Faculty of Science and Engineering

Semester: (Summer, Year:2022)
B.Sc in Computer Science and Engineering

Object Echolocation using PyTorch and Deep Learning

Prepared by:
Md: Sazib Chowdhury

September 11, 2022

Abstract

The Deep Learning domain got its attention with the popularity of Image classification models, and the rest is history. Today, we are generating future tech just from a single image input. GAN models generate an image, image segmentation models mark regions accurately, and object detection models detect everything going on, like identifying the people in a busy street just from an ordinary CCTV camera.

This project explores the implementation of object echolocation using PyTorch and deep learning techniques. The goal is to develop a system capable of detecting and localizing objects within an environment, akin to biological echolocation. This project explores the implementation of object echolocation using PyTorch and deep learning techniques. The objective is to develop a system capable of detecting and localizing objects within images using echolocation principles.

Keywords: PyTorch, Deep Learning, Object Detection, Echolocation, Computer Vision.

Contents

1 Introduction **1**

1.1 Introduction 1

1.2 Design Goals/Objective 1

1.3 Library Uses 2

2 Development/Implementation **3**

2.1 Introduction 3

2.2 Structure of the project: 3

2.3 Implementation of Object Echolocation 3

3 Performance Evaluation **8**

3.1 Simulation Environment/ Simulation Procedure 8

3.2 Results and Discussions 9

3.2.1 Results 9

3.2.2 Analysis and Outcome 11

4 Conclusions **12**

5 Bibliography **13**

List of Figures

2.1 Full Data 6
2.2 Box 6
2.3 Box New Data 7

3.1 Final perdition 10
3.2 Compare side by side 11

Index of code extracts

2.1 Data label 4
2.2 xmin ymin 4
2.3 Plot 5

1. Introduction

1.1. Introduction

The Deep Learning domain got its attention.Object Echolocation is the task of locating an instance of a particular object category in an image, specifying a tightly cropped bounding box centered on the instance. Object detection and localization are crucial tasks in computer vision with numerous applications ranging from autonomous vehicles to surveillance systems. Traditional methods often rely on visual cues alone, but this project explores the use of echolocation principles combined with deep learning techniques for enhanced object detection.

We are use two model $resnet50$, $efficientnetb7$ for better results.

For object localization, we will need not just the image and label, but also the coordinates of the bounding box containing the object. We must use image annotation/object tagging tools for preparing such a dataset, and XML is a popular file format used to store coordinate values of each image.

1.2. Design Goals/Objective

Aim of this project to detect an object from an image and create Image-bounding box and to achieve high-performance object detection with improved accuracy and efficiency.

1. Aim to create a custom data-set class for Image-bounding box data-set.
2. Aim to increase images data-set by using AI.
3. Aim to create train function.
4. Aim to predict bounding box given any image.
5. Compare Prediction and draw bound box.

1.3. Library Uses

Pandas: *Pandas is a software library written for the Python programming language for data manipulation and analysis.*

NumPy: *NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.*

OpenCV: *OpenCV is a library of programming functions mainly aimed at real-time computer vision.*

Matplotlib: *It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.*

PyTorch: *PyTorch is an open source machine learning framework based on the Torch library, used for applications such as computer vision and natural language processing, primarily developed by Meta AI.*

Scikit-learn: *The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.*

2. Development/Implementation

2.1. Introduction

object detection models deal with a much bigger problem, and have to run classification for all the regions suggested by the initial region selection layer in certain architectures. Still, the idea that we can predict the coordinates like a regression problem is interesting.

2.2. Structure of the project:

1. Set up environment.
2. Configurations.
3. Collection data-set.
4. Understand the data-set.
5. Load data-set.
6. Augmentations.
7. Create Model.
8. Create train and test.
9. Evolution and Measurement.
10. Inference.

2.3. Implementation of Object Echolocation

Load the data, Define a Convolution Neural Network, Define a loss function. Train the model on the training data, Validation predict result.

