

Implementing object Detection for Real-World Applications

Author: Arjun Dhage

Date of Graduation: May 2024

Project Committee chair: Dr. Prasad Kulkarni

Project Committee Advisor and co-chair: Dr. David O. Johnson

Project Committee Member: Dr. Cuncong Zhong

THE PROJECT IS ACCEPTED BY THE PROJECT COMMITTEE IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE.

<hr/> Dr. Prasad Kulkarni <hr/> PROJECT COMMITTEE CHAIR	<hr/> SIGNATURE <hr/>	<hr/> DATE <hr/>
<hr/> Dr. David O. Johnson <hr/> PROJECT COMMITTEE MEMBER	<hr/> SIGNATURE <hr/>	<hr/> DATE <hr/>
<hr/> Dr. Cuncong Zhong <hr/> PROJECT COMMITTEE MEMBER	<hr/> SIGNATURE <hr/>	<hr/> DATE <hr/>

Abstract:

The advent of deep learning has enabled the development of powerful AI models that are being used in fields such as medicine, surveillance monitoring, optimizing manufacturing processes, allowing robots to navigate their environment, chatbots, and much more. These applications are only made possible because of the enormous research in the fields of Neural networks and deep learning. In this paper, I'll be discussing a branch of Neural Networks called Convolution Neural Network (CNN), and how they are used for object detection tasks for detecting and classifying objects in an image. I'll also discuss a popular object detection framework called Single Shot Multibox Detector (SSD) and implement it in my web application project which allows users to detect objects in images and search for images based on the presence of objects. The main aim of the project was to allow easy access to perform detections with a few clicks.

1. Introduction:

In today's rapidly advancing technological landscape, computer vision, and object detection are pivotal in enhancing various aspects of our lives. The ability to accurately and efficiently identify objects within images and videos is not only technologically exciting but also holds immense practical significance.

Object detection is a critical component of computer vision and artificial intelligence. Its importance lies in its potential to bring automation and intelligence to many domains, significantly impacting efficiency, safety, and convenience. Whether it's enhancing autonomous driving systems to make our roads safer, optimizing supply chain management through automated inventory tracking, or improving medical imaging for more accurate diagnosis, object detection is at the heart of these advancements. This project aims to implement this technology for easy access and to understand how object detection systems work.

Object detection finds applications in various domains, including but not limited to:

- **Autonomous Vehicles:** Detecting pedestrians, other vehicles, and road signs for safe self-driving.
- **Retail:** Automated checkout, inventory management, and shoplifting prevention.
- **Healthcare:** Identifying anomalies in medical images, and assisting in surgical procedures.
- **Agriculture:** Monitoring crop health, pest detection, and yield estimation.
- **Security:** Intrusion detection, facial recognition, and object tracking.
- **Environmental Monitoring:** Wildlife conservation, pollution detection, and disaster management.

In this project, The focus will be on developing an object detection model with the following key objectives:

- **Data Collection:** Gather a diverse dataset relevant to the chosen application(s) and annotate it for training.
- **Model Development:** Develop a model using a Convolution neural network to detect objects in the images.
- **Model Training:** Train and optimize the selected model using the annotated dataset to achieve high accuracy and real-time performance.
- **Integration:** Implement the trained model into a real-world scenario, showcasing its effectiveness and potential impact.
- **Implementation:** Implement a User interface through which the user can interact with the application.
- **Evaluation:** Evaluate the model's performance using metrics like Average precision, Mean Average precision, recall, and F1-score, on real-world data.
- **Documentation:** Document the entire project, including methodology, results, and potential future improvements, in a clear and organized manner.
- **Future Work:** Any improvements or additional features that can be included which could enhance the functionality of the application.

2. Background:

Object detection has witnessed significant advancements in recent years, driven by the success of deep learning techniques, particularly convolutional neural networks (CNNs). Models like R-CNN, Faster R-CNN, RetinaNet, YOLO (You Only Look Once), and SSD (Single Shot MultiBox Detector) have become popular choices for object detection tasks. These models have demonstrated remarkable accuracy in detecting objects in images and videos.

Moreover, the availability of large labeled datasets, such as COCO (Common Objects in Context) and Pascal VOC, has contributed significantly to the success of object detection models. Transfer learning, where pre-trained models are fine-tuned on domain-specific data, has become a common practice to achieve state-of-the-art performance while reducing the need for extensive training data.

