Technical Report

Cristian Gariboldi

This document details the design, performance, and limitations of a Vision Language Action Model (VLAM) developed to predict wheel loader actions from visual input and textual commands.

1.1 Automated Dataset Preparation

The foundation of any successful machine learning model is a high-quality dataset. To avoid the time-consuming and costly process of manual annotation, I developed an automated pipeline to generate a rich, structured Question-Answering (QA) dataset for fine-tuning our VLM.

1.1.1 The QA Generation Pipeline

My approach leverages a large, pre-trained Vision Language Model (LLaVA-v1.6-34b) as a "teacher" model. This model analyzes raw images from a wheel loader's perspective and generates descriptive data based on a series of engineered prompts. The core of this pipeline is a Python script (QA_data_generation.py) that takes a directory of images as input and outputs a structured JSON file containing diverse QA pairs for each image.

The key advantage of this methodology is its scalability and flexibility. By simply modifying the prompt-generation functions within the script, this pipeline can be rapidly adapted to create datasets for entirely different domains or tasks, such as autonomous trucking, warehouse robotics, or drone-based inspection, with minimal code changes.

1.1.2 Task-Specific Prompt Engineering and Action Space

For the specific context of wheel loader operations, I engineered four distinct types of prompts to generate a comprehensive dataset entry for each image:

Scene Description: A general overview of the environment.

Object Localization: Identifying the primary target (e.g., a pile of material).

Hazard Detection: An analysis of potential safety risks.

Action Prediction: A decision on the next logical action to perform.

For the Action Prediction task, I defined a discrete action space to make the model's output both predictable and directly translatable to robotic commands. The four chosen commands represent a simplified but complete operational cycle:

DRIVE_TO_PILE: The navigation and approach phase.

DIG: The primary task execution phase.

DUMP: The task completion and offloading phase.

WAIT: A safety-oriented action, triggered by obstacles or hazards.

This action space provides a clear and concise set of commands for the model to learn, forming the basis of its decision-making capability.

1.2 Model Performance and Limitations

The automated pipeline was run on a dataset of over 2,000 images extracted from point-of-view videos of wheel loader operations.

1.2.1 Generated Dataset Statistics

The analysis of the generated QA pairs for the action prediction task revealed the following distribution:

DRIVE_TO_PILE: 1808 samples (81.9%)

DIG: 399 samples (18.1%)

DUMP: 0 samples (0.0%)

WAIT: 0 samples (0.0%)

Total Actions Counted: 2207

1.2.2 Limitations and Analysis The primary limitation of our initial generated dataset is the severe class imbalance. The actions DUMP and WAIT were never generated by the "teacher" VLM. There are two likely causes for this:

1) Visual Bias in Source Data: The source videos predominantly featured the tasks of approaching and digging into material piles. Frames showing the specific moment of dumping into a truck or situations requiring the loader to wait for a hazard to clear were rare. The VLM can only generate answers based on the visual evidence presented in the source images.

2) Implicit Prompting Conditions: The visual conditions required to trigger a DUMP (e.g., a full bucket positioned directly over a truck) or WAIT (e.g., a clear and immediate hazard in the direct path) may not have been met with sufficient frequency or clarity in the randomly sampled video frames.

1.3 Suggestions for Further Improvements

The identified limitation of data imbalance can be addressed through the following methods to create a more robust and capable model.

1.3.1 Dataset Enhancement Strategies Targeted Data Curation: Instead of random sampling, future work should involve actively seeking and extracting video segments that explicitly depict the underrepresented DUMP and WAIT scenarios.

Synthetic Data Generation: A highly effective and fast approach would be to use generative AI models (e.g., Stable Diffusion, Midjourney) to create synthetic images of these edge cases. We can generate hundreds of images showing a loader waiting for a person to cross or positioned to dump material into a truck, rapidly balancing the dataset.

1.3.2 Preparation for Fine-Tuning

To prepare the generated data for training, two final steps were performed:

1) Conversion to LLaMA Format: The dataset was converted into the standard conversational format required for fine-tuning LLaMA-based models.

2) Stratified Splitting: The dataset was split into training and evaluation sets using a stratified split. This technique ensures that the original distribution of the DRIVE_TO_PILE and DIG commands was preserved in both sets, allowing for an unbiased evaluation of the model's performance on the classes it was trained on.

2.0 Model Fine-Tuning and Performance

This section details the methodology used to fine-tune the base Vision Language Model (VLM), an analysis of its performance on the action prediction task, and a discussion of its limitations and potential applications.

2.1 Fine-Tuning Methodology

The primary objective of the fine-tuning process was to adapt a general-purpose, pre-trained VLM for the specialized domain of wheel loader action prediction.

Base Model: The foundation for this project is a Vicuna-7b-v1.5 model, which combines a powerful language model with a pre-trained vision encoder (CLIP).

Fine-Tuning Technique: To efficiently adapt this model, I employed Low-Rank Adaptation (LoRA). This technique freezes the vast majority of the model's original weights and injects small, trainable "adapter" layers. This allows for rapid fine-tuning with significantly less computational cost than training the full model, while still achieving high performance on the target task.

