Logo Detector – Model Report

Shachar Dror

October 25th, 2025

Objective

“

Build a system that detects and localizes logos within video content.

This challenge focuses on identifying specific branded elements, enabling applications such as advertising analytics, sponsorship tracking, and content moderation

”

Methodology

Exploratory data analysis

Train data: 937 images labeled with class and bounding box (near-YOLO format)

Test data: 137 images labeled with class and bounding box (near-YOLO format)

Images resolution varies mostly between SD to Full HD, with an aspect ratio mostly around 4:3 and 5:4. (fig 1)

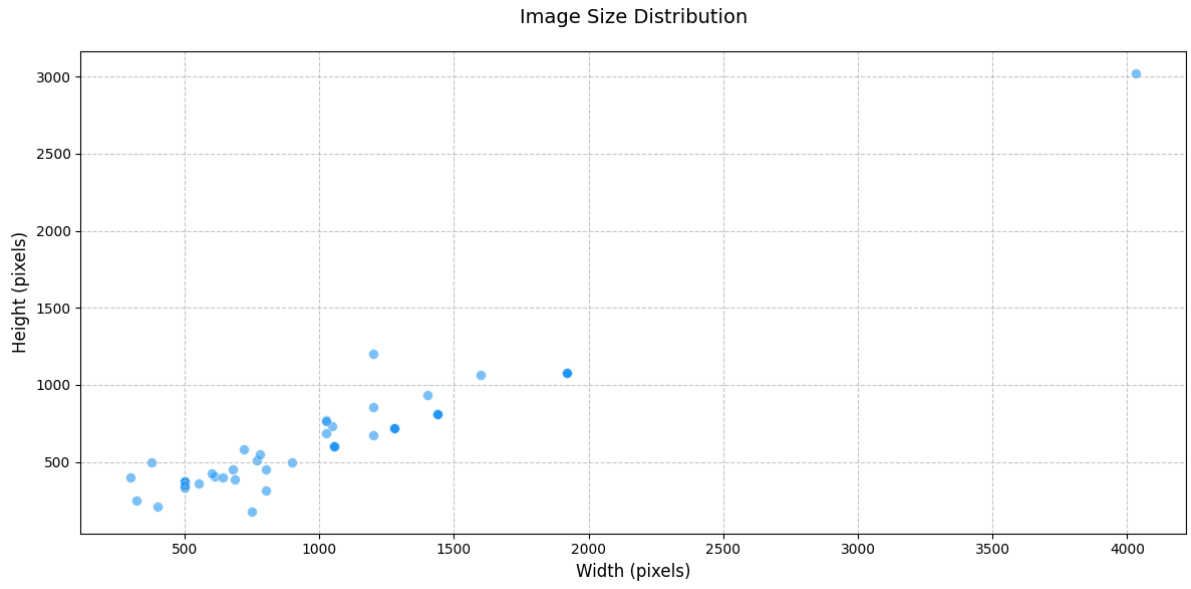

Figure 1

Object's boxes are mostly small (<5% of image size), with few outliers and light trending toward landscape orientation. (fig. 2)

