

# Scaling Multi-Frame Transformers for End-to-End Driving

Vasileios Kochliaridis<sup>a</sup>, Filippos Moutzidellis<sup>b</sup> and Ioannis Vlahavas<sup>c</sup>

*School Of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece*  
vkochlia@csd.auth.gr, fmoutzi@csd.auth.gr, vlahavas@csd.auth.gr

**Keywords:** Autonomous Vehicles, Computer Vision, Deep Learning, Imitation Learning, Transformers

**Abstract:** Vision-based end-to-end controllers hold the potential to revolutionize the production of Autonomous Vehicles by simplifying the implementation of navigation systems and reducing their development costs. However, the large-scale implementation of such controllers faces challenges, such as accurately estimating object trajectories and making robust real-time decisions. Advanced Deep Learning architectures combined with Imitation Learning provide a promising solution, allowing these controllers to learn from expert demonstrations to map observations directly to vehicle controls. Despite the progress, existing controllers still struggle with generalization and are difficult to train efficiently. In this paper, we introduce CILv3D, a novel video-based end-to-end controller that processes multi-view video frames and learns complex spatial-temporal features using attention mechanisms and 3D convolutions. We evaluate our approach by comparing its performance to the previous state-of-the-art and demonstrate significant improvements in the vehicle control accuracy. Our findings suggest that our approach could enhance the scalability and robustness of autonomous driving systems.

## 1 INTRODUCTION

Inspired by the success of Transformer-based architectures in Natural Language Processing (NLP) tasks (Vaswani et al., 2017), several works in the Computer Vision domain have also integrated them, along with Convolutional Neural Networks (CNNs) (Carion et al., 2020). Transformers have shown an improved ability to process complex visual information, leading to more efficient Vision-Based End-To-End (VBETE) navigation controllers.


VBETE navigation controllers directly compute vehicle controls, including throttle, steering, and braking (Tampuu et al., 2020), using camera input. These systems have gained popularity due to their simpler implementation and reduced development costs, compared to traditional multi-layer decision-making modules, which fuse inputs from several sensors.


One notable VBETE controller that has demonstrated impressive performance in navigation tasks is CIL++ (Xiao et al., 2023), which combines cutting-edge Deep Learning methods, such as residual blocks and Transformer Encoders, with Imitation Learning (IL), in order to perform a wide range of driving tasks.


Although very promising, it lacks generalization ability in complex and densely populated environments, which include many vehicles and pedestrians. Furthermore, it presents instability in steering, making the vehicle to oscillate around the center of its lane.

CILRL++ (V. Kochliaridis, 2024) was proposed to address its weaknesses by employing the pretrained CIL++ model as the initial policy for a Deep Reinforcement Learning (DRL) agent, which is then fine-tuned through interaction with simulated environments. Although it is a more robust approach, it requires significant computational resources and training time to reach optimal results. Our investigation revealed that this stems from the sub-optimal conditions under which CIL++ was developed. First, excessive noise during data collection lead to steering control instabilities. Second, the lack of effective data augmentation techniques reduced the model’s generalization ability. Third, it lacks motion awareness, as it employs only observations from the present, completely disregarding past information, and thus making it difficult to drive properly. Finally, it utilizes a fast but outdated feature extractor for image processing.

In this paper, we present CILv3D, an improved version of the CIL++ that addresses its weaknesses and improves its driving performance, without increasing its training time. To achieve this, we first

<sup>a</sup>  <https://orcid.org/0000-0001-9431-6679>

<sup>b</sup>  <https://orcid.org/0009-0000-4753-290X>

<sup>c</sup>  <https://orcid.org/0000-0003-3477-8825>

rectify the dataset collection process by constructing a training dataset using the autopilot from the CARLA 0.9.15 simulator (Dosovitskiy et al., 2017). Moreover, we utilize sequential inputs, including consecutive vehicle controls and a video sequence for each view, which are further processed by the backbone model and 3D convolutions. Furthermore, we replaced its backbone model with UniFormer (Li et al., 2023), an also lightweight but more advanced Transformer-based model that extracts features from images more effectively. Finally, we investigate the use of several data augmentation techniques during training to address the generalization issues.

