Journal of Robotics and Control (JRC)
Volume 5, Issue 3, 2024
ISSN: 2715-5072, DOI: 10.18196/jrc.v5i3.21713

Enhance Deep Reinforcement Learning with Denoising Autoencoder for Self-Driving Mobile Robot

Gilang Nugraha Putu Pratama [1*], Indra Hidayatulloh [2], Herman Dwi Surjono [3], Totok Sukardiyono [4]
[1,2] Department of Electrical and Electronics Engineering, Universitas Negeri Yogyakarta, Indonesia
[3,4] Department of Electronics and Informatics Engineering Education, Universitas Negeri Yogyakarta, Indonesia
Email: [1] gilang.n.p.pratama@uny.ac.id
*Corresponding Author

***Abstract*—Over the past years, self-driving mobile robots have captured the interest of researchers, prompting exploration into their multifaceted implementation. They have the potential to revolutionize transportation by mitigating human error and reducing traffic accidents. The process of deploying self-driving mobile robots can be divided into several steps, such as algorithm design, simulation, and real-world application. This research paper presents a simulation using DonkeyCar on the Mini Monaco track, employing a Soft Actor-Critic (SAC) alongside a denoising autoencoder. At this point, it is limited to the simulation, serving as a proof of concept for further research with hardware implementation. The simulation verifies that relying solely on SAC for the convergence of policy is not sufficient; it yields a mean episode length of only 28.82 steps and a mean episode reward of 0.7815. The simulation ended after 3557 steps due to the inability of SAC alone to converge, without completing a single lap. Later, by integrating the denoising autoencoder, convergence of policy can be achieved. It enables DonkeyCar to adeptly track the lane of the circuit. The denoising autoencoder plays an important role in accelerating the convergence of transfer learning. Notably, the mean reward per episode reached 2380.4387, with an average episode length of 771.71 and a total of 114357 steps taken. DonkeyCar manages to complete several laps. These results affirm the effectiveness of SAC with a denoising autoencoder in enhancing the performance of self-driving mobile robots.**

***Keywords*—Self-Driving Mobile Robot; Deep Reinforcement Learning; Donkeycar Simulation; Soft Actor-Critic; Denoising Autoencoder.**

I. Introduction

In recent years, research on autonomous systems has grown rapidly with many breakthroughs and innovations, ranging from autonomous cars, underwater vehicles, to unmanned aerial vehicles [1]–[5]. One of the topics that captures our attention is self-driving mobile robots, which hold great potential for addressing transportation problems and revolutionizing them [6]–[8]. They have the capability to mitigate human errors that lead to traffic accidents [9]–[11]. Self-driving mobile robots integrate deep reinforcement learning to further advance their capabilities [12]–[14]. There are various deep reinforcement algorithms, some of which are Deep Q-Network (DQN) [15]–[17], Deep Deterministic Policy Gradients (DDPG) [17]–[20], Policy Gradient Method [21]–[23], Proximal Policy Optimization (PPO) [24]–[27], Deep Deterministic Policy Gradient (DDPG) [28]–[30], Twin Delayed DDPG (TD3) [31]–[33], ans Soft Actor-Critic [34]–[36]. Among the various algorithms for deep reinforcement learning, Soft Actor-Critic (SAC) has proven to be a fast-converging and robust solution for training autonomous agents in complex and dynamic environments [37]–[42].

Some notable studies in robotics that utilize SAC are as follows. Wong *et al*. prevail in implementing SAC for motion planning of dual-robotic arms. Each arm is designed with 7 degrees of freedom (DoF), capable of avoiding self-collision at the same moment keeping the arms from singularities [38]. Meanwhile, Mustafa *et al*. introduce deep reinforcement learning for speed control of ultrasonic motors. It provides a Lyapunov-based reward function for SAC to verify the stability of those ultrasonic motors [39]. Lastly, Hong *et al*. conduct a simulation for stabilizing Furuta pendulum using SAC with cosine reward function. After obtaining the satisfactory results, then they transfer it to the real world. It can be justified that the transfer learning is a success, the real Furuta pendulum can be stabilized and robust enough against external disturbances [40]. Chisari *et al*. have devised a simulation for self-driving mobile robots utilizing SAC, which outperforms alternative algorithms like Model Predictive Control [43]. Those results prove that SAC can be employed for autonomous agents and various robotics applications. The rationale for choosing SAC lies in its suitability for tasks with continuous action spaces, enabling effective handling of complex and high-dimensional action spaces. Additionally, SAC typically exhibits greater sample efficiency compared to other algorithms, resulting in comparable performance with fewer samples. Using SAC (Soft Actor-Critic) can be beneficial for self-driving mobile robots that rely on camera data, especially when dealing with continuous action spaces. SAC is particularly well-suited for problems with continuous action spaces, such as steering and throttle

control in self-driving tasks. This algorithm can effectively handle the complexity of continuous actions and learn policies that enable smooth and accurate control of the vehicle based on the input from the camera. Therefore, employing SAC can be advantageous for self-driving mobile robots that operate in environments where continuous data from cameras are crucial for navigation and decision-making. Moreover, we also include pre-trained data using a denoising autoencoder (DAE) to enhance policy convergence.

An autoencoder is a type of artificial neural network used for unsupervised learning and dimensionality reduction [44]–[46]. It is designed to encode input data into a compressed representation and then decode it back to the original form. The primary objective of an autoencoder is to learn a compact and meaningful representation of the input data [47]–[49]. Specifically, a denoising autoencoder (DAE) is one kind of neural network that learns efficient data representations by training on purposely corrupted input data [50]. Similar to a typical autoencoder, the DAE consists of an encoder and a decoder, minimizing the reconstruction error between the original input and output. During training, input data are intentionally corrupted with noise, encouraging the model to focus on essential features resilient to such distortions. This unique training objective makes DAE particularly effective for tasks like image denoising and feature learning [51]. Later, by emphasizing robustness, the DAE can enhance generalization and find applications in various domains where data may exhibit noise or imperfections [52]–[54].

Adding a DAE to SAC enhances policy convergence in self-driving mobile robot by reducing noise, learning important features from input data, reducing dimensionality, and leveraging transfer learning. The DAE preprocesses noisy sensor data, such as camera images, by reducing noise and extracting relevant features. This results in more accurate state representations and faster learning. Additionally, pre-trained DAE can transfer knowledge from large datasets, further improving policy convergence and overall performance. The research contributions include the enhancement of performance in self-driving robot simulations through the integration of DAE with SAC, demonstration of the effectiveness of the combined approach in successfully navigating the track, and highlighting the potential for real-world application of SAC and DAE in self-driving technology.

The rest of this paper is structured as follows: Section II provides an overview of the methods employed in this study, including the DAE and SAC. Moving forward, Section III delves into the results obtained from the experiments. This section not only presents the outcomes but also details the environment setup and the simulations conducted, both with SAC alone and with the integration of SAC and DAE. Lastly, the conclusion of this paper is presented in Section IV, summarizing the findings and discussing the implications of the study.

II. METHODS

Here, we will delve into two subjects. Subsection II-A presents the concept of autoencoder, including denoising autoencoder. Meanwhile, subsection II-B provides a brief overview of SAC.

A. Denoising Autoencoder

Before explaining further about DAE, let us briefly explain the autoencoder. Autoencoder is a of kind artificial neural network used for unsupervised learning, which is mainly used for dimensionality reduction, feature learning, and data compression [55]–[58]. It has two main parts, namely an encoder and a decoder as depicted in Fig. 1.

