

VLMs-Guided Representation Distillation for Efficient Vision-Based Reinforcement Learning

Supplementary Material

7. Environment details

In VRL, the agent’s input is an RGB visual image. To ensure a fair comparison, the camera settings in Carla for all experiments are provided in Tab. 3. Since weather conditions can influence the quality of the visual RGB input, we standardized the weather settings across all experiments using the parameters listed in Tab. 4.

Our objective is to travel as far as possible within 500 steps without colliding [22, 23, 51, 57]. Consequently, the reward function for all our experiments is defined as follows:

$$r_t = \lambda_1 \cdot \frac{v_{agent}^\top u_{lane}}{\text{Tick}} - \lambda_2 \cdot \text{collision} - \lambda_3 \cdot \text{brake} - \lambda_4 \cdot |\text{steer}|. \quad (14)$$

Here, v_{agent} is the agent’s velocity vector. u_{lane} is the unit vector representing the direction of the lane at the agent’s current position and orientation. Tick is the simulation frequency (in Hz), which is equal to the capture frequency of the RGB camera. The coefficients $\lambda_1 = 1$, $\lambda_2 = 0.1$, $\lambda_3 = 10^{-3}$, and $\lambda_4 = 0.1$ are used to balance these components. The first term in the reward function motivates the agent to stay on the lane and travel as far as possible. The second, third, and fourth terms impose penalties for collisions, excessive braking, and overly sharp steering, respectively.

8. Additional implementation details

8.1. Network architecture

We provide the detailed network architecture of the visual encoder, policy decoder, and SSL module in our compact agent in Fig. 10, with the corresponding input and output symbols annotated in the figure.

In Fig. 10a, it is clear that our visual encoder is significantly smaller compared to the ResNet series [56]. We begin by applying four 2D convolutional layers (referred to as Conv) with ReLU activation to encode the original visual input s_t into a feature map f_t . As depicted in the figure, each Conv operation is specified with three standard parameters: kernel size, stride, and padding. The resulting feature map f_t has dimensions of (256, 6, 6). Next, f_t is flattened (Flatten) and transformed into a 512-dimensional feature vector z_t through a fully connected linear layer (Linear) followed by a LayerNorm operation.

Fig. 10b illustrates the policy decoder of the agent, consisting of three Linear layers and two ReLU activation functions. As shown in Fig. 10c, the two action-conditioned

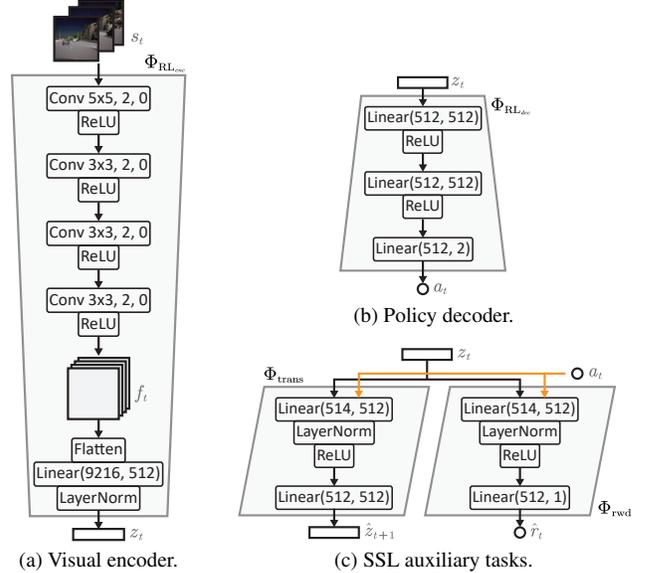


Figure 10. Detailed network architecture.

auxiliary networks, denoted as Φ_{trans} and Φ_{rwd} , take the feature vector z_t and the action a_t as inputs. Specifically, Φ_{trans} predicts the next-step feature vector \hat{z}_{t+1} , while Φ_{rwd} predicts the reward \hat{r}_t .

8.2. Hyperparameter settings

A full list of hyperparameters in DGC is provided in Tab. 5.