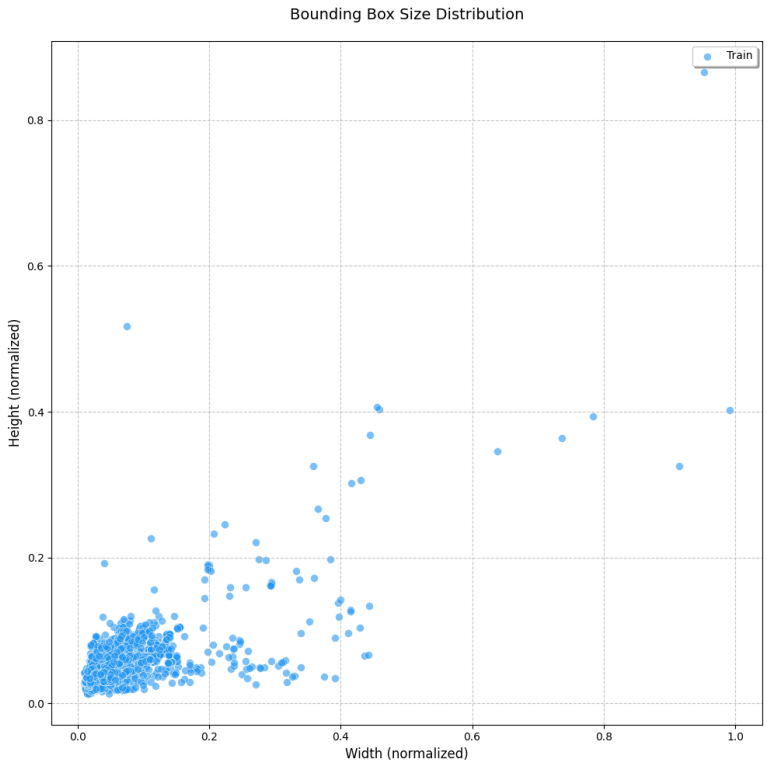

Figure 2

Average number of boxes per image is slightly above 2, with very little images having more than 5 instances. (fig. 3)

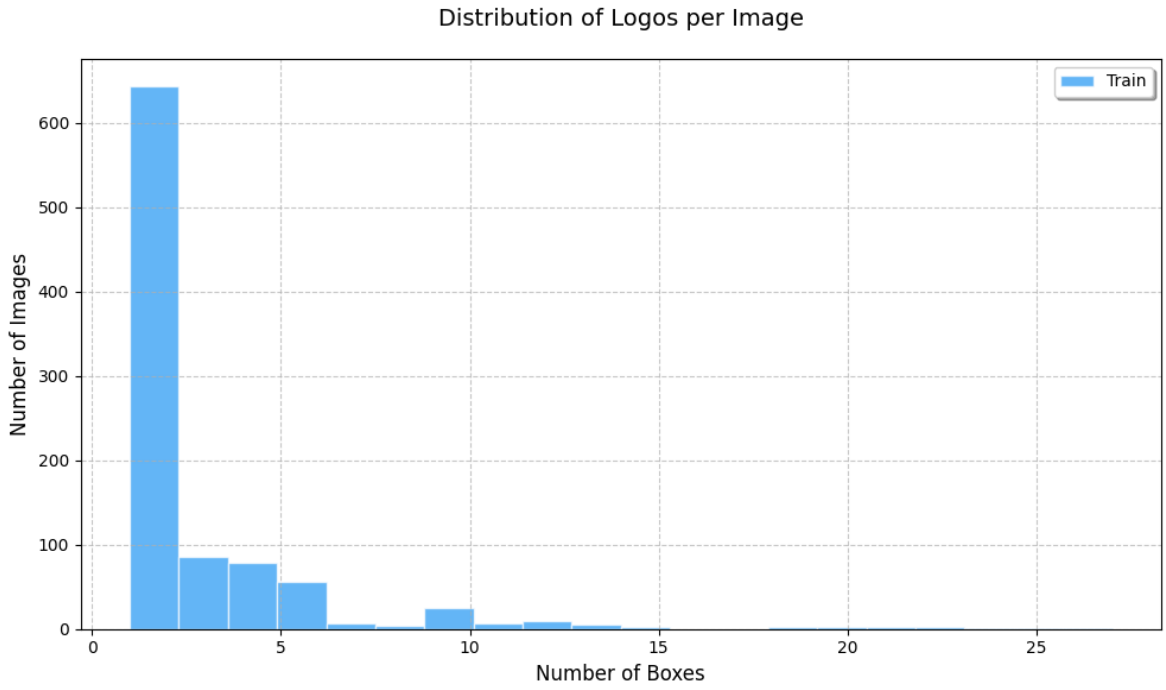

Figure 3

Class distribution is very imbalanced withing each set (train and test) and distributed differently between those sets. This might lead to biases or inherit inability in predicting correctly instances of (most likely) “sap” and “Spotify”. (fig. 4)

