STELLA

SIMULATED TRAINING ENVIRONMENTS
AND LARGE LEARNING AUTOMATA

Project Walkthrough

SE 4450 – Team 32

PROJECT SUMMARY AND TARGETS

STELLA

- STELLA is an implementation of a autonomous driving agent trained in simulations
- The project will adapt recent techniques into a usable framework and tools for others to adapt and use.
  - while also reproducing and verifying those published results

Figure 1: Autonomous vehicle in the CARLA simulator

STELLA
Project Goals

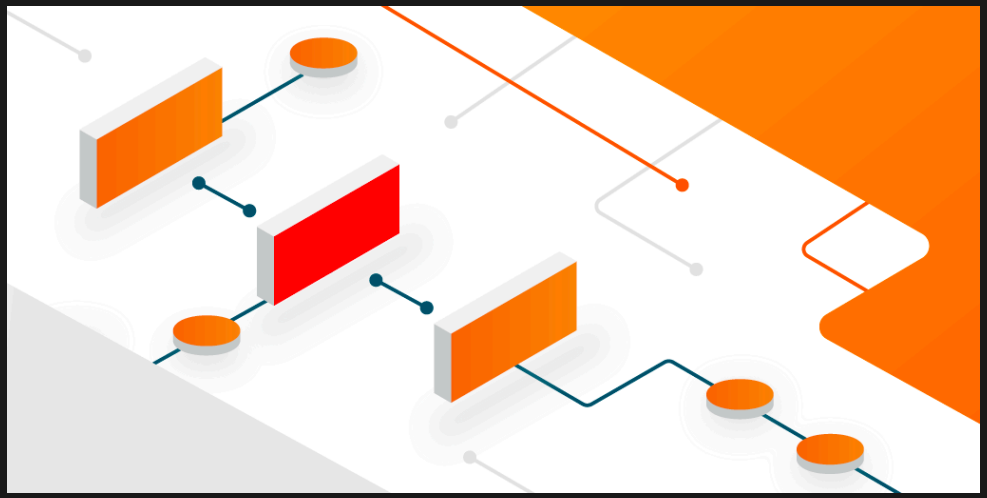


Figure 2: Data flow visualization

- The main goal is to build a working autonomous agent based off of the CarLLaVA technique by Renz et al [1]
- In the process, many tools may need to be built or adapted
- The final agent also needs to be evaluated through a variety of different metrics to compare with other frameworks and techniques

STELLA
FOSS

- While the techniques we are adapting are published for reading, the source code has not been released so there are no available implementations or reproductions of this work yet
- The project will rely on a variety of free and open source tools
- In return, the project hopes to help improve these open source tools that we use through
  - better documentation
  - bug finding and fixing
  - feature additions

METHODOLOGY

Workflow and Structures

- Setup simulator and development environment
- Collect training data
- Build initial agent
- Train, test, repeat

