Evaluation Results

Aim

The new evaluation pipeline is itself evaluated to assess the quality and reliability of the evaluations it gives to miners. We include various measures on the performance of the VLM predictions at the heart of the new scoring mechanism.

Data

We use a real source of ground-truth annotations for selected football video clips by making use of Soccernet's Tracking 2023 Dataset*. Videos from the test split of the dataset were used, each containing 750 hand annotated frames. A random subset of 60 frames from across 9 videos were sampled and used in the following evaluations.

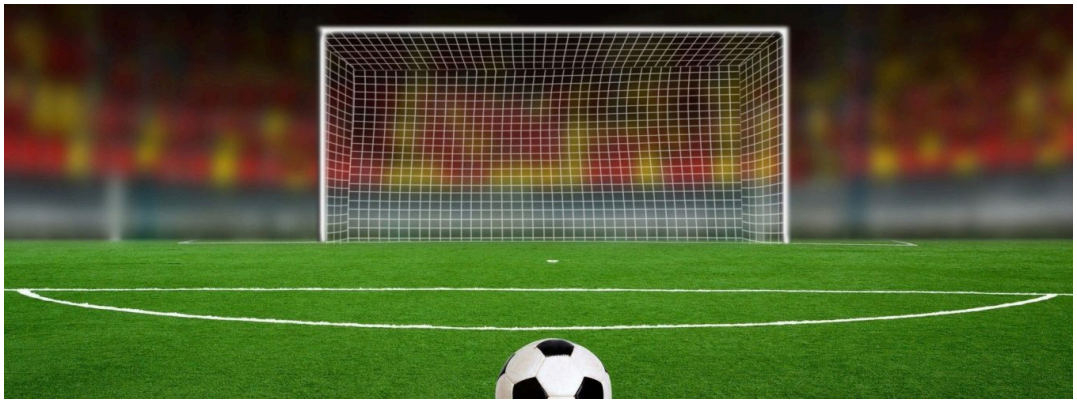
▼ Names of Video Frames Tested

	video_name	frame_number
0	SNMOT-116	1
1	SNMOT-116	101
2	SNMOT-116	201
3	SNMOT-116	301
4	SNMOT-116	501
5	SNMOT-116	701
6	SNMOT-117	1
7	SNMOT-117	101
8	SNMOT-117	201

9	SNMOT-117	301
10	SNMOT-117	401
11	SNMOT-117	501
12	SNMOT-117	601
13	SNMOT-117	701
14	SNMOT-118	1
15	SNMOT-118	101
16	SNMOT-118	201
17	SNMOT-118	301
18	SNMOT-118	401
19	SNMOT-118	501
20	SNMOT-118	601
21	SNMOT-118	701
22	SNMOT-119	1
23	SNMOT-119	101
24	SNMOT-119	201
25	SNMOT-119	301
26	SNMOT-119	401
27	SNMOT-119	501

28	SNMOT-119	601
29	SNMOT-119	701
30	SNMOT-120	1
31	SNMOT-120	201
32	SNMOT-120	301
33	SNMOT-120	401
34	SNMOT-120	601
35	SNMOT-120	701
36	SNMOT-121	1
37	SNMOT-121	101
38	SNMOT-121	201
39	SNMOT-121	301
40	SNMOT-121	401
41	SNMOT-121	501
42	SNMOT-121	601
43	SNMOT-121	701
44	SNMOT-122	1
45	SNMOT-122	101
46	SNMOT-122	201

47	SNMOT-123	1
48	SNMOT-123	101
49	SNMOT-123	201
50	SNMOT-123	301
51	SNMOT-123	401
52	SNMOT-123	501
53	SNMOT-123	601
54	SNMOT-123	701
55	SNMOT-124	101
56	SNMOT-124	201
57	SNMOT-124	401
58	SNMOT-124	501
59	SNMOT-124	601
60	SNMOT-124	701



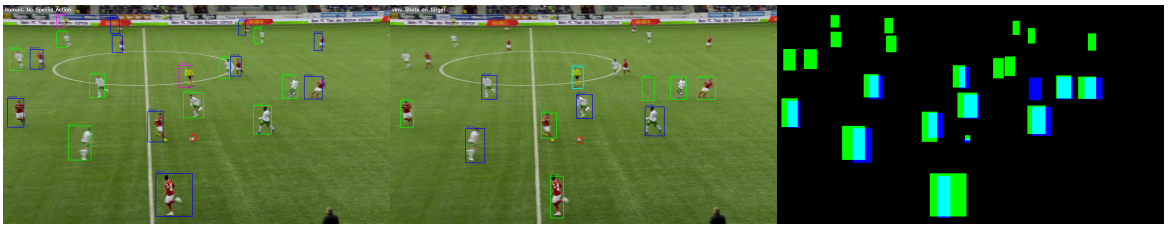
** Note: this dataset only includes football videos as opposed to wider domains which this pipeline is capable of evaluation. Furthermore, these videos are all from a particular camera angle and the human annotations only include bounding boxes (as opposed to keypoints or action labels). In future, we shall expand this evaluation to include more datasets that cover these areas*

Results

▼ Test 1: Quality of VLM-generated Pseudo GT Annotations

To measure the quality of the annotations generated by the VLM, we compare it against the human annotations and measure the absolute difference (l1 error) per frame in terms of the **object count** error (number of objects detected in a frame) and the preciseness of the size and placement of all bounding boxes by measuring their **Intersection over Union** (IoU).

