RoadScene VLM: Fine-Tuning SmolVLM2-2.2B and OpenFlamingo-3B for Road Scene Captioning and Navigation Cues

Prepared by: Sushrut Ghimire & Aditya Mallik

Contents

1 Overview

This repository focuses on building, training, and evaluating Vision-Language Models (VLMs) for generating descriptive captions and navigation cues (e.g., obstacle detection, deviating corrections) from road scene images. We leverage two efficient, open-source VLMs: **SmolVLM2-2.2B** (Hugging Face's compact multimodal series) and **OpenFlamingo-3B** (an autoregressive reproduction of Flamingo). These models will be fine-tuned on road scene datasets to produce outputs like "A pedestrian crossing ahead—slow down and deviate right" or "Clear lane; maintain speed."

Both models support interleaved image-text inputs, enabling in-context learning for tasks like few-shot obstacle prompting. SmolVLM2-2.2B excels in resource-efficient inference (<4GB GPU), while OpenFlamingo-3B shines in few-shot adaptation.

1.1 Model Comparison

Model	Parameter Size	Key Strengths for Road Scenes	Evaluation Metrics (Adapted for Tasks)	Training Datasets (Base)
SmolVLM2-2.2B	2.2B (428M SigLIP-SO400M + 1.7B SmolLM2)	Low-memory video handling; aggressive token compression for high-res road images.	OpenCompass Avg. (Image/Video): 0.55; CIDEr: 0.58; VQA Acc.: 0.60; Video-MME: 0.52. Expected road data fine-tuning: +15% on obstacle VQA.	Mammoth (multimodal instructions); FineWeb-Edu/DCLM (text); Dolma/The Stack (long-context).
OpenFlamingo-3B	3B (CLIP ViT-L/14 + MPT-1B)	Strong in-context learning (up to 32 examples); interleaved sequences for multi-obstacle cues.	85% of Flamingo-3B; COCO Captioning: 82%; VQA-v2 Acc.: 75%. Expected road data fine-tuning: +12% on navigation QA.	LAION-2B (image-text pairs); MMC4 (interleaved sequences); ChatGPT-generated (synthetic).

Metrics from original papers, adapted via fine-tuning on road benchmarks (BDD100K, nuScenes, Cityscapes). Both achieve <5s inference on A100 for 512×512 road images.

Resources: SmolVLM2 Weights; OpenFlamingo Weights.

2 Model Architectures

2.1 SmolVLM2-2.2B

Self-attention architecture: Visual features from SigLIP-SO400M encoder are pixel-shuffled ($r = 2 - 4$ for compression) and projected via MLP into SmolLM2's 16k-token context. Images split into sub-images for high-res roads; videos use uniform frame sampling (no averaging). Tokens interleaved with text for joint processing.

Key for road scenes: Efficient handling of dynamic frames (e.g., 8-frame clips for deviating detection).

