Object Detection with YOLO

Zaheer K Z

zaheer.work@gmail.com

1.1 Introduction

Object detection is a computer vision technique that identifies and classifies a particular object in a particular setting. The main goal of object detection is to scan digital images or real-life scenarios to locate instances of every object, separate them, and analyze their necessary features for real-time predictions. Object detection algorithms typically leverage machine learning or deep learning to produce meaningful results. Object detection is used in various domains, such as image annotation, vehicle counting, activity recognition, face detection, face recognition, video object co-segmentation, etc.

YOLOS is a state-of-the-art object detection algorithm that uses a single convolutional neural network to predict bounding boxes and class probabilities for multiple objects in an image. It is fast, accurate, and generalizable to different modalities. YOLOS is based on the idea of segmenting an image into smaller images. The image is split into a square grid of dimensions S×S. The cell in which the center of an object resides is the cell responsible for detecting that object. Each cell will predict B bounding boxes and a confidence score for each box. Each of these bounding boxes is made up of 5 numbers: the x position, the y position, the width, the height, and the confidence. The confidence represents the IOU between the predicted bounding box and the actual bounding box. Each cell also predicts C conditional class probabilities, one for each class. These probabilities are conditioned on the presence of an object in that cell.

The final prediction is obtained by multiplying the conditional class probabilities with the individual box confidence predictions.

YOLOS uses a Vision Transformer (ViT) as its backbone network. ViT is a novel architecture that applies transformers to computer vision tasks. Transformers are neural networks that use attention mechanisms to learn global dependencies between input and output sequences. ViT treats patches of image pixels as sequences, much like sequences of text tokens in natural language processing.

YOLOS also has a detector head that maps a generated sequence of detection representations to class and box predictions. The detector head consists of two linear layers followed by softmax and sigmoid activations.

YOLOS is trained end-to-end using a multi-task loss function that combines localization loss and classification loss. Localization loss measures how well the model predicts bounding boxes for objects, while classification loss measures how well the model predicts class labels for objects.

In this project, I aim to demonstrate how to use YOLOS models for object detection on various images and videos using Gradio, a Python library that

Object Detection with YOLO

allows us to build and share web applications for our machine learning models or data science workflows.

1.2 Data Sources

I used different types of data sources for this project, such as images, videos, webcam recordings, etc. I do not have a fixed dataset for our project, but I allow users to upload or drag and drop their own image or video files to our web application. I also allow users to record videos from their webcam using our web application.

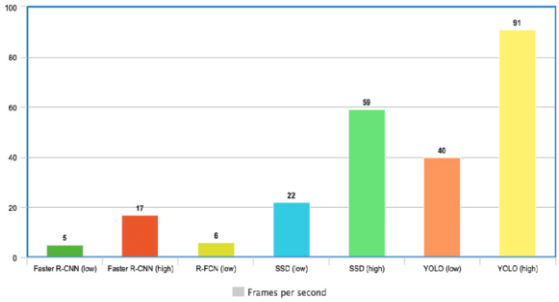
I used Gradio components to create input and output interfaces for my web application. For example, I use gr.components.Image() to create an image input component that can accept image files from users. I also used gr.components.Video() to create a video input component that can accept video files or webcam recordings from users.

I also used some predefined YOLOS models from the Hugging Face hub as our data sources. I used gr.components.Dropdown() to create a dropdown component that can let users choose from different YOLOS models available on the hub. *The models I used are:*

- yolos-tiny
- yolos-small
- yolos-base
- yolos-small-300
- yolos-small-dwr

These models have different sizes and performance trade-offs. For example, yolos-tiny is the smallest model with 6M parameters but also has the lowest accuracy. On the other hand, yolos-base is the largest model with 86M parameters but also has the highest accuracy.

Below you can see how fast YOLO is compared to other popular detectors:



1.3 Methods

I used Gradio to create a web application that can perform object detection using YOLOS models on various images and videos. I used gr.components.Page() to create separate pages for each input component in the web application. The Page class can take a list of components or interfaces and display them in different tabs.

I defined a function called model_inference() that takes an image or video and some parameters as inputs and returns an image with detected objects as output. The function uses transformers library to load feature extractors and YOLOS models from the Hugging Face hub. The function also uses PIL library to manipulate images and matplotlib library to plot results.

