Improvement over Masked Retraining Teacher-Student Framework for Domain Adaptive Object Detection

# ***Abstract***

Domain Adaptive Object Detection (DAOD) seeks to adapt detectors to new, unlabeled domains, where teacher-student frameworks have shown significant promise. The Masked Retraining Teacher-Student (MRT) framework advances this paradigm by using a Masked Autoencoder (MAE) branch to help the student model better learn target domain characteristics and overcome the challenge of noisy pseudo-labels. However, the original MRT framework utilizes a simple random masking strategy, which may sub-optimally guide the self-supervised feature learning process. This report investigates the enhancement of the MRT framework by integrating two advanced, principled masking strategies to improve the efficacy of its MAE branch.

* First, we propose an Attention-Guided Masking method that leverages the teacher model's own attention maps to identify and mask semantically salient regions, thereby focusing the reconstruction task on features most relevant to object detection.
* Second, we implement a Uniform Masking strategy to ensure a more even spatial distribution of masked patches, compelling the model to learn from a more diverse set of local contexts. We hypothesize that these deliberate masking techniques will enable the student model to learn more robust representations of the target domain, improving its ability to escape local optima caused by incorrect pseudo-labels and ultimately boosting overall detection performance. *Code can be found at* [*https://github.com/amirreza1998/MRT-release/*](https://github.com/amirreza1998/MRT-release/)

# ***Introduction***

Object detection, a cornerstone of computer vision, has achieved remarkable success in recent years, largely due to the power of deep learning models like CNNs and Transformers. These models, however, often face a significant challenge when deployed in the real world: the problem of

**domain shift**. A model trained meticulously on a "source" dataset with abundant labels can experience a drastic drop in performance when applied to a "target" dataset that has a different data distribution—for instance, due to changes in weather, camera types, or geographical location. Manually annotating a new dataset for every target domain is prohibitively expensive and time-consuming. This challenge has spurred the development of Unsupervised Domain Adaptation (UDA), a field dedicated to adapting models to new domains without requiring any target labels.

Within UDA for object detection (DAOD),

**teacher-student frameworks** have emerged as a particularly effective paradigm. In this approach, a "teacher" model, which is typically a more stable version of the detector, generates pseudo-labels on the unlabeled target data. These pseudo-labels are then used to supervise the training of a "student" model. While powerful, this method's success is fundamentally limited by the quality of the pseudo-labels. Due to the domain gap, the teacher often produces a limited number of correct boxes and many incorrect ones. These noisy labels can mislead the student, trapping it in a local optimum and leading to sub-optimal performance.

To address this very issue, Zhao et al. proposed the

**Masked Retraining Teacher-Student (MRT) framework**. MRT introduces two key innovations to improve the robustness of the student model against noisy supervision. First, it incorporates a

**Masked Autoencoder (MAE) branch** as a parallel, self-supervised task. This MAE branch forces the student model to reconstruct masked portions of feature maps from target images, helping it to "capture target domain characteristics" and become a "more data-efficient learner". Second, MRT employs a

**selective retraining mechanism**, which periodically re-initializes parts of the student model to allow it to "jump out of the local optimum biased to the incorrect pseudo labels".

The MRT framework has proven highly effective, yet its MAE component relies on a simple **random masking** strategy. This approach, while straightforward, treats all regions of an image as equally important for the reconstruction task. As the comprehensive survey by Li et al. on masked modeling illustrates, more advanced and principled masking strategies can lead to more powerful representations.

Building upon this strong foundation, our work investigates whether the performance of MRT can be further improved by enhancing its MAE branch with more sophisticated masking strategies. We hypothesize that a more deliberate masking approach can better guide the student model to learn features that are more relevant to the downstream task of object detection. Specifically, we explore two methods:

**Attention-Guided Masking**, which leverages the teacher model's attention to focus reconstruction on semantically important regions , and

**Uniform Masking**, which ensures a more even spatial distribution of masked patches to encourage learning from a wider variety of local contexts. Our goal is to demonstrate that by optimizing this core self-supervised component, we can further enhance the student's representation learning, leading to a more robust and accurate domain adaptive object detector.

