**A09 ITAI 1378 CV – Object Detection**  
​  
Martin Demel, Brandie Griffin, Jordan Allen, Sydney Chilson, Marvin Azuogu.​  
​  
Department of Science, Technology, Engineering & Math, Houston Community College​  
​  
ITAI-1378: Computer Vision – Artificial Intelligence   
​  
Patricia McManus​  
​  
October 24th, 2024.​

**Object detection cheat sheet**

**basics of object detection**

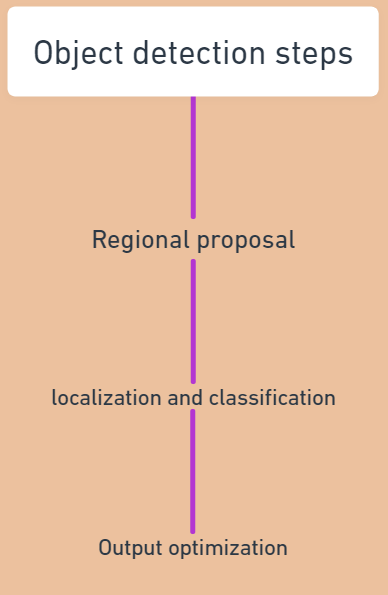
**Object detection** is the ability to identify an object of interest within an image. Object detection works by utilizing **regional proposal, localization and classification, and output optimization**. In addition, object detection consists of two different types of models, a **Two-Stage detection model and a One-Stage detection model**.

The basics steps to object detection are listed in order below:

**Regional proposal** is the first step in object detection. It is also known as regions of interest (ROI). The purpose of ROI is to locate areas with a high likelihood of a object being presence.

**Localization and classification** are the next step in object detection. In this step, a bounding box is generated, containing the x y location and height and width of the object. Furthermore, in this step, the object class is determined using a pre-trained classification neural network.

**Output optimization** is the final step in object detection. In this step the output is optimized by checking for overlapping bounding boxes and determining whether there are 2 objects in that area or not. Then if its determents that there is one object with in multiple bounding boxes the smaller of the bounding boxes are removed using a method called Non-Maximum Suppression (NMS



As previously mentioned, Object detection consists of two main types of models that differ in specific ways.

Firstly, there's **Two-Stage detection models**.

It consists of more than one model, a regional proposal network (RPN), and a classifier model.

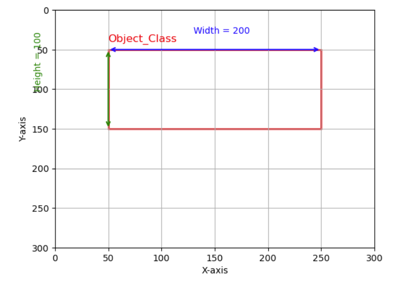
Finally, there's **One-Stage detection models**

It consists of only one model where the regional proposal network and classifier model are combined into one model. Thus, regions are classified in a single step.

The following are different components and concepts related to object detection:

**Bounding Boxes:**

bounding box, is a box that visually displays the location of an object within an image. (Add pictures ex: )

The different components of a Bounding boxes, are

Coordinates: show the starting point of the bounding box.

Dimensions: show the size of the bounding box

Class labels: show the label of the object in the bounding box.

confidence scores

The confidence scores measure convincingly the presence of an object inside the bounding box, indicating a typical value between 0 and 1. A score closer to 1 detects the predicted class object while a lower score means less certainty.

Annotations

Annotations are “Bounding Boxes” fundamental in detecting objects for model training tasks. In this process, objects detection is derived by drawing rectangular shapes (which are the bounding boxes) on all sides of an image for easy classifications.

Intersection over Union

Intersection over Union (IoU) refers to a system of measurement to assess the accuracy of model that detects object, precisely comparing between predicted bounding boxes (where the model thinks the object is) with ground box (the actual location of the box).

IoU formula is:

IoU = Area of Overlap

Aree of Union

**common object detection algorithms**

**R-CNN (Region Based Convolutional Neural Network):**

R-CNN tries to find objects in an image by first guessing where objects might be, using a method called selective search. Think of it like circling areas in a photo that could contain objects. After making these guesses, it uses a neural network to study each area closely. Then, it uses another program to decide what each area contains (whether it’s a dog, car, or tree). The downside is that R-CNN is slow because it has to carefully look at every possible object area one at a time.

**Fast R-CNN:**

Fast R-CNN improves R-CNN by making the process faster and more efficient. Instead of analyzing each potential object area separately, it first looks at the whole picture using a neural network and creates a map of the important features in the image. Then, it uses this map to quickly zoom in on the areas where objects might be. This way, it can make decisions much faster while still being accurate, saving a lot of time compared to R-CNN.

**Faster R-CNN:**

Faster R-CNN makes the process even quicker by using a special tool called a Region Proposal Network (RPN). Instead of making guesses about object locations separately, the RPN automatically suggests areas to look at, based on what the neural network has already seen. This tool is built right into the main network, which means it can quickly figure out where to look for objects and identify them without a lot of extra steps. It’s a powerful option if you need both speed and accuracy.