COCO uses JSON format to store annotated data while Pascal VOC uses XML to store annotated data.

However, not all models are equally comparable to all the other proposed models. For example, The methodology used in R-CNN, MobileNet, or VGG is different from YOLO or SSD in that the former is a two-stage detector and the latter is one one-stage detector. I'll be discussing more about the methodology in the next few topics.

In this project, I'll leverage the available datasets and models and develop a web app to allow users to interact with the model.

In the rest of the paper, I'll discuss the methodology of the process, the architecture of the model, performance, past work, results, and future work.

All the citations mentioned at the end of the paper were referred to during the work of this project and might include texts or images from these cited sources.

3. Data Collection and Pre-Processing

- **Dataset:** Pascal VOC dataset.
- **Training and Testing:** The dataset consists of 5011 images for training and validation and 9963 images for testing.
- **Classes:** These images belong to 20 classes namely 'airplane', 'bicycle', 'bird', 'boat', 'bottle', 'bus', 'car', 'cat', 'chair', 'cow', 'diningtable', 'dog', 'horse', 'motorbike', 'person', 'pottedplant', 'sheep', 'sofa', 'train', 'tvmonitor'.
- **Annotations file:** An Annotations file in XML format contains bounding boxes in x and y coordinates for the respective images.
- **Pre-Processing:** The input size of the image will be 300x300. The image will be preprocessed randomly using data augmentation techniques such as Mirroring, Cropping, Photometric Distortion, Resizing, etc.

4. Two-stage VS. One-Stage Object Detectors.

Two-stage and one-stage object detectors are the two main approaches for object detection in computer vision, and they differ in their architectural design and the way they perform object detection.

Two-Stage Object Detectors:

- **Region Proposal:** Two-stage detectors first generate region proposals in the initial stage. These proposals are potential bounding boxes that might contain objects. Common methods for generating region proposals include selective search and region proposal networks (RPN).
- **Classification and Refinement:** In the second stage, the region proposals are classified and refined. A classifier is used to determine the object class for each proposal, and bounding box regression is applied to refine the location of the proposals.
- **Examples:** Faster R-CNN, R-CNN, and Mask R-CNN are examples of two-stage detectors.

One-Stage Object Detectors:

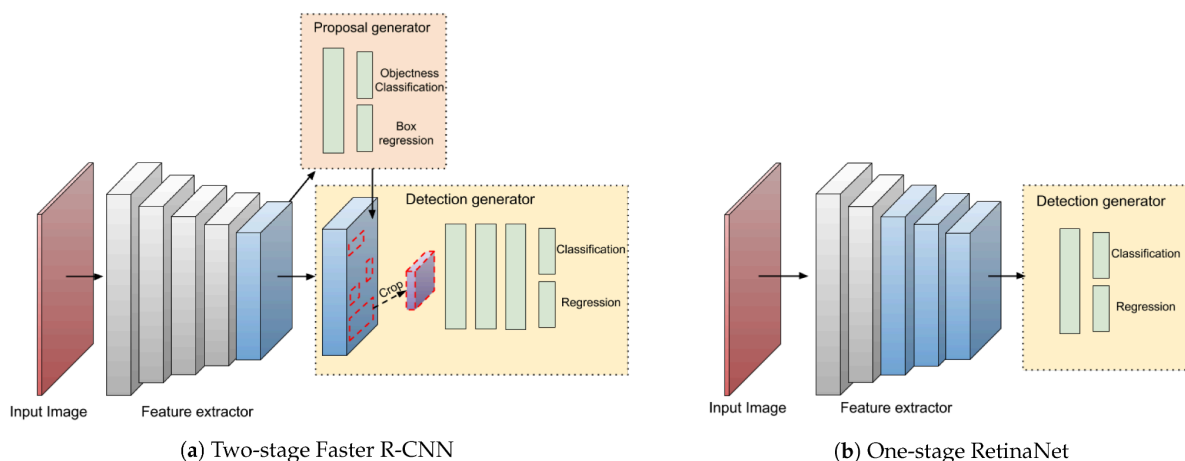
- **Simultaneous Detection:** One-stage detectors perform object detection and classification in a single step, directly on the entire image. They don't require a separate region proposal step.
- **Efficiency:** One-stage detectors are often faster and more efficient than two-stage detectors because they eliminate the need for region proposal generation.

- **Examples:** Single Shot MultiBox Detector (SSD) and You Only Look Once (YOLO) are examples of one-stage detectors.

