YOLOv9 QAT for TensorRT Detection / Segmentation

This repository contains an implementation of YOLOv9 with Quantization-Aware Training (QAT), specifically designed for deployment on platforms utilizing TensorRT for hardware-accelerated inference.
This implementation aims to provide an efficient, low-latency version of YOLOv9 for real-time detection applications.
If you do not intend to deploy your model using TensorRT, it is recommended not to proceed with this implementation.

- The files in this repository represent a patch that adds QAT functionality to the original YOLOv9 repository.
- This patch is intended to be applied to the main YOLOv9 repository to incorporate the ability to train with QAT.
- The implementation is optimized to work efficiently with TensorRT, an inference library that leverages hardware acceleration to enhance inference performance.
- Users interested in implementing object detection using YOLOv9 with QAT on TensorRT platforms can benefit from this repository as it provides a ready-to-use solution.

We use TensorRT's pytorch quntization tool to finetune training QAT yolov9 from the pre-trained weight, then export the model to onnx and deploy it with TensorRT. The accuray and performance can be found in below table.

For those who are not familiar with QAT, I highly recommend watching this video:
Quantization explained with PyTorch - Post-Training Quantization, Quantization-Aware Training

Getting started (工欲善其事，必先利其器)

For getting started, needs some steps.

Git Download

Git

- Settings：git_command

Download Archive

yolov9

```
git clone https://github.com/WongKinYiu/yolov9.git
```

```
git status
```

```
git pull
```

yolov9-qat

```
git clone https://github.com/levipereira/yolov9-qat.git
```

```
git status
```

```
git pull
```

CUDA / cudnn Info

Check the info and choose the best fit to your device.

```
nvidia-smi
```

Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow-2.14.0	3.9-3.11	Clang 16.0.0	Bazel 6.1.0	8.7	11.8
tensorflow-2.13.0	3.8-3.11	Clang 16.0.0	Bazel 5.3.0	8.6	11.8
tensorflow-2.12.0	3.8-3.11	GCC 9.3.1	Bazel 5.3.0	8.6	11.8
tensorflow-2.11.0	3.7-3.10	GCC 9.3.1	Bazel 5.3.0	8.1	11.2
tensorflow-2.10.0	3.7-3.10	GCC 9.3.1	Bazel 5.1.1	8.1	11.2
tensorflow-2.9.0	3.7-3.10	GCC 9.3.1	Bazel 5.0.0	8.1	11.2
tensorflow-2.8.0	3.7-3.10	GCC 7.3.1	Bazel 4.2.1	8.1	11.2
tensorflow-2.7.0	3.7-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.6.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.5.0	3.6-3.9	GCC 7.3.1	Bazel 3.7.2	8.1	11.2
tensorflow-2.4.0	3.6-3.8	GCC 7.3.1	Bazel 3.1.0	8.0	11.0
tensorflow-2.3.0	3.5-3.8	GCC 7.3.1	Bazel 3.1.0	7.6	10.1
tensorflow-2.2.0	3.5-3.8	GCC 7.3.1	Bazel 2.0.0	7.6	10.1

Version	Python version	Compiler	Build tools	cuDNN	CUDA
tensorflow-2.1.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.27.1	7.6	10.1
tensorflow-2.0.0	2.7, 3.3-3.7	GCC 7.3.1	Bazel 0.26.1	7.4	10.0

CUDA cudnn pytorch TensorRT

Implementation Environment

Windows11 + WSL_Ubuntu-20.04(LTS)
Windows11 / WSL_Ubuntu-20.04-LTS

Windows11

- Environment
  - python 3.10.16
  - CUDA 12.1
  - cudnn 8.8
  - pytorch

```
pip3 install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cu121
```