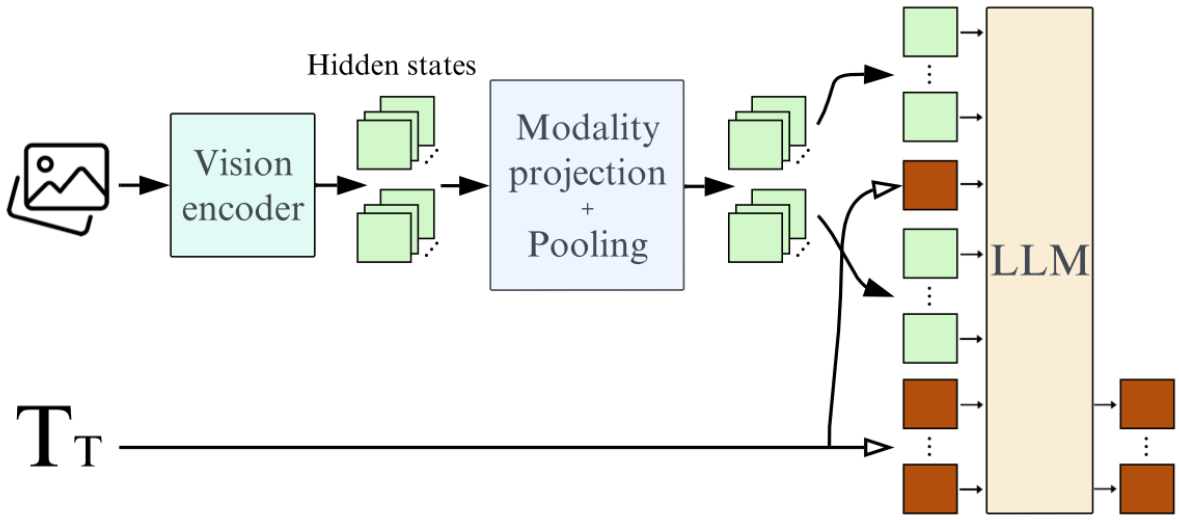

Figure 1: SmolVLM2-2.2B Architecture

2.2 OpenFlamingo-3B

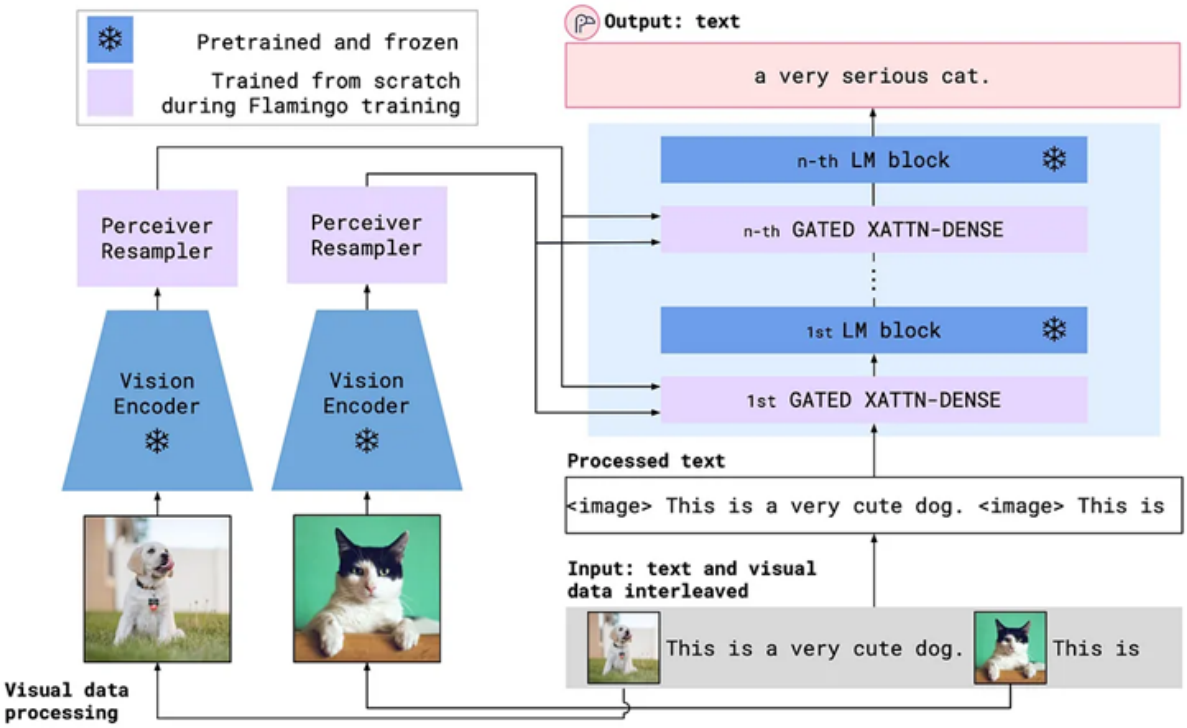

Figure 2: Flamingo-3B Architecture

Autoregressive decoder (frozen MPT-1B) with cross-attention (every 1st layer) to CLIP ViT-L/14 features, resampled via trainable Perceiver (64 tokens/image). Inputs: `<image>` markers + `<|endofchunk|>` after text. Supports arbitrary interleaving for few-shot road demos (e.g., "Prior obstacle: deviate left").