Here is an example of a frame with the human annotations (left), the generated annotations (middle) and the IoU map (right). We can see the pseudo GT annotations are almost identical to the human GT annotations, except for a miner shift in the position of the goalkeeper's bbox. However, the VLM is far from perfect. Below is an example with quite a few missing bbox detections in the background:

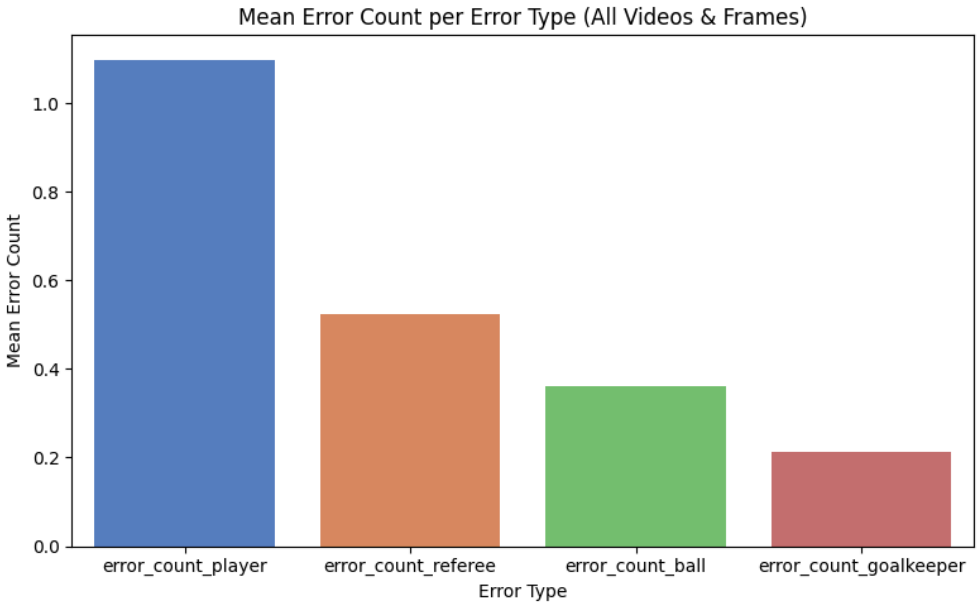


Similarly, below is an example where the VLM identifies bboxes for additional people in the foreground which were not meant to be labelled (or not labelled by the human annotators)

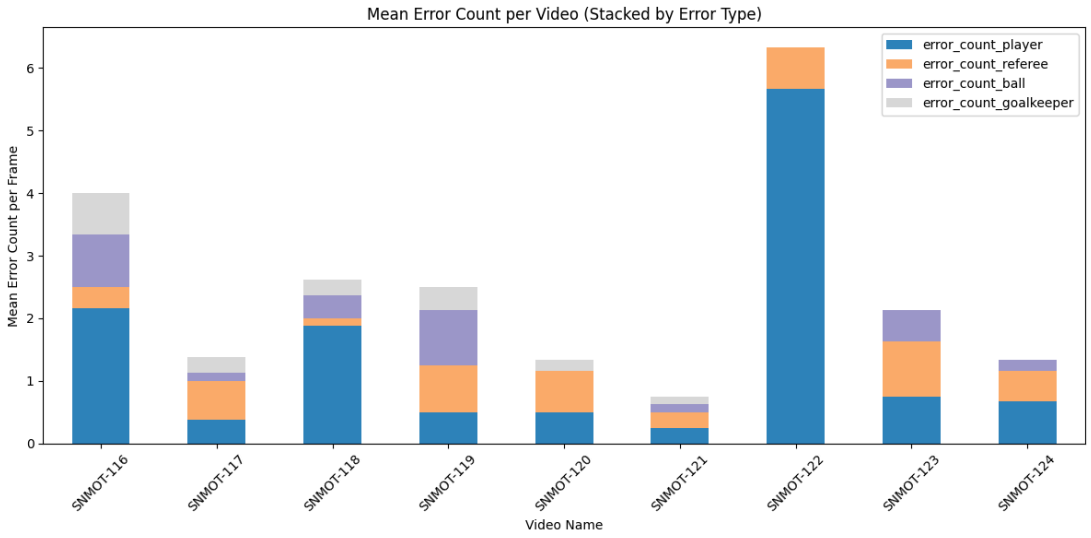
So how accurate is the VLM at generating pseudo GT annotations?

1. **Object Count Error**

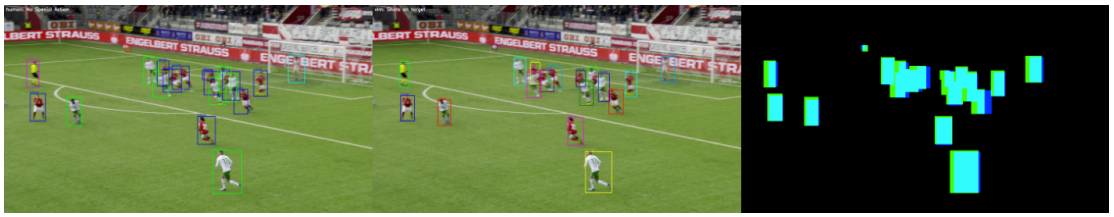
- While many frames scored perfectly in this regard (e.g. Frame 301 in video SNMOT-122), the worst result came from frame 101 in the same video (SNMOT-122)
- The mean difference/error in object count overall highlights how low this error is (the highest being the player error count with an average of 1 player either added/missing):