The remainder of this paper is organized as follows. Sections 2 and 3 present all the important literature relevant to this work. Section 4 provides a detailed explanation of our methodology, while Section 5 presents the simulation environment, the experimental setup, the metrics, and discusses the experimental results. Finally, Section 6 concludes this work and outlines the direction for future research.

## 2 RELATED WORK

In 2017, the CARLA team (Dosovitskiy et al., 2017) released the CARLA simulator, a realistic driving simulator that enables the development and testing of navigation algorithms. Moreover, they conducted a comparison of various methods for developing VBETE controllers, including end-to-end approaches, such as IL and DRL, with IL demonstrating superior performance in most of driving scenarios.

In a recent study by (Song et al., 2023), the authors introduced an improved version of initial CARLA team’s DRL approach. More specifically, the authors combined DRL with several feature extraction methods, such as the bird’s-eye-view technique and Convolutional layers, which effectively extracted comprehensive set of features from the RGB images. Despite the improved performance, their controller was only reliable in simple scenarios with low traffic density and without the presence of pedestrians.

CIRLS (Codevilla et al., 2019) was the first notable work in VBETE controllers, which employed an RGB camera and a Deep Neural Network to estimate the vehicle controls. It was trained on a limited dataset consisting of pairs of RGB images and expert vehicle control demonstrations, collected by an older version of CARLA’s autopilot module. Its architecture consists of a frozen ResNet-34 (He et al., 2016), which is used to extract feature maps from the input frames, and a Fully-Connected (FC) network, which processes these features and predicts the de-

sired vehicle controls, including steering, throttle, and braking. While this approach presented better driving skills than the DRL-based approaches, it suffers from overfitting, dataset biases due to several autopilot mistakes, and lack of generalization.

A more recent version of the CIRLS architecture, named CIL++ (Xiao et al., 2023), addressed several key limitations of its predecessor and improved the overall driving performance, without complicating the input space. First, the authors utilized a different autopilot system to construct the training dataset, named Roach (Zhang et al., 2021), which proved to be more smooth and accurate than the original CARLA’s autopilot. During dataset collection, the authors injected artificial noise into the vehicle’s control inputs to enhance generalization and reduce overfitting. Second, they employed three RGB cameras (left, front, and right) to capture a wider Horizontal Field Of View (HFOV) around the vehicle. Lastly, they integrated a Transformer Encoder, which enhances the important features from each frame, resulting in improved vehicle control predictions. Although their approach achieved state-of-the-art performance, it still faces generalization challenges and instabilities due to sub-optimal training conditions.

The CILRL++ approach (V. Kochliaridis, 2024) was proposed to address the instabilities of CIL++ by integrating IL with DRL. In this approach, the pre-trained model is further fine-tuned using a DRL algorithm called Phasic Policy Gradient (PPG) (Baker et al., 2022), along with a custom reward function designed to focus the DRL agent on rectifying the weaknesses of the initial model. To enable efficient fine-tuning with DRL, the authors employed a Kullback–Leibler Divergence loss, which address the Catastrophic Forgetting problem, where a DRL policy model forgets previously learned skills when undergoing new training. Although highly promising, this approach requires extensive training time and computational resources.

To conclude our literature review, various attempts have been made to develop end-to-end driving systems, including both IL and DRL, with most of them facing generalization issues in complex environments, as well as training difficulties. In our work, we aim to address these challenges by a) improving the data collection and training pipeline and b) introducing an improved network architecture.

## 3 BACKGROUND

This section provides a detailed description of two key architectures: the CIL++ architecture, previously

recognized as the state-of-the-art in VBETE driving controllers, and the UniFormer architecture.

### 3.1 CIL++ Architecture

CIL++ is a VBETE controller architecture that utilizes three RGB cameras (left, front, right). Each RGB image passes through a frozen ResNet-34 network, which extracts useful feature maps, each one represented by a  $5 \times 5 \times 512$  tensor. Then, each feature map is flattened and combined with positional embeddings and control embeddings, which are projected through a linear layer with 12,800 units, matching the size of the flattened feature maps, as illustrated in Figure 1. The combined features are then processed by Self-Attention blocks through a Transformer Encoder, which refines enhances the important features before passing them into the final FC network. Final, the FC network predicts the target steering and acceleration of the vehicle.