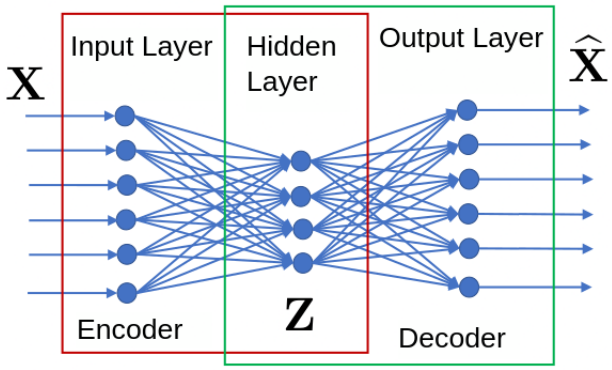

Fig. 1. Architecture of autoencoder.

The encoder (E) takes the input data and maps it into a lower-dimensional representation. It can be achieved through a series of hidden layers, typically fully connected layers, where each layer extracts increasingly abstract features from the input data denoted as $\mathbf{x}$. The output of the encoder, denoted as $\mathbf{z}$, is referred to as the latent space or bottleneck layer. It contains a compressed representation of the input data. This representation ideally captures the most important features of the input data [59]. Mathematically, the relationship between the encoder that being parameterized by α, input data, and latent space can be defined as

$$\mathbf{z} = E_{\alpha}(\mathbf{x}). \tag{1}$$

The later part is the decoder (D), which takes the compressed representation from the latent space and attempts to reconstruct the original input data. It performs the reverse operation of the encoder by mapping the latent representation back to the reconstructed input space using parameter β [60]. It can be formulated as

$$\hat{\mathbf{x}} = D_{\beta}(\mathbf{z}), \tag{2}$$

where $\hat{\mathbf{x}}$ is the reconstructed data. The decoder typically mirrors the structure of the encoder but in reverse, with each layer expanding the dimensions back to the original input size.

Usually, both are multilayer perceptrons. A single layer encoder E_α can be defined as

$$E_\alpha(\mathbf{x}) = \delta(\mathbf{W}x + b), \tag{3}$$

where $x \in \mathbf{x}$, $\mathbf{W}$ is a weighted matrix, and b is bias. Meanwhile δ is an activation function that usually either a sigmoid function or a rectified linear unit.

It yearns to to minimize the reconstruction error between the input data $\mathbf{x}$ and the output data $\hat{\mathbf{x}}$, therefore the autoencoder should be trained. In order to evaluate it, we need a task based on the reference probability distribution and reconstruction quality function. Let μ_{ref} be a probability distribution over $\mathbf{x}$ and $d : \mathbf{x} \times \mathbf{x} \rightarrow [0, \infty]$ be a reconstruction function. Hence, we can measure how much $\widehat{\mathbf{x}}$ differs from $\mathbf{x}$ based on $d(\mathbf{x}, \widehat{\mathbf{x}})$. After obtaining μ_{ref} and d, then we can define the loss function such as

$$\mathcal{L}(\alpha, \beta) = \mathbb{E}_{x \sim \mu_{\text{ref}}} \left[d(x, D_\beta(E_\alpha(x))) \right], \tag{4}$$

where $\mathbb{E}$ is the expected value operator.

After explaining the preliminaries, now we can discuss the DAE. it is originally called robust autoassociative network [61]. Technically, the main difference between a typical autoencoder and DAE is the input data. Despite using the original data to yield the latent space like a typical autoencoder, DAE uses intentionally corrupted data [62]. It makes sense since the aim is to learn a representation of the original data that is robust against noise, by training on corrupted input data and reconstructing the original [63]. Let us define the noise process based on the probability distribution μ_T over $T : \mathbf{x} \rightarrow \tilde{\mathbf{x}}$. Here, T is a function that maps the original data $x \in \mathbf{x}$ to corrupted one $\tilde{x} \in \tilde{\mathbf{x}}$.

Simply, by substituting the original input data ($\mathbf{x}$) with the corrupted one ($\tilde{\mathbf{x}}$) in equation (1), we have the latent space for DAE as

$$\mathbf{z} = E_\alpha(\tilde{\mathbf{x}}). \tag{5}$$

Furthermore, by considering the equation (2) and (5) we can define the reconstructed data $\widehat{\mathbf{x}}$ for DAE such as

$$\widehat{\mathbf{x}} = D_\beta(E_\alpha(\tilde{\mathbf{x}})). \tag{6}$$

Meanwhile, the training process can be done by solving the optimization problem such as

$$\min_{\alpha,\beta} \mathcal{L}(\alpha, \beta) = \mathbb{E}_{x \sim \mu_{\mathbf{x}}, T \sim \mu_T} \left[d(x, (D_\beta \circ E_\alpha \circ T)(x)) \right], \tag{7}$$

Those steps allow DAE to effectively remove noise from input data during the reconstruction process [64]. Essentially, DAE differs from typical autoencoders by reconstructing corrupted input, removing noise, and focusing on resilient features. This is particularly advantageous for self-driving mobile robots, where noisy sensor data can affect perception accuracy. DAE enhances perception by cleaning noisy inputs and extracting relevant features, thus improving the ability to interpret data and make informed decisions in complex environments.

B. Soft Actor-Critic

SAC, renowned for reinforcement learning tasks with continuous action spaces, is an off-policy optimization method designed for stochastic policies [65]–[67]. The policy undergoes training aimed at maximizing the balance between anticipated returns and entropy, which quantifies policy randomness and is directly tied to the exploration-exploitation dilemma. As entropy rises, the model is encouraged to explore more, potentially accelerating the learning process [68]. This approach also safeguards against premature convergence to suboptimal solutions [69]. In general, SAC can be described as in algorithm 1 Soft Actor-Critic.

Algorithm 1 Soft Actor-Critic

Initialize parameter vectors ψ, $\bar{\psi}$, θ, ϕ
for each iteration **do**
  for each environment step **do**
    $\mathrm{a}_t \sim \pi_\phi(\mathrm{a}_t \mid \mathrm{s}_t)$
    $\mathrm{s}_{t+1} \sim p(\mathrm{s}_{t+1} \mid \mathrm{s}_t, \mathrm{a}_t)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathrm{s}_t, \mathrm{a}_t, r(\mathrm{s}_t, \mathrm{a}_t), \mathrm{s}_{t+1}\}$
  end for
  for each gradient step **do**
    $\psi \leftarrow \psi - \lambda_V \widehat{\nabla}_\psi J_V(\psi)$
    $\theta_i \leftarrow \theta_i - \lambda_Q \widehat{\nabla}_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$
    $\phi \leftarrow \phi - \lambda_{pi} \widehat{\nabla}_\phi J_{pi}(\phi)$
    $\bar{\psi} \leftarrow \tau\psi + (1 - \tau)\bar{\psi}$
  end for
end for

Initially, parameters such as ψ, $\bar{\psi}$, θ, ϕ are initialized. Subsequently, it iteratively collects data from the environment, updates the parameters of the value network, Q-functions, and policy network using stochastic gradient descent methods, and performs soft updates on the target value network. These steps collectively enable the SAC algorithm to learn an optimal policy for reinforcement learning tasks involving continuous action spaces [70]–[72].

