Object Detection

Group 7

Alisha D. Rush, Muhammad Abdullah Farooq, Saima Sano, Zaid Tahir

Patricia Mcmanas

ITAI-1370

## **Cheat Sheet**

# ***Challenges and troubleshooting tips***

|  |  |  |  |
| --- | --- | --- | --- |
| Challenges and Troubleshooting | Problem | | Solution |
| Class Imbalance | One class dominates others in the dataset. | Use weighted loss or augment underrepresented classes |
| Small Object Detection | Small objects are harder to detect. | Increase resolution or fine – tune anchor boxes |
| Overfitting | Model performs well on training data but poorly on new data. | Solution is overlifting by applying data augmentation and dropout techniques. |
| Low IoU | Poor overlap between predicted and true bounding boxes. | Adjust an anchor box size or apply non-max suppression |
| Slow inference time | Model is too slow | Use faster models like YOLO or SSD, prune the network or apply model quantization |
| Blurry/Noisy Images | Image quality effects detection accuracy. | Apply noise reduction or image enhancement techniques. |

# ***Formulae:***

# ***C:\Users\LENOVA\Downloads\WhatsApp Image 2024-10-24 at 11.48.13 PM.jpeg***



# ***Tools and Libraries for Object Detection:***

* ***TensorFlow:***

TensorFlow is an open source machine learning library by Google, which offers high-level APIs for the building and training of deep learning models which includes object detection. The Tensorflow Object Detection API simplifies the process of training custom models.

***Keras:***

The Keras library is built on top of TensorFlow. It is a friendly deep learning API, ready for rapid prototyping, and building neural networks, including those models handling object detection, with very intuitive code.

***OpenCV (Open Source Computer Vision Library):***

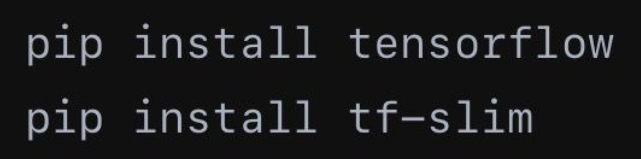
OpenCV is an open-source library for computer vision. It is quite a popular library offering software tools for processing real-time images and video streams, which are often used in object detection pipelines for the task of image augmentation and pre/post-processing by drawing bounding boxes.

These libraries and tools form the backbone when building and deploying object detection models. These are utilized to train a model, while OpenCV is the preferred choice for deployment and post-processing.

# ***Tools and Libraries Overview:***

1. ***Tensorflow***

* **Installation :**



* **Usage :**



* **Documentation** :

*TensorFlow Object Detection API*

1. ***YOLO (DarkNet)***

* **Installation :**

git clone <https://github.com/AlexeyAB/darknet.gitmake>

* **Usage :**

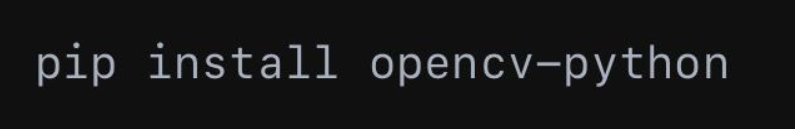
./darknet detect cfg/yolov3.cfg yolov3.weights data/dog.jpg

* **Documentation:**

./darknet defect cfg/yolov3.cfg yolov3.weights data/dog.jpg

1. ***OpenCV***

* **Installation :**

****

* **Usage :   
    
  **
* **Documentation :**

OpenCV Documentation

# ***Resources:***

* **Books :**

Deep Learning for computer vision by Adrian Rosebrock

Hands On Machine Learning with Scikit-Learn, Keras, and TensorFlow by Aurelien Geron

* **Online Tutiorials :**

Coursera : Introduction to computer vision by Andrew Ng

Udacity : Computer Vision Nanodegree

* **Websites:**

PyImageSearch

AI Shack Object Detection Tutorials

# ***Reflection:***

By working on this object detection cheat sheet, we have significantly gained the understanding of the complex topic by breaking it down into key concepts. Key elements like **bounding boxes**, **Intersection over Union (IoU)**, and various models such as **YOLO**, **SSD**, and **Faster R-CNN** were particularly interesting to explore.

Through this assignment, we have gotten a clear view of the workflow and challenges involved in object detection, which has practical implications for using these models effectively. The research into bounding boxes, IoU, and confidence scores emphasized the importance of precision in accurately identifying objects within images. Additionally, examining algorithms like R-CNN, SSD, and YOLO revealed the trade-offs between detection speed and accuracy, showcasing how each algorithm is optimized to tackle specific challenges in the field.

Creating this cheat sheet has provided us with a quick reference guide for future tasks, enabling us to independently set up and conduct experiments with object detection libraries like **TensorFlow**, **Keras**, and **OpenCV**. This newfound confidence in using these tools will be invaluable for upcoming projects in computer vision.

Moreover, I learned to identify common issues and practical debugging strategies, especially concerning challenges like class imbalance and small object detection. This cheat sheet has become an essential resource for our object detection projects, merging crucial information into a concise format that serves as a fast reference for renewing our knowledge and effectively solving problems.