# DeepStream Pipeline with Custom TRTBatchedNMS Plugin - Setup & Usage Guide

## Overview

This document describes the step-by-step process to set up and run a DeepStream 6.4 pipeline using a custom-built TRTBatchedNMS plugin from MMDeploy. The configuration allows TensorRT-based inference using custom YOLO models and removes the need for postprocessing parsers by using optimized TRT plugins.

## Prerequisites

- Docker with GPU support  
- NVIDIA GPU Drivers, CUDA, cuDNN installed  
- DeepStream 6.4 compatible base image  
- MMDeploy source code  
- Your trained YOLO model exported as a TensorRT engine

## 1. Build Custom TRTBatchedNMS Plugin

Follow these steps inside your Docker container:

# Clone MMDeploy and prepare build directory  
cd /workspace/mmdeploy  
rm -rf build  
mkdir build && cd build  
  
# (Optional) Prepare fake TensorRT SDK if not available  
mkdir -p /workspace/tensorrt\_fake/include  
cp /usr/include/x86\_64-linux-gnu/NvInfer\*.h /workspace/tensorrt\_fake/include  
mkdir -p /workspace/tensorrt\_fake/lib  
cp /usr/lib/x86\_64-linux-gnu/libnvinfer\* /workspace/tensorrt\_fake/lib  
  
# Run CMake  
cmake .. \  
 -DMMDEPLOY\_TARGET\_BACKENDS=trt \  
 -DTENSORRT\_DIR=/workspace/tensorrt\_fake \  
 -DMMDEPLOY\_BUILD\_SDK=OFF \  
 -DMMDEPLOY\_BUILD\_SDK\_PYTHON\_API=OFF  
  
# Build the plugin  
make -j$(nproc)  
  
# Output plugin will be here:  
/workspace/mmdeploy/build/lib/libmmdeploy\_tensorrt\_ops.so

## 2. Add Plugin to DeepStream Runtime

Instead of using deprecated plugin-library key, use LD\_PRELOAD before running:

export LD\_PRELOAD=/workspace/mmdeploy/build/lib/libmmdeploy\_tensorrt\_ops.so  
deepstream-app -c /workspace/configs/deepstream\_app\_config.txt

## 3. Sample pgie\_config.txt

[property]  
gpu-id=0  
model-engine-file=/workspace/training-output/exports/end2end\_fp16\_4gb.engine  
labelfile-path=/workspace/configs/labels.txt  
batch-size=1  
network-mode=2  
num-detected-classes=3  
gie-unique-id=1  
network-type=100  
cluster-mode=2

## 4. Save Docker Container (with Plugin)

# Add LD\_PRELOAD to bashrc inside container  
echo "export LD\_PRELOAD=/workspace/mmdeploy/build/lib/libmmdeploy\_tensorrt\_ops.so" >> ~/.bashrc  
  
# Exit container and save image  
exit  
  
# On host system  
docker commit <container\_id> deepstream\_trt\_ready:v1

## 5. Reuse Saved Container

docker run -it --gpus all \  
 --name deepstream\_auto\_ready \  
 --volume /home/prakash/deepstream6.4:/workspace \  
 deepstream\_trt\_ready:v1 \  
 bash  
  
# Then inside container  
deepstream-app -c /workspace/configs/deepstream\_app\_config.txt

## 6. Important Files to Keep

- /workspace/mmdeploy/build/lib/libmmdeploy\_tensorrt\_ops.so  
- /workspace/configs/deepstream\_app\_config.txt  
- /workspace/configs/pgie\_config.txt  
- /workspace/training-output/exports/end2end\_fp16\_4gb.engine  
- /workspace/configs/labels.txt  
- /workspace/deepstream-videos/\*.avi