SAC features an objective function that integrates both a reward component and an entropy term denoted as $\mathcal{H}$, weighted by the parameter α such as

$$J(\pi) = \sum\nolimits_{t=0}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \rho_\pi} \left[r(\mathrm{s}_t, \mathrm{a}_t) + \alpha \mathcal{H}(\pi(\cdot \mid \mathrm{s}_t)) \right]. \tag{8}$$

It simultaneously learns through three networks, namely the state value function, the soft Q-function, and the policy function. Each of these networks - the state value function V, the soft Q-function Q, and the policy function π - is parameterized with ψ, θ, and ϕ respectively [34].

Let us denote the distribution of the replay buffer as $\mathcal{D}$, then we can train the value network V to minimize the squared residual error using an objective function such as

$$J_V(\psi) = \mathbb{E}_{\mathrm{s}_t \sim \mathcal{D}}\left[\frac{1}{2}\left(V_\psi(\mathrm{s}_t) - \mathbb{E}_{\mathrm{a}_t \sim \pi_\phi}\left[Q_\theta(\mathrm{s}_t, \mathrm{a}_t) - \log \pi_\phi(\mathrm{a}_t \mid \mathrm{s}_t)\right]\right)^2\right], \quad (9)$$

Later, by utilizing an unbiased estimator, we can estimate the gradient of the objective function for the residual error in (9), allowing us to update the parameters of the network accordingly. The estimator can be described as

$$\widehat{\nabla}_\psi J_V(\psi) = \nabla_\psi V_\psi(\mathrm{s}_t)(V_\psi(\mathrm{s}_t) - Q_\theta(\mathrm{s}_t, \mathrm{a}_t) + \log \pi_\phi(\mathrm{a}_t \mid \mathrm{s}_t)), \quad (10)$$

where the actions are sampled based on the present policy rather than the replay buffer.

The soft Q-function Q is trained by minimizing the soft Bellman residual such that

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim \mathcal{D}}\left[\frac{1}{2}\left(Q_\theta(\mathrm{s}_t, \mathrm{a}_t) - \widehat{Q}(\mathrm{s}_t, \mathrm{a}_t)\right)^2\right], \quad (11)$$

where

$$\widehat{Q}(\mathrm{s}_t, \mathrm{a}_t) = r(\mathrm{s}_t, \mathrm{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} \sim p}\left[V_{\bar{\psi}}(\mathrm{s}_{t+1})\right]. \quad (12)$$

Just like before, it is also optimized with stochastic gradients such as

$$\widehat{\nabla}_\theta J_Q(\theta) = \nabla_\theta Q_\theta(\mathrm{a}_t, \mathrm{s}_t)(Q_\theta(\mathrm{s}_t, \mathrm{a}_t) - r(\mathrm{s}_t, \mathrm{a}_t) - \gamma V_{\bar{\psi}}(\mathrm{s}_{t+1})). \quad (13)$$

Finally, the policy parameters can be determined by minimizing the expected Kullback-Leibler divergence using an objective function such as

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}}\left[\mathrm{D}_{\mathrm{KL}}\left(\pi_\phi(\cdot \mid \mathrm{s}_t) \parallel \frac{\exp\left(Q_\theta(\mathbf{s}_t, \cdot)\right)}{Z_\theta(\mathbf{s}_t)}\right)\right]. \quad (14)$$

Let denote ϵ_t as an input noise vector, then policy function π can be reparametrized by using a neural network transformation such as the addition of noise or the incorporation of deterministic features.

$$\mathrm{a}_t = f_\phi(\epsilon_t; \mathrm{s}_t), \quad (15)$$

Henceforth, the objective in equation (14) can be rewritten as

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}}\left[\log \pi_\phi(f_\phi(\epsilon_t; \mathrm{s}_t) \mid \mathrm{s}_t) - Q_\theta(\mathrm{s}_t, f_\phi(\epsilon_t; \mathrm{s}_t))\right] \quad (16)$$

The gradient of equation (16) can be approximated with

$$\widehat{\nabla}_\phi J_\pi(\phi) = \nabla_\phi \log \pi_\phi(\mathrm{a}_t \mid \mathrm{s}_t) + \left(\nabla_{\mathrm{a}_t} \log \pi_\phi(\mathrm{a}_t \mid \mathrm{s}_t) - \nabla_{\mathrm{a}_t} Q(\mathrm{s}_t, \mathrm{a}_t)\right) \nabla_\phi f_\phi(\epsilon_t; \mathrm{s}_t) \quad (17)$$

where a_t is evaluated at $f_\phi(\epsilon_t; \mathrm{s}_t)$ [34].

III. Results and Discussion

This section discusses three key topics. Firstly, Subsection III-A elaborates on the environment setup, including an explanation of the reward policy. Next, Subsection III-B delves into simulating DonkeyCar for the self-driving mobile robot using solely the SAC algorithm, shedding light on the intricate processes involved. Lastly, Subsection III-C explores simulating DonkeyCar for the self-driving mobile robot with SAC enhanced by DAE, highlighting their combined effectiveness in autonomous driving simulation.

A. Environment Setup

The simulation setup for the self-driving mobile robot involves several components. It uses Stable-Baselines3, a reinforcement learning library, with the DonkeyCar simulator and OpenAI Gym environment. The simulation environment runs on a Linux Ubuntu 22 operating system, with 8GB of RAM. Graphics processing is performed by an NVIDIA GeForce RTX 3050, with CUDA Version 12.2 providing accelerated computation. The chosen system specifications are aimed at facilitating efficient processing of the DonkeyCar simulation with SAC and DAE. This hardware setup offers ample memory and computational capacity to ensure smooth operation and expedited training of the neural network models utilized in the algorithms.

Stable Baselines3 is a Python library built upon the PyTorch framework, designed to provide users with highly efficient implementations of reinforcement learning algorithms [73]. This library places a strong emphasis on simplicity, extensibility, and high performance, aiming to streamline the process of developing and deploying reinforcement learning models. As a successor to the original Stable Baselines library, Stable Baselines3 inherits a wide array of features and capabilities. These include access to state-of-the-art reinforcement learning algorithms, comprehensive support for both continuous and discrete action spaces, and seamless integration with OpenAI Gym environments. With its robust set of tools and functionalities, Stable Baselines3 serves as a versatile and powerful resource for researchers and practitioners alike, facilitating the exploration and implementation of advanced reinforcement learning techniques in various domains.

DonkeyCar is an open-source DIY (Do It Yourself) self-driving platform designed for small-scale robotic vehicles, typically constructed from remote-controlled cars or similar models. It offers a framework for building, customizing, and experimenting with autonomous driving algorithms and hardware setups such as cameras and lidars [74]–[76]. Additionally, the DonkeyCar Simulator, depicted in Fig. 2, plays a vital role in the DonkeyCar platform, providing users with a virtual environment for testing and training autonomous driving algorithms without the need for physical hardware [76]–[78].

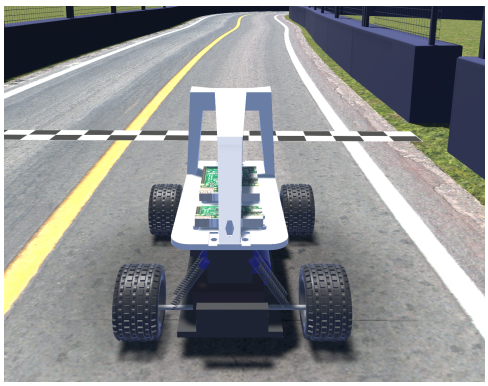

Fig. 2. DonkeyCar on simulation.