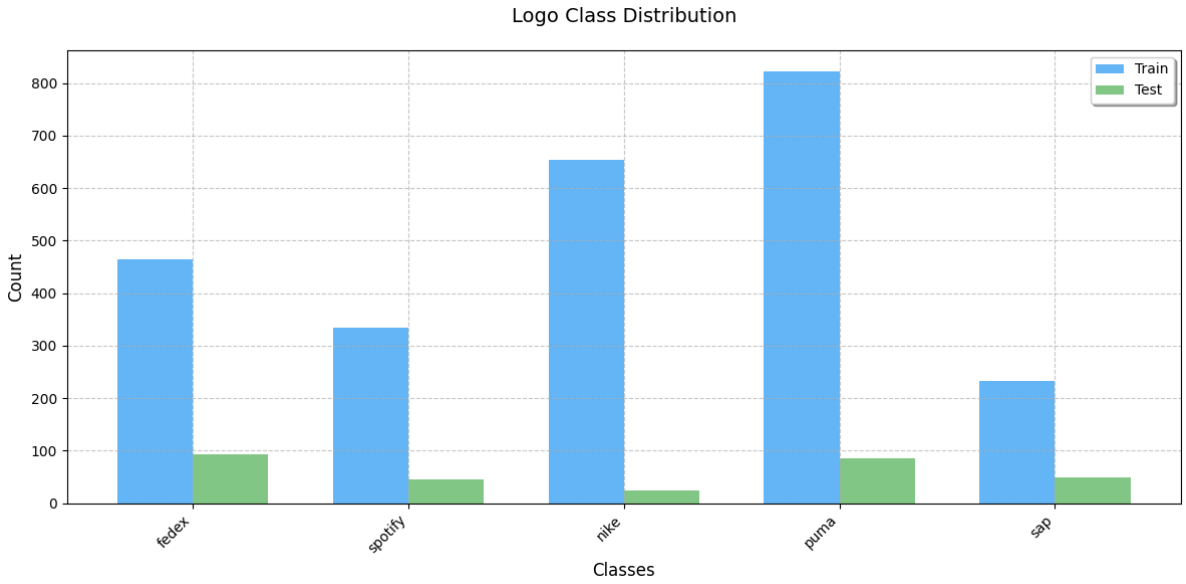

Figure 4

Base detector

For the base detector I choose YOLOv8 framework, being the SOTA for real-time object detection, and given my limited GPU & time resources. For same reason I choose nano architecture as my go to.

Improvements discussion

Data

Data improvements may come from several approaches:

- More data (the obvious)
- Balance the data
    - Enriching missing class
    - Randomly sampling the frequent classes so it will better match the less frequent ones
    - Weighting based on frequency
- Improve pre-processing and augmentation
    - Using random crops instead of resizing to avoid loss of information due to small objects
    - Increasing the number of objects per image through mosaic or cutmix
    - Inserting mirroring for better handling of mirroring effects (e.g. selfies)
- Increase image resolution – would be computational costly but should be useful for small objects. As a ballpark estimation, lets us assume a 1000X1000 image size and 100X100 bounding box (close enough to the averages) then the bounding box reduction rate when using 640X640 standard input is almost 60%. **Changing this is a very minor change in the overall model and train scheme but may become very fruitful – that is why I choose this improvement, simple yet powerful.**

Model

Model improvements may come in handy if the base architecture convergence has stagnated around lower values, meaning that the model is not rich enough to learn the problem in full. It might also be that the model shows overfitting to the train set - big decrease in test metric would indicate this, then we have an opportunity to use lighter architecture and save some resources (time/computational power).

Training

In the training section two important things that might effectively improve model convergence are the loss and learning rate. For example, we might introduce focal loss with factor 1.5 or higher to force the model into dealing with the harder samples. Learning rate is also important especially because we are using pretrained weights and we want the optimizer first to "jump" out of any local extremum it may be in, and then we need the learning rate to decay to avoid instability in convergence towards the final epochs.