### 3.2 UniFormer

Designed for efficient image classification, UniFormer architecture combines CNNs with Self-Attention mechanisms by unifying them into a transformer architecture. In comparison to ResNet, which utilizes Convolutions and Residual Blocks only, it integrates local Multi-Head Self-Attention (MHSA) in the shallow layers and global MHSA in the deeper layers, enabling it to capture global token relations effectively, while reducing computational burden. UniFormer achieves fewer parameters and FLOPs compared to other large transformer models and heavier convolution-based architectures. For instance, ResNet-34 contains 21.89 million parameters and requires 3.68(G) FLOPs, while UniFormer has a comparable 22 million parameters but only 3.6(G) FLOPs. Despite the similar complexity, it significantly outperforms ResNet in the ImageNet classification task.

## 4 METHODOLOGY

Our methodology improves upon CIL++, by incorporating a Video-Based End-To-End approach, which can be structured into three core modules. The first module improves the dataset collection process. The second module designs a video-based neural network architecture, which is used to estimate the target vehicle controls. The third module involves the training of the neural network, which includes the preprocessing of the collected data and the data augmentation techniques.

### 4.1 Dataset Collection

Although the authors of CIL++ provide the datasets which they used to train their model, we identified two main issues during their collection methodology.

#### 4.1.1 Autopilot Expert

CIL++ authors used Roach autopilot as the expert driver, which relies solely on top-view cameras to observe the environment around the vehicle and generate controls. While the original CARLA autopilot had several limitations, it has been significantly improved, reducing its perception and decision-making errors. The updated CARLA autopilot now also utilizes internal simulator data, such as the precise location and direction of every vehicle and pedestrian within the simulation, which allows it to make more reliable and accurate decisions. Additionally, CARLA’s autopilot is more user-friendly and does not require fine-tuning.

#### 4.1.2 Vehicle Control Noise

Another issue of the dataset collection process was the injection of artificial noise into the vehicle’s control, during Roach’s operation. Although adding noise is a well-known regularization technique to help reduce overfitting, in this case, it had a negative effect on the CIL++ system. More specifically, the controller exhibited a tendency to cause the vehicle to oscillate around the center of the lane, struggling to maintain steady steering. This occurred because it learned these small deviations during Roach’s navigation, leading to an inability to consistently stay centered in the lane. To avoid this issue, we apply both noise and data augmentation only during the training process, as described later in this Section.

#### 4.1.3 Dataset

We collected a diverse dataset of approximately 80,000 expert demonstrations of driving data using CARLA 0.9.15. The dataset captures a wide range of driving scenarios across all available CARLA maps, including varying weather conditions, traffic densities, and urban layouts. Each demonstration includes:

- Three synchronized camera views (left, center, right), denoted as  $L_t, F_t, R_t$  respectively.
- Vehicle state information, which is a vector ( $v_t$ ) that includes the acceleration, steering, speed and command of the vehicle, denoted as  $a_t, s_t, u_t, c_t$  respectively.

The cameras are strategically positioned to cover the left, center, and right views, collectively providing 180-degree HFOV. This setup mimics the visual