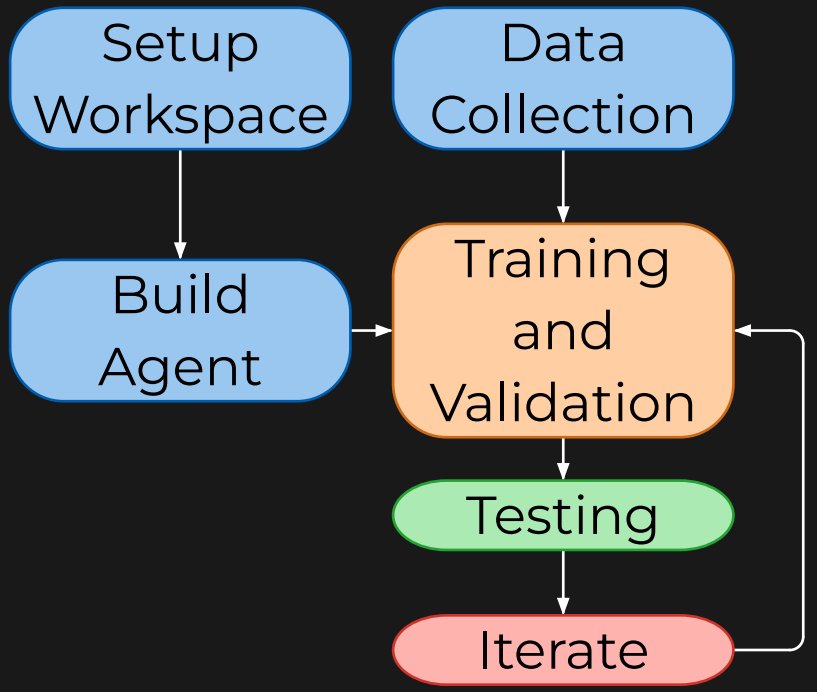


Figure 3: Workflow

Workflow and Structures

Training Data Collection / Generation

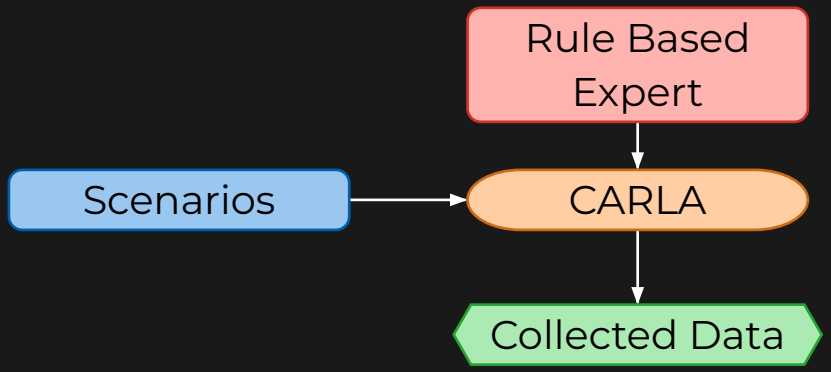


Figure 4: Data Collection

- Follows the dataset generation method in Renz et al [1]
- Rule based experts are allowed to cheat, directly sourcing exactly where other vehicles are in the simulation
- Relies on branching decision trees, rather than adaptive machine learning methods
  - more consistent at specific scenarios but worse at handling edge cases

Workflow and Structures

Training

- The collected data is then used to fine tune a pretrained model
- The pretrained CarLLaVA model has already trained on a few million images of common objects, so it has some built in intuition already

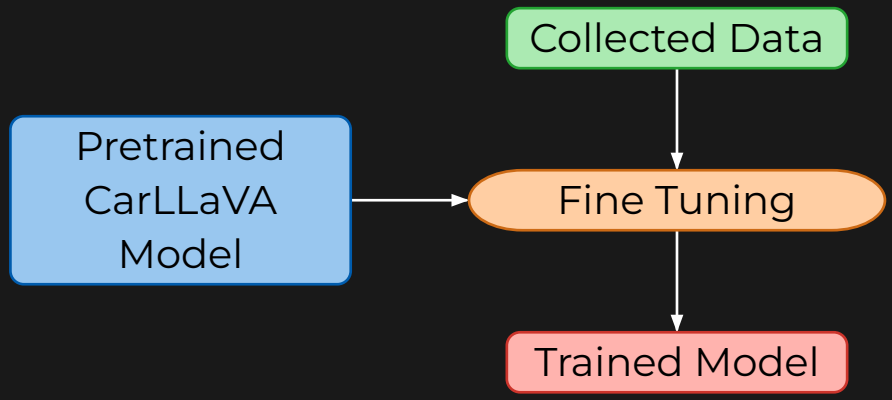


Figure 5: Training

Workflow and Structures

Autonomous Agent Internals

- The CarLLaVA model handles decision making and visual processing
- the agent then forwards the output waypoints to a proportional, integral, derivative (PID) controller
  - combined with GPS data to complete its navigation