In this research, we use the Mini Monaco track. Initially, we need to gather image data for this track on the DonkeyCar Simulator, using the teleoperated system for DonkeyCar. Our goal is to collect 10000 images from various driving scenarios on the track. This image dataset is important for training the denoising autoencoder, as it constitutes the essential input for developing resilient and insightful environment representations. The sample for collected image can be seen in Fig. 3.

Fig. 3. Captured image for Mini Monaco track.

In this environment, if the self-driving mobile robot (DonkeyCar) collides with any obstacles such as walls or other objects, the simulation will automatically reset, and the DonkeyCar will be returned to the starting point. It is crucial to establish policies for SAC to ensure that the self-driving mobile robot can navigate the track effectively. Deviating from the track is not desirable behavior for the self-driving car, so the policy must discourage such actions. Therefore, such a reward system, applicable to both SAC and SAC enhanced with DAE, is designed as

$$g = \begin{cases} -20 - 2 \times \tau, & \text{when off the track,} \\ 1 - \omega \times \tau, & \text{when on the track.} \end{cases} \tag{18}$$

Let g represent the reward, ω denote the off-center lane ratio of the track, and τ signify the throttle. The off-center lane ratio indicates the distance of the mobile robot from the center of the lane. If the off-center lane ratio increases, the mobile robot moves further away from the center, and vice versa. Consequently, each time the robot veers off the track, the reward will be decreased. A multiplier constant of 20 is used to amplify the reduction in accumulated reward, guiding the robot to avoid further deviation. Conversely, as long as the robot stays on the track, the reward will gradually increase. This incentivizes the self-driving car to maintain its position within the lane and navigate the track accurately.

B. Self-Driving Mobile Robot with SAC

The parameters used to evaluate the performance of SAC for self-driving mobile robots on the DonkeyCar simulation, particularly on the Mini Monaco track, are determined by observing the mean episode length and mean episode reward. Higher values of these parameters are indicative of better performance.

Unfortunately, the SAC experiment for the DonkeyCar simulation on the Mini Monaco track does not produce the desired outcomes. Despite multiple efforts, the deep reinforcement learning (DRL) approach using SAC failed to converge as anticipated. Throughout the simulation, the DonkeyCar consistently collided with the walls, resulting in its return to the starting point. This suggests that the DonkeyCar was unable to successfully complete a single lap on the Mini Monaco track.

The mean episode length decreases over time due to the frequent resets, causing the DonkeyCar to return to its initial position. This trend is illustrated in Fig. 4, which illustrates the gradual decline in the mean episode length throughout the experiment.

It terminated at 3557 simulation steps, with a mean episode length of 28.82 steps. Similarly, Fig. 5 illustrates a downward trend in the mean episode reward throughout the simulation, with the final mean episode reward being 0.7815

These findings suggest that the SAC approach for self-driving mobile robots in the DonkeyCar simulation on the Mini Monaco track does not achieve the desired performance and fails to converge during the experiment.

Fig. 4. Mean episode length SAC.

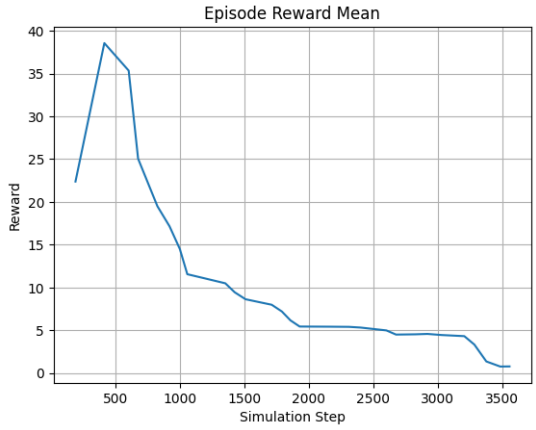

Fig. 5. Mean episode reward SAC.

C. Self-Driving Mobile Robot with DAE and SAC

After concluding that SAC alone is insufficient to achieve convergence for the self-driving mobile robot in DonkeyCar on the Mini Monaco track, we decide to integrate DAE with SAC.

The architecture of the DAE, as depicted in Fig. 6, comprises the encoder and decoder, both sharing similarities in structure and operation. Both employ sequential layers from the first to the fourth layer with convolutional operations.

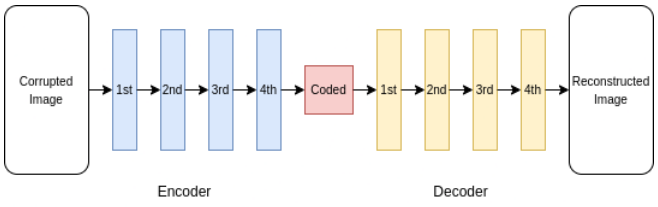

Fig. 6. Architecture of DAE.

The encoder comprises four convolutional layers followed by ReLU activation functions. Each layer applies a 2D convolution operation with specified kernel size and stride. Output channels increase from 16 to 128, indicating feature map depth. Kernel size and stride are set to 4 and 2, reducing spatial dimensions while increasing depth. ReLU introduces non-linearity, aiding in learning complex patterns [79]–[81]. Meanwhile, the decoder includes four transpose convolutional layers followed by ReLU activation functions and a final sigmoid activation function. These layers upsample the encoded latent space to reconstruct the original input image. Output channels decrease from 128 to match the input dimension, aiding in reconstruction. The sigmoid activation function scales output pixel values to the range $[0, 1]$ for representing image intensities [80]–[82].

After explaining the architecture, we can discuss the training process. The DAE is trained using $10,000$ images from the Mini Monaco track over 200 iterations, utilizing the Adam optimizer. This optimizer offers benefits such as adaptive learning rates and momentum, enabling efficient convergence by dynamically adjusting step sizes for each parameter [83]–[85]. Its combination of adaptive gradient and momentum methods makes it particularly effective for optimizing complex, high-dimensional models with sparse gradients. The training process consistently reduces the loss function over time, starting from an initial value of 7.9253 and decreasing to 1.2678 after 200 iterations, indicating successful convergence as depicted in Fig. 7.

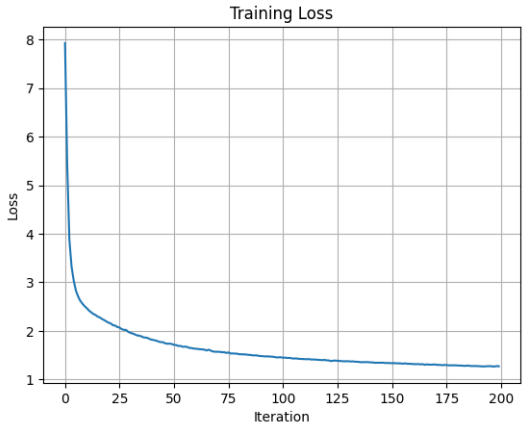

Fig. 7. Training loss DAE.

The decreasing trend in training loss indicates that the DAE is converging towards a solution. As the number of iterations increases, the model becomes more adept at reconstructing clean images from corrupted inputs. This improvement suggests that the DAE is steadily capturing more intricate details and features of the input data, leading to better reconstruction results.

After obtaining the best model of the DAE, we integrate it into our simulation of DonkeyCar on the Mini Monaco track using SAC. Enhancing SAC with DAE yields better results than solely relying on SAC. Throughout the simulation, spanning up to 114357 steps, the DonkeyCar successfully completes a full

lap on the Mini Monaco track. As a result, the mean episode length is recorded at 771.71 steps at the 114357th simulation step, as illustrated in Fig. 8.