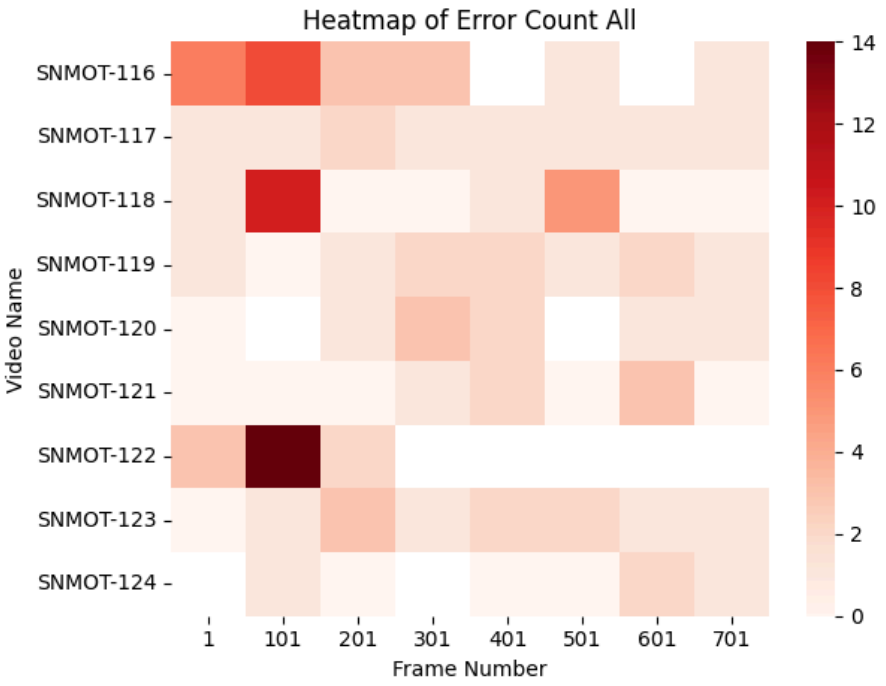
- The mean difference/error in object counts per video:



- Investigating the video with the highest error counts (SNMOT-122), we find the quality of predictions to be generally good except for one frame which had a group of players missing (2nd image shown below)

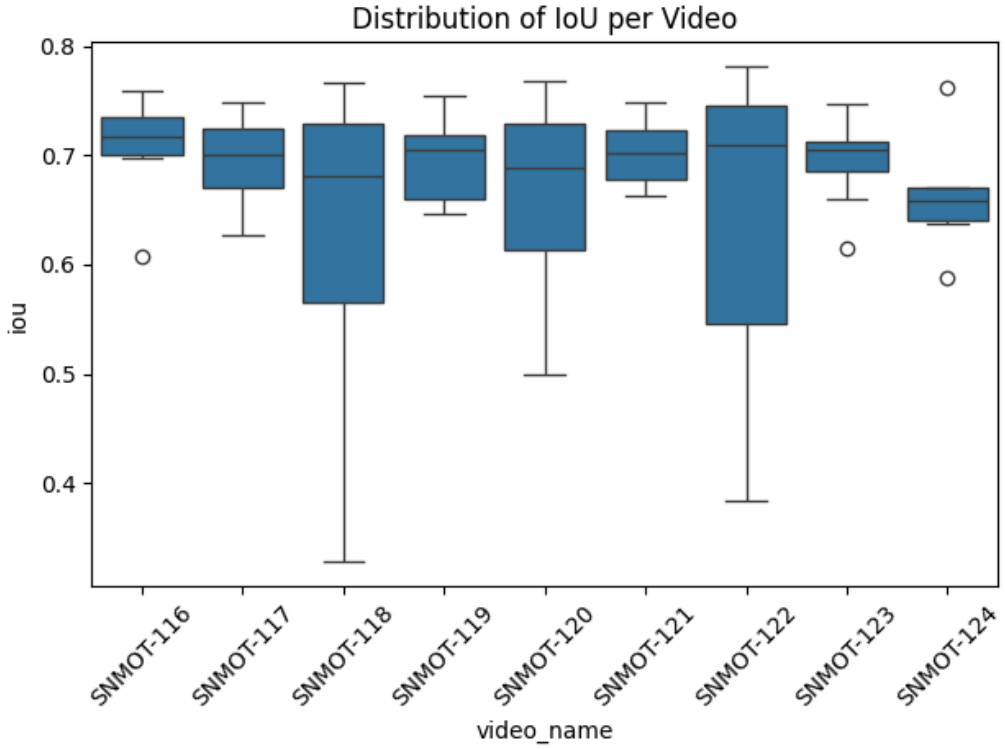


- In fact, this turns out to be the worst pseudo GT annotations of all the frames tested

