International Journal of Internet, Broadcasting and Communication Vol.16 No.1 63-71 (2024)
http://dx.doi.org/10.7236/IJIBC.2024.16.1.63

IJIBC 24-1-8

Comparative analysis of activation functions within reinforcement learning for autonomous vehicles merging onto highways

Dongcheul Lee* and Janise McNair

**Professor, Department of Multimedia Engineering, Hannam University, Korea*
Professor, Department of Electrical & Computer Engineering, University of Florida, USA
jackdclee@hnu.kr, mcnair@ece.ufl.edu

Abstract

Deep reinforcement learning (RL) significantly influences autonomous vehicle development by optimizing decision-making and adaptation to complex driving environments through simulation-based training. In deep RL, an activation function is used, and various activation functions have been proposed, but their performance varies greatly depending on the application environment. Therefore, finding the optimal activation function according to the environment is important for effective learning. In this paper, we analyzed nine commonly used activation functions for RL to compare and evaluate which activation function is most effective when using deep RL for autonomous vehicles to learn highway merging. To do this, we built a performance evaluation environment and compared the average reward of each activation function. The results showed that the highest reward was achieved using Mish, and the lowest using SELU. The difference in reward between the two activation functions was 10.3%.

Keywords: *Deep Reinforcement Learning, Autonomous Vehicle, Activation Function, Highway Merging*

1. Introduction

The emergence and evolution of autonomous vehicles signify a paradigm shift in modern transportation systems, promising a future of enhanced safety, efficiency, and convenience. These vehicles are empowered by advanced technologies, and among the critical pillars of their operation lies the application of reinforcement learning (RL) [1]. RL, a subset of machine learning, enables these autonomous systems to learn and make decisions through continuous interactions with their environment, a pivotal capability in navigating complex scenarios like merging onto highways.

Within the realm of RL, the choice of activation functions holds substantial significance in shaping the performance and adaptability of neural networks. Activation functions serve as nonlinear transformations, allowing neural networks to model complex relationships between inputs and outputs. In the context of autonomous vehicles and their interaction with highway traffic, the role of activation functions becomes particularly crucial in facilitating quick, accurate, and adaptable decision-making processes [2].

Manuscript Received: December. 15, 2023 / Revised: January. 7, 2024 / Accepted: January. 17, 2024
Corresponding Author: jackdclee@hnu.kr
Tel: +82-42-629-8373, Fax: +82-42-629-8270
Professor, Department of Multimedia Engineering, Hannam University, Korea

This paper aims to conduct an extensive comparative analysis focusing on various activation functions within the context of RL for autonomous vehicles, specifically during the challenging task of merging onto highways. In this task, the vehicle starts on a main highway but soon approaches a road junction with incoming vehicles on the access ramp. The objective of the agent is now to maintain a high speed while making room for the vehicles so that they can safely merge into the traffic. The selection of an appropriate activation function stands as a fundamental aspect influencing the vehicle's ability to perceive, learn, and act efficiently in dynamic traffic environments.

The primary objective of this study is to systematically evaluate and compare multiple activation functions commonly employed in neural networks for RL applications within autonomous vehicles. Parameters such as rewards will be rigorously analyzed and contrasted across different activation functions. By doing so, this research endeavors to identify the activation functions that best complement the demands of highway merging scenarios, enabling autonomous vehicles to seamlessly integrate into existing traffic streams.

Through a blend of empirical evaluations and theoretical considerations, this comparative analysis seeks to offer comprehensive insights into the strengths and limitations of various activation functions. Furthermore, this study aims to provide actionable guidance to researchers, engineers, and practitioners involved in the development of autonomous vehicle systems. The insights derived from this analysis will assist in informed decision-making regarding the selection and implementation of activation functions within RL algorithms, thereby enhancing the decision-making capabilities of autonomous vehicles during crucial merging maneuvers on highways.

The findings and recommendations stemming from this comparative study aspire to contribute significantly to the ongoing advancements in autonomous vehicle technology. By shedding light on the optimal activation functions for highway merging scenarios, this research aims to facilitate the creation of safer, more efficient, and adaptive autonomous vehicles, thereby accelerating their integration into our transportation systems for a future marked by intelligent mobility.

2. Related works

2.1 Activation functions

In RL, activation functions are key components used within neural networks, determining the output of individual neurons. These functions introduce non-linearity to the network, allowing it to model and understand complex relationships in data, which is crucial for learning and decision-making processes. Activation functions operate on the weighted sum of inputs plus a bias term in a neuron, transforming this aggregated input into an output. This output is then passed on to the next layer of neurons in the neural network.