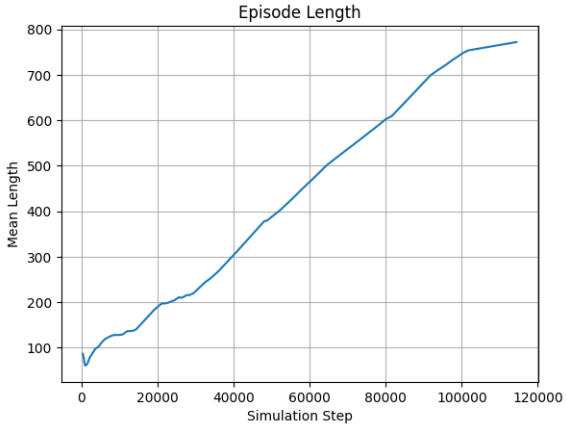

Fig. 8. Mean episode length DAE-SAC.

Additionally, the reward tends to increase over time, with the mean episode reward reaching 2380.4387 at 114357 simulation steps, as shown in Fig. 9.

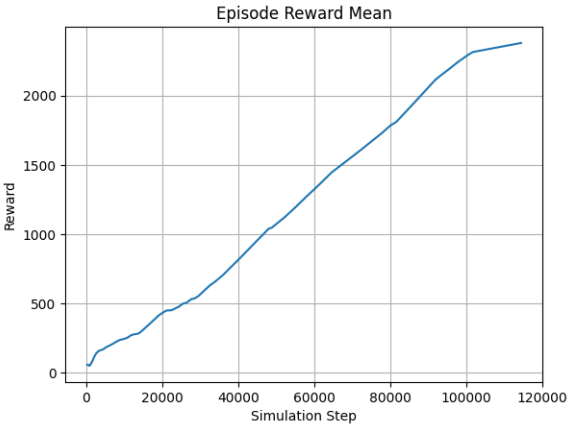

Fig. 9. Mean episode reward DAE-SAC.

The findings suggest that incorporating DAE into SAC can notably improve performance. Specifically, this integration facilitates rapid convergence, indicating the combined method's effectiveness in learning the inherent data patterns and structures. The comparison results for SAC and SAC with DAE are depicted in Table I.

TABLE I. COMPARISON OF SAC AND DAE-SAC

Metric	SAC	DAE-SAC
Number of Steps	3557	114357
Mean Episode Reward	0.7815	2380.4387
Mean Episode Length	28.82	771.71

A significant aspect of utilizing a DAE in conjunction with the SAC algorithm for self-driving mobile robots lies in the preprocessing of sensor data. The DAE is trained to reconstruct clean representations of noisy sensor inputs, such as camera images, by learning to remove noise and extract relevant features.

After training, the DAE effectively cleanses the sensor data, offering SAC algorithm with clearer and more insightful inputs. This preprocessing stage boosts the capability to develop precise and resilient policies for navigation and control tasks. Utilizing denoised sensor data in SAC enhances the learning process, allowing SAC to concentrate on extracting essential information from the input data without being obstructed by noise.

IV. CONCLUSIONS

In conclusion, the simulation of the self-driving car using DonkeyCar on the Mini Monaco track involved two methods, SAC and SAC enhanced with DAE. Initially, SAC alone failed to converge, displaying a mean episode length of only 28.82 steps and a mean episode reward of 0.7815. The simulation ended after 3557 steps due to the inability of SAC alone to converge.

However, after integrating DAE with SAC, significant improvements were observed. The DAE was trained for 200 iterations and achieved convergence, with the best model having a loss of 1.2678. The combined SAC with DAE approach successfully converged, exhibiting a mean episode length of 771.71 steps and a mean episode reward of 2380.4387. The simulation ran for an extended period, reaching 114357 steps before being terminated.

These results highlight the effectiveness of integrating DAE with SAC for enhancing the performance of the self-driving car simulation on the Mini Monaco track. The combined approach not only overcame the convergence issues faced by SAC alone but also significantly improved the mean episode length and mean episode reward, demonstrating its capability to navigate the track successfully and achieve higher rewards. As this study is solely based on simulation, future endeavors

could involve transitioning towards hardware implementation for real-world applications. Since, the integrated SAC and DAE approach holds promise for broader applicability across various autonomous driving tasks and environments, underscoring its potential for seamless real-world implementation.

REFERENCES

[1] T. Morita and S. Managi, "Autonomous vehicles: Willingness to pay and the social dilemma," *Transportation Research Part C: Emerging Technologies*, vol. 119, 2020, doi: 10.1016/j.trc.2020.102748.

[2] A. Chowdhury, G. Karmakar, J. Kamruzzaman, A. Jolfaei and R. Das, "Attacks on Self-Driving Cars and Their Countermeasures: A Survey," in *IEEE Access*, vol. 8, pp. 207308-207342, 2020, doi: 10.1109/ACCESS.2020.3037705.

[3] A. S. M. Al-Obaidi, A. Al-Qassar, A. R. Nasser, A. Alkhayyat, A. J. Humaidi, and I. K. Ibraheem, "Embedded design and implementation of mobile robot for surveillance applications," *Indonesian Journal of Science and Technology*, vol. 6, no. 2, pp. 427–440, 2021, doi: 10.17509/IJOST.V6I2.36275.

[4] Y. Weng, J. Pajarinen, R. Akrour, T. Matsuda, J. Peters and T. Maki, "Reinforcement Learning Based Underwater Wireless Optical Communication Alignment for Autonomous Underwater Vehicles," in *IEEE Journal of Oceanic Engineering*, vol. 47, no. 4, pp. 1231-1245, 2022, doi: 10.1109/JOE.2022.3165805.

[5] J. Wang, Y. Sun, B. Wang and T. Ushio, "Mission-Aware UAV Deployment for Post-Disaster Scenarios: A Worst-Case SAC-Based Approach," in *IEEE Transactions on Vehicular Technology*, vol. 73, no. 2, pp. 2712-2727, 2024, doi: 10.1109/TVT.2023.3319480.

[6] A. Nagahama, T. Saito, T. Wada and K. Sonoda, "Autonomous Driving Learning Preference of Collision Avoidance Maneuvers," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5624-5634, 2021, doi: 10.1109/TITS.2020.2988303.

[7] S. Ono, Y. Okazaki, K. Kanetsuna, and M. Mizumoto, "Egocentric, altruistic, or hypocritic?: A cross-cultural study of choice between pedestrian-first and driver-first of autonomous car," *IEEE Access*, vol. 11, pp. 108716–108726, 2023, doi: 10.1109/ACCESS.2023.3320041.

[8] B. Zhang, R. Sengoku, and H.-O. Lim, "Adaptive motion control for an autonomous mobile robot based on space risk map," *IEEE Access*, vol. 11, pp. 69553–69562, 2023, doi: 10.1109/ACCESS.2023.3292999.

[9] L. A. Dennis and M. Fisher, "Verifiable Self-Aware Agent-Based Autonomous Systems," in *Proceedings of the IEEE*, vol. 108, no. 7, pp. 1011-1026, 2020, doi: 10.1109/JPROC.2020.2991262.

[10] S. Kitajima, H. Chouchane, J. Antona-Makoshi, N. Uchida and J. Tajima, "A Nationwide Impact Assessment of Automated Driving Systems on Traffic Safety Using Multiagent Traffic Simulations," in *IEEE Open Journal of Intelligent Transportation Systems*, vol. 3, pp. 302-312, 2022, doi: 10.1109/OJITS.2022.3165769.