PyTorch Implementation Load library

	xmin	ymin	xmax	ymax	width	height
count	186.000000	186.000000	186.000000	186.000000	186.0	186.0
mean	36.150538	48.290323	193.252688	182.704301	227.0	227.0
std	25.524316	24.898749	23.673582	24.470030	0.0	0.0
min	1.000000	1.000000	88.000000	121.000000	227.0	227.0
25%	18.250000	29.000000	181.000000	165.000000	227.0	227.0
50%	29.000000	45.500000	197.500000	185.000000	227.0	227.0
75%	50.750000	67.000000	210.000000	201.000000	227.0	227.0
max	146.000000	136.000000	226.000000	227.000000	227.0	227.0

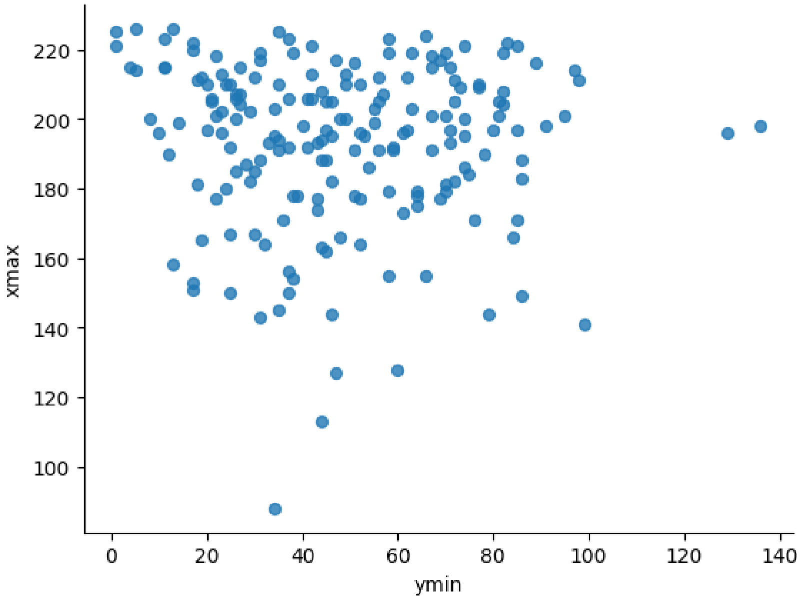
Data Visualization

Excerpt of code 2.1: Data label

```
sns.countplot(data=df,x='label')
```

Excerpt of code 2.2: xmin ymin

```
plt.figure(figsize=(12,6),dpi=200)
sns.countplot(data=df,x='xmin',order=df['xmin'].value_counts
    ().index)
plt.xticks(rotation=90);
```