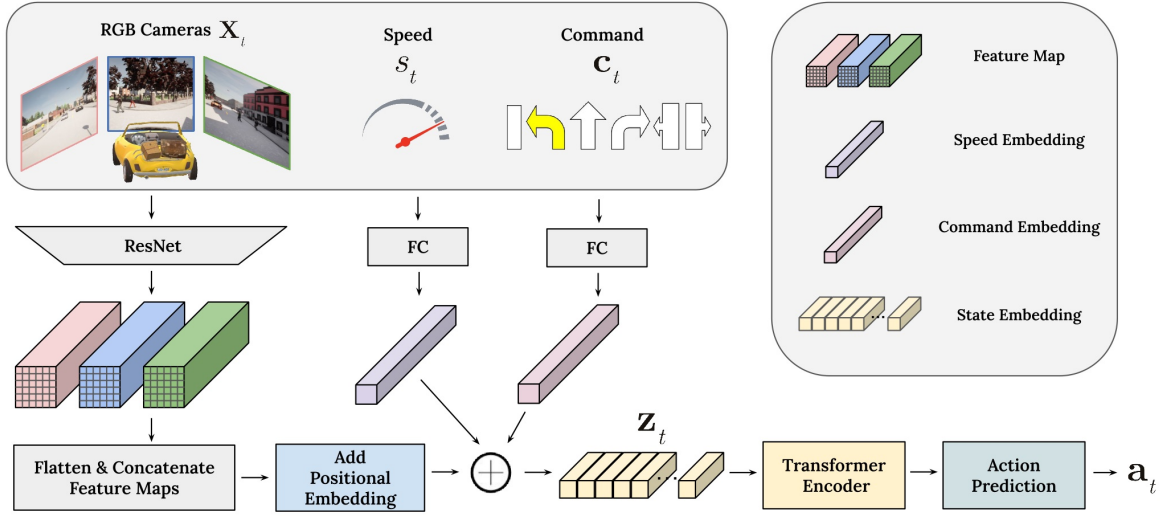


Figure 1: Overview of the original multi-view CIL++ architecture. It utilizes 3 RGB frames, along with speed and command embeddings, which are combined and pass through a Transformer Encoder.

range of human drivers and ensures comprehensive environmental perception (Xiao et al., 2023). The center camera is responsible for the front view while the other two are responsible for covering the peripheral view. This camera layout is crucial for the controller, in order to be able to perform tasks such as lane changing, avoiding obstacles or stopping at traffic lights and crossing pedestrians.

To ensure diversity in our dataset, we developed a randomization script that dynamically adjusts environmental conditions such as weather patterns, pedestrian presence and their behavior. Additionally, for each scenario, we varied the number and types of vehicles on the road, assigning each one random attributes like color, size, and destination. This approach is designed to create a robust dataset that captures a wide range of driving conditions, enhancing the model’s ability to generalize across diverse environments.

## 4.2 Network Architecture

Our network architecture is composed of four parts: Spatio-Temporal encoding, State Embedding, Transformer Encoder, and Action Prediction Network (Figure 2). At time  $t$ , the current input of the neural network consists of a set of video sequences  $SEQ_t = L_t, F_t, R_t$  of size  $N + 1$ , with each video sequence including the current and past  $N$  captured frames from each camera, as well as the vehicle control sequences, denoted as  $V_t$ . More specifically,  $L_t, F_t$  and  $R_t$  are the video sequences from left, front and right view respectively, while  $V_t$  is a matrix which includes the cur-

rent and the past  $N$  vehicle state information vectors  $v_t$ .

### 4.2.1 Vehicle State Representation

To extract temporal dependencies from the vehicle state  $V_t$ , we apply 1D convolutions with kernel size of  $N - 1$ . These convolutions are computationally efficient and can be executed without adding further computational burden. Then, we use a linear layer with 12,800 units to generate the final state embeddings. Similar to CIL++, these state embeddings are then added to the spatio-temporal embeddings extracted from each video sequence, creating a unified representation of the controller’s state.

### 4.2.2 Spatio-Temporal Encoding

Each frame is a  $224 \times 224$  RGB image containing raw pixel values in range  $[0, 255]$ . Acceleration is represented as a normalized value  $a_t \in [-1.0, 1.0]$ , where negative values indicate braking, while positive values indicate acceleration. Similarly, Steering is also a normalized value  $s_t \in [-1.0, 1.0]$ , with negative values and positive values indicating left and right steering respectively, while zero indicate that the vehicle is driving in a straight line. Finally, speed is real value  $u \in [0, 100]$ , while the navigation command  $c$  is a categorical variable with 6 distinct values: *Lane Follow*, *Turn Left*, *Move Forward*, *Turn Right*, *Change Lane Left*, *Change Lane Right*.

To extract spatial information from each frame, we replace the frozen ResNet-34 architecture with a frozen UniFormer model, which produces feature

maps of size  $B \times 7 \times 7 \times 512$ , where  $B$  denotes the batch size of the input. To capture the temporal dependencies within the video sequence, we apply a 3D convolution layer with 512 filters and kernel size of  $(N+1, N+1, N+1)$ . This generates the Spatio-Temporal Embeddings of the model, resulting in output of size  $B \times 5 \times 5 \times 512$ , which is then flattened into a vector of 12,800 features for further processing. For each flattened embedding vector corresponding to the left, front and right camera sequences, we add their respective positional Embeddings along with the vehicle State Embeddings. Finally, these Embeddings are, we concatenated vertically, resulting in a tensor of  $B \times 3 \times 12800$ , the same as in CIL++, which is then passed into a Transformer Encoder.