Inference

There are some options to improve performance within the inference scope, mostly to bring the inferred image closer to the training data but that is dependent on the augmentations – for example if we used random cropping, we might want to do inference on tiles of the image to improve detection rates in small objects.

Implementation

I used python, ultralytics and troch for implementation.

One important thing to mention is that I use a preprocessing pipeline to create a validation split – 750 train images and 187 validation images (20% as rule of thumb) and edited the labels to numerical values alongside some dataset operation to make everything compatible with ultralytics.

See in the attached documentation (GitHub).

Results

Base Model

Train results

As expected, due to the high initial learning rate, training curves as less smooth, although convergence is clear. (fig. 5)

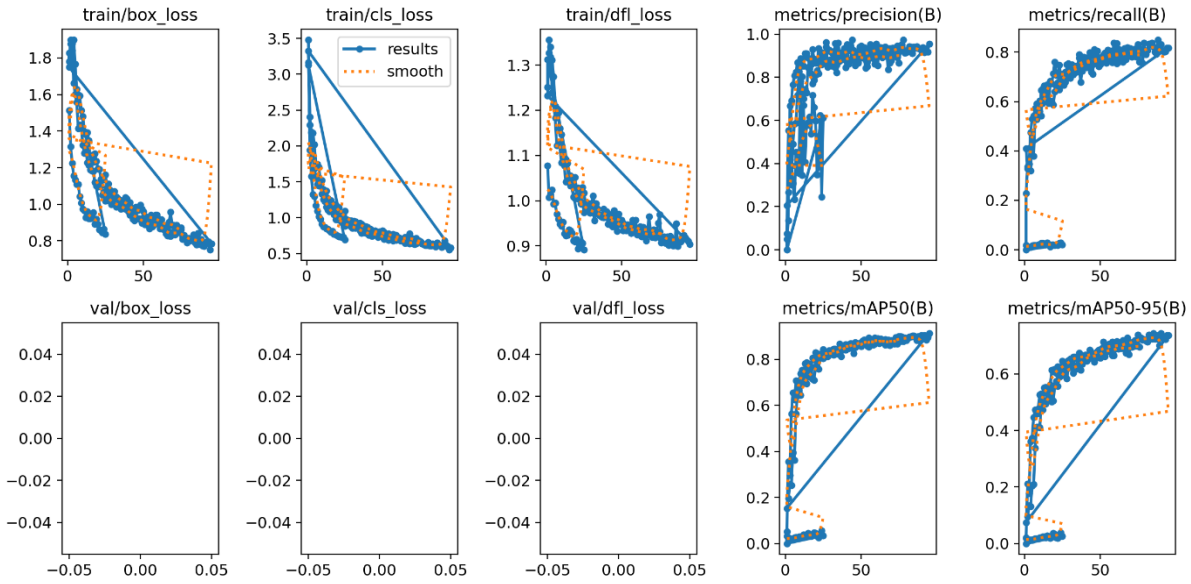

Figure 5

From the confusion matrix on the validation set we can see that there is no cross correlation between class, only between each class and background (i.e. false negative/positive). meaning our main concern should be the loss of information due to small objects (fig. 6)