[11] H. Muslim, et al., "Cut-Out Scenario Generation With Reasonability Foreseeable Parameter Range From Real Highway Dataset for Autonomous Vehicle Assessment," in *IEEE Access*, vol. 11, pp. 45349-45363, 2023, doi: 10.1109/ACCESS.2023.3268703.

[12] Y. Miyaki and H. Tsukagoshi, "Self-Excited Vibration Valve That Induces Traveling Waves in Pneumatic Soft Mobile Robots," in *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4133-4139, 2020, doi: 10.1109/LRA.2020.2978455.

[13] M. Aladem and S. A. Rawashdeh, "A Single-Stream Segmentation and Depth Prediction CNN for Autonomous Driving," in *IEEE Intelligent Systems*, vol. 36, no. 4, pp. 79-85, 2021, doi: 10.1109/MIS.2020.2993266.

[14] K. Qu, W. Zhuang, Q. Ye, W. Wu and X. Shen, "Model-Assisted Learning for Adaptive Cooperative Perception of Connected Autonomous Vehicles," in *IEEE Transactions on Wireless Communications*, 2024, doi: 10.1109/TWC.2024.3354507.

[15] N. Kodama, T. Harada, and K. Miyazaki, "Traffic signal control system using deep reinforcement learning with emphasis on reinforcing successful experiences," *IEEE Access*, vol. 10, pp. 128943–128950, 2022, doi: 10.1109/ACCESS.2022.3225431.

[16] T. Osa and M. Aizawa, "Deep reinforcement learning with adversarial training for automated excavation using depth images," *IEEE Access*, vol. 10, pp. 4523–4535, 2022, doi: 10.1109/ACCESS.2022.3140781.

[17] K. Ohashi, K. Nakanishi, Y. Yasui, and S. Ishii, "Deep adversarial reinforcement learning method to generate control policies robust against worst-case value predictions," *IEEE Access*, vol. 11, pp. 100798–100809, 2023, doi: 10.1109/ACCESS.2023.3314750.

[18] G. E. Setyawan, P. Hartono, and H. Sawada, "Cooperative multi-robot hierarchical reinforcement learning," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 9, pp. 35–44, 2022, doi: 10.14569/IJACSA.2022.0130904.

[19] S. Kotera, B. Yin, K. Yamamoto, and T. Nishio, "Lyapunov optimization-based latency-bounded allocation using deep deterministic policy gradient for 11ax spatial reuse," *IEEE Access*, vol. 9, pp. 162337–162347, 2021, doi: .

[20] K. Naya, K. Kutsuzawa, D. Owaki, and M. Hayashibe, "Spiking neural network discovers energy-efficient hexapod motion in deep reinforcement learning," *IEEE Access*, vol. 9, pp. 150345–150354, 2021, doi: 10.1109/ACCESS.2021.3126311.

[21] M. G. Khoshkholgh and H. Yanikomeroglu, "Faded-Experience Trust Region Policy Optimization for Model-Free Power Allocation in Interference Channel," in *IEEE Wireless Communications Letters*, vol. 10, no. 3, pp. 659-663, 2021, doi: 10.1109/LWC.2020.3045005.

[22] I. K. Ozaslan, H. Mohammadi and M. R. Jovanović, "Computing Stabilizing Feedback Gains via a Model-Free Policy Gradient Method," in *IEEE Control Systems Letters*, vol. 7, pp. 407-412, 2023, doi: 10.1109/LCSYS.2022.3188180.

[23] S. Takakura and K. Sato, "Structured Output Feedback Control for Linear Quadratic Regulator Using Policy Gradient Method," in *IEEE Transactions on Automatic Control*, vol. 69, no. 1, pp. 363-370, 2024, doi: 10.1109/TAC.2023.3264176.

[24] R. F. J. Dossa, S. Huang, S. Ontañón, and T. Matsubara, "An empirical investigation of early stopping optimizations in proximal policy optimization," *IEEE Access*, vol. 9, pp. 117981–117992, 2021, doi: 10.1109/ACCESS.2021.3106662.

[25] Y. Gu, Y. Cheng, C. L. P. Chen and X. Wang, "Proximal Policy Optimization With Policy Feedback," in *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 7, pp. 4600-4610, 2022, doi: 10.1109/TSMC.2021.3098451.

[26] S. Siboo, A. Bhattacharyya, R. N. Raj, and S. H. Ashwin, "An empirical study of ddpg and ppo-based reinforcement learning algorithms for autonomous driving," *IEEE Access*, vol. 11, pp. 125094–125108, 2023, doi: 10.1109/ACCESS.2023.3330665.

[27] O. Aydogmus and M. Yilmaz, "Comparative analysis of reinforcement learning algorithms for bipedal robot locomotion," *IEEE Access*, vol. 12, pp. 7490–7499, 2024, doi: 10.1109/ACCESS.2023.3344393.

[28] S. Bhattacharjee, S. Halder, Y. Yan, A. Balamurali, L. V. Iyer and N. C. Kar, "Real-Time SIL Validation of a Novel PMSM Control Based on Deep Deterministic Policy Gradient Scheme for Electrified Vehicles," in *IEEE Transactions on Power Electronics*, vol. 37, no. 8, pp. 9000-9011, 2022, doi: 10.1109/TPEL.2022.3153845.

[29] A. Candeli, G. d. Tommasi, D. G. Lui, A. Mele, S. Santini, and G. Tartaglione, "A deep deterministic policy gradient learning approach to missile autopilot design," *IEEE Access*, vol. 10, pp. 19685–19696, 2022, doi: 10.1109/ACCESS.2022.3150926.

[30] E. H. H. Sumiea, S. J. Abdulkadir, M. G. Ragab, S. M. Al-Selwi, S. M. Fati, A. AlQushaibi, and H. Alhussian, "Enhanced deep deterministic policy gradient algorithm using grey wolf optimizer for continuous control tasks," *IEEE Access*, vol. 11, pp. 139771–139784, 2023, doi: 10.1109/ACCESS.2023.3341507.

[31] N. Abo Mosali, S. S. Shamsudin, O. Alfandi, R. Omar, and N. Al-Fadhali, "Twin delayed deep deterministic policy gradient-based target tracking for unmanned aerial vehicle with achievement rewarding and multistage training," *IEEE Access*, vol. 10, pp. 23545–23559, 2022, doi: 10.1109/ACCESS.2022.3154388.

[32] J. Khalid, M. A. Ramli, M. S. Khan, and T. Hidayat, "Efficient load frequency control of renewable integrated power system: A twin delayed ddpg-based deep reinforcement learning approach," *IEEE Access*, vol. 10, pp. 51561–51574, 2022, doi: 10.1109/ACCESS.2022.3174625.

[33] O. E. Egbomwan, S. Liu and H. Chaoui, "Twin Delayed Deep Deterministic Policy Gradient (TD3) Based Virtual Inertia Control for Inverter-Interfacing DGs in Microgrids," in *IEEE Systems Journal*, vol. 17, no. 2, pp. 2122-2132, 2023, doi: 10.1109/JSYST.2022.3222262.

[34] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *35th International Conference on Machine Learning, ICML 2018*, vol. 5, pp. 2976–2989, 2018.

[35] A. Surriani, O. Wahyunggoro, and A. I. Cahyadi, "A trajectory control for bipedal walking robot using stochastic-based continuous deep reinforcement learning," *Evergreen*, vol. 10, no. 3, pp. 1538–1548, 2023, doi: 10.5109/7151701.