Each of these activation functions possesses unique characteristics that impact the performance and learning capabilities of neural networks in RL tasks. The choice of activation function should be based on the specific requirements of the task, network architecture, and computational considerations, often requiring experimentation to determine the most suitable function.

Here's a comprehensive list of various activation functions commonly used in RL, along with their features, advantages, and disadvantages:

The Rectified Linear Unit (ReLU) is a simple, computationally efficient activation function that effectively combats the vanishing gradient problem in deep neural networks [3], yet it may suffer from the "dying ReLU" issue where certain neurons become inactive, limiting their ability to update weights and potentially causing

unbounded activation for positive inputs [4].

The Softplus activation function offers a smooth, differentiable function allowing for continuous gradients, enabling non-zero outputs for all inputs, yet it retains limited usage due to computational complexity and vulnerability to vanishing gradients for extreme negative inputs [5].

The Leaky ReLU activation function addresses the vanishing gradient problem by allowing a small non-zero gradient for negative inputs, preventing "dead" neurons, but it introduces increased model complexity and may not entirely solve the issue of vanishing gradients in all cases [6].

The Parametric ReLU (PReLU) activation function extends Leaky ReLU by introducing a learned slope for negative inputs, potentially improving network performance, yet it increases model complexity with additional parameters, posing a risk of overfitting and requiring careful regularization [7].

The Gaussian Error Linear Unit (GELU) activation function, aiming to approximate a Gaussian cumulative distribution function, offers smoothness, non-linearity, and computational simplicity, exhibiting improved performance in certain neural network architectures [8].

The Scaled Exponential Linear Unit (SELU) activation function, designed to ensure self-normalizing properties within neural networks, exhibits automatic normalization, enabling stable training, reduced vanishing/exploding gradients, yet demands specific constraints on the network architecture and initializations to ensure effectiveness [9].

The Continuously Differentiable Exponential Linear Units (CELU) activation function, offering smoothness and continuity for negative inputs while allowing non-zero gradients, provides improved modeling capabilities, yet it requires additional parameters, potentially leading to increased computational complexity and overfitting risks in certain scenarios [10].

The Sigmoid-Weighted Linear Units (SiLU) activation function, known for its smoothness, non-monotonicity, and efficient computation, facilitates improved modeling capabilities with non-linearity, but its computation might be slightly more complex compared to ReLU-based functions, potentially impacting training efficiency in larger networks [11].

The Mish activation function, characterized by its smoothness, non-monotonicity, and differentiability, exhibits potential performance improvements in neural networks, yet its computational complexity and limited empirical validation across various tasks require further investigation for broader adoption [12].

2.2 Highway-env

RL benefits significantly from a simulated environment for training and evaluating the performance of the agent. Highway-env is an RL environment for decision-making in autonomous driving agents [13]. It provides a variety of environments related to highway driving scenarios such as a highway, a merge, a roundabout, a parking, an intersection, and a racetrack. In the highway, the agent drives on a multilane highway populated with other vehicles. In the merge, the agent starts on a main highway but soon approaches a road junction with incoming vehicles on the access ramp. In the roundabout, the agent approaches a roundabout with flowing traffic. In parking, the agent parks in a given space with the appropriate heading. In the intersection, the agent approaches an intersection with dense traffic. In the racetrack, it should keep lane and avoid obstacles while driving the racetrack.

For all environments, several types of observations can be used such as kinematics, grayscale image, occupancy grid, and time to collision. Kinematics is a $V \times F$ array that describes a list of V nearby vehicles by

a set of features of size *F*. The grayscale image is a *W*×*H* grayscale image of the scene, where *W* is the width and *H* is the height of the image. The occupancy grid is a *W*×*H*×*F* array that represents a grid of shape *W*×*H* around the agent in uniform rectangle cells described by *F* features. The time to collision is a *V*×*L*×*H* array, that represents the predicted time-to-collision of observed vehicles. The predictions are performed for *V* different values of the agent speed, *L* lanes on the road, and represented as one-hot encodings over *H* discretized time values.