The function performs the following steps:

- Convert the input image or video into a PIL Image object
- Use the feature extractor to convert the image into a PyTorch tensor
- Use the YOLOS model to make predictions on the tensor
- Apply softmax function to the logits to get class probabilities

Object Detection with YOLO

- Filter out patches that have low probabilities using a threshold
- Use the feature extractor to post-process the predictions and scale them to the original image size
- Plot bounding boxes and labels on top of the image using matplotlib
- Return the image as the output

I used gr.Interface() to create interfaces for each input component and passed them to the Page class. I also used some other Gradio components to create additional inputs for our web application, such as:

- gr.components.Slider() to create a slider component that can let users adjust the probability threshold for object detection

gr.components.Textbox() to create a textbox component that can let users enter comma-separated names of classes to be shown in the output image

I also used some Gradio themes to customize the appearance of our web application, such as:

- gr.themes.Glass() to create a glass-like theme with transparent background and blur effects
- gr.themes.Monochrome() to create a black-and-white theme with minimalistic style

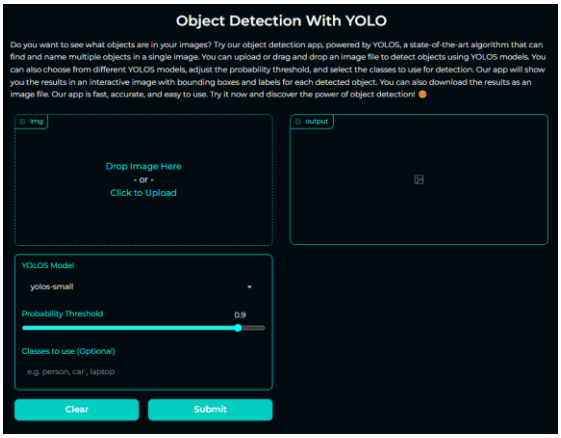
And launched our web application using the launch() method of the Page class. I also used some arguments to configure our web application, such as:

- share=True to generate a shareable link for others to interact with our web application
- debug=True to enable debugging mode and show errors and warnings in the terminal

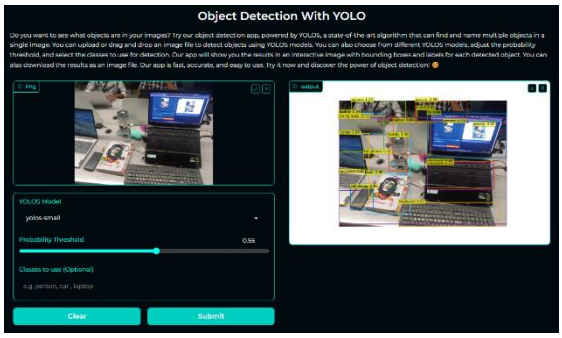
2. Results

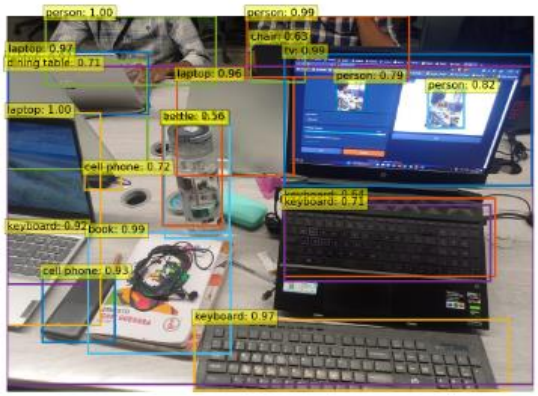
I tested my web application on various images and videos using different YOLOS models and parameters. I observed that our web application can successfully perform object detection and display the results in an interactive image with bounding boxes and labels. I also observed that different YOLOS models have different trade-offs between speed and accuracy. For example, yolos-tiny is faster but less accurate than yolos-base. I also observed that different probability thresholds and classes can affect the number and quality of detected objects. For example, a higher threshold can reduce false positives but also miss some objects. A lower threshold can increase false positives but also detect more objects. A specific class can filter out irrelevant objects but also limit the diversity of detected objects.

Here are some screenshots of our web application with different inputs and outputs:



Object Detection with YOLO





3. Conclusions

In this project, I demonstrated how to use YOLOS models for object detection on various images and videos using Gradio. I showed that YOLOS is a fast, accurate, and generalizable object detection algorithm that can work well on different modalities. I also showed that Gradio is a powerful and easy-to-use tool that can help us build and share web applications for our machine learning models or data science workflows.

I suggest some possible future work for this project, such as:

1. Adding more YOLOS models from the Hugging Face hub or other sources
2. Adding more input and output components from Gradio or other libraries
3. Adding more features and functionalities to our web application, such as downloading, sharing, or editing the results
4. Evaluating the performance and quality of our web application using metrics and feedback
5. Deploying our web application on a cloud platform or a local server

I hope that this project can inspire others to explore and experiment with object detection using YOLOS and Gradio. I also hope that this project can contribute to the advancement and dissemination of machine learning and computer vision in various domains.

References

1 https://pytorch.org/hub/ultralytics_yolov5/

2 https://github.com/ultralytics/ultralytics

3 https://neptune.ai/blog/object-detection-with-yolo-hands-on-tutorial

4 https://www.v7labs.com/blog/yolo-object-detection