localization Result for Bison Class

3) Cat Class: Examples of correct and false localization results for the Cat Class can be found in “Fig. 20”and“Fig. 21”

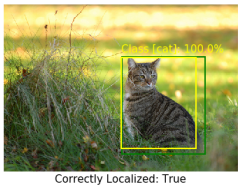

Fig. 20: Correct Localization Result for Cat Class

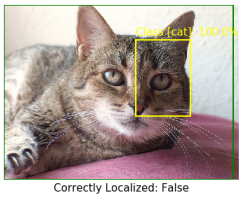

Fig. 21: Incorrect Localization Result for Cat Class

4) Chimpanzee Class: Examples of correct and false localization results for the Chimpanzee Class can be found in “Fig. 22”and“Fig. 23”

Fig. 22: Correct Localization Result for Chimpanzee Class

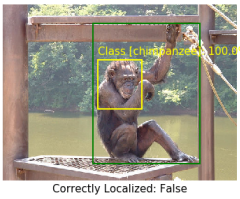

Fig. 23: Incorrect Localization Result for Chimpanzee Class

5) Dog Class: Examples of correct and false localization results for the Dog Class can be found in “Fig. 24”and“Fig. 25”

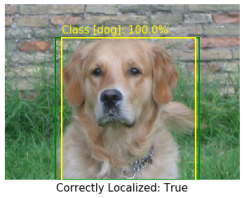

Fig. 24: Correct Localization Result for Dog Class

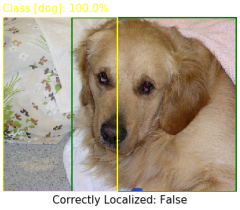

Fig. 25: Incorrect Localization Result for Dog Class

6) Eagle Class: Examples of correct and false localization results for the Eagle Class can be found in “Fig. 26”and“Fig. 27”

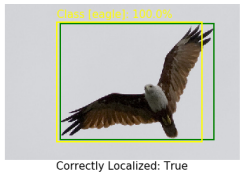

Fig. 26: Correct Localization Result for Eagle Class

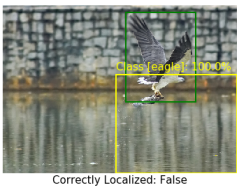

Fig. 27: Incorrect Localization Result for Eagle Class

7) Elephant Class: Examples of correct and false localization results for the Elephant Class can be found in “Fig. 28”and“Fig. 29”

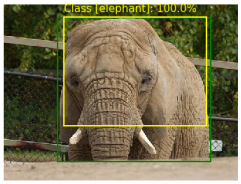

Fig. 28: Correct Localization Result for Elephant Class

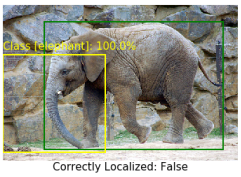

Fig. 29: Incorrect Localization Result for Elephant Class

8) Starfish Class: Examples of correct and false localization results for the Starfish Class can be found in "Fig. 30"and"Fig. 31"

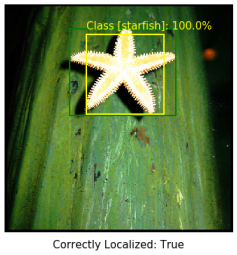

Fig. 30: Correct Localization Result for Starfish Class

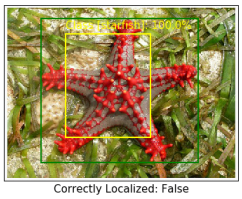

Fig. 31: Incorrect Localization Result for Starfish Class

9) Tiger Class: Examples of correct and false localization results for the Tiger Class can be found in "Fig. 32"and"Fig. 33"