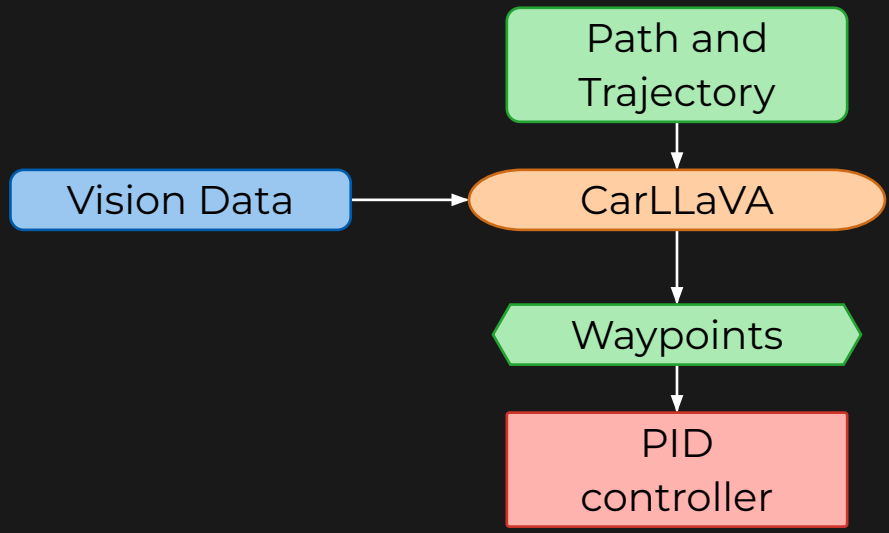


Figure 6: Agent Structure

PID Control

- PID is an algorithm for smooth transitions towards a target point using a scalar feedback value (error e)[1]
- This can be combined with GPS data to align vehicle over designated waypoints