8.3. Details of PFT and RFT

In Sec. 5.2, we perform a comprehensive comparison with various state-of-the-art (SOTA) methods, offering a detailed analysis of VLM-based approaches. First, we adapt the prompts defined in Fig. 4 to align with the Executor method, enabling the VLM to directly output precise low-level control values. These modified prompts and the corresponding CoT examples are detailed in Fig. 11.

The CoT results indicate that VLMs can effectively identify the relationships between the agent, surrounding objects, and the task, allowing for accurate determination of general action values. However, at a fine-grained level, the actions lack precision and differentiation. To address this, the PFT method utilizes these action values as supervisory signals, enabling the agent to rapidly acquire the VLM’s decision-making capabilities. Nonetheless, further refinement through VRL remains essential to achieve more precise and nuanced decision-making.

Attributes	Value	Description
Image size	[128, 128]	Width and height of the image in pixels.
FOV	60	Horizontal field of view (FOV) of the camera.
Tick	20	The RGB camera’s capture frequency in hertz.
ISO	100	The RGB camera’s sensitivity to light.
Exposure brightness	10, 12	The minimum and maximum brightness adjustments of the RGB camera in response to changing lighting conditions.
Blur intensity	0.45	The strength of motion blur effect.
Shutter speed	200	The camera shutter speed in seconds.
Gamma	2.2	The gamma correction applied to the RGB camera’s output.
Lens flare	0.1	The intensity of lens flare effects in the camera view.

Table 3. Settings of the RGB camera.

Attributes	Value	Description
Cloudiness	20	The percentage of sky covered by clouds, ranging from 0 to 100.
Precipitation	0	The intensity of rain, from 0 (no rain) to 100 (heavy rain).
Precipitation deposits	0	The creation of puddles on the surface.
Wind intensity	0	No wind when the value is 0, with 100 representing strong wind.
Fog density	20	The concentration or thickness of fog, ranging from 0 to 100.
Fog distance	0	The starting distance of the fog.
Wetness	0	The presence of moisture on surfaces, simulating wet conditions.
Sun azimuth angle	0	The azimuth angle of the sun, ranging from 0 to 360 degrees.
Sun altitude angle	10	The altitude angle of the sun, ranging from -90 to 90 degrees.

Table 4. Settings of the Carla weather.

Hyperparameter	Value
RGB frame dimensions	$128 \times 128 \times 3$
Action repeat	1
Frame skip	1
Frame stack	3
Initial sampling steps	1,000
Maximum steps per rollout	500
Total training steps	100,000
Evaluation episodes	20
Replay buffer size	10,000
Initial α (SAC)	0.01
Optimizer (Actor, Critic, α)	Adam ($\beta_1 = 0.9, \beta_2 = 0.999$)
Optimizer (Visual encoder, Auxiliary tasks)	Adam
Batch size	32
Learning rate (Actor, Critic, α)	10^{-4}
Learning rate (Visual encoder, Auxiliary tasks)	10^{-4}
VLMs-guided supervision update frequency	5
Transition and reward auxiliary task update frequency	1
Actor update frequency	2
Critic target update frequency	2

Table 5. A full hyperparameter list used in DGC.

In the RFT method, the agent’s visual encoder is replaced with a CLIP pre-trained image encoder. While the pre-trained parameters are directly loaded, they are fully fine-tuned during the VRL training process. This approach harnesses the image understanding capabilities of the pre-trained VLM but still requires fine-tuning through environmental interactions to optimize performance.

9. Additional experiments

9.1. Model statistics

Tab. 6 provides a comparison of parameter sizes and inference speeds between VLM models for reasoning and referring tasks, SSL auxiliary networks, and our proposed model. The inference times were measured on an Nvidia A100. Since ChatGPT-4o is not open-sourced, its inference time is estimated based on the duration between API request submission and data receipt. It is clear that VLMs have significantly larger model sizes and longer inference times compared to our agent, with the reasoning VLM being the most complex, followed by the referring VLM. Additionally, the transition and reward predictive heads, serving as auxiliary tasks, are relatively small in size and can efficiently assist the agent in extracting task-relevant features.

