Report - Object Detection for HW3

Experiment Setup

- Model: wraps HuggingFace RTDetrForObjectDetection (PekingU/rtdetr_r18vd by default) and decoder self-attention replaced by custom Grouped Query Attention, which can set numbers of k head and the dimension.

- Preprocess: RTDetrImageProcessor handles resizing/ normalization; collate_fn pads and builds pixel_mask; boxes kept COCO style then converted to cxcywh internally by processor.

- Hyperparams (defaults): batch_size 8, lr 1e-4, weight_decay 1e-4, epochs 500, eval_interval 1, warmup via steps (–warmup_steps) or epochs (–warmup_epochs, default 2) using linear LambdaLR; num_kv_heads 4; optional hidden_dim_GQA to set GQA embed dim; device auto cuda if available.

- Training strategy: AdamW; full RT-DETR loss (Hungarian + bbox + cls) from model outputs; per-iteration loss logging to TensorBoard; eval on val set each eval_interval; save best checkpoint when mAP improves.

Code Brief

- main.py: argument parsing; train loop with optimizer/scheduler, TensorBoard logging, periodic eval via evaluate; evaluate_only loads checkpoint and runs val/evaluation/export detections JSON.

- ds.py: GTACarDataset loads images/annotations, filters crowds, runs processor to produce tensors; collate_fn pads batch and returns masks plus ids/names.

- model.py: wraps RT-DETR, and replaces decoder self-attn with GQA; exposes processor; forward delegates to wrapped model.

- util.py: converts COCO annotations/detections to Pascal VOC-style txt for external metrics.

GQA Implementation

- GroupedQueryAttention uses num_q_heads queries and fewer num_kv_heads keys/values; K/V heads are repeated (repeat_interleave) to match Q groups.

- Linear projections for Q/K/V/O; optional pre/post projections when input/output dims differ from embed_dim; scaled dot-product attention with optional mask; dropout on attention.

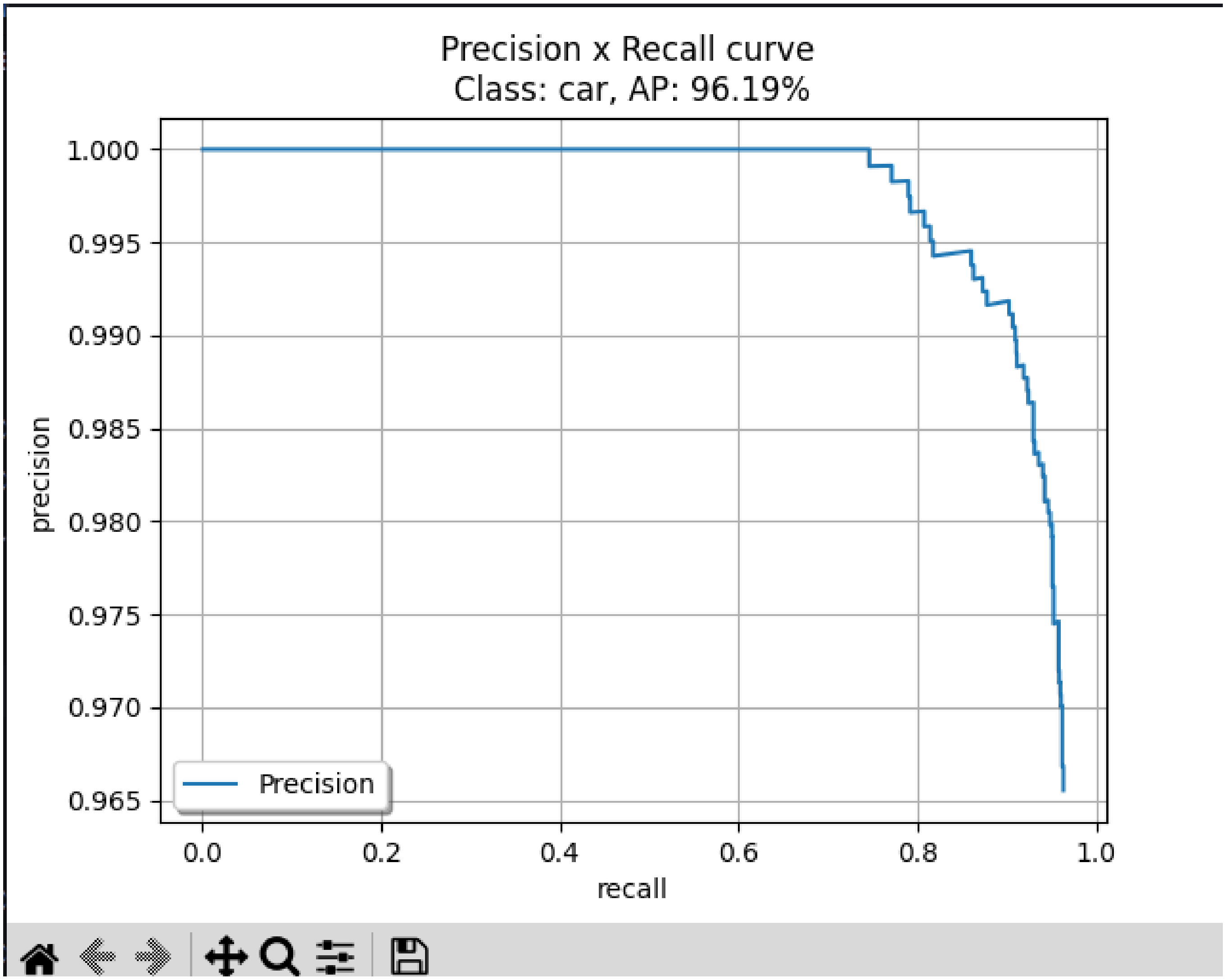
- Decoder modification: each decoder layer' s self_attn replaced with GQA configured by –num_kv_heads, –hidden_dim_GQA (defaults to model hidden size), preserving decoder input/output dims via pre/post projections.