Key differences:

- **Speed:** One-stage detectors are typically faster because they perform detection in a single pass through the network, while two-stage detectors require an additional region proposal step, which adds computational overhead.
- **Accuracy:** Two-stage detectors often achieve slightly higher accuracy because they can generate a more refined set of region proposals, which reduces the number of false positives. However, one-stage detectors have also improved significantly in accuracy.
- **Simplicity:** One-stage detectors are simpler in architecture and easier to implement, making them attractive for real-time and embedded applications.

However, Two-stage detectors are still widely used in scenarios where the highest detection accuracy is critical, such as in medical imaging or satellite imagery. One-stage detectors are popular in real-time applications where speed is of the essence like autonomous driving, robotics, and video surveillance.



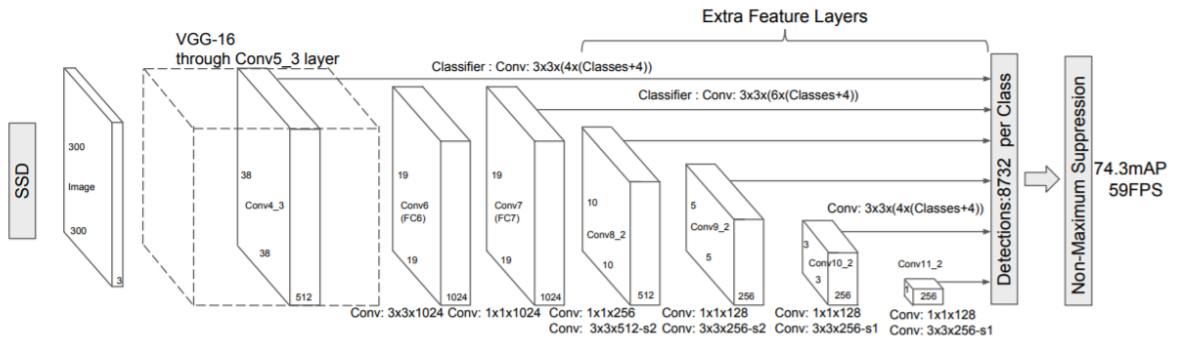
1.1 Two-stage and One-stage object detection architecture.

5. Model Development

For this project, I'll be implementing a pre-trained model called Single Shot MultiBox Detector (SSD) which is a One-stage Object Detector. I will be discussing the architecture of the models, the limitations, and any future work.

5.1 Single Shot MultiBox Detector (SSD)

The Single Shot Multi-Box Detector (SSD) is a deep learning-based object detection model where tasks of object localization and classification are done in a single forward pass of the network. It combines the benefits of both region-based and single-shot detection models, offering a good balance between accuracy and speed. Here's the Architecture and an overview of how SSD works:



1.2 Single Shot Multibox Detector Architecture

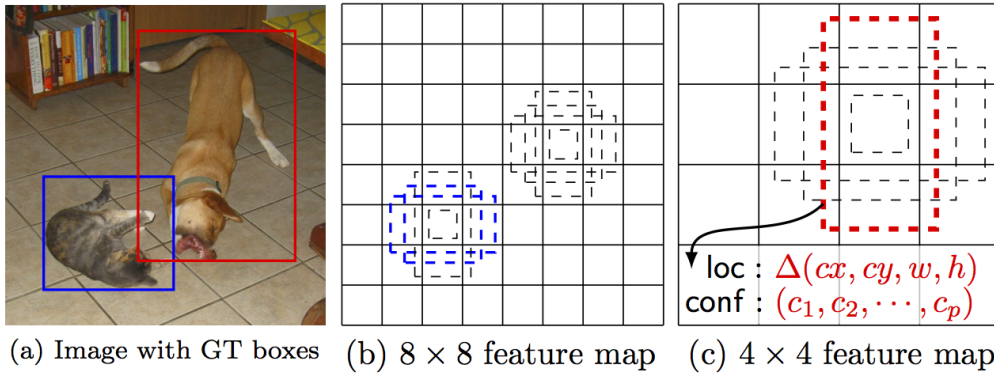
I. Feature Extraction:

SSD begins with a pre-trained convolutional neural network (CNN) known as a backbone network, such as VGG, or ResNet. In our case we will be using VGG16. This network serves as the feature extractor and is used to transform the input image into a set of feature maps.

