# Progress Report: DeepStream-MMDeploy Integration

## 1. Introduction

This report summarizes the progress made on integrating DeepStream with MMDeploy for video inference and post-processing. The objective is to perform inference using a custom YOLO-based model and process results efficiently using MMDeploy's post-processing pipeline.

**2. Initial DeepStream Setup**

Initially, DeepStream version 7.1 was used, but it was discovered that the CUDA and TensorRT versions were incompatible with the current environment. As a result, the DeepStream version was downgraded to 6.4 to match the CUDA and TensorRT versions required for smooth execution.

**3.** [**Engine File Generation and Model Inference**](NMS%20plugin%20error.docx)

The initial steps involved generating the engine file using the DeepStream model configuration and inference configuration provided in the config.txt file. Despite following the ReadMe documentation, the engine file generation encountered multiple errors. This led to the creation of a separate Conda environment with PyTorch 1.9 to ensure compatibility. Once the engine file was generated successfully, issues were encountered with MMDeploy integration inside the DeepStream container.

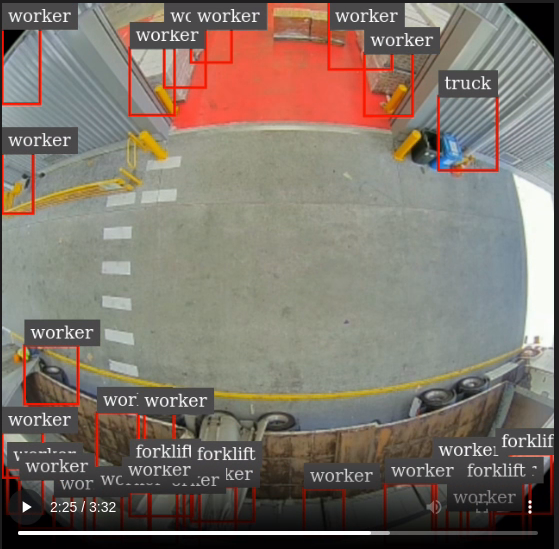
## 2. Initial Configuration and Pipeline Setup

I began by using DeepStream's default pipeline and modifying the `config\_infer\_primary.txt` file to integrate the custom YOLO model. The inference pipeline successfully detected multiple objects and displayed bounding boxes; however, I encountered an issue of false detections and inconsistent bounding boxes.