Models	Params (M)↓	Inference Time (ms) ↓
Qwen-VL-Chat	9,657	11,365
Qwen2-VL-Chat (7B)	8,291	8,912
ChatGPT-4o-mini	N/A	5,099
CLIP (ViT-B/32)	151	64
CRIS	147	130
Transition head (Φ_{trans})	0.527	1.108
Reward head (Φ_{rd})	0.265	0.429
DGC (Ours)	6	1.502

Table 6. Model statistics.

9.2. Effect of prompts

LLMs (and more recently, VLMs) are highly sensitive to prompt quality [5]. To showcase the effectiveness of our predefined prompts, we present several intuitive examples. Our primary objective is to verify that the CoT reasoning process aligns with real-world Carla scenarios and conforms to human common-sense logic. A detailed step-by-step analysis for Figs. 4b and 11b is presented below.

- Step 1 demonstrates that the VLM outputs thoroughly capture all aspects of the scenario while maintaining close alignment with real-world contexts. Notably, the VLM exhibits a precise understanding of the objects’ **existence**

as well as their **spatial relationships**. For example, in the second case shown in Fig. 11b, the ego-vehicle is positioned very close to the white SUV ahead, with no steering adjustments and traveling at high speed. This corresponds closely to the description provided by the input image and the specified actions.

- Step 2 demonstrates that the VLM can anticipate multiple possible future outcomes based on the given input. For example, in the second case shown in Fig. 4b, two scenarios are depicted: one where the pedestrian successfully crosses the road, and another where the pedestrian hesitates and comes to a stop. Both scenarios are frequently observed in real-life situations. However, while the evaluation of potential collision risks in future scenarios is generally accurate, a limitation is evident in the first example of Fig. 11b, where the potential risk involving the ego-vehicle and the vehicle in the right lane is not identified, despite this vehicle being observed in Step 1.
- In Step 3, the ego-vehicle focuses on setting priorities, such as maintaining a safe distance, keeping a consistent speed, preparing to brake, or adjusting its position to the right. This step aims to clarify the interactions between the traffic participants and the ego-vehicle within the given scenario. For example, in the second example of Fig. 11b, the ego-vehicle plans to decelerate and maintain constant attention on the SUV ahead.
- In Step 4, based on the analysis from the previous three steps, the corresponding output results are naturally derived and conform to the predefined format. Specifically, Fig. 4b generates the semantic categories of key regions, while Fig. 11b produces executable low-level action values, both aligning with the output requirements specified by the prompts.

10. Future direction

VRL is a key approach in artificial intelligence that enables agents to learn and refine decision-making skills by interacting with their physical environments. These interactions shape how agents perceive and process their surroundings, directly influencing their behavior and decisions. Future research will focus on measuring how latent representations evolve over time and testing their generalization across various scenarios. Additionally, a deeper understanding of the physical properties of the 3D world is seen as essential for improving the cognitive and decision-making abilities of VRL systems. To achieve this, efforts are being directed toward combining agents’ physical trajectory data with visual information into a unified representation, enhancing predictive planning and enabling more effective decisions. Moreover, clarifying the capability boundaries of both VRL and VLMs is critical, as it provides valuable insights into how to design effective strategies for integrating the two, leveraging their complementary strengths.

Q: [Task Definition Prompt] ... [Common Sense Prompt] ... [Chain-of-Thought Prompt] ...

A: Certainly! Let's analyze the given scenario step-by-step.



A: 1. Describe the Environment:

Image 1: The ego-vehicle is on a multi-lane highway at night. There is moderate traffic with vehicles in adjacent lanes. The road is straight with clear lane markings, and a rock face is visible on the right.

Image 2: The scene remains consistent with the first image. The vehicle is maintaining its position relative to the other cars.

Image 3: No significant changes; the vehicle continues along the straight highway with the same traffic and environment.

Historical Actions:

- Steering (0.3, 0.34): Slight right adjustments, indicating minor lane centering or alignment.
- Throttle/Braking (-0.06, -0.08): Light braking, suggesting a decrease in speed to maintain safe distance or adjust to traffic flow.