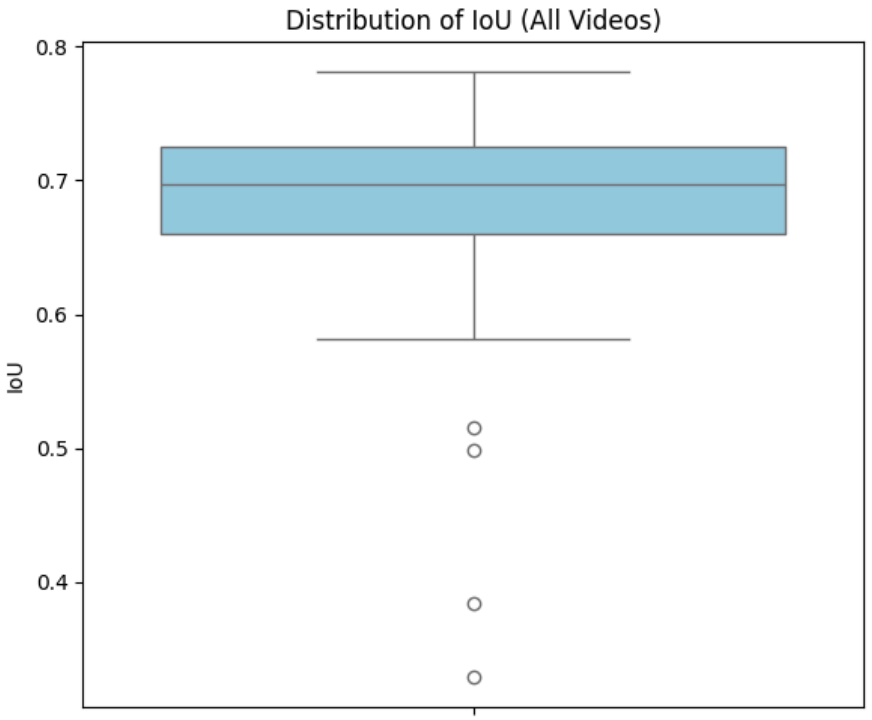


2. **Intersection Over Union (IoU)**

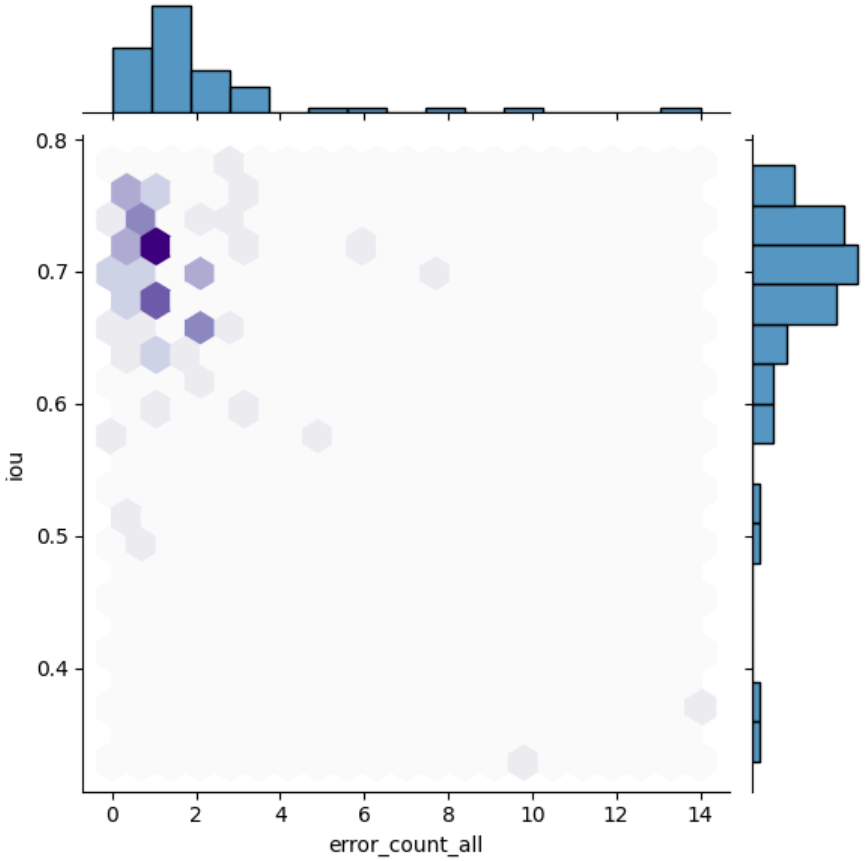
- The IoU per video is quite high (approx 70% on all videos)



- The overall IoU across all videos is again around 70%.



- The error count vs IoU shows an inverse correlation (i.e. the higher the IoU, the lower the error in object counts) which would suggest the IoU is let down by missing or additional bounding boxes (as opposed to badly placed or imprecise bounding boxes). In other words, the pseudo GT-annotations may contain errors related to added or missing bboxes (errors in count), but the size and locations of the bboxes which are predicted are generally very precise.



▼ Test 2: Quality of VLM-as-Judge

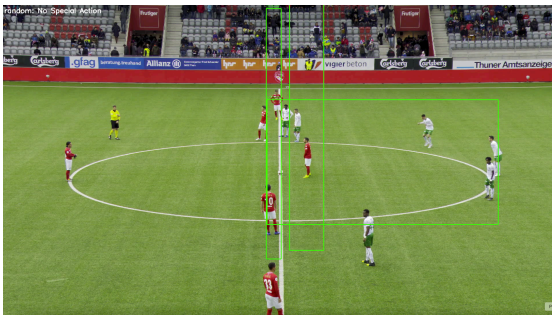
In order to assess the quality of the VLM-as-Judge's decisions, we assume the human annotations to be superior and the pseudo-GT annotations to be equal to or worse than the human annotations. We also assume that a frame with no bboxes (ie empty annotations) is always worse than the pseudo-GT annotations.

Human GT ≥ Pseudo-GT > Null Predictions

Since we know which the VLM-as-Judge should pick, we have a way of evaluating if its decisions were accurate. So we compare the human annotations to the pseudo-GT annotations and mark how often the VLM-as-judge correctly chooses the human annotation (or awards a tie) and how often it incorrectly chooses the pseudo-GT as superior. Similarly, we compare the empty/null predictions with the pseudo-GT annotations and mark how often the VLM-as-Judge correctly chooses the pseudo-GT annotations over the null predictions.