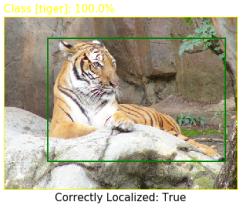

Fig. 32: Correct Localization Result for Tiger Class

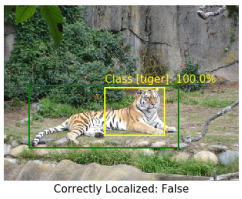

Fig. 33: Incorrect Localization Result for Tiger Class

10) Zebra Class: Examples of correct and false localization results for the Zebra Class can be found in "Fig. 34"and"Fig. 35"

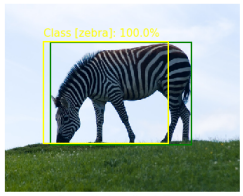

Fig. 34: Correct Localization Result for Zebra Class

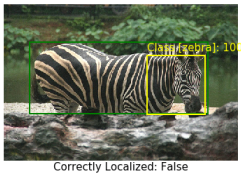

Fig. 35: Incorrect Localization Result for Zebra Class

V. Discussion

To achieve the optimum performance in our system in terms of Classification accuracy and Localization accuracy, we constructed a script to test our system according to different design decisions and parameter settings. Since we are using transfer learning with the pre-trained model of Res-Net-50 in 99% of the cases we achieved a classification accuracy of 100%. However, in order to improve the results for the localization accuracy we tested our system in terms of four different parameters and a design decision regarding the optimizer which are set as the following:

- **Batch Size:** [4, 8, 16, 32]
- **Hidden Layer Size:** [100, 500, 1000, 1375]
- **Learning Rate:** [0.01, 0.001, 0.0001]
- **Optimizer:** [SGD, Adam]

Batch sizes are often tuned to an aspect of the computational architecture on which the implementation is being executed. Such as a power of two that fits the memory requirements of the GPU or CPU hardware like 32, 64, 128, 256, and so on. And since small values result in a learning process that converges quickly at the cost of noise in the training process and whereas large values give a learning process that converges slowly with accurate estimates of the error gradient [4]. In parallel with this statement in our tests we concluded that the Batch Size value of 32 resulted in an optimal performance as computational time.

For the hidden layer size we tested these values according to the formula given below in order to prevent over-fitting [5].

$$N_h = \frac{N_s}{\alpha * (N_i + N_o)} \tag{1}$$

N_i = number of input neurons.
N_o = number of output neurons.
N_s = number of samples in training data set.
α = an arbitrary scaling factor usually 2-10.

Learning rate that is inputted to the optimizers is chosen according to the fact that a too low value never progresses, and too high value causes instability and never converges. Also, there is no learning rate that works for all optimizers. The learning rate can affect training time by an order of magnitude. Because of this, we tested three different values where "0.01" is the default learning rate for the SGD optimizer

and "0.001" is the default learning rate for the Adam optimizer. In accordance with the selected optimizer, in our findings, it was observed that using Adam as the optimizer results in faster learning. Also, standard SGD requires careful tuning of learning rates, whereas with Adam where performance is less sensitive to different learning rates.

Overall, choosing the **Batch Size** as 32, **Hidden Layer Size** as 1375, **Learning Rate** as 0.001 and the **Optimizer** as Adam results in the best performance where the classification accuracy is 100% and the localization accuracy is between 46% and 52% depends on random parameter initialization of hidden layer.

Even if we choose these values, most of our test combinations yielded similar results, if learning rate was not 0.01. This is because since we use pre-trained model, it converges 100% for training and test accuracy so their difference on results of localization accuracy are not highly distinguishable.