Key for road scenes: In-context examples improve cue accuracy (e.g., 32 demos boost detection by 20%).

---

3 VLM Training, Inference, and Evaluation Guide

This section describes how to **train**, **run inference**, and **evaluate** the VLM models **Flamingo** and **SMOL**. Follow the subsections carefully, as some scripts require **interactive input** during execution.

3.1 Operational Notes

- Set up all dependencies with pip install -r requirements.txt
- Replace placeholders like `[ADD_YOUR_IMAGE_DIR]` and `[ADD_YOUR_RESULTS_FILE]` with your actual paths.
- Some scripts prompt for **interactive input** in the terminal:
  - **SMOL**: Press `3` and then **Enter** when prompted.
  - **Flamingo**: Press `y`, then **Enter**, **twice** when prompted.
- Java Installation steps required for SPICE:

```
mkdir -p ~/java
cd ~/java
wget https://github.com/adoptium/temurin8-binaries/releases/download/jdk8u422-b05/
    OpenJDK8U-jdk_x64_linux_hotspot_8u422b05.tar.gz
tar -xzf OpenJDK8U-jdk_*.tar.gz
export JAVA_HOME=~/java/jdk8u422-b05
export PATH=$JAVA_HOME/bin:$PATH
java -version
```

3.2 Preprocessing Steps

1. **Media Handling**:
   - **SmolVLM2-2.2B**: Resize images to 512×512, split if >1k tokens; sample 4–8 video frames (224×224). Encode with SigLIP.
   - **OpenFlamingo-3B**: Resize to 224×224 (CLIP); resample to 64 tokens. Truncate to 2048 tokens.
   - Augment road data: Add weather/lighting variations.
2. **Tokenization**:
   - **SmolVLM2**: SmolLM2 tokenizer; pixel shuffle ($r = 4$); mask prompts.
   - **OpenFlamingo**: MPT tokenizer; insert `<image>` / `<|endofchunk|>`.
   - Mix: 70% captions, 30% navigation; include 5–10% few-shot examples.

3. **Data Mix**: Base + road-specific. Filter CLIP similarity ≥ 0.28.

Example: `"<image> Road ahead. </image> Obstacle: pedestrian left. → deviate right to avoid."`