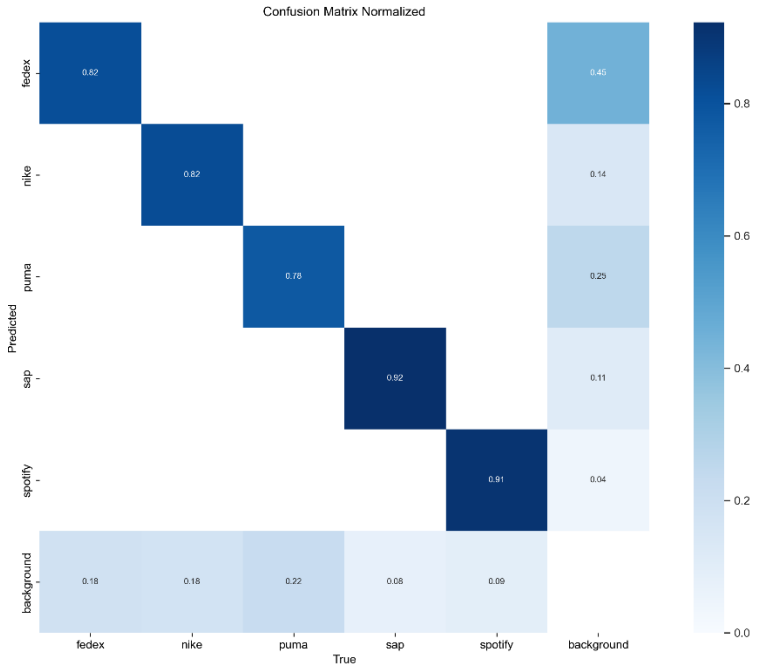

Figure 6

Test results

As expected, due to the train-test class distribution mismatch, we experience the lowest precision rate for “sap” (more frequent in test) and the lowest recall rate for “fedex” and “puma” (less frequent in test). (table 1)

Table 1

Per-Class Metrics

Class	Precision	Recall	mAP@.50	mAP@.50-.95
fedex	0.874	0.7719	0.8549	0.6971
nike	0.9447	0.8041	0.9001	0.7713
puma	0.904	0.7556	0.8529	0.6504
sap	0.8403	0.9231	0.9374	0.8189
spotify	0.9558	0.8906	0.9466	0.8564

Nonetheless, the overall performance is stable, evidence of a model with good generalizability. (table 2)

Table 2 Overall metrics

Metric	Test Set Value	Train Set Value
Precision	0.9038	0.9086
Recall	0.8291	0.8147
mAP50	0.8984	0.8874
mAP50-95	0.7588	0.7141

High Resolution Model

Train results

At higher resolution – 960x960, we can observe a smother convergence at higher rate, down to early stopping after 82 epochs (patience set to 10) over 95 epochs for base model. (fig. 7)

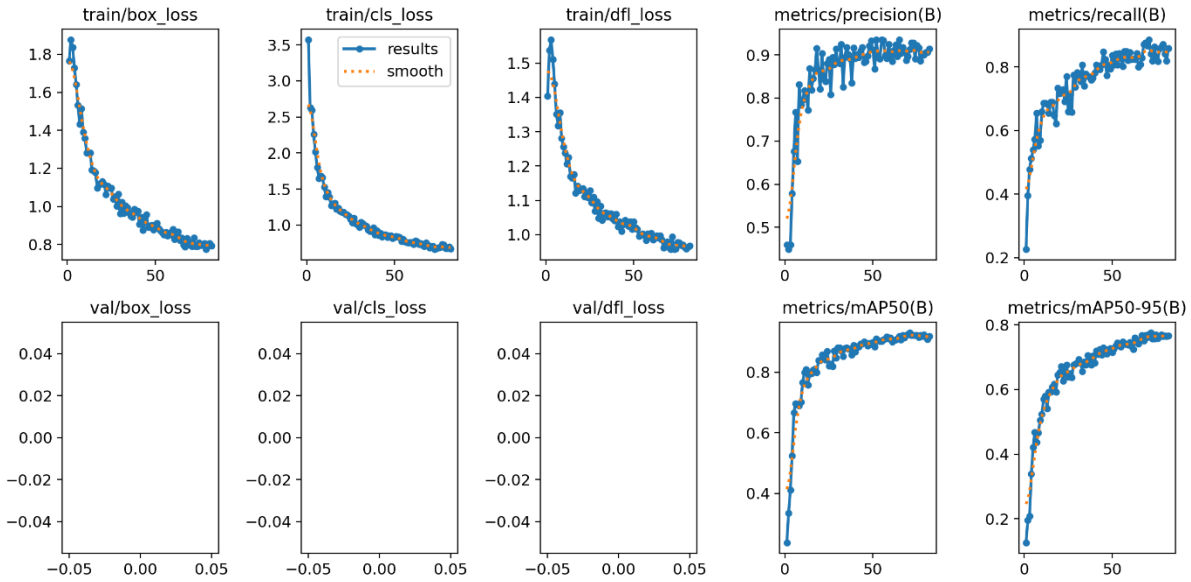

Figure 7

As for the validation set, now we do experience some cross correlation between class (fig 8)