As it can be seen from the given results in *Section IV* among the given 10 classes our model managed to accurately identify all the classes in the given test images which was expected since it is using a pre-trained model. In terms of the localization results, our results do not go up to 52%. There are two explanations for the fact that even there is good classification accuracy, localization results are not as high as classification accuracy. First, since our model classifies 10 classes and test images only involve one class, even if localized windows does not include whole body of the animal, model gives high outputs for that specific class. For example, just head of a tiger is enough for correct classification but misses the correct localization when tigers whole body exposed in the image. So, because of this fact, even if selective search algorithm founds an ideal bounding box since there is a high possibility of classification accuracy of the ideal bounding box and the box that only includes some part of the body is same, two boxes randomly selected, more accurately first box in the array is selected. There is also another possibility in this scenario which is model trained for specific part of the body and when selective search algorithm localize feature in that region, model gives higher probability than the probability that is obtained by ground-truth or similar box. Second, because of the similar color and gradients of images around objects selective search algorithm do not give good bounding boxes. For example, some images that includes elephants merges with background which has similar,specifically brown, color. Eagles are easy to localize in terms of given test images.Since background,sky and sea, is almost uniform in these images selective search algorithm gives nearly ideal bounding boxes. Also, star fish easy to localize because of its symmetric structure. Tiger images are hard for given test images since they have color changes because of illumination and background can be mixed with tiger in terms of color. Beside individual class differences object distance to camera objective is important. If object is too close, selective search algorithm picks details of the object instead of the whole object and if object is far away it does not localize tightly but includes noise.

When we increase epoch to reach 100% training accuracy, localization accuracy become worse, mostly. This can be because model learns each feature very well and even small part of the object is enough for the model to give high probability as discussed above.

Further increasing localization can be done by observing test data but and fine tuning the selective search algorithm but this can lead that model gives only good results for given data but bad results for future images.

APPENDIX

A. Contributions

1) Bartu Atabek: In this project, we divided work equally and worked every aspect of the project for further improvements in a collaborative manner. I've collaborated with the other team members for the training of the system. I've researched for resources and sample examples for transfer learning process in Pytorch. I worked on converting the model from image based one into feature vector based one. I collaborated with Görkem for the visualization of the results, i.e. the drawing of the bounding boxes, class labels. I wrote a python script for testing the system with with different input parameters/implementations (Batch Size, Hidden Layer Size,Learning Rate, Optimizer). I also wrote the Discussions section in collaboration with others on the report.

2) Utku Görkem Ertürk: First of all we divided work equally and worked every aspect of the project for further improvements in well communicated manner. I worked on image preprocessing for the Resnet50 and perform the feature extraction. We collaboratively worked on testing and implementation of our model. I implemented localization logic and worked with Bartu on localization visualization. For the report, I contributed localization part of the discussion.

3) Çağla Sözen: For the implementation of this project, we've divide the work equally among the team members. For parts where individual work was done, all members were in contact with the others to keep the whole team up-to-date about the changes and the current status of the project. Also, all of the design decisions were made in consensus. I've worked together with Görkem and Bartu together for the training and testing of the system, where we both made the implementation together and design decisions for the system as well as debugging. For my individual work in the project, I've worked on implementing the performance metrics. I've computed the confusion matrix and the statistics for the classification task. For the report, I've contributed to the sections *Introduction, Approach and Performance Metrics*.

REFERENCES

[1] L. Fei-Fei, J. Deng and K. Li, "ImageNet: Constructingalarge-scale image database", Journal of Vision, vol. 9, no.8, pp.1037-1037, 2010. Available: 10.1167/9.8.1037.

[2] R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition.IJCV, 2013

[3] "PyTorch torchvisin.models," PyTorch Documentation, PyTorch, 2020. [Online]. Available: https://pytorch.org/docs. [Accessed: Jan. 05, 2020].

[4] S. N. S. Nielsen, "How do I choose the optimal batch size?," Artificial Intelligence Stack Exchange, 01-Nov-1968. [Online]. Available: https://ai.stackexchange.com/questions/8560/how-do-i-choose-the-optimal-batch-size. [Accessed: 07-Jan-2020].

[5] R. H. R. Hyndman, “How to choose the number of hidden layers and nodes in a feedforward neural network?,” Cross Validated, 01-Aug-1960. [Online]. Available: https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw. [Accessed: 07-Jan-2020].