#### 4.2.3 Transformer Encoder

The Transformer Encoder consists of  $K$  connected Transformer Blocks, with each block producing  $M$  head units, where  $K$  and  $M$  are user-defined settings. The final Transformer Block outputs a flattened vector of size  $M$ , which is passed into the Action Prediction Network (APN), which is a Fully-Connected network. The final layer of the APN is a linear layer with 2 units, corresponding to the desired outputs: the next step acceleration and steering, denoted as  $a_{t+1}$  and  $s_{t+1}$  respectively.

#### 4.2.4 UniFormer Performance

In our study, we utilized Grad-CAM (Gradient-weighted Class Activation Mapping) (Selvaraju et al., 2020) to analyze and compare the key image regions prioritized by UniFormer and ResNet-34 during their classification tasks in the CARLA simulator. Grad-CAM generates heatmaps that highlight the pixels that are considered significant by a model for making predictions, providing insights into the quality of its feature maps. The visualizations in Figure 3 demonstrate that UniFormer focuses more on critical features, such as traffic lights, vehicles, and their surrounding areas, resulting in more precise feature representations and enabling the navigation controller to produce more accurate vehicle controls.

### 4.3 Training

For our preprocessing and training setup, we apply several transformations to the three video sequences and the vehicle state. Since the UniFormer model is pretrained on ImageNet, we apply the ImageNet normalization, using a mean of  $[0.485, 0.456, 0.406]$  and a standard deviation of  $[0.229, 0.224, 0.225]$  to standardize the input pixels. The acceleration and steering

inputs are already normalized within a range of -1 to 1, but speed, which ranges from 0 to 100, is normalized using Min-Max formula (Equation 1). Finally, the control commands, are one-hot encoded.

$$u' = \frac{u - u_{\min}}{u_{\max} - u_{\min}} \quad (1)$$

To reduce overfitting and improve the model's generalization, we introduced Gaussian noise to the timeframe data and applied various data augmentation techniques to each frame. After experimenting with several methods, we selected the augmentations that worked best for our model. These include rescaling the images to 224x224 (which is the input size for UniFormer), adjusting brightness and contrast, applying Gaussian noise, and using blurring. Certain augmentations were discarded as they negatively affected the model's performance and slowed-down the training. For instance, center cropping removed important visual information from images, so it was replaced with rescaling. Rotation was irrelevant because the vehicle's cameras have a fixed position, color jittering caused confusion by changing the colors of critical elements like traffic lights and lane markings and horizontal flipping was impractical as the steering wheel is always on the left side.

## 5 EXPERIMENTS & DISCUSSION

The objective of our experimental is (a) to assess the benefit of each architectural enhancement, (b) to compare our approach with the previous state-of-the-art methods, including CIL++, CILRL and CILRL++.

Before training CILv3D, we randomly split the training dataset into 80% (64000) of sequences for the training set and 20% (16000) for the test set. Additionally, we conduct hyperparameter tuning to further optimize its performance.

Once the model is trained, we evaluate it on multiple tasks specified by the CARLA leaderboard simulator under diverse weather conditions. These tasks include:

- Lane-merging
- Negotiations at junctions
- Managing traffic lights and signs
- Yielding to emergency vehicles
- Handling pedestrians, cyclists, and other dynamic elements

Moreover, we construct a random scenario in Town10 of the CARLA simulator to assess the performance of the developed controller on general-purpose