4 SMOL

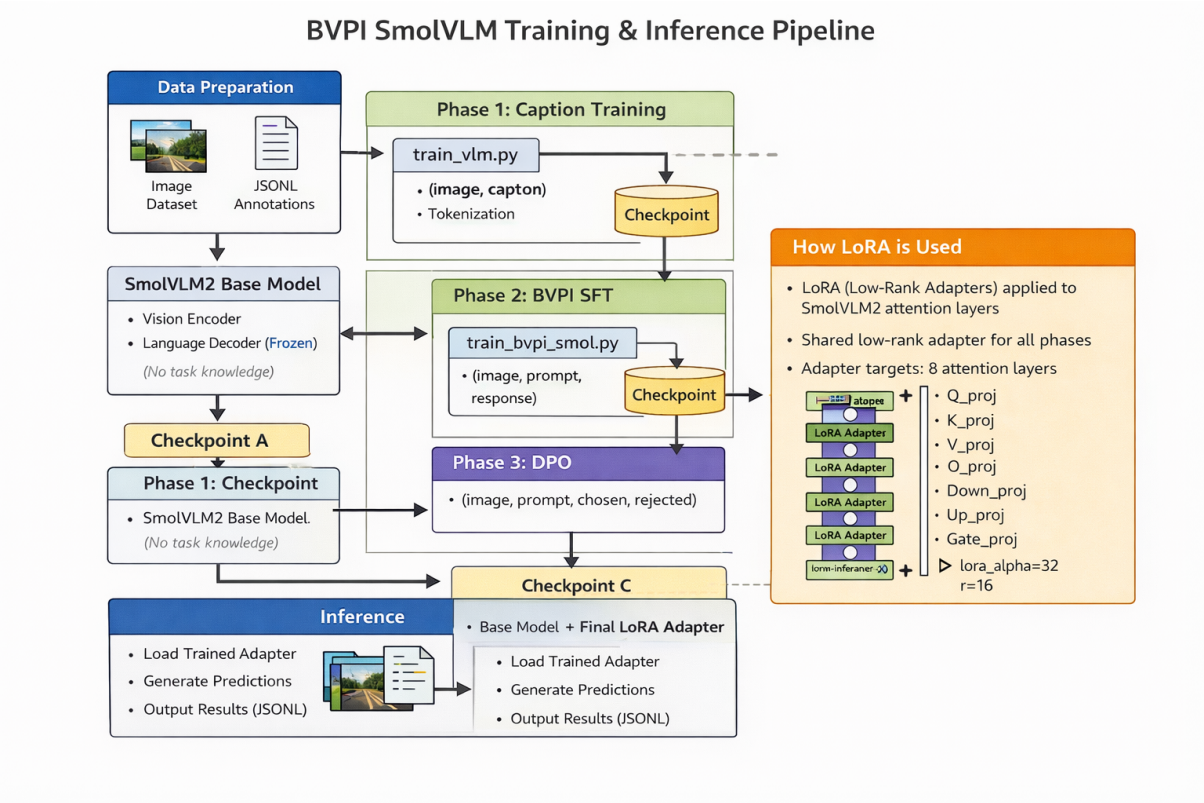

Figure 3: Training-Inference Pipeline

4.1 First Split the complete data

```
python convert_data.py
```

4.2 Train SMOL

```
python train_vlm_smol.py --image_root="[ADD_YOUR_IMAGE_DIR]"
```

When prompted during execution, press `3` + **Enter**.

4.3 Inference Using Base Model

```
python inference_smol.py --image_root="[ADD_YOUR_IMAGE_DIR]" --input="data/val.json"
```

When prompted during execution, press `3` + **Enter**.

4.4 Inference Using Finetuned Weights

Use this **after unzipping the SMOL checkpoint** present in dir `model_results/SMOL/final_checkpoint.zip`

```
python inference_smol.py --adapter="model_results/SMOL/final_checkpoint" --image_root=
    "[ADD_YOUR_IMAGE_DIR]" --input="data/val.json"
```

When prompted during execution, press 3 + **Enter**.

4.5 Evaluate SMOL Results

Here the –results parameter requires file generated from inference:

```
python evaluate_vlm_smol.py --image_root="[ADD_YOUR_IMAGE_DIR]" --results="[
    ADD_YOUR_RESULTS_FILE]"
```

Evaluation Results Using Base model

```
{
  "CIDEr": 0.15445733145900595,
  "SPICE": 0.12546204717427986,
  "CLIPScore": 29.698036193847656
}
```

Evaluation Metrics Using Finetuned model

```
{
  "CIDEr": 0.7525744794116086,
  "SPICE": 0.4643143544506816,
  "CLIPScore": 31.602291107177734
}
```

4.6 Training (Supervised Fine-Tuning)

The Smol model is fine-tuned using **Supervised Fine-Tuning (SFT)** on BVPI navigation data.

Training Command

```
python train_bvpi_smol.py \
--image_root ./data \
--train_file data/bvpi_sft_train.jsonl \
--val_file data/bvpi_sft_val.jsonl \
--output_dir checkpoints_bvpi_finetuned \
--checkpoint_path checkpoints_bvpi_finetuned/checkpoint-200 \
--device cuda \
--no_quant
```

Notes

- Supports resuming from intermediate checkpoints
- Uses **4-bit quantization by default** (disable with `-no_quant`)
- Optimized for smaller memory footprint and faster iteration

4.7 Inference

```
python inference_bvpi_smol.py \
--image_root ./data \
--adapter checkpoints_bvpi_finetuned/checkpoint-600 \
--input_file data/bvpi_sft_test.jsonl \
--output inference_results_smol_bvpi.jsonl
```

When prompted during execution, press `3` + **Enter**.