II. Multi-scale Feature Maps:

SSD uses multiple layers from the feature extractor network to obtain feature maps at different spatial resolutions. These feature maps capture information at various scales.

III. Anchor Boxes:



1.3 Image showing Anchor boxes and feature maps.

At each feature map layer, SSD associates a set of anchor boxes with different aspect ratios and scales. Anchor boxes are predefined bounding boxes that are used as reference templates for object detection. These anchor boxes are placed at various positions within the feature map grid cells.

IV. Localization and Classification:

For each anchor box, SSD predicts two types of information:

- **Localization:** It denotes the offset (x, y, width, height) from the anchor box to the actual bounding box of an object.
- **Classification:** It predicts the class probabilities for each anchor box, indicating which object class it might belong to.

V. Non-Maximum Suppression (NMS):

To eliminate duplicate detections, SSD applies non-maximum suppression. This process removes redundant bounding boxes with high overlap and retains the one with the highest confidence score.

VI. Final Detection:

After non-maximum suppression, the remaining bounding boxes with their associated class scores represent the final detected objects in the image

5.2 Key Advantages of SSD:

- **Efficiency:** SSD is known for its efficiency in terms of both speed and memory usage, making it suitable for real-time object detection.
- **Multi-scale Detection:** By using feature maps from multiple layers of the CNN, SSD can detect objects at different scales within the same image.
- **No Proposal Generation:** Unlike region-based detectors (e.g., Faster R-CNN), SSD directly predicts bounding boxes and class scores, eliminating the need for a separate proposal generation step.
- **End-to-end Training:** SSD can be trained end-to-end, which means the feature extractor and object detection components can be optimized together, resulting in better performance.
- **Versatility:** SSD is versatile and can be adapted for various object detection tasks, including multi-class and single-class detection.

5.3 Code Implementation and Libraries Used:

The architecture was implemented using PyTorch. Various other libraries include torchvision, numpy, and openCV for preprocessing and image manipulation.

6. Model Training

The SSD300 model was trained using the above-mentioned Pascal VOC dataset by using these parameters.

- **Optimizer:** Standard Gradient Descent (SGD) was used as an optimizer for the network by setting the weight_decay, learning_rate, and momentum.
- **Loss function:** Multibox loss function is used for SSD. This is a combination of confidence loss and location loss.
- **Activation function:** Rectified Linear Unit (ReLU)

Once the model is trained, the weights are saved in the “.pth” file. This can then be loaded and used for detection without having to train the model again.

7. Integration:

I'll be using a pre-trained SSD300 model, By loading the weights of this pre-trained model, Detections can be performed and easily be integrated into the application of choice.

- The architecture is built with scalability in mind. Many more object detection models of choice can be integrated into the app easily allowing users to choose different object detection models to perform detections.

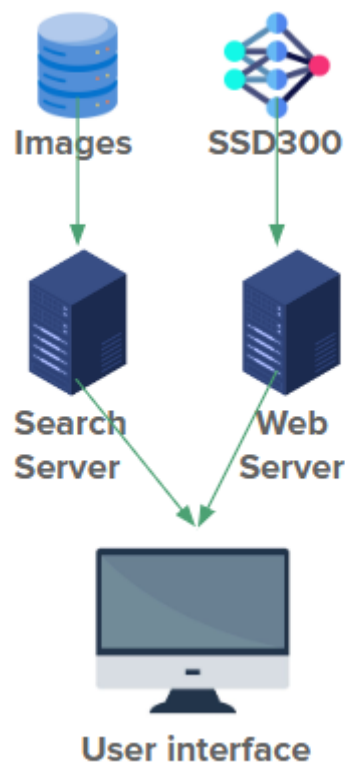
8. Implementation:

A web-based user interface will be designed to allow users to interact with the model.

- This will be implemented using Python's Flask library as the back-end server and Javascript and HTML as the front-end.
- APIs will be used to make calls to the back-end server which will be handling the preprocessing, making detections on the input image, and relaying it back to the front-end.

8.1. System Architecture

The web-based implementation of the application and its architecture are shown in the diagram below



1.4 Model Architecture of the web-app implementation

- The **Images** are the list of all the available images to search using the application. They are accessed by the search server and web server.
- The **SSD (SingleShotDetector)** is the detection model that the web server uses to perform detections on the images.
- The **Search Server** is responsible for handling search requests from the web app.
- The **Web Server** is responsible for handling image detection requests from the users.
- The **User Interface** is where the users will be performing requests and the results will be displayed.