Exploits

We also introduce 2 types of bbox exploits: a single large bbox that covers the entire frame and a random bbox exploit (a random number of bboxes scattered around the frame with random sizes). *The frames below show an example of the exploits tested. (left) Exploit 1: Single Large bbox. (right) Exploit 2: Random Bboxes*



For each frame tested, it is safe to assume that such the exploits should be judged as worse than the Pseudo-GT annotations.

Pseudo-GT > Exploit

So we also mark how often the VLM-as-Judge correctly chooses the pseudo-GT annotations over the exploits to measure how robust the VLM-as-Judge would be against such attacks.

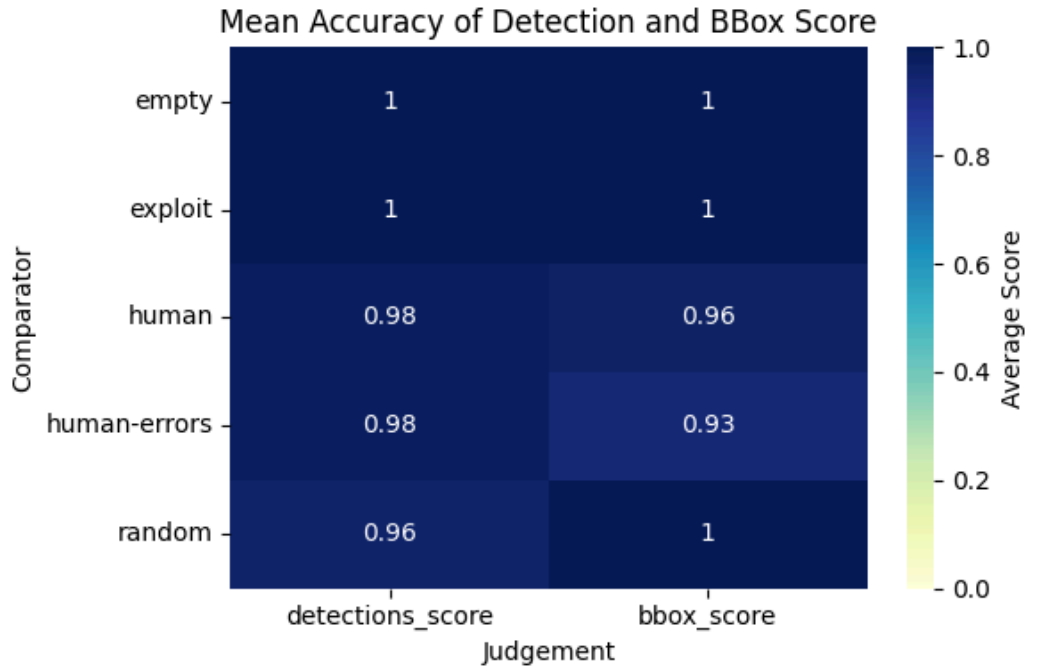
Human (Annotations) with Errors

We also test the VLM-as-judge by giving it human annotations to compare against human annotations with errors (ie by removing one bbox and the ball if present). In this test, we assume that the human annotations is better than the human annotations with errors

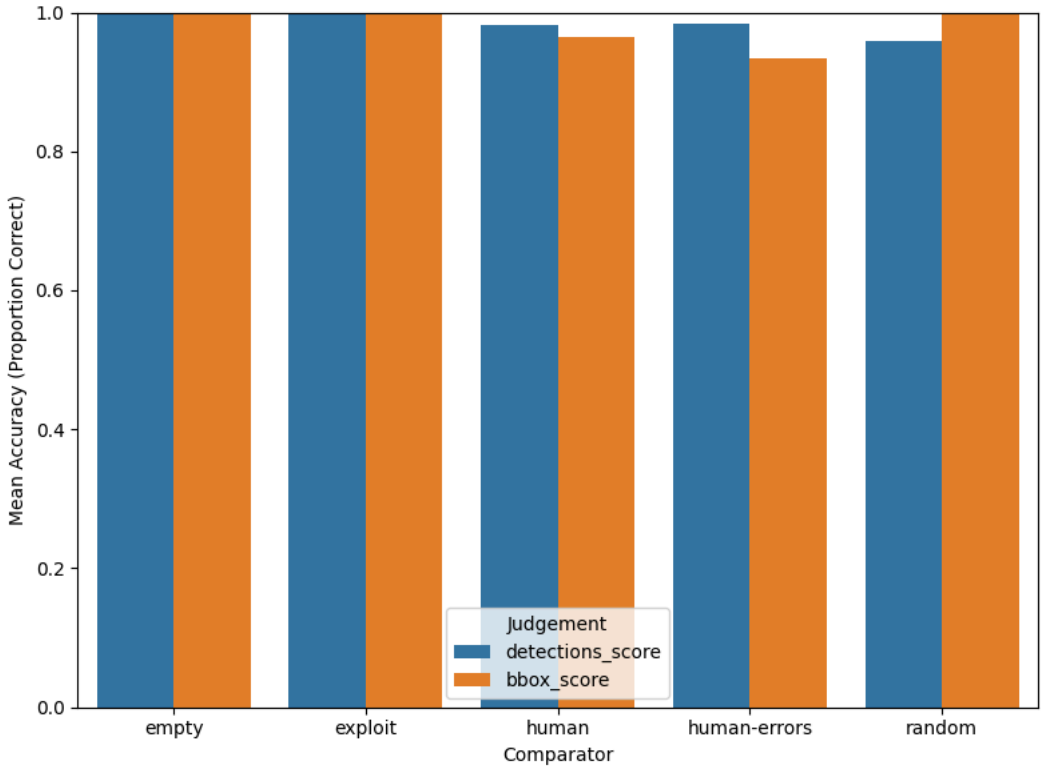
Human ≥ Human with Errors

Results

We can see the VLM-as-judge is highly accurate; when comparing pseudo-GT vs human-GT, the VLM judged correctly 98% of the time for detections and 96% of the time for bboxes! It judged correctly 100% of the time when comparing pseudo-GT vs null predictions (empty).