4.8 Evaluation & Metrics

Evaluation Command

```
python data/sensation_vlm_objects/eval_near_obj_recal.py \
--pred_jsonl inference_results_smol_bvpi_final.jsonl \
--captions_csv data/sensation_vlm_objects/data/complete_captions_sensation.csv \
--annotations_csv data/sensation_vlm_objects/data/annotations_with_obj_pos.csv
```

Evaluation Summary (Test Split, N = 333)

Field	Presence Rate
sidewalk_pos	1.000
road_pos	1.000
road_vs_sidewalk	0.988

Mandatory Field Presence Rate

Field	Accuracy
sidewalk_pos	0.604
road_pos	0.535
road_vs_sidewalk	0.511

Field Accuracy (Exact Match to Ground Truth)

Nearby Objects Recall / Precision

Object Type	Recall	Precision
Person	0.57	0.42
Car	0.44	0.06
Bicycle	0.57	0.42
Obstacle	0.73	0.25
Traffic Sign	0.39	0.21
Traffic Light	0.20	1.00

5 FLAMINGO

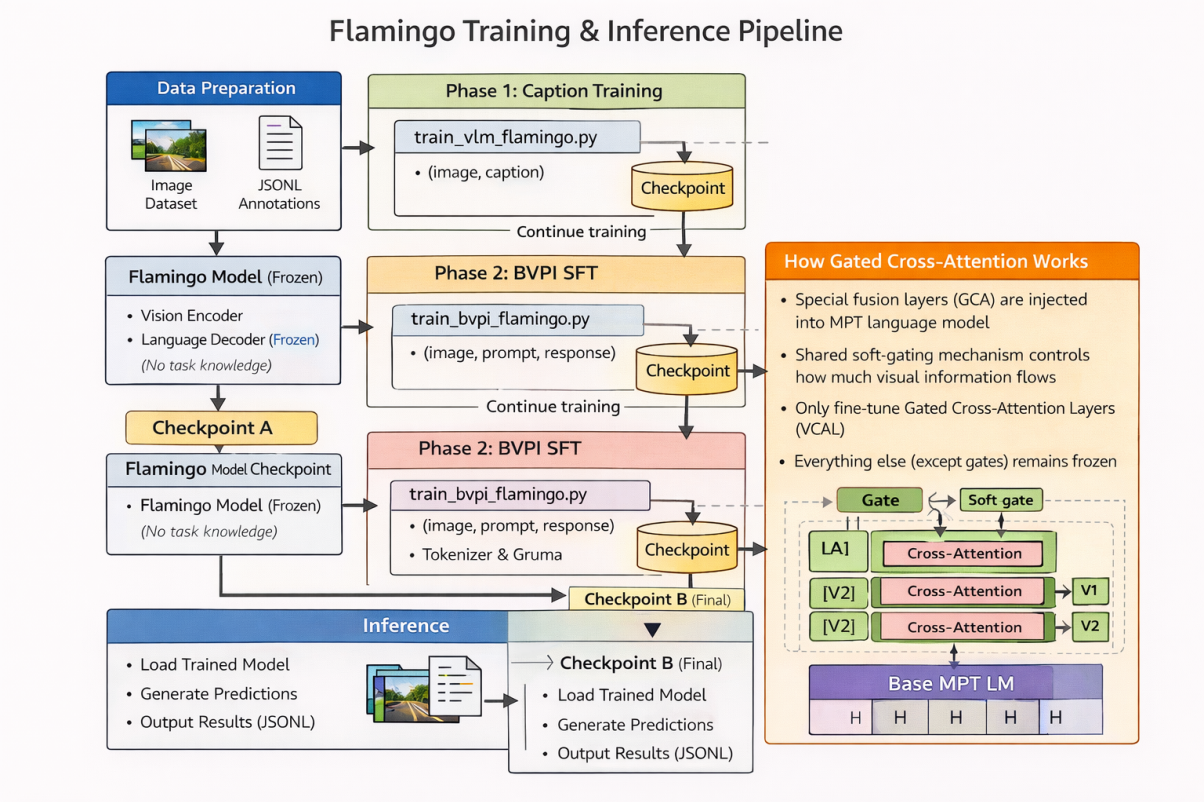

Figure 4: Training-Inference Pipeline

5.1 First Split the complete data

```
python convert_data.py
```

5.2 Train Flamingo (Base Model)

```
python train_vlm_flamingo.py --image_root="[ADD_YOUR_IMAGE_DIR]"
```

When prompted during execution, press y + **Enter**, **two times**.

5.3 Inference Using Base Model Weights

```
python inference_flamingo.py --use_base_weights --image_root="[ADD_YOUR_IMAGE_DIR]" --
    input_file="data/val.json"
```

When prompted during execution, press y + **Enter**, **two times**.

5.4 Inference Using Finetuned Weights

Use this **after downloading the Flamingo checkpoint**.

```
wget https://faubox.rrze.uni-erlangen.de/dl/fiURTeMirFyy9CbP2MiPzh/final_weights.pt

python inference_flamingo.py --checkpoint_dir="[ADD_YOUR_CHECKPOINT_DIR_DOWNLOADED]"