There are separate servers that are set up to handle search requests and object detection requests, this is to prevent any latency while performing detections on searched image results.

8.2. Functionalities of the application:

The app comes with the following functionalities that allow the user to do the following operations.

- **Upload** an image and get prediction results from the model. The results include
 - Detection time.
 - Number of objects detected by the model within the image.
- **Search** images with object names that the model can detect. The results include
 - List of Images containing the objects that were searched for.
 - Ability to perform detections on the searched images and to save them.

9. Evaluation:

9.1. Mean Average Precision (mAP):

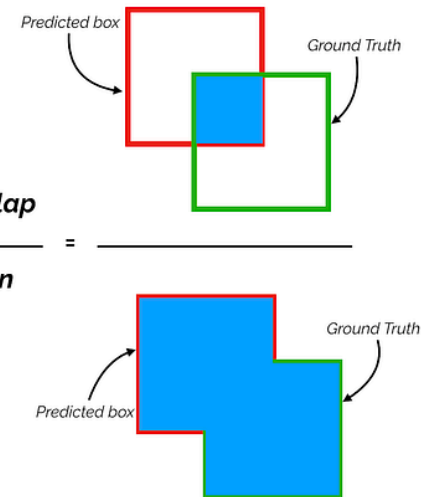
The model is evaluated using specific metrics used for object detection such as mean average precision(mAP). However, it's not the average of the precision. Precision measures how close the predictions are to the ground truth values.

$$\text{Precision} = \text{TP}/(\text{TP}+\text{FP})$$

$$\text{Recall} = \text{TP}/(\text{TP}+\text{FN})$$

mAP is not calculated by taking the average of precision values. But it's calculated by taking the mean AP over all classes and/or overall IoU thresholds, depending on different detection challenges that exist. Object detection systems make predictions in terms of a bounding box and a class label.

9.2. Intersection Over Union (IoU)



$$\text{Intersection over Union (IoU)} = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{\text{Area of Overlap}}{\text{Area of Predicted box} + \text{Area of Ground Truth} - \text{Area of Overlap}}$$

1.4 IoU for calculating mAP

Where,

$$\text{Area of Overlap} = |X_{pmax} - X_{amax}| * |Y_{pmax} - Y_{amax}|$$

$$\text{Area of Union} = (Y_{pmax} * Y_{amax}) + (X_{amax} * X_{pmax}) - \text{Area of overlap}$$

Y_{pmax} and Y_{amax} are Y predicted bounding box coordinates and Y actual bounding box coordinates respectively.

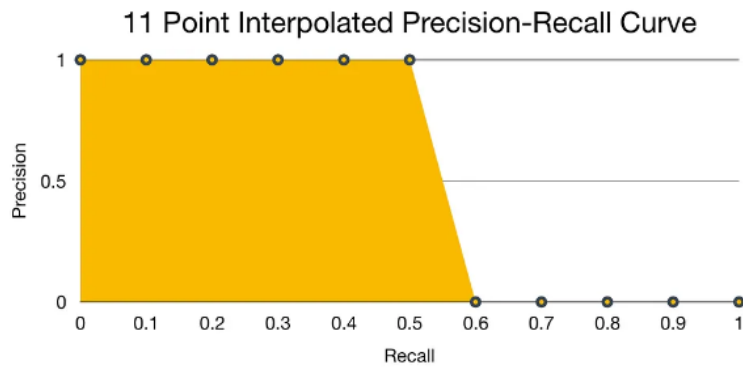
X_{pmax} and X_{amax} are X predicted bounding box coordinates and X actual bounding box coordinates respectively.

For each bounding box, we measure an overlap between the predicted bounding box and the ground truth bounding box. This is measured by IoU (intersection over union). For object detection tasks, we calculate Precision and Recall using the IoU value for a given IoU threshold.

For example, if the IoU threshold is 0.5, and the IoU value for a prediction is 0.7, then we classify the prediction as True Positive (TP). On the other hand, if IoU is 0.3, we classify it as False Positive (FP).

Once we find precision and recall, we can plot an 11-point interpolated curve and find the area under it to calculate the mAP.

