Cheetah Detection using YOLOv12

Jayson du Toit — u22571532

October 27, 2025

1 Theoretical Background and Evolution of YOLO

Object detection is a fundamental computer vision task, involving the finding and classification of objects in an image. Approaches like R-CNN use a two-stage process to do this. This is accurate, but it is computationally expensive and not suitable for real-time applications. You Only Look Once (YOLO), however, takes a different approach, framing the problem as a single regression problem. YOLO directly predicts bounding boxes and class possibilities from a full image in a single evaluation. The one-stage approach makes YOLO much more computationally efficient and ideal for real-time applications.

The evolution of YOLO versions:

- **YOLOv1 (2015):** Core concept introduced of dividing the image into a grid and letting each cell predict bounding boxes. Struggled with small and overlapping objects.
- **YOLOv2 (2016):** Introduced anchor boxes, batch normalization, multi-scale training, improving accuracy and speed.
- **YOLOv3 (2018):** Introduced Feature Pyramid Network and Darknet-53 increasing perfromance and detetction of small objects.
- **YOLOv4 (2020):** Architecture optimization was done by. introducing Bag of Freebies, Bag of Specials and mish activation.
- **YOLOv5 (2020):** Introduction of PyTorch made training easier and accessible.
- **YOLOv6 (2022):** Implementation of a Bi-directional Concatenation (BiC) module, an anchor-aided training (AAT) strategy, and an improved backbone and neck design for state-of-the-art accuracy on the COCO dataset.
- **YOLOv7 (2022):** Refined architecture and parameterized convolutions.
- **YOLOv8 (2023):** Introduced an anchor-free design (predicting object centers directly) and an advanced backbone and neck architecture.
- **YOLOv9 (2024):**Introduced Programmable Gradient Information (PGI) and Gelan architecture for better representation learning.
- **YOLOv10 (2024):** Eliminates non-maximum suppression (NMS) and optimizes various model components to reduce overhead and improve real-time efficiency.

- **YOLOv11 (2024):** Model refinement and optimization.
- **YOLOv12 (2025):** This version is "Attention-Centric," integrating attention mechanisms into the architecture to help the model focus on the most relevant features, improving performance while maintaining efficiency.

2 Practical Weaknesses of YOLO

Despite its speed and accuracy, the YOLO family has several practical weaknesses:

- **Small Object Detection:** While this has greatly improved from YOLOv3, it detects very small objects in the distance. These objects consist of very few pixels for the model to extract features from.
- **Overlapping Objects:** YOLO can struggle to distinguish multiple overlapping objects. It may only detect one object in a crowd and struggle with classifying boundaries.
- **Localization Inaccuracy:** Compared to two-stage detectors, YOLO can sometimes place bounding boxes less precisely, leaving wiggle room.
- **Novel Orientations:** When a model is trained on a standard orientation/pose, it may struggle to identify unusual orientations/poses.

3 Model Design and Approach

3.1 Data Annotation and Preparation

`Labelimg` was used to manually annotate the datasets `cheetah_training`, `cheetah_train_validation`

- **Format:** Annotations were saved in the YOLO `.txt` format.
- **Classes:** `cheetah` was defined as the only class.

3.2 Model Selection and Training

- **Model:** We selected the **YOLOv12-Small** (`yolov12s.pt`) model. This model was chosen for its training speed to make hyperparameter tuning possible.
- **Environment:** The model was trained in Python using the `ultralytics` framework and PyTorch, accelerated by an **NVIDIA GeForce RTX 2070 SUPER** GPU with CUDA.
- **Hyperparameters:** To improve performance, an advanced training run was configured with significant data augmentation and a longer training schedule:

Table 1: Tuning Process Settings

Setting	Value
Model	`yolov12s.pt`
Iterations	`30`
Epochs per Iteration	`15`
Optimizer	`AdamW`

Table 2: Search Space

Hyperparameter	Search Range
lr0	`(1e-4, 1e-1)`
weight_decay	`(0.0, 0.001)`
hsv_s	`(0.0, 0.9)`
hsv_v	`(0.0, 0.9)`

- **Final Model:** From the hyperparameter tuning, the final model was trained on these parameters:

Table 3: Final Model Parameters

Parameter	Value
Base Model	`yolov12s.pt`
Epochs	`100`
Batch Size	`16`
Image Size	`640`
Optimizer	`Auto`
Device	`0 (GPU)`
lr0 (Learning Rate)	`0.0087`
weight_decay	`0.00042`
hsv_s (Saturation)	`0.7263`
hsv_v (Brightness)	`0.44274`

4 Results of Model Performance

The model was trained for 100 epochs, and its performance was evaluated on the validation dataset. The final model weights (`best.pt`) were saved from the epoch with the highest mAP score.

Table 4: Key Performance Metrics

Metric	Value
mAP50-95 (mean AP)	0.75669
mAP50 (AP at IoU=0.5)	0.995
Precision	0.99209
Recall	1

The training graphs in Figure 1 show the model's learning progress, with losses decreasing and mAP scores increasing over time. The confusion matrix in Figure 2 visualizes the model's performance on the validation set, showing the 100% recall of the model on the validation set.