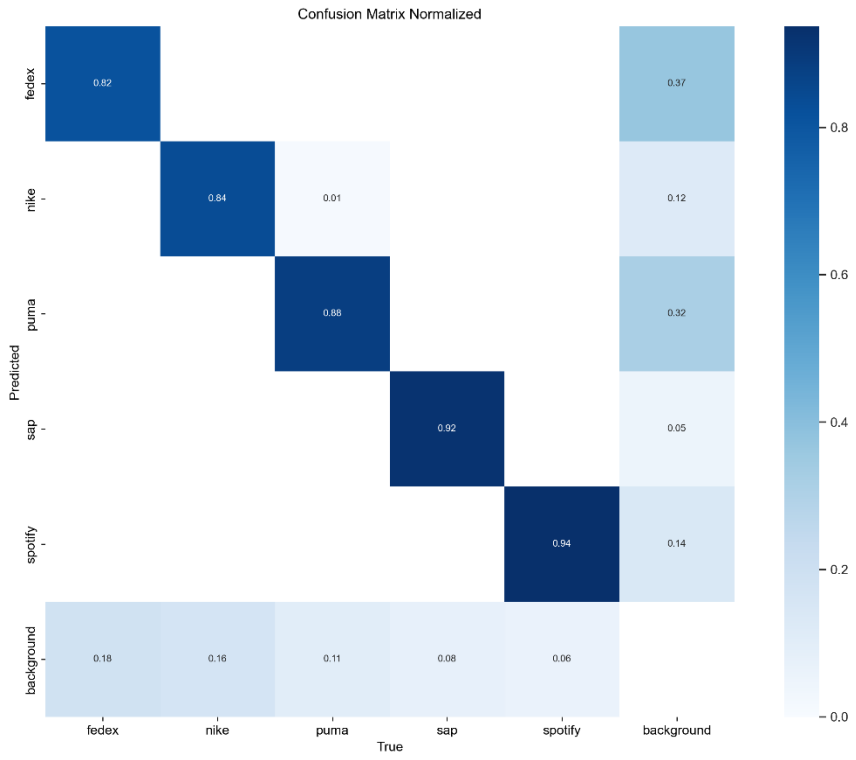

Figure 8

Since it is very low (only 2 mislabeled boxes) I shall inquire to verify the annotation, on fig. 9 row 3 column 2 we can see example of such mislabel. It seems more a problem of overlapping boxes of different classed than anything else, this can be resolved by adjusting the NMS confidence and IOU upon inference.

Test results

We can see that the high resolution model improved our per-class precision in most classes, with nike and puma suffering a little bit. (table 3)

Table 3

Per-Class Metrics

Class	Precision	Recall	mAP@.50	mAP@.50-.95
fedex	0.874	0.7719	0.8549	0.6971
nike	0.9447	0.8041	0.9001	0.7713
puma	0.904	0.7556	0.8529	0.6504
sap	0.8403	0.9231	0.9374	0.8189
spotify	0.9558	0.8906	0.9466	0.8564

With the overall results as in table 4, suggesting we've achieved a similar results on test set but at faster convergence. Further work should include adjustments of the NMS as

mentioned above and should include testing at larger scale to determine if this is actual improvement or just an effect caused by better fitting the train data (overfit)

Table 4 Overall metrics

Metric	Test Set Value	Train Set Value
Precision	0.9072	0.9268
Recall	0.7908	0.8413
mAP50	0.8728	0.9225
mAP50-95	0.7331	0.7748

Zero-shot case (section 2)

Concept

For this task I suggest using a VLM approach with one instance serving as an annotator and another as a judge. I can see that the human annotation is imperfect, that is why I believe this approach would serve best in a zero-shot case.