9.3. 11-Point Interpolated Curve:



1.5 A sample precision-recall curve.

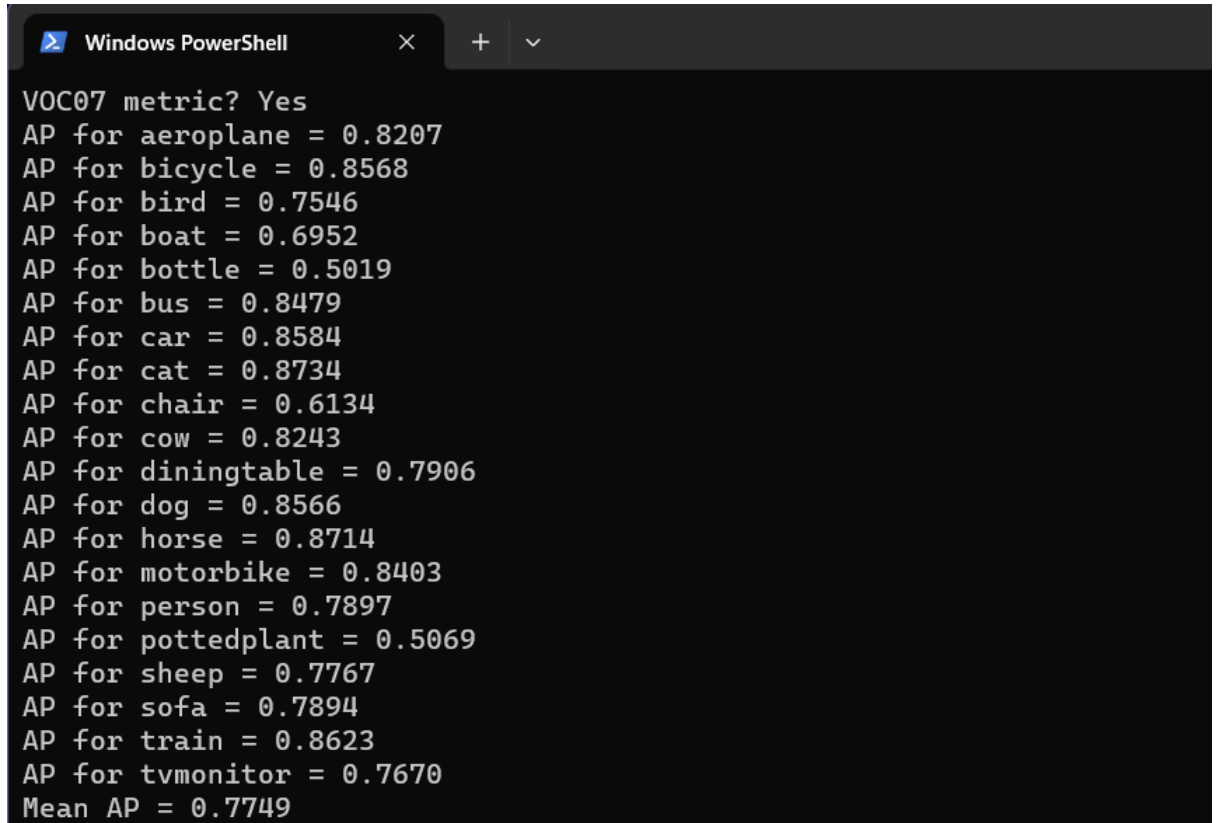
$$AP = \frac{1}{11} \sum_{Recall_i} Precision(Recall_i) = 1$$

1.6 The equation to calculate the mAP

Using the equation above we calculate the Average precision (AP) for each class by taking the area under the curve. Then we take a mean of the Average precisions of all the classes to obtain the mean average precision (mAP).

10. Test results:

10.1. Tests performed on specific device specifications:



```
Windows PowerShell
VOC07 metric? Yes
AP for aeroplane = 0.8207
AP for bicycle = 0.8568
AP for bird = 0.7546
AP for boat = 0.6952
AP for bottle = 0.5019
AP for bus = 0.8479
AP for car = 0.8584
AP for cat = 0.8734
AP for chair = 0.6134
AP for cow = 0.8243
AP for diningtable = 0.7906
AP for dog = 0.8566
AP for horse = 0.8714
AP for motorbike = 0.8403
AP for person = 0.7897
AP for pottedplant = 0.5069
AP for sheep = 0.7767
AP for sofa = 0.7894
AP for train = 0.8623
AP for tvmonitor = 0.7670
Mean AP = 0.7749
```

1.7 mAP taken by averaging mean over 20 object classes

Here we can see the test results when the detections were performed on the **VOC2007** test set using **SSD300** with **VGG16** as a backbone network with an **IoU** threshold of **0.5**. We obtain a **mAP** of **~77.5%**.