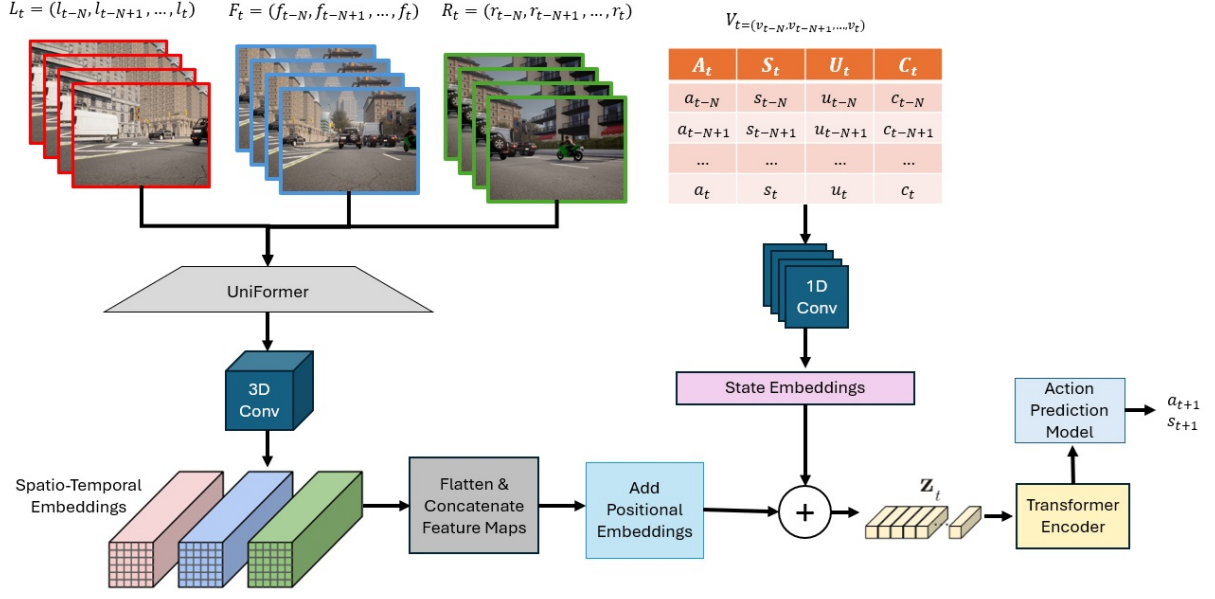


Figure 2: Overview of the multi-frame CILv3D architecture.

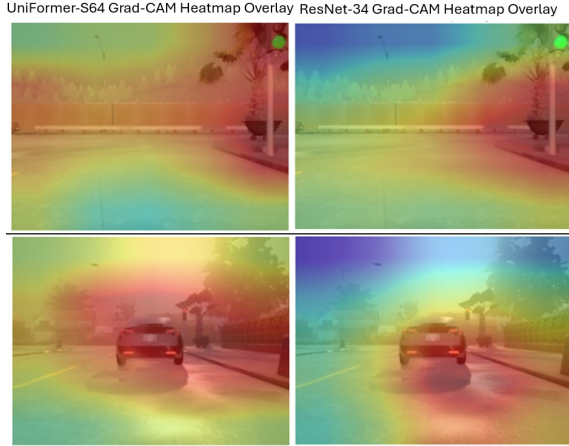


Figure 3: Grad-CAM visualizations of UniFormer and ResNet-34. Red and orange areas indicate regions of high importance, while green and blue areas reflect less significance. UniFormer can even identify small details inside an image, such as the vehicle in the left edge of the top image.

navigation tasks. In this scenario, we spawn 100 vehicles and 40 pedestrians in random locations and issue random commands and navigation waypoints to CILv3D, which navigates a vehicle. We record its navigation route in the CARLA simulator and have made a video available on the YouTube platform<sup>1</sup>. We also provide the implementation code in Github platform<sup>2</sup>.

<sup>1</sup>[https://www.youtube.com/watch?v=65k9P3mIkcY&ab\\_channel=CiLv3D](https://www.youtube.com/watch?v=65k9P3mIkcY&ab_channel=CiLv3D)

<sup>2</sup><https://github.com/kochlisGit/CILv3D>

## 5.1 Hyperparameter Tuning

Due to the time-consuming nature of Hyperparameter tuning, we manually conducted a limited search only on the most critical hyperparameters. These include the sequence size  $N$ , the number of transformer blocks ( $K$ ) and the transformer head units ( $M$ ), as well as the 1D convolution filters used during the vehicle State Representation construction stage.