Implementation

I used Gemini API out of convenience and previous experience with it. Implementation in full is in zero_shot.py under scripts. Further information in the code documentation.

Results

The model successfully detected the 7 instances of logo and correctly classified them (up to semantic differences in label – "NBA" and not the full name), it has also been able to detect one missing label (possible human error).

Appendix A – section 2 full AI summary

```
--- Model Predictions ---

adidas 0.395 0.284 0.051 0.033

nba 0.604 0.298 0.028 0.056

miami-heat 0.5 0.441 0.361 0.128

adidas 0.521 0.25 0.032 0.02

nba 0.552 0.292 0.022 0.036

adidas 0.572 0.81 0.046 0.026

nba 0.641 0.811 0.022 0.038

adidas 0.459 0.139 0.016 0.009


--- Ground Truth Labels ---

adidas 0.40000 0.28504 0.04240 0.03504

adidas 0.59000 0.80470 0.03280 0.02393

adidas 0.54020 0.24765 0.01960 0.01667

nba_national_basketball_association 0.59940 0.28568 0.02120 0.05000

nba_national_basketball_association 0.54840 0.27821 0.02000 0.03077

nba_national_basketball_association 0.63320 0.80385 0.01680 0.03419

miami_heat 0.49600 0.42201 0.29600 0.12521


--- AI Judge Evaluation ---

Of course. Here is a detailed evaluation of the AI model's logo detection
performance.

### **Logo Detection Quality Evaluation**

---

#### **1. Correct Classifications**

The model successfully identified all 7 ground truth logos with accurate class
labels and well-aligned bounding boxes.

*   **adidas (3 instances):**

    *   GT: `0.40000 0.28504 0.04240 0.03504` | Prediction: `0.395 0.284 0.051
0.033` (Correct)

    *   GT: `0.59000 0.80470 0.03280 0.02393` | Prediction: `0.572 0.81 0.046
0.026` (Correct)
```

    *   GT: `0.54020 0.24765 0.01960 0.01667` | Prediction: `0.521 0.25 0.032 0.02` (Correct)

*   **nba_national_basketball_association (3 instances):**

    *   GT: `0.59940 0.28568 0.02120 0.05000` | Prediction: `nba 0.604 0.298 0.028 0.056` (Correct)

    *   GT: `0.54840 0.27821 0.02000 0.03077` | Prediction: `nba 0.552 0.292 0.022 0.036` (Correct)

    *   GT: `0.63320 0.80385 0.01680 0.03419` | Prediction: `nba 0.641 0.811 0.022 0.038` (Correct)

*   **miami_heat (1 instance):**

    *   GT: `0.49600 0.42201 0.29600 0.12521` | Prediction: `miami-heat 0.5 0.441 0.361 0.128` (Correct)

2. False Positives

The model incorrectly identified one additional logo that was not present in the ground truth labels.

*   `adidas 0.459 0.139 0.016 0.009`

3. Missed Detections (False Negatives)

The model did not miss any of the ground truth logos.

*   None.

4. Overall Accuracy

The model's performance is excellent. It achieved a perfect recall, successfully detecting all 7 logos provided in the ground truth. The bounding boxes are generally well-placed, and the class predictions are accurate (accounting for acceptable aliases like `nba` for `nba_national_basketball_association`). The model's precision is slightly reduced by a single false positive `adidas` detection. Overall, this is a very strong and reliable result.

5. Suggestions for Human Labeler

The model detected a small `adidas` logo at coordinates `0.459 0.139` that was not included in the ground truth. It is recommended that the human annotator **re-examine this specific area of the image**. It is possible that a small, partially obscured, or low-resolution adidas logo was missed during the initial labeling process.