$$\text{PID}(e,t) = k_p e + k_i \int_0^t e\ dt + k_d \frac{de}{dt}$$

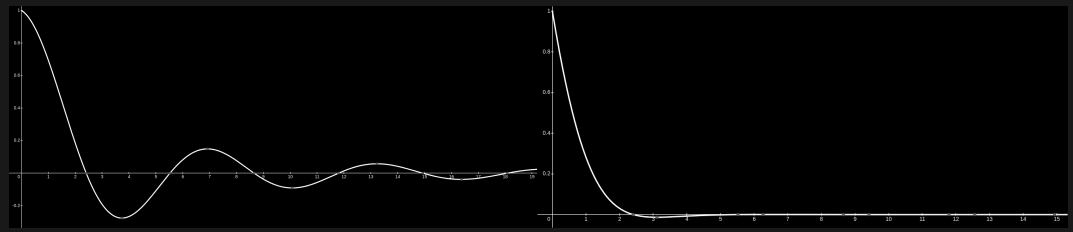


Figure 7: Oscillation Dampening Using PID control

[1] Several of our team members have experience with PID through robotics

CarLLaVa

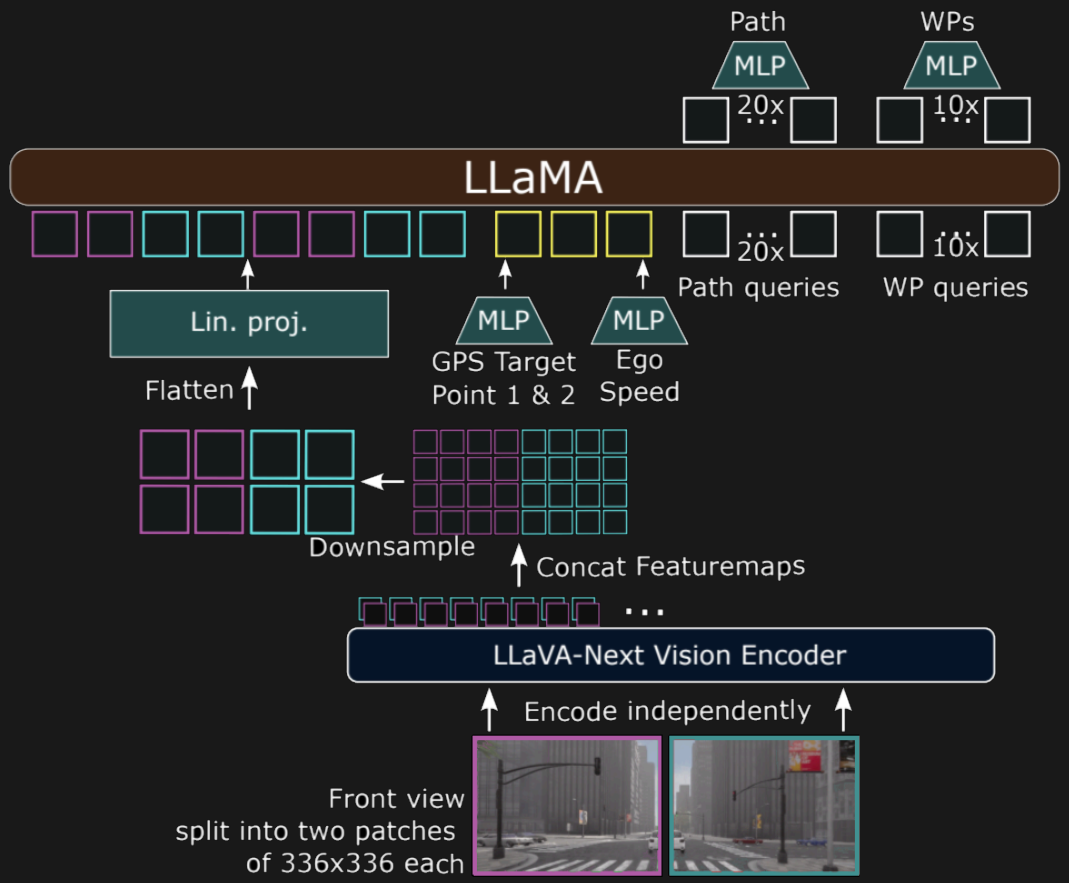


Figure 8: CarLLaVA

- CarLLaVA utilises vision encoders to embed image information in a latent vector space
- These features are then passed to the LLaMA transformer, which combines that with query, target, and status information to generate new GPS waypoints to follow