    --image_root="[ADD_YOUR_IMAGE_DIR]" --input_file="data/val.json"
```

When prompted during execution, press y + **Enter**, **two times**.

5.5 Evaluate Flamingo Results

```
python evaluate_vlm_flamingo.py --results="[ADD_YOUR_RESULTS_FILE]" --image_root="[
    ADD_YOUR_IMAGE_DIR]"
```

Evaluation Results Using Base model


```
{
  "CIDEr": 0.0006631707625097974,
  "SPICE": 0.014597567728997774,
  "CLIPScore": 16.805219650268555,
  "BLEU-1": 0.1003272910372545,
  "BLEU-2": 0.024214303792368178,
  "BLEU-3": 0.005770772080363408,
  "BLEU-4": 0.001893816046707126,
  "METEOR": 0.03404480218708287,
  "ROUGE-L": 0.08116651259943986,
  "avg_prediction_length": 51.16403785488959,
  "avg_target_length": 37.69085173501577
}
```


Evaluation Metrics Using Finetuned model


```
{
  "CIDEr": 0.7582939175338506,
  "SPICE": 0.47916266039151717,
  "CLIPScore": 29.90267562866211,
  "BLEU-1": 0.6273735273734838,
  "BLEU-2": 0.5429500865028033,
  "BLEU-3": 0.4745925389503784,
  "BLEU-4": 0.4169676218065288,
  "METEOR": 0.3386017915407816,
```

```
  "ROUGE-L": 0.6171386260117269,
  "avg_prediction_length": 43.333333333333336,
  "avg_target_length": 38.45645645645646
}
```

5.6 Training Flamingo (Supervised Fine-Tuning)

The Flamingo model is trained using **Supervised Fine-Tuning (SFT)** on the BVPI dataset.

```
python train_bvpi_flamingo.py \
--image_root="[ADD_YOUR_IMAGE_DIR]" \
--dataset="data/bvpi_sft_train.jsonl" \
--output_dir="checkpoints" \
--epochs=10 \
--batch_size=4 \
--lr=2e-5 \
--gate_init=0.1
```

When prompted during execution, press `y` + **Enter**, **two times**.