[36] H. Yong, J. Seo, J. Kim, M. Kim and J. Choi, "Suspension Control Strategies Using Switched Soft Actor-Critic Models for Real Roads," in *IEEE Transactions on Industrial Electronics*, vol. 70, no. 1, pp. 824-832, 2023, doi: 10.1109/TIE.2022.3153805.

[37] E. Prianto, M. Kim, J.-H. Park, J.-H. Bae, and J.-S. Kim, "Path planning for multi-arm manipulators using deep reinforcement learning: Soft actor–critic with hindsight experience replay," *Sensors*, vol. 20, no. 20, pp. 1–23, 2020, doi: 10.3390/s20205911.

[38] C.-C. Wong, S.-Y. Chien, H.-M. Feng, and H. Aoyama, "Motion planning for dual-arm robot based on soft actor-critic," *IEEE Access*, vol. 9, pp. 26871–26885, 2021, doi: 10.1109/ACCESS.2021.3056903.

[39] A. Mustafa, T. Sasamura and T. Morita, "Robust Speed Control of Ultrasonic Motors Based on Deep Reinforcement Learning of a Lyapunov Function," in *IEEE Access*, vol. 10, pp. 46895-46910, 2022, doi: 10.1109/ACCESS.2022.3170995.

[40] M. R. Hong, et al., "Optimizing Reinforcement Learning Control Model in Furuta Pendulum and Transferring it to Real-World," in *IEEE Access*, vol. 11, pp. 95195-95200, 2023, doi: 10.1109/ACCESS.2023.3310405.

[41] K. Kasaura, S. Miura, T. Kozuno, R. Yonetani, K. Hoshino and Y. Hosoe, "Benchmarking Actor-Critic Deep Reinforcement Learning Algorithms for Robotics Control With Action Constraints," in *IEEE Robotics and Automation Letters*, vol. 8, no. 8, pp. 4449-4456, 2023, doi: 10.1109/LRA.2023.3284378.

[42] H. Sekkat, O. Moutik, L. Ourabah, B. ElKari, Y. Chaibi, and T. A. Tchakoucht, "Review of reinforcement learning for robotic grasping: Analysis and recommendations," *Statistics, Optimization and Information Computing*, vol. 12, no. 2, pp. 571–601, 2024, doi: 10.19139/soic-2310-5070-1797.

[43] E. Chisari, A. Liniger, A. Rupenyan, L. V. Gool and J. Lygeros, "Learning from Simulation, Racing in Reality," *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 8046-8052, 2021, doi: 10.1109/ICRA48506.2021.9562079.

[44] F. Li et al., "Autoencoder-Enabled Potential Buyer Identification and Purchase Intention Model of Vacation Homes," in *IEEE Access*, vol. 8, pp. 212383-212395, 2020, doi: 10.1109/ACCESS.2020.3037920.

[45] K. Sama et al., "Extracting Human-Like Driving Behaviors From Expert Driver Data Using Deep Learning," in *IEEE Transactions on Vehicular Technology*, vol. 69, no. 9, pp. 9315-9329, 2020, doi: 10.1109/TVT.2020.2980197.

[46] P. Hartono, "Mixing Autoencoder With Classifier: Conceptual Data Visualization," in *IEEE Access*, vol. 8, pp. 105301-105310, 2020, doi: 10.1109/ACCESS.2020.2999155.

[47] P. Cristovao, H. Nakada, Y. Tanimura and H. Asoh, "Generating In-Between Images Through Learned Latent Space Representation Using Variational Autoencoders," in *IEEE Access*, vol. 8, pp. 149456-149467, 2020, doi: 10.1109/ACCESS.2020.3016313.

[48] K. Ohashi, K. Nakanishi, W. Sasaki, Y. Yasui and S. Ishii, "Deep Adversarial Reinforcement Learning With Noise Compensation by Autoencoder," in *IEEE Access*, vol. 9, pp. 143901-143912, 2021, doi: 10.1109/ACCESS.2021.3121751

[49] K. Fujiwara, H. Iwamoto, K. Hori and M. Kano, "Driver Drowsiness Detection Using R-R Interval of Electrocardiogram and Self-Attention Autoencoder," in *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 2956-2965, 2024, doi: 10.1109/TIV.2023.3308575.

[50] F. N. Khan and A. P. T. Lau, "Robust and efficient data transmission over noisy communication channels using stacked and denoising autoencoders," in *China Communications*, vol. 16, no. 8, pp. 72-82, 2019, doi: 10.23919/JCC.2019.08.007.

[51] T. Yokota, H. Hontani, Q. Zhao and A. Cichocki, "Manifold Modeling in Embedded Space: An Interpretable Alternative to Deep Image Prior," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 3, pp. 1022-1036, 2022, doi: 10.1109/TNNLS.2020.3037923.

[52] R. Al-Hmouz, W. Pedrycz, A. Balamash and A. Morfeq, "Logic-Oriented Autoencoders and Granular Logic Autoencoders: Developing Interpretable Data Representation," in *IEEE Transactions on Fuzzy Systems*, vol. 30, no. 3, pp. 869-877, 2022, doi: 10.1109/TFUZZ.2020.3043659.

[53] N. J. Zakaria, M. I. Shapiai, R. A. Ghani, M. N. M. Yassin, M. Z. Ibrahim and N. Wahid, "Lane Detection in Autonomous Vehicles: A Systematic Review," in *IEEE Access*, vol. 11, pp. 3729-3765, 2023, doi: 10.1109/ACCESS.2023.3234442.

[54] B. Rabhi, A. Elbaati, H. Boubaker, U. Pal, and A. M. Alimi, "Multilingual handwriting recovery framework based on convolutional denoising autoencoder with attention model," *Multimedia Tools and Applications*, vol. 83, no. 8, pp. 22295–22326, 2024, doi: 10.1007/s11042-023-16499-z.

[55] T. Murata, K. Fukami, and K. Fukagata, "Nonlinear mode decomposition with convolutional neural networks for fluid dynamics," *Journal of Fluid Mechanics*, vol. 882, 2020, doi: 10.1017/jfm.2019.822.

[56] W. -H. Lee, M. Ozger, U. Challita and K. W. Sung, "Noise Learning-Based Denoising Autoencoder," in *IEEE Communications Letters*, vol. 25, no. 9, pp. 2983-2987, 2021, doi: 10.1109/LCOMM.2021.3091800.

[57] H. El-Fiqi, M. Wang, K. Kasmarik, A. Bezerianos, K. C. Tan and H. A. Abbass, "Weighted Gate Layer Autoencoders," in *IEEE Transactions on Cybernetics*, vol. 52, no. 8, pp. 7242-7253, 2022, doi: 10.1109/TCYB.2021.3049583.

[58] A. Nawaz, S. S. Khan and A. Ahmad, "Ensemble of Autoencoders for Anomaly Detection in Biomedical Data: A Narrative Review," in *IEEE Access*, vol. 12, pp. 17273-17289, 2024, doi: 10.1109/ACCESS.2024.3360691.

[59] H. Anand, B. S. Sammuli, K. E. J. Olofsson and D. A. Humphreys, "Real-Time Magnetic Sensor Anomaly Detection Using Autoencoder Neural Networks on the DIII-D Tokamak," in *IEEE Transactions on Plasma Science*, vol. 50, no. 11, pp. 4126-4130, 2022, doi: 10.1109/TPS.2022.3181548.