  - TensorRT 10.1.0

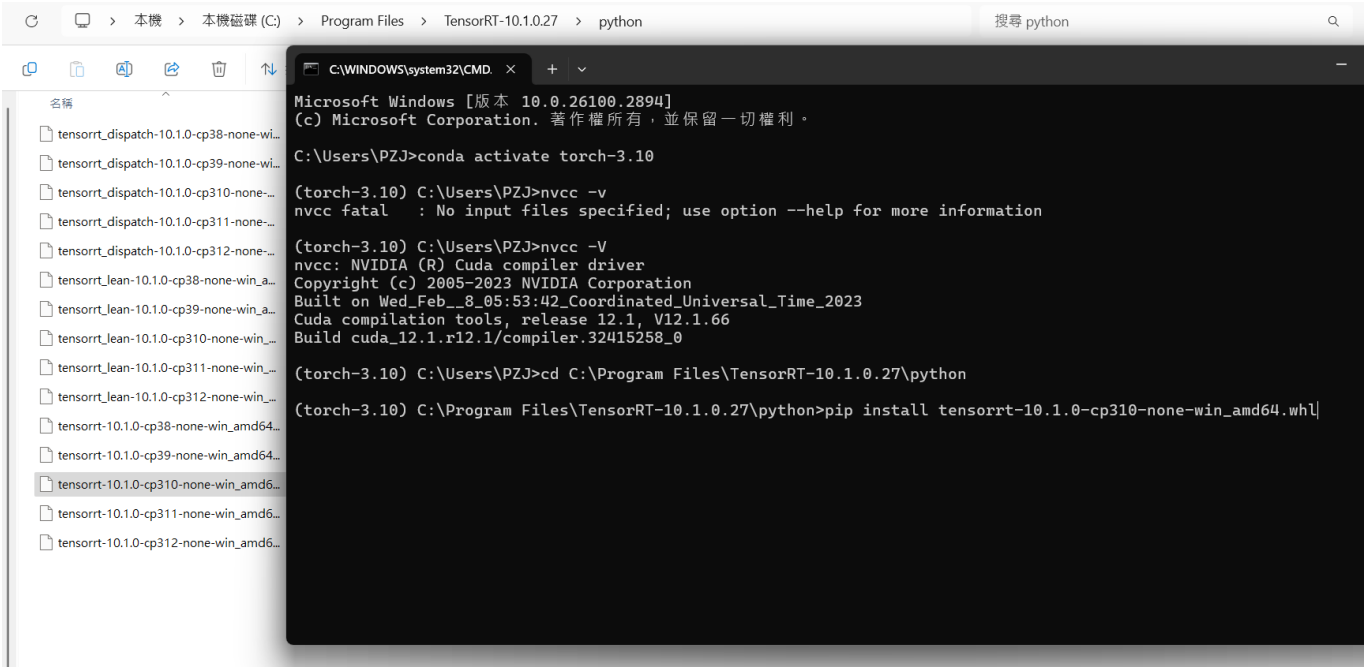
TensorRT Installation Guide

1.Extract the downloaded files.

2.Set the necessary environment variables in the installation directory.

3.Add the `bin` and `lib` folders to the `PATH`.

4.Python_Building



WSL_Ubuntu-20.04(LTS)

- Environment
  - python 3.10.16
  - CUDA 11.8
  - cudnn 8.6
  - pytorch

```
pip3 install torch torchvision torchaudio --index-url
https://download.pytorch.org/whl/cu118
```

  - TensorRT 10.1.0

TensorRT Installation Guide

- Main System Debian Installation
- Your Executing Environment (ex：torch-3.10)

1. NVIDIA TensorRT 10.x Download click the NVIDIA TensorRT License Agreement and find the archive that best fit your OS and devices.

Make sure you have activated the environment
Follow these steps to install TensorRT on WSL Ubuntu 20.04.

2. Install the Local TensorRT Repository

First, navigate to the directory where the TensorRT `.deb` package is located and install it using `dpkg`:

```
cd /mnt/c/Users/PZJ/Downloads
sudo dpkg -i nv-tensorrt-local-repo-ubuntu2004-10.1.0-cuda-11.8_1.0-1_amd64.deb
```

3. Add GPG Key and Update Package List

After installing the local repository package, copy the GPG key to the appropriate directory and update the package list:

```
sudo cp /var/nv-tensorrt-local-repo-ubuntu2004-10.1.0-cuda-11.8/*.pub
/usr/share/keyrings/
sudo apt-get update
```