ymin vs xmax

Excerpt of code 2.3: Plot

```
sns.pairplot(df)
```

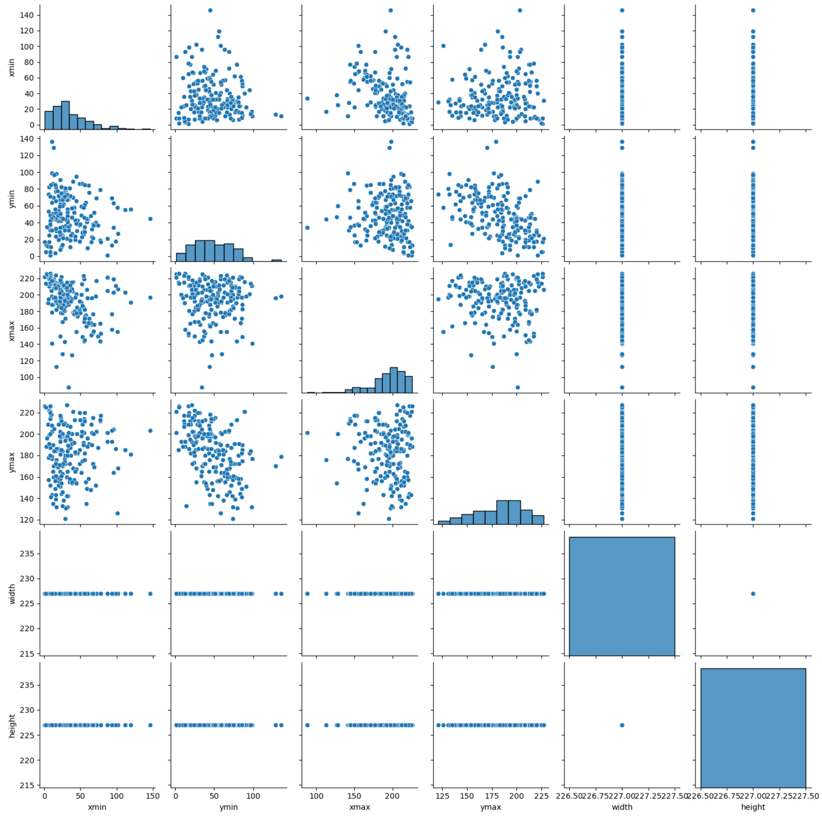


Figure 2.1: Full Data

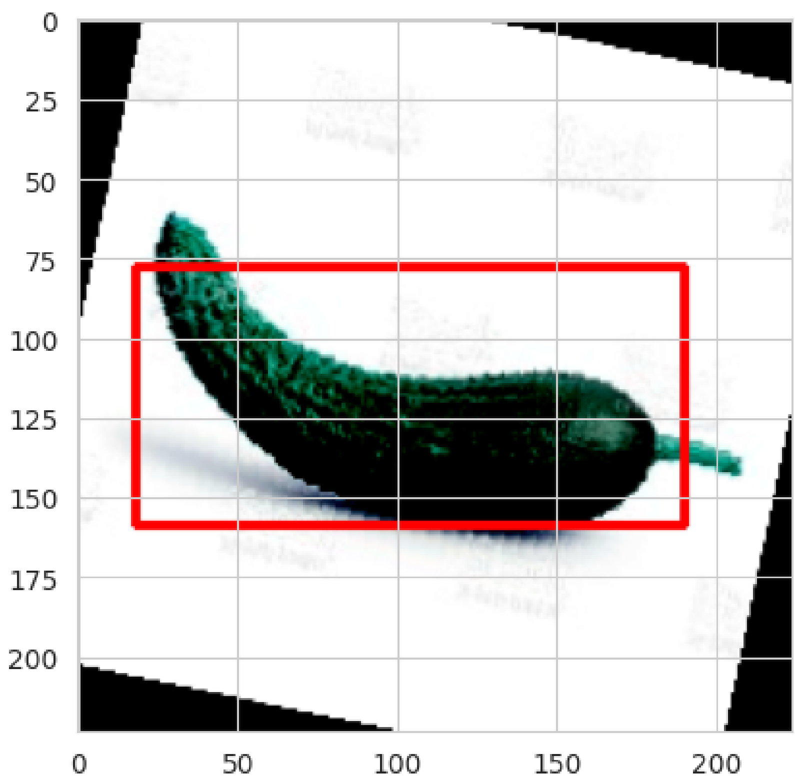


Figure 2.2: Box

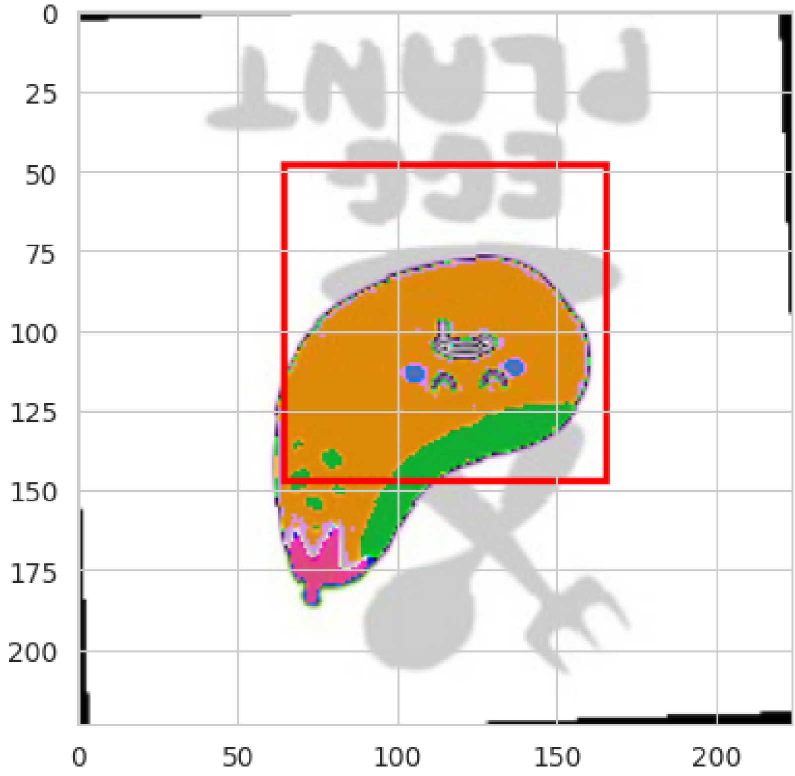


Figure 2.3: Box New Data

subfig

3. Performance Evaluation

3.1. Simulation Environment/ Simulation Procedure

Prerequisites

Supported Windows Distributions

- Windows 7 and greater; Windows 10 or greater recommended.
- Windows Server 2008 r2 and greater.

Supported macOS Version

- PyTorch is supported on macOS 10.15 (Catalina) or above.

Supported Linux Distributions

- PyTorch is supported on Linux distributions that use $glibc >= v2.17$.

Python https://www.python.org/

Python 3.7 or greater is generally installed by default on any of our supported Linux distributions, which meets our recommendation.

Anaconda https://anaconda.org/

System Requirements

A minimum RAM size of 16GB is required(Simple task around 8 GB of RAM can be employed).

A minimum storage of 512GB SSD (Solid State Drive) that can handle a 1TB HDD (Hard Drive Device). (Note: External hard drives aren't a good alternative for internal hard drives).

Processor: An Intel Core i5 or i7 processor from the at last 6^{th} generation is required.

IDE *Jupyter Notebook VS Code*

3.2. Results and Discussions

3.2.1 Results

```
# Load the pre-trained model
model = ObjLocModel()  # Replace ObjLocModel with the actual
   class of your model
model.load_state_dict(torch.load('best_model.pt'))
model.eval()

# Load the image from the specified path
image_path = "/content/dataset/train_images/cucumber_43.jpg"
image = cv2.imread(image_path)
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Convert BGR
    to RGB

# Transform the image to tensor
transform = transforms.Compose([
    transforms.ToTensor(),
])
image_tensor = transform(image)

# Perform inference on the loaded image using the model
with torch.no_grad():
    image_tensor = image_tensor.unsqueeze(0)  # Add batch
       dimension
    out_bbox = model(image_tensor)

# Visualize the bounding boxes for the loaded image
```

```
utils.compare_plots(image, None, out_bbox[0])

# Load the image and ground truth bounding box from the validation set
with torch.no_grad():
    image, gt_bbox = valid_set[20]  # Assuming valid_set contains your validation dataset
    image = image.unsqueeze(0).to(DEVICE)  # (bs, c, h, w)
    out_bbox = model(image)

    utils.compare_plots(image, gt_bbox, out_bbox[0])
```