The device specifications on which the tests were performed:

- **Processor:** 12th Gen Intel(R) Core(TM) i7-1260P 2.10 GHz
- **RAM:** 8.00 GB
- **System type:** 64-bit operating system, x64-based processor

10.2. Results from the cited paper.

The results below are calculated with different models that use VGG16 as a common backbone network with an IoU threshold of 0.5 on the Pascal VOC 2007 dataset. This was performed on Titan X GPU

System	VOC2007 test <i>mAP@0.5</i>	FPS (Titan X)	Number of Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	~6000	~1000 x 600
YOLO	63.4	45	98	448 x 448
SSD300 (VGG16)	77.2	46	8732	300 x 300
SSD512 (VGG16)	79.8	19	24564	512 x 512

2.1 The table showcases the results of testing different models on the **VOC2007** dataset.

The table 2.1 contains the following:

- The **model** used to perform the tests.
- The **mAP** (mean average precision) at an IoU (Intersection over union) threshold of 0.5.
- The **FPS** (frames per second) determines how fast the model can detect objects which can be a critical metric while using them to do live detections.
- The **number of boxes** denotes how many boxes the model generates and assigns confidence on whether an object is present in that location or not.
- The **Input resolution** is the size of the image fed into the CNN (convolution neural network).

10.3. Inferences

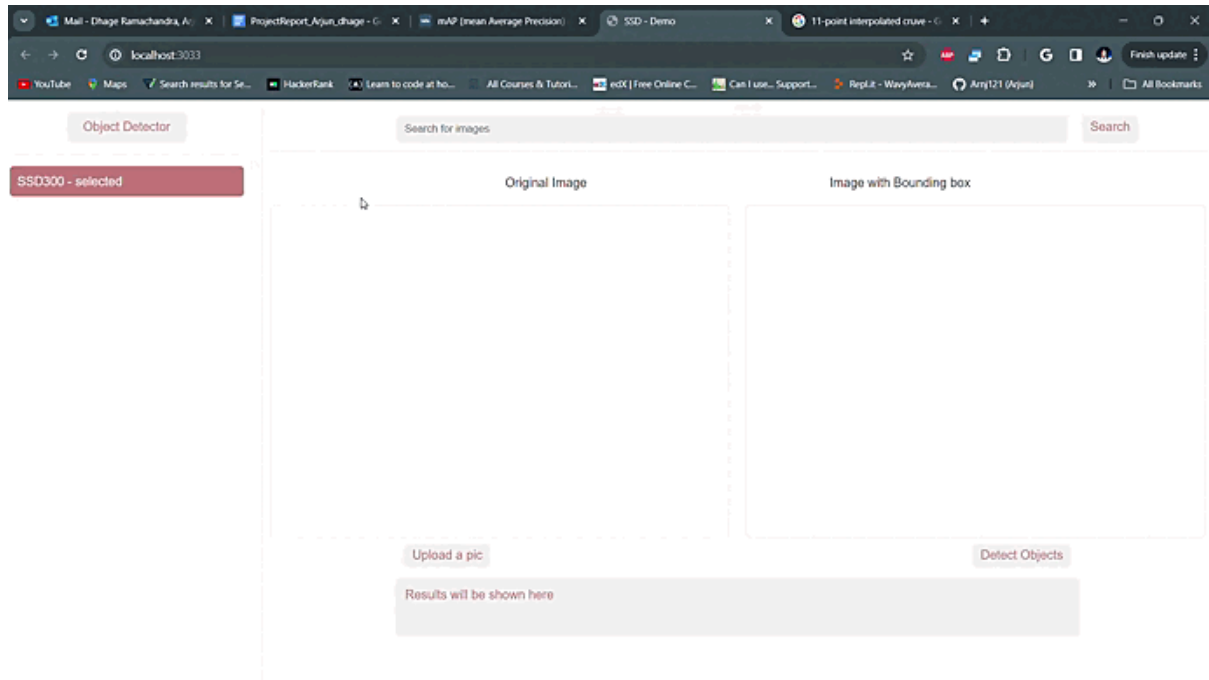
We can see that **SSD300** performs better than other models (**mAP 77.2**) at a lower resolution (**300x300**) and performs detections at a higher frame rate (**46 FPS**).