CarLLaVa

Vision Encoding

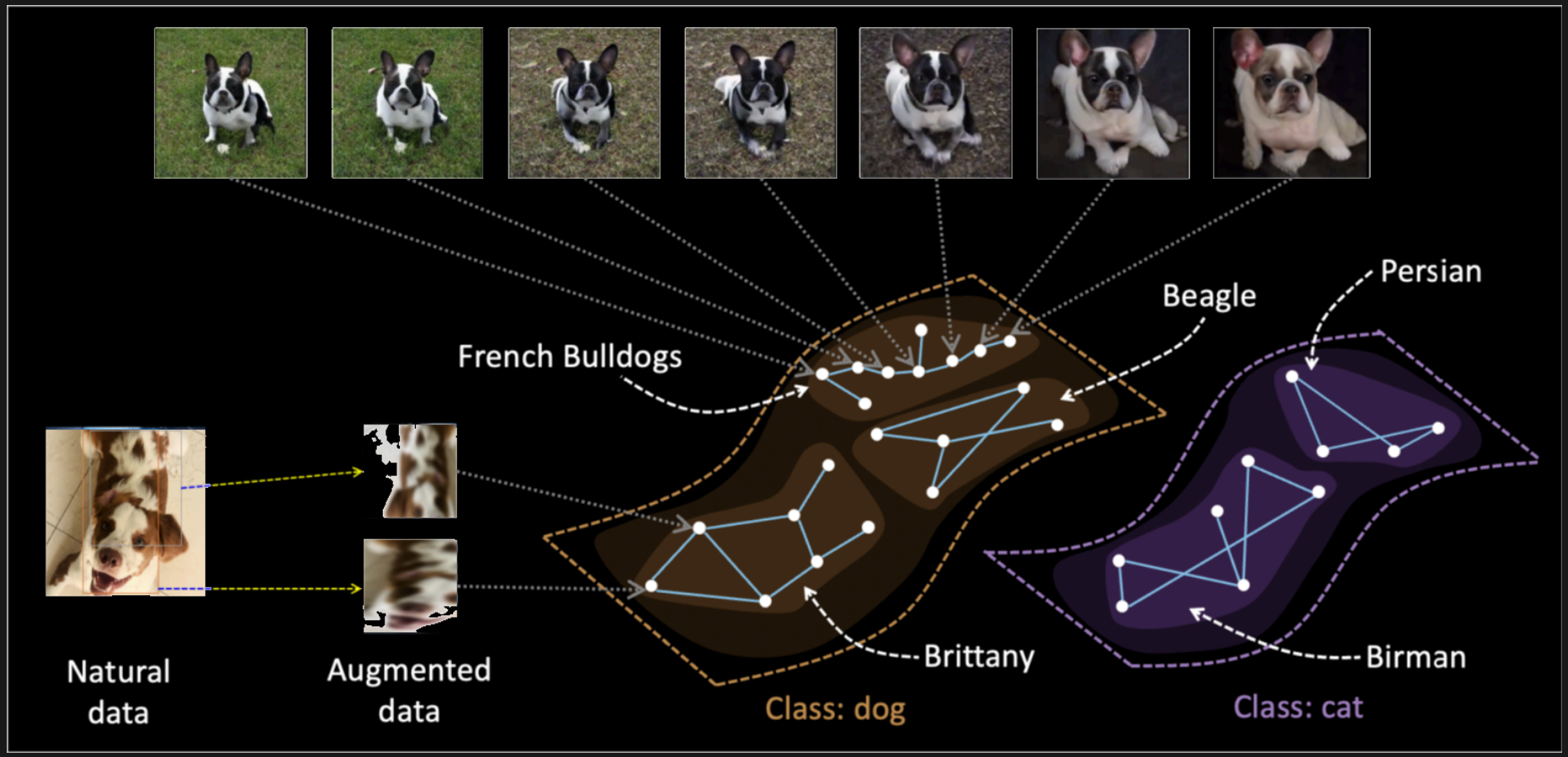


Figure 9: Vision Encoding

CarLLaVa

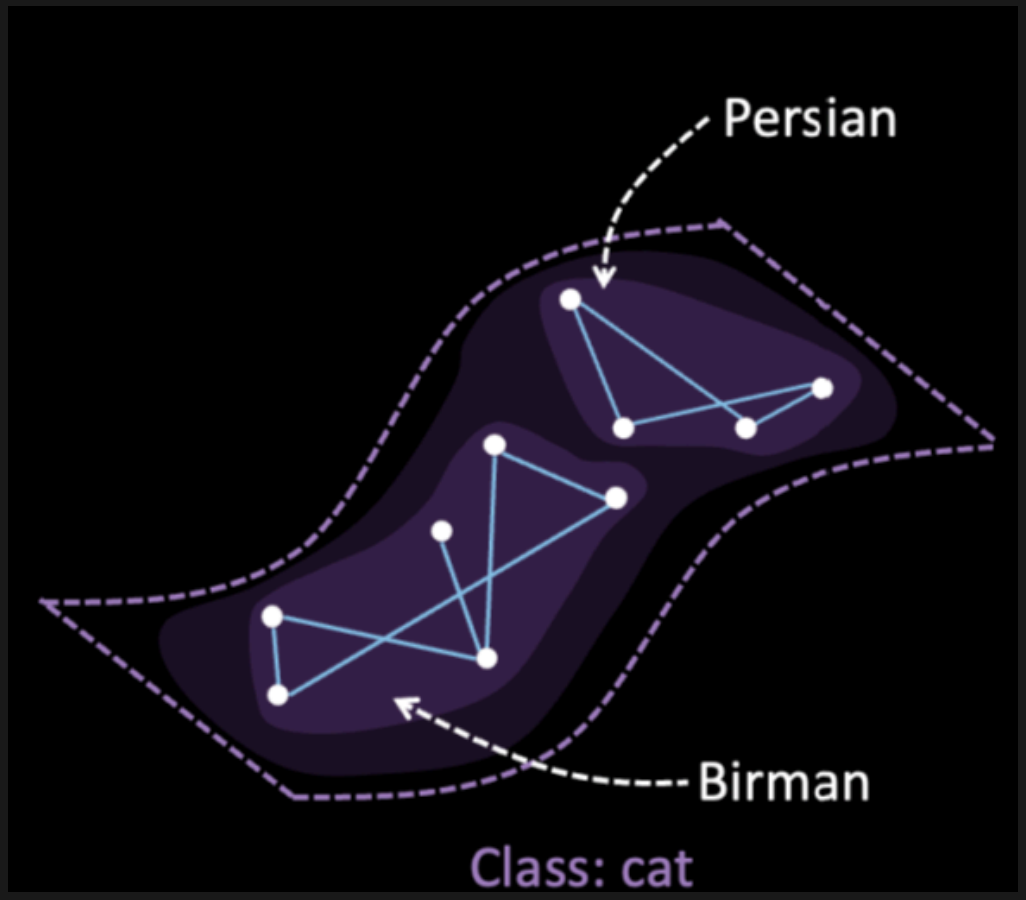
Vision Encoding

- Vision encoding maps images to some n dimensional linear space, known as a latent vector space [2]
- Related images will map onto manifolds (surfaces) in this space
  - specifically, the cosine similarity of encodings should be maximised when the images are similar

$$\text{Encode}(\text{image}) = \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix} \qquad \text{cosine similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

CarLLaVa

Vision Encoding



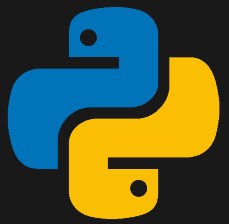
- Augmentations of the same photo should be very close to each other
- Photos of dogs map to the same surface, and the same is true for cats.
  - In reality, the manifold for any category will have a dimension much higher than 2