3. Performance evaluation methods

A deep RL agent using Proximal Policy Optimization (PPO) as a learning algorithm was created to evaluate the performance of activation functions for neural networks when self-driving cars learn to merge onto highways [14]. PPO focuses on training agents to make sequential decisions in dynamic environments while optimizing policies to maximize cumulative rewards. It operates on fundamental principles centered around iterative policy updates, aiming to strike a balance between improving the policy's performance and maintaining stability throughout the learning process. The core principles of PPO revolve around its methods for policy optimization. By leveraging a clipped surrogate objective function, PPO constrains policy updates to prevent large fluctuations that could destabilize learning. This clipped objective ensures that policy improvements remain within a specified range, promoting smoother convergence and mitigating drastic changes that might impede progress. Another crucial principle embedded within PPO is trust region optimization, which governs the extent of allowable policy updates. This mechanism limits the alterations made to the policy to a certain range, thereby maintaining a controlled learning pace. By regulating the magnitude of policy changes, trust region optimization contributes to the algorithm's stability, enabling incremental improvements without introducing excessive variability that could hinder learning stability.

The objective function in PPO is represented as:

$$L^{CLIP}(\theta) = \mathbb{E}_t[\min(r_t(\theta) \cdot A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon) \cdot A_t)] \quad (1)$$

Here, $r_t(\theta)$ denotes the ratio of the updated policy to the previous policy, A_t represents the advantage function, and ϵ is a hyperparameter that controls the extent of policy clipping.

The policy update involves adjusting the policy parameters based on the objective function to enhance the policy's performance:

$$\theta_{\text{new}} = \arg\ \max_\theta L^{CLIP}(\theta) \quad (2)$$

Here, θ_{new} denotes the updated policy parameters that maximize the clipped surrogate objective function $L^{CLIP}(\theta)$, where θ represents the policy's parameters.

The advantage function A_t estimates the advantage of taking a particular action compared to the average action's value. It's calculated using the following formula:

$$A_t = Q(s_t, a_t) - V(s_t) \quad (3)$$

Here, $Q(s_t, a_t)$ represents the action-value function, and $V(s_t)$ denotes the state-value function. The advantage function provides information on how beneficial an action is concerning the current state.

PPO incorporates a trust region approach to limit policy updates within a specific range, maintaining stability during learning. This involves imposing a constraint on the policy update to prevent large deviations:

$$\theta_{\text{new}} = \arg\ \max_{\theta} \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t \right] \tag{4}$$

Here, $\pi_\theta(a_t|s_t)$ represents the updated policy probability, and $\pi_{\theta_{old}}(a_t|s_t)$ denotes the previous policy probability.

The agent used in the evaluation used merge-v0 from Highway-env to learn highway merges. For the observation type, we used a 600×150 grayscale image. The RGB to grayscale conversion was a weighted sum, configured by the weights parameters 0.3, 0.6, and 0.1. 16 images were stacked as is customary with image observation. The agent is traveling with three other cars on a two-lane highway and encounters a single car merging at a road junction. Two cars are driving on the primary lane, and the other car is driving on the secondary lane with the agent. The actions available in this environment are continuous actions: accelerator pedal control and steering wheel control. The episode termination conditions are either success by properly merging onto the highway, or failure by colliding with a nearby car. The learning period was 20,000 timesteps for each activation function. Three convolutional hidden layers of size 256 were shared between the policy network and the value network. The number of steps to run for each environment per update was 512 where batch size was 64.

The agent should have good acceleration on the road and be able to avoid collisions. Thus, the reward function consists of a velocity term and a collision term:

$$R(s_t, a) = a \frac{v_t - v_{min}}{v_{max} - v_{min}} - b\text{collision} \tag{5}$$

where v_t, v_{min}, v_{max} are the current, minimum, and maximum speed of the agent respectively, and a, b are two coefficients, and collision is the weight of collision penalty. Since the rewards must be bounded, and the optimal policy is unaffected by scaling and shifting rewards, we choose to normalize them in the [0,1] range as a convention. It's also important to note that we prohibit negative rewards because they could incentivize the agent to terminate an episode prematurely (by causing a collision) rather than taking the risk of experiencing a negative return if no satisfactory trajectory can be found.