**SSD (Single Shot Multibox Detector):**

SSD takes a different approach by looking at the entire image in just one pass, instead of breaking it down into areas first. It uses different parts of the neural network to find objects of various sizes all at once. This makes it really fast because it doesn’t have to keep going back and forth between regions like R-CNN. It’s great for tasks where you need quick results, like in video processing or self-driving cars, but it can sometimes miss smaller objects.

**YOLO (You Only Look Once):**

YOLO is all about speed. It treats finding objects like solving a puzzle in one go. It divides the image into a grid and then checks each part of the grid to see what objects are there, figuring out their location and type at the same time. This allows it to analyze an entire image quickly, making it perfect for real-time tasks like detecting people or cars on a live camera feed. The earlier versions of YOLO weren’t great at spotting small or very detailed objects, but they have improved over time.

**tools and libraries used in object detection tasks**

The following paragraph discusses three key tools used in machine learning and computer vision TensorFlow, Keras, and OpenCV. These tools help create models and do tasks like object detection and image processing. TensorFlow and Keras are mostly used for building and training deep learning models, while OpenCV is great for working with images and videos in real-time. Using these tools together helps make machine learning projects much more manageable and efficient.

TensorFlow is a well-known tool for building machine learning models, and it works great for object detection. It has pre-built models, like the ones in the TensorFlow Object Detection API, which makes it easy to find things like cars, people, or animals in pictures. Installing TensorFlow requires using commands like pip install tensorflow, along with other packages depending on whether a GPU or CPU is being used. After loading a pre-trained model, the code can be used to detect objects in images. Alternatively, TensorFlow can be run on Google Colab, a free online platform that already has TensorFlow installed. Colab is especially helpful for larger tasks as it offers access to GPUs or TPUs, and the work can be easily saved and shared via GitHub for collaboration.

Keras is a tool that works with TensorFlow, but it’s even easier to use. It lets you build models quickly without having to write a lot of complicated code. You can use Keras to build neural networks that detect objects in images, or, if you don’t want to create a model from scratch, you can load pre-built models. Keras is included with TensorFlow, but if you need to install it separately, you can use the command pip install keras. The examples in Keras are usually short, with less than 300 lines of code, and are focused on showing how deep learning works. If you want to install or upgrade Keras, just use the command pip install --upgrade keras. All Keras examples are written as Jupyter notebooks, which can be run in Google Colab, an online environment that doesn’t need any setup. Colab also has GPU and TPU options to help make things run faster.

OpenCV is the largest computer vision library in the world and is widely used for tasks like image processing, video analysis, and object detection. While it doesn’t train deep learning models on its own, OpenCV can use pre-trained models, such as Haar Cascades, to detect objects like faces and eyes in images or video streams. The Haar Cascade method works by analyzing features in an image, such as the darker areas around the eyes compared to the nose. You can start using OpenCV by installing it with pip3 install opencv-python, and then load pre-trained models using cv::CascadeClassifier::load to detect objects. In a live video stream, like one from a webcam, the code can identify faces and eyes and draw rectangles around them. It’s a quick and efficient method that works well for real-time detection

In summary, TensorFlow, Keras, and OpenCV are valuable tools for handling object detection and machine learning tasks. TensorFlow offers pre-built models for identifying objects in images and can be used with Google Colab for larger tasks. Keras simplifies the process of building neural networks with minimal code and works well with TensorFlow. OpenCV is great at processing images and videos, and it uses models like Haar Cascades to find things like faces and eyes. When you use tools like TensorFlow, Keras, and OpenCV together, they make machine learning projects easier to handle and more efficient.

**common challenges and troubleshooting tips in object detection.**

Even though we have achieved significant advancements in object detection, challenges still appear during the implementation phase. These challenges can affect the efficiency and accuracy of the detection process, which is crucial for object detection.

**Class Imbalance**

It is one of the most common challenges in object detection. When some object classes are more frequent than others, the model can become biased towards predicting these dominant classes. This ultimately leads to poor performance on less frequent object classes.

To address this class imbalance, we must apply class weight adjustments during the training and augmentation for minority object classes, or we can perform oversampling or under-sampling that can help ensure more accurate detection.

**Small Object Detection**

Detecting small objects can be very challenging, especially when the small object is present only on several pixels. This ultimately leads to missing the object or misclassifying it.

To properly detect small objects, even if they are present only on several pixels, we must increase the input image resolution so that the small objects are present on more pixels and are easier to identify. FPN – Feature Pyramid Network can be another fix to address object detection by detecting objects at different scales.

**Bouncing Box Accuracy**

Poor bouncing box accuracy can occur when the model cannot fit the objects tightly within the bouncing boxes. It can also happen when the bouncing boxes are misaligned or not centered properly around the objects.