Training / Inference Commands

- Train from scratch/defaults: python main.py –root_dir ./hw3_dataset –batch_size 2 –lr 1e-4 –weight_decay 1e-4 –num_epochs 10 –num_kv_heads 4 –log_dir runs/
- Train with warmup and custom dims: python main.py –root_dir ./hw3_dataset –warmup_epochs 2 –hidden_dim_GQA 128 –num_kv_heads 2
- Export detections JSON during eval: python main.py –root_dir ./hw3_dataset –skip_train –checkpoint_path ./run/ckpt_epoch_best10.pth –detections_output ./runs/detections_val.json

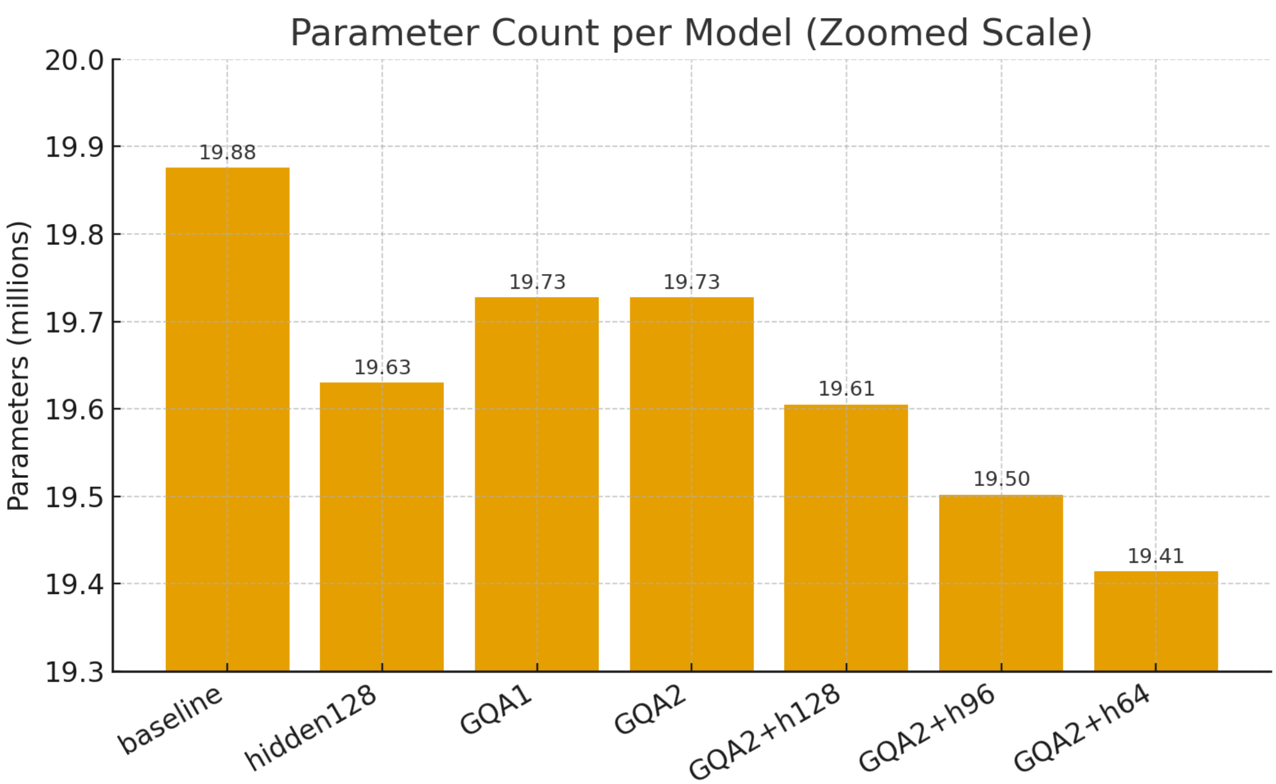
Experiment

- Validation result screenshot With number of k equals to 2 and the dimension of the hidden layer is 96, it get the 96.19% mAP. However, if the dimension of hidden layer is 64, the mAP would be 0.



- Comparision differernt number of KV head / group size or other method you used
  - Record validation result and number of parameters

    `GQA` describe swapping each decoder self-attention block with GroupedQueryAttention, which keeps the standard number of query heads but shares a smaller set of key/value heads (the `k` in the table, e.g., 4) that are repeated across queries. GQA 2 is mean the number of key/value heads is 2 compared with the baseline which use self attention with 8 key/value heads. `Hidden` rows refer to setting hidden_dim_GQA, so every GQA block first projects inputs into that reduced internal embedding size (128/96/64) before projecting back out, shrinking parameters inside the attention module itself without touching the rest of the transformer.



Model	# Params
baseline	19,875,796
hidden128	19,630,036

Model	# Params
GQA1	19,727,764
GQA2	19,727,764
GQA2 + hidden128	19,605,268
GQA2 + hidden96	19,502,020
GQA2 + hidden64	19,414,132