During this process, we trained the neural network for 50 epochs with each configuration and selected the parameters that yielded the lowest Mean Absolute Error (MAE). The sequence size  $N$  was set to 4 after experimenting with values ranging from 2 to 8, as this provided a good balance between accuracy, model complexity, and performance. The parameters  $K$  and  $M$  were set to 4 and 2048, respectively, following experimentation with  $K$  values from 1 to 6 and  $M$  values from  $\{256, 512, 1024, 2048\}$ . Finally, the best number of 1D convolution filters was found to be 512.

The rest of the training configuration is the same as CIL++. More specifically, we train the model for 1000 epochs using the Adam optimizer with an initial learning rate of 0.001 and weight decay of 0.99. The goal of Adam is to minimize the MAE of both the acceleration ( $a_{t+1}$ ) and the steering  $s_{t+1}$ , as described by equation 2.

$$\text{MAE}_{\text{total}} = \frac{1}{S} \sum_{i=1}^S \left( \text{MAE}_{a_{t+1}}^{(i)} + \text{MAE}_{s_{t+1}}^{(i)} \right) \quad (2)$$

where  $S$  is the dataset size. The selected hyperparameters can be summarized by Table 1.



Table 1: CILRL++ Configuration

Parameter	Value
Sequence Size ( $N$ )	4
Transformer Blocks ( $K$ )	4
Discount factor ( $M$ )	2048
1D Conv Filters	512
3D Conv Filters	512
Loss	MAE
Optimizer	Adam
Learning Rate ( $lr$ )	0.001
Weight Decay	0.99
Epochs (e)	1000

## 5.2 Metrics

Similar to the CIL++ approach, we use the MAE as a metric to assess how effectively the constructed neural network learns from expert demonstrations during training. Low MAE in both the training and test set provides an initial indication of how well the model might perform in navigation tasks.

To further evaluate its driving performance, we utilize the suggested metrics by the leaderboard module, which are outlined below.

1. **Avg. Driving Score:** the most important metric of the Leaderboard, which is defined as  $R_i \cdot P_i$ , where  $R_i$  and  $P_i$  express the route completion percentage and the infraction penalty respectively.
2. **Avg Route completion (%):** the completed route distance.
3. **Avg Infraction Penalty:** the maximum value of this score is 1.0 and is reduced each time the driving controller commits an infraction.
4. **Collisions with Vehicles:** the collision count with other vehicles.
5. **Collisions with Pedestrians:** the collision count with walking pedestrians.
6. **Collisions with Layout:** the collision count with objects (e.g. buildings, pavement, etc.).
7. **Red Light Infractions:** the number of red lights ignored by the vehicle.
8. **Stop Sign Infractions:** the number of stop signs ignored by the vehicle.
9. **Off-Road Infractions:** the number of times where the vehicle deviated from its target lane.
10. **Route Deviations:** the number of times where the vehicle deviated from the desired route.
11. **Route Timeouts:** the number of times where the vehicle failed to complete its route within a specified amount of time.



Figure 4: Training loss (MAE) per training epoch.

12. **Agent Blocked:** the total time during which the vehicle was inoperative.

## 5.3 Results & Discussion

The performance of each architectural enhancement is presented in Figure 4 for the training set and in Table 2 for the test dataset. It can be observed that starting from around epoch 200, CILv3D shows notably more stable training behavior with fewer fluctuations compared to other models. The CIL++ approach exhibits oscillating loss throughout the training period, while struggling to generalize effectively on the test set. When ResNet-34 is replaced by UniFormer (CIL++ - UNI), the model requires more epochs to converge, but the overall performance is improved. A significant improvement in convergence speed is also observed when sequential inputs are introduced (CIL++ - 3D). The best overall performance is achieved by CILv3D, which combines all architectural enhancements described in Section 4.

Table 2: The Train-Eval Loss (MAE) of each architecture for the Imitation Learning process.