# ***RELATED WORK***

Our research is situated at the intersection of unsupervised domain adaptation for object detection and self-supervised representation learning. We build directly upon recent advances in teacher-student frameworks and explore enhancements inspired by a broader understanding of masked modeling techniques.

1. **The Foundational Framework: MRT by Zhao et al.**

The foundational work for our investigation is the **Masked Retraining Teacher-Student (MRT) framework** proposed by Zhao et al. in their 2023 ICCV paper, "Masked Retraining Teacher-Student Framework for Domain Adaptive Object Detection." This paper directly addresses a critical challenge in Domain Adaptive Object Detection (DAOD): the prevalence of low-quality pseudo-labels generated by teacher models. The domain shift between source and target data often leads to a limited number of pseudo-boxes and a high rate of incorrect predictions, which can mislead the student model and trap it in a sub-optimal state.

To combat this, Zhao et al. introduced a novel framework with two primary contributions built upon a Deformable DETR-based teacher-student baseline:

1. **Customized Masked Autoencoder (MAE) Branch:** The core innovation of MRT is the integration of an MAE branch that operates in parallel with the main detection task. This branch takes the multi-scale feature maps from the target domain images, masks a significant portion of them, and tasks the student's transformer encoder with reconstructing the missing features via a lightweight auxiliary decoder. The key insight is that this self-supervised reconstruction task forces the student model to learn rich, intrinsic representations of the target domain's visual characteristics. By learning to predict missing context, the model becomes more data-efficient, enabling it to better leverage the limited number of high-quality pseudo-labels provided by the teacher.
2. **Selective Retraining Mechanism:** Recognizing that incorrect pseudo-labels can still corrupt the student model over time, Zhao et al. introduced a selective retraining mechanism. Periodically, they re-initialize the weights of the student's backbone and encoder, while keeping the decoder fixed. This acts as a "reset," allowing the student to escape local optima it may have fallen into due to overfitting on noisy labels. Crucially, the student is re-initialized with weights that have been pre-trained with the MAE branch, ensuring it starts from a state that is already well-adapted to the target domain. The teacher model is continually updated via Exponential Moving Average (EMA) and is not retrained, ensuring the stability of the pseudo-label generation process.

While powerful, MRT's MAE branch relies on a simple **random masking** strategy. This approach treats all regions of an image feature map as equally important for reconstruction, which may not be the most efficient or effective way to guide the model's learning process for a complex task like object detection. Our work aims to enhance this specific component of the MRT framework by exploring more deliberate and principled masking strategies.

1. **Inspiration for Advanced Masking: A Survey by Li et al.**

Our proposed improvements draw inspiration from the broader landscape of self-supervised learning, as systematically reviewed by Li et al. in "Masked Modeling for Self-supervised Representation Learning on Vision and Beyond." This comprehensive survey categorizes and analyzes the vast ecosystem of masked modeling techniques, moving beyond the simple random masking paradigm. The survey provides a clear taxonomy of innovations across four key modules of a masked modeling framework: the masking strategy, the network architecture, the reconstruction target, and the prediction head. For our project, the **masking strategy** is the most relevant module. The survey highlights several advanced strategies that offer significant advantages over random masking:

1. **Hard Sampling & Attention-Guided Masking:** This approach posits that not all image regions are equally important. Instead of masking random patches, methods like AttMask focus on masking the most semantically salient or challenging regions of an image. The difficulty of the reconstruction task is thereby increased in a meaningful way, forcing the model to learn more robust and context-aware features. In the context of object detection, this means forcing the model to reconstruct objects from their surrounding environment, a highly relevant pre-training task. The survey shows that this can be implemented within a teacher-student framework, where the teacher's attention maps can be used to guide the masking process for the student.
2. **Contextual & Uniform Masking:** This category of techniques addresses a potential shortcoming of pure random masking, where masked patches can sometimes be clustered, leaving large contiguous areas of the image visible and making the reconstruction task trivial in those areas. Methods like UnMAE propose a more structured, grid-based masking approach. By ensuring that masked patches are more evenly distributed across the image, this strategy guarantees that the model must rely on local context for reconstruction across the entire feature map, preventing it from "getting lucky" with easy-to-reconstruct regions. This encourages the learning of more consistent and spatially aware features.

By synthesizing the targeted problem and robust framework from Zhao et al. with the advanced masking methodologies categorized by Li et al., our work aims to investigate whether a more intelligent masking strategy can further unlock the potential of the MAE branch within the MRT framework, leading to improved domain adaptation and more accurate object detection on the target domain.

# ***METHOD***

# ***EXPERIMENTS & RESULTS***

The base detector is Deformable DETR. Implementation details follow Section 5.2 of the paper (Adam optimizer, LR=2e-4, batch size=8, mask ratio=0.8, etc.).

# ***Experimental setups***

## **Dataset and Preprocessing**

The database that has been provided entangled with some serious problem. First, some boxes in the dataset were out of the dimension limitation of the images. Second, the database format was in the format of Pascal VOC but we need to the coco json format. Third, some of the labels which are in the format of xmls have different name from what the images name are, and need some modification in the filename tag.

So, for addressing the former issue, “validate\_and\_clean\_coco.py” would replace and clean up databases, and the “validate\_jsons\_box.py” would show us which boxes were wrong. These invalid annotations, which can introduce significant noise and instability during model training, were automatically detected and removed using a custom data sanitization script. The script checks the coordinates of every bounding box ([x\_min, y\_min, width, height]) against the dimensions of its source image and filters out any boxes that do not lie completely within the valid pixel area. This preprocessing step ensures that our model is trained on a clean and consistent set of ground-truth data, improving training stability and the reliability of the final evaluation metrics. **A complete log of all identified out-of-bound annotations, which were subsequently removed, is provided in Appendix A for full transparency.**

Additionally, there is “validate\_and\_clean\_coco.py” available which directly validate and clean the json files of datasets.

Also the code needs to the validate and train database version of database for each of the target, source, and test, to address this issue “split\_test\_dataset.py” is used to split each dataset to the train and test.

In the Pytorch, the labels are started from 1 but in the dataset generators like Labelimg it would start it from 0, so there is necessary to convert these labels, in order to do so the code “remap\_ids.py” is needed which convert the labels of 1 to 2 and 0 to 1.

There are multiple files with problem of different filename and the jpg file, so for resolve this, first, the xml files name would be checked, and then if there is no image available in the images folder it would search with filename tag in the xml, which this method solve the problem that some images mismatch and in some cases are matched with filename tag of xml instead of xml file name with image file type extension like jpg, jpeg, png.

One of the images in the dataset has very bad error in the loading with pIL, so by utilizing “ sanity\_check\_images.py” code, I have detect it and remove it from database json to prevent from utilizing it during testing “mi\_1Kuwaiti\_BMP-3\_APCs.jpg”, also file that exist in the “.\\data\\tanks\\images\\test\\labels\\eh1\_25.xml' has got image with name of 25.webp but there is no equivalent file for it in the images directory, so I have remove it. Exactly this happen to the “eh1\_39.jpg” file.