It also scored very highly against both types of exploits tested; 96% correct and 100% correct when comparing pseudo-GT vs random bboxes.



It scored 100% correct when comparing pseudo-GT vs exploit 1 (a single large bbox). Inspecting some of the VLM's reasoning for its judgements, we see it

correctly recognises the exploit when comparing it against the Pseudo-GT annotations (Image A = Pseudo-GT, Image B = Exploit):

> *"Image A has individual bounding boxes for each player, while Image B has **a single bounding box encompassing all players**. The intersection image shows that Image A's bounding boxes are more precise for individual players...Image A's method of annotating each player separately allows for better detection of individual players, which is crucial for detailed analysis. Image B's **single bounding box is less precise and does not distinguish between players.**"*

▼ Test 3: Runtime

The settings are configurable. For this test, the following configurations were used. The VLM is `QWEN 2.5 72billion Instruct` which is served for free on `Chutes` (main API provider) and `OpenRouter` (backup API provider). There is a `3 second wait` between a failed call and a retry and there a maximum of `2 retries` per unique call per API provider. Although all calls are asynchronous, we limit the number of `concurrent calls to 3` at any given time.

Given these settings, the time taken to run the pipeline is shown below.

# frames	# miners	VLM 1: generating pseudo GT	VLM2: judge	End-to-end
1	1	15 s	21 s	36 s
3	1	27 s	1.65 mins	2.1 mins
5	1	48 s	1 min	1.8 mins
5	10	49 s	7 mins	8 mins
5	30	58 s	24 mins	25 mins

▼ view results.json

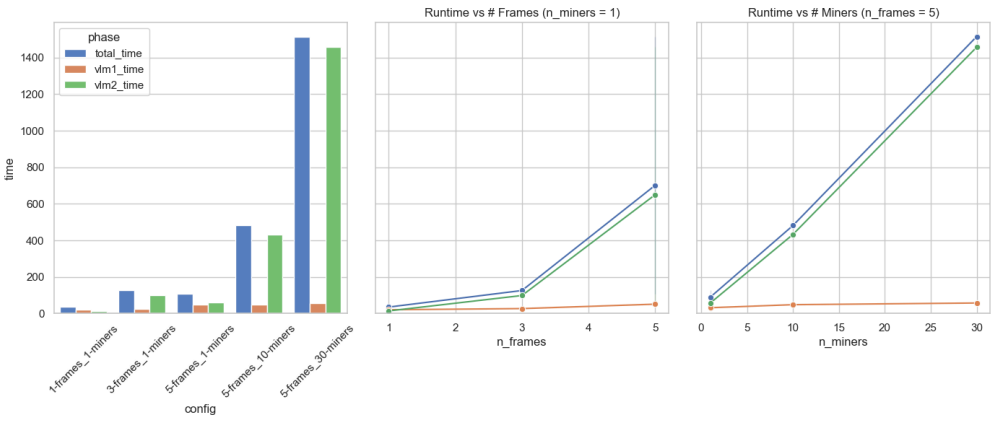

```
[
 {
  "n_frames": 1,
  "n_miners": 1,
  "n_frames_successful": {
   "test_0": 1
```

```
  },
  "total_time": 35.75398302078247,
  "vlm1_time": 21.077472925186157,
  "vlm2_time": 14.676510095596313
 },
 {
  "n_frames": 3,
  "n_miners": 1,
  "n_frames_successful": {
   "test_0": 3
  },
  "total_time": 126.26037240028381,
  "vlm1_time": 27.339020013809204,
  "vlm2_time": 98.92135238647461
 },
 {
  "n_frames": 5,
  "n_miners": 1,
  "n_frames_successful": {
   "test_0": 5
  },
  "total_time": 107.91154599189758,
  "vlm1_time": 47.91727304458618,
  "vlm2_time": 59.9942729473114
 },
 {
  "n_frames": 5,
  "n_miners": 10,
  "n_frames_successful": {
   "test_0": 5,
   "test_1": 5,
   "test_2": 5,
   "test_3": 5,
   "test_4": 5,
   "test_5": 5,
   "test_6": 5,
   "test_7": 5,
   "test_8": 5,
   "test_9": 5
  },
```

```
  "total_time": 481.7961480617523,
  "vlm1_time": 48.78150200843811,
  "vlm2_time": 433.0146460533142
 },
 {
  "n_frames": 5,
  "n_miners": 30,
  "n_frames_successful": {
   "test_0": 5,
   "test_1": 5,
   "test_2": 5,
   "test_3": 5,
   "test_4": 5,
   "test_5": 5,
   "test_6": 5,
   "test_7": 5,
   "test_8": 5,
   "test_9": 5,
   "test_10": 5,
   "test_11": 5,
   "test_12": 5,
   "test_13": 5,
   "test_14": 5,
   "test_15": 5,
   "test_16": 5,
   "test_17": 5,
   "test_18": 5,
   "test_19": 5,
   "test_20": 4,
   "test_21": 5,
   "test_22": 5,
   "test_23": 5,
   "test_24": 5,
   "test_25": 4,
   "test_26": 5,
   "test_27": 5,
   "test_28": 5,
   "test_29": 5
  },
  "total_time": 1515.0632209777832,
```