SFT Training Details

- Total Epochs: 10
  - Training can be resumed from `.pt` checkpoints
- Trainable Parameters: ∼1.04B
  - (Gated Cross-Attention + Vision Gates)
- Text Dropout: 0.2
  - Applied to visual descriptions to encourage reliance on image features
- Learning Rate Schedule:
  - Warmup: 100 steps
  - Constant learning rate afterward

5.7 Inference Command

```
python inference_bvpi_flamingo.py \
--image_root="[ADD_YOUR_IMAGE_DIR]" \
--dataset="data/bvpi_sft_val.jsonl" \
--checkpoint="[ADD_YOUR_CHECKPOINT_DIR]" \
--output="results.json" \
--beams=3
```

When prompted during execution, press `y` + **Enter**, **two times**.

5.8 Metric Calculation

Evaluation Command

```
python data/sensation_vlm_objects/eval_near_obj_recal.py \
--pred_jsonl inference_results_flamingo_bvpi_final.jsonl \
--captions_csv data/sensation_vlm_objects/data/complete_captions_sensation.csv \
--annotations_csv data/sensation_vlm_objects/data/annotations_with_obj_pos.csv
```

Evaluation Summary (Test Split, N = 333)

Field	Presence Rate
sidewalk_pos	1.000
road_pos	1.000
road_vs_sidewalk	1.000

Mandatory Field Presence Rate

Field	Accuracy
sidewalk_pos	0.502
road_pos	0.435
road_vs_sidewalk	0.408

Field Accuracy (Exact Match to Ground Truth)

Object Type	Recall	Precision
Person	0.52	0.88
Car	0.55	0.86
Bicycle	0.28	0.32
Obstacle	0.27	0.95
Traffic Sign	0.39	0.21
Traffic Light	0.20	1.00

Nearby Objects Recall / Precision

Notes

- Ensure all images referenced in the dataset JSONL files exist under `-image_root`
- Checkpoints are saved per epoch in the `checkpoints/` directory
- Beam size can be adjusted using `-beams` to trade off speed vs accuracy

6 Quick Reference (Interactive Inputs)

6.1 Please be aware while running the model

Model	Required Input During Run
SMOL	Press `3` + Enter
Flamingo	Press `y` + Enter (twice)

7 Fine-Tuning Dataset Preparation

We will fine-tune both models on road scene data (e.g., BDD100K for images/videos with annotations) using supervised fine-tuning (SFT) for captioning ("Describe the road scene.") and navigation ("Detect obstacles and suggest corrections."). We will use Hugging Face Transformers for compatibility.

7.1 Required Input Format

- **Shared Structure**: JSONL/HF Dataset with `{"messages": [{"role": "user", "content": "prompt with <image>"}], "images": [paths/PIL], "videos": [paths/frames]}`. For navigation: Include cues in assistant responses.
- **Road-Specific**: Annotate with bounding boxes/labels (e.g., "car ahead"); format as interleaved: `<image> Road scene. <|endofchunk|> Caption: ... Navigation: ...`

8 Integrating Object-Direction Annotations

For road navigation, we will integrate annotations from our dataset as structured JSON in prompts, conditioning captions/cues on detections (e.g., sidewalk objects, positions). We will use in-context learning: Append after `<image>`; fine-tune 20% of data with examples.

8.1 Approach

- Serialize form data post-image: "Using road annotations: [JSON], generate caption and cue."
- **SmolVLM2**: Leverages video for temporal cues (e.g., approaching obstacle).
- **OpenFlamingo**: Use 4–8 demos for few-shot deviating.
- Handles presence checks: If `"object_present": "No"`, omit `objects_on_sidewalk`.