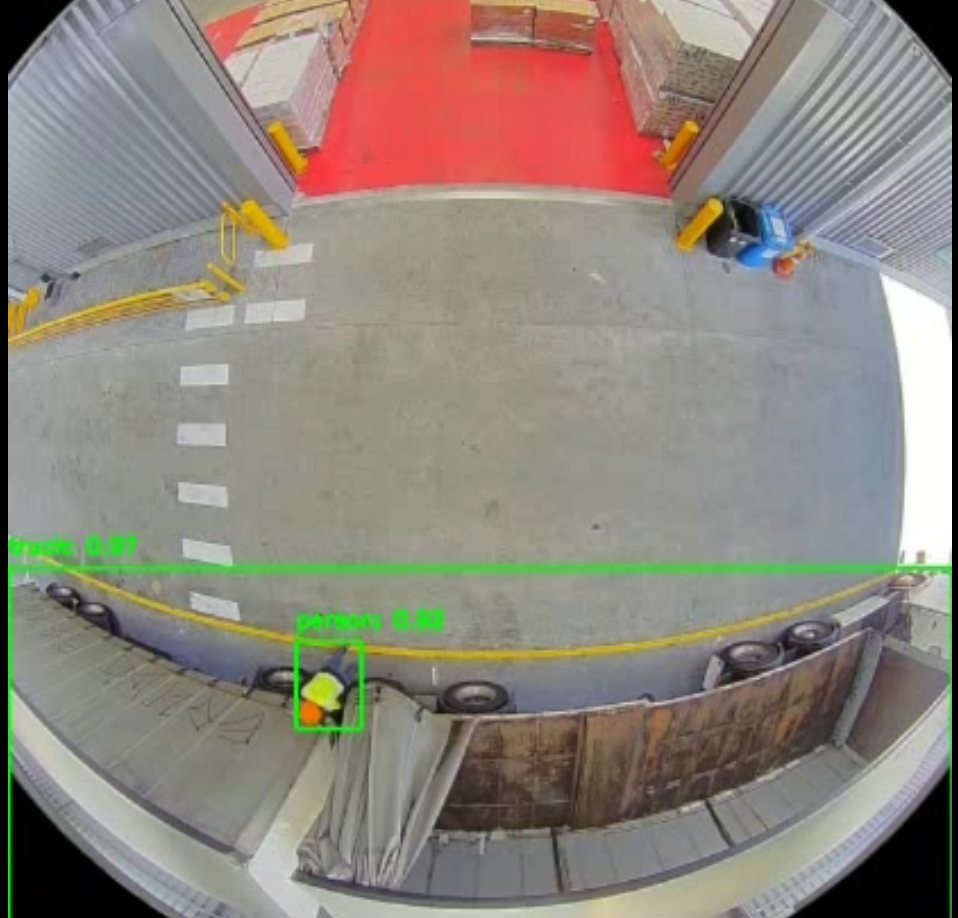
## 3. Modifying Custom Parser

To improve bounding box accuracy, I modified the custom parser file to refine post-processing. Despite these modifications, false detections persisted, indicating that the issue may be with the engine file or post-processing logic.



## 4. Engine File Testing

I extracted and tested the engine file separately using Python inference. The results showed accurate object detection when used with Python inference, suggesting that the issue was specific to the DeepStream post-processing pipeline.

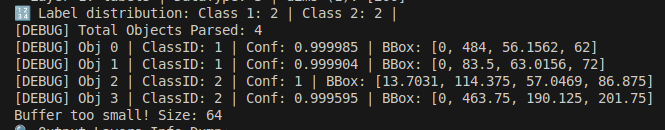


## 5. Regenerating Engine File with Post-Processing

I regenerated the engine file by incorporating custom post-processing in `deploy\_trt.py`. However, the issue of multiple detections and false positives still persisted, indicating that DeepStream's inference and post-processing were not correctly aligned.

## 6. Shifting to MMDeploy Post-Processing Hook

To resolve the issue, I decided to integrate MMDeploy’s post-processing directly into the DeepStream pipeline. I successfully hooked the MMDeploy post-processing, and the object detection results improved significantly with correct bounding box outputs.

However, a new issue emerged during the video saving process.

## 7. Buffer Size Mismatch and Issues with Saving

While trying to save the annotated output video, a buffer size mismatch error occurred. The error indicated that the mapped buffer size was significantly smaller (64 bytes) than the expected buffer size. This suggested that the buffer mapping method was not appropriate for NVMM buffers used by DeepStream.

## 8. Attempt to Use pyds for NVMM Buffers

To handle NVMM buffers correctly, I attempted to use `pyds.get\_nvds\_buf\_surface()` to map and convert NVMM buffers to a NumPy array. However, I encountered multiple errors during the installation and configuration of `pyds`. Despite building the bindings and installing `pyds`, the module was still not correctly recognized, leading to further delays in resolving the buffer size mismatch.

## 9. Key Challenges and Roadblocks

- False detections and multiple bounding boxes due to incorrect post-processing.

- Buffer size mismatch when trying to save processed frames.

- Inconsistent behavior of `pyds` when attempting to map NVMM buffers.

- Compilation and version mismatch while attempting to build and install `pyds`.

## 10. Next Steps

- Investigate alternative buffer mapping approaches to avoid size mismatches.

- Re-evaluate the pipeline and validate post-processing consistency.

- Debug and resolve `pyds` related issues to correctly map NVMM buffers.

## 11. Conclusion

Despite facing multiple challenges during the integration of DeepStream with MMDeploy, significant progress has been made in refining post-processing and object detection accuracy. The remaining issues, primarily related to buffer mapping and frame saving, need to be addressed to achieve end-to-end successful deployment.