Anchor box tuning has proven to be an effective way to address bouncing box accuracy by adjusting the anchor sizes and aspect ratios better to match the shape and size of the objects.

**Low Confidence Scores**

When the model is uncertain about the presence of an object within a bounding box, a low confidence score is given. This often leads to incorrect classification or missing detection. Confidence scores represent the model's certainty about the accuracy of its prediction. In complex environments, this can significantly affect the performance of the object detection model.

By adjusting the confidence threshold, we can capture more objects, but we must ensure that it’s balanced and that we don’t have increased false positives. Furthermore, we can retrain the model with hard negative mining, improving the model’s robustness in determining between the objects and background. Finally, the anchor box tuning, which is important in our assignment, can lead to significant improvements. We can achieve higher confidence scores and more accurate bounding box predictions by tuning it.

Object detection is a very powerful yet complex area with some of the challenges we have included in our assignment. When addressed correctly, we can significantly improve the accuracy and reliability of the model, which leads to efficient object detection.

**Additional Resources**:

Bali, R. (2024, February 6). *Object detection basics - a comprehensive beginner’s guide (part 1)*. Medium. <https://towardsdatascience.com/object-detection-basics-a-comprehensive-beginners-guide-part-1-f57380c89b78>

Park, S. (2021, July 29). *A guide to two-stage object detection: R-CNN, FPN, mask R-CNN and more*. Medium. <https://medium.com/codex/a-guide-to-two-stage-object-detection-r-cnn-fpn-mask-r-cnn-and-more-54c2e168438c>

Bessi, L. (2023, July 16). *#29 object detection: One stage vs two stage networks and evaluation.* #29 Object detection: one stage vs two stage networks and evaluation. <https://machinelearningatscale.substack.com/p/computer-vision>

"Cascade Classifier." *OpenCV Documentation*, OpenCV, <https://docs.opencv.org/4.10.0/db/d28/tutorial_cascade_classifier.html>. Accessed 22 Oct. 2024

"TF2 Object Detection." *TensorFlow Hub*, Google, <https://www.tensorflow.org/hub/tutorials/tf2_object_detection>. Accessed 22 Oct. 2024.

"Getting Started." *Keras Documentation*, <https://keras.io/getting_started/>. Accessed 22

Oct. 2024.

[Confidence Score - an overview | ScienceDirect Topics](https://www.sciencedirect.com/topics/computer-science/confidence-score#:~:text=A%20confidence%20score%20is%20calculated,(Intersection%20over%20Union)%20thresholds.)

*Papers with Code - R-CNN Explained*. (n.d.). <https://paperswithcode.com/method/r-cnn>

Weng, L. (2017, December 31). Object Detection for Dummies Part 3: R-CNN Family. *Lil’Log*. <https://lilianweng.github.io/posts/2017-12-31-object-recognition-part-3/>

Li, X., J., Wang, S., Zhao, Y., Zhao, C., & University of California, San Diego. (n.d.). *Object Detection using Faster R-CNN* [Report]. <https://noiselab.ucsd.edu/ECE228_2019/Reports/Report3.pdf>

GeeksforGeeks. (2024, January 2). *Handling imbalanced data for classification*. GeeksforGeeks. <https://www.geeksforgeeks.org/handling-imbalanced-data-for-classification/>

GeeksforGeeks. (2024, July 25). *Evaluating Object Detection Models: Methods and Metrics*. GeeksforGeeks. <https://www.geeksforgeeks.org/evaluating-object-detection-models-methods-and-metrics/>

Mandal, S., Mones, S. M. B., Das, A., Balas, V. E., Shaw, R. N., & Ghosh, A. (2021). Single shot detection for detecting real-time flying objects for unmanned aerial vehicle. In *Elsevier eBooks* (pp. 37–53). <https://doi.org/10.1016/b978-0-323-85498-6.00005-8>

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot Multibox Detector. In *European Conference on Computer Vision (ECCV)* (pp. 21-37). Springer

Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2), 303-338.

**Reflection**

Working as a team on this cheat sheet assignment for object detection helped us explore the critical concepts, gain a deeper understanding of the methodologies used for object detection, and identify challenges and workarounds that might improve the model’s capabilities.

We understood how detection models like R-CNN, Fast R-CNN, SSD, and YOLO operate and their strengths and weaknesses in terms of speed, accuracy, and real-time detection capabilities. Furthermore, we deep-dived into the importance of the bounding boxes, intersection over Union, and confidence scores, which are key metrics to ensure the precise detection and classification of the objects within the image dataset.

One of the key points during our team analysis was the recognition of class imbalance, which can significantly impact the model performance if it is not properly addressed during the testing. Techniques such as over-sampling or under-sampling and adjusting the weights can improve the model’s ability to detect less detectable classes.

By creating this cheat sheet, we reinforced the concepts and methodologies and gained insight into common troubleshooting techniques in real-life scenarios. We believe this serves as a valuable quick reference in our future tasks and will help us apply these object detection techniques effectively in both academic and work-practical real-life scenarios.