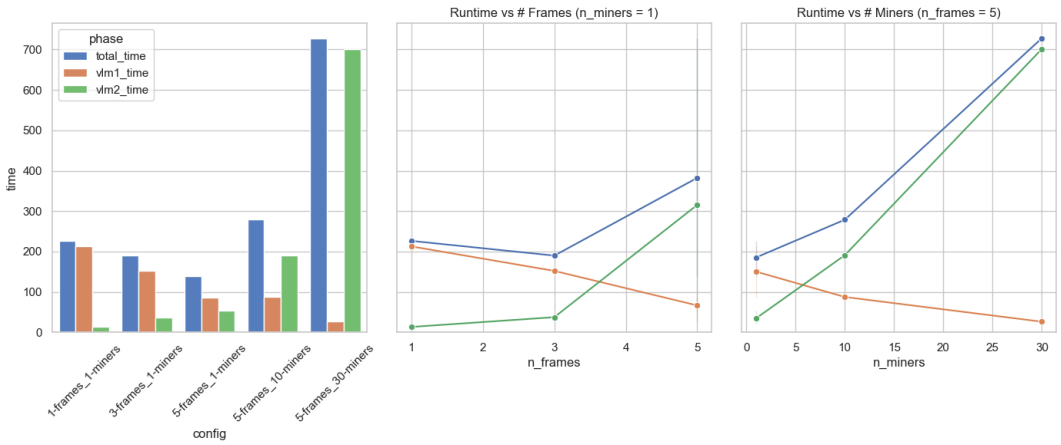
```
    "vlm1_time": 57.9199161529541,
    "vlm2_time": 1457.143304824829
  }
]
```


We notice that the time for VLM 1 goes up according to the number of frames tested (since it generates annotations per frame) but remains constant as the number of miners scales. The time taken for VLM2 goes up as the number of frames and miners scale (as it judges on a frame by frame basis).



V2: Batch-VLM-as-Judge

We modified the VLM-as-Judge to judge so that multiple frames were passed for judging in a single call. This way the VLM-as-Judge would only need to be called once per miner. We also increased the maximum number of `retries to 3` and the maximum number of `concurrent calls to 5`. The new times are as follows:



Again the times taken to generate pseudo-GT annotations remain constant with the number of miners (actually seeming to decrease slightly due to retries in

earlier calls). However, the time taken for the VLM-as-Judge was half the time taken since it only scales according to the number of miners now.

# frames	# miners	VLM2: judge (v1)	VLM2: judge (v2)
1	1	21 s	14 s
3	1	1.65 mins	38 s
5	1	1 min	54 s
5	10	7 mins	3 mins
5	30	24 mins	12 mins

▼ view .json results


```
[
 {
  "n_frames": 1,
  "n_miners": 1,
  "n_frames_successful": {
   "test_0": 1
  },
  "total_time": 226.48286509513855,
  "vlm1_time": 212.79618215560913,
  "vlm2_time": 13.686682939529419
 },
 {
  "n_frames": 3,
  "n_miners": 1,
  "n_frames_successful": {
   "test_0": 3
  },
  "total_time": 190.05780816078186,
  "vlm1_time": 152.26924324035645,
  "vlm2_time": 37.788564920425415
 },
 {
  "n_frames": 5,
  "n_miners": 1,
  "n_frames_successful": {
   "test_0": 5
  },
```

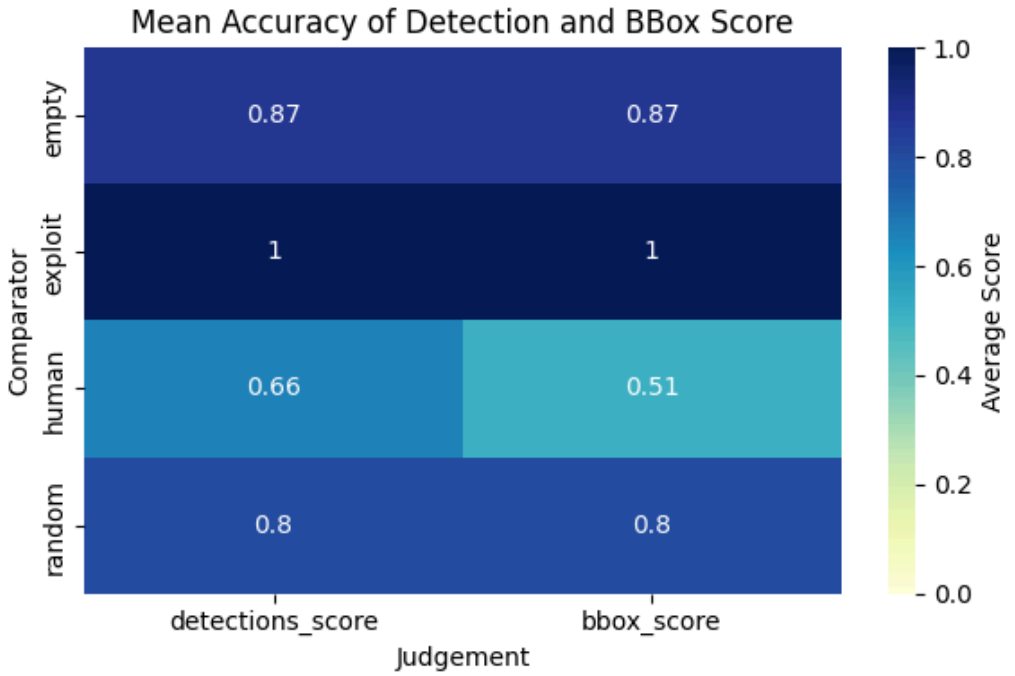
```
  "total_time": 140.18602108955383,
  "vlm1_time": 86.32097506523132,
  "vlm2_time": 53.86504602432251
 },
 {
  "n_frames": 5,
  "n_miners": 10,
  "n_frames_successful": {
   "test_0": 5,
   "test_1": 5,
   "test_2": 5,
   "test_3": 5,
   "test_4": 5,
   "test_5": 5,
   "test_6": 5,
   "test_7": 5,
   "test_8": 5,
   "test_9": 5
  },
  "total_time": 278.83640003204346,
  "vlm1_time": 87.75390100479126,
  "vlm2_time": 191.0824990272522
 },
 {
  "n_frames": 5,
  "n_miners": 30,
  "n_frames_successful": {
   "test_0": 5,
   "test_1": 5,
   "test_2": 5,
   "test_3": 5,
   "test_4": 5,
   "test_5": 5,
   "test_6": 5,
   "test_7": 5,
   "test_8": 5,
   "test_9": 5,
   "test_10": 5,
   "test_11": 5,
   "test_12": 5,
```