4. Install TensorRT and Dependencies

To install TensorRT and its necessary dependencies, run the following commands:

```
sudo apt-get install -y tensorrt
sudo apt-get install -y python3-libnvinfer-dev
sudo apt-get install -y uff-converter-tf
sudo apt-get install -y onnx-graphsurgeon
```

5. Verify Installation

To check if TensorRT has been successfully installed, use the following command:

```
dpkg -l | grep TensorRT
```

6. Test TensorRT Execution

To verify that TensorRT is functioning correctly, you can run an inference test using `trtexec` with an ONNX model:

```
trtexec --onnx=<your_model>.onnx
```

Notice

Please make sure that the TensorRT version built into the system is consistent with the version installed in Python.

Make sure environment variables are set correctly.

```
nano ~/.bashrc
```

Edit in nano

```
source ~/.bashrc
```

```
# <<< conda initialize <<<

# export PATH="/mnt/c/Users/PZJ/anaconda3/bin:$PATH"  # commented out by conda initialize
export PATH=/usr/local/cuda-11.8/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda-11.8/lib64:$LD_LIBRARY_PATH

export PATH=/usr/src/tensorrt/bin:$PATH
export LD_LIBRARY_PATH=/usr/src/tensorrt/lib:$LD_LIBRARY_PATH

# for libnvinfer_plugin.so.10
export LD_LIBRARY_PATH=/home/user_pzj/anaconda3/envs/torch-3.10/lib/python3.10/site-packages/tensorrt_libs:$LD_LIBRARY_PATH

export PYTHONPATH=/home/user_pzj/anaconda3/envs/torch-3.10/lib/python3.10/site-packages:$PYTHONPATH
```

Verify the Version

```
(torch-3.10) user_pzj@DESKTOP-20USVPU:/mnt/c/Users/PZJ$ dpkg-query -W tensorrt
tensorrt        10.1.0.27-1+cuda11.8
(torch-3.10) user_pzj@DESKTOP-20USVPU:/mnt/c/Users/PZJ$ dpkg-query -W "*nvinfer*"
libnvinfer-bin  10.1.0.27-1+cuda11.8
libnvinfer-dev  10.1.0.27-1+cuda11.8
libnvinfer-dev-cross-amd64
libnvinfer-dispatch-dev 10.1.0.27-1+cuda11.8
libnvinfer-dispatch-dev-cross-amd64
libnvinfer-dispatch10   10.1.0.27-1+cuda11.8
libnvinfer-doc
libnvinfer-headers-dev  10.1.0.27-1+cuda11.8
libnvinfer-headers-plugin-dev   10.1.0.27-1+cuda11.8
libnvinfer-lean-dev     10.1.0.27-1+cuda11.8
libnvinfer-lean-dev-cross-amd64
libnvinfer-lean10       10.1.0.27-1+cuda11.8
libnvinfer-plugin-dev   10.1.0.27-1+cuda11.8
libnvinfer-plugin-dev-cross-amd64
libnvinfer-plugin10     10.1.0.27-1+cuda11.8
libnvinfer-samples      10.1.0.27-1+cuda11.8
libnvinfer-vc-plugin-dev        10.1.0.27-1+cuda11.8
libnvinfer-vc-plugin-dev-cross-amd64
libnvinfer-vc-plugin10  10.1.0.27-1+cuda11.8
libnvinfer10    10.1.0.27-1+cuda11.8
python3-libnvinfer      10.1.0.27-1+cuda11.8
python3-libnvinfer-dev  10.1.0.27-1+cuda11.8
python3-libnvinfer-dispatch     10.1.0.27-1+cuda11.8
python3-libnvinfer-lean 10.1.0.27-1+cuda11.8
(torch-3.10) user_pzj@DESKTOP-20USVPU:/mnt/c/Users/PZJ$ pip show tensorrt
Name: tensorrt
Version: 10.1.0
Summary: TensorRT Metapackage
Home-page: https://developer.nvidia.com/tensorrt
Author: NVIDIA Corporation
Author-email:
License: Proprietary
Location: /home/user_pzj/anaconda3/envs/torch-3.10/lib/python3.10/site-packages
Requires: tensorrt-cu12
Required-by: nvidia-tensorrt
(torch-3.10) user_pzj@DESKTOP-20USVPU:/mnt/c/Users/PZJ$ python -c "import tensorrt as trt; print(trt.__version__)"
10.8.0.43
```

PS：The command `python -c "import tensorrt as trt; print(trt.__version__)"` -> (10.8.0.43) version here refers to the tensorrt-cu12 version, not the real Python TensorRT version. Among them tensorrt-cu12 is the underlying C++ package.