## **Detail of databases**

There are 2220 images available in the source data for training with 4051 annotation for 2 classes of car and tank which their annotations are available in the “train\_cocostyle\_train.json”

--- Image Dimesions ---

Average Size: 1920.00px (width) x 1080.00px (height)

Min Size: 1920px x 1080px

Max Size: 1920px x 1080px

,And the categories distributions are:

Category Name :tank Instance Count : 3985 Percentage: 99.30%

Category Name :car Instance Count : 28 Percentage: 0.70%

there are 556 images in the “train\_cocostyel\_val.json” which would be used as validate of source dataset, from it 556 images we have got annotation of 1187 of object in the total 2 category, information about images specification:

Average Size: 1920.00px (width) x 1080.00px (height)

Min Size: 1920px x 1080px

Max Size: 1920px x 1080px

And the categories distributions are:

Category Name :tank Instance Count :1183 Percentage: 99.66%

Category Name :car Instance Count :4 Percentage: 0.34%

For the test dataset, there are 641 images in the “test\_cocostyel.json” which would be used as evaluate the models final precision score, from it 641 images we have got annotation of 1145 of object in the total 2 category, and information about images specification:

Average Size: 1557.12px (width) x 972.42px (height)

Min Size: 500px x 293px

Max Size: 6000px x 4288px

And the categories distributions are:

Category Name :tank Instance Count : 950 Percentage: 82.97%

Category Name :car Instance Count : 195 Percentage: 17.03%

The target dataset which is located in the “target.json” file, it include 2870 images which contain 4324 boxes of annotations

,And information about images specification:

Average Size: 652.56px (width) x 472.05px (height)

Min Size: 256px x 150px

Max Size: 7612px x 5077px

Category Name :tank Instance Count : 950 Percentage: 82.97%

Category Name :car Instance Count : 195 Percentage: 17.03%

Part of this dataset is labeled that has been used for the validating some part of training process to get best models checkpoints to save which has better performance on the validate data of target environment, and domain. Part of these targets --roughly 50%-- are unlabeled data.

These images and their datasets are gathered from the internet resources.

## **Challenges in Replicating the Training Environment**

### **Gpu problems and cuda out of memory**

The provided training scripts, such as source\_only.sh, are configured for a distributed, multi-GPU server environment using torchrun. Adapting this setup to a more modest single-GPU workstation presented considerable hurdles, primarily concerning memory and training stability.

1. Batch Size Sensitivity and VRAM Requirements

A critical parameter for stable training of deep learning models is the batch size. We empirically observed that the model's convergence is highly sensitive to this parameter.

Local Minima with Small Batch Size: When attempting to train with a batch size of 1 to minimize memory usage, we found that the training process became unstable. The loss function would fail to decrease consistently, and the model would quickly get stuck in a local minimum. This is likely due to the noisy and high-variance gradient estimates produced by single-sample batches, which are insufficient to guide the complex model toward a good solution.

Prohibitive VRAM Consumption: To achieve stable convergence, a larger batch size was necessary. However, the Deformable DETR architecture, combined with the multiple components of the MRT framework, is extremely memory-intensive. Through experimentation on a server, we determined that a stable batch size of 6 required approximately 100 GB of VRAM. This level of memory is only available on specialized, high-end server-grade GPUs (like NVIDIA A100 80GB arrays) and is far beyond the capacity of even high-end consumer or prosumer GPUs.

2. Adapting to Consumer GPUs and Memory Swapping

Our goal was to adapt the server-based code to run on a single workstation GPU with 32 GB of VRAM. This transition highlighted a severe memory bottleneck.

The VRAM Cliff: The ~100 GB VRAM requirement for a stable batch size starkly contrasts with the 32 GB available on our target hardware. It was impossible to fit even a moderately sized batch directly into the GPU's memory.

Performance Collapse with Shared Memory: When the GPU's dedicated VRAM is exhausted, the system attempts to compensate by using shared GPU memory, which pages data to the system's main RAM. While this allows the program to run without crashing, it comes at a catastrophic performance cost. The data transfer speed between system RAM and the GPU over the PCIe bus is orders of magnitude slower than accessing the GPU's own VRAM. We observed that as soon as the memory footprint surpassed the 32 GB VRAM limit, the training speed reduced spectacularly. A single training epoch that might take an hour with sufficient VRAM could take more than a day, making experimentation and research impractical.