Figure 10: Encodings of Cat Pictures

Tools / Frameworks

CARLA

Pytorch



Python

Weights and Biases

CUDA

HuggingFace

CarLLaVA is the State of the Art

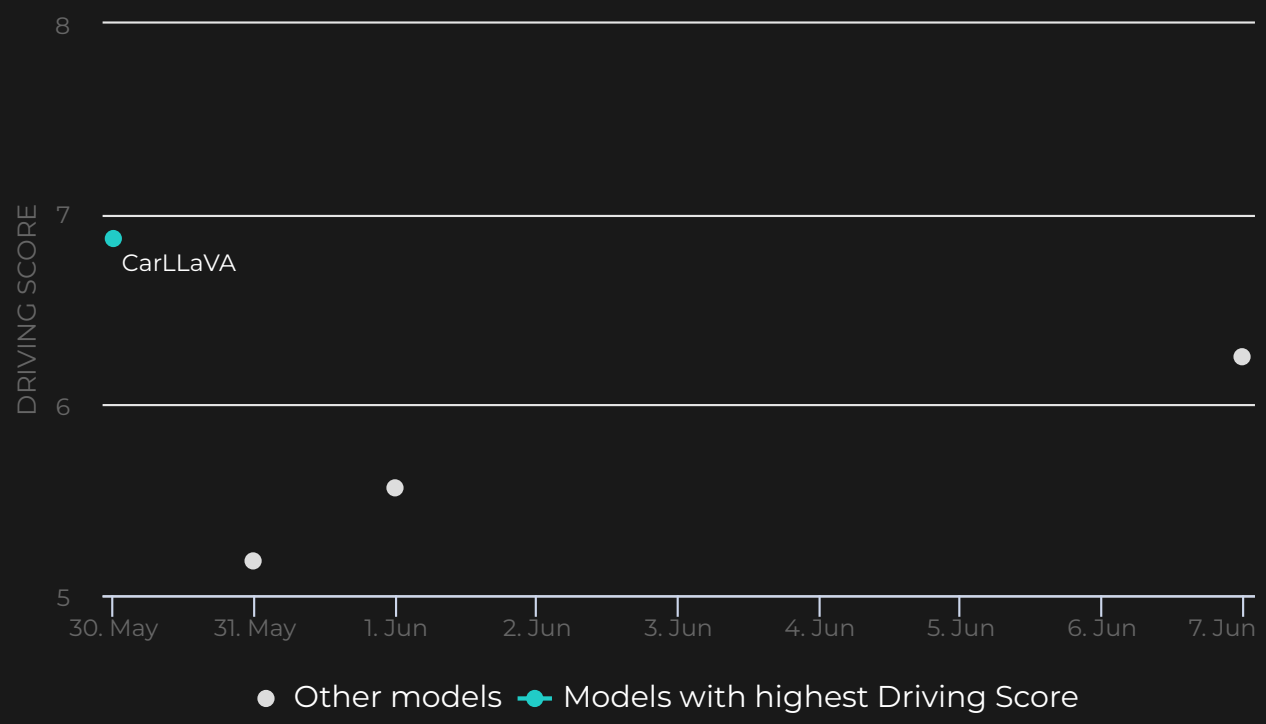


Figure 11: CARLA Leaderboard 2.0

- Dominates the leaderboard
- However, it has only been out for less than 6 months

Modifications

- if there is enough time remaining, there are a number of modifications we want to attempt

1. ViT with Registers
   - A study by Darcet et al [3] found a relatively simple modification that greatly improved the performance of ViT
2. Stereo vision
   - Internally, CarLLaVA already splits the visual field in two halves. Using two separate cameras, providing depth information may be useful to the model

Modifications

ViT with Registers

- Internally, the vision encoder in LLaVA is a form of vision transformer, or ViT
- ViT uses several transformer blocks chained sequentially
- Darcet et al.[3] added extra encoding dimensions (registers) to each block
  - and discarded them at the last layer
- Their model became much more robust, meaning that it based its decisions less on coincidences in the dataset and more on actual patterns

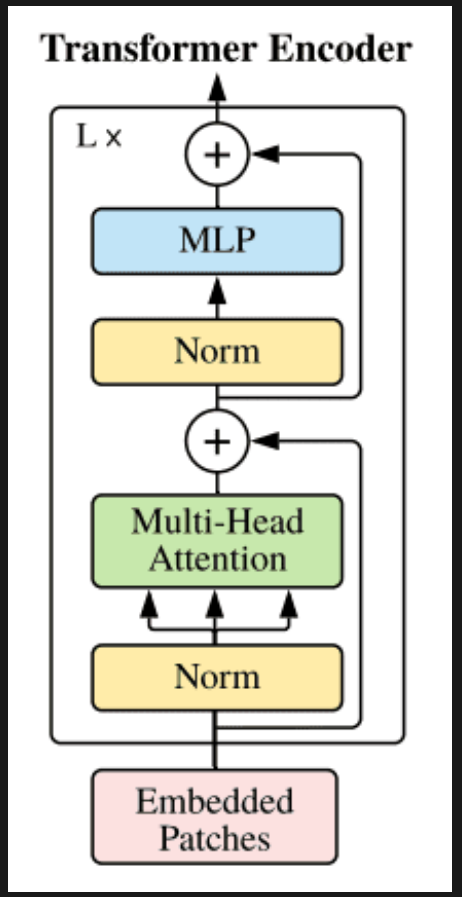


Figure 12: Transformer Structure

Modifications

Stereo Vision

- Stereo vision uses two cameras to guage depth and distance
- CarLLaVA's architecture seems to suggest it would naturally do well with information structured this way