2. Anticipate Future Outcomes:

The vehicle is likely to continue on the straight path, maintaining its lane. Traffic conditions appear stable, but vigilance is needed for any sudden changes. The slight braking action suggests an ongoing adjustment to traffic speed.

3. Set Priorities:

- Maintain Lane Position: Ensure the vehicle stays centered in its lane while adjusting to traffic.
- Monitor Traffic: Keep an eye on the vehicles ahead and in adjacent lanes for any sudden movements.
- Adjust Speed: Continue to adjust speed as necessary to maintain a safe following distance.

4. Identify Key Action:

To ensure safety and smooth operation, the ego-vehicle should maintain its current lane position and adjust speed as needed.

Critical Action: --[-0.1, 0.3]--

Throttle/Braking (-0.1): Apply slight braking to maintain a safe distance from the vehicle ahead.

Steering (0.3): Continue slight right steering to remain centered in the lane.

[Task Definition Prompt]

You will act as an autopilot expert in CARLA. You will be provided with a sequence of three front-camera images captured by the ego-vehicle, along with the corresponding historical actions. The action data consists of two elements: the first is steering, ranging from -1 to 1, where values less than 0 indicate a left turn, and values greater than 0 indicate a right turn. The second element relates to throttle and braking, where positive values represent throttle and negative values indicate braking. Your output should follow the structured reasoning outlined in the Chain-of-Thoughts prompt.

[Common Sense Prompt]

Avoid collisions with other traffic participants, such as vehicles or pedestrians. Be cautious of potential hazards. Follow all traffic rules. Proceed only when there are no dangers.

[Chain-of-Thought Prompt]

You should analyze the scenario and provide a step-by-step response according to the following guidance:

1. *Describe the Environment:* Start by describing the surrounding environment of the ego-vehicle based on the given images.
2. *Anticipate Future Outcomes:* Consider what future scenarios might arise given the current surroundings and historical actions.
3. *Set Priorities:* Determine what the ego-vehicle should prioritize in the near term based on its observations.
4. *Identify Key Action:* Based on the analysis, specify one critical action or focus area for the current observation. Present this in the format: --[throttle, steering]--.



A: 1. Describe the Environment:

Image 1: The ego-vehicle is on a multi-lane highway at night. There is moderate traffic with a white SUV directly ahead and other vehicles in adjacent lanes. The road is straight with clear lane markings.

Image 2: The ego-vehicle is closer to the white SUV. Traffic remains consistent, and there are no visible obstacles or pedestrians. The road continues straight.

Image 3: The ego-vehicle is very close to the white SUV, indicating a potential need to adjust speed to maintain a safe distance.

Historical Actions:

- Steering (0, 0): The vehicle is maintaining a straight path with no steering adjustments.
- Throttle (0.9, 0.85): High throttle values, suggesting the vehicle is accelerating or maintaining high speed.

2. Anticipate Future Outcomes:

The ego-vehicle is approaching the SUV ahead rapidly, which could lead to a collision if the distance is not adjusted. Continuation of high throttle without braking will reduce the safe following distance further. The road remains straight, so no immediate steering adjustments are necessary.

3. Set Priorities:

- Maintain Safe Distance: The primary focus should be on reducing speed to increase the distance from the SUV ahead.
- Monitor Traffic: Keep an eye on the behavior of the SUV and other vehicles for any sudden changes.
- Prepare for Braking: Be ready to apply brakes if the SUV slows down or traffic conditions change.

4. Identify Key Action:

To ensure safety, the ego-vehicle should decrease throttle and prepare to brake to maintain a safe following distance.

Critical Action: --[-0.5, 0]--

Throttle (-0.5): Apply braking to reduce speed and increase the distance from the vehicle ahead.

Steering (0): Maintain a straight path as no steering adjustment is currently needed.

(a) The predefined prompts for Executor.

(b) Some typical CoT examples of Executor.

Figure 11. Illustration of the predefined prompts for VLM-based Executor with the corresponding CoT reasoning examples.