To overcome these challenges, we implemented gradient accumulation. This technique allows us to simulate a larger effective batch size by processing smaller "micro-batches" that fit within our 32 GB VRAM limit. We process several of these micro-batches sequentially, accumulating their gradients without updating the model's weights. Only after a specified number of accumulation steps do we perform a single weight update using the aggregated gradients. This approach enabled us to achieve a stable effective batch size while working within our hardware constraints, forming the basis of our final experimental procedure.

### **Pytorch version uncomply**

The original MRT codebase was developed using earlier versions of PyTorch and its associated CUDA libraries. In order to run the experiments in a modern deep learning environment, we encountered compilation errors related to deprecated API calls within the custom CUDA kernels that power the Deformable Attention mechanism. To ensure compatibility and future-proof the code, we made the following targeted modifications.

The primary changes were applied to the core Deformable Attention source files, including ms\_deform\_im2col\_cuda.cuh, ms\_deform\_attn\_cuda.h, and ms\_deform\_attn\_cuda.cu. The key updates were:

* **CUDA Device Check:** The function value.type().is\_cuda(), used to verify if a tensor resides on a GPU device, has been deprecated in newer PyTorch versions. We replaced all instances with the modern and direct method, value.is\_cuda().
* **Tensor Data Type Retrieval:** Similarly, the method for retrieving a tensor's data type (e.g., kFloat, kHalf) via value.type() has also been deprecated. This was updated to the current standard, value.scalar\_type().

These minor but critical changes resolve compilation warnings and errors, ensuring the project can be built and run on up-to-date systems without relying on legacy APIs.

To over come these errors, a docker image of this code created and uploaded in the dockerhub in the following link <https://hub.docker.com/r/amirreza1998/mrt-release-env>

### **Major error happening in running codes, and solutions**

Two major problems has happened during running base code of mrt

# **(950 vs. 1000): A mismatch in the number of queries being processed.**

What was mismatched? The number of input tokens (queries) fed to the transformer decoder.

Why did it happen? A selection process (like taking the "top-k" proposals) was applied to the target queries (tgt), reducing the count from a total of 1000 to 950. However, the corresponding mask (tgt\_mask) was not also filtered, leaving it at its original size of 1000.

To over come this problem, we have get minimum number of top k from both mask and main target query and remove the additional from the one that have bigger dimension, this cause no problem because these are only nominated patches for the being object that we are looking for.

# **(38 vs. 40):** A mismatch in the **spatial size (width) of feature maps**.

This error happens because the architecture of your network's **decoder** (which reconstructs the features) does not perfectly reverse the transformations of the **encoder** (which analyzes the image). This is a very common issue in convolutional networks.

The primary reason for this size change is how **convolutional and pooling layers** operate. The output size of a convolution depends on the input size, kernel size, stride, and padding.

The formula is:

This small, cumulative error from multiple layers in the encoder and decoder is why your reconstructed feature map (mae\_output) ends up being **38** pixels wide, while the original (features) was **40**. When mse\_loss tries to compare them pixel by pixel, it fails.

|  |  |
| --- | --- |
| Feature | Error 1: Query Mismatch |
| What was Mismatched? | Number of transformer queries |
| Example Dimensions | 950 vs. 1000 |
| Why it Happened? | Inconsistent filtering of queries and masks. |
| Where it Happened? | Inside the transformer decoder (DeformableTransformerDecoderMAE) |

|  |  |
| --- | --- |
| Feature | Error 2: Feature Map Mismatch |
| What was Mismatched? | Spatial width of feature maps |
| Example Dimensions | 38 vs. 40 |
| Why it Happened? | Imperfect reversal of convolutions during upsampling in the decoder. |
| Where it Happened? | During the loss calculation (criterion.py) |

To overcome this problem, I have used interpolation, which means increase the size of output or decrease size of input which the former is better option.

