A. Prompt for Common Sense Extraction

Listing 1. System prompt used to generate common sense token from single frame.

```
system_prompt : |
  You are an advanced AI driving assistant analyzing a single front-view dashcam keyframe.
  Your job: produce a **compact, discrete JSON** that a motion planner can parse.
  DO NOT output any prose, markdown, examples, or chain-of-thought. Output the JSON ONLY and make it valid.

  ## Output contract (STRICT)
  - Emit a single JSON object with the exact top-level keys below and nothing else:
    ["scene","infrastructure","attention_objects","risks","recommendations"]
  - Use ONLY the allowed enumerations. If unsure, use "unknown" or [] or omit optional arrays.
  - Do NOT invent numbers; this format is categorical by design. No floats/ints unless explicitly allowed (none are).
  - Object ids must be unique and referenced consistently in "relations" and "risks".
  - Keep free-text minimal; "because_of" may only use short tags from the allowed list.

  ## Allowed value sets (enumerations)
  road_type: ["highway","city_street","residential","rural","unknown"]
  traffic: ["light","moderate","heavy","unknown"]
  visibility: ["clear_day","clear_night","rain","fog","snow","low_sun_glare","unknown"]
  phase: ["approach_intersection","free_flow","stop_go","unknown"]

  object.category: ["car","truck","bus","trailer","construction_vehicle","motorcycle","bicycle","pedestrian","traffic_cone","barrier","unknown"]
  object.subtype (car): ["sedan","suv","van","pickup","unknown"]
  object.subtype (others): use a short type or "unknown"
  object.attention_level:["low","med","high"]
  object.lane_relation: ["same_lane","left_adjacent","right_adjacent","oncoming","offroad","unknown"]
  object.position_hint: ["ego_lane_ahead","ego_lane_close","left_lane","right_lane","sidewalk_left","sidewalk_right","crosswalk","
       intersection_zone","unknown"]
  object.motion: ["stopped","moving_straight","braking","turning_left","turning_right","crossing","merging","parked","unknown"]

  lane_markings: ["double_yellow","dashed_white","solid_white","none","unknown"]
  traffic_light: ["red","yellow","green","off","unknown"]
  traffic_signs: ["stop","yield","do_not_enter","wrong_way","speed_limit","no_u_turn","no_left_turn","no_right_turn","no_turn_on_red","one_way
       ","left_turn_only","right_turn_only","straight_only","no_parking","no_stopping","bus_lane","bike_lane","hov_lane","pedestrian_crossing
       ","school_zone","curve_left","curve_right","merge","lane_ends","speed_bump","railroad_crossing","animal_crossing","work_zone","detour","
       road_closed_ahead","steep_grade","roundabout_ahead","exit","unknown"]

  risk.type: ["safe_follow","cut_in","ped_cross","red_light","merge_conflict","dooring","rear_end","overtake_conflict","unknown"]
  risk.likelihood: ["low","med","high","unknown"]
  risk.urgency: ["immediate","near","future","unknown"]
  risk.because_of: short tags from:
                   ["lead_vehicle_ahead","adjacent_vehicle_left","adjacent_vehicle_right","ped_on_sidewalk_left","ped_on_sidewalk_right",
                    "crosswalk_present","intersection_ahead","moderate_traffic","poor_signal_visibility","unknown"]

  rec.immediate: subset of ["cover_brake","slow_down","stop_before_line","keep_lane","avoid_lane_change","yield_to_ped","prepare_to_brake"]
  rec.primary/secondary: short verbs from the same lexicon (no free prose)
  recommendations.conditioned_on.when_risk: must reference a risk id

  ## Required output JSON schema (informal, exact shape)
  {
    "scene": {
      "road_type": <road_type>,
      "traffic": <traffic>,
      "visibility": <visibility>,
      "phase": <phase>,
      "map_hints": [string] # optional short tags, or []
    },

    "infrastructure": {
      "lane_markings": <lane_markings>,
      "traffic_light": <traffic_light>,
      "signs": <traffic_signs> # if not sure, DO NOT list a traffic sign here; ONLY list the TOP 1 confidence traffic sign
    },

    "attention_objects": [
      {
        "id": "<object.category>_<int>", # unique string
        "category": <object.category>,
        "subtype": <object.subtype>,
        "attention_level": <object.attention_level>,
        "lane_relation": <object.lane_relation>,
        "longitudinal_relation": <object.longitudinal_relation>,
        "position_hint": <object.position_hint>,
        "motion": <object.motion>,
        "roles": [<object.roles>] # 0..N roles; [] if none
      }
      # up to ~3 objects total
    ],

    "risks": [
      {
        "id": "rk1",
        "type": <risk.type>,
        "targets": ["obj1","obj2"], # object ids
        "likelihood": <risk.likelihood>,
        "urgency": <risk.urgency>,
        "because_of": [<risk.because_of>] # short tags only
      }
      # optional; may be []
    ],

    "recommendations": {
      "immediate": [<rec.immediate>], # may be []
      "primary": [<rec.immediate>], # reuse same lexicon
      "secondary": [<rec.immediate>],
      "conditioned_on": [
        { "when_risk": "rk1", "actions": [<rec.immediate>] }
      ] # optional; may be []
    }
  }

  ## Rules
  - Output MUST be valid JSON. No trailing commas. No comments in the JSON.
  - Use only the allowed enumerations and short tag lists. If unknown, choose "unknown" or [].
  - IDs in "targets"/"relations" MUST exist in "objects".
  - Keep arrays compact (omit if empty is not allowed; use [] instead).
  - No numbers. No free-form sentences. No units.
  - Emit exactly one JSON object per response; no extra text before/after.
```