[60] T. -W. Ban, "Compressed Feedback Using Autoencoder Based on Deep Learning for D2D Communication Networks," in *IEEE Wireless Communications Letters*, vol. 12, no. 4, pp. 590-594, 2023, doi: 10.1109/LWC.2023.3234574.

[61] M. Kramer, "Autoassociative neural networks," *Computers & Chemical Engineering*, vol. 16, no. 4, pp. 313–328, 1992, doi: 10.1016/0098-1354(92)80051-A.

[62] Y. Qiu, Y. Yang, Z. Lin, P. Chen, Y. Luo and W. Huang, "Improved denoising autoencoder for maritime image denoising and semantic segmentation of USV," in *China Communications*, vol. 17, no. 3, pp. 46-57, 2020, doi: 10.23919/JCC.2020.03.005.

[63] P. Singh and A. Sharma, "Attention-Based Convolutional Denoising Autoencoder for Two-Lead ECG Denoising and Arrhythmia Classification," in *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1-10, 2022, doi: 10.1109/TIM.2022.3197757.

[64] X. Li, Z. Liu and Z. Huang, "Deinterleaving of Pulse Streams With Denoising Autoencoders," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 6, pp. 4767-4778, 2020, doi: 10.1109/TAES.2020.3004208.

[65] A. Alajmi, W. Ahsan, M. Fayaz and A. Nallanathan, "Intelligent Resource Allocation in Backscatter-NOMA Networks: A Soft Actor Critic Framework," in *IEEE Transactions on Vehicular Technology*, vol. 72, no. 8, pp. 10119-10132, 2023, doi: 10.1109/TVT.2023.3254138.

[66] Z. He, L. Dong, C. Song and C. Sun, "Multiagent Soft Actor-Critic Based Hybrid Motion Planner for Mobile Robots," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 12, pp. 10980-10992, 2023, doi: 10.1109/TNNLS.2022.3172168.

[67] R. Ma, Y. Wang, S. Wang, L. Cheng, R. Wang and M. Tan, "Sample-Observed Soft Actor-Critic Learning for Path Following of a Biomimetic Underwater Vehicle," in *IEEE Transactions on Automation Science and Engineering*, 2023, doi: 10.1109/TASE.2023.3264237.

[68] S. Wang, R. Diao, C. Xu, D. Shi and Z. Wang, "On Multi-Event Co-Calibration of Dynamic Model Parameters Using Soft Actor-Critic," in *IEEE Transactions on Power Systems*, vol. 36, no. 1, pp. 521-524, 2021, doi: 10.1109/TPWRS.2020.3030164.

[69] A. R. Heidarpour, M. R. Heidarpour, M. Ardakani, C. Tellambura and M. Uysal, "Soft Actor–Critic-Based Computation Offloading in Multiuser MEC-Enabled IoT—A Lifetime Maximization Perspective," in *IEEE Internet of Things Journal*, vol. 10, no. 20, pp. 17571-17584, 2023, doi: 10.1109/JIOT.2023.3277753.

[70] M. Haklidir and H. Temeltaş, "Guided Soft Actor Critic: A Guided Deep Reinforcement Learning Approach for Partially Observable Markov Decision Processes," in *IEEE Access*, vol. 9, pp. 159672-159683, 2021, doi: 10.1109/ACCESS.2021.3131772.

[71] N. Gholizadeh, N. Kazemi and P. Musilek, "A Comparative Study of Reinforcement Learning Algorithms for Distribution Network Reconfiguration With Deep Q-Learning-Based Action Sampling," in *IEEE Access*, vol. 11, pp. 13714-13723, 2023, doi: 10.1109/ACCESS.2023.3243549.

[72] A. R. Sayed, X. Zhang, G. Wang, C. Wang and J. Qiu, "Optimal Operable Power Flow: Sample-Efficient Holomorphic Embedding-Based Reinforcement Learning," in *IEEE Transactions on Power Systems*, vol. 39, no. 1, pp. 1739-1751, 2024, doi: 10.1109/TPWRS.2023.3266773.

[73] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021.

[74] H. Yun and D. Park, "Virtualization of self-driving algorithms by interoperating embedded controllers on a game engine for a digital twining autonomous vehicle," *Electronics*, vol. 10, no. 17, 2021, doi: 10.3390/electronics10172102.

[75] A. Bayuwindra, L. Wonohito and B. R. Trilaksono, "Design of DDPG-Based Extended Look-Ahead for Longitudinal and Lateral Control of Vehicle Platoon," in *IEEE Access*, vol. 11, pp. 96648-96660, 2023, doi: 10.1109/ACCESS.2023.3311850.

[76] T. Uetsuki, Y. Okuyama and J. Shin, "CNN-based End-to-end Autonomous Driving on FPGA Using TVM and VTA," *2021 IEEE 14th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*, pp. 140-144, 2021, doi: 10.1109/MCSoC51149.2021.00028.

[77] N. K. Manjunath, A. Shiri, M. Hosseini, B. Prakash, N. R. Waytowich and T. Mohsenin, "An Energy Efficient EdgeAI Autoencoder Accelerator for Reinforcement Learning," in *IEEE Open Journal of Circuits and Systems*, vol. 2, pp. 182-195, 2021, doi: 10.1109/OJCAS.2020.3043737

[78] A. Fatima, A. K. Gowda, C. C U, M. Tauseef and B. Y, "Implementation of Driverless Car," *2023 International Conference on Advances in Electronics, Communication, Computing and Intelligent Information Systems (ICAECIS)*, pp. 447-452, 2023, doi: 10.1109/ICAECIS58353.2023.10170382.

[79] J. S. Hieber, "Nonparametric regression using deep neural networks with relu activation function," *Annals of Statistics*, vol. 48, no. 4, pp. 1875–1897, 2020, doi: 10.1214/19-AOS1875.

[80] Y. Terada and R. Hirose, "Fast generalization error bound of deep learning without scale invariance of activation functions," *Neural Networks*, vol. 129, pp. 344–358, 2020, doi: 10.1016/j.neunet.2020.05.033.

[81] M. Tanaka, "Weighted sigmoid gate unit for an activation function of deep neural network," *Pattern Recognition Letters*, vol. 135, pp. 354–359, 2020, doi: 10.1016/j.patrec.2020.05.017.

[82] T. Szandała, "Review and comparison of commonly used activation functions for deep neural networks," *Studies in Computational Intelligence*, vol. 903, pp. 203–224, 2021, doi: 10.1007/978-981-15-5495-7.

[83] S. Bera and V. K. Shrivastava, "Analysis of various optimizers on deep convolutional neural network model in the application of hyperspectral remote sensing image classification," *International Journal of Remote Sensing*, vol. 41, no. 7, pp. 2664–2683, 2020, doi: 10.1080/01431161.2019.1694725.

[84] R. Llugsi, S. E. Yacoubi, A. Fontaine and P. Lupera, "Comparison between Adam, AdaMax and Adam W optimizers to implement a Weather Forecast based on Neural Networks for the Andean city of Quito," *2021 IEEE Fifth Ecuador Technical Chapters Meeting (ETCM),* pp. 1-6, 2021, doi: 10.1109/ETCM53643.2021.9590681.

[85] R. M. Schmidt, F. Schneider, and P. Hennig, "Descending through a crowded valley - benchmarking deep learning optimizers," *International Conference on Machine Learning*, vol. 139, pp. 9367–9376, 2021.