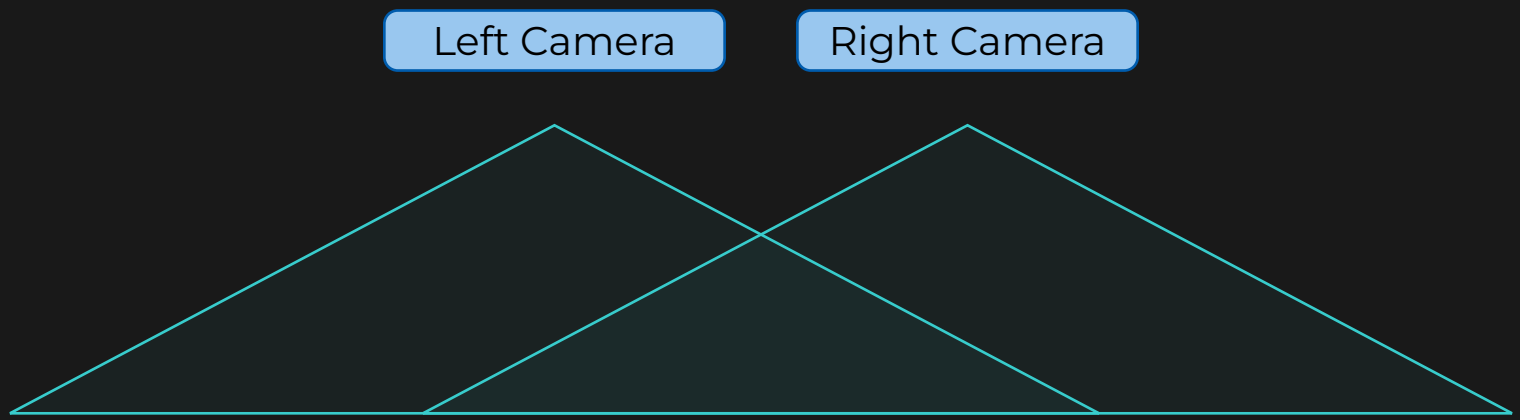


Figure 13: Stereo Vision

MEASUREABLE OUTCOMES

Training Metrics

- We can measure how closely the model can emulate the behaviour of the rule-based expert through training metrics like accuracy and loss
- Often, the dataset is split into 2-3 sections
  - Training
    - Trains the model, but represents the exact subset of data the model is familiar with
  - Evaluation
    - Used to periodically measure how generalizable the model is, and adjust hyperparameters
  - Test
    - Unseen by the model until the final evaluation, so that it only measures generalizability with no effect on the model itself

CARLA Metrics

Figure 14: Unreal 4 CARLA

- CARLA has 3 built in metrics
- RC: Route completion
  - How much of a route was completed
- IS: Infraction score
  - Negative score for infractions
- DS: Driver score
  - Aggregated RC and IS score, with compensation for IS when achieving higher RC scores

MAPS and SENSORS

- There are two commonly used tracks in CARLA, that contain a variety of scenarios and environments to navigate: MAPS and SENSORS
- These are the two tracks used for leaderboards

Figure 15: Highway in CARLA

References

Literature

[1] K. Renz *et al.*, «CarLLaVA: Vision language models for camera-only closed-loop driving». [Online]. Disponibile su: https://arxiv.org/abs/2406.10165

[2] J. Z. HaoChen, C. Wei, A. Gaidon, e T. Ma, «Provable Guarantees for Self-Supervised Deep Learning with Spectral Contrastive Loss». [Online]. Disponibile su: https://arxiv.org/abs/2106.04156

[3] T. Darcet, M. Oquab, J. Mairal, e P. Bojanowski, «Vision Transformers Need Registers». [Online]. Disponibile su: https://arxiv.org/abs/2309.16588

[4] H. Wang *et al.*, «OpenLane-V2: A Topology Reasoning Benchmark for Unified 3D HD Mapping», in *NeurIPS*, 2023.

[5] T. Li *et al.*, «LaneSegNet: Map Learning with Lane Segment Perception for Autonomous Driving», in *ICLR*, 2024.

Image Sources

https://www.youtube.com/watch?v=J4pHRM1Q5nQ
https://blog.tensorflow.org/2024/02/graph-neural-networks-in-tensorflow.html
https://arxiv.org/pdf/2406.10165
https://arxiv.org/abs/2106.04156
https://paperswithcode.com/sota/carla-leaderboard-2-0-on-carla
https://arxiv.org/abs/1706.03762
https://carla.org/