Training Parameters: The model was trained for one epoch using the train_dataset.json generated in the previous step. Key hyperparameters, managed by a DeepSpeed Zero-3 configuration, include a LoRA rank of 128, a learning rate of 2e-5, and an effective batch size of 8.

2.2 Evaluation Protocol

To quantitatively measure the effectiveness of fine-tuning, I evaluated both the original base model and our fine-tuned model against a held-out evaluation set.

Dataset: The evaluation was performed on eval_dataset.json, which was created via a stratified split to ensure the distribution of action commands mirrored the training set.

Task: For each image in the dataset, the model was provided with the prompt, "You are operating a wheel loader. Based on the image, what is the most logical next action to perform?..." and its generated text was evaluated against the ground-truth answer.

Metrics: Performance was measured using standard classification metrics:

1) Overall Accuracy: The percentage of correctly predicted actions.
2) Classification Report: A detailed breakdown of precision, recall, and F1-score for each action class.

3) Confusion Matrix: A table showing where the model's predictions succeeded and where they were confused between classes.

2.3 Performance Analysis and Results

The results clearly demonstrate the profound impact of LoRA fine-tuning on the specialized dataset.

2.3.1 Baseline Performance (Base Model)

The original, general-purpose VLM performed poorly on this specialized task, achieving an Overall Accuracy of only 18.33%.

Classification Report & Confusion Matrix Analysis: The detailed results show that the base model has an extreme prediction bias. It overwhelmingly predicted DIG for nearly all inputs (437 times out of 442), regardless of the image content. While this led to a high recall for the DIG class (97%), its precision was terrible (18%), and it failed almost completely on the majority class, DRIVE_TO_PILE (3 correct out of 362). This performance is only slightly better than random chance and confirms that the base model lacks the specific contextual understanding required for this task.

2.3.2 Fine-Tuned Model Performance After LoRA fine-tuning, the model's performance improved dramatically, achieving an Overall Accuracy of 83.71%.

Classification Report & Confusion Matrix Analysis: This represents a +65.38% absolute improvement in accuracy, demonstrating that the fine-tuning process was highly effective. The model is now able to correctly predict DRIVE_TO_PILE with 89% recall and DIG with 59% recall. The severe prediction bias of the base model has been successfully corrected.

2.4 Limitations Despite the significant performance gains, the model has several key limitations:

Inability to Predict Unseen Actions: As identified during dataset analysis, the training data contained no examples of the DUMP or WAIT actions. Consequently, the fine-tuned model is fundamentally incapable of predicting these commands, which is its most significant operational limitation.

Confusion Between Similar States: The confusion matrix shows that the model still sometimes confuses DRIVE_TO_PILE (39 misclassifications) and DIG (33 misclassifications). This is likely because the visual difference between "approaching a pile" and "being right at a pile" can be subtle, and the model's single-frame analysis struggles to distinguish these sequential states.

Lack of Temporal Context: The model operates on static, single frames. It has no memory of past actions or a concept of the future. A real operator's decision is based on a sequence of events (e.g., "I just finished digging, so now I need to find a truck to dump into"). Our model currently lacks this crucial temporal reasoning.

2.5 Potential Real-World Applications

Even with its limitations, the developed model serves as a strong foundation for several real-world applications:

Operator Assist System: The model can function as an intelligent "co-pilot," highlighting the most probable next task for an operator or flagging potential hazards they might have missed, thereby increasing efficiency and safety.

Automated Fleet Monitoring: Deployed across a fleet, the system could automatically log the activities and states of machinery ("Loader #7 performed 47 DIG cycles"), providing valuable data for operations management without manual input.

Perception Module for Autonomy: The model is an ideal perception component within a larger autonomous system. Its action predictions can serve as a primary input to a higher-level decision-making and motion planning module.

2.6 Suggestions for Further Improvements and Robotics Integration

To move from this successful prototype to a field-ready system, the following steps are recommended:

2.6.1 Further Improvements

Address Data Imbalance: The highest priority is to enrich the dataset with curated or synthetically generated examples of the DUMP and WAIT actions. This is essential for teaching the model the full operational cycle.

Incorporate Temporal Data: The model should be enhanced to process short video clips or a sequence of recent frames instead of a single image. This would provide the temporal context needed to better distinguish between sequential actions like DRIVE_TO_PILE and DIG.

Integrate Machine State: The model's reasoning can be improved by providing it with the loader's own state (e.g., current speed, bucket is full/empty) as additional text in the prompt. This gives the model more context to make a logical decision.

2.6.2 Integration into Robotics Systems Action Command Interface: The model's discrete action outputs are well-suited for a robotics architecture. These commands can be directly mapped to states in a Behavior Tree or State Machine, which are standard robotics control structures. For example, the DRIVE_TO_PILE output would trigger the "Navigate to Target" behavior.

Integration with Motion Planning: The VLM's localization output ("the pile is in the front-left") can be converted into coordinates to provide a goal for a motion planning module (e.g., a navigation stack), which would then generate the precise path and motor commands.

Create a Closed-Loop System: For full autonomy, the VLM must operate in a closed loop. The system would follow this cycle: 1) The VLM analyzes the current camera frame and predicts an action. 2) The robotics control system executes that action for a short duration. 3) A new camera frame is captured, and the cycle repeats, allowing the machine to continuously perceive and react to its environment.