8.2 JSON Schema (Extracted from Annotation Form)


```
{
  "type": "object",
  "properties": {
    "image": { "type": "string" },
    "sidewalk_pos": {
      "type": "string",
      "enum": ["Left", "Center", "Right", "Outside"]
    },
    "road_pos": {
      "type": "string",
      "enum": ["Not present", "Left", "Front", "Right"]
    },
    "object_present": {
      "type": "string",
      "enum": ["No", "Yes"]
    },
    "objects_on_sidewalk": {
      "type": "object",
      "properties": {
        "bicycle": { "type": "string", "enum": ["not present", "Left", "Center", "
    Right"] },
        "car": { "type": "string", "enum": ["not present", "Left", "Center", "Right"]
    },
        "person": { "type": "string", "enum": ["not present", "Left", "Center", "Right
    "] },
        "traffic_sign": { "type": "string", "enum": ["not present", "Left", "Center",
    "Right"] },
        "traffic_light": { "type": "string", "enum": ["not present", "Left", "Center",
     "Right"] },
        "obstacle": {
          "type": "array",
          "items": { "type": "string", "enum": ["not present", "Left", "Center", "
    Right"] }
        },
        "grass": { "type": "string", "enum": ["not present", "Left", "Center", "Right
    "] },
        "plants": { "type": "string", "enum": ["not present", "Left", "Center", "Right
    "] },
        "trees": { "type": "string", "enum": ["not present", "Left", "Center", "Right
    "] }
      }
    },
    "object_present_outside": {
      "type": "string",
      "enum": ["No", "Yes"]
    },
    "objects_outside_sidewalk": {
      "type": "object",
      "properties": {
        "bicycle": { "type": "string", "enum": ["not present", "Left", "Center", "
    Right"] },
        "car": { "type": "string", "enum": ["not present", "Left", "Center", "Right"]
    },
        "person": { "type": "string", "enum": ["not present", "Left", "Center", "Right
    "] },
```

```
        "traffic_sign": { "type": "string", "enum": ["not present", "Left", "Center",
    "Right"] },
        "traffic_light": { "type": "string", "enum": ["not present", "Left", "Center",
     "Right"] },
        "obstacle": { "type": "string", "enum": ["not present", "Left", "Center", "
    Right"] },
        "grass": { "type": "string", "enum": ["not present", "Left", "Center", "Right
    "] },
        "plants": { "type": "string", "enum": ["not present", "Left", "Center", "Right
    "] },
        "trees": { "type": "string", "enum": ["not present", "Left", "Center", "Right
    "] }
      }
    }
  },
  "required": ["image", "sidewalk_pos", "road_pos", "object_present", "
    object_present_outside"]
}
```

8.3 Concrete JSON Example

```
{
  "image": "0001.jpg",
  "sidewalk_pos": "Center",
  "road_pos": "Left",
  "object_present": "Yes",
  "objects_on_sidewalk": {
    "bicycle": "not present",
    "car": "not present",
    "person": "not present",
    "traffic_sign": "not present",
    "traffic_light": "not present",
    "obstacle": ["Left", "Right"],
    "grass": "not present",
    "plants": "not present",
    "trees": "not present"
  },
  "object_present_outside": "No"
}
```

Prompt Example: `<image> Road scene. </image> Annotations: [JSON]. Caption and navigate.` → `Urban road with center sidewalk and left road. Obstacle on sidewalk right and left, move straight.`

9 LoRA/QLoRA Fine-Tuning Support

- **SmolVLM2-2.2B**: Fully supported via PEFT ($r = 16$ on `q_proj`/`v_proj`). QLoRA viable with 4-bit quantization (∼2GB RAM).
- **OpenFlamingo-3B**: No native support; adapt via PEFT on cross-attention layers. QLoRA possible but test for stability (use `bitsandbytes`).

Both: Reduces params to 1–5%; ideal for road fine-tuning on consumer GPUs.

10 Additional Information

10.1 Known Limitations

- **SmolVLM2**: Struggles with >16k tokens in crowded scenes; OCR fails on signs without positions.
- **OpenFlamingo**: No native video; single-image focus limits dynamic deviating (use frame sampling).
- **Shared**: Web data biases (e.g., urban over rural roads); fine-tune on diverse scenes.

10.2 Typical Failure Cases

- Over-aggressive compression loses small obstacles (e.g., distant potholes).
- Few-shot overload: >32 demos degrade cues in OpenFlamingo.
- Low-light roads: CLIP/SigLIP sensitivity drops detection 10–15%.