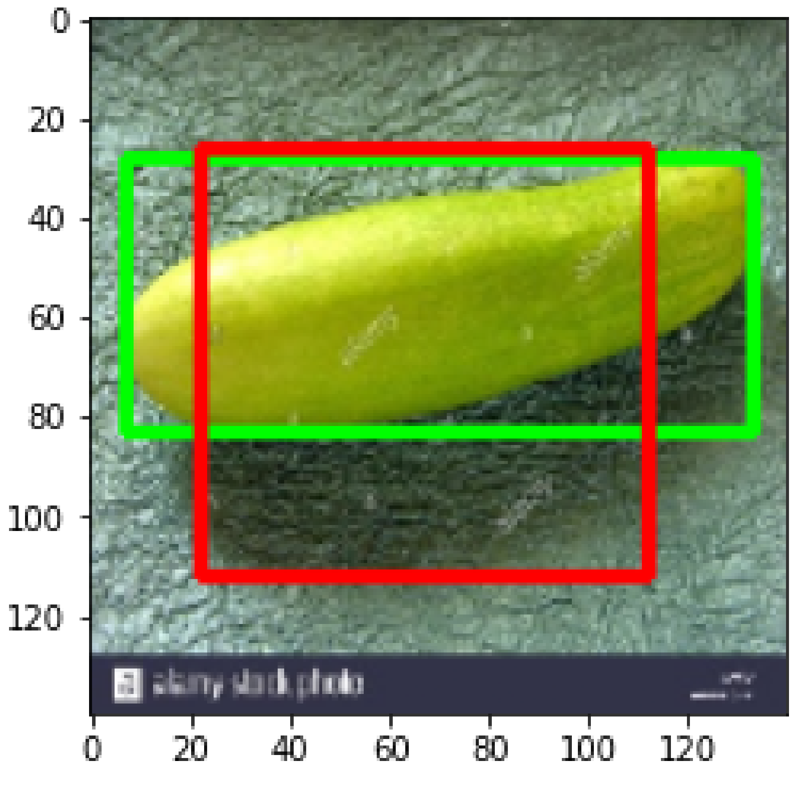
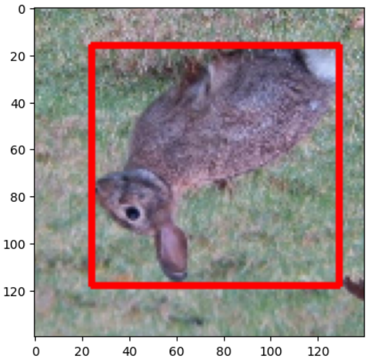
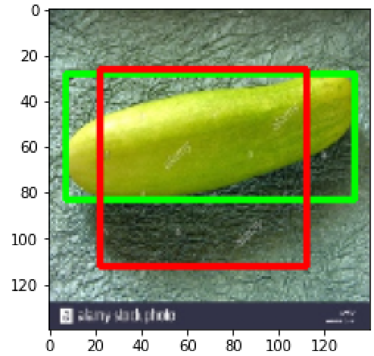


Figure 3.1: Final perdition



[Not predict]



[Predict]

Figure 3.2: Compare side by side

3.2.2 Analysis and Outcome

We got a bounding box over the image. The Model .pt file generate by pytorch then it predict the value red color is human generate light green color is predicted result.

In the prediction script, we get the model architecture first from our earlier network, and then we move the model to our preferred device: a GPU from our Gradient Notebook.

Once that's done we load the model's state_dict() as we had discussed earlier in the validation. We set the model to eval state, and use the preprocessing function to get the image ready to be fed to the model

A loss function computes a value that estimates how far away the output is from the target.

The main objective is to reduce the loss function's value by changing the weight vector values through backpropagation in neural networks.

4. Conclusions

In conclusion, the project demonstrates the feasibility of using PyTorch and deep learning techniques for object echolocation. The developed system shows promising results in detecting and localizing objects within images, laying the groundwork for further advancements in this field.

Here we do not just have to get the classification right, but get a tight and correct bounding box too. Handling both makes it complex for the model, so improving the data set and architecture will help you there.

5. Bibliography

[1] L. Bottou, “Stochastic gradient descent tricks,” Neural Netw., Tricks Trade, vol. 1, no. 1, pp. 421–436, 2012.

[2] J. Dean et al., “Large scale distributed deep networks,” in Proc. Neural Inf. Process. Syst. (NIPS), 2012, pp. 1–11.

[3] Accurate Object Localization in Remote Sensing Images Based on Convolutional Neural Networks

[4] J. Han, D. Zhang, G. Cheng, L. Guo, and J. Ren, “Object detection in optical remote sensing images based on weakly supervised learning and high-level feature learning,” IEEE Trans. Geosci. Remote Sens., vol. 53, no. 6, pp. 3325–3337, Jun. 2015.

[5] G. Cheng and J. Han, “A survey on object detection in optical remote sensing images,” ISPRS J. Photogram. Remote Sens., vol. 117, pp. 11–28, Mar. 2016.

[6] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” in Proc. Int. Conf. Mach. Learn., May 2013, pp. 1139–1147.