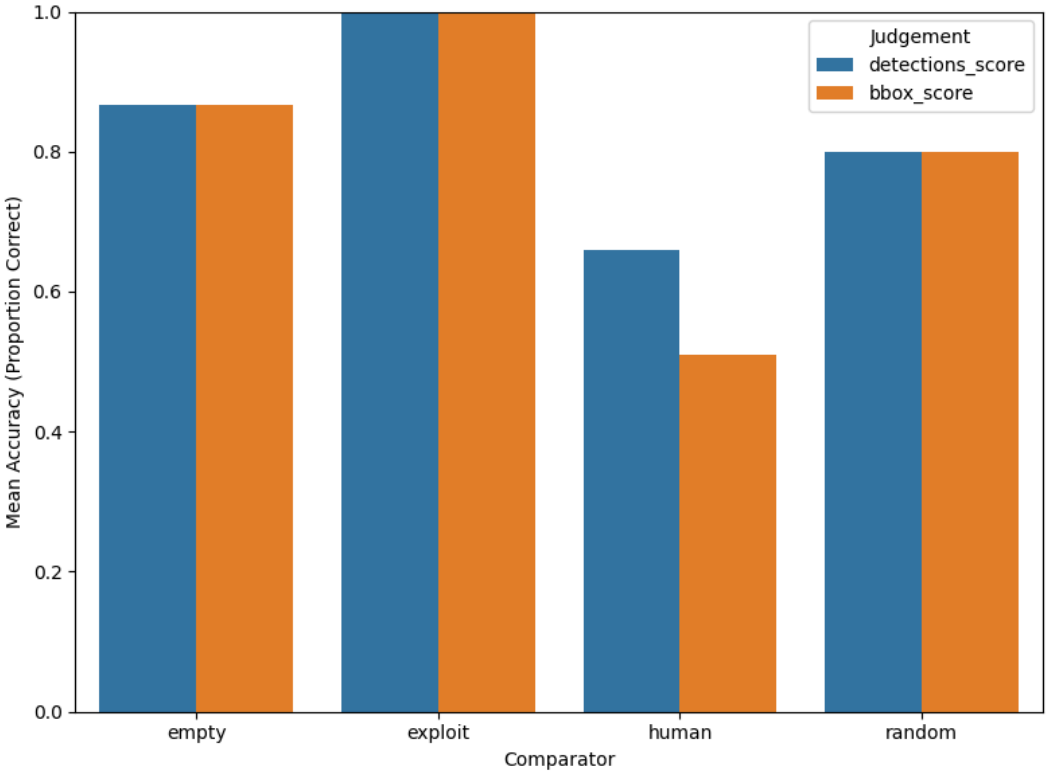
```
      "test_13": 5,
      "test_14": 5,
      "test_15": 5,
      "test_16": 5,
      "test_17": 5,
      "test_18": 5,
      "test_19": 5,
      "test_20": 5,
      "test_21": 5,
      "test_22": 5,
      "test_23": 5,
      "test_24": 5,
      "test_25": 5,
      "test_26": 5,
      "test_27": 5,
      "test_28": 5,
      "test_29": 5
    },
    "total_time": 727.876012802124,
    "vlm1_time": 26.6161367893219,
    "vlm2_time": 701.2598760128021
  }
]
```


However, this approach cannot handle too many frames at once due to the model's limited context window:

> {'object': 'error', 'message': "The input (48695 tokens) is longer than the model's context length (32768 tokens).", 'type': 'BadRequestError', 'param': None, 'code': 400}

And even if the size of the image is reduced, the quality of the VLM-as-Judge drops.





So we abandoned this approach in favour of the slower frame-by-frame version

Conclusions

- The quality of the VLM-generated pseudo-GT annotations: The pseudo GT-annotations may contain errors now and again related to added or missing bboxes (errors in count), but the size and locations of the bboxes which are predicted are generally very precise!
- The quality of the VLM-as-Judge: This part proves to be highly accurate (96%-100%) and the VLM was robust against the 2 types of exploits tested.