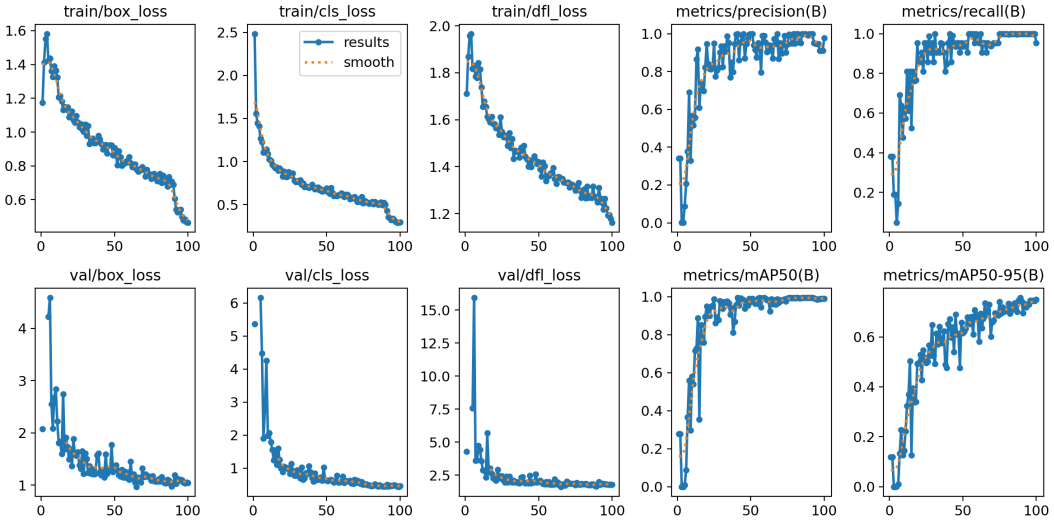

Figure 1: Training metrics graph (mAP, precision, recall, and loss).

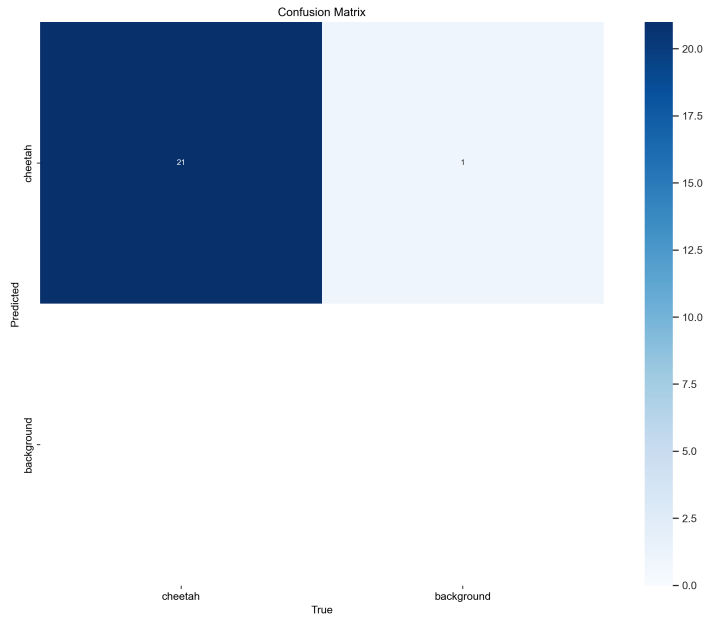

Figure 2: Confusion matrix for the 'cheetah' class.

5 Critical Analysis and Conclusion

5.1 Analysis

The final YOLOv12m model performs well, achieving a final mAP50-95 of **0.75669**, indicating that the boxes the model draws are quite tight and close to the annotated bounding boxes. It successfully identifies cheetahs in the test images and places appropriate bounding boxes around them. The mAP50 of **0.995** is a more forgiving score, but it indicates the precision of overall box placement. The very high presision and recall further show that the model is quite accurate. Due to the lack of other animal classes in the training dataset, the trained model identifies some of the tigers in the test set as cheetahs, with less confidence, however. This is to be expected as the model is trained to identify cheetahs, but not distinguish them from other objects that share similar traits. Further, the YOLO weakness of small objects, as well as the lack of far-away cheetahs in the training set, causes the model to struggle to identify cheetahs when they are situated farther away (smaller) in the image.

5.2 Conclusion

From this project, we see that YOLOv12 is a very easy-to-use and powerful object detector. It was able to accurately identify cheetahs in the test images as well as find them in videos. It is shown how important the training data set is for identification models, as it can only learn from the data it is provided. Further, it was identified that the small object weakness still persists if it is not specifically tackled by the training dataset. Doing hyper paremeter tuning on just a few parameters did help

increase the models verall performance and allowed us to achive very high metric scores.