4. Performance evaluation results

To evaluate the impact of the activation function on the agent's learning to merge highways, we graphically compared the rewards over time during learning, as shown in Figure 1. The speed at which the agents learned varied greatly between the activation functions, but by the 20k timestep at the end of learning, they were mostly

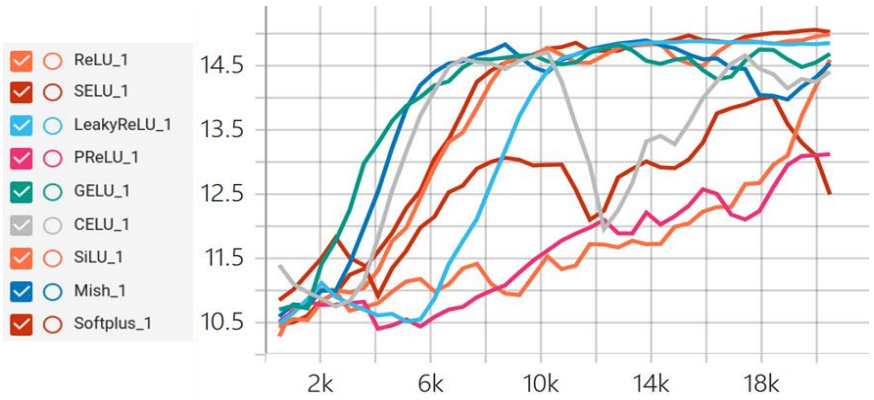

Figure 1. A plot illustrating rewards based on timestamps during the agent's learning

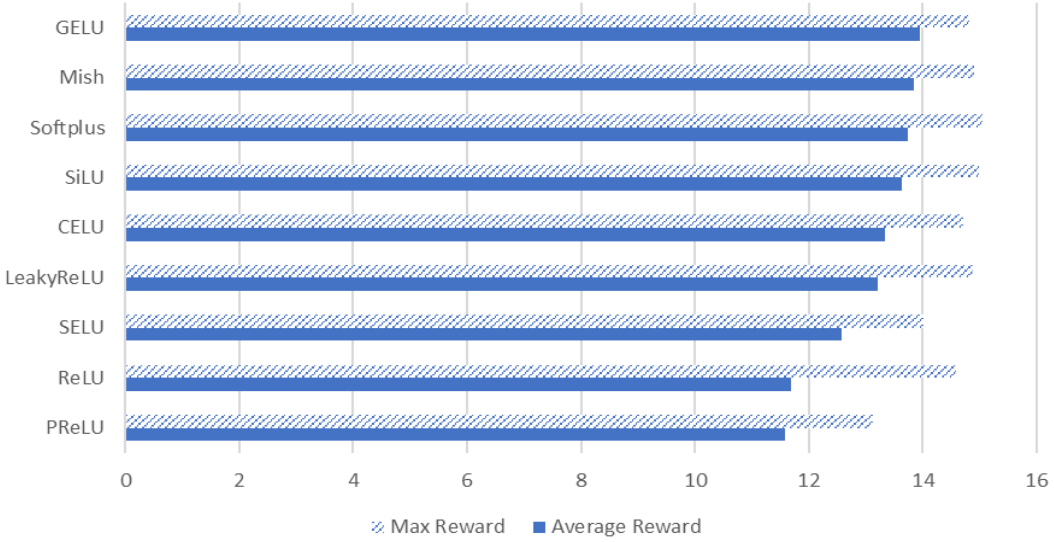

Figure 2. A graph showing the average and maximum rewards while the agent is learning

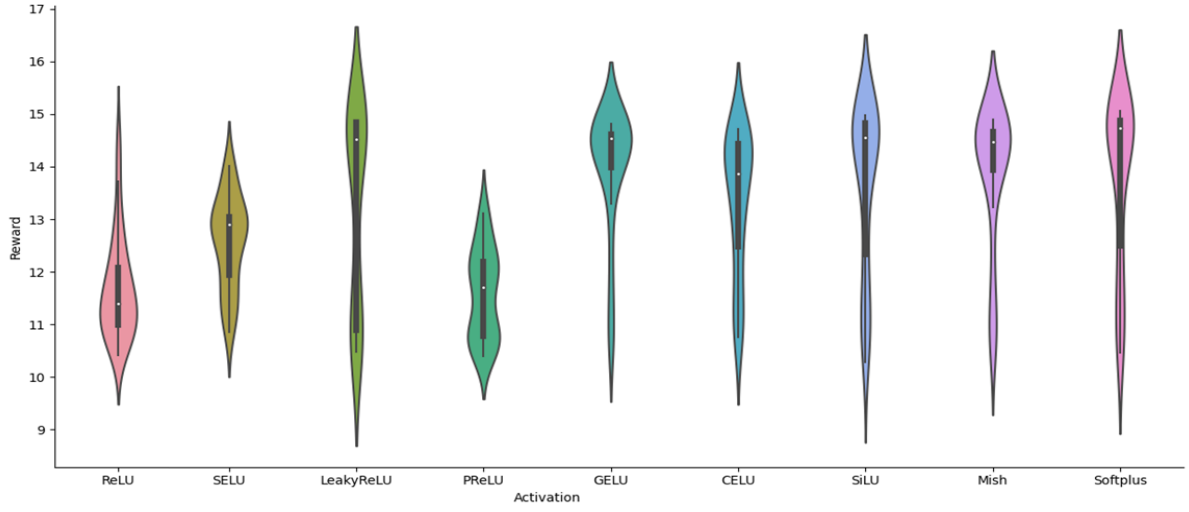

Figure 3. A violin plot displaying rewards based on timestamps during the agent's learning