Architecture	Training Loss	Validation Loss
CIL++	0.0107	0.01387
CIL++ - Uni	0.0101	0.01180
CIL++ - 3D	0.0097	0.01184
CILv3D	<b>0.0092</b>	<b>0.01059</b>

The results of the evaluation experiments in the CARLA simulator are shown in Table 3. CILv3D slightly outperforms CILRL++ in several Leaderboard challenges, without requiring additional fine-tuning using DRL. On the other hand, CILRL++ achieved the best performance in a few specific tasks, such as reducing collisions with pedestrians and the environment, as well as receiving fewer infraction penalties. This is due to the extensive fine-tuning applied to CIL++ for these particular scenarios, with

Table 3: Evaluation Table of a) CIL++, b) CILRL, c) CILRL++ and d) CILv3D using CARLA Leaderboard.

Score	CIL++	CILRL	CILRL++	CILv3D
Avg. Driving Score (%)	2.59	0.09	10.01	<b>11.73</b>
Avg. Route Completion (%)	10.93	0.29	14.44	<b>15.12</b>
Avg. Infraction Penalty	0.44	0.47	<b>0.64</b>	0.59
Collisions with Vehicles	<b>0.0</b>	<b>0.0</b>	<b>0.0</b>	<b>0.0</b>
Collisions with Pedestrians	256.21	2759.67	<b>52.82</b>	128.43
Collisions with Layout	411.01	14496.83	<b>255.22</b>	285.66
Red Light Infractions	9.34	<b>0.0</b>	7.98	8.83
Stop Sign Infractions	5.77	846.62	<b>0.0</b>	2.25
Off-Road Infractions	253.67	1457.57	186.47	<b>177.53</b>
Route Deviations	<b>0.0</b>	144.4	76.74	55.81
Route Timeouts	<b>0.0</b>	330.96	<b>0.0</b>	<b>0.0</b>
Agent Blocked	399	10985.68	222.65	<b>198.69</b>

prolonged scaling. Despite this, CILv3D remains the best-performing approach overall, achieving the highest average driving score and route completion with less amount of training time.

## 6 CONCLUSION & FUTURE WORK

In this paper, we present CILv3D, which addresses several key limitations of CIL++ by refining the dataset collection process, incorporating multi-frame and sequential inputs and utilizing a more advanced backbone model. Furthermore, we address the generalization issues by exploring effective data augmentation techniques and injecting noise during the training process.

Our experimental results in several scenarios included in the CARLA Leaderboard indicate that CILv3D achieves higher driving score than CIL++ and CILRL++, in terms of overall driving score and route completion. Notably, CILv3D achieves these improvements without the need for extensive fine-tuning techniques, such as utilizing DRL methods, resulting in reduced training time and computational requirements. Although CILv3D demonstrates promising results, several directions for future research could further enhance its navigation performance, such as a 360° view of the environment, which could potentially improve the controller’s decisions at lane change tasks.

## REFERENCES

- Baker, B., Akkaya, I., Zhokov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. (2022). Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654.
- Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer.
- Codevilla, F., Santana, E., López, A. M., and Gaidon, A. (2019). Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9329–9338.
- Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. (2017). Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- Li, K., Wang, Y., Zhang, J., Gao, P., Song, G., Liu, Y., Li, H., and Qiao, Y. (2023). Uniformer: Unifying convolution and self-attention for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10):12581–12600.
- Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2020). Grad-cam: visual explanations from deep networks via gradient-based localization. *International journal of computer vision*, 128:336–359.
- Song, Q., Liu, Y., Lu, M., Zhang, J., Qi, H., Wang, Z., and Liu, Z. (2023). Autonomous driving decision control based on improved proximal policy optimization algorithm. *Applied Sciences*, 13(11):6400.
- Tampuu, A., Matiisen, T., Semikin, M., Fishman, D., and Muhammad, N. (2020). A survey of end-to-end driving: Architectures and training methods. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4):1364–1384.
- V. Kochliaridis, E. Kostinoudis, I. V. (2024). Optimizing pretrained transformers for autonomous driving. In *13th EETN Conference on Artificial Intelligence (SETN)*. ACM.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- Xiao, Y., Codevilla, F., Porres, D., and López, A. M. (2023). Scaling vision-based end-to-end autonomous driving with multi-view attention learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1586–1593. IEEE.
- Zhang, Z., Liniger, A., Dai, D., Yu, F., and Van Gool, L. (2021). End-to-end urban driving by imitating a reinforcement learning coach. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15222–15232.