However, **SSD512** has a higher mAP (**79.8**) but outputs a lower frame rate (**19 FPS**) at a higher resolution (**512x512**).

In conclusion, **SSD300** achieves results comparable to **SSD512** and outputs a higher frame rate with lower resolution.

*NOTE: The model performance (**FPS**) can vary depending on the hardware that is used to make the predictions. For example, **NVIDIA's RTX 6000 ADA GPU** might perform significantly faster and produce higher frame rates than **Titan X GPU**.*

11. Output:



1.8 A web app implementation of an object detection model using SSD300.

In the above demonstration, we can see how the app functions.

- First, an image is uploaded and a detection is performed on the uploaded image.
- Secondly, An object to search for, which is a **car** is typed into the search box, which displays images of cars.
- Furthermore, An image is selected from the result which shows a **detect** button to perform detections on the searched image, hence validating that the image displayed does contain the object searched for.

Link to the demo video: [SSD-Demo](#)

12. Applications: Where is it useful?

Object detection has versatile applications in fields such as

- **Security Surveillance:** By detecting and tracking objects in real-time, we can make inferences about the environment.
- **E-commerce:** Improve product search functionality by allowing users to find items similar to what they upload.
- **Healthcare:** Aid medical imaging by identifying and analyzing anomalies in medical scans.
- **Smart Agriculture:** Monitor crop health and detect pests or diseases using aerial imagery.

- **Automated Inventory Management:** Streamline inventory processes by automating the tracking and counting of items.
- **Artificial Intelligence in Photography:** Assist photographers in tagging and organizing images based on detected objects.
- **Social Media Content Moderation:** Automatically filter and moderate content by identifying inappropriate objects.
- **Autonomous Vehicles:** Contribute to the perception system of self-driving cars by detecting obstacles and pedestrians.
- **Augmented Reality:** Enhance AR applications by recognizing and interacting with real-world objects.

The application that the model is being used for depends on the nature of the data the model is trained on. In this project, The model was trained to identify 20 common objects using the Pascal VOC dataset. For the model to be used in healthcare, it must be trained on the dataset that we wish for the model to detect.

The search feature, which is the project's main feature, searches for similar images based on keywords among 9000+ images by detecting and tagging each image with the objects present and comparing them with the image or keyword being searched, which finds application in photography, search engines, also help visually challenged people describe the image.

12.1. Training Object detection models for use in any application.

The versatility of these object detection models to be trained on any dataset and have accurate predictions is one of the prime characteristics that allow them to be used in any field.

We can train the model to detect breast cancer by training it on a dataset containing images of breast cancer. We can also train the model to tag and organize similar images or search for similar images.

Object detection models output data in the form of bounding box coordinates and the objects detected in the image, this data can be used in the following ways

- To count the number of objects using objects detected.
- To verify if any 2 objects are closer or far away in proximity using bounding box coordinates.
- To check the position of an object in an image for specific use cases.

And much more.

As discussed above, the data can be used depending on the use case for an application and the model can be trained on any dataset given there are annotations for objects in each image.

13. Future Work:

This Project has proved to be a great learning experience, from learning about the basics of computer vision to implementing powerful libraries such as OpenCV, and PyTorch, and evaluating pre-trained models. The final step of the project was developing a web app where users could upload images to generate predictions and search images based on the presence of objects being searched.

Additional features:

- Currently, only SSD is implemented but I plan to integrate multiple models which showcase how SSD fares against them.
- Also, the ability to perform detections by uploading video files and searching videos for the presence of objects being searched is being worked on as well.

14. References:

- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016, December 29). SSD: Single shot multibox detector. arXiv.org. <https://arxiv.org/abs/1512.02325>
- Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., & Murphy, K. (2017, April 25). Speed/accuracy trade-offs for modern convolutional object detectors. arXiv.org. <https://arxiv.org/abs/1611.10012>
- Amdegroot. (n.d.). Amdegroot/ssd.pytorch: A pytorch implementation of single shot Multibox Detector. GitHub. <https://github.com/amdegroot/ssd.pytorch/tree/master>
- VGG - Torchvision main documentation. (n.d.). <https://pytorch.org/vision/main/models/vgg.html>
- Yohanandan, S. (2020, June 9). Map (mean average precision) might confuse you! Medium. <https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2>

Appendix

Source code link: [Source code](#)

Link to the demo video: [SSD-Demo](#)