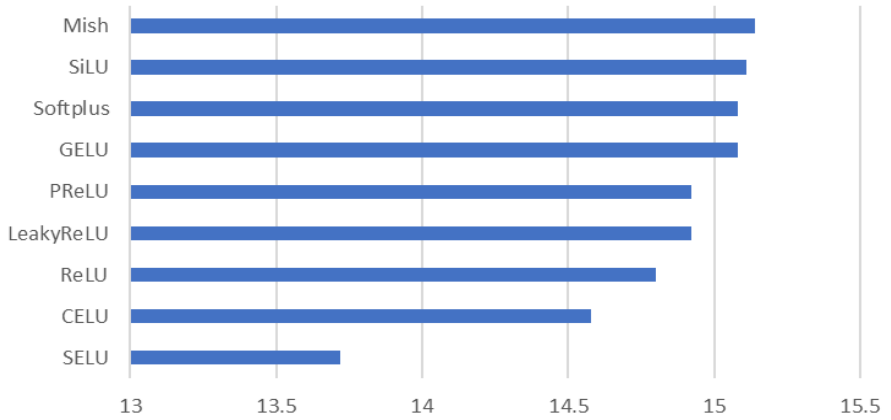

Figure 4. A graph showing the average rewards while the agent is testing

similar, with the exception of PReLU. Some activation functions, such as mish, SiLU, and GELU show a rapid increase in reward as time progresses, indicating effective learning or optimization. CELU and SELU show more volatility, with the reward oscillating up and down, which might suggest overfitting, exploration and policy oscillation. If the agent is trained in batches, it may overfit to particular samples within the batch, leading to a temporary increase in rewards that is not sustainable as new batches are introduced [15]. Also, agents often employ exploration strategies that randomly select actions to discover new strategies or areas of the state space that might yield higher rewards. This exploration can lead to fluctuations in the reward as the agent may temporarily choose suboptimal actions in search of better long-term strategies. Furthermore, the process of policy improvement can sometimes lead to oscillation. As an agent updates its policy, it may alternate between different strategies, each of which is optimal under slightly different conditions or assumptions.

For PReLU, we can see in Figure 2 that both the average and maximum rewards are lowest, and the learning rate increases very slowly. On the other hand, the average reward was highest for GELU and lowest for PReLU, with a difference of 2.37 (20.5%). The maximum reward was highest for Softplus and lowest for PReLU, with a difference of 1.94 (14.8%).

Figure 3 shows a violin plot of the data in Figure 1. ReLU shows a distribution with a wide spread, indicating variability in the reward values, with a lower median compared to others. LeakyReLU and PReLU shows a broad distribution similar to ReLU but with a slightly higher median reward. Some activation functions like GELU, CELU, SiLU, Mish, and Softplus appear to have a higher median reward and less variability, which might suggest a more stable performance under the conditions of this specific dataset and problem.

Figure 4 plots the average reward for testing an agent that has completed training for each activation function. Each function was tested 1000 times. The most rewarding activation functions were Mish, SiLU, Softplus, and GELU, while SELU was the least rewarding of the other functions. Mish, SiLU, Softplus, and GELU all performed well in training, so they performed well in testing. Mish had the highest average reward and SELU the lowest, with a difference of 1.42 (10.3%).

However, when comparing Figure 2 and Figure 4, except for the four activation functions that performed well, there was a difference in the reward ranking between training and testing. Because, first of all, the model may be overly specialized to the training environment during training and not adapt to the testing environment. In particular, if you use an activation function that indirectly helps prevent overfitting during training, you may not get the maximum reward during training, but you may get better results during testing. Also, if you use an

activation function that encourages exploration during training, you may get less reward because you try multiple strategies. In the case of a parametric activation function, this can happen because the parameters can determine the level of exploration. However, as a result of exploration, it is possible to find a better policy in testing and get a higher reward. In the case of PReLU, the lowest reward was obtained during training, but the 5th best reward was obtained during testing. Furthermore, some activation functions allow for faster learning, but longer learning can lead to worse results. In the case of SELU, we can see that the reward continues to rise until 18k timesteps and then drops sharply. This resulted in the lowest reward when testing.