[Grab your reader’s attention with a great quote from the document or use this space to emphasize a key point. To place this text box anywhere on the page, just drag it.]

***Appendix***

## **Appendix A: Dataset Sanitization Log**

### **train\_cocostyle\_train.json**

Found 2220 images, 4051 annotations, and 2 categories.

Valid Category IDs are: [1, 2]

[WARNING] Annotation ID 365 (Image ID: 337) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999991837, 531.1504799999999, 39.00096, 52.65] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 417 (Image ID: 350) is partially or fully outside image boundaries.

-> BBox: [-0.0009600000000205, 617.6903400000001, 568.01088, 379.61028] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 425 (Image ID: 354) is partially or fully outside image boundaries.

-> BBox: [1823.3702400000002, 586.42002, 96.62975999999999, 51.990120000000005] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 1234 (Image ID: 1006) is partially or fully outside image boundaries.

-> BBox: [766.5590400000001, 822.08034, 711.8303999999999, 257.9202] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 1332 (Image ID: 1015) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999991837, 655.79004, 105.29088, 87.81048] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 1417 (Image ID: 1022) is partially or fully outside image boundaries.

-> BBox: [1769.3299200000001, 633.48048, 150.67008, 106.57008] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 1462 (Image ID: 1026) is partially or fully outside image boundaries.

-> BBox: [1436.37024, 733.28976, 483.63071999999994, 268.07976] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 1915 (Image ID: 1115) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999991837, 505.15974000000006, 72.90048, 23.64012] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 1966 (Image ID: 1124) is partially or fully outside image boundaries.

-> BBox: [1845.7603199999999, 499.80996, 74.24064, 24.790319999999998] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 2092 (Image ID: 1143) is partially or fully outside image boundaries.

-> BBox: [1890.20064, 574.0297200000001, 29.80032, 23.17032] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 2099 (Image ID: 1143) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999991837, 624.4803, 113.9808, 43.97004] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 2180 (Image ID: 1149) is partially or fully outside image boundaries.

-> BBox: [1672.5004800000002, 483.80003999999997, 247.49952, 121.80024] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 2205 (Image ID: 1151) is partially or fully outside image boundaries.

-> BBox: [1854.40032, 587.8602000000001, 65.60064, 33.97032] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 2332 (Image ID: 1180) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999920783, 668.46978, 390.38016, 115.79004] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 2334 (Image ID: 1181) is partially or fully outside image boundaries.

-> BBox: [-0.0009600000000205, 668.4703199999999, 385.91040000000004, 115.79975999999999] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 2389 (Image ID: 1197) is partially or fully outside image boundaries.

-> BBox: [1420.6809600000001, 584.8102799999999, 499.31904000000003, 277.89047999999997] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 2491 (Image ID: 1256) is partially or fully outside image boundaries.

-> BBox: [1688.5593600000002, 528.82038, 231.44064, 78.03972] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 2501 (Image ID: 1265) is partially or fully outside image boundaries.

-> BBox: [-0.0009600000000027364, 432.77058, 46.50048, 39.12948] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 2989 (Image ID: 1725) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999991837, 531.1504799999999, 39.00096, 52.65] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 3041 (Image ID: 1738) is partially or fully outside image boundaries.

-> BBox: [-0.0009600000000205, 617.6903400000001, 568.01088, 379.61028] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 3044 (Image ID: 1739) is partially or fully outside image boundaries.

-> BBox: [-0.0009600000000205, 617.68926, 463.9008, 424.81043999999997] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 3121 (Image ID: 1772) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999991837, 549.0504, 44.4, 37.1196] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 3720 (Image ID: 2322) is partially or fully outside image boundaries.

-> BBox: [1880.4009600000002, 586.3995, 39.6, 116.90028000000001] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 3858 (Image ID: 2394) is partially or fully outside image boundaries.