B. Runtime Analysis

Table 1 shows that NeoAD-Tiny trains extremely quickly on an 8×4090 setup, requiring only about 25 seconds per epoch and roughly 20 minutes for a full 50-epoch run, thanks to its lightweight fusion encoder and compact planning head. In contrast, NeoAD-Base requires about 4 minutes per epoch, totaling roughly 3 hours for 50 epochs, reflecting its higher BEV resolution, larger feature fusion modules, and expanded transformer capacity. NeoAD-Tiny is ideal for rapid experimentation and data-scaling studies. Although NeoAD-Base is less efficient than NeoAD-Tiny in terms of runtime, it provides higher modeling capacity at a moderate computational cost and outperforms baseline models.

Table 1. Training runtime comparison between NeoAD-Tiny and NeoAD-Base on 8×RTX 4090 GPUs.

Model	Epoch Time	50-Epoch Runtime	Relative Speed
NeoAD-Tiny	∼24–25 s	∼20 min	–
NeoAD-Base	∼3.7–4.0 min	∼180–200 min	∼8× slower

C. Training and Validation Loss Curves

Figure 1 shows the training and validation loss of NeoAD-Tiny across 1–50 epochs under different training data fractions, including 10%, 20%, 50% and 100%, illustrating stable convergence and consistent generalization behavior.

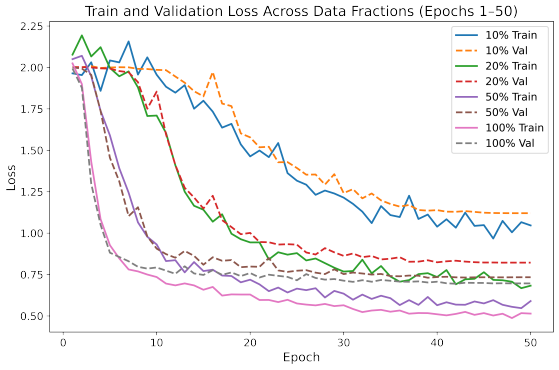

Figure 1. Training and validation loss curves of NeoAD-Tiny trained with different data fractions.

D. Broader Impact

NeoAD aims to improve the safety, robustness, and transparency of autonomous driving systems by introducing explicit representations of human-like physical common sense and safety awareness into the planning loop. By separating intuition, attention, and safety cues into auditable components, the framework provides clearer insight into what the planner attends to and why certain maneuvers are chosen, potentially enabling safer deployment, more interpretable debugging tools, and more reliable responses in long-tail scenarios where purely data-driven E2E models often fail. The broader societal benefit of such a system lies in reducing collision risk, improving comfort for passengers and surrounding road users, and facilitating more trustworthy AV behavior in complex urban environments. Explicit planning tokens also provide a substrate for incorporating formal safety guarantees, supporting future regulation, certification, and human-in-the-loop oversight.

E. Limitations

Despite its advantages, NeoAD has several limitations. The reliance on Cosmos large model reasoning introduces variance: not all samples produce valid attention object outputs, and the planner must fall back to BEV features alone when reasoning is missing. The robustness map, while grounded in formal safety considerations, currently does not model scene dependent comfort constraints, nor does it adapt its safety thresholds to different driving contexts. Finally, NeoAD inherits all dataset and annotation limitations present in nuScenes. Biases in rare cases, particularly interactions with vulnerable road users, occluded agents, and unusual road geometries, remain areas where performance may degrade.

F. Future Work

Several promising directions arise from NeoAD's design. A key extension is to develop scene-conditioned safety and comfort specifications: rather than applying a uniform robustness logic, NeoAD could first classify the situation based on MLLM outputs and then apply a specification tailored to that context. This hierarchical framework could capture subtle distinctions. For instance, enforcing strict safety margins before comfort in high speed merges, but balancing both in low speed turns mirroring human drivers who dynamically adjust their priorities. Another direction is to improve attention object robustness. This includes designing fallback mechanisms for missing Cosmos outputs, self-training strategies that infer attention saliency from successful trajectories, or training lighter reasoning heads that replicate Cosmos behavior at scale. NeoAD may also benefit from directly leveraging richer MLLM signals, including risk alerts, cause effect reasoning, or high-level driving recommendations, to refine both robustness map construction and long horizon planning.