

# Execute\_QAT

---

## Execute Record

[Record](#)

## Steps :

---

### Train the Model Using [Training Session](#):

#### Data preparation

```
bash scripts/get_coco.sh
```

- Download MS COCO dataset images ([train](#), [val](#), [test](#)) and [labels](#). If you have previously used a different version of YOLO, we strongly recommend that you delete [train2017.cache](#) and [val2017.cache](#) files, and redownload [labels](#)

#### Single GPU training

```
# train yolov9 models
!python train_dual.py --workers 8 --device 0 --batch 16 --data data/coco.yaml --img 640 --cfg models/detect/yolov9-c.yaml --weights yolov9-c.pt --name gelan-c --hyp hyp.scratch-high.yaml --min-items 0 --epochs 100 --close-mosaic 15 --patience 10

# train gelan models
!python train.py --workers 8 --device 0 --batch 16 --data data/coco.yaml --img 640 --cfg models/detect/gelan-c.yaml --weights gelan-c.pt --name gelan-c --hyp hyp.scratch-high.yaml --min-items 0 --epochs 100 --close-mosaic 15 --patience 10
```

#### Multiple GPU training

```
# train yolov9 models
!python -m torch.distributed.launch --nproc_per_node 8 --master_port 9527 train_dual.py --workers 8 --device 0,1,2,3,4,5,6,7 --sync-bn --batch 128 --data data/coco.yaml --img 640 --cfg models/detect/yolov9-c.yaml --weights '' --name yolov9-c --hyp hyp.scratch-high.yaml --min-items 0 --epochs 500 --close-mosaic 15

# train gelan models
!python -m torch.distributed.launch --nproc_per_node 4 --master_port 9527 train.py --workers 8 --device 0,1,2,3 --sync-bn --batch 128 --data data/coco.yaml --img 640 --cfg models/detect/gelan-c.yaml --weights '' --name gelan-c --hyp hyp.scratch-high.yaml --min-items 0 --epochs 500 --close-mosaic 15
```

## Reparameterize the Model [reparameterization.py](#):

ex : FP32 -> FP16

## Proceed with Quantization

```
!python qat.py quantize --weights "C:\Users\PZJ\Desktop\yolov9\yolov9-  
main_measure\yolov9-main\yolov9-c7-converted.pt" --data data/coco.yaml --hyp  
./data/hyps/hyp.scratch-high.yaml --name qat_yolov9_reparameterize --exist-ok
```

## **Eval Pytorch / Eval TensorRT (Will generate the val\_trt file if succeed) :**

### Evaluate using Pytorch

```
!python qat.py eval --weights "C:\Users\PZJ\Desktop\yolov9\yolov9-  
main_measure\yolov9-main\yolov9-c7-converted.pt" --name  
eval_qat_yolov9_reparameterize
```

### Evaluate using TensorRT

**Make sure to generate the .onnx through [Export ONNX](#) first !!**

```
./scripts/val_trt.sh  
runs/qat/qat_yolov9_reparameterize_SiLU/weights/qat_best_yolov9-c7-converted.pt  
data/coco.yaml 640
```

### Generate TensorRT Profiling and SVG image

```
./scripts/val_trt.sh  
runs/qat/qat_yolov9_reparameterize_SiLU/weights/qat_best_yolov9-c7-converted.pt  
data/coco.yaml 640 --generate-graph
```

## Export ONNX

## Export ONNX Model without End2End

```
!python export_qat.py --weights "C:\Users\PZJ\Desktop\yolov9\yolov9-  
main_measure\yolov9-  
main\runs\qat\qat_yolov9_reparameterize_SiLU\weights\qat_best_yolov9-c7-  
converted.pt" --include onnx --dynamic --simplify --inplace
```

## Export ONNX Model End2End

```
!python export_qat.py --weights "C:\Users\PZJ\Desktop\yolov9\yolov9-  
main_measure\yolov9-  
main\runs\qat\qat_yolov9_reparameterize_SiLU\weights\qat_best_yolov9-c7-  
converted.pt" --include onnx_end2end
```

## Deployment with Tensorrt

```
/usr/src/tensorrt/bin/trtexec \  
--onnx=runs/qat/qat_yolov9_reparameterize_SiLU/weights/qat_best_yolov9-c7-  
converted.onnx \  
--int8 --fp16 \  
--useCudaGraph \  
--minShapes=images:1x3x640x640 \  
--optShapes=images:4x3x640x640 \  
--maxShapes=images:8x3x640x640 \  
--saveEngine=runs/qat/qat_yolov9_reparameterize_SiLU/weights/qat_best_yolov9-c7-  
converted.engine
```

## Benchmark

---

Note: To test FP16 Models (such as Origin) remove flag `--int8`

```
# Set variable batch_size and model_path_no_ext  
# Mixed Precision  
export batch_size=4  
export  
filepath_no_ext=runs/qat/qat_yolov9_reparameterize_SiLU/weights/qat_best_yolov9-  
c7-converted  
trtexec \  
--onnx=${filepath_no_ext}.onnx \  
--fp16 \  
--int8 \  
--saveEngine=${filepath_no_ext}_bs4_MP_SiLU.engine \  
--timingCacheFile=${filepath_no_ext}.engine.timing.cache \  
--warmUp=500 \  
--duration=10 \  
--useCudaGraph \  
--useSpinWait \  
--noDataTransfers \  
--minShapes=images:1x3x640x640 \  
--optShapes=images:${batch_size}x3x640x640 \  
--maxShapes=images:${batch_size}x3x640x640
```

# Use Timing Comparison

功能	val_trt.sh 指令	trtexec 指令
模型輸入	ONNX 模型 (qat_best_best.onnx)	ONNX 模型 (qat_best_yolov9-converted.onnx)
功能	驗證流程，可能包含推理、效能測試、精度計算	轉換模型為 TensorRT 引擎，並進行效能測試
數據集設定	使用 data/coco.yaml	不涉及數據集驗證
動態輸入形狀	可能默認封裝設定	明確指定 minShapes, optShapes, maxShapes
精度選項	未顯式提及 FP16/INT8 模式	明確指定 --fp16, --int8
額外功能	生成計算圖 (--generate-graph)	無此功能
執行細節	封裝於腳本中，細節未展開	直接執行 trtexec，所有參數顯式定義

## Result

### yolov9-QAT(SiLU)

- SiLU\_MP
- SiLU\_INT8
- SiLU\_FP16

### yolov9-QAT(ReLU)

- ReLU\_MP