-> BBox: [766.5590400000001, 822.08034, 711.8303999999999, 257.9202] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 3956 (Image ID: 2403) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999991837, 655.79004, 105.29088, 87.81048] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 4066 (Image ID: 2412) is partially or fully outside image boundaries.

-> BBox: [1391.1504, 746.7498, 528.85056, 330.67008] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 4539 (Image ID: 2503) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999991837, 505.15974000000006, 72.90048, 23.64012] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 4590 (Image ID: 2512) is partially or fully outside image boundaries.

-> BBox: [1845.7603199999999, 499.80996, 74.24064, 24.790319999999998] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 4593 (Image ID: 2513) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999849729, 688.1797799999999, 127.34015999999998, 179.44956] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 4716 (Image ID: 2531) is partially or fully outside image boundaries.

-> BBox: [1890.20064, 574.0297200000001, 29.80032, 23.17032] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 4723 (Image ID: 2531) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999991837, 624.4803, 113.9808, 43.97004] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 4804 (Image ID: 2537) is partially or fully outside image boundaries.

-> BBox: [1672.5004800000002, 483.80003999999997, 247.49952, 121.80024] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 4829 (Image ID: 2539) is partially or fully outside image boundaries.

-> BBox: [1854.40032, 587.8602000000001, 65.60064, 33.97032] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 4956 (Image ID: 2568) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999920783, 668.46978, 390.38016, 115.79004] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 4958 (Image ID: 2569) is partially or fully outside image boundaries.

-> BBox: [-0.0009600000000205, 668.4703199999999, 385.91040000000004, 115.79975999999999] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 5013 (Image ID: 2585) is partially or fully outside image boundaries.

-> BBox: [1420.6809600000001, 584.8102799999999, 499.31904000000003, 277.89047999999997] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 5115 (Image ID: 2644) is partially or fully outside image boundaries.

-> BBox: [1688.5593600000002, 528.82038, 231.44064, 78.03972] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 5125 (Image ID: 2653) is partially or fully outside image boundaries.

-> BBox: [-0.0009600000000027364, 432.77058, 46.50048, 39.12948] vs Image Size: [1920, 1080]

### **Train\_cocostyle\_val.json:**

[WARNING] Annotation ID 420 (Image ID: 351) is partially or fully outside image boundaries.

-> BBox: [-0.0009600000000205, 617.68926, 463.9008, 424.81043999999997] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 497 (Image ID: 384) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999991837, 549.0504, 44.4, 37.1196] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 1096 (Image ID: 934) is partially or fully outside image boundaries.

-> BBox: [1880.4009600000002, 586.3995, 39.6, 116.90028000000001] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 1433 (Image ID: 1023) is partially or fully outside image boundaries.

-> BBox: [1739.1004800000003, 587.5497, 180.90048, 56.990520000000004] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 1442 (Image ID: 1024) is partially or fully outside image boundaries.

-> BBox: [1391.1504, 746.7498, 528.85056, 330.67008] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 1969 (Image ID: 1125) is partially or fully outside image boundaries.

-> BBox: [-0.0009599999999849729, 688.1797799999999, 127.34015999999998, 179.44956] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 3049 (Image ID: 1742) is partially or fully outside image boundaries.

-> BBox: [1823.3702400000002, 586.42002, 96.62975999999999, 51.990120000000005] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 4041 (Image ID: 2410) is partially or fully outside image boundaries.

-> BBox: [1769.3299200000001, 633.48048, 150.67008, 106.57008] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 4057 (Image ID: 2411) is partially or fully outside image boundaries.

-> BBox: [1739.1004800000003, 587.5497, 180.90048, 56.990520000000004] vs Image Size: [1920, 1080]

[WARNING] Annotation ID 4086 (Image ID: 2414) is partially or fully outside image boundaries.

-> BBox: [1436.37024, 733.28976, 483.63071999999994, 268.07976] vs Image Size: [1920, 1080]