5. Conclusion

This paper evaluates which activation function is advantageous for an autonomous vehicle to use when learning to merge onto a highway through RL. For this purpose, an agent was created and nine activation functions commonly used in RL were alternately used to learn highway merging and compared. The results showed that the function with the highest average reward during training was GELU and the lowest was PReLU. The average reward difference between the two was 2.37 (20.5%). When testing the trained agent, Mish had the highest average reward and SELU the lowest, with a difference of 1.42 (10.3%). In future work, we will evaluate the performance of deep RL agents on a number of other factors that are required to learn highway merging. This will allow us to see what factors are effective for different merging situations and different learning algorithms. We will also analyze the data from the velocity terms and collision terms that make up the reward function to see how they affect the reward results.

References

[1] D. Lee, "Comparison of Reinforcement Learning Activation Functions to Maximize Rewards in Autonomous Highway Driving," *The Journal of the Institute of Internet, Broadcasting and Communication (JIIBC)*, Vol. 22, No. 5, pp. 63-68, 2022.
DOI: https://doi.org/10.7236/JIIBC.2022.22.5.63

[2] A. Alkhouly, A. Mohammed, and H. Hefny, "Improving the Performance of Deep Neural Networks Using Two Proposed Activation Functions," *IEEE Access*, Vol. 9, 2021.
DOI: https://doi.org/10.1109/ACCESS.2021.3085855

[3] Z. Hu, J. Zhang, and Y. Ge, "Handling Vanishing Gradient Problem Using Artificial Derivative," *IEEE Access*, Vol. 9, 2021.
DOI: https://doi.org/10.1109/ACCESS.2021.3054915

[4] S. Douglas and J. Yu, "Why RELU Units Sometimes Die: Analysis of Single-Unit Error Backpropagation in Neural Networks," *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, 2018.
DOI: https://doi.org/10.1109/ACSSC.2018.8645556

[5] H. Zheng, Z. Yang, W. Liu, J. Liang and Y. Li, "Improving deep neural networks using softplus units," *2015 International Joint Conference on Neural Networks (IJCNN)*, 2015.
DOI: https://doi.org/10.1109/IJCNN.2015.7280459

[6] J. Xu, Z. Li, B. Du, M. Zhang and J. Liu, "Reluplex made more practical: Leaky ReLU," *2020 IEEE Symposium on Computers and Communications (ISCC)*, 2020.
DOI: https://doi.org/10.1109/ISCC50000.2020.9219587

[7] Y. Tsai, Y. Jheng and R. Tsaih, "The Cramming, Softening and Integrating Learning Algorithm with Parametric ReLU Activation Function for Binary Input/Output Problems," *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019.
DOI: https://doi.org/10.1109/IJCNN.2019.8852023

[8] J. Kang et al., "An Improved 3D Human Pose Estimation Model Based on Temporal Convolution with Gaussian Error Linear Units," 2022 8th International Conference on Virtual Reality (ICVR), 2022.
DOI: https://doi.org/10.1109/ICVR55215.2022.9848068

[9] Z. Huang, T. Ng, L. Liu, H. Mason, X. Zhuang and D. Liu, "SNDCNN: Self-Normalizing Deep CNNs with Scaled Exponential Linear Units for Speech Recognition," *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.
DOI: https://doi.org/10.1109/ICASSP40776.2020.9053973

[10] J. Barron, "Continuously Differentiable Exponential Linear Units," *arXiv preprint arXiv:1704.07483*, 2017.
DOI: https://doi.org/10.48550/arXiv.1704.07483

[11] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, Vol. 107, 2018.
DOI: https://doi.org/10.1016/j.neunet.2017.12.012

[12] D. Misra, "Mish: A self regularized non-monotonic activation function," *arXiv preprint arXiv:1908.08681*, 2019.
DOI: https://doi.org/10.48550/arXiv.1908.08681

[13] E. Leurent, "An Environment for Autonomous Driving Decision-Making," *GitHub repository: https://github.com/eleurent/highway-env*, 2018.

[14] J. Schulman, et al., "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.
DOI: https://doi.org/10.48550/arXiv.1707.06347

[15] C. Garbin, Z. Xingquan, and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," *Multimedia Tools and Applications*, Vol. 79, 2020.
DOI: https://doi.org/10.1007/s11042-019-08453-9