If didn't obey may cause：`Attributeerror: 'nonetype' object has no attribute 'num_io_tensors'`

execute_qat

Other Testing

Verify CUDA

```
nvcc -V
```

or

```
deviceQuery
```

```
Device 0: "NVIDIA GeForce RTX 4050 Laptop GPU"
  CUDA Driver Version / Runtime Version          12.7 / 11.8
  CUDA Capability Major/Minor version number:    8.9
  Total amount of global memory:                 6140 MBytes (6438780928 bytes)
MapSMtoCores for SM 8.9 is undefined.  Default to use 128 Cores/SM
MapSMtoCores for SM 8.9 is undefined.  Default to use 128 Cores/SM
  (20) Multiprocessors, (128) CUDA Cores/MP:     2560 CUDA Cores
  GPU Max Clock rate:                            2355 MHz (2.36 GHz)
  Memory Clock rate:                             7825 Mhz
  Memory Bus Width:                              96-bit
  L2 Cache Size:                                 25165824 bytes
  Maximum Texture Dimension Size (x,y,z)         1D=(131072), 2D=(131072, 65536), 3D=(16384, 16384, 16384)
  Maximum Layered 1D Texture Size, (num) layers  1D=(32768), 2048 layers
  Maximum Layered 2D Texture Size, (num) layers  2D=(32768, 32768), 2048 layers
  Total amount of constant memory:               65536 bytes
  Total amount of shared memory per block:       49152 bytes
  Total number of registers available per block: 65536
  Warp size:                                     32
  Maximum number of threads per multiprocessor:  1536
  Maximum number of threads per block:           1024
  Max dimension size of a thread block (x,y,z): (1024, 1024, 64)
  Max dimension size of a grid size    (x,y,z): (2147483647, 65535, 65535)
  Maximum memory pitch:                          2147483647 bytes
  Texture alignment:                             512 bytes
  Concurrent copy and kernel execution:          Yes with 1 copy engine(s)
  Run time limit on kernels:                     Yes
  Integrated GPU sharing Host Memory:            No
  Support host page-locked memory mapping:       Yes
  Alignment requirement for Surfaces:            Yes
  Device has ECC support:                        Disabled
  Device supports Unified Addressing (UVA):      Yes
  Device supports Compute Preemption:            Yes
  Supports Cooperative Kernel Launch:            Yes
  Supports MultiDevice Co-op Kernel Launch:      No
  Device PCI Domain ID / Bus ID / location ID:   0 / 1 / 0
  Compute Mode:
     < Default (multiple host threads can use ::cudaSetDevice() with device simultaneously) >

deviceQuery, CUDA Driver = CUDART, CUDA Driver Version = 12.7, CUDA Runtime Version = 11.8, NumDevs = 1, Device0 = NVIDIA GeForce RTX 4050 Laptop GPU
Result = PASS
```

Verify cudnn

```
cat /usr/include/cudnn_version.h | grep CUDNN_MAJOR -A 2
```

or

```
cat /usr/include/cudnn_version.h | grep CUDNN_MAJOR -A 2
```

```
(torch-3.10) user_pzj@DESKTOP-20USVPU:~$ cat /usr/include/cudnn_version.h | grep CUDNN_MAJOR -A 2
#define CUDNN_MAJOR 8
#define CUDNN_MINOR 6
#define CUDNN_PATCHLEVEL 0
--
#define CUDNN_VERSION (CUDNN_MAJOR * 1000 + CUDNN_MINOR * 100 + CUDNN_PATCHLEVEL)

/* cannot use constexpr here since this is a C-only file */
```